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

Bachelor Thesis

Clustering-based Audio Segmentation with Applications to Music Structure Analysis

submitted by

Radu Curticapean

submitted

23.11.2009

Supervisor

PD Dr. Meinard Müller

Advisor

PD Dr. Meinard Müller

Reviewers

PD Dr. Meinard Müller
Jun.-Prof. Dr. Matthias Hein

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 in Lieu of an Oath

I hereby confirm 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)

Danksagung

Ein erster Dank gilt meinem Betreuer Meinard Müller, dessen Hilfe bei der vorliegenden Arbeit schon mit der Auswahl des Themas begann. Ohne seine wertvollen Hinweise und Ratschläge, die ungezwungene Atmosphäre sowie das offene Ohr, auf das ich mich auch in kurzfristigen Situationen verlassen konnte, wäre diese Arbeit vermutlich nicht möglich gewesen.

Ebenso danke ich meinen Eltern, Dan und Dana Curticapean, ohne die schon aus rein logischen Gründen die vorliegende Arbeit nicht vorliegen könnte. Durch ihre Unterstützung insbesondere in den letzten Monaten haben sie einen großen Teil zu der vorliegenden Arbeit beigetragen. Ebenso danke ich Volkmar Heinze für die indirekte Unterstützung, die ich durch ihn erhalten habe.

Für die Bereitschaft, die Zweitkorrektur vorzunehmen, seine ergänzenden Hinweise zum Thema, dem sich die vorliegende Arbeit widmet, sowie seine Vorlesungen, die mich in eines der für mich interessantesten Gebiete der Informatik eingeführt haben, danke ich zudem Matthias Hein.

Dank gilt auch meinen Freunden, denen ich mich in der Schlussphase der Arbeit weniger widmen konnte als mir lieb war. Für das dafür aufgebrachte Verständnis sowie das Interesse an meiner Arbeit sowie meinem bzw. unserem Studienfach danke ich ihnen.

Zu dieser Arbeit haben zudem Pengming Wang, Kamil Faber, Anton Krohmer sowie Robert Künnemann beigetragen, indem sie sich kurzfristig dazu bereit erklärt haben, die Arbeit korrekturzulesen. Neben der unerbittlichen Tilgung von Tippfehlern haben sie auch geholfen, einige Passagen von möglichen Missverständnissen zu bereinigen. Damit haben sie sich einen großen Dank verdient. Weiterhin danke ich ihnen, Philipp von Styp-Rekowsky und Karl Bringmann für die interessanten Unterhaltungen über das Thema.

Zudem danke ich Peter Grosche und Verena Konz für die durch Klebeband-Rollen offen gehaltene Büro-Tür, durch die ich regelmäßig mit mehr oder weniger ausgereiften Ideen treten durfte und mir immer zumindest ein gemurmeltes Feedback erhoffen konnte.

Schließlich möchte ich der Studienstiftung des deutschen Volkes danken, einer Institution, die mich während meines bisherigen Studiums insbesondere in Hinblick auf die Breite meines Horizonts förderte und dadurch verhindern konnte, dass meine Faszination für die Informatik das Interesse an anderen Fächern verdrängt. Auch an dieser Stelle muss ich zudem Meinard Müller für den interdisziplinären Aspekt dieser Arbeit danken. Wenn Informatik-Studenten mit Nebenfach Musikwissenschaft eine Minderheit darstellen, dann stellen derartige Studenten, denen es zudem ermöglicht wird, eine thematisch zwischen beiden Fächern angesiedelte Abschlussarbeit zu verfassen, eine besonders glückliche Minderheit dar.

Contents

A round egg can be made square according to how you cut it.
Japanese proverb

1 Introduction

1.1 Problem Setting

Music segmentation in its full generality is the task of dividing musical audio data into sections, i.e. finding a mapping from time indices to some set of labels that satisfies a pre-imposed and application-dependent criterion. One might, for instance, think of a decomposition of a song into natural parts such as chorus, verse, bridge, etc. However, music segmentation could also mean dividing a song into segments that correspond to single notes or chords. Finally, a segmentation could also be based on the instrumentation of a song, i.e. discriminate between parts that feature different instruments.

In most cases, the segmentation we called ‘natural’ would be the ideal result of a music segmentation method since this kind of segmentation would correspond to the perceived structure of music. While for popular music the segmentation into chorus, verse, etc. is evident in most cases, structure gets more complicated in classical music, where even musicologists can debate over the ‘right’ segmentation of a piece.

To avoid subjectivity, we will focus on pieces with unambiguous structure, i.e. mostly popular music. As an example, the rock song *Rock and Roll Queen* by *The Subways*, fulfilling the criterion of structural disambiguity, is segmented as follows:

Start time (s)	End time (s)	Segment label
0	19.7	intro
19.7	47	chorus
47	67	verse
67	95	chorus
95	115	verse
115	129	interlude
129	152.8	chorus
152.8	168.1	ending

Table 1.1: Example natural segmentation of *Rock and Roll Queen*

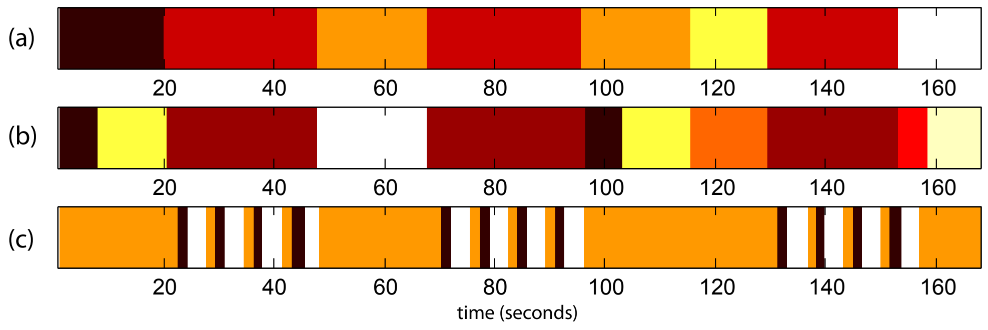


Figure 1.1: Plots of human segmentations for *Rock and Roll Queen*, where regions of same color share the same segment label. **(a)** natural segmentation **(b)** segmentation based on perceived timbre **(c)** harmonical segmentation based on played chord.

Figure 1.1 shows different human segmentations of a song. As with most pictures in this thesis, time progresses in seconds along the x-axis. Furthermore, equi-colored regions correspond to time intervals that share the same segment label. In the first segmentation, labels correspond to chorus, verse, etc., while the second segmentation takes into account only the instrumentation of the song. Finally, the third segmentation has been obtained by considering the sequence of chords played in the song.

Observing that most songs in popular music exhibit a quite clear partition into chorus, verses, etc. that is reflected in timbral, rhythmical and harmonical qualities of the constituents of this partition, we try to use these features as a foundation on which to build segmentation methods. In a certain way, these properties provide a way of approximating the previously mentioned and not properly defined 'natural segmentations', making a computational approach possible in the first place. However, by choosing this approximation, we make the following assumption to musical structure:

Assumption 1.1 (Homogeneity Assumption)**.** *Every segment is characterized by intrinsic timbral, rhythmical and/or harmonical properties that are given within the segment, but don't occur in other segments.*

Of course, methods building on this model assumption are likely to fail on pieces that don't fulfill it. For instance, one might think of classical piano music, where timbral properties will change insignificantly over a piece, rhythm plays only a minor role and the harmony within a theme can change arbitrarily. The more complex segment structure of this type of music is based on repetitions rather than intrinsical properties, yielding the following alternative model assumption:

Assumption 1.2 (Repetition Assumption)**.** *Segments are characterized by specific sequences of feature characteristics that repeat within a piece. Especially, a completely inhomogeneous sequence of time frames can be called a segment whenever this sequence is repeated at least once in the piece.*

1.2 Mathematical Formulation

Definition 1.2.1. *An* ***audio file*** A *in general is a waveform representation of audio data with a certain* ***duration*** T*. Neglecting temporal discretization for now, we reduce an audio file to its* ***temporal domain*** $A = [0, T) \subset \mathbb{R}$*. Imagine this interval as a timeline of the audio file, as can be seen in Figure 1.2, where* $A = [0, 16)$*.*

Definition 1.2.2. *For any audio file, we fix a finite set of* ***labels*** $\mathcal{Y}$ *with* $y \in \mathcal{Y}$ *being some string. In our example, we have* $\mathcal{Y} = \{A, B\}$*. Since* $\mathcal{Y}$ *is finite, we can always find a bijection* $\mathcal{Y} \to \{1, \dots, K\}$ *with* $K \in \mathbb{N}$ *being the* ***label count****.*

Definition 1.2.3. *A* ***segment*** S *within an audio file* A *is a pair consisting of a* ***spanned interval*** $I(S) = [s, t) \subseteq A$ *and some* ***segment label*** $L(S) \in \mathcal{Y}$*. In Figure 1.2 the four segments* S_1, S_2, S_3*, and* S_4 *are present, e.g.* S_1 *with* $I(S_1) = [0, 4)$ *and* $L(S_1) = A$*.*

Definition 1.2.4. *A* ***segmentation*** $\mathcal{S}$ *of an audio file* A *is a set of segments* $\{S_1, \dots S_l\}$ *whose spanned intervals partition* A*, i.e. they are pairwise disjoint and furthermore* $I(S_1) \cup \ldots \cup I(S_l) = A$*. For a given segmentation* $\mathcal{S}$*, the set of* ***segment boundaries*** $B(\mathcal{S})$ *is defined as the set of points in time that have two adjacent segments* S_i *and* S_j*:*

$$B(\mathcal{S}) = \{x \in A \mid \exists\ S_i, S_j : I(S_i) = [s, x) \ \wedge\ I(S_j) = [x, t)\}$$

In our example, $B(\mathcal{S}) = \{4, 8, 11\}$. Now, we introduce the segmentation function as an alternative way of describing a segmentation. Note that this definition is equivalent to the previous one if no subsequent segments in a segmentation share a common label.

Definition 1.2.5. *The* ***segmentation function*** $\mathcal{S}' : A \to \mathcal{Y}$ *is a total function mapping from* A *to labels. Segments are connected sets in which* $\mathcal{S}'$ *stays constant, i.e. for every segment* $S \in \mathcal{S}$ *it holds that* $\forall x \in I(S) : \mathcal{S}'(x) = L(S)$*. When identifying* A, B *with* 1*, respectively* 2*, we get the segmentation function plotted in Figure 1.2.*

Definition 1.2.6. *For any segmentation* $\mathcal{S}$ *and each label* $y \in \mathcal{Y}$*, the* ***cluster*** C_y *is defined as the union of the spanned intervals of all segments labeled as* y*. Formally:*

$$C_y = \bigcup_{S \in \mathcal{S}\ :\ L(S) = y} I(S) = \{x \mid \mathcal{S}'(x) = y\}$$

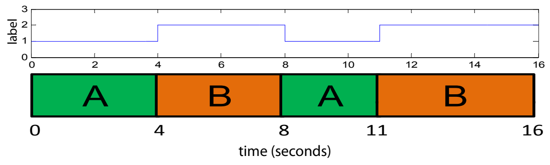


Figure 1.2: **(bottom)** example audio file of 16 seconds length with segmentation having $\mathcal{Y} = \{A, B\}$ **(top)** corresponding segmentation function $\mathcal{S}'$ with labels on y-axis.

1.3 Current and Possible Applications

In general, any application that relies on structural information about music benefits from music segmentation at least in that it can be used as a pre-processing step for subsequent algorithms. As it turns out, a vast number of such applications exist, among which we present only a selection in this section.

Of course, automated segmentations are unlikely to replace their human-annotated counterparts, but in applications involving several thousands of audio files, the personal effort needed to create a human segmentation for each file is too high, thus creating a need for automatic music segmentation.

Maybe the most evident application of music segmentation is **intra-piece navigation**: Opposed to commonly known media players, where navigation within a piece is possible only by selecting a time position to play back from, this navigation method allows for more intuitive navigation features, such as a *'next chorus'* button or a visual representation of the song structure as seen in Figure 1.1 and Figure 1.3. While this could significantly improve usability of live DJ applications, such as the system described in [2], musicologists could also benefit from intra-piece navigation since especially classical recordings often feature very long tracks that lack cue points.

An existing prototype for a music player supporting intra-piece navigation is a system called *SmartMusicKiosk* developed by Goto and described in [10]. Furthermore, we implemented *MultiStructure*, a simple plugin within the SyncPlayer framework, described in [5], allowing for intra-piece navigation according to multiple segmentations. A screenshot of this plugin can be seen in the right part of Figure 1.3, showing it during playback of a simple audio file that has been segmented by a human in the first row and by an algorithm in the second row. Clicking on the colored rectangles allows for jumping to the corresponding segment during playback.

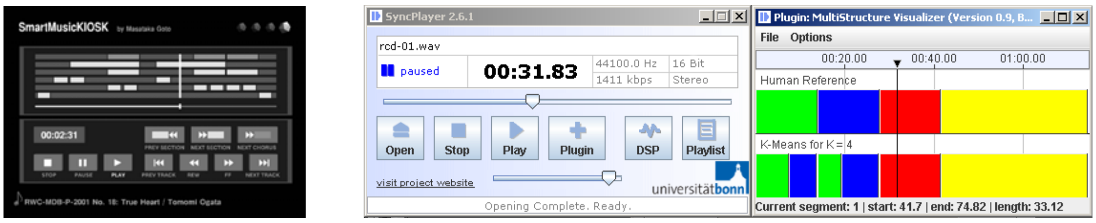


Figure 1.3: Intra-piece navigation systems: **(left)** SmartMusicKiosk implemented on a tablet PC, reproduced from [10], **(right)** SyncPlayer with MultiStructure plugin showing two segmentations.

Based on segmentation, one can also solve the task of **music summarization**, where one tries to extract a small excerpt from a piece that represents the whole piece well, which for popular music often equals to **chorus detection**. The latter is of great use for online music stores where databases consisting of millions of songs need to be processed automatically to provide preview clips.

While *homogeneity-based* music segmentation is an interesting problem on its own, its practical importance will be rather found in the pre-processing steps of music information retrieval applications, since the output of a successful segmentation will consist of segments that are intrinsically homogeneous. Subsequent information retrieval algorithms can then process each segment under this assertion.

For instance, homogeneity-based music segmentation can be used to improve **comparison** of different songs. Consider an example song A consisting of several segments $S_1, S_2, \dots S_n$ in a certain order and another song B that consists of exactly the same segments, but in a different order. Human listeners would immediately recognize B as a remix of A and therefore assign high similarity to this pair of songs. However, when considering DTW-based approaches for comparison, these songs would be judged very dissimilar as there is no possibility of aligning the songs with a high match. When decomposing both songs according to homogeneous parts and comparing these parts, the correspondence between the individual parts can be recognized.

Building upon this segment-level dissimilarity of songs, music segmentation could find a possible application in **DJ assistance tools**. If the similarity values of segments of a large set of songs are known, one could automatically compute a sequence of songs such that each song ending matches the intro of the subsequent song, yielding a seamless mix of songs in a playlist, as visualized in Figure 1.4. In this idealized playlist consisting of three songs, transitions, as depicted by red arrows, occur only between segments that have high similarity, indicated by equal colors.

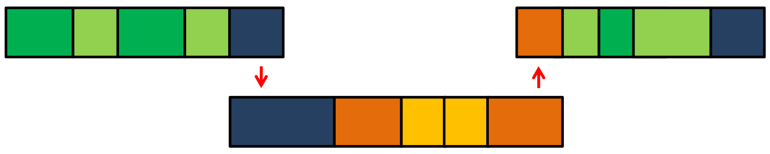


Figure 1.4: Seamless mixing in DJ applications: transitions (red arrows) between songs occur only between similar intro and ending parts (same color).

When considering harmonical qualities of the audio file, homogeneity-based music segmentation corresponds to **chord detection**, i.e. the homogeneous clusters in such a segmentation will correspond to parts of the music that have roughly the same harmonic content. By comparing this content to a previously determined set of chord templates, the clusters can even be labelled according to the prevailing chord.

Furthermore, homogeneity-based music segmentation can be used as a **preprocessing step for repetition-based segmentation** in order to reduce an audio file to a sequence of segment labels. Considering again harmonical properties, a song could for instance be reduced to a sequence of chords, reducing the amount of data needed to be stored in a database intended for music information retrieval and speeding up retrieval tasks based on brute-force operations on self-similarity matrices.

1.4 Related Work

Being a very fundamental problem behind many questions in music information retrieval, music segmentation has received early attention by researchers in this field. Generally, solution strategies can be divided into repetition-based and homogeneity-based methods according to the model assumption chosen. In the literature, these methods are often equivalently called sequence-based, respectively state-based methods.

Furthermore, some authors discriminate between music segmentation and music structure analysis by defining music segmentation as the task of finding only boundaries between segments, whereas music structure analysis is defined according to our definition of music segmentation in Section 1.1.

A precursor of music segmentation, as presented in [4] by Cooper and Foote, focussed on automatic music summarization as described in the previous subsection. Here, a fixed-length subinterval of a song with maximal average similarity to the rest of the song is extracted out of the self-similarity matrix S of the song, which was introduced by the same authors in [7]. The average similarity score for a segment is computed as the mean value of the submatrix $S(I,:)$. In Figure 1.5, we see a self-similarity matrix to the left and three average similarity scores $s_l(x)$ for intervals of different lengths $l = 10, 20, 30$ starting at x. The maximum of each curve represents the optimal excerpt start. Since this approach can only identify segments of fixed length that repeat sufficiently often within a song, we can classify it a repetition-based approach.

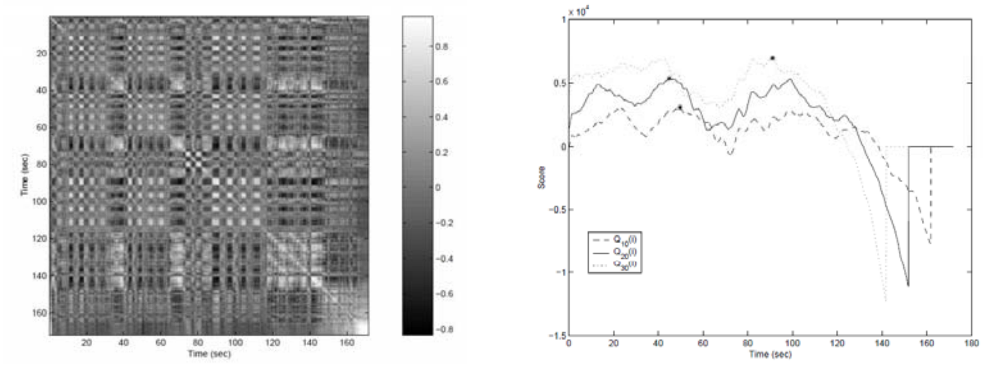


Figure 1.5: Music summarization, as reproduced from [4]: **(left)** Self-similarity matrix for MFCCs on *The Magical Mystery Tour*, **(right)** average similarity of intervals of length 10 (dashed line), 20 (solid line) and 30 (dotted line) seconds to the rest of this song.

In their paper [8], the same authors introduced the novelty-based approach to music segmentation we will encounter in Chapter 5.2. Here, the homogeneity assumption is used to produce an intermediate segmentation which is then in turn segmented by the repetition assumption. In this method, homogeneity-based segmentation is used as a preprocessing step for repetition-based music structure analysis.

In [9], that appeared in the same year, Goodwin and Laroche use linear discriminant analysis (LDA) to project features into an a priori learned feature subspace. For such a projected feature sequence, they compute boundaries by means of dynamic programming under the homogeneity assumption. As they note, the LDA feature mapping can be considered a way of introducing a priori information into the dissimilarity measure between features. However, as they don't provide evaluation results, it can't be concluded that learning a feature subspace provides an improvement to music segmentation.

Building upon previous papers, Levy and Sandler introduce an approach to music segmentation in [13] that builds on Hidden Markov Models. In their paper, they assume that timbre in musical audio files is generated by a Gaussian mixture model, i.e. every segment has a characteristic set of states, each generating a Gaussian distribution of feature vectors. A musical audio file can then be regarded as having an underlying sequence of states that generates the feature sequence we can observe. Assuming the Markov property of the state transitions, meaning that the probability of every state transition depends only on the current state and on no previous history, this feature generation model can be viewed as a HMM. By training a HMM with a fixed number of states to the feature sequence and decoding this sequence with the trained HMM, the most probable sequence of timbre states can be computed, as we did in Figure 1.6 for the first 95 seconds of the song *Rock and Roll Queen*.

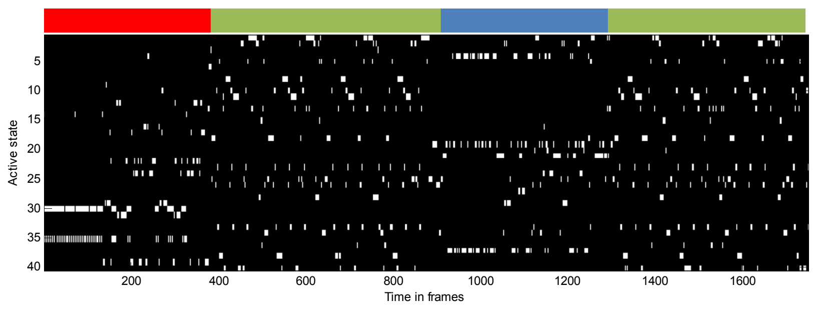


Figure 1.6: **(top)** Human segmentation for the first 95 seconds of *Rock and Roll Queen*, **(bottom)** sequence of HMM states decoded from a 40-state HMM trained on timbre features.

However, this sequence doesn't yet represent the final segmentation. Instead, the local distribution of states for every point in the feature sequence is computed by a local histogram. To obtain this, the sequence of states is smoothed with a window of fixed length and the resulting sequence of distributions is clustered by an algorithm that resembles the EM-algorithm, which can be again thought of as a generalization of the K-means algorithm. In general, the approach can be therefore seen as a homogeneity-based method.

1.5 Organization of Thesis

Inspired by our modularized implementation of the methods presented in this thesis, we follow the path of an input audio file through a music segmentation algorithm, i.e. we start at the tail of the leftmost arrow in Figure 1.7, proceeding through the segmentation pipeline, until finally ending up as a segmentation.

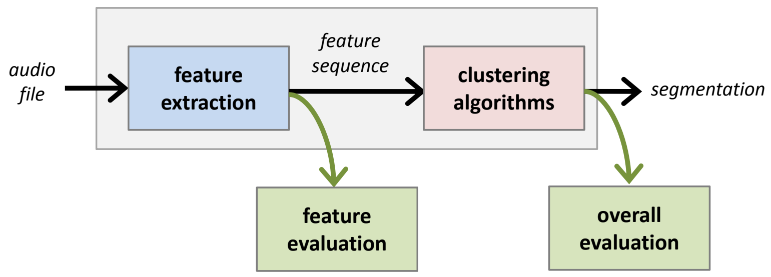


Figure 1.7: Schematic illustration of modularized music segmentation as considered in this thesis

The different stages we pass along this way, depicted as colored boxes, correspond to chapters and the ordering of chapters corresponds to the ordering of the boxes in x-direction. The part concerning clustering algorithms receives special attention and therefore two chapters, as it constitutes a main part of this thesis. The following list gives a rough overview of the contents of each chapter:

- In Chapter 2, we explore the problem of **feature extraction**, i.e. we are given an audio signal and try to find a representation of this signal that allows for music segmentation by modelling musical properties that could be subject to the homogeneity assumption.
- In Chapter 3, we discuss methods for **feature evaluation**. These methods will help us in the process of deciding which features to choose for subsequent use.
- In Chapter 4, **time-ignoring clustering algorithms** are introduced. These algorithms try to find clusters in the feature data but neglect temporal information contained in the feature sequence.
- Building upon the algorithms described in the previous chapter, Chapter 5 deals with **time-dependent clustering**, where temporal information is used to improve music segmentation.
- In Chapter 6, we present detailed **evaluation results** for a selected set of audio files and interpret the results of different algorithms on this set while also presenting musical interpretations for the results.
- Finally, Chapter 7 contains the **conclusions** and hints to **future work**.

1.6 Contribution

The main contribution of this thesis lies in the modular structure used to tackle the problem of music segmentation. This modularization enables for comparison of different algorithms on different feature types and especially demonstrates the performance influence of refinements on feature and/or algorithm design.

This modular framework for music segmentation was implemented in the prototyping language MATLAB, yielding a possibility of processing large sets of input songs automatically and therefore enabling us to perform evaluation on typical musical audio data. Furthermore, many ideas for segmentation algorithms and improvements to already given algorithms can be implemented quickly in this framework.

Furthermore, we introduce the notion of feature evaluation in Chapter 3, which in some sense puts the cart before the horse, since instead of measuring the quality of a segmentation result when using an algorithm that is given a feature sequence, we rather try to measure the extent of feature separation within a feature sequence when segmented according to a reference segmentation. This approach yields two algorithm-independent evaluation scores each having their benefits and drawbacks.

By using agglomerative clustering as a segmentation algorithm, we explore the possibilities of hierarchical segmentations in homogeneity-based clustering. Agglomerative clustering algorithms yield dendrograms, i.e. tree-like graphical representations of the hierarchical structure that can be found in an audio file. However, this clustering method depends heavily on the cluster dissimilarity function chosen and further work is needed to construct more meaningful dissimilarity functions, especially with respect to musical interpretations of cluster dissimilarity.

Building upon standard agglomerative clustering, we introduce time-dependency in agglomerative clustering by restricting the possible choices for merging different clusters and by manipulating the input to agglomerative clustering. The resulting clustering algorithm is then sped up by means of dynamic programming, yielding a simple yet efficient segmentation algorithm.

Furthermore, we introduce a new method for computing novelty functions and show that it has a sound statistical interpretation. Though theoretically not equivalent, we show that the results obtained by this method are comparable to another novelty detection method proposed in [8].

Last but not least, we present detailed evaluation results for a selected set of musical audio files, yielding indicators for the practical utility of our methods and musical interpretations for the results obtained with different algorithms on different features.

Colors, like features, follow the changes of the emotions.
Pablo Picasso

2 Feature Extraction

2.1 Introduction

2.1.1 Musical Motivation

The process of transforming an audio file into a more meaningful representation that facilitates subsequent processing by measuring musical properties is referred to as *feature extraction*. Within the music segmentation framework, we consider three feature types to incorporate musical information that allows for homogeneity-based segmentation: timbre, tempo and chroma features.

Recalling the homogeneity assumption defined in Section 1.1, we consider an audio file to fulfill this assumption with respect to a musical property, such as timbre, if both of the following requirements hold:

- The characteristics of this property, that should vary over an audio file, allow for discriminating different true segments
- The characteristics of the property have low variance within a true segment

The **timbre** of a sound, also called *tone color*, is a musical variable denoting the tonal qualities a sound features in its frequency spectrum apart from its pitch, i.e. the loudness distribution of harmonics. Accordingly, two tones of different pitch can still share the same timbre. However, two tones played by different instruments, e.g. by a piano and an electric guitar, don't share the same timbre. In the context of music segmentation, we will give only rough approximations to timbre and can therefore reduce the timbre of a sound to the instrumentation of this sound. The musical motivation for using timbre features is also given by the fact that musical structure often correlates with instrumentation in popular songs, as our detailed evaluation in Chapter 6 will show along some examples from different genres. Therefore, in popular music different instrumentations will likely correspond to different segment types and furthermore the instrumentation within segments is likely to stay constant.

As the musical definition of rhythm is very complex and incorporates different temporal scales featuring different types of rhythms, we consider only tempo features. The **tempo** of a sound is the frequency a beat or some regular rhythmical pulse occurs in this sound. For popular music, we don't expect tempo features to provide useful information as the tempo within such songs often stays constant over different segment types. For classical music, we expect tempo features to yield better results than timbre features, as musical form in classical music often correlates with tempo indications.

The last musical property we consider, the **chroma** of a sound, finally abstracts from timbre and tempo in order to consider a sound to be a set of pitches. In this definition, pitches are considered only modulo octave equivalence, meaning that $C1$ and $C2$ are equivalent with respect to chroma. In general, we don't expect chroma features to provide useful information for homogeneity-based clustering, except for very simple songs featuring a correlation between harmonies and segment labels. However, this correlation would imply the existence of large parts without any harmonical changes, which is very unlikely even in popular music.

2.1.2 Remarks on Feature Sequences

From the computational perspective, feature extraction methods compute **feature sequences** $X = (x_1, \ldots, x_N)$ from audio files, where each **feature vector** x_i is a row vector in a **feature space** $\mathcal{X}$, which in our case can be interpreted as $\mathbb{R}^d$ for some dimension d. Considered as a matrix, a feature sequence is an $l \times d$ matrix with time progressing within columns and feature vector dimensions progressing within rows. We also allow to consider a feature sequence as a set of feature vectors in our notation.

Note that X is a finite sequence of N vectors, whereas the temporal domain of an associated audio file $A = [0, T)$ is a time interval. This is explained by the fact that feature extraction also consists of *temporal discretization*, i.e. the domain is divided into N intervals for which a feature point is computed each.

Every feature vector $x_i \in X$ carries information extracted from a small interval $W_i \subset A$ with a certain window length w that is assumed to be constant. In general, the ensemble of all intervals W_i is *no* partition of A, since we allow subsequent subintervals to share a common interval, the **overlap interval** $O_i = W_i \cap W_{i+1} \neq \emptyset$ of length o, again assumed to be constant for all i. Building upon these definitions that apply to all feature extraction methods we are going to introduce in this chapter, we also define the **feature rate** $f = \frac{N}{T}$ of a feature sequence X of length N for an audio file $A = [0, T)$ as the ratio of the number of features in X to the length of the audio file. Furthermore, we define the **overlap ratio** as the ratio $\frac{o}{w}$ of the overlap length to the window length.

Since this will facilitate notation later on, we also introduce the notion of a discrete audio file. In this definition we set the domain of an audio file to a finite set of N indices in order to clarify the correspondence between feature sequences and audio files.

Definition 2.1.1. *A* ***discrete audio file*** A' *is defined as* $A' = \{1, \ldots, N\} \subset \mathbb{N}$. *This definition is similar to Definition 1.2.1, where audio files were introduced that featured intervals as their temporal domains. To facilitate notation, we will denote* A' *by* $[1 : N]$.

Given the feature rate f, we can map time indices from a feature sequence X of length N and time indices from a discrete audio file $A' = [1 : N]$, to these of a continuous counterpart A by considering the injective mapping $dc : [1 : N] \to [0, T)$ defined as

$$dc(i) = \frac{i-1}{f} = (i-1)\frac{T}{N}$$

The converse mapping $cd : [0, T) \to [1 : N]$ is defined as

$$cd(t) = \lfloor f \cdot t \rfloor + 1 = \left\lfloor \frac{N}{T} t \right\rfloor + 1$$

Note that due to the existence of these mappings, discrete audio files and continuous audio files are almost equivalent. In subsequent sections, we will therefore use the discrete formulation whenever we consider time progressing in time indices, which we also call **frames**. The continuous formulation will be used whenever time is considered to progress in seconds, e.g. in plots or in human reference segmentations.

All other definitions in Section 1.2, i.e. the notions of segments, segmentations, segmentation functions, and clusters, carry over to this discrete setting when considering the following two changes:

- The **spanned interval** of a segment S is a discrete grid of values $I(S) = [s : t]$.
- The set of **segment boundaries** of a segmentation $\mathcal{S}$ is the following set: $B(\mathcal{S}) = \{x \in A \mid \exists\ S_i, S_j : I(S_i) = [s : x-1] \ \wedge\ I(S_j) = [x : t]\}$

2.2 Naive Spectrum Features

First of all, we introduce *naive spectrum features*. These simple spectrum-based features serve as a baseline against which we will compare more advanced features. Though rudimentarily, they already allow for measuring timbral properties within a time window and illustrate the basic ideas common to the more interesting features. In the MPEG7 standard, a similar feature is known as *AudioSpectrumEnvelope*, described in [3], where also an enhancement of this feature, the *AudioSpectrumProjection* is presented. We will return to this feature in Section 2.3.

In order to compute naive spectrum features from an audio signal, one first needs to compute a spectrogram of the signal, i.e. a sequence of Fourier spectral vectors indicating the evolution of the frequency distribution over an audio file. This can be done easily with a discrete short-time Fourier transform (STFT) that computes the spectrum within small windows which we choose to be Hann windows of length 50 ms with an overlap value of $\frac{1}{4}$. We discard the phase information provided by the STFT. For further information on spectrograms and Fourier transforms, we point to [14].

Motivated by the observation that human perception of pitch follows a logarithmic law, the spectrogram is now processed by an exponential re-binning of the frequency axis, transforming the previously linear axis into a log-frequency axis. The central frequency c_i and the bandwidth w_i of each bin are computed as follows:

$$c_i = 80 \cdot 2^{\frac{i-1}{12}} \text{ Hz} \qquad\qquad w_i = c_i \cdot 2^{\frac{1}{12}}$$

For signals with a sample rate of 22 KHz, this yields a log-spectrogram with 85 log-frequency coefficients, starting at 80 Hz and ranging up to the Nyquist frequency of 11 KHz. The y-axis of this log-spectrogram roughly approximates *perceived* pitch.

In a second step, the amplitude values in the log-spectrogram are also mapped to a logarithmic scale in order to approximate the equally logarithmic perception of loudness. We choose the following mapping, where $x_{\max}$ denotes the maximum of all log-spectrogram values over the whole signal:

$$x \mapsto \log\left(1 + 10 \cdot \frac{x}{x_{\max}}\right)$$

Figure 2.1 shows the intermediary results of the extraction process for naive spectrum features on the song *Rock and Roll Queen*. The spectrogram obtained after the rescaling described in the previous paragraph can be seen in the first subplot.

Now, the log-spectrogram is convolved with a Hann window of size s along the frequency axis. This convolution smoothes the features along this axis and condenses the energy of frequencies in a frequency bin of width s to one single weighted average. We choose

$$s = 2\left\lfloor \frac{85}{12} \right\rfloor = 14$$

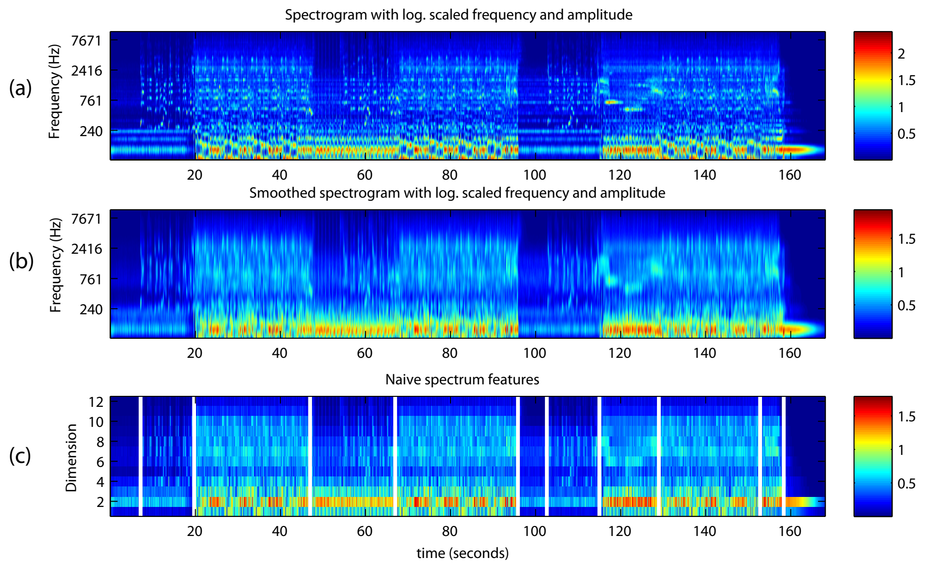


Figure 2.1: Feature extraction steps for *Rock and Roll Queen*: **(a)** spectrogram with log-scaled frequency axis and amplitude, **(b)** spectrogram smoothed with Hann window along y-axis, **(c)** 12-dimensional feature sequence after binning

in order to smooth the log-spectrogram in such a way that each y-unit is influenced by approximately $\frac{1}{12}$ of all log-frequency coefficients in the spectrogram.

From the resulting smoothed log-spectrogram, as plotted in the second subplot of Figure 2.1, we now choose 12 rows corresponding to frequency bins in such a way that the first and last bin have center frequencies of 80 Hz, respectively 6834 Hz, and the difference between center log-frequencies of any two consecutive rows is constant. Then, we discard the remaining rows to obtain a condensed log-spectrogram for 12 frequency bins that have equal distance on the *log*-frequency scale, which implies that the ratio between the centers of two subsequent frequency bins is constant. The resulting condensed spectrogram can now be considered as a sequence of 12-dimensional feature vectors, each describing a roughly binned spectrum of the signal in a window of about 50 ms length. The concrete number of rows (12) is chosen in order to facilitate comparison of naive spectrum features to other features.

The result of this feature extraction is plotted in the third subplot of Figure 2.1 along with white segment boundaries taken from the human annotation. As we notice, the features yield a clear distinction between chorus, intro and verse segments. In total, naive spectrum features condense the whole spectrum into 12 bins in order to give a rough estimate of the evolution of timbral qualities over time.

2.3 MFCC Features

In this section, we will present the *Mel Frequency Cepstrum* and the resulting feature, the *Mel Frequency Cepstrum Coefficients (MFCCs)*, a more sophisticated way of measuring musical timbre that was developed in the context of speech processing.

To obtain MFCC features, an STFT of the audio signal, as described in the previous section, is computed first, with amplitudes again being rescaled logarithmically. Then, components of the resulting spectral vectors are binned according to the mel scale, a psychoacoustic model for perceived pitch. This is achieved by using a set of 40 frequency bins whose frequency responses are plotted in Figure 2.2. The result of this spectral binning is a sequence of 40-dimensional spectral vectors.

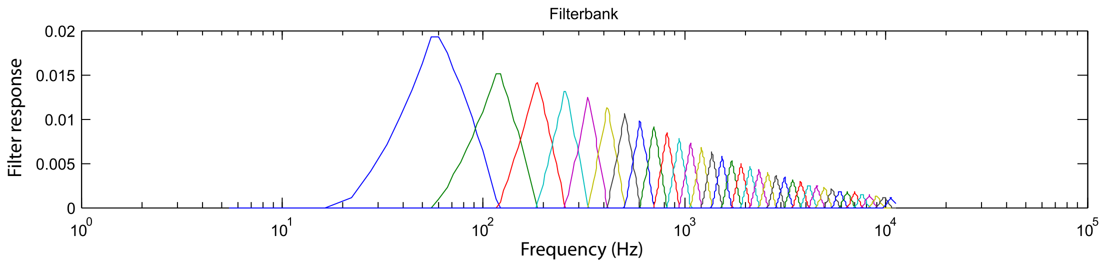


Figure 2.2: Filter responses for each mel frequency band, plotted s.t. each color corresponds to one band. Note that the x-axis is scaled logarithmically. Figure produced with [15].

The basic idea underlying MFCC features is that, after this preprocessing step, the resulting spectral vectors can be represented in another basis. By projecting the spectral vectors into a lower-dimensional basis that has a useful interpretation in terms of signal processing, we hope to extract timbral properties of the music.

This idea is also common to the *AudioSpectrumProjection* features described in [3]. Here, a feature sequence consisting of d'-dimensional *AudioSpectrumEnvelope* feature vectors is transformed by principal compenent analysis (PCA): First, a d-dimensional linear subspace of $\mathbb{R}^d$ is computed that minimizes the mean approximation error to the original vectors when projected into this subspace. Then, *AudioSpectrumProjection* feature vectors are computed as projections of the original feature vectors onto this subspace. As in general $d < d'$, the projection step can be considered as a data compression step.

While this interpretation in terms of data reduction is valid for AudioSpectrumProjection features, the basis transformation performed in the computation of MFCC features is motivated by a common model of human speech assuming that speech is the result of a filtering process involving a carrier signal produced in the vocal tract which is convolved with a filter function arising in the mouth. Following common practice, we transfer the speech processing motivation for MFCC features to the music processing setting, however noting that the behaviour of MFCC features in the presence of multiple instruments is unclear.

Furthermore, we note that in the model assumption underlying MFCC features, both filter and carrier signal can vary over time, but they are assumed to be constant within a single STFT window. Convolution with the filter in the time domain corresponds to a multiplication with the Fourier-transformed filter in the frequency domain given by the STFT. Therefore, each spectral vector can be considered as the componentwise multiplication of a spectral vector obtained from the carrier signal and a spectral vector corresponding to the Fourier transformation of the convolution kernel. Furthermore, as we take a component-wise logarithm, the multiplication is transformed to an addition.

Now, we are ready to give the interpretation of the basis change performed in the computation of MFCC features. By the Fourier transform and the subsequent application of the logarithm on the spectral vectors, we have transformed the filter application to a simple vector addition. In order to separate the carrier signal from the filter, we now have to decompose the spectral vectors, each of which can be regarded as a sum of a carrier and a filter component, into their single components, as we are interested only in the filter component.

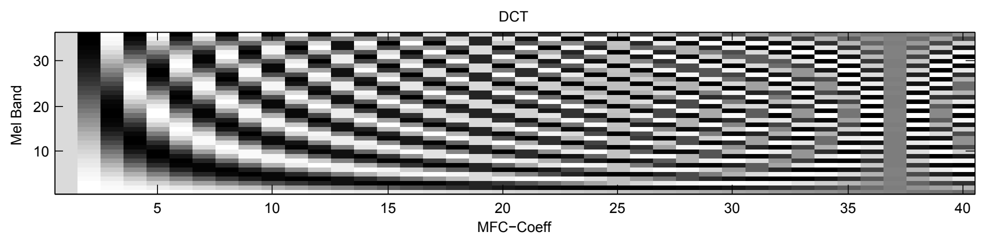


Figure 2.3: 40 DCT basis vectors, plotted as columns arranged along x-axis, s.t. the gray value at (i, j) indicates the weight that is assigned to the i-th mel band by the j-th DCT vector. Figure produced with [15].

At this point, the discrete cosine transform (DCT) is applied in order to provide exactly this kind of decorrelation. The DCT is taken on the spectral vectors and represents every vector as a linear combination of DCT basis vectors, as given by Figure 2.3, where every column corresponds to a DCT basis vector. Note that the first DCT basis vectors vary slowly over the frequency domain, whereas the last vectors feature high-frequent changes. By representing the spectral vectors in this basis, the spectrum is decomposed into 40 cosine basis vectors with increasing frequency. Now, we assume that the carrier signal is found in the higher coefficients, as it features overtones that are not present in the filter signal, and overtones create a comb-like pattern in the spectrum which is found in the higher DCT basis vectors.

Finally, by discarding the upper coefficients of the DCT-transformed spectral vectors, as motivated above, along with the first coefficient corresponding to the average value of features, we obtain 12-dimensional MFCC features. The implementation of the feature extraction process is based on the MA Toolbox, cf. [15]. Feature sequences for MFCCs can be seen in Chapter 6.

2.4 Tempo Features

Apart from timbral qualities, music also has rhythmical properties arising from the fact that music contains impulses, such as note onsets, played in a temporal succession. Furthermore, in most music a steady pace of impulses of different intensities can be found: the beat. For instance, a sequence of equidistant impulses in which every fourth impulse is louder than the other impulses has the famous 4/4 beat. The nontrivial task of determining the correct beat of music is referred to as beat tracking.

While more complex rhythmical structures than the beat exist, we restrict ourselves to the **tempo**, an even simpler concept. We choose this simplification due to robustness concerns that arise when considering complex rhythmical structures. For this, we neglect the rhythmical accentuation induced by different intensities of the impulses and only consider the temporal distance between impulses, yielding a period length T, whose inverse $\frac{1}{T}$ describes the frequency of impulses and is measured in beats per minute (BPM).

In order to compute tempo features, an audio file first has to be scanned for note onsets. Following the explanation in [11], we do this by first computing the STFT, yielding the same starting point as plotted in the first subplot of Figure 2.1 in Section 2.2, i.e. a sequence of spectral column vectors that can also be considered as a set of amplitude row vectors (signals) for every frequency bin.

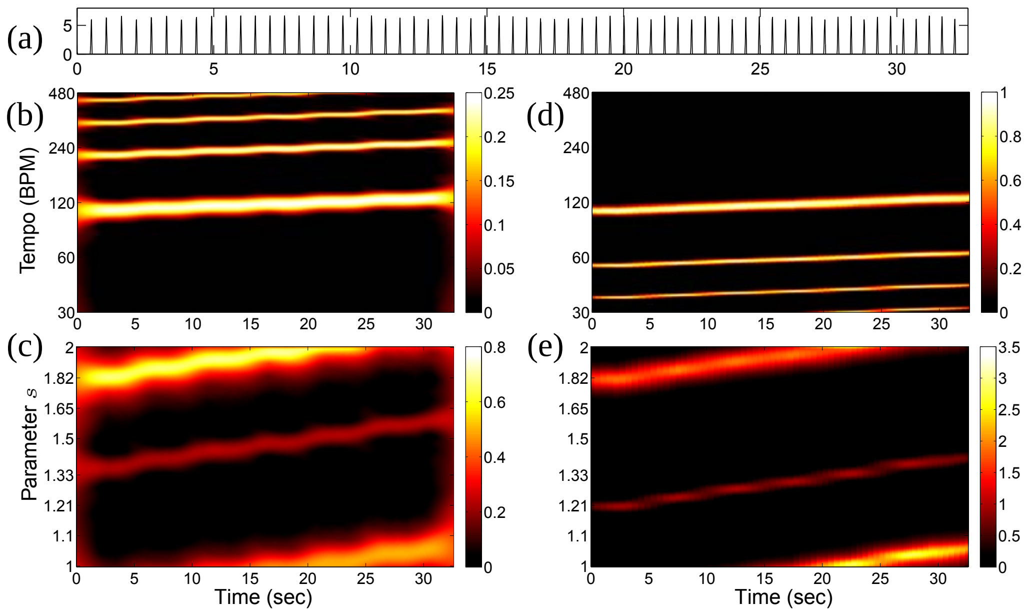


Figure 2.4: Tempo features for a click track with increasing click frequency: **(a)** onset signal for click track, **(b)** Fourier-based tempogram, **(c)** cyclic Fourier-based tempogram, **(d)** autocorrelation-based tempogram, **(e)** cyclic autocorrelation-based tempogram. Reproduced from [11]

Now, for every frequency bin, a novelty measure is introduced that measures the amount of change in the amplitude signal corresponding to this frequency bin. This novelty measure is computed by taking the discrete derivative of the amplitude signal, yielding high values at onsets, i.e. points in time with strong amplitude increases. By summing the positive parts of these novelty measures over all frequencies, the novelty measures are condensed into a single onset signal that is supposed to be high at note onsets. Subplot (a) of Figure 2.4 shows an onset signal for an audio file consisting of clicks whose distance decreases linearly over time. One could theoretically identify peaks in onset signals to get a binary notion of onsets, but following [11], we proceed with two other approaches.

By computing the STFT of the onset signal in windows of a significantly higher length than in the previous application of the STFT, periodicity patterns in the onset signal can be revealed. For the onset signal in the first subplot of Figure 2.4, we obtain the spectrogram plotted in subplot (b). We notice that harmonics of the beat frequency also produce peaks in the STFT. This is a natural effect caused by the fact that, if a sine wave with frequency f has peaks coinciding with the peaks in the onset signal, a sine wave of frequency $k \cdot f$ for $k \in \mathbb{N}$ also has this property. We would like to suppress this effect in order to reduce ambiguity in tempo features.

Before doing so, we first present another method of extracting periodicity information from the onset signal which employs the local autocorrelation of signals obtained by convolving subsignals in windows of about 6 seconds length with temporally shifted copies. More precisely, an onset signal $s(x)$ is convolved with the shifted signal $s(x - t)$, where t indicates a temporal offset, and the signals are restricted to the window mentioned above. Due to this kind of convolution, T-periodic signals will exhibit high autocorrelation for $t \approx T$. By considering $f = \frac{1}{t}$, period lengths can be transformed to beat frequencies. The autocorrelation values for the click track introduced in Figure 2.4 are plotted in subplot (d). A similar tempo confusion as described above arises, as autocorrelation-based features also feature peaks in the subharmonics of f, i.e. the frequencies $\frac{1}{k} \cdot f$ for $k \in \mathbb{N}$, due to the fact that any T-periodic signal is also $k \cdot T$-periodic, which will cause high autocorrelation values at these multiples of T. Again, we wish to reduce this effect.

To this goal, a postprocessing step is proposed in [11] that enables grouping of harmonics corresponding to the same underlying beat frequency. First, the beat frequency axis is rescaled logarithmically, yielding a constant distance of c on this axis between all frequencies f_1, f_2 that have $f_1 = cf_2$. Then, the beat frequencies are binned in a way similar to the binning in chroma features, as described in the next section, by expanding the notion of octave equivalence to beat frequencies: Arbitrary beat frequencies f_1, f_2 are octave equivalent if $f_1 = 2^k f_2$, and every set of frequencies with pairwise octave equivalent constituents, e.g. $\{\ldots, 220, 440, 880, \ldots\}$ is condensed to a single bin by adding up the tempo features along the rows corresponding to frequencies in this set.

The resulting cyclic Fourier-based and autocorrelation-based tempo features are finally plotted in subplots C, respectively E of Figure 2.4.

2.5 Chroma Features

Chroma features measure the harmonic content of a musical audio file, i.e. the signal energy in subbands consisting of frequencies corresponding to musical pitch classes. This yields a sequence of 12-dimensional chroma feature vectors such that every dimension corresponds to a pitch class in the equal-tempered scale. Again, we give only a brief overview and refer to [14] for a more detailed exposition.

Compared to naive spectrum features, chroma features are similar in the sense that they measure the energy of an audio signal in certain frequency bins. However, while for naive spectrum features, the bins were chosen to give rough indicators for timbre, the bins chosen for chroma features capture musical pitch, therefore creating a big difference between these feature types.

In order to extract chroma features, a set of 120 filters is introduced that extract from an audio file the subbands corresponding to musical pitch, with filter i corresponding to note i in the numbering specified by the MIDI standard. The center frequency f_i of each pitch is computed by

$$f_i = 2^{\frac{i-69}{12}} \cdot 440$$

Plugging in the pitch A4, which has number 69 in the MIDI standard, cf. [14], we get the expected $f_{69} = 440$. For A3 with MIDI number 57, we have $f_{57} = 220$. Note that the ratio of the two frequencies equals 2, since they span an octave, making them octave equivalent.

The audio signal is filtered with the 120 filters each, yielding 120 subband signals. For every subband, the *short-time mean squared power* is computed by squaring the subband signal and then convolving it with a rectangular window of length w. This process yields 120 subband energy signals that are now condensed into 12 signals by adding up all signals corresponding to the same note modulo octave equivalence, i.e. the signal corresponding to the note C is the sum of the subband energy signals corresponding to C0, C1, C2, ..., C9. Finally, the 12 signals can be interpreted as a feature sequence of 12-dimensional chroma vectors, as plotted for several songs in Section 6.1.

This information in general won't be useful for homogeneity-based clustering, since even very simple pop songs feature an alternation of at least 2 chords in most segments, as we will see in Subsection 6.1.1. However, chroma features can be used to demonstrate the use of music segmentation as a preprocessing step by clustering not towards a segmentation representing the musical form but rather towards a chord-level segmentation in which segments correspond to intervals without harmonical changes. Furthermore, since most segment boundaries feature a harmonical change, a homogeneity-based segmentation approach on chroma features is likely to find most of the true segment boundaries, but also many false boundaries.

3 Feature Evaluation Methods

We now try to measure the quality of features, i.e. their utility in a subsequent clustering step. Note that we are not striving for a feature evaluation step that could be used to select features during execution of a segmentation algorithm. Instead, we are trying to identify promising features in the 'design phase' of our segmentation algorithm. In this phase, we are given a human reference segmentation for every audio file along with a sequence of features that we wish to evaluate against the human reference.

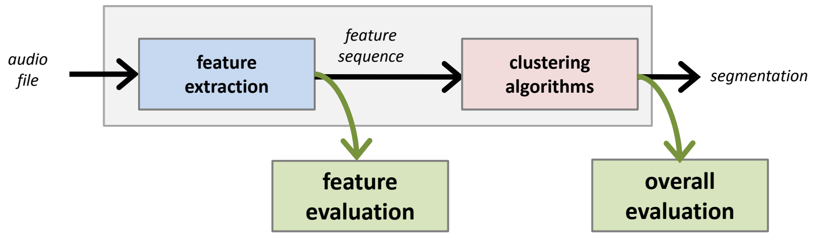


Figure 3.1: Feature evaluation approaches and their position and input specification in the modularized framework proposed in this thesis, marked with green

One approach to this problem could consist of a so-called *wrapper method*, i.e. fixing some clustering algorithm as a reference algorithm and then comparing the performance of this algorithm when fed with data from different features computed from the same song: This way, we would obtain a natural *algorithm-dependent* feature score that directly allows for practical interpretations.

This approach has the major advantage of producing practical scores that also account for the possible transformations towards a more suitable representation for clustering a feature set might undergo in the clustering step. We will discuss this method in Chapter 6. In this chapter, we instead focus on *algorithm-independent* feature evaluation methods that don't rely on clustering algorithms and we present two different approaches for this kind of feature evaluation: The first method relates the quality of a feature sequence to distance-related properties. The second method tries to establish a notion of feature ambiguity in order to compare different feature types with respect to their unambiguity.

3.1 Dissimilarity-based Evaluation

Remembering our model assumption, we try to translate the notion of intra-cluster similarity and inter-cluster dissimilarity into mathematical notation. Since we want to formalize the notion of (dis-)similarity, we require a dissimilarity measure $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ that maps each pair of feature vectors to a non-negative number. Our approach is based on the following two scores inspired by the objective function minimized by the K-means algorithm.

- **Homogeneity within clusters:** In the candidate feature sequence, every part corresponding to one true segment should exhibit small pairwise dissimilarities between points. We average these dissimilarity values into the **intra-cluster dissimilarity** D_k^{intra}:
  $D_k^{intra} = \frac{1}{|C_k|^2} \sum_{i,j \in C_k} d(x_i, x_j)$
- **Separation of clusters:** Feature sets corresponding to different segments should be separated, i.e. the average dissimilarity between vectors from C_k to C_l should be large. This quantity will be called **inter-cluster dissimilarity** $D_{k,l}^{inter}$:
  $D_{k,l}^{inter} = \frac{1}{|C_k| \cdot |C_l|} \sum_{i \in C_k} \sum_{j \in C_l} d(x_i, x_j)$

According to these scores, a feature sequence follows the human segmentation if D_k^{intra} is low for every label $k \in \mathcal{Y}$, and $D_{k,l}^{inter}$ is high for every pair of labels $(k, l) \in \mathcal{Y}^2$. To allow for comparison across different pieces, we condense the values:

$$D^{intra} = \frac{1}{K} \sum_{k=1}^{K} D_k^{intra} \quad \text{and} \quad D^{inter} = \frac{2}{K(K-1)} \sum_{k=1}^{K} \sum_{l=k+1}^{K} D_{k,l}^{inter}$$

It is important to note that both D^{inter} and D^{intra} depend completely on the choice of $d(\cdot, \cdot)$. We can therefore evaluate not only the features, but also the dissimilarity measure chosen on the features, which could prove especially useful when considering manipulated dissimilarity measures in Section 5.3. For now, we choose the following dissimilarity measures for evaluation:

- **Cosine dissimilarity:** $d(x_i, x_j) = 1 - \frac{x_i^T x_j}{|x_i| \cdot |x_j|}$
- **Euclidean distance:** $d(x_i, x_j) = \sqrt{(x_i - x_j)^T (x_i - x_j)}$

Finally, we condense D^{inter} and D^{intra} into one single value $D = D^{inter} / D^{intra}$ that measures the ratio between inter- and intra-cluster dissimilarity. Higher D means better homogeneity and separation of features under the evaluated dissimilarity measure.

3.1.1 Examples for Dissimilarity-Based Evaluation

As a first example, consider the toy data set depicted in the left part of Figure 3.2 that shows 100 two-dimensional feature vectors, 50 each drawn from two bivariate Gaussian distributions $G_1 = \mathcal{N}((-2,0)^T, I)$ and $G_2 = \mathcal{N}((2,0)^T, I)$. Feature vectors depicted in red are drawn from G_1, blue vectors are drawn from G_2. In the reference segmentation underlying this example, equi-colored points are assigned to the same cluster. This data set represents a state of feature separation that we wish to find in promising feature sets. Consequently, the dissimilarity-based feature score D should have a large value. Note that for all examples in this subsection, we set the dissimilarity measure to the Euclidean distance.

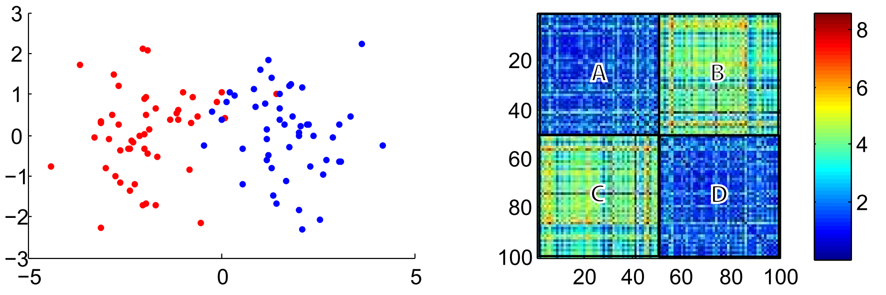


Figure 3.2: **(left)** Feature set drawn from Gaussians with unit variance and mean distance 4, **(right)** self-dissimilarity matrix with diagonal blocks A, D and off-diagonal blocks B, C.

In order to get a better intuitive understanding of dissimilarity-based feature evaluation, the self-dissimilarity matrix of the feature set is plotted in the right part of Figure 3.2. For this, the feature set has been ordered to a sequence in such a way that every red vector precedes all blue vectors. The color value at the point (i, j) in this plot encodes the dissimilarity value $d(x_i, x_j)$.

The values $D^{intra}_{red} = 1.8$ and $D^{intra}_{blue} = 1.75$ can be interpreted as the mean values of the squares A and D in the right part of Figure 3.2. Similarly, $D^{inter}_{red,blue} = 3.89$ can be interpreted as the mean value of the off-diagonal square B, which equals to that of square C, since these squares are transposes of each other due to the symmetry of dissimilarity measures. Furthermore, visual inspection of the dissimilarity matrix clearly shows a block-like structure. Consequently, the value for D^{intra} is low compared to the value of D^{inter}, yielding the desired high ratio $D = 2.18$.

For the second example depicted in Figure 3.3, 50 feature vectors have been drawn from two Gaussians $G_1 = \mathcal{N}((-1,0)^T, I)$ and $G_2 = \mathcal{N}((1,0)^T, I)$ each, halving the distance between means compared to Figure 3.2. While D^{intra} doesn't change significantly compared to the previous setting since the variances of the Gaussians we sample from didn't change, $D^{inter} = 2.48$ is lower due to the smaller distance between means. Therefore, D decreases to 1.44, yielding the expectedly low feature separation value. Furthermore, we barely recognize a block-structure in the dissimilarity matrix.

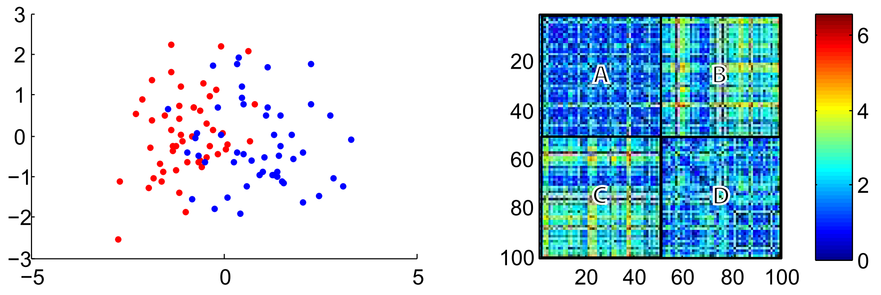


Figure 3.3: **(left)** Feature set drawn from Gaussians with unit variance and mean distance 2, **(right)** self-dissimilarity matrix with diagonal blocks A, D and off-diagonal blocks B, C.

While the value D is able to measure feature separation appropriately for simple settings, its performance can drop significantly in some malicious scenarios. For instance, consider the data set in Figure 3.4, this time drawn from Gaussians $G_1 = \mathcal{N}((0,0)^T, diag(2,1))$ and $G_2 = \mathcal{N}((2,0)^T, diag(0.1,0.1))$. We notice that the blue points corresponding to G_2 overlap with the red points drawn from G_1, and we would therefore intuitively assign a low feature separation value to this data set.

However, $D^{intra} = 1.6$ is relatively low, since it is the average of a high $D^{intra}_{red} = 3.03$ and a very low $D^{intra}_{blue} = 0.17$. Furthermore, $D^{inter} = 2.84$ is high, since there are many red points with high distance to blue points as the red cluster is intrinsicially inhomogeneous. This yields a final value of $D = 1.77$ which doesn't reflect the fact that the set of blue points is contained within the set of red points. Again, the results can be confirmed visually: Squares A and D have very high, resp. very low mean values. Furthermore, squares B and C exhibit almost constant values along one axis due to the fact that the distances from any red point to all blue points are almost equal.

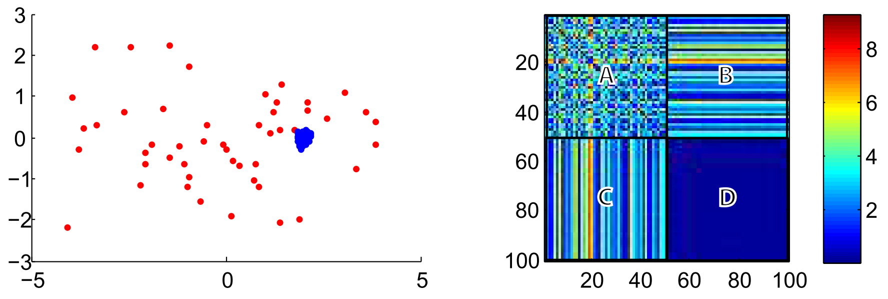


Figure 3.4: **(left)** Feature set drawn from Gaussians with high variance (red) and tiny variance (blue) such that the blue sample is contained in the red sample, **(right)** self-dissimilarity matrix with diagonal blocks A, D and off-diagonal blocks B, C.

3.1.2 Discussion of Properties

The ratio D has the useful property of being invariant with respect to global rescalings of the features for dissimilarity measures d satisfying $d(c\cdot x_i, c\cdot x_j) = c\cdot d(x_i, x_j)$ since the multiplication of all feature vectors by a constant c can be factored out of $d(c\cdot x_i, c\cdot x_j)$, yielding $c \cdot D_k^{intra}$ and $c \cdot D_{k,l}^{inter}$. In the subsequent summation, c can be factored out again, yielding in total:

$$D = \frac{c \cdot D^{inter}}{c \cdot D^{intra}} = \frac{D^{inter}}{D^{intra}}$$

Since cosine dissimilarity is already invariant to global rescalings, the invariance of D to these transformations doesn't add any benefits. However, when using the Euclidean distance, which is linear in the way described above, the previously described invariance allows for evaluation of different feature types whose normalization method can also differ. This property might seem trivial, but in a later subsection, we will encounter another evaluation score that suffers from lacking such an invariance.

The condensing of different values to one single average value however imposes a significant problem to a quantitative comparison of different scores computed by the described dissimilarity-based approach. While one could think of more elaborate ways of averaging the K, respectively $\binom{K}{2}$ values needed to compute D^{inter}, respectively D^{intra}, any such method also increases the complexity of the feature evaluation method and renders intuitive interpretations of the score more difficult.

However, we try to consider at least one such alternative by setting:

$$D_{max}^{intra} = \max_{k\in\mathcal{Y}} D_k^{intra}$$

Building on this maximal intra-dissimilarity, we define, analogously to D, the value $D_{max} = D^{inter}/D_{max}^{intra}$. In this score, we compare the average inter-dissimilarity to the maximal (the worst) intra-dissimilarity, hoping that unwanted effects as described in the third example of the preceding subsection are cancelled. For a comparison of D and D_{max}, we refer to the detailed evaluation in Chapter 6.

In conclusion, the main advantage of the dissimilarity-based score lies in the fact that it can evaluate different dissimilarity measures and in fact doesn't depend on the features themselves, but rather on the dissimilarity values computed for pairs of features, which will be central for algorithms in subsequent chapters. Furthermore, as we will see later, most clustering algorithms try to optimize clustering objectives that incorporate concepts similar to the notion of intra- and inter-cluster dissimilarity. Therefore, dissimilarity-based feature evaluation scores can provide indicators to the performance of arbitrary clustering algorithms in retrieving the human segmentation from a feature sequence.

3.2 Classification-Based Evaluation

In this subsection, we establish a reformulation of the segmentation problem into a classification problem in order to use concepts inspired by machine learning for feature evaluation. More precisely, we estimate the certitude of the best possible classifier, the *Bayes classifier*, on our data set and then use this value as a notion of feature quality. However, in order to evaluate the classifier, one needs to assume that features are drawn from a known distribution. Since we are not given such a distribution, we estimate it. Roughly speaking, the certitude value then measures the amount of 'peak-likeness' of such a distribution.

As will turn out, classification-based feature scores also have a geometrical interpretation that is able to cope with pathological examples such as the one discussed in Subsection 3.1.2, where overlapping features received an unjustifiedly high feature evaluation score. After having introduced some necessary definitions in the next two subsections, this geometrical interpretation will be made more explicit.

3.2.1 Basics from Machine Learning

We first introduce some notions inspired by statistical learning, yielding a probabilistic interpretation for the feature evaluation score to be introduced:

Definition 3.2.1. *To start, a* ***classifier*** $f : \mathcal{X} \to \mathcal{Y}$ *is a function mapping points from a* ***feature space*** $\mathcal{X}$*, like in our case* $\mathbb{R}^d$*, to a set of* ***labels*** $\mathcal{Y} := \{1, \ldots, K\}$*.*

Definition 3.2.2. *The* ***conditional distribution*** *of labels* $P(Y \mid X)$ *is a probability distribution over the random variables* X *and* Y *that range over* $\mathcal{X}$ *and* $\mathcal{Y}$ *respectively. Figuratively, the conditional distribution assigns an array of label probabilities to every feature point. We call the tuple* $(\mathcal{X}, \mathcal{Y}, P)$ *a* ***classification problem****.*

Definition 3.2.3. *The* ***Bayes classifier*** f^* *is an optimal classifier that knows the full conditional distribution and assigns to each feature point the most probable label:*

$$f^*(x) = \arg\max_{y \in \mathcal{Y}} P(Y = y \mid X = x)$$

Definition 3.2.4. *For a classification problem, the* ***certitude function*** $C_f : \mathcal{X} \to [0, 1]$ *describes the unambiguity of* f^**. At any point* $x \in \mathcal{X}$*,* $C_f(x)$ *is defined as follows:*

$$C_f(x) = \max_{y \in \mathcal{Y}} P(Y = y \mid X = x)$$

Note that $C_f(x) \geq \frac{1}{K}$, i.e. for each classification problem the certitude has a lower bound depending on the number of labels which can always be attained by a uniform distribution on the labels. This inhibits comparison of certitude values across different classification problems, creating a need for:

Definition 3.2.5. *We define the* ***normalized certitude function*** $C_f^0(x)$ *by subtracting the lower bound and rescaling the result to* $[0,1]$*:*

$$C_f^0(x) = \frac{K}{K-1}(C_f(x) - \frac{1}{K})$$

Up to now, the certitude functions we introduced assumed that the Bayes classifier always outputs the correct label with respect to a human segmentation. In the label-dependent certitude function, we drop this assumption and instead consider the probability of the correct label given a feature vector under the conditional distribution:

Definition 3.2.6. *The* ***label-dependent certitude function*** $C_l : \mathcal{X} \to [0,1]$ *denotes the probability of the label assigned by a human segmentation function* $\mathcal{S}'$ *:*

$$C_l(x) = P(Y = \mathcal{S}'(x) \mid X = x)$$

An interesting fact about the certitude function, the normalized and the label-dependent certitude function is the possibilty of plotting their value as a curve when ordering the feature vectors according to the feature sequence. This plot then yields an indicator for the ambiguity of features over the domain of a piece. To allow for better comparison across audio files, we finally condense the certitude function into its expected value.

Definition 3.2.7. *The* ***expected certitude*** $\mathcal{C}_f$ *is defined as* $\mathcal{C}_f := \mathbb{E}_X[C_f(X)]$*, and analogously the* ***expected normalized certitude*** *and* ***expected label-dependent certitude*** *are defined as* $\mathcal{C}_f^0 := \mathbb{E}_X[C_f^0(X)]$ *and* $\mathcal{C}_l := \mathbb{E}_X[C_l(X)]$*.*

Given a conditional distribution estimated from the feature sequence of a piece, we employ $\mathcal{C}_f^0$, with $\mathbb{E}_X$ taken over the feature sequence only, as a feature evaluation score. The higher $\mathcal{C}_f^0$, the more time frames have one single label with high conditional probability. For small $\mathcal{C}_f^0$, the feature sequence yields a random classifier.

Remember that for the (normalized) certitude value, it does not matter whether the Bayes classifier labeled the feature point x_i correctly. Here, we measure only the certitude, regardless of a possible misclassification. In the label-dependent certitude value however, only the conditional probability of the label assigned by the human segmentation counts towards the feature evaluation score. We will analyze both values with respect to a set of example audio files in Chapter 6.

Generally, the possibility of plotting the certitude functions over the domain of an audio file, as can be seen in the following example and at the end of Subsection 3.2.2, is an advantage of classification-based evaluation compared to dissimilarity-based evaluation. However, feature vectors need to be embedded in an Euclidean space, as will turn out in Subsection 3.2.2.

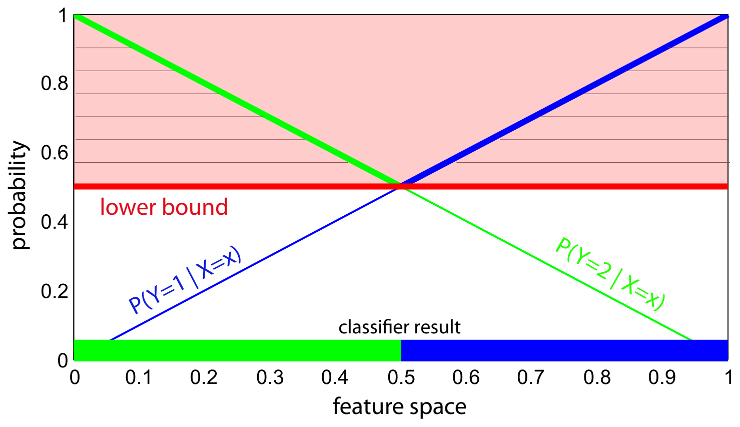


Figure 3.5: Simple two-class classification problem with complete information on the one-dimensional feature space $[0, 1]$, discussed in detail in Example 3.2.1

Example 3.2.1. *Consider as an example the two-class classification problem depicted in Figure 3.5: $\mathcal{X} = [0, 1]$, $\mathcal{Y} = \{1, 2\}$, corresponding to green and blue, the distribution of X is considered to be uniform on $[0, 1]$, and the following conditional distribution indicates the label probabilities for every $x \in \mathcal{X}$:*

$$P(Y = 1 \mid X = x) = 1 - x \quad \textit{and} \quad P(Y = 2 \mid X = x) = x$$

The Bayes classifier f^* on this problem is just $f^*(x) = 1_{x>0.5} + 1$. This function is plotted as a colored stripe onto the x-axis.

The lower bound on C_f is $\frac{1}{K} = 0.5$ and corresponds to the red line. The certitude function is drawn with thick lines above the red lower bound. Note that $C_f(0) = C_f(1) = 1$ and C_f reaches the lower bound at 0.5. The expected certitude C_f is:

$$C_f = \mathbb{E}_X[C_f(X)] = \int_0^1 C_f(x) \cdot P(x)\ dx = 2 \int_0^{0.5} 1 - x\ dx = 0.75$$

From this value we get the expected normalized certitude in the following way, exploiting linearity of the expected value:

$$C_f^0 = \frac{K}{K-1}(C_f - \frac{1}{K}) = 0.5$$

Note the shaded area limited by the certitude function and the red lower bound and note that C_f^0 is the ratio of the shaded area to the area of the red rectangle. Since the Bayes classifier is consistent with the human segmentation on this feature set, the certitude function agrees with the label-dependent certitude function, yielding $C_l = C_f = 0.75$.

3.2.2 Density Estimation

Given some independently drawn and identically distributed samples $x_1, \dots x_N$, in our case feature vectors $x_i \in \mathcal{X}$, drawn from an unknown underlying distribution $\mathcal{D}$, the task of inferring this unknown distribution is called *density estimation*. Approaches to this problem can be divided into the class of *parametric methods*, where one tries to fit the parameters of some parameterized class of distributions, such as Gaussians, to the data, and that of *non-parametric methods*. Since we have no assumptions in our feature distribution, we choose the latter.

More precisely, we use *kernel density estimation*. This means introducing a basis density function $\mathcal{B}_i$ for every x_i in the feature sequence[1] (X), which in our case will be that of an isotropic Gaussian centered at the sample point: $\mathcal{B}_i \sim \mathcal{N}(x_i, \sigma^2 I)$. The estimated density $\mathcal{D}_{est}$ is then just the pointwise average over all basis densities:

$$\mathcal{D}_{est} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{B}_i$$

Therefore, each sample point x increases the estimated density within a small region around it. This region will be called the influence region of x from now on. The choice of the size of this region, i.e. the choice of σ^2 plays a major role for kernel estimation, as illustrated in the following figure, where only the right image resembles the Gaussian distribution the samples were drawn from.

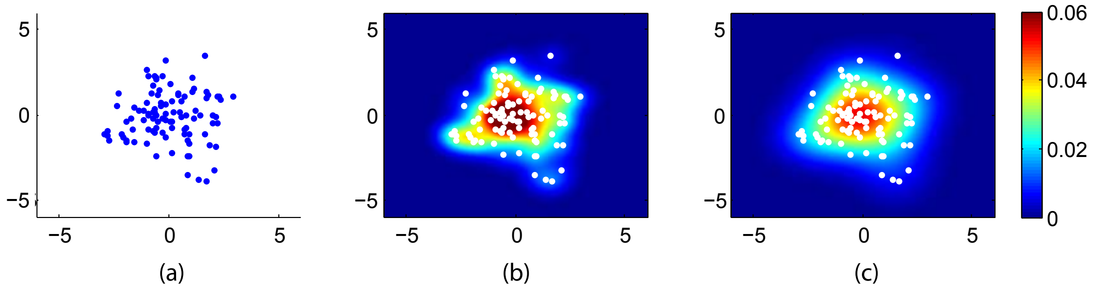


Figure 3.6: Influence of different kernel sizes in kernel density estimation: **(a)** samples drawn from $\mathcal{N}((0,0)^T, 1.5I)$, **(b)** estimated density $\mathcal{D}_{est}$ with $\sigma^2 = 0.3$, **(c)** $\mathcal{D}_{est}$ with $\sigma^2 = 1$.

For normalized features, we will consider three values for σ^2: 0.05, 0.1 and 0.2. We now try to estimate the unknown conditional distribution $P(Y \mid X = x)$ from the previous subsection for every x in the feature sequence (X). As our label set is finite and has K elements, we can do this by estimating all probabilites $P(Y = y \mid X = x)$. Furthermore, we assume that all labels have the same prior probabilty $P(Y = y) = \frac{1}{K}$. This allows for the derivation on the next page:

[1]In this subsection, we denote a feature sequence by (X) in order to prevent confusion with X

$$
\begin{aligned}
P(Y = y \mid X = x) \ &= \ \frac{P(X = x \mid Y = y)P(Y = y)}{P(X = x)} \\
= \ \frac{P(X = x \mid Y = y)\frac{1}{K}}{\sum_{i=1}^{K} P(X = x \mid Y = i)\frac{1}{K}} \ &= \ \frac{P(X = x \mid Y = y)}{\sum_{i=1}^{K} P(X = x \mid Y = i)}
\end{aligned}
$$

In the first step, we used Bayes' rule. Then, we expanded the marginal probability according to the law of total probability, using the fixed label prior. In the third step, we reduced the fraction, yielding that $P(Y = y \mid X = x)$ is equal to the probability of x under the label y divided by the sum of the probabilities of x under all labels.

The class-conditional density $p(X \mid Y = y)$ needs to be estimated: For every $y \in \mathcal{Y}$, we determine it by density estimation, restricted to samples from the cluster C_y, i.e. we estimate the distribution of feature vectors labelled as y in the reference segmentation. From this, we can also get the marginal $P(X = x)$ by marginalizing over Y, as can be seen in the denominator of the last fraction. Now, remember that our certitude feature score depends only on $x_i \in (X)$. We therefore evaluate $P(Y \mid X = x_i)$ only for $x_i \in (X)$, not for the whole feature space $\mathcal{X}$.

The results of density estimation together with the notion of a certitude function lead to the following interpretation: Consider some feature vector x having the label y. A set I of feature vectors, which might be a singleton set consisting only of x, can be found s.t. every vector $x' \in I$ lies within the influence region of x, i.e. within a region where its Gaussian kernel function attains a value significantly larger than 0. Now, depending on the label of x', say y', this feature vector will increase the value of $P(Y = y' \mid X = x)$. If the influence region contains mostly points of one single label, say y'', this will create a peaked distribution for $P(Y \mid X = x)$, therefore also yielding a high certitude value for x. If even $y'' = \mathcal{S}'(x)$, i.e. the label y'' matches the label for x in the human segmentation, the label-dependent certitude function also attains a high value.

Note that the choice of σ^2 in the density estimation part also allows for measuring the robustness of features against additive Gaussian noise with variance σ^2: In the limit of $\sigma^2 \to 0$, every (noiseless) feature vector attributes a Dirac distribution to the total estimated density, placing all probabilty mass exactly on the feature vector and yielding maximal certitude. For higher σ^2, the mass is spread over a larger space through noise, therefore decreasing the certitude. Consequently, if the certitude drops only at high kernel sizes, the evaluated features are good at discriminating between true segments even in the presence of strong noise. For naive spectrum and chroma features, this noise can even be interpreted as Gaussian noise added to every frequency band.

However, the strong dependence of the classifier-based evaluation score on the kernel size used for density estimation suggests a score that makes this dependence explicit. One might for instance think of considering as evaluation result not only the certitudes evaluated for three fixed kernel sizes but rather a song-dependent function describing the correlation between kernel size and certitude. The area under this curve could then indicate the stability of features with respect to noise more accurately. However, in a practical setting, this area again has to be approximated by finite sums. Since the evaluation of the estimated density can be costly, we rather restrict the evaluation to three values for σ^2.

The figure shows two conditional distributions estimated from MFCC features that have been extracted from the *Highway Song*. Along the x-axis time progresses, while column i represents the distribution $P(Y \mid X = x_i)$ for x_i being the i-th feature vector in (X). Note that increasing σ^2 shifts each column of the conditional distribution towards a uniform distribution, therefore decreasing the certitude. In particular, the limit of $\sigma^2 \to 0$ will lead to a certitude of 1, as already pointed out. To get more meaningful results, we will evaluate features with respect to three choices of σ^2 in Chapter 6.

The curves in the second and fourth subplot are the certitude function and the label-dependent certitude function computed for the estimated conditional distribution each. In general, the certitude function is just the column-wise maximum over the conditional distribution. Note that, due to boundary effects, as described in Section 5.1, both functions attain low values at segment boundaries. Furthermore, we notice that only some segments still exhibit high certitude values after the increase of kernel size, e.g. the verse part starting at 20 seconds and peaks at 65 and 163 seconds.

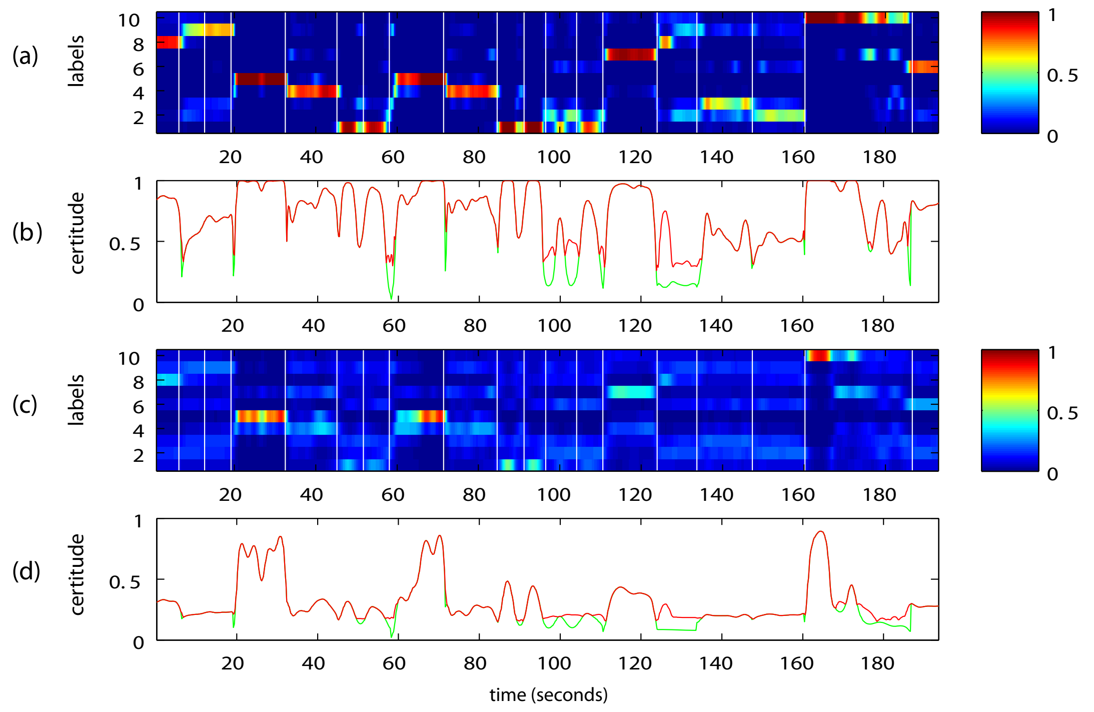


Figure 3.7: Density estimation on smoothed (kernel size 3 sec.) MFCC features computed for *Highway Song*: **(a)** Conditional distribution estimated with $\sigma^2 = 0.05$, plotted in such a way that the color value at (i, j) represents $P(Y = i \mid X = x_j)$. **(b)** Red: certitude function for conditional distribution in (a), green: label-dependent certitude function. **(c)** Conditional distribution for $\sigma^2 = 0.2$, **(d)** as for (b), but with conditional distribution (c).

It is absurd to divide people into good and bad.
People are either charming or tedious.

Oscar Wilde

4 Time-Ignoring Clustering

In this chapter, we focus on the reformulation of music segmentation in terms of clustering, i.e. we search for a partition of a feature sequence and its audio file into clusters such that the dissimilarity of feature vectors within clusters is minimized, while dissimilarity between clusters is maximized. Actually, we encountered main principles of clustering methods for the first time when introducing dissimilarity-based evaluation in Chapter 3.1, where we especially defined the values D^{intra} and D^{inter} in order to measure the extent to which the homogeneity assumption was given in a file.

In the setting of this chapter, we are given the feature sequence to a file which we try to segment automatically in such a way that the result resembles the human reference segmentation associated with the file. Of course, this task is futile if the human segmentation did not follow any rules. At this point, the homogeneity assumption has to be used in order to presume that the song structure is reflected in the distribution of homogeneous regions within the feature sequence.

Since clustering methods allow for identification of homogeneous regions under some (dis-)similarity measure, we can use them for music segmentation under the homogeneity assumption. The definition of a homogeneous region however is part of the chosen clustering method since it is determined by the clustering goal the algorithm tries to optimize. It is worth mentioning that most of the goals featured in known clustering methods lead to NP-complete optimization problems that need to be relaxed or approximated.

To create baseline algorithms, we ignore temporal properties like the order of the feature sequence and rather speak of a feature *set* within this chapter. Accordingly, this feature set can also be viewed as a set of feature points in $\mathbb{R}^d$, allowing for a geometrical interpretation of clustering. However, this interpretation fails when considering arbitrary dissimilarity measures, since these in general lack a geometric interpretation.

Note that the ordering of the feature points given by the feature sequence would allow to recreate a temporal path through this cloud of feature vectors. As we neglect the feature sequence, such a path can't be created in time-ignoring clustering.

4.1 K-Means Clustering

In K-means (or minimum sum-of-squares) clustering, feature vectors are assigned to clusters $C_1 \dots C_K$, each of them having a cluster centroid μ_k defined as the unweighted average of all vectors assigned to this cluster. The centroid can be regarded as a prototype vector indicating the expected feature vector within a cluster. Note that the number of clusters K is part of the input, meaning that the algorithm does not have to infer this value itself.

The goal of K-means clustering now is to find an assignment of feature vectors to clusters (and therefore to cluster centroids) that minimizes the sum of squared distances from every feature vector to its assigned cluster centroid. The objective value of a segmentation $\mathcal{S}$ is formalized as follows:

$$obj(\mathcal{S}) = \sum_{C_k \in \mathcal{S}} \sum_{x_i \in C_k} ||x_i - \mu_k||^2$$

While this objective function looks rather simple on first sight, it leads to an infeasible combinatorial optimization problem: At least since [1], it is known that finding an optimal segmentation with respect to the above defined obj is NP-hard for $K \geq 2$ if the feature vectors are drawn from $\mathbb{R}^d$ for arbitrary d. An exponential-time algorithm is known whose runtime behaves like $O(n^{dK})$ where n is the number of feature vectors, therefore running in polynomial time for fixed d and K. However, even this ‘polynomial’ algorithm is of no use in our setting, where $d \approx 12$ and $5 \leq K \leq 10$ in most of the cases.

We therefore have to resort to heuristic algorithms for K-means clustering, among which the best known is the K-means algorithm. This simple algorithm uses the idea of bootstrapping in order to iteratively refine an initially random segmentation until possibly converging to an optimal segmentation (cf. [17]) and works as follows:

1. Randomly select K vectors from the feature set X as centroids $\mu_1 \dots \mu_K$. Optionally, an explicit set of initial centroids can be specified manually.
2. Assign each feature vector to its nearest centroid, yielding the following clusters:
$$C_k \leftarrow \{x_j \mid k = \arg\min_l ||x_j - \mu_l|| \}$$
3. Recompute the cluster centroids with respect to the new clusters:
$$\mu_k \leftarrow \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i$$
4. Repeat steps 2 and 3 until after some iteration, no point is assigned to another cluster than it was assigned to before the iteration. In this case, the algorithm has converged. If after a fixed number of iterations, no convergence occured, report failure. At this point, one could restart the algorithm.

Certainly, this algorithm is not guaranteed to converge to the optimal solution. For general sets of feature vectors, its performance is influenced heavily by the choice of the initial centroids. In order to cope with this dependence on the initial guess, some more refined strategies are proposed in several implementations, among which we mention the following:

- Instead of selecting the initial centroids randomly out of the set of feature vectors, rather uniformly select a small subset $X' \subset X$ of the set of feature vectors and then run the K-means algorithm on X', with initial centroids drawn uniformly out of X'. The centroids determined by this first run can then serve as initial centroids for a subsequent execution of the K-means algorithm on the whole set of feature vectors X.
- Run the K-means algorithm on a fixed number of randomly sampled initial centroid sets, possibly chosen by the previously described refined sampling method, yielding many segmentations. Then output the best segmentation with respect to obj.

In our MATLAB implementation of the music segmentation framework we use the function `kmeans` in the *Matlab Statistics Toolbox* as an implementation of the K-means algorithm. For our application, we run the K-means algorithm on 100 different initial centroid sets, each set being determined by the first strategy, where we randomly choose a subset containing 10% of the feature set. The evaluation results with respect to several choices of K are presented in Chapter 6.

4.2 Agglomerative Clustering

While in K-Means clustering, features were clustered in order to minimize the sum of squared distances to cluster centroids, in agglomerative clustering a hierarchical idea is applied to the clustering problem. Hierarchical algorithms in general start with a trivial segmentation of a feature set, e.g. the segmentation consisting of only one cluster or the segmentation that assigns every feature vector its own singleton cluster. Then, these algorithms iteratively refine a current segmentation by joining similar or splitting intrisically dissimilar clusters until reaching the other trivial segmentation. Agglomerative clustering algorithms choose the first way, divisive ones choose the second.

Since the result of transforming a trivial segmentation to another trivial segmentation is highly uninteresting for practical purposes, several possibilities allow for obtaining practical segmentation results out of the transformation *process*, cf. [6]:

- The hierarchical algorithm can stop when a fixed number of clusters K has been obtained in the current segmentation. This parameter must be part of the input.
- At some point in the transformation process of an agglomerative clustering algorithm, the most similar pair of clusters that can be joined can be more dissimilar than some threshold t. Depending on the choice of t, this could be a good point to stop the algorithm. Again, t is part of the input.

4.2.1 Cluster Dissimilarities

A central notion that remained unclear in this short introduction to hierarchical clustering was the notion of cluster dissimilarity. The cluster dissimilarity $D(C_i, C_j)$ between two clusters C_i and C_j can be defined in several ways, always presuming a dissimilarity measure d for features, as required in Section 3.1:

- The average linkage value D_{avg} between clusters is the **mean** dissimilarity between all pairs of feature vectors that have exactly one element in each cluster. Formally, $D_{avg}(C_i, C_j) = \frac{1}{|C_i| \cdot |C_j|} \sum_{k \in C_i} \sum_{l \in C_j} d(x_k, x_l)$
- The single linkage value D_s between clusters is the **minimal** dissimilarity between all pairs of feature vectors that have exactly one element in each cluster. Formally, $D_s(C_i, C_j) = \min_{k \in C_i} \min_{l \in C_j} d(x_k, x_l)$
- The complete linkage value D_c between clusters is the **maximal** dissimilarity between all pairs of feature vectors that have exactly one element in each cluster. $D_c(C_i, C_j) = \max_{k \in C_i} \max_{l \in C_j} d(x_k, x_l)$

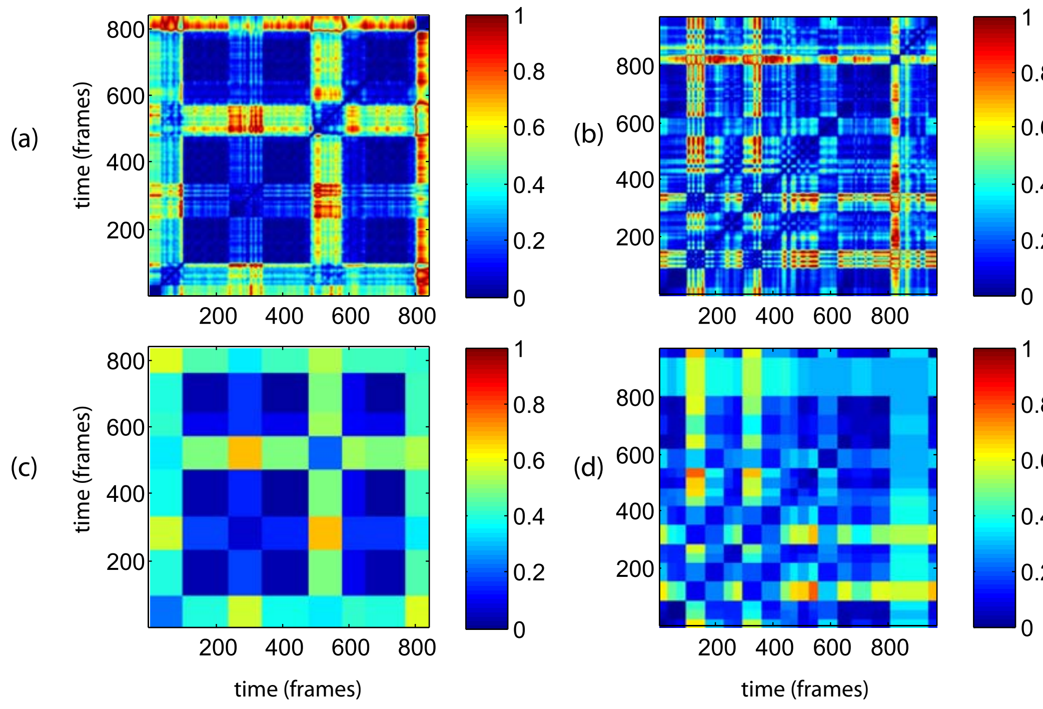


Figure 4.1: From dissimilarity matrices to cluster-dissimilarities: **(a)** cosine self-dissimilarity matrix for *Rock and Roll Queen*, **(b)** cosine self-dissimilarity matrix for *Highway Song*, **(c)** pairwise average linkage values for segments in human segmentation for (a), **(d)** average linkage values for (b).

Both single and complete linkage values suffer from outliers in features, meaning that some malicious pair of feature vectors can dominate the linkage value for two clusters which then doesn't depend on the remaining pairs of feature vectors in these clusters. The average linkage value however is robust against such outliers, due to the fact that every pair of feature vectors can contribute only a fraction of $\frac{1}{|C_i|\cdot|C_j|}$ to the mean value.

Recall the definition of a self-dissimilarity matrix H: $H(i,j) = d(x_i, x_j)$. In order to get a better visual understanding of the average linkage value, in Figure 4.1 the self-dissimilarity matrices of two audio files and the corresponding average linkage values computed for the segments in the human reference segmentations are plotted. Every submatrix $H(I(S_i), I(S_j))$ of H, where S_i and S_j are segments in a human reference segmentation, is replaced by a constant submatrix of the same dimensions. This submatrix contains only the value $D_{avg}(I(S_i), I(S_j))$ which is equal to the mean value of $H(I(S_i), I(S_j))$, yielding the matrices of average linkage values plotted in (c) and (d).

Therefore, the matrices in (c) and (d) show the average linkage values between different segments. For instance, the first, fifth and seventh segment in the song *Rock and Roll Queen* have high dissimilarity to all other segments with respect to average linkage.

4.2.2 Clustering Algorithm

The pseudocode for an agglomerative clustering algorithm is printed below: This algorithm is given the following input: an $N \times N$ self-dissimilarity matrix H, a cluster dissimilarity measure, in our case the average linkage D_{avg}, and a threshold value t. Note that the algorithm doesn't depend on the feature sequence itself, but rather on H.

The algorithm stores an array A of already computed cluster dissimilarities, speeding up computation. A result set R of clusters will contain the result of the algorithm after all iterations. This set R can be transformed to a segmentation $\mathcal{S}$ by decomposing each cluster into temporally connected intervals and assigning these intervals a common label.

In step 1, the algorithm starts by setting R to the trivial set of clusters that consists only of singleton sets. Then, in step 2, it computes the average linkage between all such clusters, which can be directly read off the self-dissimilarity matrix H since all clusters are singletons at this point. At this stage, average linkage values correspond directly to feature dissimilarity values.

In every iteration of step 3, the two clusters with minimal cluster dissimilarity to each other are located by using the matrix A of already computed cluster dissimilarities. Then, the algorithm checks whether this dissimilarity value is below the threshold t. Note that in every iteration of the loop in step 3, new clusters C_a and C_b are searched for. If no such pair of clusters can be found, the algorithm stops and has found the set of clusters R. If the threshold value was chosen too high and during execution R consists of only one cluster, the algorithm also stops since it cannot find two different clusters.

The merging of clusters in step 4 consists of replacing C_a by the union $C_a \cup C_b$. Furthermore, the row and the column corresponding to C_b are deleted from A. In step 5, the values for A are finally updated by replacing the row and column corresponding to C_a with the new average linkage values to all other clusters.

```
1. Given an audio file $B = [1 : N]$, set $C_i = \{i\}$ for all $i \in B$ and $R \leftarrow \{C_1, \dots, C_N\}$.
2. For all $i, j \in 1, \dots, N$, set $A(C_i, C_j) \leftarrow H(i, j)$.
3. While $A(C_a, C_b) < t$ for $C_a \neq C_b$ that minimize $A(C_a, C_b)$, repeat 4 and 5:
4. Set $C_a \leftarrow C_a \cup C_b$, delete $C_b$ from $R$ and the corresp. row and col. from $A$.
5. For all $C_i \in R$, set $A(C_a, C_i) \leftarrow D_{avg}(C_a, C_i)$.
   Then, for all $C_i \in R$, set $A(C_i, C_a) \leftarrow A(C_a, C_i)$.
```

This algorithm already saves some computation time compared to a naive implementation by storing cluster-dissimilarity values computed in step 5 in the array A. However, this still is not optimal since $O(N)$ average linkage values are recomputed from scratch in every iteration. This overhead will be reduced in Subsection 5.3.2.

4.2.3 Dendrograms

Finally, the results of agglomerative clustering can be visualized by means of a plotting technique called **dendrogram**, cf. [12]. The dendrogram corresponding to an execution of agglomerative clustering on a set of feature vectors is a binary tree in which nodes correspond to cluster merging events that occur during the execution of the algorithm.

A dendrogram has a leaf for every singleton cluster obtained from the trivial initial segmentation in the agglomerative clustering algorithm. Inner nodes correspond to clusters merged by the algorithm, i.e. the two children C_a, C_b of an inner node C_c are merged to C_c during the execution. Furthermore, C_c is assigned its own height which corresponds to the value $D_{avg}(C_a, C_b)$. This way, homogeneous clusters can be found at the bottom of a dendrogram and more inhomogeneous clusters are found in higher parts.

In Figure 4.2, a dendrogram is plotted together with three segmentations that have been obtained by agglomerative clustering with threshold values 0.05, 0.15 and 0.25. Note that the segmentations feature smaller segments as the threshold value decreases. Dendrograms as well as the exact implementation of agglomerative clustering were given by `dendrogram` and `linkage` from the *Matlab Statistics Toolbox*.

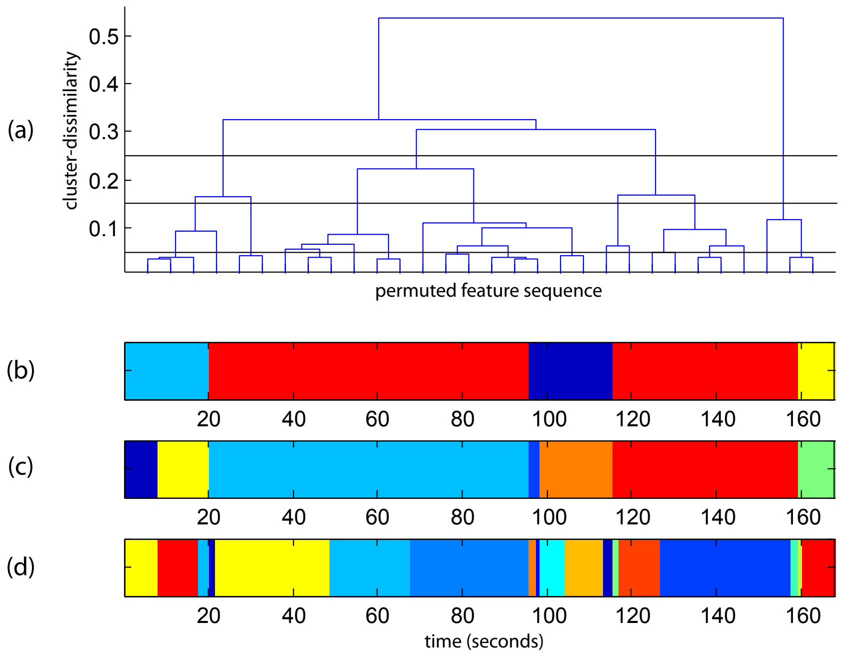


Figure 4.2: Example segmentation of *Rock and Roll Queen* with agglomerative clustering: **(a)** dendrogram with visual aids (feature sequence is permuted in order to avoid crossing lines, merging events occuring below some threshold are omitted) and black horizontal lines indicating cutoff thresholds. **(b)** Segmentation obtained by cutting the dendrogram at the threshold value 0.25 **(c)** as for (b), but with threshold value 0.15, **(d)** as for (c), but with threshold value 0.05.

Yeah, it should've been forever
It all seems to make so much sense
But after a while
You realize time flies
Porcupine Tree - Time flies

5 Time-Dependent Clustering

The clustering algorithms previously described ignore time, which is reflected in the order of the feature sequence. Therefore, the resulting segmentations were sensitive to frequent segment changes, which in the worst case can lead to completely useless segmentations, as will turn out Chapter 6. The tendency of time-ignoring clustering to find more segment boundaries than necessary is caused by the fact that e.g. in a pop song, a time-ignoring clustering algorithm would rather merge a short theme to its repetition at another position in the song than extend the cluster corresponding to the theme to span adjacent themes.

In this section we introduce some methods that allow for time-dependent clustering, i.e. clustering that does not depend solely on the feature set, but rather incorporates temporal information that can be extracted from the feature sequence. These ideas should prevent the problem described above.

Among the methods described in this chapter we first introduce a feature-level manipulation of the feature sequence that accentuates homogeneity within clusters. This method can be used as a preprocessing step for all segmentation methods.

The second method does not try to find homogeneous regions. Instead, it looks for small time intervals with extreme inhomogeneity, hoping that these intervals correspond to segment boundaries. Here, temporal context is introduced only by means of a temporal window that selects a short connected subinterval out of the whole feature sequence.

Within the last two methods, we modify agglomerative clustering in a way such that temporal information is part of the input. First, we manipulate the self-dissimilarity matrices that are part of the input to agglomerative clustering such that the dissimilarity of feature vectors also depends on the temporal distance they have to each other in the feature sequence. In a second approach, we modify the agglomerative clustering algorithm to only merge clusters that are temporally adjacent.

5.1 Feature Smoothing

5.1.1 Increasing Homogeneity by Smoothing

In order to enhance temporal coherence in a feature sequence $X = (x_1, \ldots, x_N)$, we use *feature smoothing*. In this approach every feature vector x_i is replaced by a weighted mean computed within a small temporal subinterval of radius w around x_i. The weights within the interval are given by a window function W. This yields the *smoothed* feature sequence $X' = (x'_1, \ldots, x'_N)$ that is computed as follows:

$$x'_i = \sum_{k=-w}^{w} W(k) \cdot x_{i+k}$$

When replacing features with the local weighted mean, variation of feature vectors decreases in small scales. This effect is caused by the weighted averaging above, which can be equivalently interpreted as a row-wise convolution of X with a one-dimensional window function W. For Gaussian windows, this convolution corresponds to a *blurring* or *low-pass filtering* of the feature components, decreasing the magnitude of high-frequent variation and therefore increasing homogeneity within segments.

Furthermore, smoothing allows dealing with repeating patterns shorter than the smoothing length that in general prevent a segment from being homogeneous. Consider the following feature sequence depicted in Figure 5.1. This sequence is a concatenation of two segments S_1, S_2 that each in turn consist of a repeating alternation of feature vectors. For S_1, these vectors are the canonical basis vectors e_1, e_2, e_3, while for S_2, these vectors are $f_1 = (1, 0, 1)^T$, $f_2 = (1, 1, 0)^T$ and $f_3 = (0, 1, 1)^T$. Abusing regular expressions for describing the sequences, S_1 can be denoted by $((e_1)^5(e_2)^5(e_3)^5)^{10}$, and analogously, S_2 corresponds to $((f_1)^5(f_2)^5(f_3)^5)^{10}$. Note that time-ignoring clustering algorithms should find six clusters that each contain only one distinct feature vector, creating an abundance of segments with length of 5 frames. However, when smoothing the features with a Gaussian kernel of size $w = 40$ frames, as plotted in the second subplot, the local alternation is blurred out, creating homogeneous regions in S_1 and S_2.

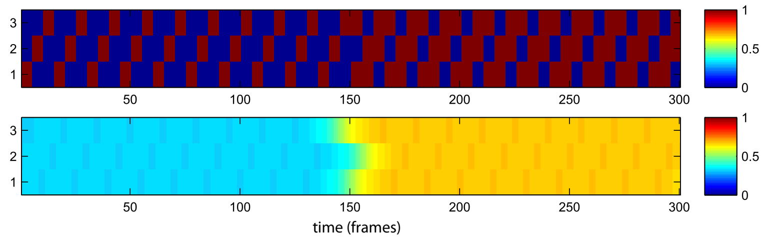


Figure 5.1: Feature smoothing transforms small-scale repetitive structure to homogeneity structure: **(top)** repetitive feature sequence consisting of two segments **(b)** smoothed feature sequence (smoothing kernel of 40 frames), fulfilling the homogeneity assumption.

5.1.2 Effects at Segment Boundaries

Due to the convolution applied in feature smoothing, segment boundaries that appeared clearly in the unsmoothed feature sequence are blurred, making it harder to estimate the exact segment boundary and introducing a boundary region of length about $2w$ between two homogeneous segments S_1, S_2 consisting of linear combinations of feature vectors found at the boundary regions of S_1 and S_2.

Consider as an example the feature sequence depicted in Figure 5.2. Here, all feature vectors in S_1 have the coordinates $x_a = (1,0)^T$, while all vectors in S_2 have coordinates $x_b = (0,1)^T$, as one can see in the first subplot. When smoothing with a Gaussian window of radius $w = 25$, as in the second subplot, the parts of S_1 and S_2 that have temporal distance larger than 25 frames from the segment boundary are not changed since all values in the support of this window are constant. However, the subsequence consisting of indices $[26 : 75]$ is subject to Gaussian blur.

In the third subplot, every feature point x_i is described as a convex combination of x_a and x_b, yielding $x_i = \lambda_1 x_a + \lambda_2 x_b$. The coefficients λ_1,λ_2 are just the entries $x_i(1)$ and $x_i(2)$ and are plotted in blue, resp. green. Note the crossfade-like behaviour of the two curves in the section $[26 : 75]$, showing the change in the proportions of λ_1 and λ_2. For x_{50}, we have $\lambda_1 = \lambda_2 = 0.5$, making this point the average of x_a and x_b.

A problem induced by feature smoothing is that agglomerative clustering tends to place all feature vectors in temporal proximity of a segment boundary into singleton clusters, since these vectors aren't part of a homogeneous region, as we saw in the previous example. While this imposes a problem to clustering, we will soon see another method that is able to exploit exactly these inhomogeneous regions for boundary detection.

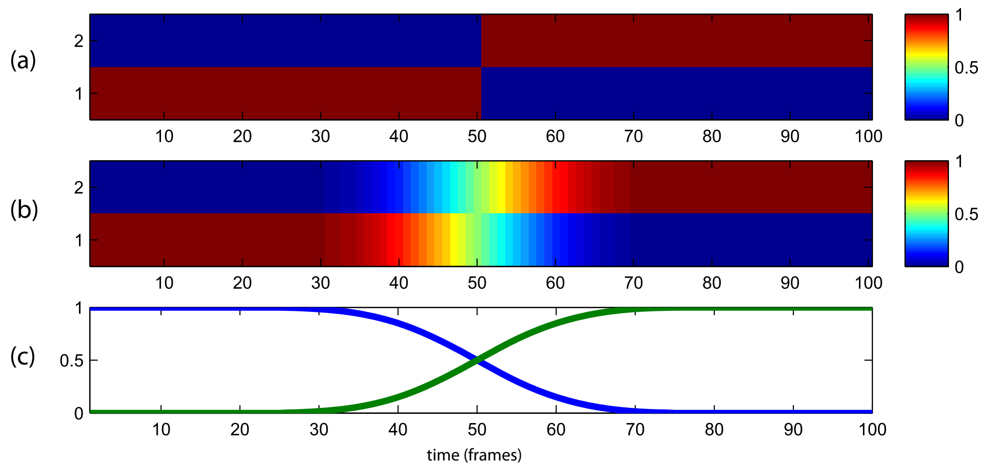


Figure 5.2: Smoothing of segment boundaries: **(a)** simple two-dimensional feature sequence consisting of feature vectors e_1 and e_2, **(b)** feature sequence smoothed with kernel size $w = 25$ frames, **(c)** plots of coefficients for smoothed feature sequence represented as convex combination of x_a and x_b

5.2 Novelty Detection

While feature smoothing presents a simple way of increasing the temporal coherence of a feature sequence, its main drawback lies in its tendency to level differences across segment boundaries in case the smoothing kernel is chosen too large.

Therefore, **novelty functions** that indicate segment boundaries might be of great use when combined with feature smoothing. A novelty function for an audio file A is a function $nov : A \to \mathbb{R}^+$ that attains low values within segments and high values in the local neighborhood of a segment boundary.

5.2.1 Previous Novelty Functions

Such a measure has been discussed in [8], where novelty functions are computed by correlation of a checkerboard kernel shifted along the diagonal of the self-similarity matrix associated with a feature sequence. In this section, we present an alternative way for obtaining novelty functions which is based on statistical properties of the feature sequence. This approach yields results similar to the novelty functions in [8] and can therefore provide a statistical interpretation of this correlation-based novelty detection method.

First, we recapitulate the implicit definition of the novelty functions introduced in [8]. Here, a Gaussian checkerboard kernel K_G of radius L is defined similarly to:

$$K_G(x,y) = 1_{\{|x|\le L, |y| \le L\}} \cdot sgn(x) \cdot sgn(y) \cdot \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

This kernel is non-zero only in the square $[-L, L]^2$, where it attains positive values only for (x, y) with $sgn(x) = sgn(y)$, and negative values elsewhere. The truncation of the checkerboard kernel is done for performance reasons.

After choosing σ, K_G is correlated along the main diagonal of the self-similarity matrix $S \in \mathbb{R}^{N\times N}$ to obtain the novelty function f_{GC} (GC stands for **G**aussian **C**heckerboard):

$$f_{GC}(i) = \sum_{k=i-L}^{i+L} \sum_{l=i-L}^{i+L} S(k,l)K(k-i, l-i)$$

Consider the checkerboard kernel as an $(2L+1)\times(2L+1)$ matrix divided into four $L\times L$ blocks. Negative values can be found in the blocks off the main diagonal, positive values are located in the blocks containing the main diagonal. When this kernel is shifted along the main diagonal of S, as in the sum above, structures that resemble this kernel will yield a high value for f_{GC}. However, any such occurrence of a checkerboard kernel in the self-similarity matrix is likely to be a boundary since it describes a submatrix of the similarity matrix in which two homogeneous regions are adjacent (high similarity in the blocks containing the main diagonal), but have high inter-dissimilarity (low similarity in the blocks off the main diagonal).

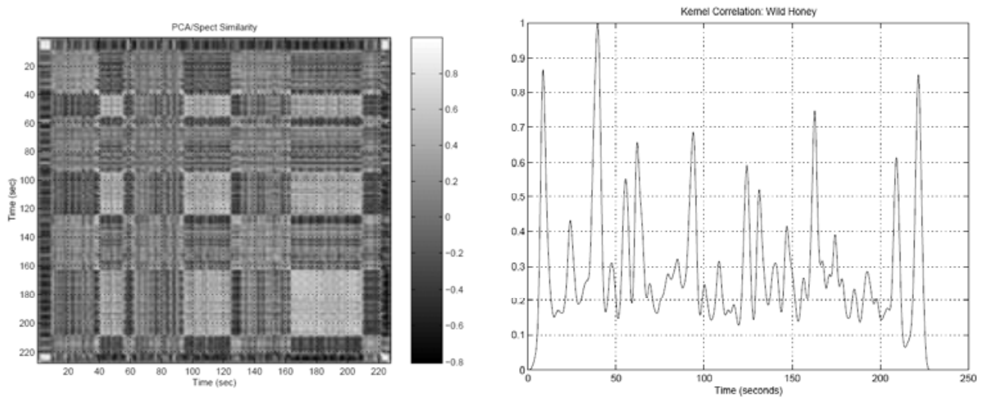


Figure 5.3: **(left)** Self-similarity matrix on MFCC features for *Wild Honey*, **(right)** plot of novelty function f_{GC}. Reproduced from [8].

Figure 5.3, reproduced from the original paper [8], shows a novelty function for MFCC features computed on the song *Wild Honey.* One immediately notices block-like structures in the self-similarity matrix (left). Furthermore, as the kernel moving along the main diagonal of the matrix hits a corner between two such blocks, it creates a peak in the novelty curve plotted to the right-hand side.

In order to facilitate comparison to our approach, we now modify the previous definition, choosing a box checkerboard kernel instead of the Gaussian checkerboard kernel. Such a kernel K_B of radius L is just the following piecewise constant function:

$$K_B(x,y) = 1_{\{|x|\leq L, |y|\leq L\}} \cdot sgn(x) \cdot sgn(y) \cdot \frac{1}{(2L+1)^2}$$

Correlating this kernel in the same way as defined above for K_G, but this time on the *dissimilarity* matrix, yields f_{BC} (BC for **B**ox **C**heckerboard).

In order to provide a more detailed comparison, we also introduce the novelty functions f_{GF} and f_{BF}, with abbreviations standing for **G**aussian **F**lat, respectively **B**ox **F**lat. These functions are obtained by correlating the (dis-)similarity matrix in exactly the same way as above, but with a non-checkerboard Gaussian, respectively box kernel, i.e. every occurence of $K(x,y)$ is replaced by $|K(x,y)|$, neutralizing the checkerboard-like sign changes.

While the flat kernels seem to bring with them a significant deterioration of the quality of novelty functions, even these kernels have an interpretation under the homogeneity assumption: As segments have low intrinsic dissimilarity, the correlation with a flat box kernel will yield low values within segments. When the kernel hits a boundary, a higher correlation can be observed as half of the submatrix to correlate against contains high dissimilarity values. Note that the flat kernel would also detect high novelty at points within segments that have high dissimilarity to all of their local temporal neighborhood. However, feature smoothing helps in reducing these dissimilarity values.

5.2.2 Local Statistics

We define local statistics, i.e. the **local mean** $\mu_w[i] \in \mathbb{R}^d$ and the **local covariance matrix** $Cov_w[i] \in \mathbb{R}^{d\times d}$ with window size $w \in \mathbb{N}$ for a feature point $x_i \in X$ as follows:

$$\mu_w[i] = \frac{1}{2w+1} \sum_{k=i-w}^{i+w} x_k$$

$$Cov_w[i] = \frac{1}{2w+1} \sum_{k=i-w}^{i+w} (x_k - \mu_w[i])(x_k - \mu_w[i])^T$$

As defined, local statistics yield values for each point x_i in the feature sequence that depend only on the feature points in a subinterval $W_i = [i - w : i + w]$ centered around i. They have the following properties:

- The local mean is the empirical mean of the multivariate probability distribution that generates the feature points in W_i.
- The local covariance matrix is symmetric, positive semidefinite and measures the covariance between feature coordinates in W_i.

$Cov_w[i]$ can be diagonalized by means of an eigenvalue decomposition, yielding eigenvalues $\lambda_1 \geq \ldots \geq \lambda_d$ and associated eigenvectors $v_1, \ldots, v_d$. For a covariance matrix, this corresponds to a **principal component analysis**:

The eigenvector v_1 corresponding to the largest eigenvalue λ_1 indicates the first **principal axis** of the windowed feature set W_i, meaning that a projection of the feature points in W_i onto the subspace spanned by v_1 maximizes the variance λ_1 along this projection with respect to all possible choices of spanning vectors. The same holds for v_2 when restricted to be orthogonal to v_1. This way, we obtain an orthonormal basis of $\mathbb{R}^d$ for every feature point x_i. Now, the first eigenvector/value pair (v_1, λ_1) associated to x_i has two interesting properties that help us in detecting segment boundaries:

1. Under the homogeneity assumption, all variances along principal axes and especially λ_1 should be small when $W_i \subset I(S)$ for some segment S, since homogeneity limits the variation of features within $I(S)$. This situation is depicted in the left part of Figure 5.4,
2. When i is a time index with distance w to a segment boundary between segments S_1 and S_2, W_i is no longer homogeneous and reaches its maximal inhomogeneity when i is exactly on the segment boundary. Assuming that the distance between S_1 and S_2 is significantly larger than the intra-cluster distances of S_1 and S_2, the first principal axis will point to the direction of highest variance, cf. Figure 5.4, which equals to the direction of the line segment between the centroids of $W_i \cap I(S_1)$ and $W_i \cap I(S_2)$ whose length depends on λ_1.

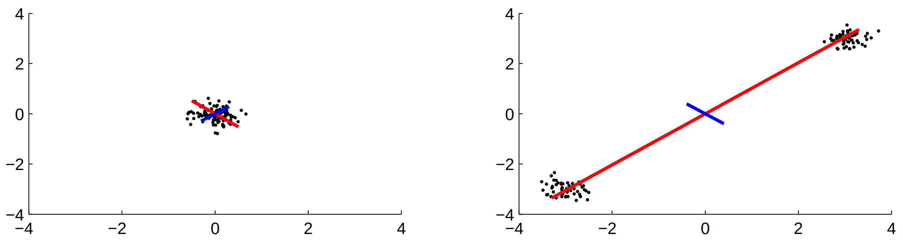


Figure 5.4: **(left)** PCA for Gaussian samples with low variance (red, blue lines correspond to first, second eigenvector, scaled according to corresponding eigenvalue), **(right)** PCA for samples drawn from two Gaussians with low variance each and high distance between means. Lengths are not to scale.

Now, we can plot the value $\lambda_1[i]$ over time and call the resulting novelty function f_{CF} (where CF stands for **C**ovariance **F**irst Eigenvector). This can be done in linear time. However, the runtime of an eigendecomposition of a $(2w+1)\times(2w+1)$ matrix is hidden in the asymptotic notation. For better performance, we upper bound the first eigenvalue by the sum of all eigenvalues, which is equal to the trace of the local covariance matrix and therefore easy to compute:

$$\sum_{k=1}^{2w+1} \lambda_k = tr(\Lambda) = tr(Q\Lambda Q^T) = tr(Cov_w[i])$$

Here, Λ corresponds to a diagonal matrix with $\Lambda_{i,i} = \lambda_i$. In the second equation, we used the invariance of the trace under multiplication with orthogonal matrices. For the third equation, we consider the eigendecomposition of $Cov_w[i]$. Finally, we define the novelty function $f_{CT}(i) = tr(Cov_w[i])$ (where CT stands for **C**ovariance **T**race).

5.2.3 Segmentation with Novelty Functions

In a straightforward way, a novelty function f can be used as a means to segment an audio file: Just compute the local maxima of f over the audio file and set these points in time as potential boundaries. However, the computation of these maxima is not easy in the presence of noise.

While more sophisticated approaches for detecting peaks in novelty functions exist, we restrict ourselves to a simple one: For a given novelty function f and a threshold value $c \in \mathbb{R}$, the set $N_c(f)$ of points in time with high novelty is defined.

$$N_c(f) = \{x \in A \mid f(x) \geq c\}$$

Let us consider the set of points where the sign of the discrete derivative f' changes in a subinterval of radius t from a positive to a negative sign, and intersect this set with $N_c(f)$. This way, we obtain the set of boundaries B:

$$B = N_c(f) \cap \{x \in A \mid (\forall y \in [x-t : x] : f'(y) \geq 0) \;\wedge\; (\forall y \in [x+1 : x+t] : f'(y) \leq 0)\}$$

5.2.4 Evaluation of Novelty Functions

As required, a novelty function should attain high values near segment boundaries in the human segmentation. Furthermore, the values at points in time that don't belong to a segment boundary should be low. While this informal definition of the requirements to a novelty function could be transformed to a formal evaluation score by introducing a measure that measures the peak-likeness of the novelty function in the proximity of boundaries in a human segmentation, we rather reformulate the boundary detection task as a two-class classification problem and choose other scores originating from machine learning, as we will reuse these scores in Chapter 6.

By thresholding the novelty function, possible candidates for segment boundaries have been extracted from the function in the last subsection, yielding a candidate segmentation $\mathcal{S}_A$. Interpreting the boundary detection task as a classification problem in the sense that the ensemble of novelty function and segmentation method is considered as a classifier for the label set *'boundary'* or *'no boundary'* over the audio file, we can use standard measures from machine learning for evaluation, always comparing the boundary set of a candidate segmentation $\mathcal{S}_A$ against that of a reference segmentation $\mathcal{S}_H$ produced by a human.

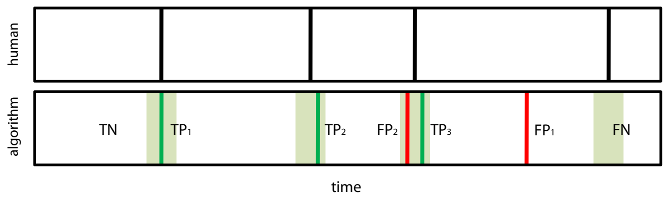


Figure 5.5: **(top)** Example reference segmentation with black boundaries provided by human, **(bottom)** candidate segmentation with tolerance regions in light green. Occurences of true negatives (TN), three kinds of (green) true positives (TP), two kinds of (red) false positives (FP) and false negatives (FN) are indicated by vertical lines.

In this evaluation setting, as shown in Figure 5.5, where a human reference segmentation is compared with a candidate segmentation below, at most one of the following situations can occur at a given point in time x:

- $x \in B(\mathcal{S}_A)$ and $x \in B(\mathcal{S}_H)$, rendering x a **true positive**, like the green line TP_1.
- $x \in B(\mathcal{S}_A)$ and $x \notin B(\mathcal{S}_H)$, rendering x a **false positive**, like the red line FP_1.
- $x \notin B(\mathcal{S}_A)$ and $x \in B(\mathcal{S}_H)$, rendering x a **false negative**, like the 'line' FN.
- $x \notin B(\mathcal{S}_A)$ and $x \notin B(\mathcal{S}_H)$, rendering x a **true negative**, like the 'line' TN.

Up to now, the set of **true positives** is the set $TP = B(\mathcal{S}_H) \cap B(\mathcal{S}_A)$. However, as we don't want to discard unexact candidate boundaries in immediate temporal proximity to a reference boundary, we relax the notion of a true positive by introducing a tolerance region of ± 3 seconds around each reference segment boundary. In Figure 5.5 tolerance regions are shown in light green. We consider a candidate boundary a true positive if it lies within such a tolerance region, like the green lines TP_2 and TP_3 in this figure. In every tolerance region, at most one true positive can be found, all other positives are considered as false positives, like the red line FP_2. False negatives and true negatives are not counted as the number of non-boundary points is much higher than the number of positives, since boundaries are distributed sparsely over an audio file. Now, the **recall** value R (also referred to as **true positive rate**), the **precision** value P, and the **F-measure** F are defined as follows.

$$R = \frac{|TP|}{|B(\mathcal{S}_H)|} \qquad\qquad P = \frac{|TP|}{|B(\mathcal{S}_A)|} \qquad\qquad F = 2\frac{P \cdot R}{P + R}$$

For every novelty measure and each song in a test set described in Chapter 6, we computed the threshold value that yields a segmentation maximizing the F-measure. This maximal F-measure is listed in the left part of Table 5.1. In this evaluation we fixed the kernel sizes for novelty kernels to 3 seconds.

Furthermore, we can also compare two different novelty functions f_1, f_2 directly by just averaging their L_1-distance over the audio file $A = [0, T)$, yielding the denominator in the following fraction. To ensure comparability, we divide this value by the average novelty in both novelty curves. In total, $D(f_1, f_2)$ now measures the average L_1-distance between f_1 and f_2 divided by the mean novelty, where the mean is taken over f_1 and f_2:

$$D(f_1, f_2) = \frac{1}{2} \cdot \frac{\int\limits_A |f_1(x) - f_2(x)|\, dx}{\int\limits_A f_1(x)\, dx + \int\limits_A f_2(x)\, dx}$$

SongID	f_{CT}	f_{CF}	f_{GC}	f_{GF}	f_{BC}	f_{BF}	$D(f_{CT}, f_{CF})$	$D(f_{GC}, f_{GF})$	$D(f_{BC}, f_{BF})$
RockAndRollQueen	0.53	0.57	0.57	0.62	0.57	0.53	0.22	0.59	0.53
HighwaySong	0.77	0.74	0.67	0.72	0.73	0.77	0.15	0.59	0.53
ThatsWhatIGet	0.63	0.70	0.64	0.67	0.70	0.63	0.19	0.63	0.60
RWC-G-M01-tr01	0.48	0.49	0.43	0.41	0.42	0.48	0.24	0.63	0.58
Brahms	0.29	0.30	0.31	0.30	0.30	0.29	0.23	0.62	0.57
Gaynor	0.67	0.62	0.64	0.67	0.62	0.67	0.30	0.64	0.63
RM-C003	0.32	0.26	0.31	0.26	0.34	0.32	0.23	0.63	0.59
JazzSuite	0.47	0.59	0.59	0.59	0.63	0.47	0.16	0.63	0.56
Beatles	0.52	0.60	0.25	0.25	0.37	0.59	0.30	0.70	0.73
ZagerEvans	0.57	0.45	0.48	0.48	0.44	0.57	0.23	0.61	0.58
Average	0.53	0.53	0.49	0.50	0.51	0.53	0.23	0.63	0.59

Table 5.1: **left:** F-Measures for optimal segmentations on novelty functions computed from 10 test audio files, **right:** pairwise comparison on audio files according to $D(f_1, f_2)$

However, as substantial problems arise when the compared functions feature different normalizations, we compare only novelty functions that differ in the last letter of their denoting symbol, i.e. we compare only functions derived from checkerboard kernels to their non-checkerboard counterparts. Furthermore, we compare f_{CT} to f_{CF}. The results are given in the right part of Table 5.1.

In this table, we notice that the average F-measure over the set of test songs is numerically equal for the novelty functions f_{CT} and f_{CF}, suggesting that a novelty function based on the local covariance matrix can indeed use the trace as well as the first eigenvalue. Furthermore, the novelty curves in subplot (d) of Figure 5.6 on the next page are almost identical, yielding low values of the distance functional $D(f_{CT}, f_{CF})$. Therefore, f_{CT} and f_{CF} can be regarded as very similar.

When comparing Gaussian checkerboard kernels with box checkerboard kernels according to the F-measure, we conclude that the actual shape of the convolution kernel doesn't play a significant role, as the average F-measures over the test set are nearly equal. We also notice that f_{GC} and f_{BC}, i.e. the blue curves of subplots B and C of Figure 5.6 look very similar, as do the red curves corresponding to f_{GF} and f_{BF}.

Finally, we compare checkerboard kernels against non-checkerboard (flat) kernels. Again, we notice that, with respect to the F-measure, novelty functions computed from checkerboard kernels almost do not differ from the non-checkerboard counterparts, yielding also similar average values for the F-measure. However, the distance functional between such pairs of functions always attains higher values than $D(f_{CT}, f_{CF})$. This observation is also reflected in a visual inspection of the red and blue curves in subplots B and C of Figure 5.6. Note especially that the novelty functions derived from checkerboard kernels are less smooth than their non-checkerboard counterparts.

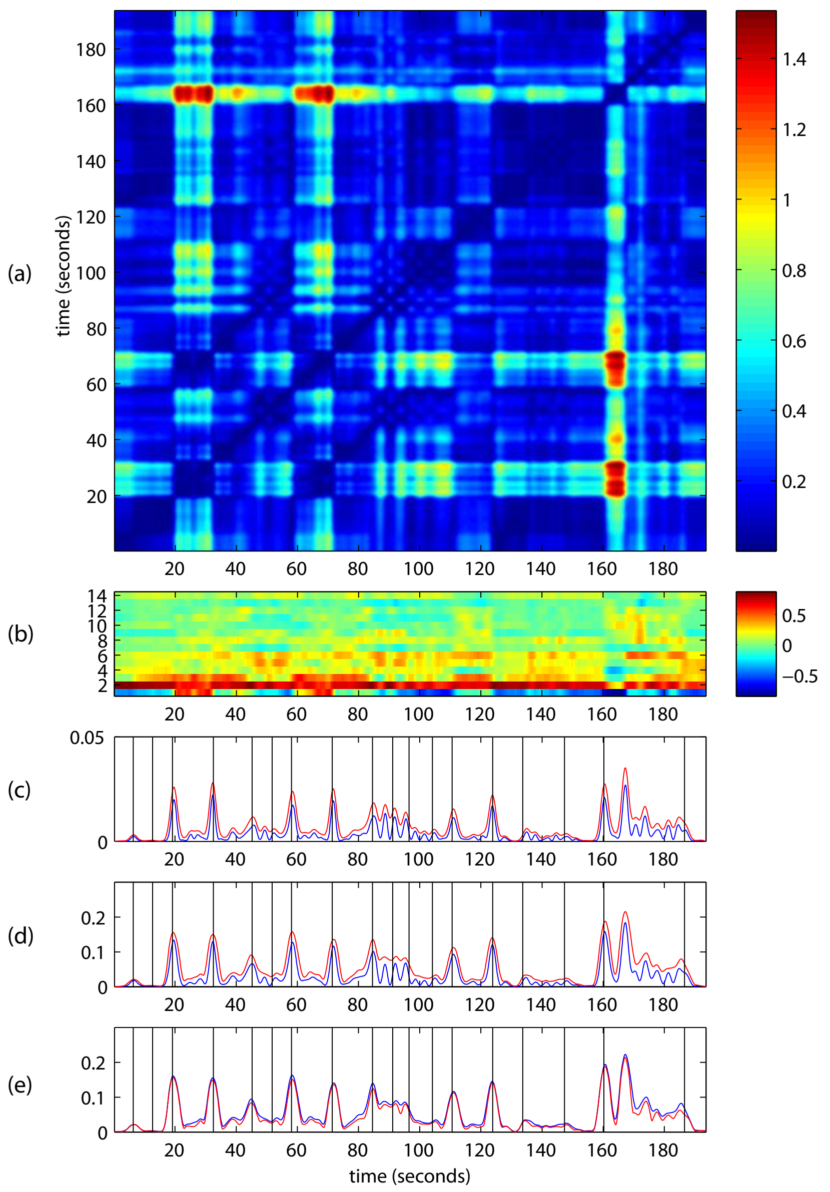


Figure 5.6: Evaluation of novelty functions on *Highway Song* with kernel size fixed to 3 seconds: **(a)** cosine self-dissimilarity matrix computed for (b), **(b)** MFCC feature sequence on song, **(c)** blue: f_{GC} (Gaussian Checkerboard), red: f_{GF} (Gaussian Flat), **(d)** blue: f_{BC} (Box Checkerboard), red: f_{BF} (Box Flat), **(e)** blue: f_{CT} (Covariance Trace), red: f_{CF} (Covariance First Eigenvalue). Black vertical lines correspond to boundaries in the reference segmentation.

5.3 Time-Dependent Agglomerative Clustering

Since the main idea of the approach to time-dependent clustering described in the previous subsection consists of splitting the audio file at time frames that exhibit a high novelty score, this clustering approach can be considered a divisive method. In the following we try to employ the opposite of divisive clustering, namely agglomerative clustering, as described in Chapter 4.2, in a time-dependent setting.

5.3.1 Self-Similarity Manipulation

First, one could process the self-(dis-)similarity matrix by manipulation of blocks near the main diagonal, which corresponds to manipulating the self-similarity of whole time intervals. This can be used in order to increase self-similarity within previously determined intervals and decrease similarities to other parts of the song.

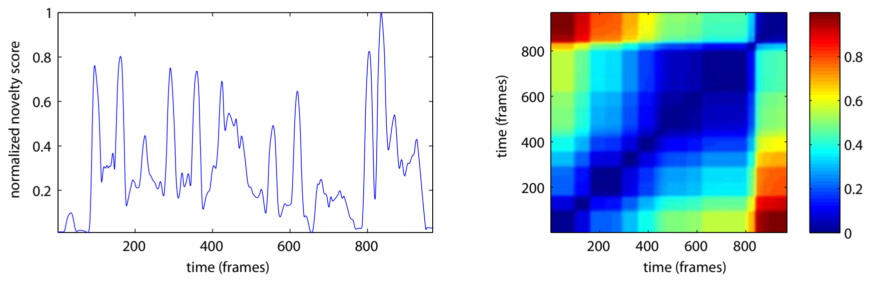


Figure 5.7: **(left)** Normalized novelty curve for *Highway Song*, **(right)** self-dissimilarity matrix D_{nov} computed from the novelty curve.

One could for instance use the novelty function as described in the previous subsection in order to generate a self-dissimilarity matrix D_{nov} that depends completely on a normalized novelty function $nov : [0 : T] \to [0, 1]$ that has been rescaled to attain values only in $[0, 1]$. For each interval $I = [s : t]$ with small $nov(x)$ for all $x \in I$, the submatrix $D_{nov}(I, I)$ then has low self-dissimilarity. Furthermore, if nov reaches peaks near s and t, the submatrix $D_{nov}(I, I)$ is part of a block-like structure surrounded by regions of higher dissimilarity, as visualized in Figure 5.7.

The conversion from a novelty function to a self-dissimilarity matrix can be achieved by setting the dissimilarity of two feature vectors at positions s, t to the area under the novelty curve in the interval $[s : t]$. In the discrete setting, this is formalized as follows, with p being an exponent that allows for attenuating low novelty values and accentuating high values:

$$D_{nov}(i, j) = \sum_{k=j}^{i-1} nov(k)^p + \sum_{k=i+1}^{j} nov(k)^p$$

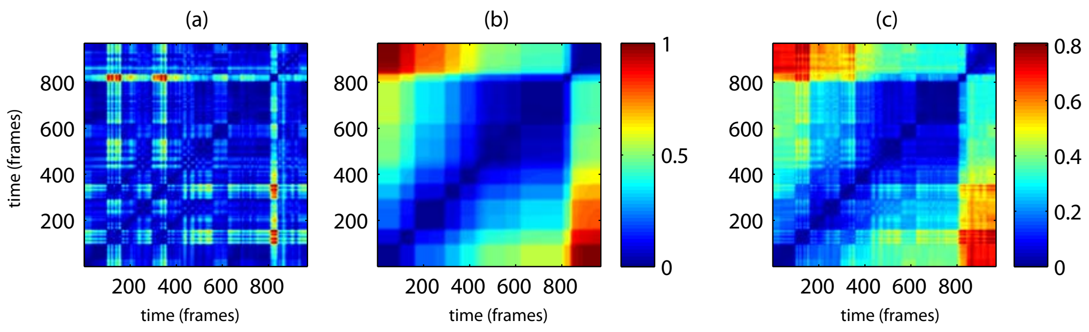


Figure 5.8: **(a)** Self-dissimilarity matrix D_{feat} computed on feature sequence, **(b)** dissimilarity matrix D_{nov} derived from novelty curve, **(c)** combined matrix D_{com} with double weight for D_{nov}.

Theoretically, the matrix D_{nov} could be used for agglomerative clustering on a novelty function. However, this approach is unsatisfying since we could also directly use the novelty function in order to compute a segmentation.

Instead, the self-dissimilarity matrix D_{nov} obtained by the conversion from a novelty function according to the formula above can be combined with a self-dissimilarity matrix D_{feat} obtained by comparing feature vectors with respect to a dissimilarity measure in the usual way. Formally, this corresponds to a pointwise convex combination of D_{nov} and D_{feat} with weight factors α_{nov} and α_{feat}, yielding D_{com}:

$$D_{com}(i,j) = \alpha_{nov} D_{nov}(i,j) + \alpha_{feat} D_{feat}(i,j)$$

An example for a combined self-dissimilarity matrix can be seen in Figure 5.8, where D_{nov} corresponds to the matrix plotted in Figure 5.7 and D_{feat} is computed from MFCC features on the *Highway Song* with respect to cosine dissimilarity. Here, the values $\alpha_{nov} = 2$ and $\alpha_{feat} = 1$ have been chosen to assign double weight to D_{nov}. As one can notice in the figure, the dissimilarity between intervals that have high temporal distance grows in the number of peaks of the novelty function spanned by the temporal distance.

The combined self-dissimilarity matrix D_{com} can be clustered by the agglomerative clustering algorithm described in Section 4.2. At this point, the previously discussed fact that this algorithm depends only on the self-dissimilarity matrix turns out to be very useful since this allows clustering a manipulated self-dissimilarity matrix in the first place. Compare this e.g. to the situation in the K-means algorithm, where the input sequence has to consist of features embedded in some vector space allowing for computing centroids. It is not clear how to transform D_{com} and especially D_{nov} back to such a vector representation.

However, our experience shows that this approach does not yield significant improvements to music segmentation, possibly due to the dependence on the weight factors α_{nov} and α_{feat}. Therefore, we present another method for modifying agglomerative clustering.

5.3.2 Time-Restricted Agglomerative Clustering

The second possibility to introduce temporal coherence to agglomerative clustering is a modification of the hierarchical clustering algorithm itself. While hierarchical clustering in general merges any two clusters with lowest inter-dissimilarity, we can restrict it to merge only adjacent clusters, yielding the following algorithm that is described with the same notation as in Subsection 4.2.2:

1. Given an audio file $B = [1 : N]$, set $C_i = \{i\}$ for all $i \in B$ and $R \leftarrow \{C_1, \dots, C_N\}$.
2. For all $i, j \in 1, \dots, N$, set $A(C_i, C_j) \leftarrow d(x_i, x_j)$.
3. While $A(C_a, C_{a+1}) < t$ for C_a, C_{a+1} that minimize $A(C_a, C_{a+1})$, repeat 4 and 5:
4. Set $C_a \leftarrow C_a \cup C_{a+1}$, delete C_{a+1} from R and the corresp. row and col. from A.
5. For all $C_i \in R$, set $A(C_a, C_i) \leftarrow \frac{1}{|C_a| \cdot |C_i|} \sum\limits_{k \in C_a} \sum\limits_{l \in C_i} d(x_k, x_l)$.
   Then, for all $C_i \in R$, set $A(C_i, C_a) \leftarrow A(C_a, C_i)$.

In the first step, a singleton cluster C_i is created for every time index i. The ensemble of all clusters is denoted by R. The value $A(C_i, C_j)$ denotes the inter-cluster dissimilarity of clusters C_i and C_j and in step 2, it is initialized to the corresponding value $d(x_i, x_j)$ of the feature dissimilarity. Up to this point, this algorithm works in exactly the same way as agglomerative clustering in Section 4.2.

However, when checking the loop condition in step 3, we restrict the search for the next pair of clusters that should be merged to adjacent clusters. This way, every cluster obtained by this clustering method is a connected subinterval of the domain of the input audio file, opposed to general agglomerative clustering.

In step 4, we merge the pair of adjacent clusters determined in step 3 to one single cluster and delete the surplus rows and columns from A. This resulting cluster is guaranteed to be connected, since it is the union of two adjacent connected clusters. In step 5, we update the value of A to the new inter-cluster distances from and to C_a. In our implementation, this step doesn't involve the double sum described above, which would lead to an unnecessary computational overhead. Instead, we exploit the fact that previous inter-cluster distances have already been computed and can be reused for the computation of the new inter-cluster distances by means of dynamic programming. Our implemented update steps 4 and 5 look as follows:

4. For all $C_i \in R$: $A(C_a, C_i) \leftarrow \frac{|C_a| A(C_a, C_i) + |C_{a+1}| A(C_{a+1}, C_i)}{|C_a| + |C_{a+1}|}$.
   Then, for all $C_i \in R$: $A(C_i, C_a) \leftarrow A(C_a, C_i)$.
5. Set $C_a \leftarrow C_a \cup C_{a+1}$, delete C_{a+1} from R and the corresp. row and col. from A.

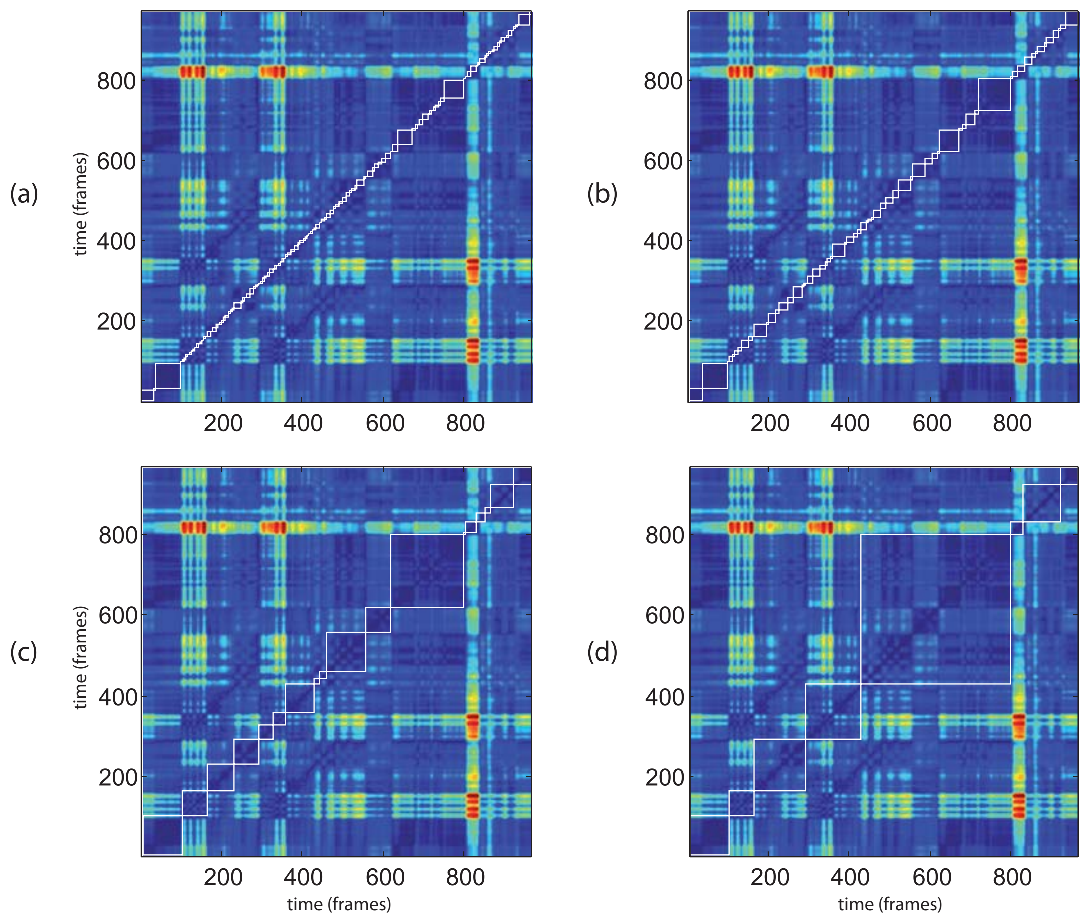


Figure 5.9: Results of time-restricted agglomerative clustering for *Highway Song*, shown as white squares in the self-dissimilarity matrix that considered to be homogeneous by the algorithm after: **(a)** 800 iterations, **(b)** 900 iterations, **(c)** 950 iterations, **(d)** 960 iterations.

An execution of the algorithm is demonstrated in Figure 5.9. Each subfigure shows the self-dissimilarity matrix of MFCC features on the *Highway Song* along with the segmentation obtained by the time-restricted agglomerative clustering algorithm after a number of iterations. When projecting all white squares to the x- or y-axis, the corner points correspond to segment boundaries. In this setting we did not specify a threshold t but instead let the algorithm run until the segmentation consists of one trivial cluster.

As expected, the squares delimit submatrices that feature a low mean dissimilarity value, which correspond to homogeneous segments in the feature sequence. Note that the algorithm first creates small clusters before creating larger squares. This behaviour is due to the fact that feature smoothness implies a low dissimilarity value in direct proximity of the main diagonal of the dissimilarity matrix. Therefore, clusters will try to stay within this proximity as long as possible.

6 Evaluation Results

6.1 Detailed Evaluation

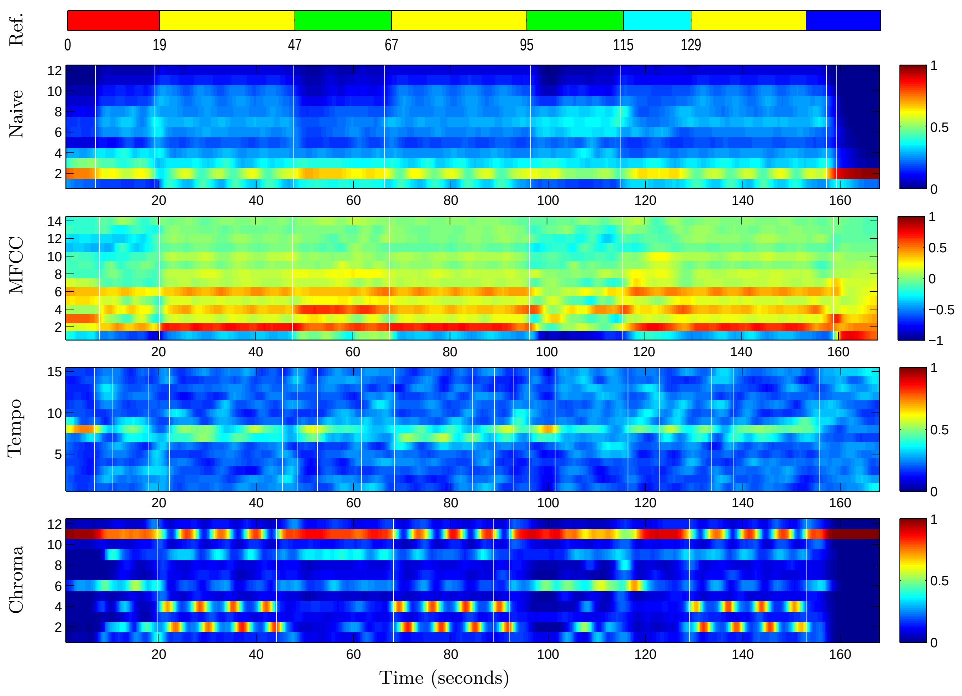
In this section, we present detailed evaluation results for a selected set of 9 test audio files. While the first songs belong to the genres of popular or rock music, and we therefore expect good evaluation results, some classical pieces are also part of the test set, on which we expect our methods to fail.

In this evaluation setup, every song receives a double page: Every left side starts with the human reference segmentation provided by the author, followed by a batch-generated set of plots depicting feature sequences computed from the track. This set consists of four plots corresponding to naive spectrum features, MFCC features, cyclic Fourier-based tempo features and chroma features. Every feature sequence is computed with a feature rate of 5 Hz and has been smoothed with a Gaussian kernel of 5 seconds length.

The second half of the left page is devoted to numerical performance results. Each algorithm presented in the previous chapters is executed with different choices of parameters on all four feature sequences. Then the precision (P) and recall (R) values for segment boundary detection are listed, along with the F-measure. Note that in this setup we evaluate only the boundaries of the segmentation. The highest F-measure in each column is printed in bold and the algorithm and parameter choice yielding this optimal segmentation can be read off the very left column. Furthermore, the boundaries of the optimal segmentation for every feature sequence are plotted in white lines over the corresponding feature sequence. For the K-means algorithm, we vary the fixed number K of clusters. To facilitate comparison, we don't use a threshold value for clustering dendrograms in agglomerative clustering *(Aggl.)*, but instead cut the dendrogram such that a fixed number K of clusters remain. Similarly, we specify the number of segments in time-restricted agglomerative clustering *(AgTime)*. Furthermore, feature evaluation values, as defined in Chapter 3, are listed in the table at the bottom of every left page.

On the right page of an evaluation double page, we discuss the specific results, point out interesting facts and set the evaluation data on the left page into a musical context.

6.1.1 The Subways - Rock and Roll Queen



	Naive			MFCC			Rhythm			Chroma		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
K-Means, K = 2	0.00	0.00	0.00	1.00	0.43	0.60	0.22	0.71	0.33	0.17	0.57	0.26
K-Means, K = 3	0.43	0.43	0.43	0.75	0.43	0.55	0.18	0.86	0.29	0.17	0.71	0.27
K-Means, K = 4	0.63	0.71	**0.67**	0.60	0.43	0.50	0.14	0.86	0.24	0.12	0.71	0.20
K-Means, K = 5	0.23	1.00	0.37	0.71	0.71	**0.71**	0.13	0.86	0.22	0.13	1.00	0.23
K-Means, K = 6	0.23	1.00	0.37	0.56	0.71	0.63	0.13	1.00	0.24	0.09	0.71	0.16
Aggl., K = 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.17	0.57	0.26
Aggl., K = 3	0.00	0.00	0.00	0.50	0.29	0.36	0.13	0.29	0.18	0.11	0.57	0.19
Aggl., K = 4	0.00	0.00	0.00	0.40	0.29	0.33	0.19	0.57	0.29	0.11	0.57	0.18
Aggl., K = 5	0.20	0.14	0.17	0.33	0.29	0.31	0.19	0.57	0.29	0.12	0.71	0.21
Aggl., K = 6	0.43	0.43	0.43	0.25	0.29	0.27	0.17	0.86	0.29	0.09	0.71	0.17
Ag.Time, K = 5	0.00	0.00	0.00	0.50	0.29	0.36	0.00	0.00	0.00	0.75	0.43	0.55
Ag.Time, K = 8	0.43	0.43	0.43	0.43	0.43	0.43	0.14	0.14	0.14	0.57	0.57	**0.57**
Ag.Time, K = 11	0.50	0.71	0.59	0.30	0.43	0.35	0.20	0.29	0.24	0.40	0.57	0.47
Ag.Time, K = 16	0.40	0.86	0.55	0.33	0.71	0.45	0.27	0.57	0.36	0.27	0.57	0.36
Ag.Time, K = 20	0.32	0.86	0.46	0.26	0.71	0.38	0.26	0.71	**0.38**	0.21	0.57	0.31

Evaluation of algorithms on all features w.r.t. precision (P), recall (R) and F-measure (F)

	Naive			MFCC			Rhythm			Chroma		
D^{inter}, D^{intra}, D	0.39	0.22	1.76	0.84	0.52	1.60	0.33	0.26	1.28	0.56	0.37	1.52
D^{inter}, D^{intra}_{max}, D_{max}	0.39	0.42	0.93	0.84	0.76	1.10	0.33	0.31	1.07	0.56	0.53	1.07
Certitude for $\sigma = 0.2, 0.05, 0.01$	0.25	0.35	0.61	0.39	0.68	0.94	0.22	0.27	0.52	0.32	0.50	0.74
Label-Dep. Cert. for $\sigma = 0.2, 0.05, 0.01$	0.24	0.34	0.60	0.38	0.67	0.93	0.21	0.26	0.51	0.30	0.47	0.72

Dissimilarity-based and classification-based evaluation results

The rock song *Rock and Roll Queen*, performed by *The Subways*, has a very simple structure consisting of an introduction part and alternating chorus and verse segments that are interrupted by an interlude. Altogether, five different segment labels are present in the human segmentation, listed in order of first occurence:

intro (red), **chorus** (yellow), **verse** (green), **interlude** (cyan) and **ending** (blue).

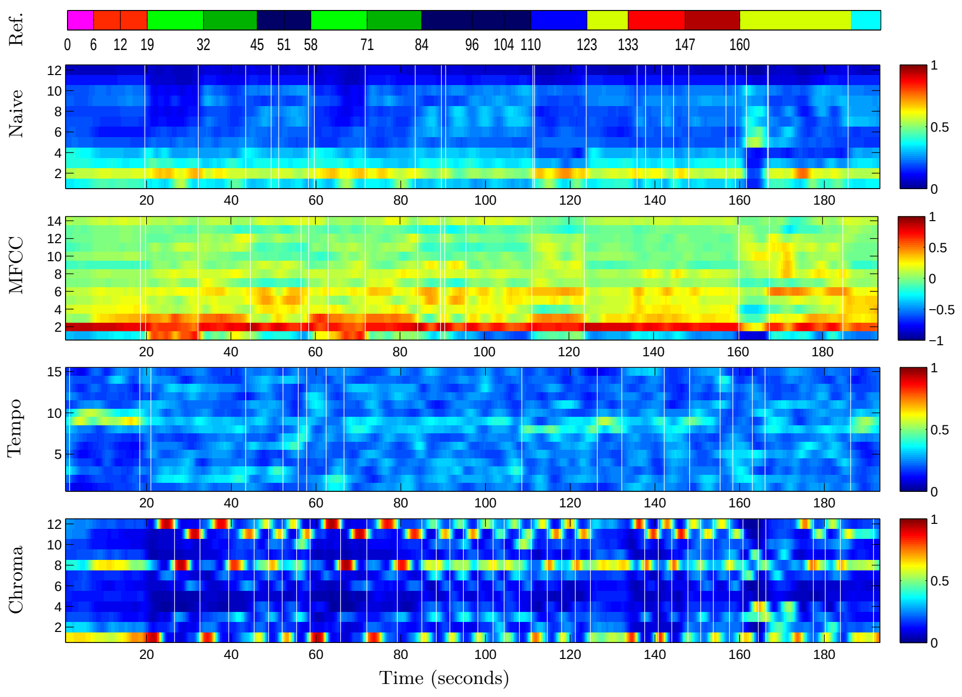
This structure is reflected mainly in timbral properties of the song, i.e. during verse parts, the electric guitar is played muted, while during chorus parts, distortion sets in. For this particular song, to some extent even harmonical homogeneity is given, since during the verse and intro parts only one single chord, namely a B^b-powerchord, is played. Especially in the intro part, this chord can be seen clearly in the high values for components 6 and 11 (note that $11+7 \equiv 6 (\text{mod } 12)$). However, the chorus sections in this song consist of three alternated chords, namely E^b, D^b and B^b, making them inhomogeneous with respect to Chroma features. As the song features almost no variation in tempo, the tempo features fail to provide information that might be useful for segmentation.

As one can see in the first and second feature sequence, both naive spectrum features and MFCC features reflect the timbral structure described above. This visual impression is also reflected in the feature evaluation scores, with naive spectrum features and MFCC features yielding optimal values for D.

In particular, the best segmentations with respect to naive spectrum features and MFCC features don't differ significantly, except for the boundary at 160 seconds, which is doubled in the segmentation obtained for naive spectrum features. This boundary introduces a false positive which decreases the precision value for this segmentation compared to the optimal segmentation for MFCC features. Furthermore, the optimal F-measures for naive spectrum features and MFCC features don't differ significantly. Finally, except for the segmentation based on chroma features, every segmentation detects a false positive at about 8 seconds where a vocal part sets in.

Interestingly, chroma features succeed in finding some tricky segment boundaries. For instance, robustness of chroma features under the timbral change at 8 seconds is a wanted effect in this case and suggests further investigation of possible combinations of features. Furthermore the segment boundary at 129 seconds, that is hidden in timbral features, is also recovered in the segmentation on chroma features.

6.1.2 System of a Down - Highway Song



	Naive			MFCC			Rhythm			Chroma		
	P	R	F	P	R	F	P	R	F	P	R	F
K-Means, K = 2	0.50	0.06	0.10	0.88	0.39	0.54	0.38	0.17	0.23	0.34	0.72	**0.46**
K-Means, K = 3	0.41	0.50	0.45	0.50	0.50	0.50	0.30	0.44	0.36	0.27	0.83	0.41
K-Means, K = 4	0.48	0.56	0.51	0.53	0.56	0.54	0.29	0.67	0.41	0.21	0.83	0.34
K-Means, K = 5	0.52	0.72	**0.60**	0.52	0.61	0.56	0.35	0.67	0.46	0.21	0.83	0.33
K-Means, K = 6	0.42	0.78	0.55	0.71	0.67	**0.69**	0.33	0.72	0.45	0.20	0.89	0.33
Aggl., K = 2	0.50	0.06	0.10	0.50	0.06	0.10	0.40	0.11	0.17	0.00	0.00	0.00
Aggl., K = 3	0.25	0.06	0.09	0.83	0.28	0.42	0.43	0.17	0.24	0.14	0.22	0.17
Aggl., K = 4	0.63	0.28	0.38	0.71	0.28	0.40	0.44	0.22	0.30	0.24	0.83	0.37
Aggl., K = 5	0.73	0.44	0.55	0.67	0.56	0.61	0.40	0.22	0.29	0.22	0.83	0.35
Aggl., K = 6	0.62	0.44	0.52	0.65	0.61	0.63	0.33	0.22	0.27	0.22	0.83	0.34
Ag.Time, K = 5	0.25	0.06	0.09	0.75	0.17	0.27	0.75	0.17	0.27	0.00	0.00	0.00
Ag.Time, K = 8	0.14	0.06	0.08	0.86	0.33	0.48	0.57	0.22	0.32	0.00	0.00	0.00
Ag.Time, K = 11	0.40	0.22	0.29	0.90	0.50	0.64	0.50	0.28	0.36	0.20	0.11	0.14
Ag.Time, K = 16	0.47	0.39	0.42	0.73	0.61	0.67	0.47	0.39	0.42	0.20	0.17	0.18
Ag.Time, K = 20	0.42	0.44	0.43	0.58	0.61	0.59	0.47	0.50	**0.49**	0.26	0.28	0.27

Evaluation of algorithms on all features w.r.t. precision (P), recall (R) and F-measure (F)

	Naive			MFCC			Rhythm			Chroma		
D^{inter}, D^{intra}, D	0.22	0.11	1.99	0.63	0.26	2.40	0.33	0.21	1.58	0.60	0.44	1.37
D^{inter}, D^{intra}_{max}, D_{max}	0.22	0.30	0.71	0.63	0.63	1.00	0.33	0.26	1.29	0.60	0.83	0.73
Certitude for $\sigma = 0.2, 0.05, 0.01$	0.11	0.15	0.32	0.20	0.47	0.88	0.11	0.16	0.39	0.15	0.30	0.69
Label-Dep. Cert. for $\sigma = 0.2, 0.05, 0.01$	0.11	0.14	0.29	0.19	0.44	0.86	0.11	0.15	0.38	0.12	0.25	0.66

Dissimilarity-based and classification-based evaluation results

The *Highway Song*, performed by the band *System of a Down*, belongs to the genre of Metal and represents the second song in the set of well-tempered songs we are considering in this evaluation. The increase in the track number comes with an increased complexity of the human reference segmentation, featuring 10 labels:

intro A (pink), **intro B** (red), **verse A** (light green), **verse B** (dark green), **bridge** (dark blue), **interlude** (light blue), **pre-chorus** (lime), **chorus A** (red), **chorus B** (dark red) and **ending** (cyan).

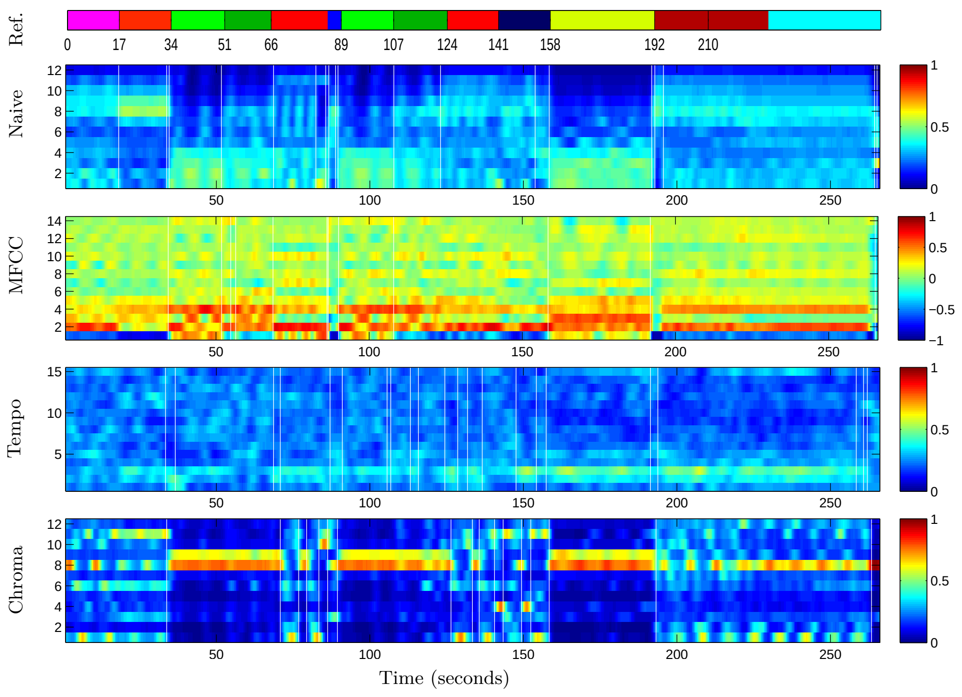
The song features significant changes in timbral and dynamical properties across segment boundaries. Therefore, the best segmentations are again obtained for timbral features with F-measures of 0.6 and 0.69. No optimal segmentation recognizes the segment boundaries between the intro parts. However, at least the boundary at 6 seconds stands out clearly for the human ear. Note that MFCC, naive, and chroma features capture this boundary, while rhythm features fail to do so.

Furthermore, the feature evaluation scores indicate that MFCC features are the optimal features on this song, as given by $D = 2.4$. Note that, with respect to D_{max}, rhythm features show better results due to the overall homogeneity of this feature sequence in reference segments, yielding a value of D_{max}^{intra} nearly equal D^{intra} for rhythm features.

Again, we observe that the repetitive structure of the song is represented best in the Chroma feature sequence. Homogeneous regions in this sequence correspond to chords played by bass and electric guitar and the sequence of homogeneous regions reflects the sequence of chords each segment consists of. When clustering according to homogeneity, this Chroma sequence is of no direct use, as can be seen in the poor F-measure.

Note however, that regardless of the segment number K chosen, segmentations based on chroma features yield the highest recall values among all feature sequences for this song. The low F-measure is explained solely by the poor precision values. Now, this combination of values has a direct musical interpretation: In many cases, a segment boundary is also characterized by a harmonical change. Therefore, harmonical changes, which can be detected by chroma-based segmentation methods, are likely to contain most of the boundaries in the human reference segmentation. The other direction does not hold: Not every harmonical change implies a segment boundary, as can be seen in the first verse part consisting of three different chords. Here, every second harmonical change is recognized as a segment boundary, yielding a precision value of roughly $1/3$.

6.1.3 Nine Inch Nails - That's What I Get



	Naive			MFCC			Rhythm			Chroma		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
K-Means, K = 2	0.42	0.62	0.50	0.62	0.62	0.62	0.36	0.62	**0.46**	0.22	0.46	0.30
K-Means, K = 3	0.29	0.69	0.41	0.43	0.69	0.53	0.21	0.69	0.33	0.23	0.62	0.33
K-Means, K = 4	0.23	0.69	0.35	0.32	0.69	0.44	0.18	0.62	0.28	0.14	0.54	0.23
K-Means, K = 5	0.18	0.69	0.29	0.26	0.69	0.37	0.15	0.62	0.24	0.15	0.77	0.25
K-Means, K = 6	0.18	0.85	0.30	0.26	0.77	0.38	0.13	0.62	0.22	0.14	0.77	0.24
Aggl., K = 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.45	0.38	0.42
Aggl., K = 3	0.53	0.62	0.57	0.33	0.08	0.13	0.22	0.38	0.28	0.46	0.46	0.46
Aggl., K = 4	0.47	0.69	0.56	0.67	0.62	**0.64**	0.22	0.38	0.28	0.41	0.54	**0.47**
Aggl., K = 5	0.47	0.69	0.56	0.50	0.62	0.55	0.23	0.46	0.31	0.41	0.54	0.47
Aggl., K = 6	0.45	0.69	0.55	0.36	0.62	0.46	0.22	0.54	0.31	0.24	0.62	0.34
Ag.Time, K = 5	0.75	0.23	0.35	0.75	0.23	0.35	0.00	0.00	0.00	0.50	0.15	0.24
Ag.Time, K = 8	0.86	0.46	0.60	0.86	0.46	0.60	0.29	0.15	0.20	0.43	0.23	0.30
Ag.Time, K = 11	0.70	0.54	0.61	0.70	0.54	0.61	0.40	0.31	0.35	0.40	0.31	0.35
Ag.Time, K = 16	0.53	0.62	0.57	0.53	0.62	0.57	0.40	0.46	0.43	0.40	0.46	0.43
Ag.Time, K = 20	0.53	0.77	**0.63**	0.42	0.62	0.50	0.32	0.46	0.37	0.32	0.46	0.37

Evaluation of algorithms on all features w.r.t. precision (P), recall (R) and F-measure (F)

	Naive			MFCC			Rhythm			Chroma		
D^{inter}, D^{intra}, D	0.36	0.17	2.17	0.88	0.47	1.88	0.26	0.18	1.43	0.64	0.32	1.98
D^{inter}, D^{intra}_{max}, D_{max}	0.36	0.24	1.51	0.88	0.82	1.08	0.26	0.23	1.16	0.64	0.57	1.13
Certitude for $\sigma = 0.2, 0.05, 0.01$	0.13	0.21	0.45	0.27	0.60	0.92	0.11	0.14	0.30	0.18	0.34	0.66
Label-Dep. Cert. for $\sigma = 0.2, 0.05, 0.01$	0.13	0.20	0.43	0.26	0.58	0.92	0.11	0.13	0.29	0.17	0.32	0.64

Dissimilarity-based and classification-based evaluation results

The Industrial song *That's What I Get*, performed by the band *Nine Inch Nails* is an example for a song featuring very clear segment boundaries, but also extreme dynamical and timbral variation which can also occur within a segment. The human segmentation to this song again consists of 10 labels:

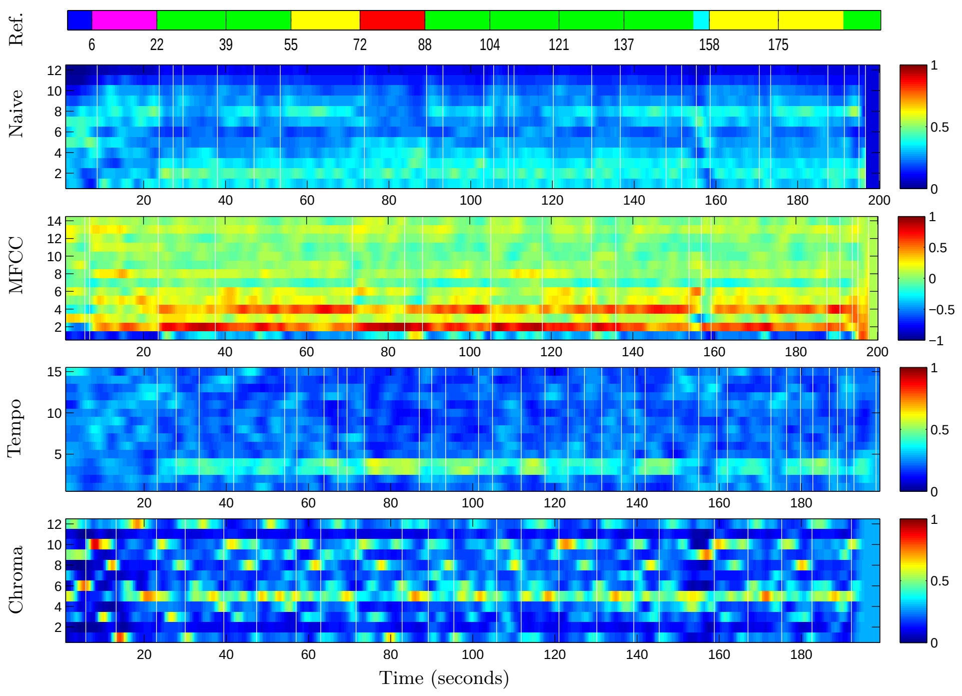
intro A (pink), **intro B** (red), **verse A** (light green), **verse B** (dark green), **chorus A** (red), **interlude** (blue), **bridge** (dark blue), **pad solo** (lime), **chorus B** (dark red), **ending** (cyan)

In our test playlist, this song is the first one to feature synthesizer sounds, for instance pads, i.e. ambient background instruments that are mostly created using saw-like waveforms subject to bandpass filters that restrict the frequency range of the instrument to mid-frequencies. For instance, apart from quiet rhythmical sounds, a pad is the only instrument present in the pad solo part. Furthermore, synthesizers are also used as rhythmical instruments in this song. These instruments characterized by sharp sounds featuring many overtones. Apart from drums, including a heavily distorted snare drum with long reverb, also vocal parts are present in this song, mainly during the verse, chorus and interlude parts.

Again, timbre features yield the best segmentations. As the feature sequence computed for naive spectrum features creates a nearly perfect block-like structure, the best segmentation on this sequence is obtained for time-restricted agglomerative clustering, yielding the near-optimal F-measure of 0.63, which however is worse than expected. This relatively poor value is explained by the fact that time-restricted agglomerative clustering exhibits some problems at the segment boundaries at seconds 34, 89 and 192. At these points in time, the segmentation contains several bounds that should be subsumed to a single bound. Of course, the recall value doesn't suffer from this problem and therefore attains a high value of 0.77, which means that 10 out of 13 bounds have been retrieved successfully. As naive spectrum features recognize the overtones of the synthesizer setting in at second 17, they succeed in finding this bound, as compared to the other features.

Note that on this song, naive features and MFCC features yield nearly the same optimal F-measures. However, due to the boundary confusion described above, we still think that naive features in fact perform better on this song. This belief is backed up by the feature evaluation scores D and D_{max}, which show optimal values on naive spectrum features.

6.1.4 Gloria Gaynor - I Will Survive



	Naive			MFCC			Rhythm			Chroma		
	P	R	F	P	R	F	P	R	F	P	R	F
K-Means, K = 2	0.60	0.23	0.33	0.50	0.23	0.32	0.22	0.31	0.26	0.48	0.85	**0.61**
K-Means, K = 3	0.44	0.92	**0.60**	0.36	0.77	0.49	0.29	0.54	0.38	0.31	0.92	0.46
K-Means, K = 4	0.41	0.92	0.57	0.37	0.77	0.50	0.28	0.77	**0.41**	0.21	0.85	0.33
K-Means, K = 5	0.40	0.92	0.56	0.33	0.69	0.45	0.24	0.85	0.37	0.24	0.92	0.38
K-Means, K = 6	0.31	0.92	0.46	0.25	0.85	0.39	0.22	0.77	0.34	0.19	0.92	0.32
Aggl., K = 2	0.67	0.15	0.25	1.00	0.08	0.14	0.50	0.23	0.32	0.50	0.08	0.13
Aggl., K = 3	0.43	0.23	0.30	0.50	0.08	0.13	0.28	0.38	0.32	0.33	0.08	0.13
Aggl., K = 4	0.43	0.23	0.30	0.75	0.23	0.35	0.25	0.38	0.30	0.60	0.23	0.33
Aggl., K = 5	0.50	0.31	0.38	0.67	0.31	0.42	0.28	0.69	0.40	0.33	0.23	0.27
Aggl., K = 6	0.50	0.38	0.43	0.57	0.31	0.40	0.28	0.69	0.40	0.27	0.23	0.25
Ag.Time, K = 5	0.50	0.15	0.24	0.50	0.15	0.24	0.25	0.08	0.12	0.25	0.08	0.12
Ag.Time, K = 8	0.57	0.31	0.40	0.57	0.31	0.40	0.29	0.15	0.20	0.43	0.23	0.30
Ag.Time, K = 11	0.40	0.31	0.35	0.50	0.38	0.43	0.30	0.23	0.26	0.30	0.23	0.26
Ag.Time, K = 16	0.40	0.46	0.43	0.53	0.62	0.57	0.20	0.23	0.21	0.20	0.23	0.21
Ag.Time, K = 20	0.37	0.54	0.44	0.53	0.77	**0.63**	0.21	0.31	0.25	0.26	0.38	0.31

Evaluation of algorithms on all features w.r.t. precision (P), recall (R) and F-measure (F)

	Naive			MFCC			Rhythm			Chroma		
D^{inter}, D^{intra}, D	0.36	0.20	1.80	0.99	0.45	2.18	0.34	0.19	1.74	0.65	0.43	1.52
D^{inter}, D^{intra}_{max}, D_{max}	0.36	0.44	0.82	0.99	0.72	1.37	0.34	0.25	1.37	0.65	0.68	0.95
Certitude for $\sigma = 0.2, 0.05, 0.01$	0.17	0.25	0.46	0.33	0.58	0.86	0.16	0.23	0.44	0.20	0.34	0.68
Label-Dep. Cert. for $\sigma = 0.2, 0.05, 0.01$	0.17	0.24	0.44	0.32	0.54	0.84	0.16	0.21	0.39	0.19	0.31	0.66

Dissimilarity-based and classification-based evaluation results

We now consider the famous disco song *I Will Survive*, performed by *Gloria Gaynor*. This song has a simple structure consisting of 7 segment labels, especially featuring only a single chorus type and a single verse type:

intro A (blue), **intro B** (pink), **verse** (green), **chorus** (yellow), **bridge** (red), **interlude** (cyan) and **ending** (green).

Although the structure of this song is rather simple, it is defined by repetitions. For instance, the verse part occurs in six instances without significant variation of timbre or chroma features. In fact, when neglecting lyrical content and voice intonation, even chorus and verse part can be seen as equal. Except for the interlude and bridge parts, the chroma feature sequence is a simple repetition of the subsequence given by the first verse part.

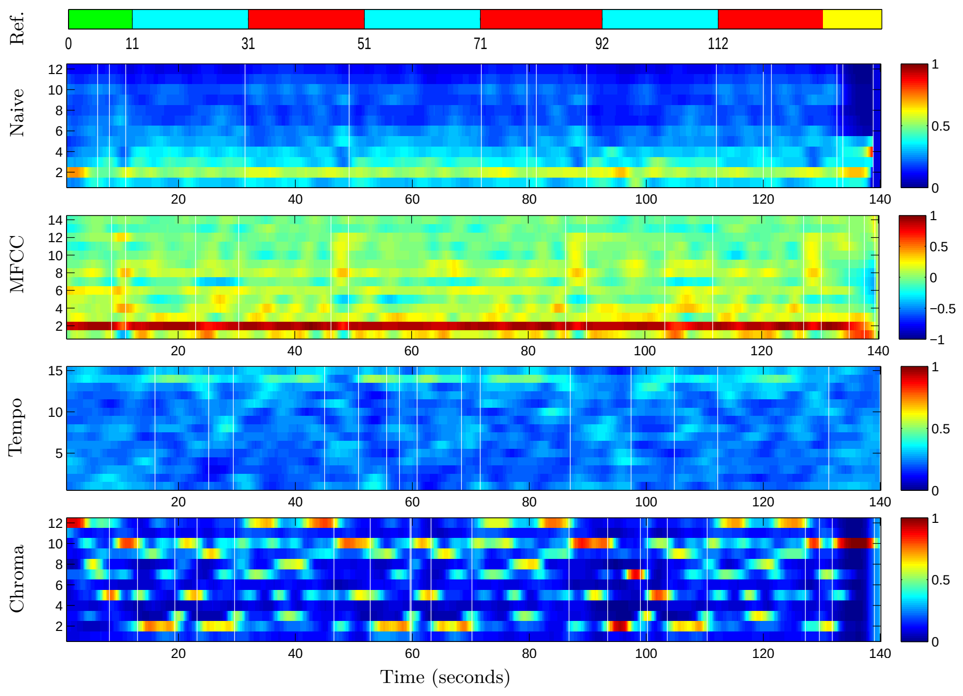
As every such verse part ends with an E major chord, which is succeeded by the A minor chord of the next occurence of a verse part, chroma features yield optimal recall values and - surprisingly - also an optimal F-measure of roughly the same value as for timbre features.

Though rhythm features show the worst results on this song, they can discriminate between the intro part and the subsequent first verse part. This is shown both by visual inspection of the feature sequence and the bound in the optimal segmentation at 22 seconds. The main reason for this result is the fact that during the intro part, no beat is present and therefore, the main beat frequency of about 116 BPM can not be recognized by the tempo features. In fact, at several other points in time, e.g. about 150-160 seconds, the beat breaks off, yielding noisy tempo features. Apart from these exceptions, a steady peak is seen in coefficients 3-4, corresponding to the tempo of 116 BPM.

Again, we see that timbre features, especially MFCC features, yield the optimal F-measures for this song. Furthermore, we notice that the segmentation on MFCC features could be improved by a post-processing step that eliminates extremely short segments. These short segments create accumulations of boundaries, as can be seen at second 6 and in the ending part.

In conclusion, the performance is better than expected, as the homogeneity assumption, the cornerstone of homogeneity-based music segmentation was not expected to be given in this song. However, MFCC features attain a high value of D. This value agrees with the fact that MFCC features yield the highest F-measure on this song.

6.1.5 The Beatles - Help



	Naive			MFCC			Rhythm			Chroma		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
K-Means, K = 2	0.00	0.00	0.00	0.09	0.29	0.13	0.29	0.57	0.38	0.30	0.43	0.35
K-Means, K = 3	0.33	0.71	**0.45**	0.11	0.43	0.18	0.24	0.71	0.36	0.33	1.00	**0.50**
K-Means, K = 4	0.24	0.71	0.36	0.15	0.86	0.26	0.23	0.86	0.36	0.21	1.00	0.35
K-Means, K = 5	0.24	0.86	0.38	0.14	0.86	0.24	0.22	0.86	0.35	0.18	1.00	0.31
K-Means, K = 6	0.23	0.86	0.36	0.14	0.86	0.24	0.21	0.86	0.34	0.16	1.00	0.27
Aggl., K = 2	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.14	0.22	0.19	0.43	0.26
Aggl., K = 3	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.29	0.36	0.14	0.43	0.21
Aggl., K = 4	0.00	0.00	0.00	0.25	0.14	0.18	0.33	0.71	**0.45**	0.13	0.43	0.19
Aggl., K = 5	0.00	0.00	0.00	0.30	0.43	0.35	0.24	0.71	0.36	0.12	0.43	0.19
Aggl., K = 6	0.00	0.00	0.00	0.27	0.43	0.33	0.26	0.86	0.40	0.16	0.86	0.27
Ag.Time, K = 5	0.00	0.00	0.00	0.25	0.14	0.18	0.25	0.14	0.18	0.25	0.14	0.18
Ag.Time, K = 8	0.00	0.00	0.00	0.43	0.43	0.43	0.14	0.14	0.14	0.29	0.29	0.29
Ag.Time, K = 11	0.10	0.14	0.12	0.40	0.57	0.47	0.20	0.29	0.24	0.40	0.57	0.47
Ag.Time, K = 16	0.13	0.29	0.18	0.40	0.86	**0.55**	0.20	0.43	0.27	0.33	0.71	0.45
Ag.Time, K = 20	0.21	0.57	0.31	0.32	0.86	0.46	0.21	0.57	0.31	0.26	0.71	0.38

Evaluation of algorithms on all features w.r.t. precision (P), recall (R) and F-measure (F)

	Naive			MFCC			Rhythm			Chroma		
D^{inter}, D^{intra}, D	0.31	0.22	1.40	0.66	0.49	1.36	0.26	0.19	1.32	0.80	0.59	1.36
D^{inter}, D^{intra}_{max}, D_{max}	0.31	0.43	0.73	0.66	0.62	1.06	0.26	0.26	0.97	0.80	0.61	1.31
Certitude for $\sigma = 0.2, 0.05, 0.01$	0.28	0.34	0.49	0.33	0.49	0.88	0.26	0.30	0.49	0.39	0.66	0.95
Label-Dep. Cert. for $\sigma = 0.2, 0.05, 0.01$	0.28	0.33	0.48	0.32	0.49	0.87	0.26	0.28	0.44	0.38	0.65	0.95

Dissimilarity-based and classification-based evaluation results

Now, we consider the pop song *Help*, performed by *The Beatles*, as another example for an audio file featuring a simple structure, which in this case consists of only four different segment labels:

intro (green), **verse** (cyan), **chorus** (red), **ending** (yellow).

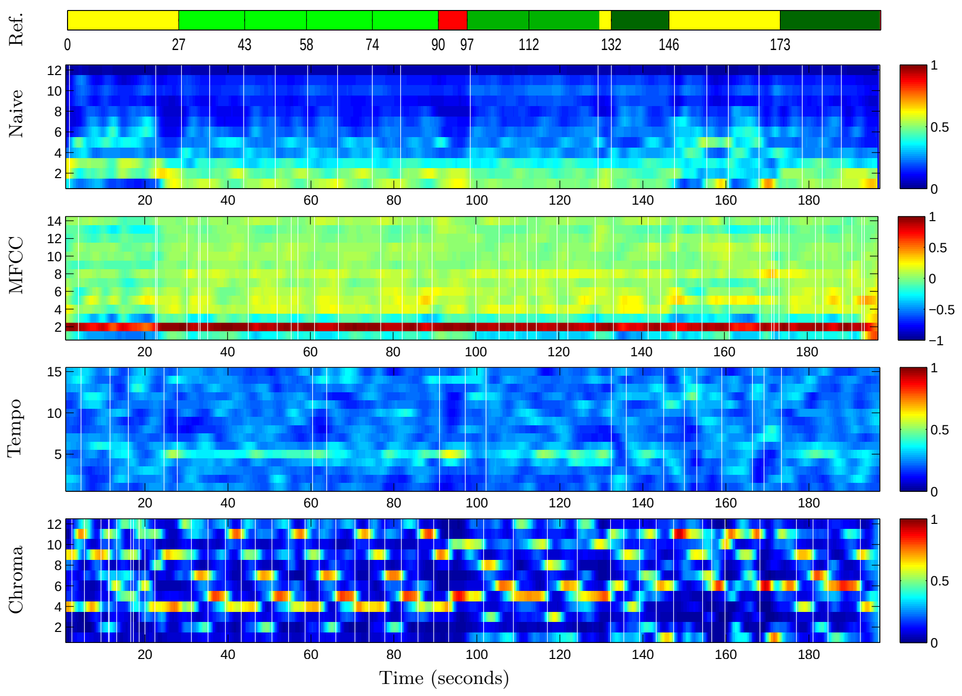
Remarkable values can be found in the evaluation scores derived from segmentations on chroma features: Starting from $K = 3$, the K-means algorithm yields a recall value of 1, together with a precision of about one third. This yields an optimal F-measure of 0.5, which surpasses these of all other but MFCC features. The same reasons as discussed before explain this effect.

However, an inspection of the optimal segmentation on MFCC features shows that the relatively poor F-measure of 0.55 is only due to a poor precision value. We confirm this observation by inspecting the optimal segmentation on MFCC features, as plotted above, in order to notice that problems arise mainly by doubling of segment boundaries. This effect is natural, as the transitions between segments in this song often feature a short vocal solo in high pitch, which is also reflected in the feature sequence for naive spectrum features that shows coefficients 5-6 featuring higher values in these parts. Transitions of this kind arise at seconds 8, 47 and 87.

As the optimal segmentation for MFCC features was obtained by time-restricted agglomerative clustering, the precision value could be improved significantly by introducing a model for segment lengths that directly penalizes segments of very short length, as proposed as future work in Section 7.1.

Again, we notice that repetition-based segmentation could also provide good results, as indicated by clear repetitions in the sequence of chroma features.

6.1.6 Zager and Evans - In the Year 2525



	Naive			MFCC			Rhythm			Chroma		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
K-Means, K = 2	0.14	0.08	0.11	0.13	0.08	0.10	0.25	0.42	0.31	0.00	0.00	0.00
K-Means, K = 3	0.43	0.75	**0.55**	0.11	0.08	0.10	0.20	0.42	0.27	0.03	0.08	0.05
K-Means, K = 4	0.30	0.83	0.44	0.21	0.75	**0.33**	0.20	0.67	0.30	0.03	0.08	0.04
K-Means, K = 5	0.29	0.83	0.43	0.21	0.75	0.33	0.22	0.83	0.34	0.13	0.50	**0.20**
K-Means, K = 6	0.27	0.83	0.41	0.19	0.75	0.31	0.20	0.83	0.33	0.13	0.50	0.20
Aggl., K = 2	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.17	0.29	0.05	0.08	0.06
Aggl., K = 3	0.14	0.08	0.11	0.00	0.00	0.00	0.75	0.25	0.38	0.04	0.08	0.06
Aggl., K = 4	0.13	0.08	0.10	0.00	0.00	0.00	0.29	0.33	0.31	0.03	0.08	0.05
Aggl., K = 5	0.11	0.08	0.10	0.00	0.00	0.00	0.25	0.33	0.29	0.06	0.17	0.08
Aggl., K = 6	0.10	0.08	0.09	0.00	0.00	0.00	0.29	0.42	0.34	0.05	0.17	0.08
Ag.Time, K = 5	0.25	0.08	0.13	0.00	0.00	0.00	0.50	0.17	0.25	0.25	0.08	0.13
Ag.Time, K = 8	0.14	0.08	0.11	0.14	0.08	0.11	0.43	0.25	0.32	0.14	0.08	0.11
Ag.Time, K = 11	0.20	0.17	0.18	0.10	0.08	0.09	0.40	0.33	0.36	0.10	0.08	0.09
Ag.Time, K = 16	0.20	0.25	0.22	0.07	0.08	0.07	0.33	0.42	0.37	0.07	0.08	0.07
Ag.Time, K = 20	0.32	0.50	0.39	0.11	0.17	0.13	0.32	0.50	**0.39**	0.05	0.08	0.06

Evaluation of algorithms on all features w.r.t. precision (P), recall (R) and F-measure (F)

	Naive			MFCC			Rhythm			Chroma		
D^{inter}, D^{intra}, D	0.41	0.15	2.70	0.62	0.24	2.58	0.31	0.18	1.68	0.85	0.52	1.64
D^{inter}, D^{intra}_{max}, D_{max}	0.41	0.37	1.11	0.62	0.48	1.29	0.31	0.30	1.02	0.85	0.77	1.10
Certitude for $\sigma = 0.2, 0.05, 0.01$	0.18	0.24	0.48	0.21	0.32	0.65	0.16	0.20	0.40	0.26	0.59	0.94
Label-Dep. Cert. for $\sigma = 0.2, 0.05, 0.01$	0.17	0.22	0.44	0.19	0.29	0.63	0.15	0.18	0.36	0.23	0.55	0.93

Dissimilarity-based and classification-based evaluation results

With the pop song *In The Year 2525*, performed by the duo *Zager and Evans*, we again see a candidate for repetition-based music segmentation. Similar to the previous song, *I Will Survive*, we have only one segment type for verse and chorus each. As this verse part is modulated several times during the song, we consider each modulated version as a segment of a new type. In total, this amounts to six distinct segment labels:

intro A (yellow), **verse A** (light green), **interlude A** (red), **verse B** (lime green), **interlude B** (yellow part in seconds 129-132), **verse C** (dark green).

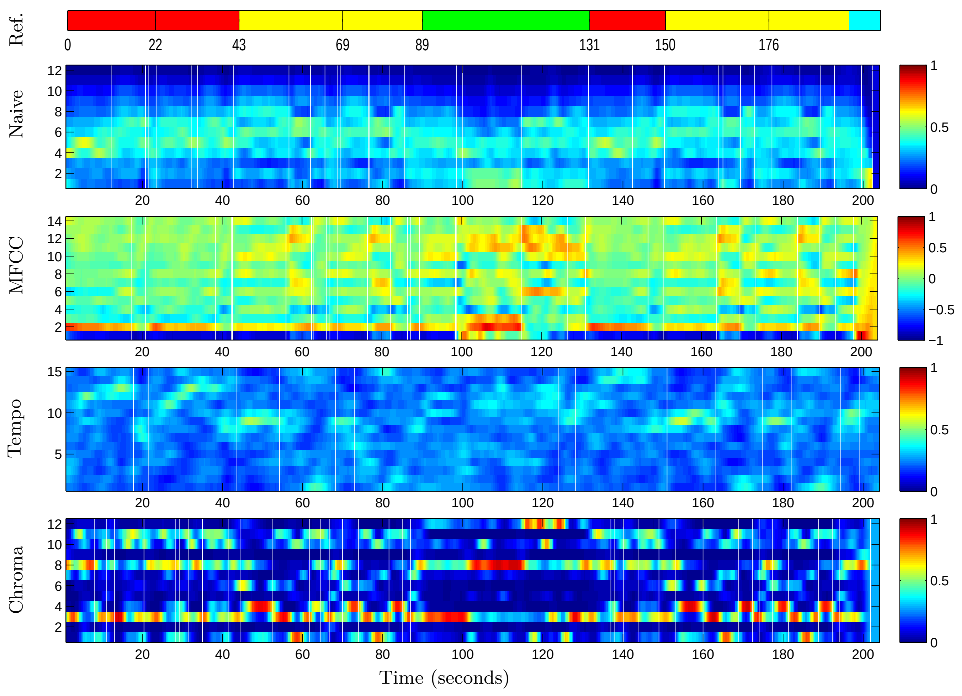
Strangely, MFCC features perform significantly worse than naive spectrum features on this song. While they yield similar recall values on the optimal segmentation, they suffer from poor precision. Furthermore, visual inspection of the feature sequence shows a constant peak in the second row of the MFCC feature sequence, suggesting that some error occured during feature computation. The feature evaluation score D also indicates that MFCC features should provide better results on this song.

The best segmentation with an F-measure of 0.55 is obtained for naive spectrum features. MFCC and tempo features yield significantly worse results. A similar result is obtained for the feature evaluation scores: With a value of 2.70 for D, naive spectrum features outperform the other features.

An interesting property of the musical structure is reflected in the chroma sequence: After four harmonically identical repetitions of the verse part, this part is transposed upwards by a semitone at second 90, then repeated three times, before another upward transposition by a semitone sets in at second 132. The harmonical transitions can be spotted in the chroma feature sequence: The transposition by one semitone corresponds to a cyclic shift of the chroma vector.

Note that even a repetition-based segmentation method would generally fail to detect this repetition, as it is no exact repetition, but rather a modulated repetition. Some music segmentation algorithms proposed by other authors, cf. [10], can deal with such modulated repetitions to some extent.

6.1.7 Johannes Brahms - Hungarian Dance No. 5



	Naive			MFCC			Rhythm			Chroma		
	P	R	F	P	R	F	P	R	F	P	R	F
K-Means, K = 2	0.18	0.38	0.24	0.08	0.13	0.10	0.22	0.50	0.31	0.17	0.38	0.23
K-Means, K = 3	0.15	0.50	0.24	0.08	0.13	0.10	0.17	0.50	0.26	0.20	0.88	**0.33**
K-Means, K = 4	0.10	0.38	0.15	0.08	0.13	0.10	0.13	0.50	0.20	0.18	0.88	0.30
K-Means, K = 5	0.15	0.75	0.25	0.14	0.50	0.22	0.17	0.88	0.29	0.16	1.00	0.28
K-Means, K = 6	0.19	0.75	**0.30**	0.23	0.88	**0.37**	0.17	0.88	0.29	0.13	1.00	0.23
Aggl., K = 2	0.00	0.00	0.00	0.33	0.13	0.18	0.43	0.75	**0.55**	0.00	0.00	0.00
Aggl., K = 3	0.00	0.00	0.00	0.33	0.13	0.18	0.35	0.75	0.48	0.00	0.00	0.00
Aggl., K = 4	0.00	0.00	0.00	0.25	0.13	0.17	0.23	0.75	0.35	0.00	0.00	0.00
Aggl., K = 5	0.00	0.00	0.00	0.20	0.13	0.15	0.22	0.75	0.34	0.00	0.00	0.00
Aggl., K = 6	0.13	0.38	0.19	0.17	0.13	0.14	0.28	1.00	0.43	0.13	0.50	0.20
Ag.Time, K = 5	0.00	0.00	0.00	0.25	0.13	0.17	0.25	0.13	0.17	0.25	0.13	0.17
Ag.Time, K = 8	0.00	0.00	0.00	0.14	0.13	0.13	0.29	0.25	0.27	0.14	0.13	0.13
Ag.Time, K = 11	0.20	0.25	0.22	0.10	0.13	0.11	0.30	0.38	0.33	0.10	0.13	0.11
Ag.Time, K = 16	0.20	0.38	0.26	0.13	0.25	0.17	0.20	0.38	0.26	0.13	0.25	0.17
Ag.Time, K = 20	0.16	0.38	0.22	0.11	0.25	0.15	0.26	0.63	0.37	0.16	0.38	0.22

Evaluation of algorithms on all features w.r.t. precision (P), recall (R) and F-measure (F)

	Naive			MFCC			Rhythm			Chroma		
D^{inter}, D^{intra}, D	0.39	0.26	1.47	1.18	0.73	1.62	0.31	0.25	1.21	0.77	0.56	1.38
D^{inter}, D^{intra}_{max}, D_{max}	0.39	0.44	0.88	1.18	0.95	1.23	0.31	0.30	1.02	0.77	0.75	1.04
Certitude for $\sigma = 0.2, 0.05, 0.01$	0.30	0.38	0.64	0.51	0.79	0.97	0.27	0.31	0.55	0.39	0.72	0.97
Label-Dep. Cert. for $\sigma = 0.2, 0.05, 0.01$	0.29	0.38	0.63	0.49	0.78	0.97	0.26	0.29	0.52	0.38	0.70	0.97

Dissimilarity-based and classification-based evaluation results

With the famous *Hungarian Dance No. 5*, written by *Johannes Brahms*, of which we consider a recording conducted by *Eugene Ormandy*, we start the block of classical music in our test playlist. Based on the experience that musical structure is significantly more complex in classical music than in popular music, we expect a significant performance decrease along with this change of genre. In the human segmentation for this piece, we distinguish between four different parts:

A (red), **B** (yellow), **C** (green) and **D** (cyan).

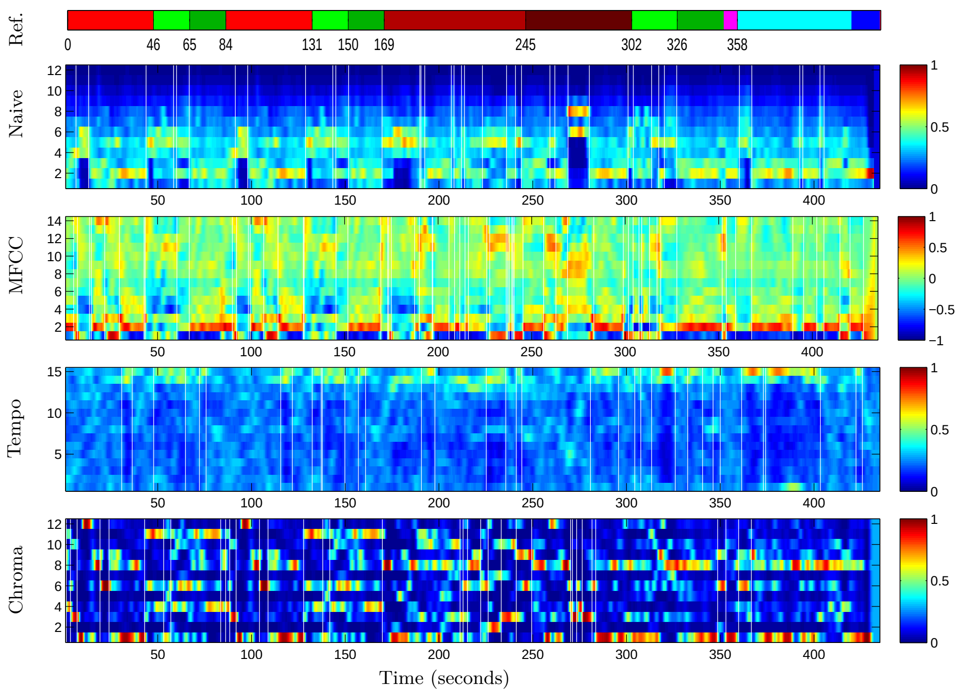
First, we notice that the sequence of chroma features shows high variation especially within segments. An exception to this general observation is found in part C, where long intervals of about 10 seconds feature roughly constant chroma vectors. As segment boundaries imply harmonical changes for this piece, we again obtain very good recall values, however at the expense of poor precision, making homogeneity-based segmentation on chroma features useless in this case. This is also reflected in the feature evaluation scores. Again, a repetition-based algorithm on chroma features would probably yield significantly better results.

Particularly interesting results can be found in the rhythm feature sequence. Note that this feature sequence yields an optimal F-measure of 0.55 and all other optimal segmentations produce very poor F-measures around 0.33, making rhythm features the optimal features for homogeneity-based clustering. Furthermore, visual inspection of the feature sequence confirms the tempo variations a human spots when listening to this song. For instance, the segment found in the optimal segmentation at about 20 seconds is certainly no segment in terms of musical form, but its high tempo stands out clearly, making it a justified false positive.

Interestingly, as the instrumentation changes at this point, MFCC features also capture this false positive. As part A is repeated twice, MFCC features also recognize this boundary twice. Naive spectrum features fail completely to recover this boundary. Apart from this single segment, comparing the two timbre features, we see that MFCC features yield superior results for the same algorithm and parameter choice.

We conclude, judging by the optimal F-measures on timbre features, that this song doesn't fulfill the homogeneity assumption on timbre features. In fact, we expect homogeneity-based music segmentation to generally produce poor results on classical music, as this kind of music often defines its structure by repetitive rather than by homogeneous parts. However, note that the feature evalation results don't validate this assumption by very low values: This effect is caused by the low intra-cluster homogeneity of clusters, indicated by high values for D_{intra} and D_{intra}^{max} which also imply higher values for D^{inter}. However, when restricting our view to D_{intra} and D_{intra}^{max}, we can at least conclude that intra-cluster homogeneity is not given in this piece.

6.1.8 Ludwig van Beethoven - Fifth Symphony, First Movement



	Naive			MFCC			Rhythm			Chroma		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
K-Means, K = 2	0.17	0.58	**0.27**	0.10	0.25	0.14	0.19	0.58	**0.29**	0.09	0.25	0.13
K-Means, K = 3	0.16	0.58	0.25	0.08	0.33	0.13	0.13	0.75	0.22	0.08	0.50	0.14
K-Means, K = 4	0.16	0.83	0.26	0.14	0.75	**0.23**	0.09	0.67	0.16	0.09	0.58	0.15
K-Means, K = 5	0.10	0.83	0.18	0.13	0.75	0.22	0.09	0.83	0.17	0.08	0.67	0.15
K-Means, K = 6	0.10	1.00	0.19	0.11	0.83	0.19	0.11	1.00	0.20	0.07	0.58	0.13
Aggl., K = 2	0.00	0.00	0.00	0.13	0.33	0.18	0.17	0.08	0.11	0.00	0.00	0.00
Aggl., K = 3	0.00	0.00	0.00	0.11	0.33	0.16	0.13	0.08	0.10	0.00	0.00	0.00
Aggl., K = 4	0.17	0.25	0.20	0.11	0.42	0.18	0.11	0.08	0.10	0.11	0.33	**0.17**
Aggl., K = 5	0.16	0.25	0.19	0.11	0.42	0.17	0.16	0.58	0.25	0.10	0.33	0.15
Aggl., K = 6	0.16	0.25	0.19	0.10	0.42	0.16	0.15	0.58	0.24	0.09	0.33	0.15
Ag.Time, K = 5	0.00	0.00	0.00	0.25	0.08	0.13	0.25	0.08	0.13	0.00	0.00	0.00
Ag.Time, K = 8	0.00	0.00	0.00	0.14	0.08	0.11	0.14	0.08	0.11	0.00	0.00	0.00
Ag.Time, K = 11	0.00	0.00	0.00	0.10	0.08	0.09	0.10	0.08	0.09	0.00	0.00	0.00
Ag.Time, K = 16	0.13	0.17	0.15	0.13	0.17	0.15	0.07	0.08	0.07	0.00	0.00	0.00
Ag.Time, K = 20	0.16	0.25	0.19	0.11	0.17	0.13	0.05	0.08	0.06	0.05	0.08	0.06

Evaluation of algorithms on all features w.r.t. precision (P), recall (R) and F-measure (F)

	Naive			MFCC			Rhythm			Chroma		
D^{inter}, D^{intra}, D	0.47	0.31	1.50	1.16	0.87	1.33	0.32	0.24	1.34	0.87	0.70	1.24
D^{inter}, D^{intra}_{max}, D_{max}	0.47	0.48	0.98	1.16	1.19	0.98	0.32	0.36	0.90	0.87	0.93	0.93
Certitude for $\sigma = 0.2, 0.05, 0.01$	0.15	0.23	0.41	0.29	0.60	0.93	0.13	0.17	0.35	0.20	0.47	0.84
Label-Dep. Cert. for $\sigma = 0.2, 0.05, 0.01$	0.13	0.19	0.35	0.24	0.58	0.92	0.12	0.15	0.30	0.16	0.41	0.82

Dissimilarity-based and classification-based evaluation results

With the famous First Movement of the *Fifth Symphony* written by *Beethoven*, we consider a piece that is very likely to cause homogeneity-based segmentation to fail. First of all, the musical structure in this movement is very complex, as common to classical music and in particular to symphonies. Even in repetition-based clustering, problems could arise due to modulation of motives which is common for this kind of music: For instance, motives and themes can be repeated by different instruments, in transposed pitch, with rhythmical variations, etc.

When segmenting this movement according to musical form, we obtain the following labels:

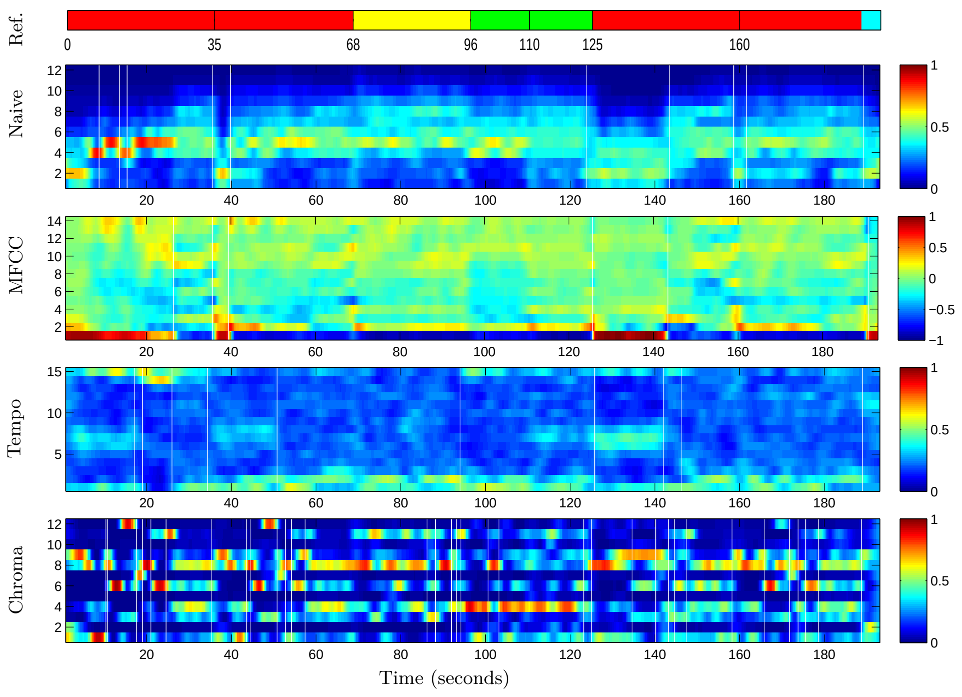
A (red), **B** (light green), **C** (dark green), **D** (bordeaux), **E** (dark red), **F** (pink), **G** (cyan), **H** (blue).

As no homogeneity is given within the segments in the human reference segmentation, both average and maximal intra-cluster dissimilarity values are high for all features. Note that this implies one of the highest inter-cluster dissimilarity values for MFCC features among all audio files in the test set, as for every pair of labels, two intrinsically inhomogeneous clusters are compared, yielding a high inter-dissimilarity value. More advanced feature evaluation measures should compensate this effect.

As expected, the resulting segmentations are more or less random, which is indicated by poor F-measures. We just point to the oboe solo starting at 267 seconds that is captured in the segmentations computed for naive spectrum features and MFCC features. This (false) segment is nearly the only time interval in this piece that features timbral homogeneity.

Furthermore, the values for D_{intra} and D_{intra}^{max} also indicate the lack of homogeneity in this piece. As pointed out before, the total evaluation score D suffers from the fact that high intra-cluster dissimilarity also implies an elevated inter-dissimilarity score D^{inter}. Nevertheless, we observe that the values for D are among the worst in our whole evaluation set, reflecting the fact that the optimal F-measures also consist of very low values.

6.1.9 Shostakovich - Jazz Suite, Second Waltz



	Naive			MFCC			Rhythm			Chroma		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
K-Means, K = 2	0.40	0.57	**0.47**	0.50	0.43	**0.46**	0.43	0.43	0.43	0.04	0.14	0.06
K-Means, K = 3	0.36	0.57	0.44	0.36	0.57	0.44	0.33	0.43	0.38	0.14	0.57	0.22
K-Means, K = 4	0.33	0.57	0.42	0.29	0.57	0.38	0.21	0.71	0.32	0.15	0.71	**0.24**
K-Means, K = 5	0.24	0.86	0.38	0.25	0.71	0.37	0.19	0.86	0.31	0.10	0.57	0.17
K-Means, K = 6	0.23	1.00	0.37	0.27	1.00	0.42	0.14	0.71	0.24	0.12	1.00	0.22
Aggl., K = 2	0.40	0.57	0.47	0.50	0.43	0.46	0.00	0.00	0.00	0.00	0.00	0.00
Aggl., K = 3	0.36	0.57	0.44	0.43	0.43	0.43	0.14	0.14	0.14	0.05	0.14	0.07
Aggl., K = 4	0.33	0.57	0.42	0.33	0.43	0.38	0.10	0.14	0.12	0.04	0.14	0.07
Aggl., K = 5	0.27	0.57	0.36	0.30	0.43	0.35	0.20	0.43	0.27	0.11	0.57	0.18
Aggl., K = 6	0.25	0.57	0.35	0.27	0.43	0.33	0.25	0.57	0.35	0.11	0.57	0.18
Ag.Time, K = 5	0.25	0.14	0.18	0.50	0.29	0.36	0.25	0.14	0.18	0.00	0.00	0.00
Ag.Time, K = 8	0.14	0.14	0.14	0.43	0.43	0.43	0.29	0.29	0.29	0.00	0.00	0.00
Ag.Time, K = 11	0.30	0.43	0.35	0.30	0.43	0.35	0.40	0.57	**0.47**	0.00	0.00	0.00
Ag.Time, K = 16	0.27	0.57	0.36	0.27	0.57	0.36	0.33	0.71	0.45	0.13	0.29	0.18
Ag.Time, K = 20	0.21	0.57	0.31	0.26	0.71	0.38	0.32	0.86	0.46	0.16	0.43	0.23

Evaluation of algorithms on all features w.r.t. precision (P), recall (R) and F-measure (F)

	Naive			MFCC			Rhythm			Chroma		
D^{inter}, D^{intra}, D	0.40	0.23	1.71	1.11	0.71	1.57	0.34	0.24	1.46	0.78	0.50	1.55
D^{inter}, D^{intra}_{max}, D_{max}	0.40	0.39	1.02	1.11	1.16	0.96	0.34	0.37	0.93	0.78	0.72	1.08
Certitude for $\sigma = 0.2, 0.05, 0.01$	0.30	0.42	0.62	0.46	0.68	0.92	0.27	0.35	0.60	0.37	0.68	0.97
Label-Dep. Cert. for $\sigma = 0.2, 0.05, 0.01$	0.27	0.33	0.56	0.38	0.63	0.92	0.26	0.30	0.54	0.35	0.65	0.97

Dissimilarity-based and classification-based evaluation results

The last classical piece in our test set is the famous *Jazz Suite, Second Waltz*, written by *Shostakovich* and conducted by *Yablonsky*. Though this piece features a rather simple structure, we expect no good results for homogeneity-based clustering, as we in general expect classical music to fulfill the repetition assumption rather than the homogeneity assumption. Again, the human reference segmentation is based on repetitive structure. In total, we obtain the following segment labels:

A (red), **B** (yellow), **C** (green) and an ending part **D** (cyan).

Interestingly, the first row in the MFCC feature sequence attains high values exactly for parts featuring solos played by wind instruments. From seconds 1-20, a clarinet is playing the famous theme of this waltz, while from seconds 125-140, it is picked up by a bassoon. This example shows that MFCC features indeed allow for timbre recognition in the presence of melodic variation. Note especially that in seconds 1-20, dependency on the fundamental frequency arises only in the higher MFCC rows, i.e. in coefficients 10-14. Recalling Section 2.2, we expect this kind of behaviour.

Apart from this desired property, visual inspection of MFCC features shows that even some segment boundaries that were not found by the optimal segmentation on MFCC features can be spotted visually, e.g. the boundaries at 68, 96, 110 and 160 seconds. We suppose that time-restricted agglomerative clustering would be able to recognize these boundaries when provided with a better parameter choice, i.e. some K between 9 and 10. Note that the optimal F-measure for time-restricted agglomerative clustering on MFCC features is in fact likely to be found for $K \in \{9, 10\}$, as the F-measure increases for smaller K and decreases for larger K, suggesting that the maximum is found in between.

Furthermore, the tempo feature sequence captures the tempo changes found in this piece. Note that two segment-wise almost constant peaks can be found in this sequence, namely one in the coefficients 6-7 at seconds 1-20, and one in the first three coefficients. This discriminative effect enables rhythm features to produce optimal F-measures equal to these of timbre features.

7 Conclusions

7.1 Summary

The chapters preceding this conclusion introduced music segmentation as a general problem in order to present and evaluate solution strategies that rely on the assumption that musical structure can be modeled in terms of the homogeneity assumption. This assumption required an audio file to display similar musical properties within clusters and dissimilar properties across different clusters, two central requirements we called intra-cluster similarity and inter-cluster dissimilarity.

By introducing different features and roughly describing their extraction process in Chapter 2, we pointed out several methods to measure musical qualities of an audio file in terms of a sequence of feature vectors that can be extracted automatically from an audio signal. In this thesis, we gave an overview over three different musical properties, namely the timbre, tempo and chroma, and presented naive spectrum features and MFCC features as a way to express timbral properties, which we assumed to be best suited for homogeneity-based music segmentation. As we a priori considered tempo and chroma properties to be less suited for this goal, we introduced only one feature type for each property, namely cyclic Fourier-based tempo features and chroma features. Judging by Chapter 6, homogeneity-induced structure of music was seen mostly in timbre feature sequences for audio files we a priori assumed to fulfill the homogeneity assumption.

After introducing features, we proposed methods for feature evaluation with respect to human segmentations in Chapter 3. This idea was intended to give an algorithm-independent indicator for the extent to which a feature sequence represents a human segmentation in terms of the homogeneity assumption. Furthermore, it should formalize the visual results we could observe in Chapter 6.

In order to compute segmentations from feature sequences, we discussed two clustering algorithms in Chapter 4, namely the K-means algorithm and agglomerative clustering, that each allow for identifying homogeneous clusters in point clouds derived from a feature sequence while neglecting its ordering. These algorithms served as baselines whose main purpose was to compare more complex algorithms to their results.

As we considered time-ignoring clustering approaches not to model the inherent temporal component of music, starting in Section 5.1, we introduced and discussed feature smoothing and pointed out its unwanted effect at boundaries in a feature sequence. This effect was then exploited in Section 5.2 to create a covariance-based novelty function that detects boundaries by searching for maximally inhomogeneous regions in a feature sequence, which under the homogeneity assumption correspond to boundary regions. This novelty function was compared to other functions both with respect to a distance-like functional on pairs of novelty functions and the quality of the optimal segmentation that could be derived from novelty functions.

Furthermore, we introduced methods for dealing with the order of feature sequences in the framework of agglomerative clustering in Section 5.3. In a first method, which we do not expect to work in practice, a direct manipulation of the self-similarity matrix that is input to agglomerative clustering was used to create a block-structured similarity matrix. In this matrix, the positions and sizes of the blocks were determined by a novelty function. In the second method we introduced, the agglomerative clustering algorithm was modified to merge only adjacent clusters, which implies that at every point in the execution of this time-restricted agglomerative clustering algorithm, clusters are intervals in the domain of the input audio file. This effect distinguished this algorithm from the time-ignoring clustering algorithms introduced in Chapter 4.

Finally, we evaluated the algorithms on a set of selected songs of varying structural complexity that partly fulfilled the homogeneity assumption. To this goal, we compared segmentations found by the algorithms to human segmentations with respect to precision, recall and F-measures on the boundary sets. For each algorithm, a set of possible parameter choices was proposed and the choice that resulted in the segmentation maximizing the F-measure was considered in detail. The optimal F-measures we obtain could possibly serve as an upper bound to the performance of homogeneity-based segmentation algorithms with a *fixed* parameter choice.

Furthermore, we also computed both dissimilarity-based and classification-based feature evaluation scores for this test set and noticed that in some cases, they indeed correlate with the F-measures computed from optimal segmentations. However, several problems could be noticed for dissimilarity-based feature evaluation, including for instance the problem of summarizing a multitude of values to one single score. Furthermore, we observed that high intra-cluster dissimilarity in most cases also implies high values for inter-cluster dissimilarity, therefore deteriorating the final evaluation score. The second method, classification-based feature evaluation, was based on both a probabilistic and a geometric interpretation of feature unambiguity, and was used in Section 3.2 in order to visualize feature ambiguity over an audio file. However, when it comes to a practical feature evaluation score, two major shortcomings prevented this method from producing the desired results: dependency on global scaling of the features and the problem of condensing a whole curve into one value.

7.2 Future Work

An interesting track that might be explored in the near future starts with the observation that the results of homogeneity-based segmentation can be used as a starting point for repetition-based segmentation. For instance, a feature sequence that has been condensed to a string-like sequence of chord symbols can be checked very easily for transposed repetitions of segments.

Furthermore, improvements can also be made to homogeneity-based segmentation itself. In Chapter 2 of this thesis, we only considered features capturing a single aspect of music, namely timbre, rhythm or harmony. It might be of great interest to combine different feature types into one mixed feature that captures all of the three properties mentioned above. One might for instance concatenate the feature vectors and introduce some weighting for the coordinates or simply create new self-dissimilarity matrices by computing a convex combination of the self-dissimilarity matrices corresponding to timbre, rhythm and harmony features. In both cases, the weights could be learned on training sets consisting of songs of a single genre, yielding optimal choices of weight factors for different genres of music.

As the central component of agglomerative clustering is the cluster-dissimilarity measure used to compare different segments, further work should also be invested in the development of more complex cluster-dissimilarity measures. For instance, the average linkage value between two segments S_1, S_2, which is the measure we used in this thesis, is the mean of *all* dissimilarities between pairs of feature vectors. In particular, every point in time in S_1 is compared to every point in time in S_2, i.e. S_1 and S_2 are compared as sets, not as sequences. By computing a DTW-like cost measure for segments, similar to [16], we could introduce time-dependency into the cluster-dissimilarity measure.

Another possibility to improve cluster-dissimilarity measures could consist of introducing a segment length regularizer that penalizes segments of untypical lengths, such as extremely short or extremely long segments. This regularizer could be learned by processing a large number of human annotations and estimating the distribution of segment lengths, possibly again restricted to single music genres. We expect this regularizer to improve time-restricted agglomerative clustering significantly.

With the aid of such cluster-dissimilarity measures, agglomerative clustering could in the end possibly serve as a method for repetition-based hierarchical clustering. This method might be able to create dendrograms capturing musical form on different scales, i.e. the leaves could correspond to single notes, intermediate levels to motives, themes and verses, while the root could represent the whole song. Consider for instance a song featuring a structure of type 'ABCCABCC'. The single letters could be interpreted as motives, sequences such as 'AB' or 'CC' could correspond to themes, etc.

Bibliography

[1] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. NP-hardness of euclidean sum-of-squares clustering. *Mach. Learn.*, 75(2):245–248, 2009.

[2] Tue Haste Andersen. Mixxx: towards novel dj interfaces. In *NIME '03: Proceedings of the 2003 conference on New interfaces for musical expression*, pages 30–35, Singapore, Singapore, 2003. National University of Singapore.

[3] Michael Casey. General sound classification and similarity in mpeg-7. *Org. Sound*, 6(2):153–164, 2001.

[4] Matthew L. Cooper and Jonathan Foote. Automatic music summarization via similarity analysis. In *ISMIR 2002, 3rd International Conference on Music Information Retrieval, Paris, France, October 13-17, 2002, Proceedings*, 2002.

[5] David Damm, Christian Fremerey, Frank Kurth, Meinard Müller, and Michael Clausen. SyncPlayer - Multimodale Wiedergabe, Navigation und Suche in heterogenen digitalen Musikkollektionen. In Thomas Mandl, Norbert Fuhr, and Andreas Henrich, editors, *Proceedings of the Workshop on Lernen-Wissen-Adaptivität (LWA 2008)*, pages 1–8, Würzburg, Germany, October 2008. GI.

[6] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2 edition, November 2000.

[7] Jonathan Foote. Visualizing music and audio using self-similarity. In *MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 77–80, New York, NY, USA, 1999. ACM.

[8] Jonathan T. Foote and Matthew L. Cooper. Media segmentation using self-similarity decomposition. In *In Proc. SPIE Storage and Retrieval for Multimedia Databases*, pages 67–75, 2003.

[9] M.M. Goodwin and J. Laroche. Audio segmentation by feature-space clustering using linear discriminant analysis and dynamic programming. In *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, pages 131–134, Oct. 2003.

[10] Masataka Goto. Smartmusickiosk: music listening station with chorus-search function. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 31–40, New York, NY, USA, 2003. ACM.

[11] Peter Grosche, Meinard Müller, and Frank Kurth. Cyclic tempogram, a mid-level tempo representation for music signals. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Dallas, TX, USA, 2010. IEEE. submitted.

[12] Matthias Hein. *Machine Learning.* Lecture Notes, February 2009.

[13] M. Levy and M. Sandler. Structural segmentation of musical audio by constrained clustering. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(2):318–326, Feb. 2008.

[14] Meinard Müller. *Information Retrieval for Music and Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[15] Elias Pampalk. A matlab toolbox to compute similarity from audio. In *Proceedings of the 2004 International Conference on Music Information Retrieval*, pages 254–257, New York, NY, USA, 2004.

[16] Jouni Paulus and Anssi Klapuri. Music structure analysis by finding repeated parts. In *AMCMM '06: Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 59–68, New York, NY, USA, 2006. ACM.

[17] M J Sabin and R M Gray. Global convergence and empirical consistency of the generalized lloyd algorithm. *IEEE Trans. Inf. Theor.*, 32(2):148–155, 1986.