Automatic Detection of the Musical Structure within Pieces of Music

Masters Thesis

Alexander Wankhammer

Supervisor: DI Dr. Alois Sontacchi
Assessor: Univ.-Prof. Dr. Robert Höldrich

Graz, June 2011

Institute of Electronic Music and Acoustics
University of Music and Performing Arts, Graz

Automatic Detection of the Musical Structure within Pieces of Music
Alexander Wankhammer

Supervisor: DI Dr. Alois Sontacchi
Assessor: Univ.-Prof. Dr. Robert Höldrich

Institute of Electronic Music and Acoustics
University of Music and Performing Arts, Graz
June 2011

Abstract

Music Structure Discovery (MSD) for popular music is a well known task in *Music Information Retrieval* (MIR). In this thesis a new approach for finding the musical structure of a piece of music is proposed. The algorithm is based on the search for repeated vertical slices inside a modified bar level Self Distance Matrix (SDM) using a template matching algorithm. After an initial segmentation is found based on the analysis of the template matching results, a post processing step helps to further investigate the found musical structure by searching for repeated sub-sequences in the preliminary segmentation.

The new idea of template matching instead of trying to find explicit blocks or off-diagonal lines inside the SDM is independent of any specific characteristics of the investigated matrix and can therefore be used on a wide range of different songs. The proposed algorithm has been evaluated on a well known dataset consisting of 13 full albums by *The Beatles* and the evaluation results have been compared to five state-of-the-art MSD algorithms. It could be shown that the segmentation performance of the proposed system on the chosen testing corpus is significantly better than the performance of any other system included in the evaluation.

In a complementary experiment, the detected musical structure is used to investigate the influence of focusing the audio analysis to distinct representative song segments, when trying to find the genre label for a song. The results indicate that the additional knowledge can help to significantly improve the results of the classification system when focusing on representative sections instead of arbitrary parts, if very short excerpts (< 10 seconds) are used for classification. For longer excerpts, no difference for the classification of representative and arbitrary parts has been found.

Automatische Erkennung von Liedabschnitten in Musikstücken
Alexander Wankhammer

Betreuer: DI Dr. Alois Sontacchi
Begutachter: Univ.-Prof. Dr. Robert Höldrich

Institut für Elektronische Musik und Akustik
Universität für Musik und Darstellende Kunst, Graz
Juni 2011

Kurzfassung

Die automatische Erkennung von Liedabschnitten ist eine typische Problemstellung im Forschungsgebiet *Music Information Retrieval* (MIR). In dieser Diplomarbeit wird ein neuer Ansatz zur automatischen Bestimmung der inneren Struktur von Musikstücken präsentiert. Über die Detektion wiederkehrender Muster innerhalb einer über Taktabschnitte berechneten Self Distance Matrix (SDM), wird zunächst ein Profil aller sich wiederholenden Bereiche des untersuchten Musikstückes erstellt. Die finale Segmentierung des Stückes wird schließlich in mehreren Schritten unter Berücksichtigung aller zeitlichen Abhängigkeiten der gefundenen Bereiche festgelegt.

Dank der neuartigen Methode, einen Mustererkennungs-Algorithmus zur Detektion wiederkehrender Strukturen innerhalb einer SDM einzusetzen, ist der Algorithmus weitestgehend unabhängig von spezifischen Charakteristika der untersuchten Matrix und liefert stabile Ergebnisse für ein breites Spektrum von Musikstücken. Das vorgestellte System wurde basierend auf einem weit verbreiteten Datensatz bestehend aus 13 Alben der *Beatles* evaluiert und die Evaluierungsergebnisse wurden mit fünf state-of-the-art Music Structure Discovery (MSD) Algorithmen verglichen. Dabei konnte gezeigt werden, dass der vorgestellte Algorithmus für den verwenden Datensatz signifikant bessere Segmentierungs-Ergebnisse liefert als die in die Evaluierung eingebundenen Referenz-Systeme.

In einem die Arbeit ergänzenden Experiment wurde zusätzlich analysiert welchen Einfluss es hat, wenn die zur Genre Klassifikation eines Liedes herangezogenen musikalischen Merkmale eines Stückes nur innerhalb repräsentativer Liedabschnitte extrahiert werden. Dabei konnten bei der Klassifikation sehr kurzer repräsentativer Abschnitte (< 10 Sekunden) signifikant bessere Klassifizierungs-Ergebnisse erzielt werden als bei der Klassifikation zufällig gewählter Abschnitte derselben Länge. In den Klassifizierungsergebnissen längerer repräsentativer und zufällig gewählter Abschnitte zeigten sich hingegen keine signifikanten Unterschiede.

To my father.

Contents

1 Introduction 7

1.1 Music Structure Discovery (MSD) . . . 7

1.1.1 Existing Work . . . 8

1.1.2 Applications . . . 9

1.2 Outline of the Thesis . . . 10

2 Audio Feature Extraction 14

2.1 Mel Frequency Cepstral Coefficients (MFCC) . . . 15

2.2 Fluctuation Patterns (FP) . . . 16

2.3 Constant Q-Transform (CQT) and Chroma . . . 16

2.4 Beat Detection and Averaging . . . 19

2.4.1 Meter Induction . . . 21

3 Recurrence Analysis 23

3.1 Self Distance Matrix (SDM) . . . 24

3.1.1 SDM Mapping and Combination . . . 26

3.2 Template Matching . . . 29

3.2.1 Normalized Cross Correlation (NCC) . . . 29

3.2.2 Recurrence Matrix . . . 31

3.2.3 Repetition Surface . . . 32

3.3 Novelty Function . . . 35

4 Time Interval Data 38

4.1 Segment Grouping . . . 40

4.2 Subset Sequences . . . 42

4.2.1 Repetition Quality . . . 43

4.2.2 Prototype Segment . . . 44

4.3 Segment Combination . . . 46

4.3.1 Two Overlapping Intervals . . . 46

4.3.2 Two Overlapping Interval Sequences . . . 48

4.3.3 Multiple Overlapping Interval Sequences . . . 50

4.4 Preliminary Segmentation . . . 52

4.4.1 Detection of Missed Boundaries . . . 52

5 Temporal Pattern Mining **54**
5.1 Substring Patterns 54
5.1.1 Substring Search 55
5.2 Quality Measures 56
5.2.1 Path Search 58
5.3 Final Segmentation 60

6 Evaluation and Results **61**
6.1 Dataset 61
6.2 Evaluation Metrics 62
6.3 Results 64
6.4 Discussion 66

7 Segment Related Genre Classification **71**
7.1 Automatic Genre Classification 71
7.2 Selected Classification System 75
7.3 Classification Experiments 76
7.3.1 Testing Corpus 76
7.3.2 Thumbnail Generation 76
7.3.3 Experiments 77
7.4 Results 78
7.5 Discussion 81

List of Figures

1 Overview on the main processing stages of the algorithm. 11

2 Mapping function from the linear frequency scale to the Mel Scale (left) and grouping of DFT bins according to 36 triangular Mel filters (right). . 15

3 Compressed 12 band Mel spectrogram (left) and resulting fluctuation pattern (right) calculated for a combined signal with one 300 Hz tone modulated by a 3 Hz sine (band 5) and a 800 Hz tone modulated by a 7 Hz sine (band 9). 17

4 Constant Q-spectrogram based on a semitone scale (left) and the related chromagram (right). The separation of the constant Q-spectrogram into 5 octaves is indicated by horizontal lines. 19

5 Based on a 40 band Mel spectrogram (top), the onset detection function $\boldsymbol{o}(t)$ (bottom) is computed. 20

6 The raw autocorrelation $\boldsymbol{a}(\tau)$ of $\boldsymbol{o}(t)$ and the perceptual weighting window (left) are combined into a perceptually weighted autocorrelation $\boldsymbol{a}_w(\tau)$ (right). 20

7 Based on characteristic maxima inside $\boldsymbol{a}(\tau)$, the meter of a song can be estimated. This example shows $\boldsymbol{a}(\tau)$ for a song played in common time. Therefore, additional correlation maxima (besides the maximum at τ_{max}) are expected at the following characteristic multiples of τ_{max}: $4, 8$ and 16. 22

8 Trajectory of a 2-D sequence and the resulting RP (ϵ was set to 0.5 in this example). 24

9 Example for an SDM calculated based on individual vector pairs (upper triangular matrix) and an SDM calculated based on embedded vectors and an embedding dimension of 4 (lower triangular matrix). 25

10 Example for the manifestation of characteristic structures (off diagonal lines, square blocks) in a RP, based on a simple 1-D sequence. 26

11 Based on a histogram (left, top), Otsu's method finds the optimal threshold to separate the given values into two classes. The found separation point γ is used as input parameter for the mapping function (Equation 11) to perform the mapping of the given SDMs. 28

12 The template image (middle, *IEM* logo) is searched inside the combined search image (left). The maximum correlation value inside $\boldsymbol{C}$ (right) indicates the detected position of the template. 30

13 As indicated with dashed boxes, maxima in the cross correlation vector (left, bottom) mark possible repetitions of the template $\boldsymbol{T}_k$ in the related search image $\boldsymbol{I}_k$. The recurrence matrix (right) is created by storing consecutive correlation vectors side by side into a matrix. In this example, the weak off diagonal lines, indicating repeated sequences inside the original SDM, are nicely represented as horizontal lines in the recurrence matrix. . 32

14 The median-filtered recurrence matrix $\boldsymbol{R}_f$ (left) is binarized based on a peak detection algorithm, which is column-wise applied to the matrix. As indicated, the used template width (in this example 8 beats) has to be incorporated when defining time lags between detected segments. 33

15 Examples for overlapping conflicts of related segment pairs found inside the repetition surface. 34

16 The final correlation kernel $\boldsymbol{K}$ is calculated starting with a radial Gaussian function (a.) combined with a diagonal weighting matrix $\boldsymbol{D})(b.)$, multiplied by a checkerboard kernel $\boldsymbol{K}_b$ (c.). 36

17 This Figure shows three different (normalized) versions of the novelty function: $\boldsymbol{\eta}_1$ (full SDM + kernel without diagonal weighting); $\boldsymbol{\eta_2}$ (mapped SDM + kernel without diagonal weighting); $\boldsymbol{\eta_3}$ (mapped SDM + kernel with diagonal weighting). It can clearly be noted that the correlation with the full SDM only exhibits very strong block structures ($\boldsymbol{eta}_1$), while the correlation with the mapped SDM reveals much more details ($\boldsymbol{eta}_2$). Additionally, the proposed diagonal mapping significantly reduces the overall offset in ($\boldsymbol{eta}_3$) and shows very clear peaks at potential segment borders. 37

18 Allen's interval relations. Each relation can be read as: "A (*relation*) B". . 38

19 Relations inside the repetitions surface. 40

20 Redundancies and higher order relations inside the repetitions surface. . . 42

21 The time intervals combined into one subset and the resulting sequence of related time intervals. (For simplicity, only the unique intervals related to the subset $\boldsymbol{\Phi}$ are depicted in this Figure. Identical source or target segments are ignored.) 43

22 Basic parameters of a prototype segment (top) and example of interval sequence represented by the prototype for a given set of parameters (bottom). 45

23 Operations of the algorithm based on the regions defined inside and outside a main interval $\boldsymbol{t}_m$ considering potential overlaps of starts s_s and stops e_s of a sub interval. 47

24 Exemplary processing of two overlapping intervals. 49

25 Exemplary processing of two overlapping sequences (two prototype segments). 50

26 Exemplary processing of multiple overlapping interval sequences. 51

27 Segment *A* is identified as unusually long segment. The first occurrence of the pattern inside the 4 identically repeated sections is extracted (top, right) and analyzed using the novelty function. As the maximum inside $\boldsymbol{\eta}$ is above t_η, it indicates that the detected segment *A* should be split into two segments *A'* and *C*. 53

28 Example for a substring of length 4 $\{A - B - C - D\}$ and all possible sub sequences based on the deletion of different borders. 55

29 Example for a graph representation of all possible combinations of 4 patterns. The exemplary path $\{in \rightarrow AB \rightarrow CD \rightarrow out\}$ (indicated by dashed gray lines) shows the weights included if the patterns A and B and the patterns C and D are combined. 56

30 Gaussian weighting window defining the quality of different (combined) pattern lengths for a song played in common time ($4/4$). 57

31 To find an estimate of the similarity of two or multiple musical patterns, the respective slices inside the mapped SDM (left) are correlated. In this example, the patterns C and D should be combined, as they represent almost identical SDM slices, while the other patterns should remain split (no correlation). 59

32 Autoamtically detected segmentation (*Det.*), SDM and reference segmentation (*Ref.*) of the song *"Head Over Feet"* by *Alanis Morisette*. 60

33 Segmentation example: R_r 60%, P_r 62.5%, F_r 61.2%, $1{-}m$ 0.66, $1{-}f$ 0.75 64

34 Scatterplot of *segment fragmentation* against *missed boundaries* for the *TUT Beatles* corpus (left) and the *Mixed* corpus (right). 66

35 (Example 1) *The Beatles - I Saw Her Standing There* 67

36 (Example 2) *The Beatles - Devil In Her Heart* 67

37 (Example $3a$) *Britney Spears - Hit Me Baby One More Time* 68

38 (Example $3b$) *The Beatles - All I've Got To Do* 68

39 (Example 4) *Chicago - Old Days* 69

40 (Example 5) *The Clash - Should I Stay Or Should I Go* 70

41 As indicated, focusing the classification to selected segments based on a pre-processing stage, does not modify the basic classification algorithm, but only provides additional side information. (Dashed lines indicate optional paths.) 72

42 Homogeneity profiles of all genres. The bars indicate the average coverage of each song by different genres (bar 1 $\rightarrow$ predominant genre, bars $1-3$ $\rightarrow$ sub-genres). 81

1 Introduction

The steadily growing amount of multimedia content (e.g. in online music collections and other media) calls for the development of entirely new concepts of analysis, classification and organization of the available bulk of data. According to this needs, one growing branch in the field of music-related signal processing is *Music Information Retrieval* (MIR).

The main objective of MIR systems may best be described as *musical data mining* as these systems allow to automatically extract information from music in digital formats that otherwise has to be defined by human annotators in a time consuming-process. The automatic processing does not only help to drastically reduce the afford of maintaining consistent meta-data for fast-growing song collections, but also provides new ways of searching, organizing and discovering music.

To accomplish this relatively complex tasks, the research in MIR is often based on the interaction of multiple techniques from interdisciplinary fields like psychoacoustics, musicology, signal processing and data mining. Additionally, many MIR systems (e.g. music recommendation) are based on the interaction of multiple smaller sub systems, all developed to extract some specific characteristics from an audio file[1].

This thesis will focus on the extraction of one specific characteristic of a song: the musical structure. After a short introduction to the field of music structure discovery, an outline of the thesis will be given and the the main stages of the proposed algorithm will be addressed.

1.1 Music Structure Discovery (MSD)

If we listen to music, we are literally surrounded by repetitive structures and varying patterns. We will hear different combinations of melodic and harmonic progressions, ongoing rhythmic movements and on a wider time scale changes in the timbre and the dynamics of the song. As any section of a song with some kind of inner similarity or consistency may be interpreted as *musical segment*, the definition of the *musical structure* is never unambiguous and strongly depended on the investigated temporal scale.

According to [Dav66], this hierarchical structure within pieces of music can be compared to the structure of a story: starting with individual *notes*, one can form *phrases* from

[1]A good review on many current content-based MIR techniques is given in [CVG$^+$08]

sequences of notes and *sentences* from two or more *phrases*. On a wider time scale, *sentences* can again be combined into *paragraphs*, forming the highest structural level of the song.

Despite this variety of temporal descriptors, most MSD algorithms aim to detect a specific high-level musical structure, in music theory referred to as the *musical form*. The musical form can be seen as the decomposition of a song into its major building blocks, corresponding to the previously introduced notion of musical *paragraphs*. When formally describing the musical form, every building block has its own label and can occur at various times throughout the song. Typical labels in popular music are for example *intro*, *verse*, *chorus*, *bridge* and *outro*.

Although the exact determination of the musical form is not always unambiguous, most people will unconsciously split songs into closely related blocks when listening to music. Therefore, detecting the musical form has been found to be a reasonable and natural objective for automatic MSD systems [LS08].

1.1.1 Existing Work

Since the creativity of composers is the only limit for the variety of differences and similarities among song segments, several different approaches to solve the problem of music structure discovery have been developed. A very detailed overview on existing systems is given in [PMK10], therefore, only the main concepts will briefly be mentioned at this point.

Typically, multiple features that have been found to be adequate descriptors for either one or several different aspects of human cognition of music, are initially derived from a short time spectral representation of the audio file. Therefore, the most commonly used features in MSD are often based on timbre [PLBR02], [Foo00], pitch and harmony [BW05], rhythm or a set of multiple descriptors [LS08], [Pee07], [PK09], [WSS09].

Once the feature sequences are extracted, the search for repetitive parts and related sections can mainly be focused on two different temporal qualities of the investigated audio file: *sequences* and *states* [Pee07].

Sequence based approaches Sequence-based approaches try to find clear repetitions of consecutive feature sequences in the audio file [BW05], [Pee07], [Ero07]. Therefore, these algorithms are sometimes also referred to as *repetition-based* methods [PMK10].

State based approaches State-based approaches handle the feature sequence as a succession of different (similar) states and try to find relations, for example, by applying clustering algorithms [ANSS05] or hidden Markov models (HMM) [PLBR02], [AS01]. As transitions between segments are detected by finding transitions between homogeneous sections, these methods are sometimes also called *homogeneity-based methods* [PMK10].

In Section 3 it will be shown in more detail, how these two basic qualities of a feature sequence (repeated and/or homogeneous sections) become apparent, when a Self Distance Matrix (SDM) is used to visualize and analyze the temporal structure of the underlying song. Given a feature vector sequence $\boldsymbol{V}$ consisting of single feature vectors $\boldsymbol{v}_i, i = 1, 2, \ldots, N$, each value of the SDM $\boldsymbol{S}(i, j)$ represents the distance between a pair of feature vectors. Off-diagonal lines inside $\boldsymbol{S}$ correspond to the repetition of a certain *sequence* of consecutive feature vectors, whereas rectangular blocks represent the occurrence of multiple overall similar feature vectors, potentially belonging to the same *state*. More details on these aspects of SDMs are given in Section 3.1.

1.1.2 Applications

Although the development of MSD systems can be motivated by various preliminary considerations, they will basically all aim to focus some action (e.g. data analysis, feature extraction, similarity estimation, music analysis) to specific sections of a song. Specifically in MIR, the musical structure offers important side information to other algorithms, as any further processing of the piece can be bound to specific time intervals.

Audio Thumbnailing Audio thumbnailing describes the process of finding an expressive preview sequence of a song, based on one or multiple representative sections. In MIR, multiple systems for audio thumbnailing have been proposed [MLC06], [BW05], [ZS07], all depending more or less on the musical structure of the underlying song. Therefore, audio thumbnailing is a typical MIR task directly profiting from prior knowledge on the musical structure.

Segment related feature extraction Besides an obvious application like the generation of thumbnails, almost any algorithm developed to extract some higher-level musical information from a piece of music can profit from prior knowledge on the musical structure. Unrepresentative sections can be ignored and dependencies among sections can be exploited to improve the individual results, for example, by pairwise comparison of related sections.

Based on these considerations, the experiments presented in Section 7 are also motivated by the aim of focusing a typical MIR task to specific sections of the audio file. The detected musical structure will be used to investigate the influence of focusing the audio analysis to distinct representative song segments, when trying to find a genre label for a song.

Segment related search Algorithms searching for specific sequences inside databases of audio data like *query by humming* [WHH$^+$08], could profit from pre-segmented audio files, as only one occurrence of each repetitive section of the file has to be stored and processed. This may not only reduce the computational cost for many applications, but also provide faster and more stable results.

Augmented user interfaces In addition to the proposed MIR applications, the results from MSD system could also be used to improve the usability of a wide range of existing audio applications, for example, by allowing a more intuitive navigation within pieces when using audio players or Digital Audio Workstations [Got06], [BGP06]. Programs working with labeled sections of an audio file would also allow to easily skip certain sections, or constrain specific modifications to selected segments or borders within the file.

Although this is only a short overview on some potential fields of application where MSD systems could be used as a pre-processing stage, it shows their high potential in various fields of music related signal processing.

1.2 Outline of the Thesis

In this diploma thesis, a novel approach for the detection of recurring sections within pieces of music that should be closely related to the musical form is introduced. Most Sections will represent a major stage of the proposed MSD system and will start with a short introduction on the techniques used to implement the respective part of the algorithm.

Audio Feature Extraction (Section 2) As already introduced, the search for repeated sections within an audio file has to be based on sequences of extracted feature vectors. For the presented approach, the well known Mel frequency cepstral coefficients (*MFCC*)

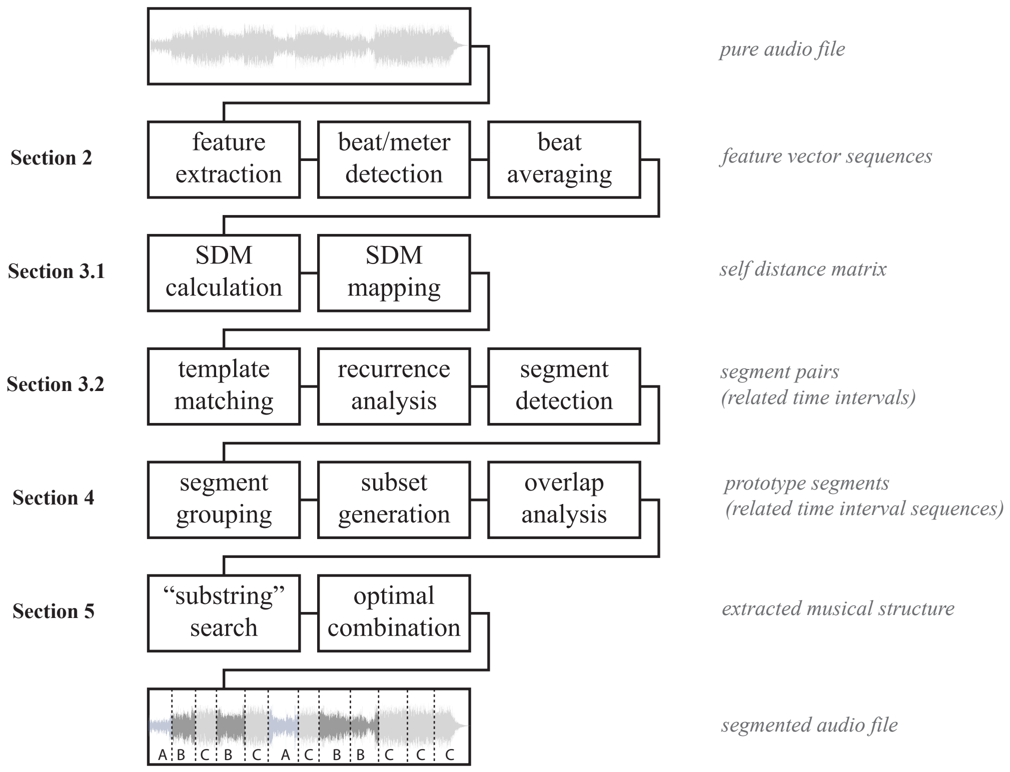


Figure 1: Overview on the main processing stages of the algorithm.

are used to represent timbral qualities of the audio material, the chroma or pitch class profile (PCP) as an abstract descriptor for pitch and harmonic progression and the so called fluctuation patterns (*FP*) as indicator for rhythmic changes.

Additionally, the extracted features are averaged within the period of one beat, to offer a tempo-invariant time base for further computations and a stable representation of the extracted features.

Recurrence Analysis (Section 3) Multiple techniques to search for repetitive patterns and structures inside feature vector sequences to find the musical structure of a song have been developed. The proposed system will focus on a representation of the feature vectors based on Self Distance Matrices (SDM) which are a special form of recurrence plots (RP).

As already mentioned before, MSD systems based on the analysis of SDMs typically focus on two aspects of repetitions, *states* and *sequences*, which can be related to off-diagonal lines (repeated sequences) and rectangular blocks (areas of high similarity) inside the SDM. This may lead to problems, as many SDMs reveal very complex structures and clear lines or blocks will not always be detectable. Therefore, these algorithms are somehow limited and are likely to fail, if no typical structure can be found.

In this thesis it is proposed to directly exploit the inherent symmetry of SDMs, by comparing the overall similarity of vertical SDM slices using a template matching algorithm. This approach is independent of any specific structures inside the matrix and enables the algorithm to find recurrences even if they are not related to obvious patterns in the original SDM.

Template Matching (Section 3.2) Before a template matching algorithm is applied to find repetitions of vertical slices inside the SDMs, noisy information inside the computed SDMs is further reduced by a mapping operation that helps to enforce areas of high similarity while it suppresses areas of low similarity. The individually mapped SDMs are then combined to represent the final "image" for the template matching algorithm.

For template matching, Normalized Cross Correlation is applied, a well known template matching algorithm used for image registration in digital image processing. The correlation results are processed and stored into a matrix, the repetition surface, representing all segment pairs found by the template matching algorithm. Based on the relations of all found pairs, segment pairs representing different repetitions of the same section of a song are grouped into subsets.

Time Interval Data (Section 4) Since the found subsets will typically reveal overlapping parts, an algorithm to resolve conflicts of overlapping time-interval data is designed and applied to the given subsets. The algorithm aims to exploit all relations of repetitions throughout the song, as changes to one segment of a subset should have a direct impact on all subset members. Therefore, each subset is only represented by one prototype segment and its related starting positions. Changes to this prototype segment automatically modify all occurrences of the related subset.

Temporal Pattern Mining (Section 5) After a preliminary segmentation has been found by the proposed algorithm, a post processing step helps to further investigate the preliminary segmentation by searching for repeated sub-sequences, as such sequences indicate potentially over segmented areas.

The search for an optimal combination of all found sub-sequences can be formulated as a path search in a weighted graph. By defining quality measures for each vertex and edge of the graph, based on the lengths as well as the inner and inter similarity of all possible sub-sequences, an optimal path through the graph (optimal combination of segments inside repeated sub-sequences) can be found. After incorporating the found changes into the preliminary segmentation, the final segmentation of the song can be defined.

Evaluation and Results (Section 6) The *boundary detection* performance and the *segmentation* performance of the algorithm have been evaluated on a well known data set of more than 170 songs from 13 Beatles albums. The quality of the automatically detected segment boundaries is calculated using classical F-measure and the full segmentation performance of the algorithm, including lengths and relations of all segments, is calculated using pairwise F-measure. To the best of our knowledge, the obtained F-measures are the highest ever reported for the used data set.

Segment Related Genre Classification (Section 7) In a complementary experiment, the detected musical structure is used to investigate the influence of focusing the audio analysis to distinct representative song segments, when trying to find a genre label for a song. The results indicate that this additional knowledge can help to significantly improve the results of the classification system when focusing on representative sections instead of arbitrary parts, if very short excerpts (< 10 seconds) are used for classification. For longer excerpts, no difference for the classification of representative and arbitrary parts has been found.

2 Audio Feature Extraction

As mentioned in the introductory Section 1.1.1 on existing work, the search for repeated or homogeneous sequences within the audio file has to be based on sequences of extracted feature vectors. This has to be done since not a lot of perceptually relevant information can be directly extracted from the pure waveform of an audio file.

In the presented approach, the well known Mel frequency cepstral coefficients (*MFCC*) are used to represent timbral qualities of the audio material, the chroma or pitch class profile (PCP) as an abstract descriptor for pitch and harmonic progression and the so called fluctuation patterns (*FP*) as indicator for rhythmic changes.

As a pre-processing step, the signal is converted to mono and resampled to 22.050 Hz to reduce the computational cost for all following processing steps. As the selected features all rely on a short time spectral representation of the input signal, a Hanning windowed Short Time Fourier Transform (STFT) of length $N_{STFT} = 1024$ is computed. To assure a constant temporal resolution of $\approx 23ms$ the hop-size between adjacent frames is set to $k_{STFT} = 512$.

$$X[m,k] = \sum_{n=0}^{N_{STFT}-1} x[n]w[n-m]e^{-j\frac{2\Pi nk}{N}}, \quad k = 0,\dots,N_{STFT}-1 \tag{1}$$

where $w[m]$ is the Hanning window of length N_{STFT} positioned at time index m.

It is well known that the frequency resolution of the human auditory system is approximately logarithmic in frequency and therefore collides with the linear frequency resolution of the Discrete Fourier Transform (DFT) [FZ07]. Therefore, the calculation of features designed to mimic the human perception of music typically has to be based on some auditory scale, representing a mapping from the linear frequency scale to an approximately logarithmic and perceptually more relevant scale.

In the next short Sections outlining the computation of the used features, it will be shown that all three features incorporate this discrepancy in frequency resolution: In analogy to their name, MFCC coefficients are based on the Mel scale [SV40], the FP are calculated based on a further compressed Mel scale and since chroma vectors are descriptors for the pitch and harmonic content, they are calculated based on a filterbank resembling semitones of the equal tempered scale.

2.1 Mel Frequency Cepstral Coefficients (MFCC)

MFCCs have been used in audio and speech applications for many years and have been found to be relatively simple but powerful descriptors for the timbral qualities of sound [Ero01], [LWOO08]. Although the definition of timbre is ambiguous due to subjective ways of interpreting the term, for musical signals it may best be explained as the property resembling differences of audio signals despite the same pitch and rhythm.

Equation (2) shows the mapping from a linear frequency scale to the Mel scale. The resulting mapping function is depicted in Figure 2. In the presented approach 36 Mel filters are applied to transform the linear power spectrum $\|X(m,k)\|^2$ to a Mel power spectrum.

$$Mel(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2}$$

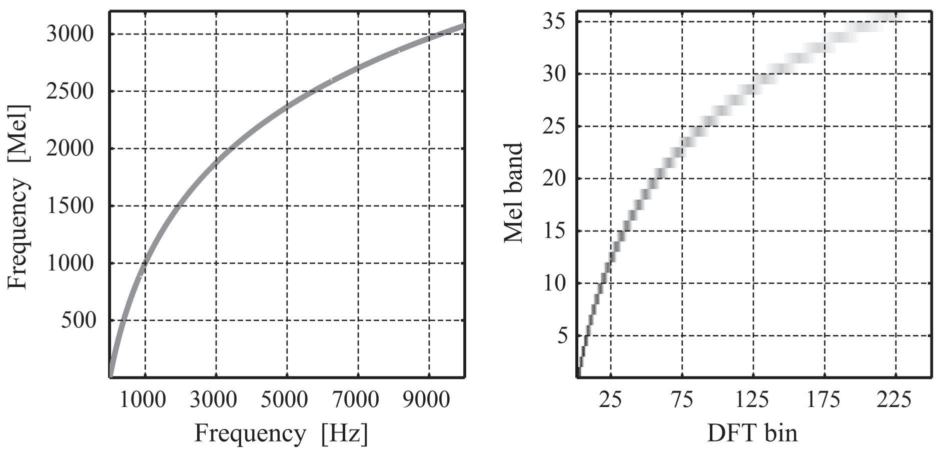


Figure 2: Mapping function from the linear frequency scale to the Mel Scale (left) and grouping of DFT bins according to 36 triangular Mel filters (right).

The descriptive power of MFCCs is based on the fact that the the computation incorporates two transformations of the linear spectrum according to important properties of the human auditory system:

- a mapping from the linear frequency scale to an auditory scale (the Mel scale, resembling the human perception of pitch)
- a representation of the mapped values by their logarithm (following the human perception of loudness)

As MFCCs are a widely used and extensively discussed feature, no detailed description of all precessing steps will be given here. For a more detailed description, please refer to [Sla98]. It may be noted that 10 coefficients are used in the presented algorithm, excluding the zeroth coefficient (the average energy of a frame).

2.2 Fluctuation Patterns (FP)

Fluctuation Patterns (also called rhythm patterns) have been introduced in [PRM02] to describe the perceived fluctuation of amplitude modulated tones. The term fluctuation strength was first mentioned in [Fas82] and describes different sensations, depending on the modulation frequency. According to [FZ07], the perceived fluctuation strength has been found to be most intense around $\approx$ 4Hz and decreases until $\approx$ 15Hz, where the notion of roughness starts. In the field of MIR, FPs have been used to cover similarities among audio files (or within audio files) which are not included in other typical spectral descriptors.

Although the calculation of FPs was initially based on the ERB scale, where each band was additionally converted to Sone (perceived loudness), the used system follows an approach presented in [Pam06], based on a Mel spectrogram. This is done, as the Mel spectrogram is readily available from the calculation of the MFCCs. A detailed description of the calculation of FPs is given in [Pam06].

Since the fluctuation of tones can only be determined by observing longer sequences of multiple consecutive frames, FPs are not a frame level feature. As all features will later be averaged over the period of one beat and the distances of features will be calculated over one bar (see Section 3.1), FPs are directly calculated within windows of one bar. This leads to observation windows between ≈ 1.5 and ≈ 4 seconds or ≈ 70 and ≈ 200 frames (see constraints for valid tempos in Section 2.4). To receive the individual modulation frequencies, a 64 point Fourier Transform is applied to each band.

It may be noted that the resulting fluctuation patterns (see Figure 3) are serialized and directly stored as column vectors.

2.3 Constant Q-Transform (CQT) and Chroma

The constant-Q spectrogram based on the constant-Q transform of a signal, can be interpreted as the output of a geometrically spaced filterbank, where the ratio $Q = {}^{f}/_{\Delta f}$ of center frequency to bandwidth of each filter is defined to be constant [Bro90]. Con-

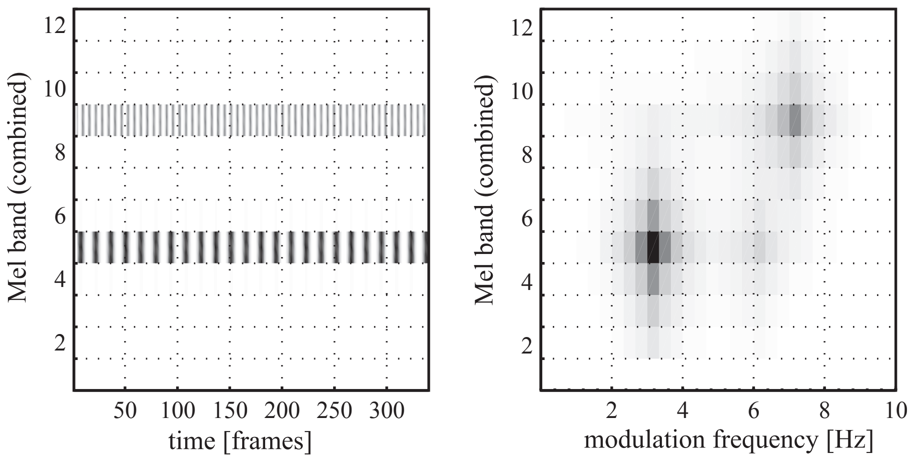


Figure 3: Compressed 12 band Mel spectrogram (left) and resulting fluctuation pattern (right) calculated for a combined signal with one 300 Hz tone modulated by a 3 Hz sine (band 5) and a 800 Hz tone modulated by a 7 Hz sine (band 9).

sidering the frequency relations of the equal tempered scale, the center frequencies of the individual filters can be calculated as follows to represent any division of the musical octave:

$$f_{k_{cq}} = 2^{\frac{k_{cq}-1}{B}} f_{min} \tag{3}$$

where $f_{k_{cq}}$ represent the individual center frequencies, B determines the number of bins per octave and f_{min} represents the center frequency of the lowest bin of interest. Based on the tradition of occidental music[2], the individual center frequencies have been aligned to the semitone scale by selecting $B = 12$ bins per octave and setting f_{min} to the center frequency of the lowest semitone, leading to a representation of signal energy distributed on a semitone scale.

Based on these parameters, the constant-Q transform of a signal $x[n]$ could be directly calculated as

$$X^{cq}[k_{cq}] = \sum_{n=0}^{N[k_{cq}]-1} w[n, k_{cq}] x[n] e^{-j\omega_{k_{cq}} n}, \tag{4}$$

where $w[n, k_{cq}]$ describes a windowing function of length $N[k_{cq}]$ that equals a bandpass filter in the frequency domain, centered at frequency $\omega_{k_{cq}}$. As the bandwidth of the respective filter is inversely proportional to $N[k_{cq}]$ and the Q for all filters should remain

[2]Even though there do occur notes like the harmonic seventh that are derived from the overtone scale and would fit a quarter tone scale more tightly [BR87].

constant, the window lengths $N[k_{cq}]$ are chosen inversely proportional to the related center frequency.

As the direct calculation of the constant-Q transform is computationally expensive, it has been shown that the Fourier transform of the signal and the corresponding windowing functions, referred to as the spectral kernels, can be used to efficiently calculate the transform in the frequency domain [BP92].

Additionally, similar to the approach presented in [SK10], octave-wise processing was implemented. This is done, as very long Fourier transforms would be needed to represent a semitone resolution in low frequency bands. For octave-by-octave processing, only one set of spectral kernels representing the constant-Q filters for the highest octave is calculated. After the highest octave of the entire signal is transformed, the signal is decimated by a factor of 2 and a new STFT using the same parameters as before is computed.

Since the highest octave of the decimated signal represents the octave below the octave processed in the previous step, the same spectral kernels are applied, resulting in the constant-Q transform of the respective octave. This can be repeated, until the lowest octave of interest has been computed.

As the resulting constant-Q spectrograms are based on different sampling rates, missing values for lower octaves are simply linearly interpolated, resulting in a constant-Q spectrogram with high temporal and spectral resolution.

Chroma The chroma or Pitch Class Profile (PCP) has been demonstrated to be a successful basic indicator of the harmonic and melodic progressions of music as it measures the spectral energy related to the 12 semitones of the well-tempered scale albeit their real-world frequencies [BW01]. Based on the previously calculated CQT spectrogram, the chroma vector can be computed by summing the energy of all bins belonging to one tone $c(k_{cq}) = mod(k_{cq}, 12)$.

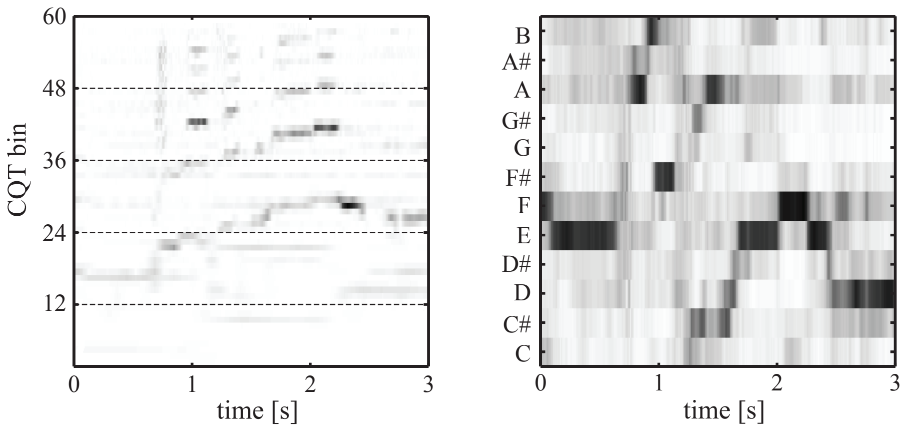


Figure 4: Constant Q-spectrogram based on a semitone scale (left) and the related chromagram (right). The separation of the constant Q-spectrogram into 5 octaves is indicated by horizontal lines.

2.4 Beat Detection and Averaging

After feature calculation, the features are averaged over the period of one beat, as beat-averaging offers a tempo-invariant time base for further computations and therefore offers a stable representation of the extracted features. The beat is often defined as the rhythm or tempo, one would intuitively start tapping with a foot or finger while listening to a piece of music. It can therefore be seen as the elementary temporal unit of a song.

The beat-averaged feature vectors (MFCC, chroma) are computed by calculating the mean of all frames falling between two beat positions. To avoid a blurring of these beat averaged results by transient events, an offset of about $45ms$ (two frames) at the beginning and end of every beat interval has been defined.

For beat detection, a method proposed by Ellis [Ell07] based on dynamic programming has been selected. In this approach, beat positions are found by incorporating two constraints derived from the human perception of tempo:

- (approximately) constant inter-beat intervals
- beat locations at onsets of audio events

To get an estimate for the overall tempo, an onset strength envelope $\boldsymbol{o}(t)$ based on changes of energy at the output of a 40 band Mel filterbank is calculated in a first step (see Figure 5). Maxima in $\boldsymbol{o}(t)$ correspond to potential onset positions. An estimate of the overall

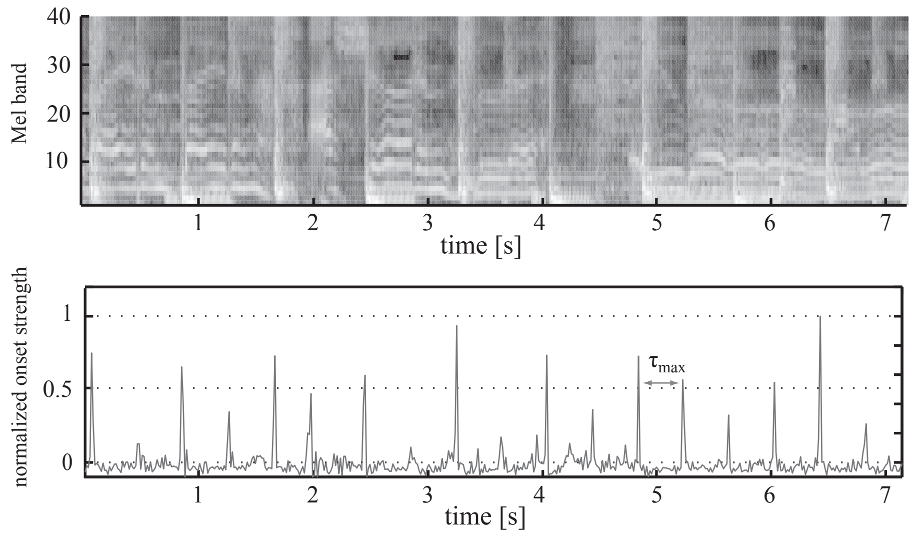


Figure 5: Based on a 40 band Mel spectrogram (top), the onset detection function $\boldsymbol{o}(t)$ (bottom) is computed.

tempo of the investigated song is then extracted, by calculating the autocorrelation $\boldsymbol{a}(\tau)$ of $\boldsymbol{o}(t)$ (see Figure 6).

$$\boldsymbol{a}(\tau) = \sum_t \boldsymbol{o}(t)\boldsymbol{o}(t-\tau) \tag{5}$$

As the human perception of tempo is biased towards 120 bpm [MM06], $\boldsymbol{a}(\tau)$ is weighted by a perceptual window. The maximum in this weighted auto correlation function $\boldsymbol{a}_w(\tau_{max})$ corresponds to the detected tempo period and allows for an estimate of the global tempo (see Figure 6).

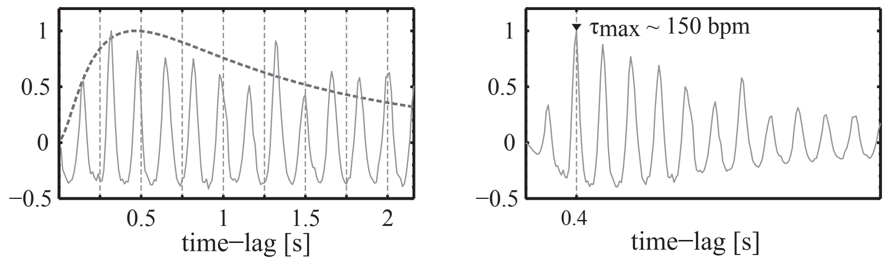


Figure 6: The raw autocorrelation $\boldsymbol{a}(\tau)$ of $\boldsymbol{o}(t)$ and the perceptual weighting window (left) are combined into a perceptually weighted autocorrelation $\boldsymbol{a}_w(\tau)$ (right).

Finally, dynamic programming is used to maximize an objective function by positioning as many beat locations as possible at potential onsets. For more details on the computation of $\boldsymbol{o}(t)$ and the dynamic programming stage, please refer to [Ell07].

An additional constraint has been incorporated to the tempo estimation step, by restricting the range of valid beats-per-minute between 50 and 160 *bpm*. This is done as the following beat averaging process should lead to relatively uniform temporal resolutions. Values under the lower threshold are doubled and values above the upper threshold are divided by 2.

2.4.1 Meter Induction

As not only the beat of the investigated song, but also the underlying meter offers important information for the following segment detection stage, the proposed system has been extended by a simple meter induction algorithm. The search for the meter related to the song is mainly motivated by the aim of finding musically relevant temporal thresholds for different stages of the segment detection algorithm.

It is well known that structures of 2, 4 and 8 bars are important building blocks in many compositions [Dav66]. Therefore, upper thresholds (minimum lengths) and lower thresholds (maximum lengths) for the segment detection algorithm (e.g. defining the minimum length of "relevant" structures) can be defined based on these durations. To allow deviations from the exact theoretical values, an additional tolerance of $\pm 15\%$ for both types of thresholds is included.

Example: thresholds related to 8 bars for a song played in common time:

- $t_8 \rightarrow$ plain threshold (exact length): $(8 * 4\ beats) = 32$ beats
- $t_{<8} \rightarrow$ upper threshold (minimum length): $(8 * 4\ beats) * 0.85 \approx 27$ beats
- $t_{>8} \rightarrow$ lower threshold (maximum length): $(8 * 4\ beats) * 1.15 \approx 37$ beats

For meter induction, a relatively simple algorithm, similar to the one presented in [Eck05], has been developed. Songs will typically not only exhibit periodicities based on the level of one beat, but also on the level of multiple beats forming one bar and multiple bars forming one phrase. Therefore, one can expect that besides the correlation maximum $\boldsymbol{a}(\tau_{max})$ related to the found tempo or one beat-period, additional higher-level periodicities reflecting the number of beats per measure $\boldsymbol{a}(\tau_1)$, two measures $\boldsymbol{a}(\tau_2)$ and four measures $\boldsymbol{a}(\tau_4)$ may also represent prominent peaks in $\boldsymbol{a}(\tau)$. For example, the autocorrelation function calculated for a song played in common time (4/4) will typically show high autocorrelation values for lags related to the period of 1, 4, 8 and 16 beats. Therefore, autocorrelation functions showing high values at these characteristic lags are very likely to belong to a song played in common time (see Figure 7).

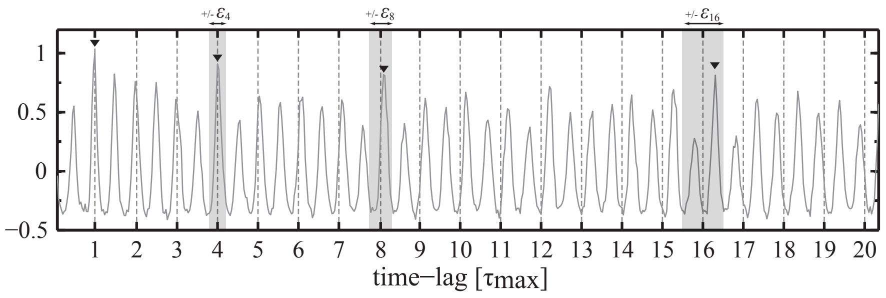


Figure 7: Based on characteristic maxima inside $\boldsymbol{a}(\tau)$, the meter of a song can be estimated. This example shows $\boldsymbol{a}(\tau)$ for a song played in common time. Therefore, additional correlation maxima (besides the maximum at τ_{max}) are expected at the following characteristic multiples of τ_{max}: $4, 8$ and 16.

This observation can be generalized by evaluating multiple characteristic lag-combinations representing a list of candidate meters. In the given approach, the correlation values in $\boldsymbol{a}(\tau)$ according to one, two and four measures for every meter type are evaluated. Therefore, the types of relevant meters are defined in a vector $\boldsymbol{b} = [2, 3, 4, 6]$, according to the related beats per measure. Based on this vector, the cumulative sums for each measure type $m = 1, \ldots, 4$ can be calculated as follows:

$$\boldsymbol{\mu}(m) = \sum_{k=[1,2,4]} max\{\boldsymbol{a}\left([\boldsymbol{b}(m) * k * \tau_{max} - \epsilon_{k_m}, \cdots, \boldsymbol{b}(m) * k * \tau_{max} + \epsilon_{k_m}]\right)\} \quad (6)$$

where ϵ_{k_m} is an adaptive threshold allowing deviations from exact multiples of the beat period of $\pm\ 5\%$, to account for small deviations in periodicity. An estimate of the underlying meter can then be defined by finding the argument m (measure type) related to the largest cumulative sum.

$$m_{max} = \arg\max_{m}\{\boldsymbol{\mu}(m)\} \quad (7)$$

3 Recurrence Analysis

As already mentioned in Section 1.1.1, multiple techniques to search for repetitive patterns and structures inside feature vector sequences to find the musical structure of a song have been developed. The presented approach focuses on a representation of the feature vectors based on Self Distance Matrices (SDM) which are a special form of recurrence plots (RP).

Recurrence plots have initially been developed for nonlinear data analysis [GC00]. Given a multi-dimensional feature vector sequence (e.g. data from multiple sensors), the temporal trajectory evolving from successive measurements of this sequence is observed. The multi dimensional space traversed by these trajectories is often referred to as the phase space. Individual segments of the trajectory following a similar path in phase space reveal a certain recurrent behavior of the system. If the distance of such paths lies below a certain threshold over a certain temporal interval, these intervals are defined to be similar and are marked in a binary matrix. The concept of RPs therefore allows to visualize and intuitively analyze the recurrent behavior of dynamical systems.

Given a sequence $\boldsymbol{V} = \{\boldsymbol{v}_i | i = 1, \ldots, N\}$ the RP can be calculated as

$$\boldsymbol{R}(i,j) = \begin{cases} 1, & \text{if } \|\boldsymbol{v}_i - \boldsymbol{v}_j\| < \epsilon, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

where ϵ is a threshold distance and $\|\cdot\|$ denotes the norm between two vectors.

An example showing the relation of a simple 2-dimensional temporal trajectory and its related RP is depicted in in Figure 8.

An additional parameter in RP analysis is the embedding dimension ρ. It defines how many time instances of a feature sequence are combined to calculate the RP. If $\rho > 1$, $\boldsymbol{R}(i,j)$ is constructed by computing distances between all pairs of embedded vectors $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$.

$$\overbrace{\left[\boldsymbol{v}_i^T, \boldsymbol{v}_{i+1}^T, \ldots, \boldsymbol{v}_{i+(\rho-1)}^T\right]}^{\boldsymbol{e}_i} \qquad \overbrace{\left[\boldsymbol{v}_j^T, \boldsymbol{v}_{j+1}^T, \ldots, \boldsymbol{v}_{j+(\rho-1)}^T\right]}^{\boldsymbol{e}_j}$$

The simplified idea behind the embedding dimension of recurrence plots states that each single observed parameter of a dynamic system (e.g. air pressure) contains important information about the dynamics of the whole system (e.g. weather). By defining an "adequate" embedding dimension, the behavior of the overall system can be reconstructed by only using one embedded parameter.

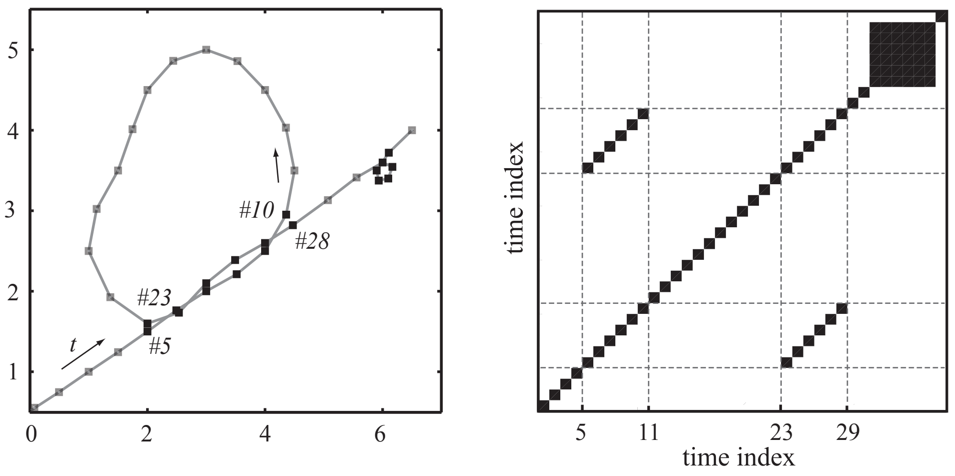


Figure 8: Trajectory of a 2-D sequence and the resulting RP (ϵ was set to 0.5 in this example).

3.1 Self Distance Matrix (SDM)

As already mentioned, many MSD systems use a similar representation to find recurrent parts in feature vector sequences, the Self Distance Matrix (SDM) [Foo00], [BW05], [PK06]. In contrast to RPs, SDMs do not use a threshold parameter ϵ, but directly save the resulting distance between individual pairs of vectors into a matrix, so SDMs are not binary.

In [PK06] it has been proposed to use measure level similarity matrices instead of frame or beat based matrices for MSD, as bars are the smallest natural building blocks of a higher-scale musical structure. The presented approach is inspired by a very similar idea based on concept of the embedding dimensions known from recurrence plots (RP).

The typical form to calculate an SDM can be outlined as follows:

$$\boldsymbol{S}(i,j) = \|\boldsymbol{v}_i - \boldsymbol{v}_j\| \quad \text{or embedded} \quad \boldsymbol{S}(i,j) = \|\boldsymbol{e}_i - \boldsymbol{e}_j\| \tag{9}$$

Although music is not a natural dynamic system, its recurring structure reveals clear dependencies on multiples of beats and bars. As one bar may be considered to be the smallest relevant building block for a musical structure, one can focus the recurrence analysis to the level of bars by using an embedding dimension according to the found meter. For example, when analyzing songs written in common time, ρ will be set to 4, corresponding to the 4 beats forming one bar. Larger values of ρ will tend to suppress too much detail in the recurrence structures of the investigated sequences, while smaller values may result in too much noisy, short term structures in $\boldsymbol{S}$. An example of the effect of using an embedding dimension $\rho > 1$ can be found in Figure 9. One can see that increasing the embedding dimension clearly enhances the clarity of structures in the SDM, while noisy parts are suppressed.

In the presented system, the Euclidean Distance is used for computing the three bar level distance matrices $\boldsymbol{S}_{MFCC}$, $\boldsymbol{S}_{chroma}$ and $\boldsymbol{S}_{FP}$.

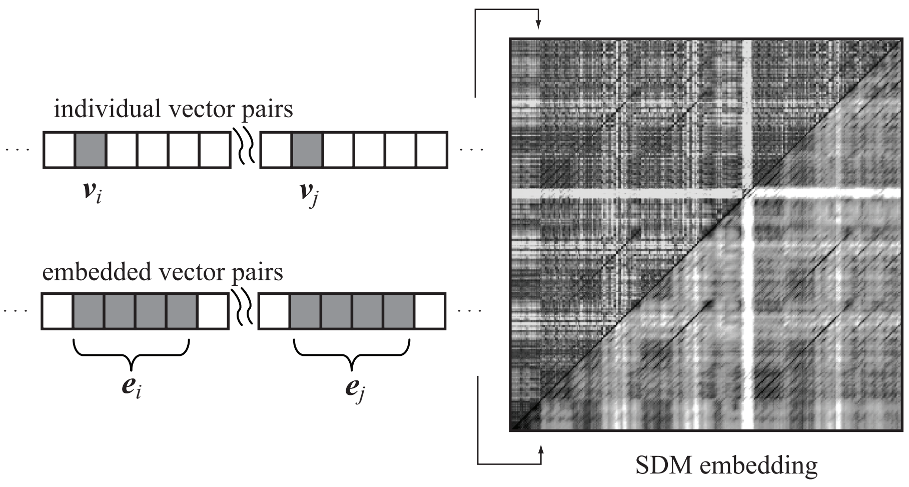


Figure 9: Example for an SDM calculated based on individual vector pairs (upper triangular matrix) and an SDM calculated based on embedded vectors and an embedding dimension of 4 (lower triangular matrix).

When analyzing SDMs, it is important to know something about the main characteristics and typical structures these matrices will reveal:

- SDMs are symmetric with respect to their main diagonal
- the axes of the SDMs used in the presented approach are both time axes with the unit of one beat
- clear repetitions of individual sub-sequences of a feature vector sequence lead to off diagonal lines
- sub-sequences with a high inner similarity become apparent as square blocks

Figure 10 shows how the mentioned structures (off diagonal lines, square block) can occur in a RP for a simple 1-D sequence.

Many MSD algorithms focus on one or the both of these structures when analyzing SDMs. This may lead to problems, as many SDMs reveal very complex structures and clear lines

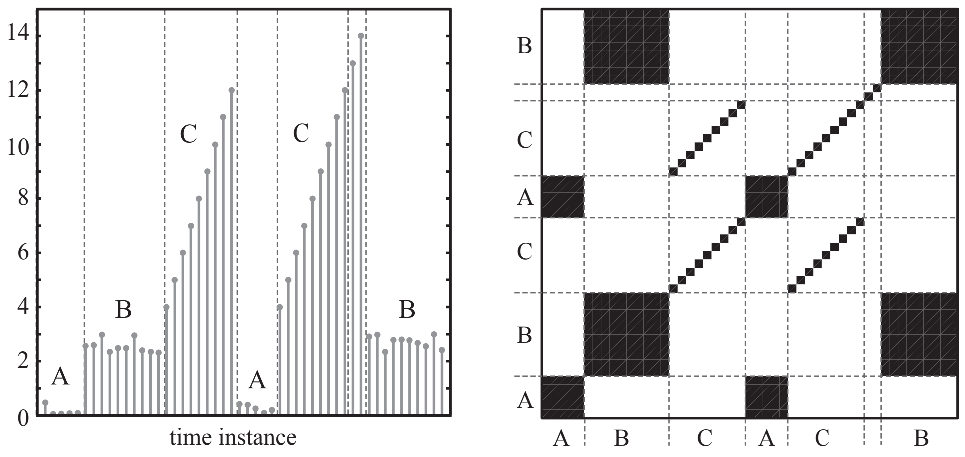


Figure 10: Example for the manifestation of characteristic structures (off diagonal lines, square blocks) in a RP, based on a simple 1-D sequence.

or blocks will not always be detectable. Therefore, these algorithms are somehow limited and are likely to fail, if no typical structure can be found.

To directly exploit the inherent symmetry of SDMs, the presented system is based on the comparison of the overall similarity of vertical SDM slices [WCB10]. This approach is independent of any specific SDM structures and enables the algorithm to find recurrences even if they are not related to obvious patterns in the original SDM. As normalized SDMs can be interpreted as grayscale images, the search for recurring vertical slices can be performed by template matching algorithms known from image processing (see Section 3.2).

3.1.1 SDM Mapping and Combination

To improve the "image" for the template matching algorithm, noisy information inside the matrices $\boldsymbol{S}_{MFCC}$, $\boldsymbol{S}_{chroma}$ and $\boldsymbol{S}_{FP}$ is further reduced by a mapping operation. The SDMs are normalized to unit interval $[0, 1]$ and mapped individually based on a continuous function that helps to enforce areas of high similarity while it suppresses areas of low similarity (see Equation 10 and Figure 11). The individually mapped SDMs are then combined to represent the final "image" for the template matching algorithm.

$$\boldsymbol{S}_{map} = 0.5 - 0.5 * \tanh\left[\pi * \lambda * (\boldsymbol{S} - \gamma)\right] \tag{10}$$

To find an adequate threshold γ for this mapping operation, an iterative algorithm has been developed based on *Otsu's method.* Otsu's method is a widely used histogram based image thresholding technique that tries to minimize the intraclass variance of two classes of variables to find a good threshold for the separation of data values in grayscale images. For more detailed information, please refer to [Ots75].

As the thresholding operation should be focused on the relevant range of values in the normalized matrix $\boldsymbol{S}$ (areas representing high similarities), the search for an optimal threshold is initialized based on all values in $\boldsymbol{S}$ smaller than 0.5. A schematic overview of the following iterative search for an optimal mapping threshold κ is listed in Algorithm 1.

Algorithm 1 Iterative search for mapping threshold

```
1: $r \leftarrow 0$
2: $\kappa \leftarrow 0.5$
3: $\mu \leftarrow 0.01$
4: $thr_{min} \leftarrow 0.05$
5: $thr_{max} \leftarrow 0.1$
6: while $r < thr_{min}$ or $r > thr_{max}$ do
7:     $\gamma \leftarrow getThreshold\,[\boldsymbol{S}, \kappa]$
8:     $\boldsymbol{S}_{map} \leftarrow performMapping\,[\boldsymbol{S}, \gamma]$
9:     $r \leftarrow evalRecurrence\,[\boldsymbol{S}]$
10:    if $r < thr_{min}$ then
11:        $\kappa \leftarrow \kappa + \mu$
12:    else
13:        $\kappa \leftarrow \kappa - \mu$
14:    end if
15: end while
```

The algorithm starts with Otsu's method to find an initial value of γ (Line 7) and performs the mapping operation (Line 8). To evaluate the quality of the resulting mapped matrix $\boldsymbol{S}_{map}$, a quality measure inspired by the *recurrence rate* of RPs (a measure known from quantitative recurrence analysis [GC00]) is computed (Line 9). As SDMs are not binary, the resulting value r is called *overall recurrence*

$$r = \frac{1}{N^2} \sum_{i,j}^{N} \boldsymbol{S}_{map}(i,j), \tag{11}$$

where N is the number of rows and columns of $\boldsymbol{S}_{map}$.

Large values of r indicate (too) many high values in $\boldsymbol{S}_{map}$ and call for a stricter mapping, as more non-relevant elements in $\boldsymbol{S}_{map}$ should be suppressed. In contrast, low values of r indicate that too many elements in $\boldsymbol{S}_{map}$ have been suppressed and suggest a more relaxed mapping. Although the structures in SDMs of different songs can significantly differ, experiments showed that *overall recurrence* values between 0.05 and 0.1 lead to clear matrices, keeping important structures while adequately suppressing noisy areas.

By iteratively adapting κ with a step size μ (Lines 11 and 13), the range of values used to find the mapping threshold γ is re-defined at every iteration, until the *overall recurrence* r of the resulting matrix $\boldsymbol{S}_{map}$ falls into the previously defined range $[0.05 < r < 0.1]$ (Line 6).

Depending on the resulting threshold γ, the parameter λ is set automatically to map 0 to 1. Although the resulting matrix looks similar to a binary recurrence plot, the continuous mapping preserves the fine structure of highly repetitive areas, which is important for template matching (see Figure 11).

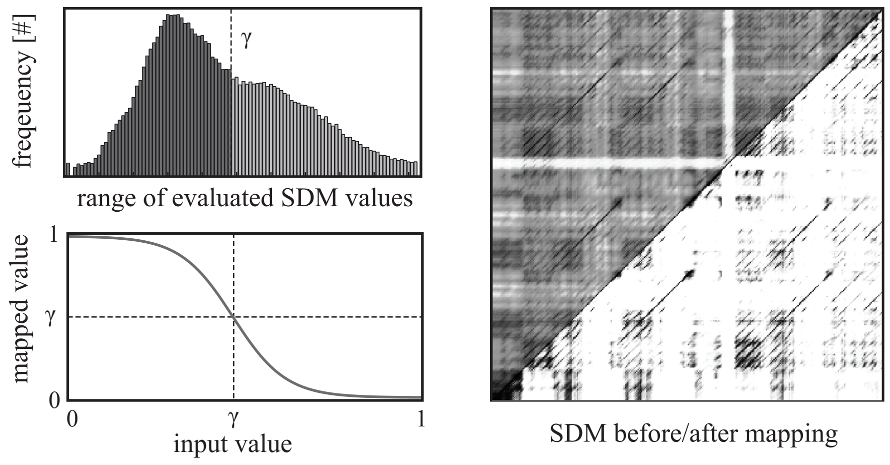


Figure 11: Based on a histogram (left, top), Otsu's method finds the optimal threshold to separate the given values into two classes. The found separation point γ is used as input parameter for the mapping function (Equation 11) to perform the mapping of the given SDMs.

SDM combination As three different SDMs have been calculated based on the three feature vector sequences ($\boldsymbol{S}_{MFCC}$, $\boldsymbol{S}_{chroma}$ and $\boldsymbol{S}_{FP}$), these individually mapped SDMs have to be combined into one matrix $\boldsymbol{S}_{cmb}$. This is done by simple point-wise summation, again followed by a normalization to unit interval.

$$\boldsymbol{S}_{cmb} = \frac{\boldsymbol{S}_{map,MFCC} + \boldsymbol{S}_{map,chroma} + \boldsymbol{S}_{map,FP}}{max\{\boldsymbol{S}_{map,MFCC} + \boldsymbol{S}_{map,chroma} + \boldsymbol{S}_{map,FP}\}} \tag{12}$$

3.2 Template Matching

Based on the combined, mapped matrix $\boldsymbol{S}_{cmb}$ representing the recurrent behavior of all investigated features, Normalized Cross Correlation (NCC), a template matching algorithm, can now be used to find repetitive vertical slices inside the matrix. As already mentioned, by exploiting the inherent symmetry of SDMs, the template matching algorithm compares different similarity profiles (vertical SDM slices) and makes the detection of repetitions widely independent of any distinct structures in the matrix.

3.2.1 Normalized Cross Correlation (NCC)

One famous and widely used algorithm in the field of template matching (sometimes called image registration) is Normalized Cross Correlation (NCC) [Lew95]. Using NCC for finding a template image $\boldsymbol{T}$ within a search image $\boldsymbol{I}$ results in a cross correlation matrix $\boldsymbol{C}$, where locations with high correlation values indicate possible detections of the template. This can be computationally expensive, as the NCC has to be computed at all possible positions of the template with respect to the search image.

$$\boldsymbol{C}(i,j) = \frac{\sum_{x,y}\left[\boldsymbol{I}(x,y)-\overline{I}_{i,j}\right]\left[\boldsymbol{T}(x-i,y-j)-\overline{T}\right]}{\sqrt{\sum_{x,y}\left[\boldsymbol{I}(x,y)-\overline{I}_{i,j}\right]^2\sum_{x,y}\left[\boldsymbol{T}(x-i,y-j)-\overline{T}\right]^2}} \tag{13}$$

Equation 13 shows the general form of the NCC. The sums run over x, y in the region under the template positioned at i, j, $\overline{I}$ is the mean of the search image in the same region and $\overline{T}$ is the mean of the template. A template matching example is given in Figure 12.

Template Image and Search Image

As templates in the presented approach are always simply vertical or horizontal slices of the search image (due to the inherent symmetry of the SDM, vertical and horizontal slices represent the same 90° shifted pattern), they only need to be shifted into one direction across the search image. Therefore, the number of necessary computations is drastically reduced compared to a full two-dimensional evaluation. To find recurring structures inside the SDM, vertical slices are selected as template images and all horizontal shifts of these templates across the search image are evaluated. This results in one cross correlation vector $\boldsymbol{c}_k$ per template.

The width of the templates is set to the number of beats w representing two bars. Although successive beats within individual bars already represent short musical structures, the

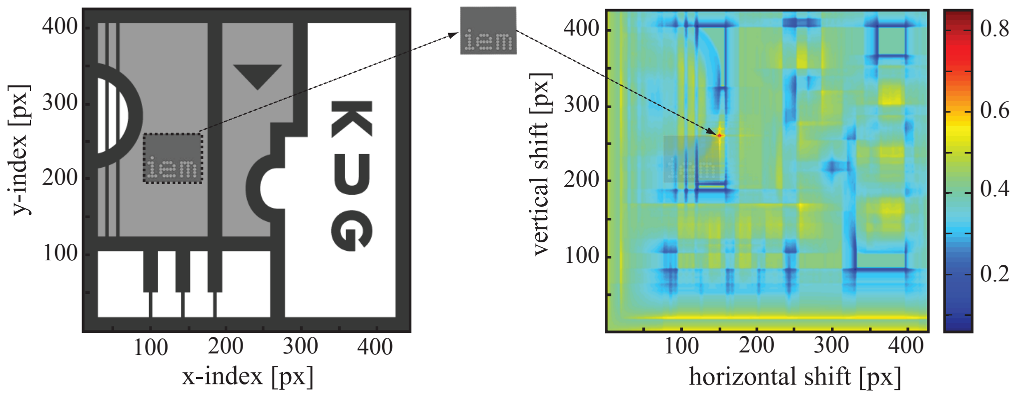


Figure 12: The template image (middle, *IEM* logo) is searched inside the combined search image (left). The maximum correlation value inside $\boldsymbol{C}$ (right) indicates the detected position of the template.

combination of two bars can basically be seen as the smallest sequential building block for song segments. To preserve a temporal resolution on the level of beats, the hop size between adjacent templates (and search images) is set to 1 (beat).

By defining k as the horizontal starting index of a template slice $\boldsymbol{T}_k$, one can compute the individual values of the related correlation vector $\boldsymbol{c}_k$, by iteratively correlating $\boldsymbol{T}_k$ with all slices of the related search-image. The respective slices representing the template image $\boldsymbol{T}_k$ and all related search image slices $\boldsymbol{I}_{k,n}$ can formally be expressed as follows:

$$\boldsymbol{T}_k = \begin{pmatrix} s_{k,1} & s_{k+1,1} & \dots & s_{k+w-1,1} \\ s_{k,2} & s_{k+1,2} & \dots & s_{k+w-1,2} \\ \vdots & \vdots & \ddots & \vdots \\ s_{k,N} & s_{k+1,N} & \dots & s_{k+w-1,N} \end{pmatrix}, \quad \boldsymbol{I}_{k,n} = \begin{pmatrix} s_{l,1} & s_{l+1,1} & \dots & s_{l+w-1,1} \\ s_{l,2} & s_{l+1,2} & \dots & s_{l+w-1,2} \\ \vdots & \vdots & \ddots & \vdots \\ s_{l,N} & s_{l+1,N} & \dots & s_{l+w-1,N} \end{pmatrix}$$

where $l = k + w + n$, as the inner correlation of a template to itself is not evaluated, and $n = 0, 1, \dots, N - k - w$ is the index of all individual search images related to the template.

As the detection of structures using NCC should be independent of offsets inside the search or template image, $\boldsymbol{T}_k$ and $\boldsymbol{I}_{k,n}$ are converted to zero mean matrices.

$$\tilde{\boldsymbol{T}}_k = \boldsymbol{T}_k - \overline{T}_k \quad and \quad \tilde{\boldsymbol{I}}_{k,n} = \boldsymbol{I}_{k,n} - \overline{I}_{k,n} \tag{14}$$

The final NCC vector is computed by calculating the inner product of the zero mean template image with all related zero mean search image slices, normalized by their norms.

$$\boldsymbol{c}_k(n) = \left\langle \frac{\tilde{\boldsymbol{T}}_k}{\|\tilde{\boldsymbol{T}}_k\|}, \frac{\tilde{\boldsymbol{I}}_{k,n}}{\|\tilde{\boldsymbol{I}}_{k,n}\|} \right\rangle \tag{15}$$

3.2.2 Recurrence Matrix

Performing these operations for all templates, results in a set set of correlation vectors. These vectors have different lengths, as the horizontal dimension of of the search image becomes smaller for each new template. Therefore, the length N_c related the longest correlation vector $\boldsymbol{c}_1$ corresponds to the dimension of the SDM minus the size of one template (as the first template was not compared to itself). All vectors are zero padded to length N_c and stored into a lower triangular matrix, referred to as the recurrence matrix $\boldsymbol{R}$.

$$\boldsymbol{R} = \begin{pmatrix} c_{1,1} & 0 & \dots & 0 \\ c_{1,2} & c_{2,1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ c_{1,N_c} & c_{2,N_c-1} & \dots & c_{N_c,1} \end{pmatrix}$$

As depicted in Figure 13, sequences of high correlation values form horizontal lines in the recurrence matrix $\boldsymbol{R}$. Such sequences occur, if multiple successive templates correlate with successive slices of the search image. Therefore, they represent potential repetitions of segments after a certain time lag, similar to the off-diagonal lines inside the original SDM.

In most cases $\boldsymbol{R}$ will expose a mixture of solid horizontal lines, partially broken lines and short non-relevant patterns. To remove most of the non relevant structures inside $\boldsymbol{R}$, a simple post-processing operation is performed by applying a sliding median filter along the horizontal axis of $\boldsymbol{R}$. As only horizontal lines longer than ≈ 2 bars are considered to represent relevant structures, the length of the median filter is set to $n_{med} = t_{<4}$ bars. The resulting filter removes non-representative structures (shorter than $^1/_2\, n_{med}$) and enhances the homogeneity of relevant lines.

$$\boldsymbol{R}_f(i,j) = median\{[\boldsymbol{R}\left(i, j - \frac{n_{med}}{2}\right), \dots, \boldsymbol{R}\left(i, j + \frac{n_{med}}{2}\right)]\} \tag{16}$$

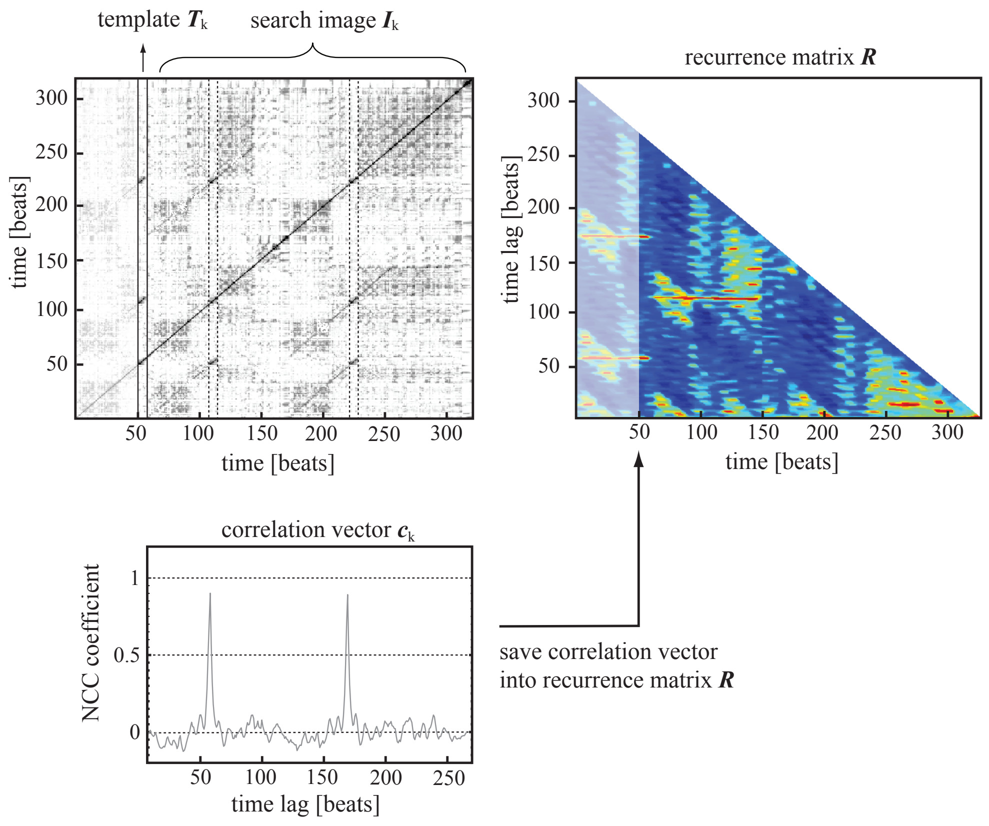


Figure 13: As indicated with dashed boxes, maxima in the cross correlation vector (left, bottom) mark possible repetitions of the template $\boldsymbol{T}_k$ in the related search image $\boldsymbol{I}_k$. The recurrence matrix (right) is created by storing consecutive correlation vectors side by side into a matrix. In this example, the weak off diagonal lines, indicating repeated sequences inside the original SDM, are nicely represented as horizontal lines in the recurrence matrix.

3.2.3 Repetition Surface

To find all valid repetitions, the correlation maxima inside the columns $\boldsymbol{r}_{1,2,\ldots,N_c}$ of $\boldsymbol{R}_f$ are detected by a peak detection algorithm, using a minimum distance of ≈ 2 bars between peaks. To decide whether a peak in $\boldsymbol{r}_j$ represents a valid detection or not, an individual threshold for each bar of the song is computed. Although detection thresholds for template matching methods based on NCC are often set to very high values (e.g. 0.95 to 0.99), it has to be considered that repetitions within songs are never perfect and the median filtering will remove peaks of very high correlation values. Therefore, similar as for the mapping threshold γ_k, Otsu's method is applied to each row of $\boldsymbol{R}_f$, neglecting all values smaller than 0.5, to find a good threshold for valid peaks.

The detected peaks inside each column are marked as valid detections in binary vectors. These vectors are stored column-wise into a matrix $\boldsymbol{R}_b$ (see Figure 14), again showing repetitive sequences as horizontal lines. Short discontinuities within the lines of $\boldsymbol{R}_b$ as well as line deviations of ± 2 $beats$ are adjusted in a last post processing step, resulting in a clean, well-defined representation of all found repetitions.

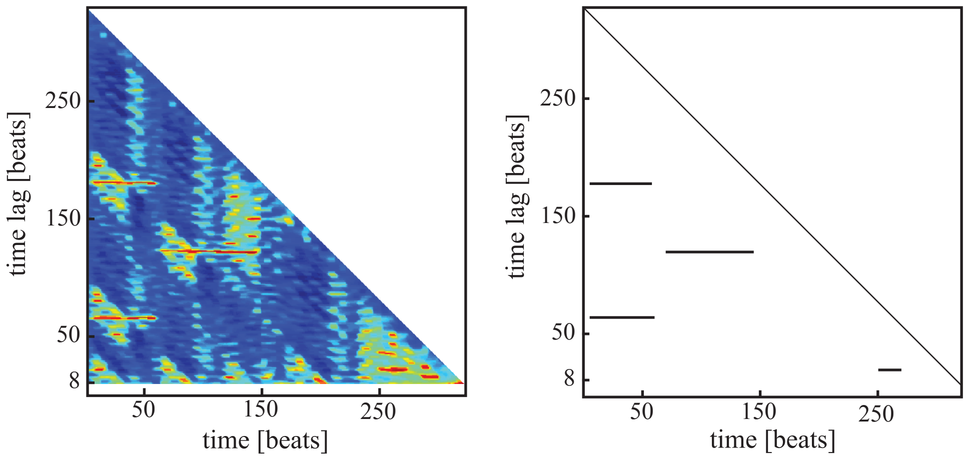


Figure 14: The median-filtered recurrence matrix $\boldsymbol{R}_f$ (left) is binarized based on a peak detection algorithm, which is column-wise applied to the matrix. As indicated, the used template width (in this example 8 beats) has to be incorporated when defining time lags between detected segments.

This representation is called *repetition surface*, as each horizontal line describes two time intervals: the song section which is repeated (horizontal location and length of the line) and the related repetition after a certain time lag or offset (vertical offset of the line). Therefore, each line represents a *segment pair* $\boldsymbol{\varphi}$ and can be characterized as follows:

$$\boldsymbol{\varphi} = [s, e, o] \tag{17}$$

where s and e are the horizontal-indices of the start and end of the repeated song section and o is the time lag between between this section and its repetition. More specifically, this means that the part played during the time interval $\boldsymbol{t}_s = [s, e]$ is repeated after o beats during another time-interval $\boldsymbol{t}_r = [s+o, e+o]$. Therefore, the interval $\boldsymbol{t}_s$ is defined as the *source segment* related to interval $\boldsymbol{t}_r$, the *target segment* (or repetition of $\boldsymbol{t}_s$).

As these repetitions will often not readily represent parts of the musical form, but only repetitions of potentially overlapping, similar musical sequences on a lower semantic scale, these repetitions are called *musical patterns*. It is important to note, that any segment pair with an identical source *or* target segment represents another occurrence of the same part.

In Section 4, all these relations among segment pairs will be investigated and an algorithm to resolve all conflicts of overlapping intervals will be developed (see Figure 15).

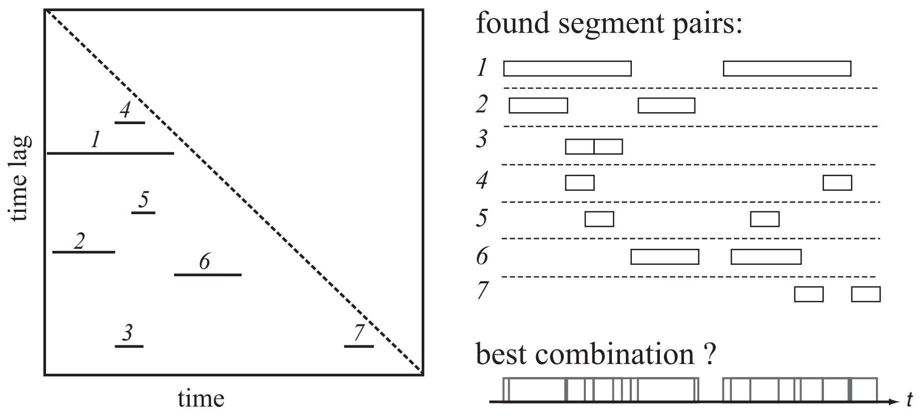


Figure 15: Examples for overlapping conflicts of related segment pairs found inside the repetition surface.

3.3 Novelty Function

One downside when using sequential search algorithms like iterative template matching to find repeated patterns, is their inability to separate two or more different patterns that always occur in the exact same order. For example, when trying to detect the pattern sequence $\{A \rightarrow A \rightarrow B \rightarrow C \rightarrow A \rightarrow B \rightarrow C\}$, the split between segment B and C will not be detected.

One simple but well known method developed to find transitions of patterns in a SDM is the audio novelty function $\boldsymbol{\eta}$, which was initially proposed in [Foo00]. In Section 4.4.1, a modified version of this function will be used to post-process unusually long segments.[3]

Basically, $\boldsymbol{\eta}$ results from correlating a checkerboard-like kernel matrix $\boldsymbol{K}$ along the main diagonal of a full SDM. Segment borders are then found by detecting peaks inside $\boldsymbol{\eta}$. The presented approach basically follows this main idea, although a few modifications to enhance the quality of the detected borders will be introduced.

As musical sections of high inner similarity typically form block-like structures along the main diagonal of the SDM, borders between these sections can be detected by finding the position where two such blocks diagonally adjoin each other. Considering the previously presented NCC algorithm, correlating the checkerboard along the main diagonal is therefore similar to finding good matches of a template image (kernel) and inside a search image.

The basic, binary checkerboard kernel is constructed by merging 4 sub matrices into one quadratic matrix. Assuming that $\boldsymbol{C}$ is a binary nxn matrix where all values are set to 1, the checkerboard can easily be created as

$$\boldsymbol{K}_b = \begin{pmatrix} -\boldsymbol{C} & +\boldsymbol{C} \\ +\boldsymbol{C} & -\boldsymbol{C}. \end{pmatrix} \tag{18}$$

In [Foo00], it has been suggested to apply a radial Gaussian weighting function to the kernel, to gradually reduce the influence of values lying far away from the center of the kernel, to focus the correlation result to the currently evaluated position.

$$\boldsymbol{G}(x,y) = e^{-\left(\frac{(x-N-1/N)^2}{2\sigma^2}+\frac{(y-N-1/N)^2}{2\sigma^2}\right)} \tag{19}$$

[3]Although the novelty function is used much later in the processing chain (after finding a preliminary segmentation), it is introduced it in the current Section, as it is directly related to the processing of the SDM and represents an interesting supplement to the proposed method.

where $2*N+1$ is the dimension of the square kernel. The variance σ is set to 0.5, offering a good attenuation but not a total elimination of values lying near the margin of the kernel.

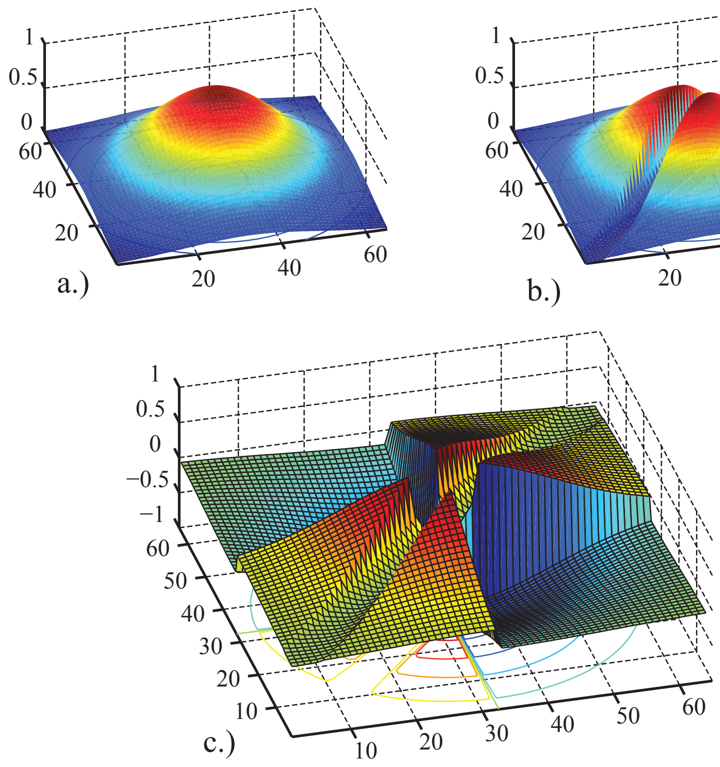


Figure 16: The final correlation kernel $\boldsymbol{K}$ is calculated starting with a radial Gaussian function (a.) combined with a diagonal weighting matrix $\boldsymbol{D})(b.)$, multiplied by a checker-board kernel $\boldsymbol{K}_b$ (c.).

In addition to this proposed weighting, high influence of areas representing strong bar level dependencies, but no real long-term similarities, should be avoided. Therefore, a negative Gaussian weighting window $\boldsymbol{g}$ of width $2 * B + 1$, where B is the number of beats representing one bar, is applied to the main diagonal of the kernel and the first B side diagonals.

$$\boldsymbol{g}(x) = -e^{-\left(\frac{(x-B-1/B)}{2\sigma^2}\right)} \tag{20}$$

For this window, σ is set to $1/e$.

If $\boldsymbol{D}_b$ is a binary matrix, where all values are set to one, values of the main diagonal and the first B side diagonals can now be multiplied with the respective weighting values from $\boldsymbol{g}$. The main diagonal is denoted as $diag(\boldsymbol{D}_0)$ and side diagonals as $diag(\boldsymbol{D}_{-1})$ $diag(\boldsymbol{D}_{+1})$ (first left and right side diagonal).

$$diag(\boldsymbol{D})_d = diag(\boldsymbol{D})_d \cdot \boldsymbol{g}(d + B + 1) \tag{21}$$

where $d = [-B, \dots, B]$.

The full kernel (see Figure 16) can therefore be constructed as follows

$$\boldsymbol{K} = \boldsymbol{K}_b \circ \boldsymbol{G} \circ \boldsymbol{D} \tag{22}$$

Additionally, the kernel $\boldsymbol{K}$ is correlated with the mapped matrix $\boldsymbol{S}_{cmb}$ instead of using a full SDM, to obtain clearer peaks in $\boldsymbol{\eta}$ (see Figure 17). As depicted in Figure 17, the final border candidates are represented by peaks in the novelty function.

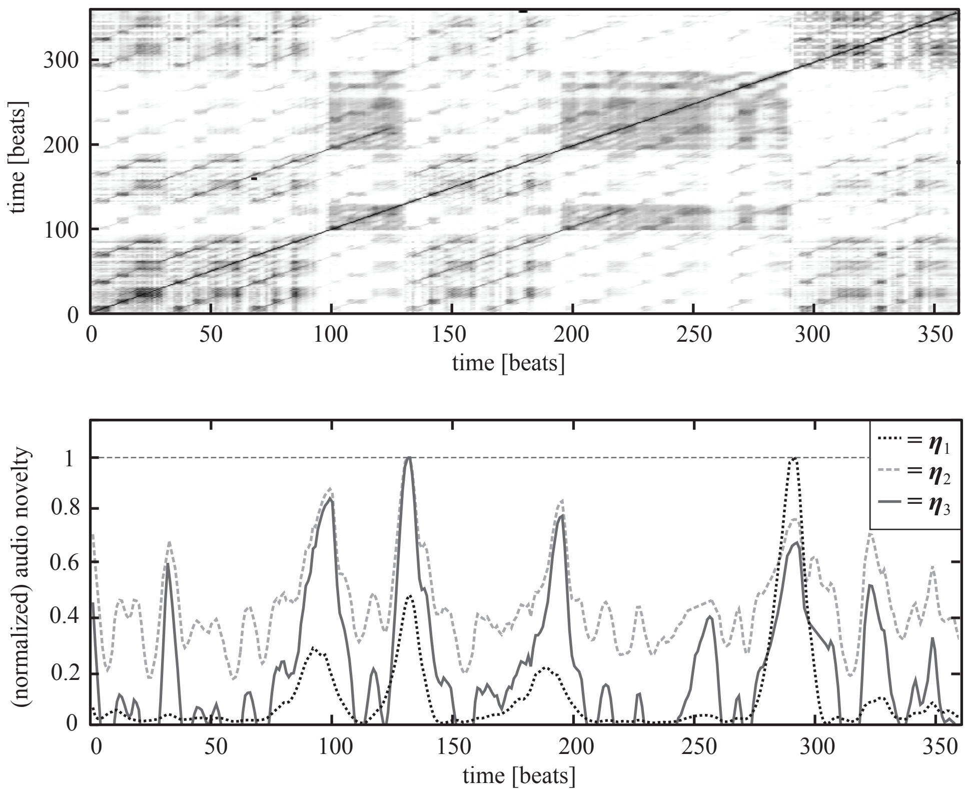


Figure 17: This Figure shows three different (normalized) versions of the novelty function: $\boldsymbol{\eta}_1$ (full SDM + kernel without diagonal weighting); $\boldsymbol{\eta_2}$ (mapped SDM + kernel without diagonal weighting); $\boldsymbol{\eta_3}$ (mapped SDM + kernel with diagonal weighting). It can clearly be noted that the correlation with the full SDM only exhibits very strong block structures ($\boldsymbol{eta}_1$), while the correlation with the mapped SDM reveals much more details ($\boldsymbol{eta}_2$). Additionally, the proposed diagonal mapping significantly reduces the overall offset in ($\boldsymbol{eta}_3$) and shows very clear peaks at potential segment borders.

4 Time Interval Data

Considering the relations among all segment pairs represented by the repetition surface, the problem of describing and resolving conflicts of overlapping time-interval data has to be examined. As each segment pair describes a certain musical pattern, the problem of overlapping patterns as well as the relations among different patterns have to be investigated.

The analysis and interpretation of time-interval data has intensively been discussed in the field of temporal pattern mining [MF10]. Although the final segmentation system will not be directly based on a known time-interval mining method, a short overview on the basic concepts in the field will be given, to define a common understanding of the used taxonomies and techniques.

The relations of time-interval data can be quite complex. For example, if one wants to uniquely describe all potential temporal arrangements of two time intervals, 13 relations have to be defined. They are known as *Allen's interval relations* [All83] (see Figure 18) and are the basis for many temporal pattern mining systems.

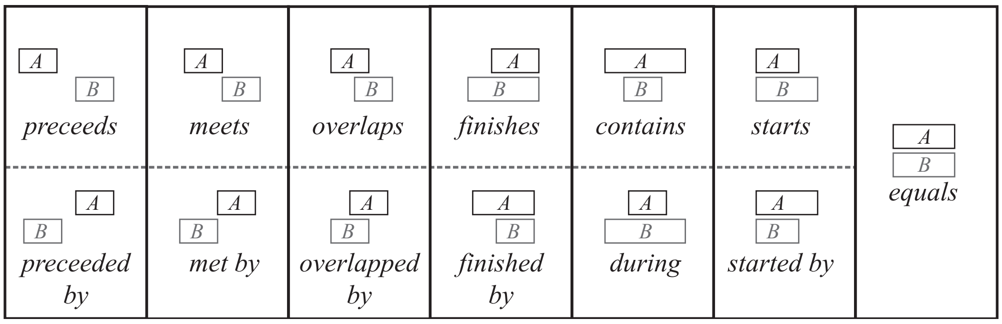


Figure 18: Allen's interval relations. Each relation can be read as: "A (*relation*) B".

Before focusing on the implemented segment detection system, a couple of basic formal descriptions related to time-interval data have to be defined. Following a detailed list of definitions given in [MF10], the formal definitions relevant for the succeeding segment detection stage are introduced.

As already denoted in Section 3.2, a simple *time interval* can be defined by the tuple

$$\boldsymbol{t} = [s, e], \quad s \leq e \tag{23}$$

where s describes the *starting* position of the interval and e the corresponding *ending*.

Therefore, each interval represents a set of time instances

$$\boldsymbol{t} = [s, e] \leftarrow \{s, \ldots, e\}. \tag{24}$$

Accordingly, the *duration* of an interval can be computed as

$$d([s, e]) = e - s + 1. \tag{25}$$

An interval related to a label σ selected from a set of unique symbols S (e.g. $\sigma \in S = \{A, B\}$) is called *symbolic interval* and can be defined by the triple

$$\boldsymbol{t} = [\sigma, s, e] \tag{26}$$

Additionally, when analyzing multiple intervals, a *sequence* of intervals can be defined as

$$\mathbb{T} = \{[\sigma_i, s_i, e_i] \mid i = 1, \ldots, N\} \tag{27}$$

and the intervals of the sequence are are non-overlapping if $(s_i < e_i < s_j < e_j) \forall\, i < j$. Next, the order of two intervals is defined as

$$\boldsymbol{t}_i < \boldsymbol{t}_j \Leftrightarrow (s_i < s_j) \vee (s_i = s_j \wedge e_i < e_j) \tag{28}$$

and two intervals t_a, t_b *overlap* within the intersection interval t_{ab}, if

$$\boldsymbol{t}_{ab} = \boldsymbol{t}_a \cap \boldsymbol{t}_b = \{s_m, \ldots, e_m\} \cap \{s_s, \ldots, e_s\} \neq \emptyset. \tag{29}$$

Based on this interval, the intervals t_{a-b} and t_{b-a}, representing the parts of t_a and t_b which do not overlap, can be defined as follows:

$$\boldsymbol{t}_{a-b} = \boldsymbol{t}_a \setminus \boldsymbol{t}_{ab}, \quad \boldsymbol{t}_{b-a} = \boldsymbol{t}_b \setminus \boldsymbol{t}_{ab} \tag{30}$$

Finally, interval $\boldsymbol{t}_a$ partially overlaps interval $\boldsymbol{t}_b$ from left ($\boldsymbol{t}_a < \boldsymbol{t}_b$) or right ($\boldsymbol{t}_a < \boldsymbol{t}_b$), if

$$\boldsymbol{t}_a \cap \boldsymbol{t}_b \neq \emptyset \ \wedge \ \boldsymbol{t}_a \setminus \boldsymbol{t}_b \neq \emptyset \tag{31}$$

and fully overlaps $\boldsymbol{t}_b$, if

$$\boldsymbol{t}_b \subset \boldsymbol{t}_a \tag{32}$$

As already mentioned, these definitions will be used in the following Sections to describe relations and conflicts of overlapping time interval data.

4.1 Segment Grouping

Typically, many musical patterns detected inside the repetition surface will be repeated more than once and will therefore be related to multiple segment pairs inside the *repetition surface*. Each group of such pairs forms an individual subset representing all occurrences of the corresponding pattern. To find all these subsets, the relationships between lines in the repetition surface have to analyzed and interpreted.

As depicted in Figure 19, the most intuitive way of finding all such related pairs, lies in the analysis of the geometric relations inside the repetition surface. If one defines two imaginary lines through any point in the repetition surface, a vertical line and a diagonal line, the intersections of these imaginary lines with the horizontal-axis form a right-angled triangle together with the chosen point. Based on this triangle, two important observations can be made:

- All detections which are located inside the adjacent side of this triangle (vertical line) represent different repetitions of the same time-instance. Therefore, they are all related to the *same source segment* and represent *different target segments*.
- All detections which are located inside the hypotenuse of this triangle (diagonal line) represent the same time instance hit by different repetitions. Therefore, they are all related to *different source segments* and represent the *same target segment*.

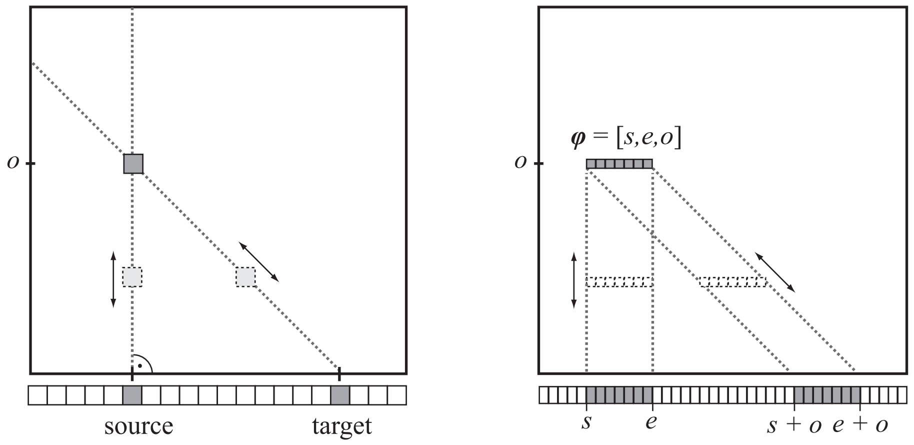


Figure 19: Relations inside the repetitions surface.

An individual segment pair $\varphi_k = [s_k, e_k, o_k]$ is represented by a line in the repetition surface running from index $[o_k, s_k]$ to index $[o_k, e_k]$. Based on the proposed definitions this pair can be represented by two symbolic time intervals with identical labels.

$$\boldsymbol{t}_{k,s} = [\sigma_k, s_k, e_k], \quad \boldsymbol{t}_{k,r} = [\sigma_k, s_k + o_k, e_k + o_k] \tag{33}$$

where the label $\sigma_k = k$ indicates the index of the pair, $\boldsymbol{t}_{k,s}$ is the time interval representing the source segment and $\boldsymbol{t}_{k,r}$ is the time interval related to the repetition of this segment.[4]

When these observations are expanded to every point of a line in the repetitions surface (see Figure 19), one can easily define all segment pairs related to a certain repetition. Considering the previously defined segment pair $\boldsymbol{\varphi_k}$, it can formally be checked if another pair is directly related to this pair as follows:

$$\boldsymbol{\varphi}_k \leftarrow \boldsymbol{\varphi}_l \;\Leftrightarrow\; (\boldsymbol{t}_{l,s} \approx \boldsymbol{t}_{k,s} \;\vee\; \boldsymbol{t}_{l,s} \approx \boldsymbol{t}_{k,r} \;\vee\; \boldsymbol{t}_{l,r} \approx \boldsymbol{t}_{k,s} \;\vee\; \boldsymbol{t}_{l,r} \approx \boldsymbol{t}_{k,r}) \tag{34}$$

Direct interval relations To evaluate if two time intervals are approximately equal $(\boldsymbol{t}_k \approx \boldsymbol{t}_l)$, their starts and ends are compared. To account for potential detection errors, two time-intervals are defined as *approximately equal*, if the difference between their starts and stops is smaller than a given tolerance δ.

$$\boldsymbol{t}_k \approx \boldsymbol{t}_l \Leftrightarrow (|s_k - s_l| < \delta \;\wedge\; |e_k - e_l| < \delta) \tag{35}$$

Following the concept of musically motivated thresholds, δ is set to $t_{<2}$ bars. Multiple experiments proofed that the threshold is typically large enough to compensate deviations introduced by detection errors, while it prevents the merging of musically relevant sections. The search for related pairs of repetitions can now be performed based on this definition of identity.

Higher order interval relations As the repetition surface shows all repetitions of a musical pattern at each occurrence of this pattern, it will reveal a high amount of redundancy. For example, given a pattern which is repeated four times, four repetitions will be detected for the first occurrence, three for the second occurrence (which is also the first repetition) and so on. If a pattern is repeated m times and all of its repetitions are correctly detected, theoretically $n = \frac{m(m-1)}{2}$ segment pairs will be related to the pattern. Therefore, without detection errors, the repetitions found for the first occurrence already accurately define all occurrences of the corresponding song segment.

In many cases, however, not all repetitions of each occurrence are detected and some relations between patterns may not be directly detectable using the conditions given in

[4]Considering the described computations, the temporal units are always beats.

Equation 27. In this case, the mentioned redundancy of detections can be exploited by finding relations of segments via other segments. Figure 20 illustrates this idea. Let $\boldsymbol{t}_{a,s}$, $\boldsymbol{t}_{b,s}$ and $\boldsymbol{t}_{c,s}$ be time intervals representing the source segments of three segment pairs and let $t_{a,r}$, $t_{b,r}$ and $t_{c,r}$ be their corresponding repetitions. If $t_{a,r} \equiv \boldsymbol{t}_{b,s}$ and $\boldsymbol{t}_{b,r} \equiv \boldsymbol{t}_{c,s}$ all three pairs are implicitly related to the first source interval ($\boldsymbol{t}_{a,s} \rightarrow \boldsymbol{t}_{b,s} \rightarrow \boldsymbol{t}_{c,s} \rightarrow \boldsymbol{t}_{c,r}$) and $\boldsymbol{\varphi}_a$ will correctly be related to $\boldsymbol{\varphi}_b$ and φ_c.

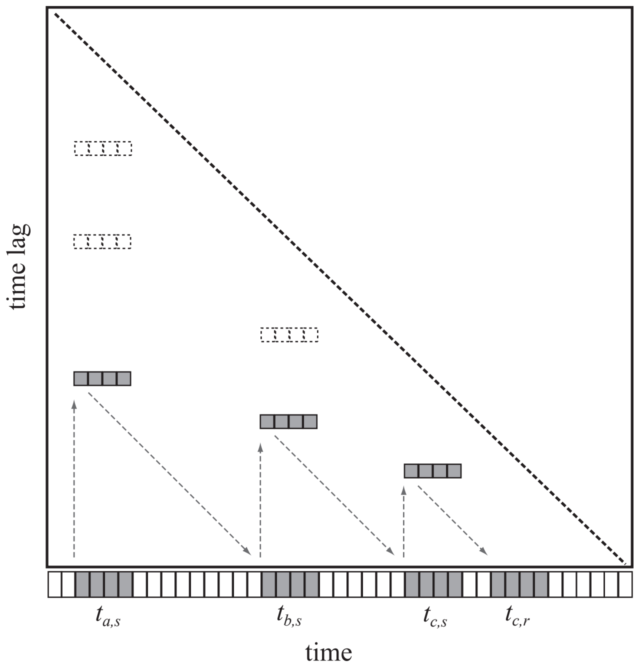


Figure 20: Redundancies and higher order relations inside the repetitions surface.

After finding all related pairs, G subsets $\boldsymbol{\Phi}_{1,2,\ldots,G}$ can be defined. Each subset $\boldsymbol{\Phi}_i = \{\boldsymbol{\varphi}_n | n = 1, \ldots, N_i\}$, where N_i is the number of segment pairs related to the subset, contains all time intervals (repetitions) representing a certain musical pattern.

4.2 Subset Sequences

In the previous Sections a relatively long pre-processing chain has been passed to find a stable representation of all repeated sections of a song. These sections are now represented as subsets, containing groups of related segment pairs. If a unique arbitrary label is defined for each subset, the time intervals related to a subset will form a sequence of identically labeled, non-overlapping time intervals, a *subset sequence*.

As the direct combination of all subset sequences will reveal many overlapping parts, the

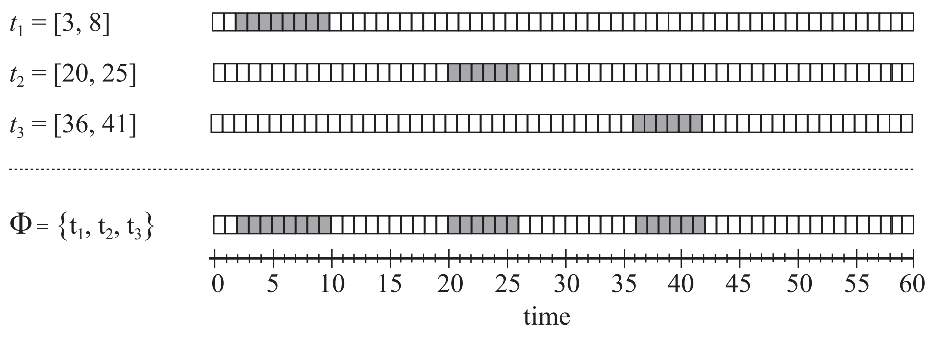


Figure 21: The time intervals combined into one subset and the resulting sequence of related time intervals. (For simplicity, only the unique intervals related to the subset $\mathbf{\Phi}$ are depicted in this Figure. Identical source or target segments are ignored.)

initial detection process has to be followed by the search for an optimal combination of these sequences. Before an algorithm to find this optimal combination can be developed, each sequence has to be related to a quality attribute, allowing to compare the "relevance" of overlapping sequences.

4.2.1 Repetition Quality

In this Section, the *repetition strength* κ will be introduced. It will later be used to define the order in which the found subsets are iteratively compared in the final segment combination algorithm (see Section 4.3). The higher the *repetition strength* of a subset, the earlier it is processed and the greater is its influence on the final segmentation.

The first attribute that can directly be extracted from each subset, is the average length of all related segment pairs. As longer repetitions tend to represent more stable detections, the *mean segment length* λ can be defined based on the length of all N_i segment pairs of a subset $\mathbf{\Phi}_i = \{\boldsymbol{\varphi}_n | n = 1, \ldots, N_i\}$:

$$\lambda_i = \frac{1}{N_i} \sum_{n=1}^{N_i} \frac{(d(t_{n,s}) + d(t_{n,r}))}{2} \tag{36}$$

Additionally, the *recurrence matrix* $\boldsymbol{R}$ (see Section 3.2.2) can be used to define another quality measure for all individual segment pairs. Based on the overall mean μ_{all} of all correlation vectors forming $\boldsymbol{R}$

$$\mu_{all} = mean\{[\boldsymbol{c}_1^T, \ldots, \boldsymbol{c}_{N_c}^T]\} \tag{37}$$

and the mean μ_n of the correlation values related to segment pair $\boldsymbol{\varphi}_n$

$$\mu_n = mean\{[\boldsymbol{R}(o_n, s_n), \dots, \boldsymbol{R}(o_n, e_n)]\}, \tag{38}$$

the following *quality measure* for each pair inside a subset can be computed.

$$q_n = \frac{\mu_n - \mu_{all}}{1 - \mu_{all}} \tag{39}$$

This measure can be interpreted as a probability value. Pairs related to high probabilities represent stable recurrences which are likely to be relevant for the final segmentation, while pairs with a very low quality measure (e.g. below zero) are likely to represent mis-detections.

Based on the mean of the *quality measures* of all segment pairs related to a subset, a combined *repetition quality* υ_i for the subset can be defined.

$$\upsilon_i = \frac{1}{N_i} \sum_{n=1}^{N_i} q_n \tag{40}$$

By combining the *mean segment length* λ_i and the *repetition quality* υ_i, the overall *repetition strength* κ_i of subset $\boldsymbol{\Phi}_i$ can finally be calculated as follows:

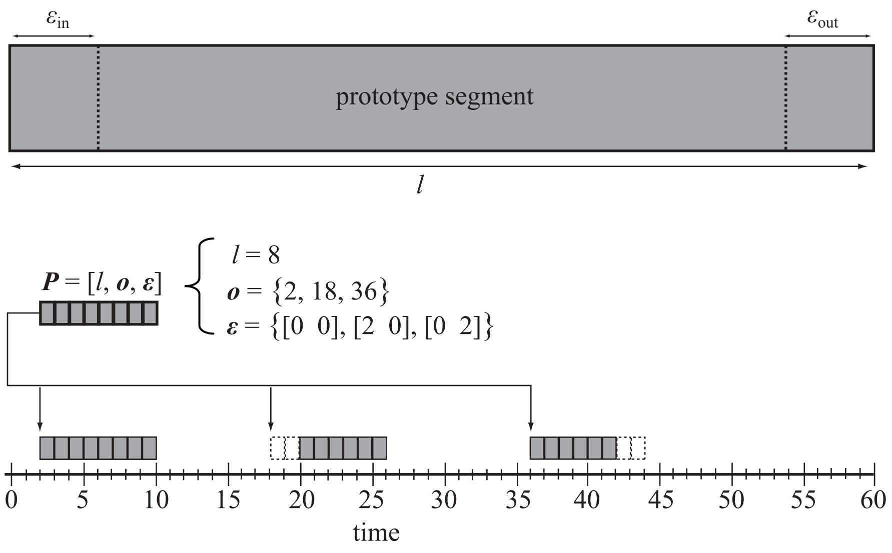
$$\kappa_i = \upsilon_i \cdot \lambda_i \tag{41}$$

4.2.2 Prototype Segment

As the individual intervals of subset sequences all represent the same musical pattern, changes to any interval of the sequence must have a direct impact on every other interval inside the subset. These inner dependencies can easily lead to very complex relations and confusing data structures when comparing sequences of intervals.

Therefore, a new representation for interval sequences with inner dependencies, based on individual *prototype* intervals $\boldsymbol{t}_p$ (see Figure 22) has been developed. This representation is much more intuitive and easier to handle than multiple dependent intervals, as any change to the prototype interval will directly change all its shifted versions.

A nice analogy to such a prototype interval is a loop extracted from a song, when working with a typical audio sequencer. The bare loop (e.g. labeled *"part 1"*) represents a certain time interval of the song, without any temporal information on its relative positioning



ε_{in}
ε_{out}
prototype segment
l
$\boldsymbol{P} = [l, \boldsymbol{o}, \boldsymbol{\varepsilon}]$
$l = 8$
$\boldsymbol{o} = \{2, 18, 36\}$
$\boldsymbol{\varepsilon} = \{[0\ \ 0], [2\ \ 0], [0\ \ 2]\}$
0
5
10
15
20
25
30
35
40
45
50
55
60
time

Figure 22: Basic parameters of a prototype segment (top) and example of interval sequence represented by the prototype for a given set of parameters (bottom).

inside the song. If this loop is copied to all positions where similar parts occur (similar enough to talk about a repetition of the respective segment), the resulting loops inside the sequencer will represent a sequence of non-overlapping time intervals related to the same label: a subset.

As $\boldsymbol{t}_p$ is only defined by its length, its start can be ignored and only the length of the longest related segment pair is needed. Based on a vector $\boldsymbol{l}$ containing all lengths of segments inside a subset

$$\boldsymbol{l}(n) = max\{d(\boldsymbol{t}_{n,s}), d(\boldsymbol{t}_{n,r})\}, \quad n = 1, \ldots, N, \tag{42}$$

where N is the number of segment pairs inside the respective subset $\boldsymbol{\Phi}$, the maximum length inside the set can easily be computed as $l_{max} = max\{\boldsymbol{l}\}$.

Again, considering multiple occurrences of a loop in a sequencer, the lengths of these occurrences may slightly differ. Therefore, one can typically use a cursor to trim the start or stop of each occurrence to correctly fit the underlying segment. Following this analogy, each repetition can be related to an individual *skip in* and *skip out* value to allow slightly later starts or earlier stops of the respective occurrence. By combining each prototype time interval with all its related starting positions and skips, a full *prototype segment* $\boldsymbol{P}$ describing N segment pairs can be defined as follows (see Figure 22):

$$\boldsymbol{P} = (l, \boldsymbol{o}, \boldsymbol{\epsilon}) \tag{43}$$

where $\boldsymbol{o} = \{o_n | n = 1, \ldots, N\}$ represents a set of shifts and $\boldsymbol{\epsilon} = \{[\epsilon_{in,n}, \epsilon_{out,n}] | n = 1, \ldots, N\}$ represents a set of *skip in* $\epsilon_{in,n}$ and *skip out* $\epsilon_{out,n,}$ values.

4.3 Segment Combination

In this Section, initially the simple problem of resolving overlapping conflicts of two intervals will be addressed. The findings from this simple two-interval case will then be used to describe the problem to overlapping interval sequences with inner dependencies. This final problem will represent the current situation of the processing chain - a set of overlapping subset sequences.

4.3.1 Two Overlapping Intervals

For this example, the interval related to the stronger *repetition quality* κ (see Section 4.2.1) is called the *main interval* $\boldsymbol{t}_m$ and its start and endpoints $[s_s, e_s]$ *main borders*. The weaker interval and its borders are called *sub interval* $\boldsymbol{t}_s$ and *sub borders* $[s_m, e_m]$.

Assuming that these intervals overlap, an ordered list $sort[s_m, e_m, s_s, e_s]$ describing the boundaries of three intervals can be created. One of them represents the overlapping area

$$\boldsymbol{t}_{a+b} = \boldsymbol{t}_a \cap \boldsymbol{t}_b$$

and the others the non-overlapping parts:

$$\boldsymbol{t}_{a-b} = \boldsymbol{t}_a \setminus \boldsymbol{t}_{a+b} \ , \quad \boldsymbol{t}_{b-a} = \boldsymbol{t}_b \setminus \boldsymbol{t}_{a+b}$$

Depending on the actual relative positions of $\boldsymbol{t}_s$ and $\boldsymbol{t}_m$, some of these intervals could either be empty (e.g. if $s_m = s_s$) or too short to represent valid intervals. Based on a given minimum length δ for valid intervals, the overlapping conflict can be resolved by applying a couple of rules:[5]

- The positions of the main borders are static and cannot be removed or shifted, nor can the borders change their type (e.g. a start remains a start).

[5]The number behind each rule indicates the line in Algorithm 2, where the rule is applied.

- A sub border can be inserted between the main borders to split the main interval into two intervals, if the distance to the closest main border is larger than δ. ($\rightarrow$7)
- A sub border lying *inside* the main interval, but too close ($< \delta$) to a main border of the *same* type, is considered to represent a shifted version of this border. ($\rightarrow$5)
- A sub borders lying *inside* the main interval, but too close ($< \delta$) to a main border of a *different* type is ignored. ($\rightarrow$9)
- A sub border lying *outside* the main interval forms the border of a new interval, if the distance to the closest main border is larger than δ. ($\rightarrow$13, 15)
- A sub border lying *outside* the main interval, but too close ($< \delta$) to a main border is ignored. ($\rightarrow$17)

A schematic depiction of the rules defined above is given in Figure 23.

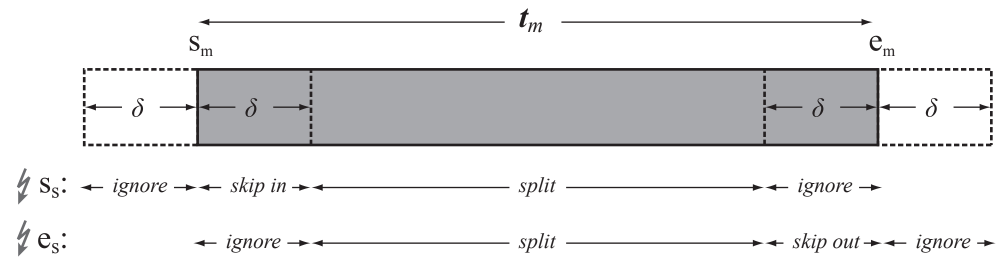


Figure 23: Operations of the algorithm based on the regions defined inside and outside a main interval $\boldsymbol{t}_m$ considering potential overlaps of starts s_s and stops e_s of a sub interval.

Additionally, the individual processing steps are summarized in Algorithm 2.

It is important to note that the area of the sub interval overlapping with the main interval is processed completely independent from its parts lying outside the main interval. For example, if the sub interval partially overlaps the main interval from left, the interval can be interpreted as two intervals: one non-overlapping interval lying in front of the main interval followed by an interval that is totally overlapped by the main interval. This observation will be useful when dealing with multiple overlapping intervals, as it allows to process the individual parts of a sub interval overlapping with different main intervals independently.

Although the areas outside the main interval are fully processed after running through this algorithm, the remaining borders inside the main interval that have been marked as *related* ($\rightarrow$ 5) to a main border or found to indicate a *split* ($\rightarrow$ 7) of the main interval have to be further processed.

Algorithm 2 Processing of Sub Borders

```
1: for $k = 1$ to $2$ do
2:     $b \leftarrow \boldsymbol{t}_s(k)$
3:     if $s_m < b < e_m$ then
4:         if $|d([\boldsymbol{t}_m(i), b])| < \delta$ then
5:             mark border to be related to $\boldsymbol{t}_m(i)$
6:         else if $min\{|d([s_m, b])|, |d([e_m, b])|\} > \delta$ then
7:             mark border to represent a split for $\boldsymbol{t}_m$
8:         else
9:             ignore border
10:        end if
11:    else
12:        if $b < s_m$ and $d([b, s_m]) > \delta$ then
13:            new segment $\leftarrow [b, s_m - 1]$
14:        else if $b > e_m$ and $d([e_m, b]) > \delta$ then
15:            new segment $\leftarrow [e_m + 1, b]$
16:        else
17:            ignore border
18:        end if
19:    end if
20: end for
```

For the simple two interval case, borders marked as *related* can be ignored, as they simply indicate that a repetition of the interval related to the respective border would start or end a little bit earlier than the investigated interval. These borders will later be used to define the *skip in* and *skip out* values, when defining interval sequences based on prototype segments (see Section 4.3.2).

If one or two borders of the sub interval are marked as *splits*, the main interval will be separated into a new set $\boldsymbol{T}_m$ of two or three new segments, created as follows:

$$\boldsymbol{T}_{new} \leftarrow \begin{cases} \{[s_m, s_s - 1], [s_s, e_s], [e_s + 1, e_m]\} & \text{if } splits \leftarrow s_s \wedge e_s, \\ \{[s_m, s_s - 1], [s_s, e_m]\} & \text{if } split \leftarrow s_s, \\ \{[s_m, e_s], [e_s + 1, e_m]\} & \text{if } split \leftarrow e_s, \end{cases}$$

A schematic representation of the described segmentation process is given in Figure 24.

4.3.2 Two Overlapping Interval Sequences

The found solution can now be extended to repeated intervals or interval sequences. For this example, two interval sequences with one or multiple partially overlapping intervals are assumed. As introduced in Section 4.2.2, interval sequences can be described by

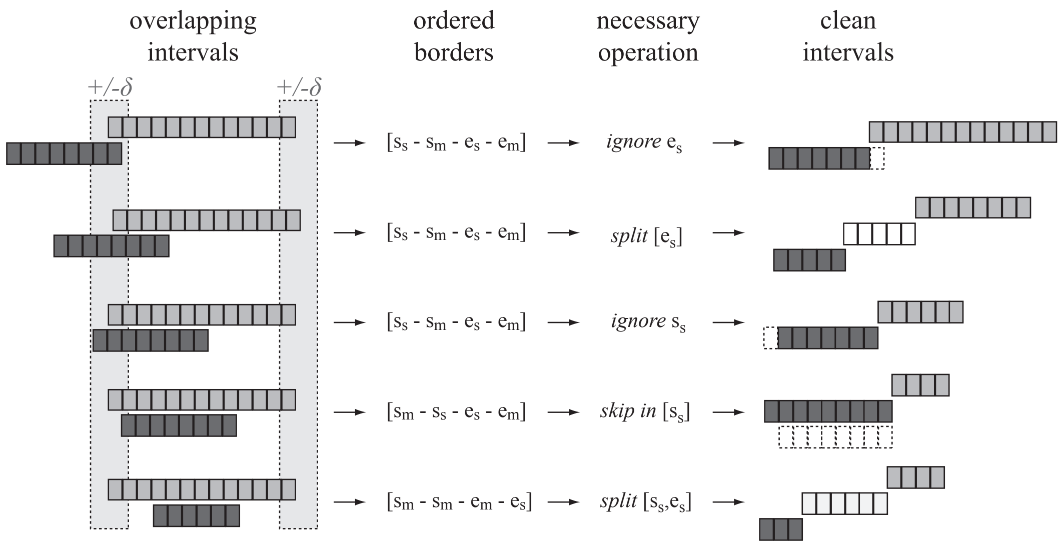


Figure 24: Exemplary processing of two overlapping intervals.

prototype segments. Therefore, following the notation of the previous Section, the two investigated prototypes are called *main* prototype $\boldsymbol{P}_m$ and *sub* prototype $\boldsymbol{P}_s$. It will be shown that not too much has to be changed in the proposed processing when dealing with prototype segments instead of single intervals.

Each overlap of segment pairs is automatically projected onto the two prototype segments. Prototypes can therefore be treated like individual intervals and the found interval borders are iteratively processed with the proposed algorithm. As the relative overlapping positions of all interval pairs of two prototypes have to be identical, overlaps that would lead to ambiguous (different) segmentations of one of the prototypes are ignored.

Similar as for the two interval case, the initial main and sub prototypes can be split into multiple new prototypes, but now not only the borders of the respective intervals have to be changed, but also the shifts or positions of repetitions need to be updated (see Figure 25).

For example, let us assume two interval sequences, where one interval of the sub sequence overlaps one interval of the main sequence from the left. By projecting these overlaps onto the two related prototypes $\boldsymbol{P}_m$ and $\boldsymbol{P}_s$, one can see that the sub prototype is cut at the relative position c_s (indicating the start of overlapping area with the main prototype) and the main prototype is cut at the relative position c_m (the end of the overlapping area).

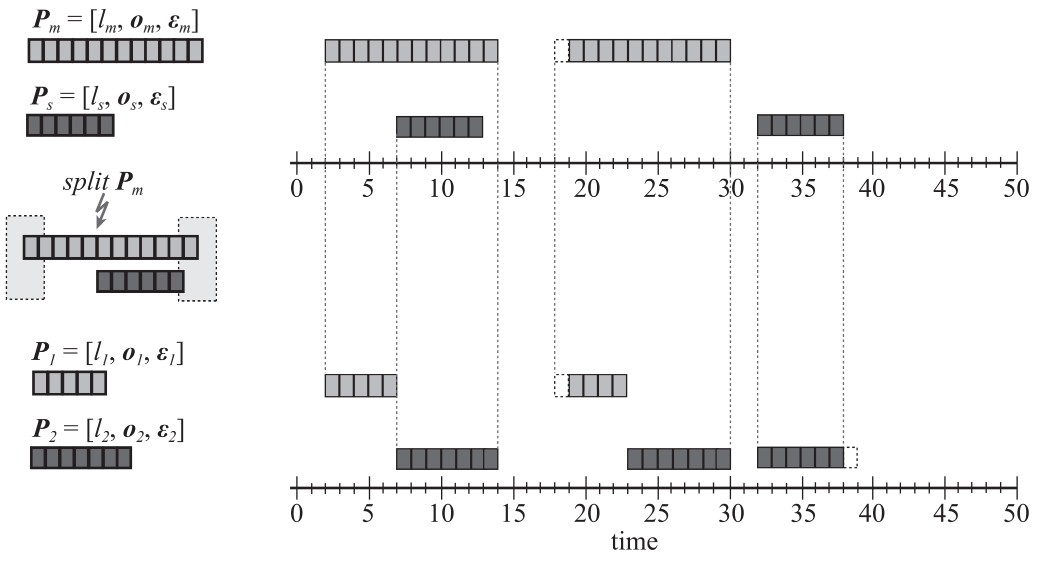
Given the two original prototype segments,

$$\boldsymbol{P}_m = (l_m, \boldsymbol{o}_m, \boldsymbol{\epsilon}_m)$$
$$\boldsymbol{P}_s = (l_s, \boldsymbol{o}_s, \boldsymbol{\epsilon}_s)$$

the three new prototype segments can be defined as follows ($\mathbf{0}$ denotes a zero-vector):

$$\begin{aligned} \boldsymbol{P}_1 &= (l_s - c_s, \boldsymbol{o}_s, \boldsymbol{\epsilon}_s) & \epsilon_{s,out} &\leftarrow \mathbf{0} \\ \boldsymbol{P}_2 &= (c_m, \boldsymbol{o}_m, \boldsymbol{\epsilon}_m), & \epsilon_{m,in} &\leftarrow \mathbf{0} \\ \boldsymbol{P}_3 &= (l_m - c_m, \boldsymbol{o}_m + c_m, \boldsymbol{\epsilon}_m) & \epsilon_{m,in} &\leftarrow \mathbf{0} \end{aligned}$$

Note that the first new prototype is based on the sub prototype, while the others are based on the main prototype. The *skip out* values of the first and second new prototype as well as the *skip in* values of the third prototype are set to a zero vector, as all related intervals are cut at the same position. Additionally the lengths of all prototypes are adapted and the shift of the third prototype has been increased.



$P_m = [l_m, o_m, \varepsilon_m]$
$P_s = [l_s, o_s, \varepsilon_s]$
split P_m
$P_1 = [l_1, o_1, \varepsilon_1]$
$P_2 = [l_2, o_2, \varepsilon_2]$
0 5 10 15 20 25 30 35 40 45 50
0 5 10 15 20 25 30 35 40 45 50
time

Figure 25: Exemplary processing of two overlapping sequences (two prototype segments).

4.3.3 Multiple Overlapping Interval Sequences

In this Section, the example is extended to the evaluation of multiple interval sequences with arbitrary overlaps. Since the relations of more than two sequences have to be observed, each pair of sequences is processed in descending order, following the respective *repetition strengths* κ_i. Besides this extension, no real modifications regarding the processing of two prototypes presented in the previous example have to be introduced.

As depicted in Figure 26, the processing of the first sequence pair is identical to the processing of two sequences and will result in a preliminary segmentation, representing the new *main* sequence. Although the intervals are now based on different prototypes, one can again project all overlaps of the sub sequence with any interval of the main sequence onto the respective prototype segment and iteratively resolve all overlapping conflicts.

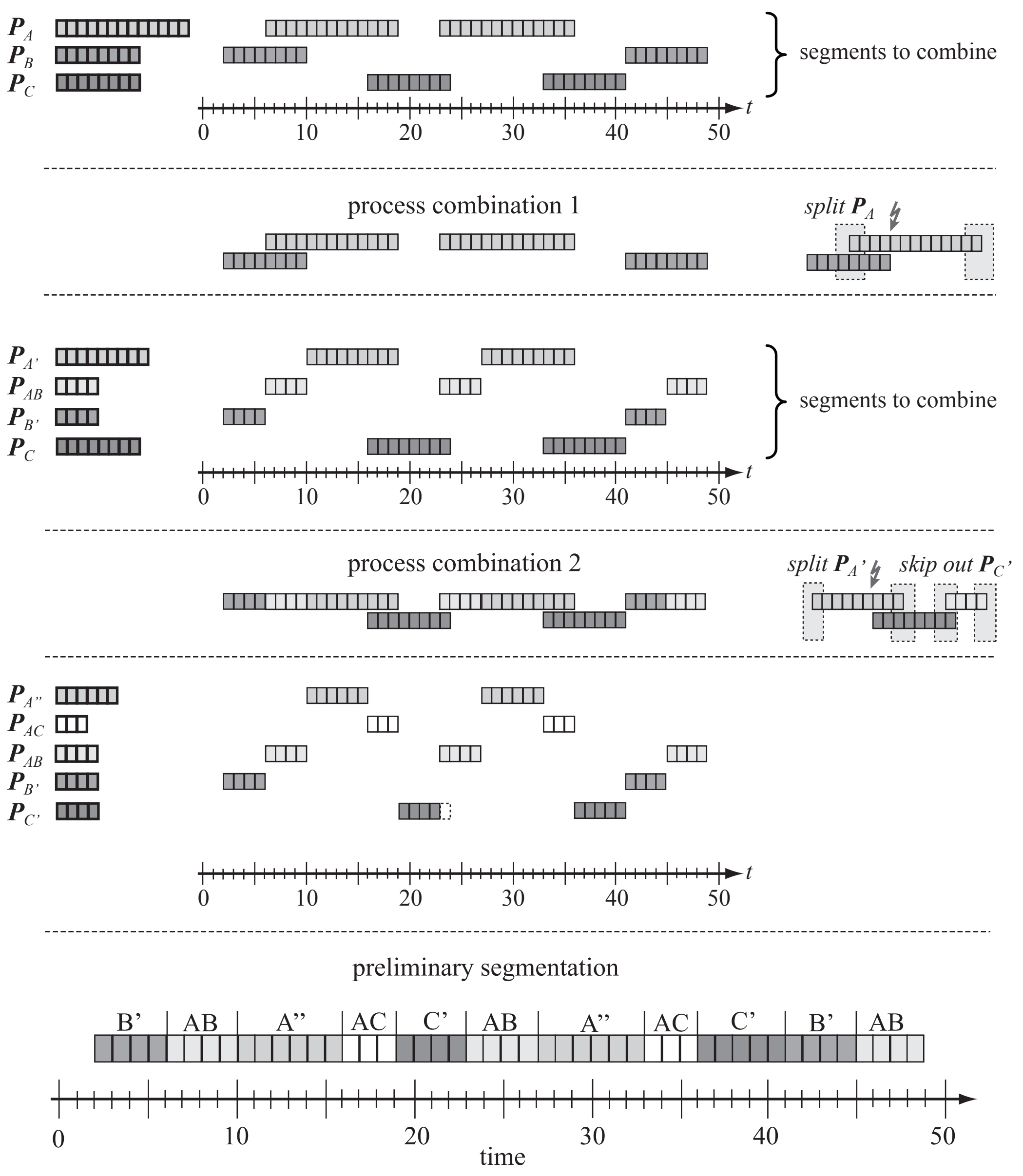


Figure 26: Exemplary processing of multiple overlapping interval sequences.

4.4 Preliminary Segmentation

Based on the subsets found in Section 4.1 and the algorithm introduced in the previous Section, a final set of non overlapping prototypes can be found. These prototypes will form a non-overlapping, labeled interval sequence - the preliminary segmentation $\mathbf{\Gamma}$ of the investigated song.

The threshold δ, defining the minimum length for a valid segment as well as the *skip in* and *skip out* areas of found segments (see Algorithm 2) is set to $t_{<4}$ beats. This threshold guarantees that any segment included in the preliminary segmentation is long enough to represent a musically relevant section of the song.

$$\mathbf{\Gamma} = \{\boldsymbol{P}_i |\, i = 1, \dots, N_P\} \;=\; \{[s_k, e_k, \sigma_k] \mid k = 1, \dots, N_t\} \tag{44}$$

where N_P is the number of found prototype segments and N_t is the total number of intervals represented by all prototype segments.

4.4.1 Detection of Missed Boundaries

As already mentioned in Section 3.3, one downside of the proposed sequential algorithm is its (theoretical) inability to separate two or more different patterns that always occur in the exact same order. Practically, most of these patterns will still be detected, as they are typically not perfectly repeated and "cross-detections" can be found by exploiting all relations of segments inside the *repetition surface*. Still, if a long musical pattern covers two or more always identically repeated sub patterns, these sub patterns will be represented by only one prototype in the preliminary segmentation.

Missed borders between sub patterns will typically lead to unusually long time intervals in the preliminary segmentation. Therefore, a post processing on all prototypes longer than a given threshold δ_s is performed, to reveal potentially missed borders. The threshold δ_s used to identify unusually long segments is set to $t_{>12}$ beats.

The modified novelty function introduced in Section 3.3 is now applied to the first occurrence of each found prototype longer than δ_s. The size of the used kernel is set to 4 bars. As only specific, repeated sections are investigated, the kernel only has to be shifted along the part of the main diagonal in the respective area.

$$\boldsymbol{\eta}(k) = \sum_{m=-N}^{N} \sum_{n=-N}^{N} \boldsymbol{K}(m,n)\boldsymbol{S}_{cmb}(r+k+m, r+k+n), \quad k = [1, \dots, l] \tag{45}$$

where r is the positions of the section within the song, k is the investigated position inside this section, and l is the length of the repetition.

Since no wrong borders should be introduced at this stage, only very high peaks in the novelty function (see Figure 27) are considered to indicate additional borders. Therefore, an empirical threshold $t_\eta = {}^{N}\!/_{2}$ based on the size of the used kernel is introduced. Despite its simplicity, t_η turned out to be a good threshold for relevant detections. If a peak above this threshold is found, the prototype representing the investigated time interval is split at the respective position and the preliminary segmentation $\mathbf{\Gamma}$ is updated.

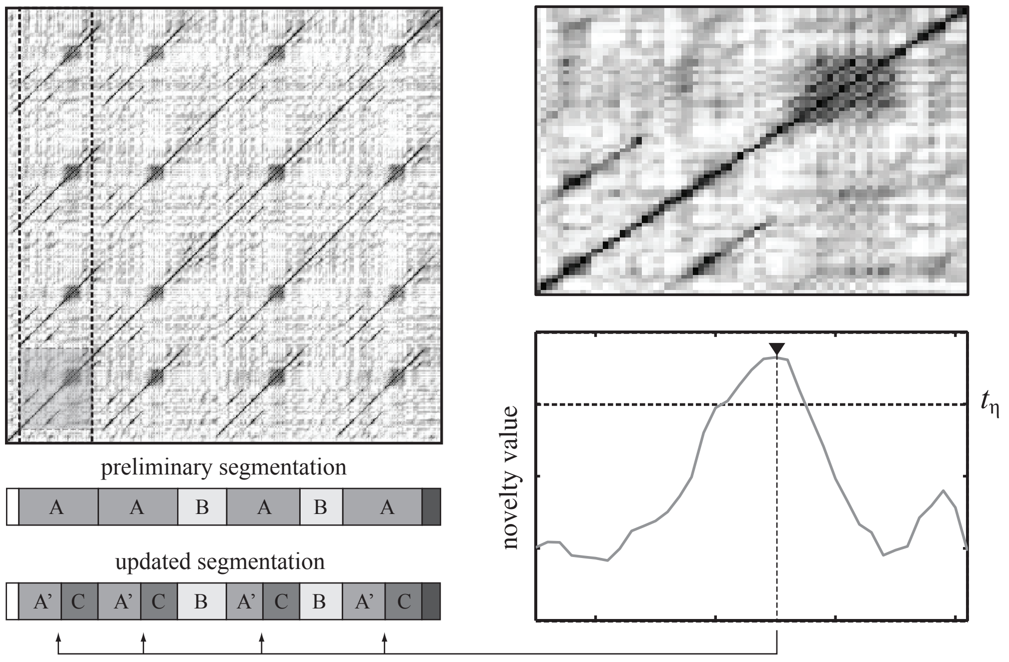


Figure 27: Segment *A* is identified as unusually long segment. The first occurrence of the pattern inside the 4 identically repeated sections is extracted (top, right) and analyzed using the novelty function. As the maximum inside $\boldsymbol{\eta}$ is above t_η, it indicates that the detected segment *A* should be split into two segments *A'* and *C*.

5 Temporal Pattern Mining

Now that a preliminary segmentation of the investigated song has been found, a post processing step helps to further investigate the segmentation by searching for repeated sequences in the preliminary segmentation.

If the exact temporal structure of the individual intervals is ignored, a simplified representation of the segmentation, only based on their temporal ordering, can be defined. In addition to the found, labeled time intervals, each time interval not related to any label is denoted as *structural gap* G. This simplified depiction of the musical structure can be represented as a sequence $\mathbb{S}$ of labels or symbols.

$$\mathbb{S} = \{G \rightarrow A \rightarrow B \rightarrow C \rightarrow G \rightarrow A \rightarrow B \rightarrow C \rightarrow C\}$$

When analyzing this sequence, always identically repeated sub-sequences (e.g. $A \rightarrow B$) represent the repeated sequential occurrence of certain prototypes and therefore indicate potentially over segmented areas. According to [Moe10], such sequences are called *substring patterns* as they could be replaced by single super-strings (merged prototypes) $AB = \{A \leftrightarrow B\}$.[6]

To investigate if such an over segmentation may have occurred, the found substring patterns inside the given preliminary segmentation have to be further processed.

5.1 Substring Patterns

Let us assume N consecutive musical patterns forming one repeated sequence. By removing different borders between these patterns, one can form multiple new sequences, all describing a different segmentation of the initial sequence. All (new) sub sequences that could be included is such a segmentation are shown in Figure 28, for a given sequence of 4 patterns.

To decide which of these segmentations optimally represents the underlying musical pattern(s), the quality of each potential segmentation has to be evaluated. This quality can be based on two factors: the length of the combined patterns and the similarities among all patterns that have to be combined to form a specific segmentation (see Section 5.2).

Finding the optimal combination of all sections forming a subsequence can be formulated

[6]Although some of these patterns may just have been generated by applying the novelty function, most of them originate from operations resolving overlapping conflicts of intervals.

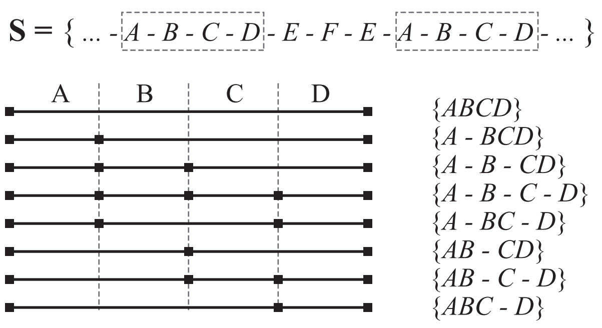


Figure 28: Example for a substring of length 4 $\{A - B - C - D\}$ and all possible sub sequences based on the deletion of different borders.

as a path search in a weighted graph[7] $\mathbb{G}(V, E)$, where V is a set of vertices v_i representing all possible combinations of neighboring sections (see Figure 29) and E is a set of directed edges between adjacent vertices e_i. By defining adequate quality measures for each vertex and edge, the vertices traversed by the minimum-cost path through this graph will represent the optimally combined-sections of the investigated subsequence.

Therefore, an algorithm to find all relevant substring patterns in the given segmentation as well as quality measures for the vertices and edges of the graph $\mathbb{G}(V, E)$ have to be defined.

5.1.1 Substring Search

Starting with the first symbol in $\mathbb{S}$, the search for substrings is performed by iteratively comparing all positions of individual symbols. As long as all positions of two (or multiple) symbols always follow each other, a substring is generated (see Algorithm 3).

The resulting set $\boldsymbol{s}_{all}$ will contain all substrings (longer than one symbol) found for the given symbol sequence. To find the final set of relevant substrings, all substrings in the set are compared. Starting with the most frequent substring, which is naturally relevant, all other substrings are compared to the symbols included in the relevant string and substrings containing any of the symbols are deleted. This procedure is repeated for all other substrings in the order of their number of occurrences.

[7]More precisely, the graph representing all possible combinations of segments is a directed acyclic weighted graph, as the direction of the edges in the graph is defined and no path starting at a certain vertex will lead back to this vertex.

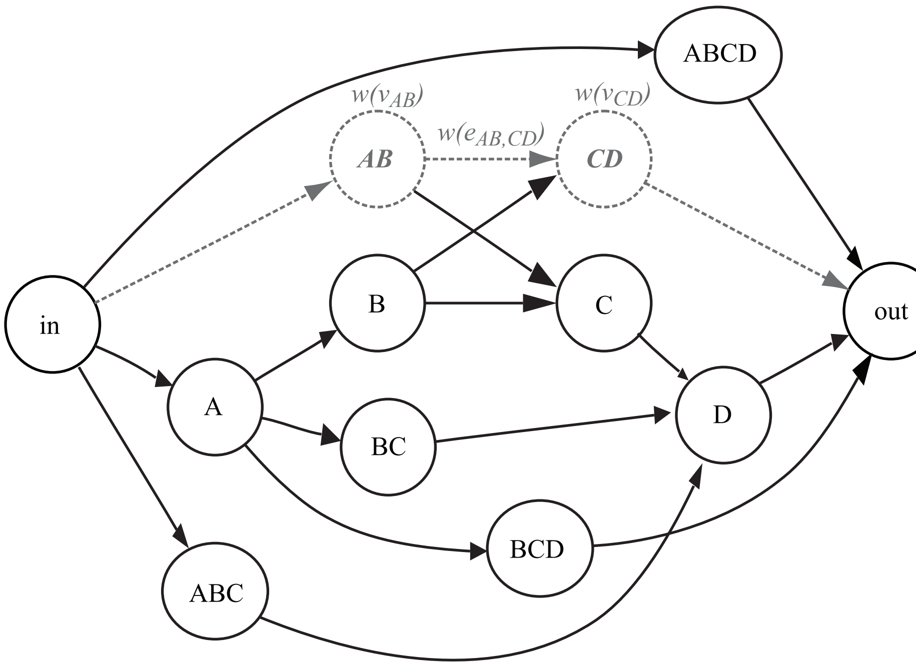


Figure 29: Example for a graph representation of all possible combinations of 4 patterns. The exemplary path $\{in \rightarrow AB \rightarrow CD \rightarrow out\}$ (indicated by dashed gray lines) shows the weights included if the patterns A and B and the patterns C and D are combined.

5.2 Quality Measures

The conditions to evaluate if and how prototypes and their related musical patterns forming a substring should be combined, can be based on the following basic rules:

- A high similarity between two patterns (prototypes) indicates that the patterns should be combined, while a low similarity indicates that the patterns should better remain separated.
- Based on a *desired* length λ_d, combined patterns (e.g. AB) leading to a combined length λ_{AB} close to λ_d should be merged, while combined patterns which would be too long (e.g. $\lambda_{AB} \gg \lambda_d$) should better remain split.

Desired Length The definition of a *desired* length λ_d is relatively easy. Based on music theoretical considerations, this length can be set to 8 bars (a typical length for musical segments). A Gaussian weighting function, whose center point is defined by the number of beats representing λ_d, can then be constructed. If the length λ_c of a combined prototype is exactly λ_d bars, the weighting related to this length will be 1, while longer and shorter

Algorithm 3 Search for Substrings

```
1: $\boldsymbol{s}_{all} \leftarrow \{\}$
2: for $k = 1$ to $l - 1$ do
3:     $\boldsymbol{s}_{cur} \leftarrow \mathbb{S}(k)$
4:     $i \leftarrow 1$
5:     while $find(\mathbb{S} = \mathbb{S}(k)) + i = find(\mathbb{S} = \mathbb{S}(k+i))$ do
6:         $\boldsymbol{s}_{cur} \leftarrow [\boldsymbol{s}_{cur}, \mathbb{S}(k+i)]$
7:         $i \leftarrow i + 1$
8:     end while
9:     if $length(\boldsymbol{s}_{cur} > 1)$ then
10:        $\boldsymbol{s}_{all} \leftarrow \{\boldsymbol{s}_{all}, \boldsymbol{s}_{cur}\}$
11:    end if
12: end for
```

combinations are weighted according to the respective values of the Gaussian function (see Figure 30)

$$\boldsymbol{g}(\lambda_c) = e^{-\left(\frac{(\lambda_c - \lambda_d - 1)/\lambda_d}{2\sigma^2}\right)}, \tag{46}$$

where λ_c is the combined length of two or more patterns and σ is set to $1/2.5$. Values falling outside this window (longer than $\approx 2 \cdot \lambda_d$) are set to zero.

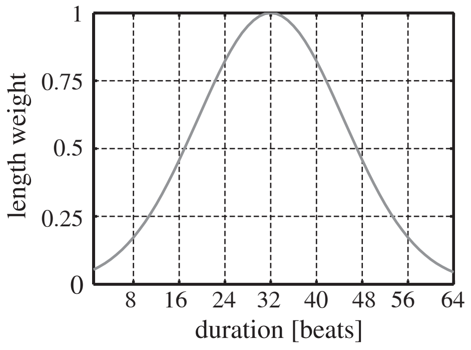


Figure 30: Gaussian weighting window defining the quality of different (combined) pattern lengths for a song played in common time ($4/4$).

Segment Similarities For the definition of the similarities of two or more prototypes, the template matching method introduced in Section 3.2.1 can be used. Previously in the algorithm, NCC has been applied to compare the repetitions of 2 bar slices inside the SDM to find all segment candidates. Similarly, NCC can now be used to compare longer slices of the SDM, representing the individual prototypes of a repeated sequence.

To compute the inter similarity of two prototypes $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$ of a repeated sequence, the respective slices of the SDM have to be compared.

The first occurrence of each prototype inside the SDM, related to the time interval $\boldsymbol{t}_{i,1} = [s_i, e_i]$, is selected to represent the prototype. Therefore, the SDM slice will start at column s_i and end at column e_i. As two prototypes will typically not have the same length, the slice related to the shorter prototype is considered as template image and the longer slice as search image. The similarity of the prototypes is then defined by the maximum correlation value in the resulting correlation vector $\boldsymbol{c}_{ij}$ (see Figure 31).

Combined Quality Measures As each vertex v_i represents one possible combination of prototypes, the weight related to the vertex $w(v_i)$ is defined by the *inner similarity* and the *combined length* of the merged sections. Assuming a vertex representing the combination of N sections (defined by N prototypes $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_N\}$, where l_n is the length of prototype $\boldsymbol{P}_n$) , the following weights can be computed:

The average inner similarity

$$w_s = \frac{1}{N-1} \sum_{n=1}^{N-1} max(\boldsymbol{c}_{n,n+1}), \tag{47}$$

the weight of the combined lengths

$$w_l = \boldsymbol{g}\left(\sum_{n=1}^{N} l_n\right) \tag{48}$$

and the final combined weight of the vertex v_n

$$w(v_i) = 1 - \frac{w_s + w_l}{2}. \tag{49}$$

(If a vertex represents only an individual section, its weight is simply defined by its length: $w(v_i) = w_l$)

Accordingly, the weight of an edge $w(e_{ij})$ is defined by the inter similarity of the (combined) prototypes connected by the respective edge.

5.2.1 Path Search

Considering the graph $\mathbb{G}(V, E)$ (see Figure 29) representing all possible combinations of subsets and all the weights that have been defined for the edges and vertices of the

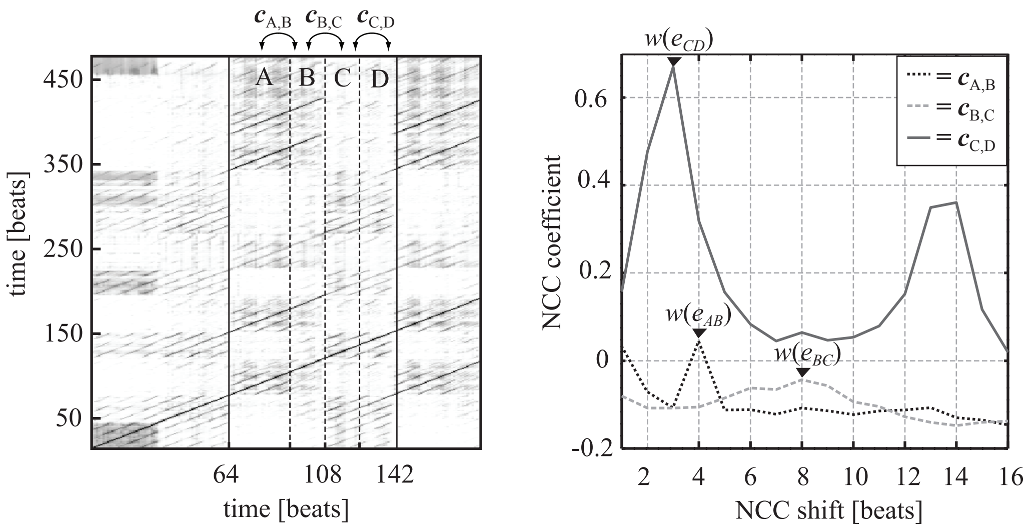


Figure 31: To find an estimate of the similarity of two or multiple musical patterns, the respective slices inside the mapped SDM (left) are correlated. In this example, the patterns C and D should be combined, as they represent almost identical SDM slices, while the other patterns should remain split (no correlation).

grap, all paths through the graph can now be evaluated, to find the optimum combination of subsets. A complete path $\boldsymbol{P}$ represents one possible way from the *origin* (v_0) to the *destination* (v_N), by traversing the respective edges inside the graph. Therefore, given an ordered list $\boldsymbol{L}$ of length N_p with the indices of all vertices belonging to a possible segmentation, the related path can be represented as a list of all vertex pairs.

$$\boldsymbol{P} = \{(v_0, v_{\boldsymbol{L}(1)}, \dots, v_{(\boldsymbol{L}(N_P))}, v_N\} \tag{50}$$

The cost of the path is the combined cost of all traversed vertices and edges.[8]

$$c(\boldsymbol{P}) = \sum_{n=1}^{N_p-1} w(v_{\boldsymbol{L}(n)}) + w(e_{\boldsymbol{L}(n),\boldsymbol{L}(n+1)}) \tag{51}$$

By finding the minimum cost path in the set of all paths, $\mathbb{P} = \{P_k | k = 1, \dots, N_k\}$, the path representing the optimum segmentation of the related subsequence can be found.

$$\boldsymbol{P}_{opt} = \arg\min_{\boldsymbol{P}}\{c(\boldsymbol{P})\} \tag{52}$$

[8]As v_0 and v_N and their related edges only represent the origin and ending of the graph, these vertices and related edges are not related to any weights.

5.3 Final Segmentation

After incorporating the information regarding potential sub-sequences into the preliminary segmentation, the musical structure of the investigated song is almost completely defined. In a last step, short gaps < 2 bars are simply filled by the segment following the gap and larger gaps > 2 bars are defined as non repeated segments and are provided with a unique symbol. Figure 32 shows an example for a typical segmentation found by the algorithm (more example segmentations are given in Section 6.4).

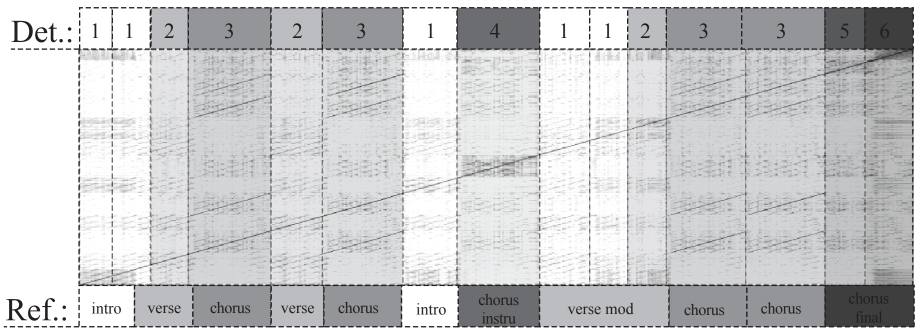


Figure 32: Autoamtically detected segmentation (*Det.*), SDM and reference segmentation (*Ref.*) of the song *"Head Over Feet"* by *Alanis Morisette*.

6 Evaluation and Results

Defining a ground truth for musical segmentation is a very crucial task as people tend to define transitions between segments very differently. For example, repetitions within a *verse* or any other section can be annotated as one large or two smaller parts, depending on the subjective interpretation of temporal structures of the annotating person. Therefore, multiple segmentations of one and the same song could be considered as right or wrong, depending on which ground truth is used to evaluate the results.

Another problem are repetitions of parts which are not exactly identical to prior occurrences of the part (e.g. new instrumentation). Some people may tend to assign a new label to such a modified repetition, while others may not. Therefore, the musical form of a song is rarely unambiguous and basically only the composers could ultimately define the *true* musical form of their compositions. Still, every evaluation of musical structure has to be based on a pre defined ground truth. A detailed discussion on the problem of the ambiguity of ground truth annotations and the topic of evaluating structuring algorithms can be found in [Pei07].

In literature, many different evaluation metrics for MSD algorithms have been proposed. To allow for a direct comparison to other evaluation results published in literature, the performed evaluation is based on the most commonly used metrics for the respective tasks (standard and pairwise *F-measure*). Additionally, the evaluations have been performed on one of the most frequently deployed datasets in the field.

6.1 Dataset

For the main evaluation of the system, a famous dataset consisting of 174 songs from *The Beatles* (in literature often referred to as *TUT Beatles*) was selected.[9] This dataset has widely been used before by other authors to evaluate MSD systems.

Additionally, an extensive evaluation of 5 well known MSD algorithms (Peiszer [PLR08], Levy [LS08], Barrington [BCL10], Paulus [PK08]) has been conducted in [Smi10] based on this dataset and the published results allow for a direct comparison of the presented approach to the algorithms evaluated in this test. In addition to these algorithms, another recently published result on the same test set (Kaiser [KS10]) was included to the comparison.

[9] http://www.cs.tut.fi/sgn/arg/paulus/structure.html

The 174 songs represent 13 full albums and offer a range of different types of songs, although they have all been composed and played by the same band. As already mentioned, the annotations are publicly available at and have been applied without any modifications.

Besides the main testing corpus, another small dataset (*Mixec* corpus) has been added to the evaluation, to include a little more variety of musical styles. This second, smaller body has been labeled by either professional musicians and/or musicologists for the MPEG-7 working group.[10] The corpus is similar to the one used in [LS08] and consists of 20 pop songs by artists like Alanis Morisette, Björk, Madonna or The Spice Girls. Although the annotations for the selected songs are also publicly available at, a direct comparison to other systems is not possible, as our selection of songs was mainly based on the availability of songs and has not been used in other publications before.

Since the labeling in both sets was done in seconds, all found segment borders had to be converted back from beat-level to seconds, granting a common time base for the automatically detected segmentation and the annotated ground truth.

6.2 Evaluation Metrics

When evaluating MSD system, mainly two qualities of the systems are of interest: the *boundary detection* performance and the *segmentation* performance.

Boundary Detection The boundary detection performance of an algorithm is typically evaluated by standard *F-measure*. This measure does not evaluate any relations of boundaries or found sections, but simply compares the positions of the automatically detected boundaries to the boundaries inside the annotations. As the detections will (almost) never be perfect, most authors allow deviations of ± 3 seconds from the true boundaries.

Based on this threshold, one can define *true positives* (detected border $\rightarrow$ reference border), *false positives* (detected border $\rightarrow$ no reference border) and *false negatives* (missed reference border). By the relations of the resulting values, the *recall* r, *precision* p and the *F-measure* f for the related segmentation can be calculated.

$$r = \frac{truePos}{truePos + falseNeg} \tag{53}$$

$$p = \frac{truePos}{truePos + falsePos} \tag{54}$$

[10]http://www.elec.qmul.ac.uk/digitalmusic/downloads

$$f = \frac{2pr}{p+r} \tag{55}$$

A high *recall* shows that most of the reference borders have been found (but maybe other found borders are wrong), while a high *precision* indicates that most found borders correspond to positions of reference borders (but maybe some reference borders have been missed). Therefore, a high *F-measure* corresponds to a balanced relation of found and missed borders and indicates an overall successful boundary detection performance.

Segment Detection For the evaluation of the segmentation performance, two different evaluation metrics which have already been used in musical structure detection before [LF08] have been selected: the *pairwise F-measure* and the *directional Hamming distance*.

The *pairwise F-measure* F is a standard evaluation metric for clustering algorithms and represents the harmonic mean of the pairwise precision P_r and recall rate R_r. Given a set of identically labeled pairs of beats in the reference segmentation $\mathbb{G}_r$ and the same type of set in the automatically detected segmentation $\mathbb{G}_d$, the measures are defined as follows: ($|\cdot|$ denotes the cardinality of the respective set)

$$P_r = \frac{|\mathbb{G}_d \cap \mathbb{G}_r|}{|\mathbb{G}_d|} \tag{56}$$

$$R_r = \frac{|\mathbb{G}_d \cap \mathbb{G}_r|}{|\mathbb{G}_r|} \tag{57}$$

$$F = \frac{2 \cdot P_r \cdot R_r}{P_r + R_r} \tag{58}$$

A low pairwise *precision* rate is an indicator for under segmentation, while a low pairwise *recall* rate indicates over segmentation. The pairwise *F-measure* therefore describes an overall quality of the found segmentation (see Figure 33).

The second metric is the directional Hamming distance [ANSS05]. Given the reference segmentation as a sequence of n segments $\mathbb{R} = \{S_{\mathbb{R}}^1, S_{\mathbb{R}}^2, \ldots, S_{\mathbb{R}}^n\}$ and the automatically detected segmentation as a sequence of m segments $\mathbb{D} = \{S_{\mathbb{D}}^1, S_{\mathbb{D}}^2, \ldots, S_{\mathbb{D}}^m\}$, the directional Hamming distance is denoted by $\mathbb{D}_H(\mathbb{R} \Rightarrow \mathbb{D})$. For each segment $S_{\mathbb{D}}^i$ from the detected segmentation, a segment $S_{\mathbb{R}}^j$ from the reference segmentation is associated so that the overlap of the segments $S_{\mathbb{D}}^i \cap S_{\mathbb{R}}^j$ is maximal. The directional Hamming distance is then defined as:

$$\mathbb{D}_H(\mathbb{R} \Rightarrow \mathbb{D}) = \sum_{S^i_{\mathbb{D}}} \sum_{S^k_{\mathbb{R}} \neq S^j_{\mathbb{R}}} |S^i_{\mathbb{D}} \cap S^k_{\mathbb{R}}| \tag{59}$$

Similarly, the *inverse* directional Hamming distance can be symmetrically computed for $\mathbb{D}_H(\mathbb{D} \Rightarrow \mathbb{R})$. Normalizing the resulting distances by the number of beats N of the underlying track allows us to derive two error rates: the missed boundaries (or miss rate) $m = \mathbb{D}_H(\mathbb{R} \Rightarrow \mathbb{D})/N$ and the segment fragmentation (or false alarm rate) $f = \mathbb{D}_H(\mathbb{D} \Rightarrow \mathbb{R})/N$. High values of m (low values of $1 - m$) indicate under segmentation, while high values of f (low values of $1 - f$) indicate over segmentation (see Figure 33).

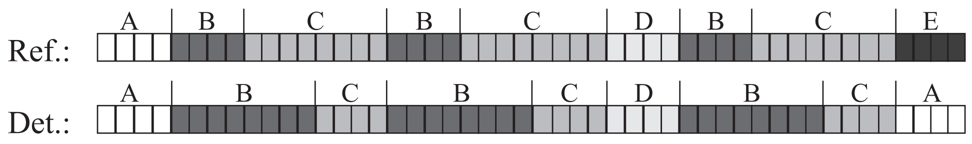


Figure 33: Segmentation example: R_r 60%, P_r 62.5%, F_r 61.2%, $1{-}m$ 0.66, $1{-}f$ 0.75

6.3 Results

Table 1 shows the overall boundary detection performance as well as the overall segmentation performance of the system presented in this thesis on the *TUT Beatles* dataset. The results are based on standard F-measure (*Boundaries*) and pairwise F-measure (*Segments*). They may be of particular interest, as they can directly be compared to other results published in literature.

To the best of our knowledge, Table 2 shows all previously reported evaluation results based on the same evaluation metrics and the same testing corpus. The direct comparison to Table 1 shows that the presented system outperforms all other systems in terms of the achieved F-measures for both, the boundary detection as well as the segmentation performance.

Although the increase in performance for the boundary detection does not show a significant difference ($p < 0.05$) to the best priorly reported result, the performance increase for the segmentation is significant and proofs the stability of the presented system.

As already mentioned, in addition to the main evaluation, another small test on a second dataset (*Mixed* corpus) has been performed. The results for this set are summarized in Table 3. The evaluation results on the full segmentation are almost identical to the results

Table 1: Segmentation Results (TUT Beatles)

	R (%)	P (%)	**F** (%) $\perp$ CI
Boundaries	69.8	62.8	**64.2** $\perp$ 3.5
Segments	72.3	67.6	**67.9** $\perp$ 2.3

Table 2: Previously reported results (TUT Beatles)

	Boundaries (F%)	Segments (F%)
Peiszer	61.7	59.6
Levy	58.1	59.6
Barrington	55.5	57.2
Paulus	55.0	59.9
Kaiser	-	62.1

based on the *TUT Beatles* corpus, indicating the stability of the segment detection performance of the system. At the same time it is interesting to note that the boundary detection results are significantly worse.

In the next Section, the reasons for the diverging boundary detection results for the two testing corpora will be discussed based on a selected segmentation example. It will also be shown, why the boundary detection results do not always influence the segmentation performance of the system.

Table 3: Segmentation Results (Mixed)

	R (%)	P (%)	**F** (%) $\perp$ CI
Boundaries	62.8	58.3	**57.1** $\perp$ 9.7
Segments	70.1	65.9	**65.6** $\perp$ 6.2

As additional song-based evaluation metrics, the *missed boundaries* $(1 - m)$ and the *segment fragmentation* $(1 - f)$ based on the Directional Hamming distance have been computed. The scatter plots in Figure 34 based on these two metrics indicate a relatively well balanced relation between slightly over- and under-segmented songs. This shows that the algorithm does *not* suffer from a tendency to generate over- or under-segmented descriptions of the musical structure (see Section 6.4).

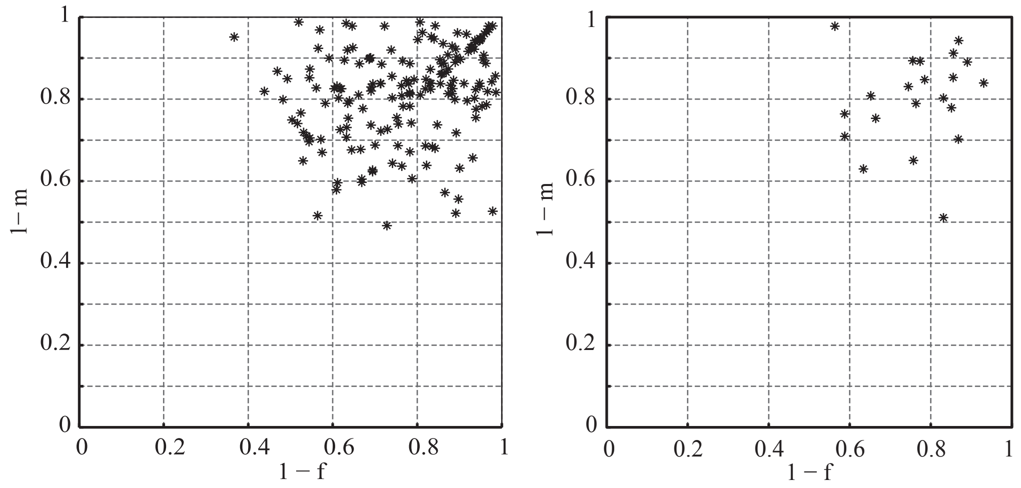


Figure 34: Scatterplot of *segment fragmentation* against *missed boundaries* for the *TUT Beatles* corpus (left) and the *Mixed* corpus (right).

6.4 Discussion

The results presented in the previous Section proof that the concept of template based SDM analysis followed by the proposed processing chain is an interesting and promising alternative to other SDM based methods. To outline some of the strengths and problems of the algorithm, a short list of example segmentations will now be presented. The respective SDMs are included in the examples to show which *patterns* of the SDM have been selected by the algorithm to represent relevant sections of the song.

The following examples always show the detected segmentation (*Det.:*), the reference segmentation (*Ref.:*) and the related SDM. Additionally, a summary of the individual evaluation results related to each example is given in Table 4.

Example 1**: *Perfect Segmentation*** The first example shows the (almost) perfect segmentation of a Beatles song. Although the main patterns found by the algorithm are relatively obvious, a method based on the pure detection of off diagonal lines may not have been able to correctly detect the correct *verse* sections. An algorithm only based on the detection of rectangular blocks obviously would have failed.

Example 2**: *Typical Segmentation*** The second example shows a more or less *typical* segmentation produced by the algorithm. Although this time the patterns inside the SDM are not as obvious as before, most of the segment borders have been detected correctly

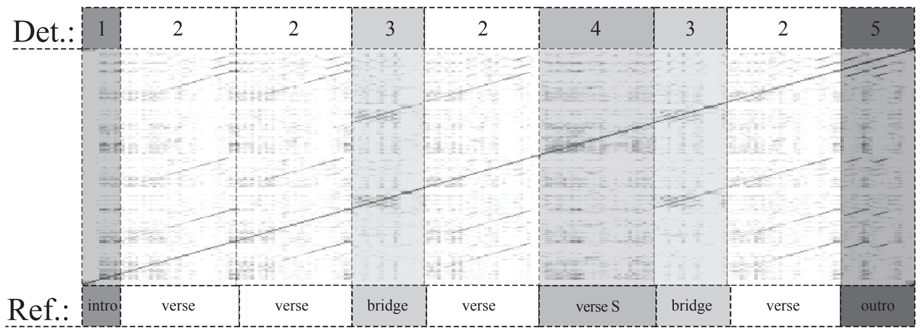


Figure 35: (Example 1) *The Beatles - I Saw Her Standing There*

(slightly shifted related to the ground truth). When looking at the patterns inside the SDM, the found segmentation is reasonable, although it does not perfectly match the given ground truth segmentation. The pattern representing the second verse has been identified as individual segment and the final refrain section has been split into a refrain section and an individual outro section.

As already mentioned, this result is typical for content based segmentation algorithms. Even if a good representation of repetitive sections based on the pure musical content can be found, there is no guarantee that the found sections also perfectly represent the musical form of the song, as many other important characteristics influencing the musical form, like lyrics or the musical style, can not be incorporated into the found representation.

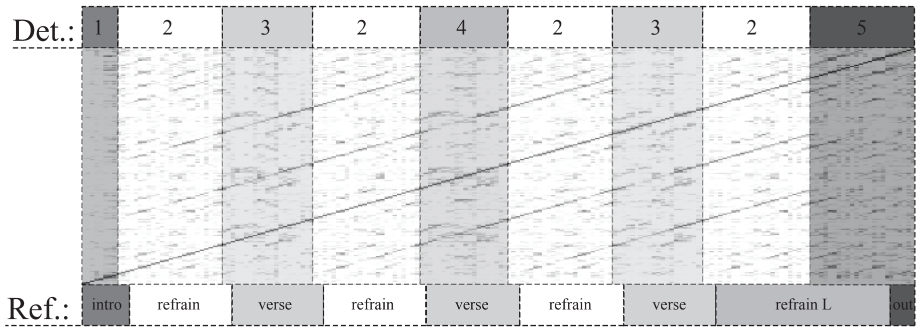


Figure 36: (Example 2) *The Beatles - Devil In Her Heart*

Example 3: *Over- and under-segmentation* The following two examples show two typical cases of over- and under-segmented songs. The first example (under-segmentation) shows that the main repetitive structure inside the SDM has correctly been detected by the algorithm, but the fine-structure given by the reference segmentation has been missed.

Although this specific example also shows some inconsistencies in the reference segmentation (the different segmentation of *verses* and *refrains*), the results show that sequential structures which are always identically repeated (and do not represent a concatenation of two vertical blocks) can not be separated by the presented algorithm.

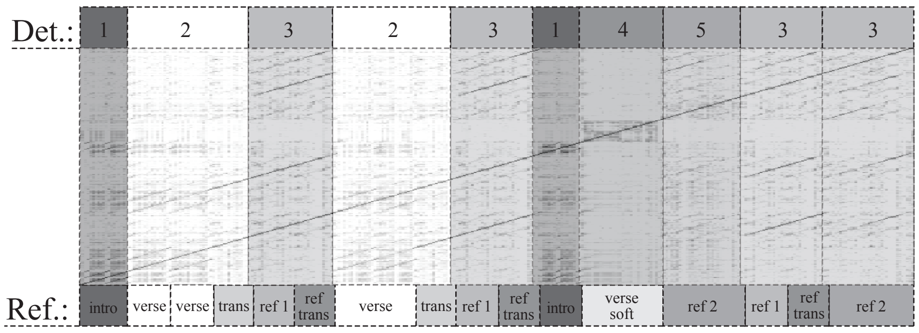


Figure 37: (Example $3a$) *Britney Spears - Hit Me Baby One More Time*

The second example shows the over-segmentation of the *verse* of a song. Again, the structures inside the SDM have correctly been detected by the algorithm (this time, also an algorithm based on the detection of off-diagonal lines and vertical blocks would presumably have succeeded), but they are not in-line with the given reference segmentation. Additionally, the *intro* of the song has not been detected correctly.

When listening to the song, the errors made by the algorithm can easily be explained as the intro is musically almost identical to the second part of the verse and the verse consists of two musically significantly different parts. This example again shows that reference segmentations representing the musical form of a song are often influenced by additional factors besides the pure musical similarity.

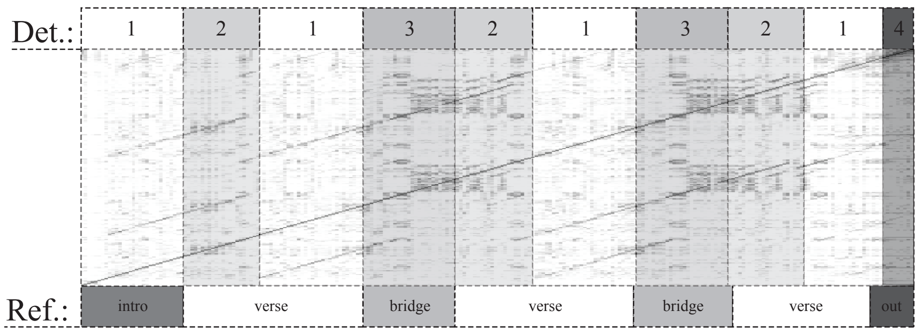


Figure 38: (Example $3b$) *The Beatles - All I've Got To Do*

Example 4**: *Border Detection and Structure Detection*** As mentioned in the previous Section, the border detection results for the *Mixed* corpus are significantly worse than the results achieved for the *TUT Beatles* corpus. This difference may be explained based on the following observation.

Most songs played by *The Beatles* do not show a strong "inner" similarity of individual sections. In contrast, the verses and choruses of pop-songs are often composed by two or more repetitions of basically identical parts. In such cases, the algorithm often identifies all parts as individual repetitions of the same section, leading to too many borders in the segmentation and poor boundary detection results (see Figure 39).

The evaluation results for the full segmentation do not indicate this over-segmentation, as all sections belong to the same part. Comparing the labels related to each beat with ground truth labels (see Section 6.2) does not reveal borders between identical parts. Therefore, this "inner" over-segmentation of segments can only be identified by looking at the border detection results.

The following example shows a typical case of multiple "inner" over-segmented parts. In particular the final section based on the repetition of multiple identical parts (often slightly modified repetitions of the chorus) is typical for many pop songs. The algorithm separates these sections into individual repetitions, although they are only represented as one section in the annotated ground truth.

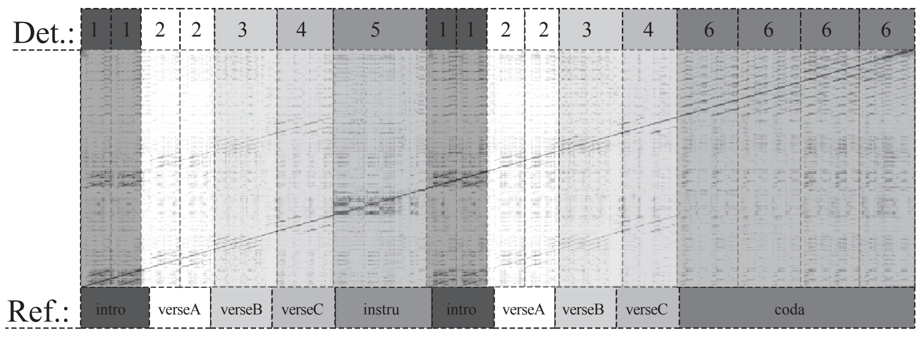


Figure 39: (Example 4) *Chicago - Old Days*

Example 5**: *Repeated Sequences*** The last example shows a song which is mainly composed of one repeated musical sequence (except for the chorus sections). Although the algorithm is able to find this repeated sequence, the sequence is split at a position which does not represent the real musical structure of the song. This happens, as a part of the sequence already occurs in the second half of the intro. The algorithm detects this part and as a consequence interchanges the first and second half of the sequence.

Although the segmentation performance is acceptable (see Table 4), the boundary detection performance is extremely bad, as all found boundaries (expect for the boundaries of the chorus sections) are mis-aligned with respect to the reference boundaries.

This example was also chosen to show that segmentation algorithms purely based on repetitive structures will always fail on songs, where individual sections do not musically differ. As already mentioned, in these cases, new approaches incorporating the detection of a singing voice or even the lyrics would probably be the only way to select the right borders.

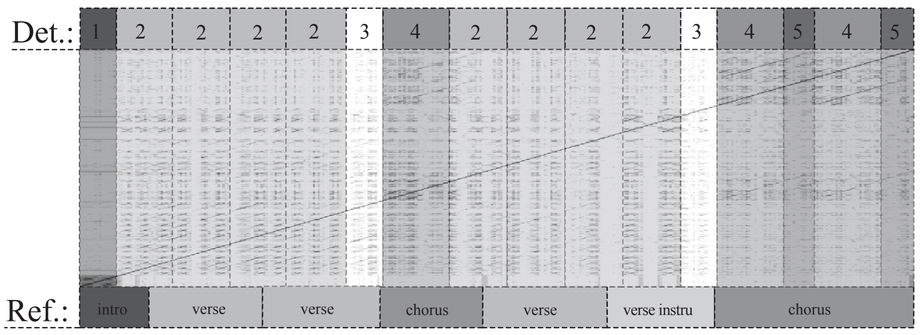


Figure 40: (Example 5) *The Clash - Should I Stay Or Should I Go*

Table 4 shows the individual evaluation results for all example segmentations. It may be interesting to note that some results may be worse than expected while others may be better than they should "intuitively" be. This shows that the chosen evaluation metrics will not always be able to perfectly describe the segmentation performance of a system, but still give a (strong) indication, if a system generally produces reasonable results or not.

Table 4: Evaluation results for the example segmentations

	Song Title	r	p	f	R_r	P_r	F_r	$1-m$	$1-f$
1.	I Saw Her Standing . . .	1.00	1.00	1.00	0.96	0.95	0.96	0.98	0.98
2.	Devil In Her Heart	0.87	0.87	0.87	0.62	0.68	0.64	0.80	0.74
$3a$.	Hit Me Baby. . .	0.53	0.89	0.66	0.40	0.79	0.54	0.54	0.87
$3b$.	All I've Got To Do	0.80	0.44	0.57	0.78	0.52	0.62	0.80	0.66
4.	Old Days	1.00	0.50	0.66	0.89	0.75	0.81	0.94	0.86
5.	Should I Stay Or. . .	0.33	0.13	0.19	0.61	0.68	0.64	0.78	0.76

7 Segment Related Genre Classification

As already mentioned in Section 1.1.2, knowing the musical structure of a song can be useful in many applications, in particular in the field of MIR. It has also been stated that the most obvious applications profiting from such prior knowledge are automatic thumbnail generation and improved inter section navigation for music players. Both applications can directly use the information extracted by the MSD system without any sophisticated post processing.

Besides such obvious fields of applications, the musical structure could also be used to improve or modify other higher level MIR systems, as it enables them to focus their processing on specific parts of the song. For example, systems for automatic instrument recognition could process each part separately and exploit the dependencies of parts - as identically labeled parts will typically have similar instrumentations.

In this Section, another potential area of application that may profit knowledge about the musical structure of songs is investigated: automatic genre classification. Since most genre classification systems use feature vectors which are computed either from the whole audio file or from short arbitrary excerpts to find an adequate genre label [SZM06], these systems may profit from restricting this excerpt to representative parts.

Although no explicit detection of the most representative part(s) (e.g. a chorus detection) has been performed in the segment detection system presented in this thesis, one may still state that the most representative parts are also the most frequent parts (which will often be the chorus sections).

Based on this idea, three experiments have been performed to investigate the influence of knowing something about the internal structure of a song (extracted with the presented algorithm) when aiming to extracts its genre label [WS11]. After a short introduction to the field of musical genre classification, the performance of a state of the art genre classification system based on different types of input data (generated with and without prior knowledge on the underlying song structures) will be evaluated.

7.1 Automatic Genre Classification

Due to the constant growth of music data archives, the automatic classification and organization of large song collections has become a prominent issue. Genre classification is of particularly great importance, since the genre label is still one of the most relevant descriptors when organizing and browsing large music collections.

Musical genres evolved over the years as categories to describe certain common similarities and characteristic of different types of music. Therefore, the musical genre does not represent an immanent attribute of a song like its tempo or melody. Additionally, the on-going evolving process led to many blurred boundaries among different genres and varying genre taxonomies [PC00]. Considering this problematic concept, automatic genre classification purely based on the audio content of a song without any additional cultural information, is a very complex task with a lot of ambiguities.

Despite this problems, a lot of different systems for automatic genre classification have successfully been developed during the last decade. Therefore, a short review on the principle task of genre classification will be given, before the conducted experiments are presented in Section 7.3. This overview may help to better understand the positioning of the proposed pre-processing step as a part of a complete genre classification system.

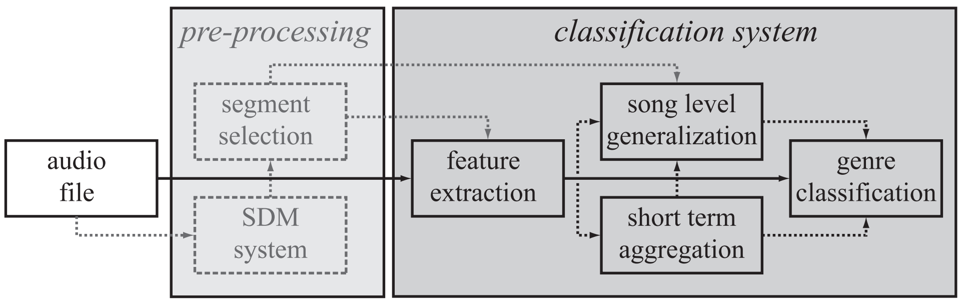


Figure 41: As indicated, focusing the classification to selected segments based on a pre-processing stage, does not modify the basic classification algorithm, but only provides additional side information. (Dashed lines indicate optional paths.)

Feature Extraction

Similar to most pattern classification systems, a set of appropriate features has to be extracted from the audio file in a first step. Typically, the audio file is split into short (e.g. $\approx 50ms$) (overlapping) frames (sometimes called *analysis windows*) and multiple features are computed within each frame. A large amount of frame-based features have been suggested for automatic genre detection [TC02], [Pee04], [LR05], but typically low-level descriptors based on a spectral representation of the investigated audio frame are used.

If higher-level features are used, aiming to describe certain long-term characteristics of the file, like rhythmic, harmonic or timbral movements, the corresponding analysis frames have to be expanded to longer windows (e.g $\approx 1 - 3s$), as short analysis windows can not capture any information on the temporal evolution (rhythm) or tonal stability (harmony) of an audio file. Therefore, either longer *analysis windows* are directly applied, or the evolution of certain features is observed over multiple time frames (see Short Term Aggregation).

Each feature vector $\boldsymbol{x}$ extracted in this first stage, is composed of the individual feature values x_i and describes a point in a d-dimensional feature space χ.

$$\boldsymbol{x} = (x_1, \ldots, x_d)^T, \quad \boldsymbol{x} \in \chi$$

Short Term Aggregation (optional)

As already mentioned, depending on the used set of features, it may be necessary or at least beneficial, to observe certain features over multiple frames, to compute a perceptually relevant description of the investigated audio file. The temporal dimension of such an aggregation window can either be based on a fixed length (*texture windows*) [TC02] or on specific characteristics of the song (e.g. inter onset intervals [Wes05], beat intervals [SZ05]). Typically, the aggregation is either performed by using the low order statistics (*mean, variance*) of individual features over multiple time frames, or by defining a feature extraction process which directly covers the respective amount of frames.

Song Level Generalization (optional)

Most approaches generalize the sequence of individual (aggregated) feature vectors into a single, song-level feature representation, before the final classification is performed. This can be done by applying a summarization function to the individual dimensions of the extracted feature vector sequence. Typical summarization functions are the mean, variance, median or certain percentiles. The resulting values can either be used to describe a statistical model of the temporal distribution of features (e.g. individual Gaussians [LS01], Gaussian Mixture Models [TC02]) or, if the statistical descriptors are *unwrapped* they can directly be interpreted as one new, song-level feature vector [ME05].

Classification

At this point, either multiple frame level feature vectors, multiple aggregated feature vectors, or a song level representation (song level vector/distribution) will form the input of a classifier. Basically any known classification scheme can now be applied to the provided features.

The two main approaches to classify the given data into genres are based on *supervised* and *unsupervised* classification. While supervised classification systems are trained based on a certain genre taxonomy, unsupervised systems cluster the data in a non-supervised way and the resulting classes will emerge purely from the objective similarities of features. Since the genre label is not really an immanent attribute of a song, but influenced by cultural factors, supervised classification methods, have been found to perform in a more stable way than unsupervised systems [SZM06], as they profit from the cultural information provided by a given set of genre labels.

Depending on the type (vectors, distributions) of feature representation, the main classification techniques used for genre classification may be outlined as follows:

- distances between aggregated vectors
  - K-nearest neighbors (K-NN), e.g. [TC02]
- distances between statistical models
  - Earth mover's distance (EMD) e.g. [AP02]
- decision boundaries between aggregated vectors
  - Support Vector Machines (SVM), e.g. [SWP10]
  - Artificial Neural Networks (ANN), e.g. [SZ05]
- statistical classifiers
  - Gaussian mixture models (GMM), e.g. [TC02]

If not one generalized representation over the whole song is used for classification, but individual representations on the level of frames or aggregation windows, the individual classification results have to be combined. One simple but famous technique to perform this operation is majority voting, where the class assigned to the majority of related frames or windows is considered to be the class for the respective song.

Pre-Processing: Segment Selection

The advantages and disadvantages of different combinations of features, aggregation techniques and classification schemes have extensively been discussed in literature [SZM06], [TL08]. Still, it is important to note that despite their differences, all systems have one major thing in common: the set of feature vectors (no matter what features are used), has to be extracted from certain section of the investigated song. Some approaches extract the features from the whole song, while many other approaches use a 30 second window of each song starting 30 seconds after the beginning of the song (to exclude non-representative intro) to extract the different features.

Based on this observation, the influence of focusing the section(s) used for feature extraction to representative parts of a song will now be investigated based on a state-of-the-art genre classification system. It has to be noted that the following experiments do not directly aim to improve or modify of the given classification system, but they have been designed to *investigate* the influence of an additional pre-processing stage (see Figure 41) on the resulting classification performance.

7.2 Selected Classification System

The genre classification system selected for the presented experiments was developed by Klaus Seyerlehner et al. [SWP10][11] and ranked in the top positions of the *MIREX 2009* genre classification task.[12] It is based on a block processing framework and uses a set of spectral descriptors (Spectral Pattern, Delta Spectral Pattern, Variance Delta Spectral Pattern, Logarithmic Fluctuation Pattern, Correlation Pattern, Spectral Contrast Pattern) extracted within windows of different lengths. To get a final representation of the extracted features, all features are generalized over the whole file using different generalization functions. A detailed description of the block processing framework is given in [SS09] and a detailed description of the used features can be found in [SWP10].

The feature vector resulting from the generalized spectral descriptors, is then fed into a Support Vector Machine (SVM). The SVM implementation provided by the *WEKA toolbox* [HFH$^+$09], a powerful open source classification toolbox based on the Java programming language, has been used to perform the classification task. Following the settings of the selected reference system, the standard parameters of the WEKA SVM are used.

[11]We would like to thank the authors for kindly offering their prototype system to perform our experiments.

[12]http://www.music-ir.org/mirexwiki

7.3 Classification Experiments

7.3.1 Testing Corpus

The songs selected for the experiments have been randomly drawn from a publicly available dataset.[13] A subset of 8 genres was choosen from the set, aiming to include a wide range of different styles: Alternative & Punk, Classical, Dance, Easy Listening, Hip-Hop, Jazz & Blues, R&B, Rock & Pop. For the experiments, 40 songs per genre have been selected, leading to a total corpus of 320 songs.

It has to be mentioned that some "ill-defined" or potentially overlapping genres (e.g. Rock & Pop, Alternative & Punk) have intentionally been included into the selected list, as the task should mimic a real world situation, where no selection on the used genre taxonomies may be possible. Although the problem of overlapping genre taxonomies and ambiguities within genres was proofed when listening to individual songs, no modifications on the given genre labels have been performed, as the original labels have directly been selected by the respective artists.

7.3.2 Thumbnail Generation

Naturally, the musical structure of a song can directly be used to generate representative previews or thumbnails of the respective song. Assuming that the most frequently repeated parts also represent the most characteristic sections, thumbnail generation will typically focus on these parts.

For the experiment, the following thumbnails of different lengths have been generated:

- $30s$ section, consisting of a concatenation of the most frequent segments (max. 3, where always the first occurrence of each segment is used)
- $10s$ section, consisting of the most frequent segment
- $5s$ section, consisting of the most frequent segment

Individual parts of composite thumbnails are concatenated by beat synchronous cross fading, to allow salient transitions between the parts. As typically only sections of a musical segment are used for the thumbnails, these sections are extracted from the center of the respective part.

[13]http://www.seyerlehner.info

In particular the 30 second parts turned out to be excellent song previews that could directly be used as classical thumbnails in a music library, although such an implementation was not the goal of the experiment. The 10 and 5 second thumbnails may be too short to use them as classical music previews, but they are generated to evaluate how much information of a song is really needed to perform an automatic classification of its genre.

In addition to these thumbnails, a special version of the song excluding all non-repetitive parts has been generated, consisting of a concatenation of all repeated segments. The proposed list of thumbnails including this representation form *Set 1* of the experiment.

The second set, *Set 2*, consists of segments of the same lengths as the thumbnails, but in contrast to *Set 1*, they are simply extracted 30 seconds after the beginning of each song. Therefore, they form more or less arbitrary sections that may - or may not include representative parts. Also the non-modified, full song is included into this set.

7.3.3 Experiments

In a first step, the segmentations of all 320 files are extracted using the segment detection system presented in this thesis. The resulting segments are then individually saved into sub-files. Based on these files, the proposed set of thumbnails is generated, resulting in 7 additional files per song (3 representative thumbnails, 3 arbitrary segments and the complete song without non-repetitive parts).

Next, the feature extraction stage of the selected genre classification system is applied to extract one feature vector from each available representation of the song: the full song, all individual song parts (for repeated parts, only the first occurrence of each part is used) and the 7 additional thumbnails. The stored feature vectors are then converted into the ARFF file format, as this format is the standard input format for the WEKA classification toolbox [HFH^{+}09].

Based on this set of feature vectors, the following classification experiments have been conducted:

Experiment 1 The aim of the first experiment was to investigate how the classification performance changes, if the features used for classification are extracted from representative sections of the song instead of arbitrary parts. Therefore, all files of the previously defined sets (*Set 1* and *Set 2*) are classified into genres, allowing a direct comparison of all pairwise results (e.g. 30 second parts from both sets).

The experiment was conducted using the option of 10 times 10 fold cross validation provided by the WEKA toolbox, to retain statistically meaningful estimates of the achieved classification accuracies.

Experiment 2 In the second experiment all segments of a song are individually classified into a genre.[14] Based on the individual classification results, an extended majority voting rule is applied to find a common genre label for the song.

To incorporate the *duration* of all labeled segments, not only the number of detections related to each genre is evaluated (e.g. 3 parts $\rightarrow genreA$, 2 parts $\rightarrow genreB$), but the durations of all song parts covered by each genre are compared. Naturally, the predominant genre covering the largest proportion of the song is applied as common genre label to the investigated song.

Each run of the experiment was conducted using simple 10 fold cross validation, as the train and test sets of each run had to be manually selected to avoid that parts of the same song occur in the test and training set.

Experiment 3 The main intention of the last experiment was to investigate the inner (genre) homogeneity of files. The individual classification results from the second experiment are analyzed to see if mis-classifications are mainly caused by too much inner diversity of songs, making it difficult to select one genre label for the song (e.g. individual genre labels for each part), or if the main problem is based on the inherent problem of blurry or ill-defined borders among certain genres.

7.4 Results

Experiment 1 The results of experiment 1 are summarized in Table 5. Results showing significant differences between pairs of thumbnails according to the pairwise t-test included in the WEKA toolbox (based on a confidence level of $p = 0.05$), are marked with an asterisk ($*$).

Based on this measure, the tests on the given datasets show a significant difference in classification performance for all thumbnails ≤ 10 seconds, but no significant differences for the 30 second thumbnails and the whole song (with and without non repetitive parts).

[14]More precisely, not all individual segments of the song are classified, but only the first occurrence of each repeated section. The classification of this segment is assumed for all other occurrences of the segment.

Therefore, the results indicate that the classification performance of the selected genre classification system is not very sensitive to the selection of test and training data, as long as the "hit by chance" of a representative section is relatively large (e.g. 30 seconds). (This assumption is qualitatively proofed in experiment 3.) Still, it could be shown that focusing feature extraction to representative parts may be an important strategy when aiming to reduce the area selected for feature extraction.

Table 5: Classification Accuracies (%) | $mean(std)$

Section	Set 1	Set 2
Full Song	59.16 (1.39)	60.13 (1.32)
30 sec	58.69 (1.35)	58.13 (1.44)
10 sec	56.91 (1.55)$*$	52.75 (1.28)
5 sec	51.06 (1.69)$*$	47.63 (1.15)

Experiment 2 In the second experiment, the individual parts of each song have been classified into individual genres. Based on these detections, a single genre label for each song has been found using an extended majority voting rule (see introduction to experiment). The overall classification accuracy for this test was 58.8%, showing no significant difference to the results achieved by the direct classification of the full song or the concatenation of all repeated parts of a song (see Table 5). This again indicates that focusing the classification on specific parts when trying to find a common genre label does not automatically help to improve the overall classification accuracy.

As expected, the majority of songs related to well defined genres like *Classical* or *Hip-Hop* has been labeled correctly, while the results of vague genres like *Pop & Rock* or *Easy Listening* show a high amount of mis-classifications (see Table 6). For example, only 27% of the segments related to songs of the category *Rock & Pop* are marked according to their reference genre.

To investigate, if the found mis-classifications are mainly caused by too much inner diversity of songs, making it difficult to assign one genre label to the song (e.g. too many individual genre labels inside a song), or if the main problem is the inherent problem of blurry or ill-defined borders among certain genres, it has been decided to perform an additional experiment on the dataset.

Experiment 3 In this last experiment, the average inner homogeneity of all songs in the test set has been evaluated, by analyzing how many percent of each song are in average

Table 6: Confusion matrix for *Experiment 2*

	Alt./Punk	Classical	Dance	Easy List.	Hip-Hop	Jazz/Blues	R&B	Rock/Pop
Alt./Punk	**25**	1	0	3	0	2	1	8
Classical	1	**35**	0	4	0	0	0	0
Dance	0	0	**30**	1	2	2	2	3
Easy List.	1	9	1	**19**	1	3	2	4
Hip-Hop	1	0	4	0	**31**	0	3	1
Jazz/Blues	2	2	2	8	3	**16**	4	3
R&B	2	0	1	3	5	3	**23**	3
Rock/Pop	17	0	0	6	1	3	4	**9**

covered by the pre-dominant genre, compared to the percentages covered by other genres detected inside the same song. For example, if 70% of a song are covered by genre "A", 20% by genre "B" and 10% by genre "C", the homogeneity profile of the song (considering a maximum number of 4 genres) would be $[70, 20, 10, 0]$.

This investigation is performed individually for each genre as well as for the whole testing corpus. It is important to note that the reference genre of each song is *ignored*, as the predominant genre is not automatically the reference genre (see experiment 2).

The results listed in Table 7 and Figure 42 show that in average the pre-dominant genre of a song typically clearly covers the largest proportion of the song. As this is not only true for "strong" genres like *Classical*, but also for "problematic" genres like *Rock & Pop*, the results indicate that mis-classifications, at least when considering the used dataset, are not primarily caused by a too high variety of different styles within songs, but by poorly defined borders between genres.

Table 7: Homogeneity profiles of songs (%) | $mean(std)$

	main genre	sub-genre 1	sub-genre 2	other-genres
Alt./Punk	*74.4* (4.2)	19.2 (3.7)	4.8 (2.3)	1.6
Classical	*85.7* (4.2)	12.0 (3.4)	1.8 (1.2)	0.5
Dance	*78.2* (4.4)	15.5 (3.5)	5.2 (2.8)	1.1
Easy List.	*70.1* (4.7)	20.5 (3.7)	7.6 (3.0)	1.8
Hip-Hop	*81.8* (4.3)	14.4 (3.8)	3.0 (2.1)	0.8
Jazz/Blues	*70.9* (4.2)	21.0 (3.5)	6.1 (2.6)	2.0
R&B	*76.8* (4.3)	18.0 (3.8)	3.8 (2.1)	1.4
Rock/Pop	*73.8* (4.1)	20.6 (3.6)	4.6 (2.4)	1.0
Overall	**76.2** (4.3)	**17.6** (3.7)	**4.6** (2.4)	**1.6**

These observations may also explain the relatively low influence on the genre classification performance, when integrating knowledge about the musical structure to the classification process. The chance of mainly including "representative" segments of a song is very high, if $\approx 76\%$ of a song represent the same genre characteristics and a 30 sec-

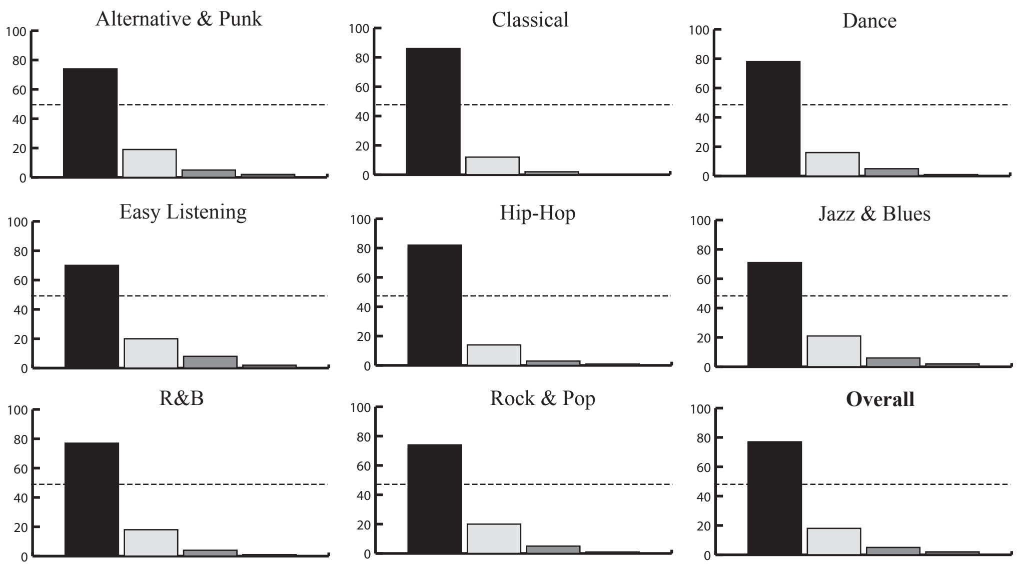


Figure 42: Homogeneity profiles of all genres. The bars indicate the average coverage of each song by different genres (bar $1 \rightarrow$ predominant genre, bars $1-3 \rightarrow$ sub-genres).

ond part (see experiment 1) is extracted as template of the song. This is in particular true, if such a part is extracted 30 seconds after the beginning of the song, as potentially unrepresentative sections like the *intro* and *outro* are automatically excluded.

7.5 Discussion

The presented results strongly support the notion that genre labels are problematic categories when trying to organize music collections - especially if only one genre label per song is allowed.

The conducted experiments indicate that incorporating knowledge about the musical structure of a song is a good and important strategy, when only short excerpts of songs need to be classified into individual genres. For systems allowing multiple genre labels per song, the classification of individual parts can provide important information, as any classification system can directly be used to find all genres related to a song. Besides, the individual homogeneity profiles of each song song could be useful indicators when trying to identify *non representative* songs in large databases and may help to better understand certain genre ambiguity problems when analyzing classification results.

References

[All83] J. Allen, “Maintaining knowledge about temporal intervals,” *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[ANSS05] S. Abdallah, K. Noland, M. Sandler, and M. S, “Theory and evaluation of a bayesian music structure extractor,” in *Proc. of the 6th International Conference on Music Information Retrieval (ISMIR)*, 2005, pp. 420–425.

[AP02] J. Aucouturier and F. Pachet, “Music similarity measures: What’s the use,” in *Proceedings of the 3rd International Conference of Music Information Retrieval (ISMIR)*, 2002, pp. 157–163.

[AS01] J. Aucouturier and M. Sandler, “Segmentation of musical signals using hidden markov models,” *Preprints-Audio Engineering Society*, 2001.

[BCL10] L. Barrington, A. Chan, and G. Lanckriet, “Modeling music as a dynamic texture,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 602–612, 2010.

[BGP06] G. Boutard, S. Goldszmidt, and G. Peeters, “Browsing inside a music track, the experimentation case study,” in *Proc. of the 1st Workshop on Learning Semantics of Audio Signals*, 2006, pp. 87–94.

[BP92] J. Brown and M. Puckette, “An efficient algorithm for the calculation of a constant q transform,” *Acoustical Society of America*, vol. 92, pp. 2698–2698, 1992.

[BR87] R. Bosanquet and R. Rasch, *An elementary treatise on musical intervals and temperament*. Diapason Press, 1987, vol. 4.

[Bro90] J. Brown, *Calculation of a constant Q spectral transform*. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of Technology, 1990.

[BW01] M. Bartsch and G. Wakefield, “To catch a chorus: Using chroma-based representations for audio thumbnailing,” in *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*. IEEE, 2001, pp. 15–18.

[BW05] ——, “Audio thumbnailing of popular music using chroma-based representations,” *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 96–104, 2005.

[CVG$^+$08] M. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, “Content-based music information retrieval: current directions and future challenges,” *Proc. of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[Dav66] C. Davie, *Musical structure and design*. Dover Publications Inc, 1966.

[Eck05] D. Eck, "Finding meter in music using an autocorrelation phase matrix and shannon entropy," in *Proc. of the 6th International Conference on Music Information Retrieval (ISMIR*, 2005, pp. 504–509.

[Ell07] D. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.

[Ero01] A. Eronen, "Comparison of features for musical instrument recognition," in *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*. IEEE, 2001, pp. 19–22.

[Ero07] ——, "Chorus detection with combined use of mfcc and chroma features and image processing filters," in *Proc. of the 10th International Conference on Digital Audio Effects (DAFx-07)*, 2007, pp. 229–236.

[Fas82] H. Fastl, "Fluctuation strength and temporal masking patterns of amplitude-modulated broadband noise," *Hearing Research*, vol. 8, no. 1, pp. 59–69, 1982.

[Foo00] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *Proc. of IEEE International Conference on Multimedia and Expo*, vol. 1, 2000, pp. 452–455.

[FZ07] H. Fastl and E. Zwicker, *Psychoacoustics: facts and models*. Springer-Verlag New York Inc, 2007.

[GC00] J. Gao and H. Cai, "On the structures and quantification of recurrence plots," *Physics Letters A*, vol. 270, no. 1-2, pp. 75–87, 2000.

[Got06] M. Goto, "A chorus section detection method for musical audio signals and its application to a music listening station," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1783–1794, 2006.

[HFH$^+$09] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[KS10] F. Kaiser and T. Sikora, "Music structure discovery in popular music using non-negative matrix factorization," in *Proc. of the11th International Conference of Music Information Retrieval (ISMIR)*, 2010, pp. 429–434.

[Lew95] J. Lewis, "Fast normalized cross-correlation," in *Vision Interface*, vol. 10, 1995, pp. 120–123.

[LF08] H. Lukashevich and I. Fraunhofer, "Towards quantitative measures of evaluating song segmentation," in *Proc. of the 9th International Conference of Music Information Retrieval (ISMIR)*, 2008, pp. 375–380.

[LR05] T. Lidy and A. Rauber, “Evaluation of feature extractors and psycho-acoustic transformations for music genre classification,” in *Proc. of the 6th International Conference on Music Information Retrieval (ISMIR)*, 2005, pp. 34–41.

[LS01] B. Logan and A. Salomon, “A music similarity function based on signal analysis,” in *IEEE International Conference on Multimedia and Expo (ICME)*, 2001.

[LS08] M. Levy and M. Sandler, “Structural segmentation of musical audio by constrained clustering,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 318–326, 2008.

[LWOO08] R. Loughran, J. Walker, M. ONeill, and M. OFarrell, “The use of mel-frequency cepstral coefficients in musical instrument identification,” in *Proc. of the International Computer Music Conference (ICMC)*, 2008.

[ME05] M. Mandel and D. Ellis, “Song-level features and support vector machines for music classification,” in *Proc. of the 6th International Conference on Music Information Retrieval (ISMIR)*, 2005.

[MF10] F. Mörchen and D. Fradkin, “Robust mining of time intervals with semi-interval partial order patterns,” in *Proc. of the SIAM Conference on Data Mining (SDM)*, 2010, pp. 315–326.

[MLC06] M. S. Mark Levy and M. Casey, “Extraction of high-level musical structure from audio data and its application to thumbnail generation,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 5, 2006, pp. V–13–V–16.

[MM06] M. McKinney and D. Moelants, “Ambiguity in tempo perception: What draws listeners to different metrical levels?” *Music Perception*, vol. 24, no. 2, pp. 155–166, 2006.

[Moe10] F. Moerchen, “Temporal pattern mining in symbolic time point and time interval data,” in *Proc. of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010.

[Ots75] N. Otsu, “A threshold selection method from gray-level histograms,” *Automatica*, vol. 11, pp. 285–296, 1975.

[Pam06] E. Pampalk, “Computational models of music similarity and their application in music information retrieval,” Ph.D. dissertation, Vienna University of Technology, 2006.

[PC00] F. Pachet and D. Cazaly, “A taxonomy of musical genres,” in *Proc. on Content-Based Multimedia Information Access (RIAO)*, 2000, pp. 1238–1245.

[Pee04] G. Peeters, “A large set of audio features for sound description (similarity and classification) in the cuidado project,” IRCAM, Analysis, Synthesis Team, 2004.

[Pee07] ——, “Sequence representation of music structure using higher-order similarity matrix and maximum-likelihood approach,” in *Proc. of the 8th International Conference on Music Information Retrieval (ISMIR)*, 2007.

[Pei07] E. Peiszer, “Automatic audio segmentation: Segment boundary and structure detection in popular music,” Masters thesis, Vienna University of Technology, 2007.

[PK06] J. Paulus and A. Klapuri, “Music structure analysis by finding repeated parts,” in *Proc. of the 1st ACM workshop on Audio and music computing multimedia*. ACM, 2006, p. 68.

[PK08] ——, “Music structure analysis using a probabilistic fitness measure and an integrated musicological model,” in *Proc. of the 9th International Conference of Music Information Retrieval (ISMIR)*, 2008, pp. 369–374.

[PK09] ——, “Music structure analysis using a probabilistic fitness measure and a greedy search algorithm,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 6, pp. 1159–1170, 2009.

[PLBR02] G. Peeters, A. La Burthe, and X. Rodet, “Toward automatic music audio summary generation from signal analysis,” in *Proc. of the 3rd International Conference on Music Information Retrieval (ISMIR)*, 2002, pp. 94–100.

[PLR08] E. Peiszer, T. Lidy, and A. Rauber, “Automatic audio segmentation: Segment boundary and structure detection in popular music,” in *Proc. of the 2nd International Workshop on Learning the Semantics of Audio Signals (LSAS)*, 2008.

[PMK10] J. Paulus, M. Müller, and A. Klapuri, “Audio-based music structure analysis,” in *Proc. of the 11th International Conference on Music Information Retrieval (ISMIR)*, 2010, pp. 625–636.

[PRM02] E. Pampalk, A. Rauber, and D. Merkl, “Content-based organization and visualization of music archives,” in *Proceedings of the 10th ACM international conference on Multimedia*. ACM, 2002, pp. 570–579.

[SK10] C. Schörkhuber and A. Klapuri, “Constant-q transform toolbox for music processing,” in *Proc. of the 7th Sound and Music Conference (SMC)*, 2010.

[Sla98] M. Slaney, “Auditory toolbox, version 2, technical report no: 1998-010,” 1998.

[Smi10] J. Smith, “A comparison and evaluation of approaches to the automatic formal analysis of musical audio,” 2010.

[SS09] K. Seyerlehner and M. Schedl, “Block-level audio feature for music genre classification,” *online Proc. of the 5th Annual Music Information Retrieval Evaluation eXchange (MIREX-09)*, 2009.

[SV40] S. S. Stevens and J. Volkman, “The relation of pitch to frequency: A revised scale,” *Am. J. Psychol.*, 1940.

[SWP10] K. Seyerlehner, G. Widmer, and T. Pohle, “Fusing block-level features for music similarity estimation,” in *Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10)*, 2010, pp. 225–232.

[SZ05] N. Scaringella and G. Zoia, “On the modeling of time information for automatic genre recognition systems in audio signals,” in *Proc. of the 6th International Conference on Music Information Retrieval (ISMIR)*, 2005, pp. 666–671.

[SZM06] N. Scaringella, G. Zoia, and D. Mlynek, “Automatic genre classification of music content: a survey,” *IEEE Signal Processing Magazine*, vol. 23, no. 2, pp. 133–141, 2006.

[TC02] G. Tzanetakis and P. Cook, “Musical genre classification of audio signals,” *IEEE transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[TL08] A. P. Thomas Lidy, Andreas Rauber, “Audio music classification using a combination of spectral, timbral, rhythmic, temporal and symbolic features,” *online Proc. of the 4th Annual Music Information Retrieval Evaluation eXchange (MIREX-08)*, 2008.

[WCB10] A. Wankhammer, I. Clarkson, and A. Bradley, “Music structure discovery based on normalized cross correlation,” in *Proc. of the 13th International Conference on Digital Audio Effects (DAFx-10)*, 2010, pp. 488–493.

[Wes05] K. West, “Finding an optimal segmentation for audio genre classification,” in *Crawford and Sandler*, 2005, pp. 680–685.

[WHH$^+$08] L. Wang, S. Huang, S. Hu, J. Liang, and B. Xu, “An effective and efficient method for query by humming system based on multi-similarity measurement fusion,” in *Proc. of the International Conference on Audio, Language and Image Processing (ICALIP)*, 2008, pp. 471–475.

[WS11] A. Wankhammer and A. Sontacchi, “Segment related classification for automatic genre detection,” in *37. Deutsche Jahrestagung für Akustik: DAGA 2011.* DAGA, 2011.

[WSS09] A. Wankhammer, P. Sciri, and A. Sontacchi, “Chroma and mfcc based pattern recognition in audio files utilizing hidden markov models and dynamic prgramming,” in *Proc. of the 12th International Conference on Digital Audio Effects (DAFx-09)*, 2009, pp. 401–407.

[ZS07] T. Zhang and R. Samadani, “Automatic generation of music thumbnails,” in *IEEE International Conference onMultimedia and Expo*, 2007, pp. 228–231.