# Music Structure Analysis

An Exploration of and Improvements on the
Distance-based Segmentation and Annotation Approach

*Author:*
Jesper S. Kuiper

*Supervisor:*
Dr. M.P. (Marijn) Schraagen

*Second Supervisor:*
Prof. dr. C.J. (Kees) van Deemter

A 15EC thesis for the
Bachelor of Science in Artificial Intelligence


Faculty of Humanities
Utrecht University
The Netherlands
June 26, 2020

# Music Structure Analysis

An Exploration of and Improvements on the Distance-based Segmentation and Annotation Approach

**Jesper Kuiper**

## Abstract

This thesis explores the commonly used distance-based segmentation and annotation (DSA) approach to music structure analysis. In an attempt to improve upon already existing DSA algorithms, this thesis proposes methods of combining multiple musical representations into a single fusion of features, and applies them to different approaches of distance-based segmentation and annotation. This thesis found that a fusion of a timbre feature together with a chroma feature has the most chance of increasing the quality of a DSA approach, even though the experiments in this thesis yield a statistically insignificant increase in performance. Additionally, this thesis presents numerous ways to increase performance of DSA feature fusion more. Finally, this thesis puts the DSA approach in context to other approaches of music structure analysis, such as the segmentation by annotation approach, that makes use of Deep Neural Networks. The big advantage of DSA is that it isn't prone to over-fitting and might therefore be easier to use on multiple genres or recording types. A final discussion of the explainability of different MSA approaches is held.

**Keywords:** Music Information Retrieval, Music Structure Analysis, Distance-based Segmentation and Annotation

*More things than are dreamed about, unseen and unexplained*
*We suspend our disbelief, and we are entertained*
*Mystic rhythms – capture my thoughts*
*Carry them away...*

NEIL ELLWOOD PEART
Rush - Mystic Rhythms

# Contents


**1 Introduction** **5**
- 1.1 Introduction . . . 5
- 1.2 Music Structure Analysis . . . 5
  - 1.2.1 Relevance to Artificial Intelligence . . . 5
  - 1.2.2 Focus of this Thesis . . . 6
- 1.3 Aim and Structure of this Thesis . . . 7
  - 1.3.1 Goals for this Thesis . . . 7
  - 1.3.2 Structure of this thesis . . . 8

**2 Related Work** **9**
- 2.1 The DSA Pipeline . . . 9
  - 2.1.1 Feature Vector selection . . . 9
  - 2.1.2 The Self-Similarity Matrix . . . 12
  - 2.1.3 DSA Approaches for Music Structure Analysis . . . 14
- 2.2 Evaluating MSA Algorithms . . . 19
  - 2.2.1 Ground Truth: Different Datasets for MSA . . . 19
  - 2.2.2 Evaluation Approaches . . . 20
  - 2.2.3 Frameworks for Approach Comparison . . . 21
- 2.3 Feature Fusion in DSA . . . 21
- 2.4 Approaches beyond DSA . . . 22

**3 Methodology** **23**
- 3.1 Feature Fusion . . . 23
  - 3.1.1 Matrix Stack . . . 23
  - 3.1.2 Novelty-curve Summation . . . 23
  - 3.1.3 Similarity Network Fusion . . . 24
- 3.2 Means of Implementation . . . 25
- 3.3 Means of Evaluation . . . 26

**4 Evaluation** **29**
- 4.1 Structural Segmentation . . . 29
  - 4.1.1 Results . . . 29
  - 4.1.2 Conclusion . . . 30
- 4.2 Segment Labelling . . . 31
  - 4.2.1 Results . . . 31
  - 4.2.2 Conclusion . . . 31
- 4.3 Discussion . . . 32
  - 4.3.1 Observations from the Experiments . . . 32
  - 4.3.2 Assessment of the Quality of the Dataset . . . 33
  - 4.3.3 Possible Improvements . . . 34

**5 DSA In Context** **38**
5.1 Comparing Performance . . . . . 38
5.2 Comparison of MSA approaches . . . . . 39
5.2.1 DSA in comparison with SbA . . . . . 39
5.2.2 DSA Compared to State of the Art . . . . . 40
5.2.3 The Debate of Explainability . . . . . 40
5.3 The Future of MSA . . . . . 41

**Appendices** **47**

**A Links** **48**
A.1 MSAF . . . . . 48
A.2 Salami . . . . . 48

**B Track Identifiers** **49**

**C SNF Implementation** **50**

# Chapter 1

# Introduction

## 1.1 Introduction

Humans subconsciously divide pieces of music up into different and sometimes recurring sections. However, it's hard to define the way these sections differ from each other. This is due to the fact that human judgement of pieces of music is inherently subjective.

One cause of subjectivity in music judgement and understanding lies in the hierarchical structure of music [36]. Individual notes together form chords and groups of chords represent chord progressions or phrases. These are then grouped into higher structures, and so on. Different people could form their judgement based on different levels in the hierarchy, or they could have a different hierarchical representation of the same piece.

Another reason this subjectivity exists is due to the fact that, in their judgement, humans always include what a piece of music does to *them* and what *they* do because of the music [8]. For instance, if someone says a specific part of a track is very happy, that's because that part made them happy, or because they imagined someone could become happy listening to this part of the track. This inclusion of emotion in human music understanding makes it even more likely that different persons construct their hierarchical structure in a different way relative to one another.

Humans generally have no problem to identify whether they are listening to a verse, chorus or a solo in a track. However, due to the inherent subjectivity of such structure, and therefore the discrepancy between different persons, it's hard to precisely determine for a track its different properties that define its musical structure.

## 1.2 Music Structure Analysis

*Music structure analysis*, or MSA, is the practice of computationally extracting a description of the structure of a piece of music. An example of *musical structure* on multiple hierarchical levels can be seen in Figure 1.1. Music structure analysis is an important branch of music information retrieval and is of importance for humans and computers alike.

### 1.2.1 Relevance to Artificial Intelligence

MSA not only lets us humans understand our own music better, it can also help computers gain insight for instance in the similarity or dissimilarity between different tracks, and how certain structures are used for certain genres and why others are not. Furthermore, MSA is part of the foundation of automatic music generation [26] and audio thumbnailing [30]. Knowledge of music structure can not only be used for the automatic generation of a

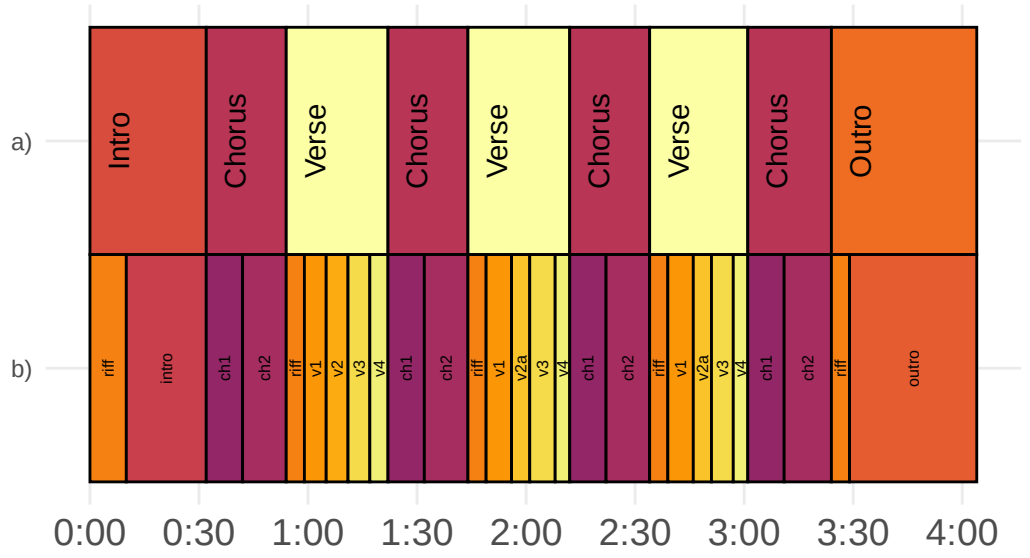



Figure 1.1: Example of the music structure of 'Mrs. Robinson' by Simon & Garfunkel. a) represents the topmost level of musical structure, whereas b) represents the musical structure on a lower hierarchical level.

track, but also for validation of already generated pieces to make sure that its structure somewhat complies to the musical structure humans are used to.

Being able to extract the musical structure of a piece of music is the first step in making intelligent agents understand the very concept of 'music'. The end goal would be an agent's true *apperception* [15] of music, i.e. that an agent would be able to fathom the internal patterns that underlie a piece of music to an arbitrary level of complexity. The structural analysis of a piece of music can then be seen as one of the first levels of complexity and is thus a paramount step in achieving *machine music apperception*.

### 1.2.2 Focus of this Thesis

This thesis focuses on extracting the topmost level of the musical structure, which are the *musical sections* as *intro*, *verse*, and *chorus*. Refer to Figure 1.1a for an example of such a musical structure on 'Mrs. Robinson' by Simon & Garfunkel. Within this branch of MSA, the *segmentation* of a track is defined as the determination of where transitions from one musical section, or *segment*, are made to another. The *annotation* is then defined as the determination of what *functional role* a certain part of a track has, i.e. whether a specific part is, or belongs to, a verse or a chorus, or to something else.

This thesis focuses mainly on Western popular music. This means that this thesis assumes functional roles and tonic properties that are common in this type of music. In the following sections, two global approaches for this top-level MSA are described.

#### Segmentation by Annotation (SbA)

In the *Segmentation by Annotation* or SbA-approach, the music track that is analyzed is first split up into multiple smaller parts of equal size. The size of these parts can be either of a fixed length (e.g. 10 ms), or based on the tempo of the track (e.g. each beat). Each part is then individually annotated, such that they all have a functional role. Consecutive parts that have the same annotated functional role are then grouped as a single functional group, or segment. This approach largely revolves around finding a good algorithm or model for the automatic annotation of the smaller parts.

Up until this point, SbA isn't yet used for this branch of MSA. One of the main applications of SbA at this moment is to classify pieces of audio data as speech, music or noise [20].

#### Distance-based Segmentation and Annotation (DSA)

*Distance-based Segmentation and Annotation* or DSA uses a measure of distance or dissimilarity within the track to extract the musical structure. The track is first converted into a vector representation, where each vector represents tonic features for each frame of the track. One frame is commonly defined as a single beat, or a certain time period. This representation can be used to calculate inter-vector distance. The DSA approach can then be subdivided into two subtasks.

First, this approach uses this distance function to compute the most probable positions of the segment boundaries, or the locations where one segment transitions to another. Intuitively, if two consecutive pieces of music have a high dissimilarity, they will probably belong to different segments and will likely have a segment boundary in between them.

When the segment boundary locations are extracted, the segments are clustered into groups that have the same functional role, again using a distance measure. Intuitively, if two segments have a high similarity to one another, they will probably have the same functional role.

This approach, first introduced by Foote [16], is very common for music structure analysis, and indeed a lot of proposed methods for MSA have used some kind of DSA. It is however very hard to compare the performance of all these proposed methods, as they all use different parameters, datasets and evaluation methods. There has been an attempt to evaluate and compare multiple different implementations using a single framework [52], however the comparison proved inconclusive.

#### Feature Extraction

Something these approaches have in common, is the extraction of information from the audio source file, that the algorithm uses for its analysis. A raw audio wave file is very hard to use directly for processing, which is why the data from the file is first converted into a more information-rich set of $n$-dimensional feature vectors. This conversion step determines what kind of information the next processing steps will have to work with, and it is thus important that the resulting vectors adequately reflect change in the music.

There has been some evidence [55] that a single method of feature extraction might not be sufficient to encapsulate all information from an audio file. This might have negative implications for use in music structure analysis, however the use of multiple feature extraction methods, especially in the context of DSA hasn't been explored a lot yet.

## 1.3 Aim and Structure of this Thesis

This thesis focuses on the DSA approach to music structure analysis. It is complementary to the thesis of Van Boven [4], that focuses on the SbA approac to MSA. In a similar fashion to Van Boven [4], this this thesis aims to provide an in-depth overview and possible improvement of the current state of the art of music structure analysis.

### 1.3.1 Goals for this Thesis

This thesis aims to evaluate the performance of Distance-based Segmentation and Annotation algorithms for music structure analysis. For this, it is important to evaluate possible additions to this approach, so as to improve its performance. Furthermore, the DSA approach needs to be compared to other approaches of MSA, as to evaluate its relevance to the current state of the art of music structure analysis.

In particular, this thesis evaluates the performance gain that can be achieved by fusing multiple kinds of vector representations into one processing pipeline, within the context of DSA. This is called *feature fusion*. Finally, this thesis aims to put the performance of the DSA-approach as a whole in context with the performance of the Segmentation by Annotation approach.

This can be summarized into the following two research questions that this thesis attempts to answer:

1. Can the performance of state-of-the-art DSA-based approaches to MSA be improved with feature fusion?
   (a) What are the different approaches to DSA, and how do they compare?
   (b) How can feature fusion be added to such approaches?
2. How do DSA-based approaches compare to SbA-based or other MSA approaches?
   (a) How does the performance of DSA-based approaches compare to the performance of other approaches?
   (b) What are other properties of DSA-based approaches that might be advantageous or disadvantageous in comparison to other approaches?

### 1.3.2 Structure of this thesis

In Chapter 2, the DSA pipeline is discussed, including common DSA methods and their properties. Additionally, the state of feature fusion in DSA and MSA, and the way MSA algorithms are commonly evaluated are discussed. In Chapter 3, methods of feature fusion are presented, and it will be discussed how this thesis evaluates and compares these methods of feature fusion to non-fused DSA methods. In Chapter 4, the results from these experiments are listed and discussed. Using these results, an attempt will be made to answer research question 1. in Chapter 5, the DSA approach as a whole will be put into context to other approaches of MSA, including the SbA approach that has been implemented by Van Boven [4]. In this chapter, research question 2 will be answered and an outlook for the future of MSA will be discussed.

# Chapter 2

# Related Work in Distance-based Segmentation and Annotation

This chapter attempts to answer research question 1a: What are the different approaches to DSA, and how do they compare? To this end the general pipeline of DSA will be discussed. In Section 2.1 all the approaches and implementations that are commonly used in DSA are discussed for each step in the DSA pipeline. Section 2.2 will discuss common approaches used to evaluate and compare MSA algorithms. Section 2.3 discusses the current state of feature fusion in DSA-based MSA. Finally, Section 2.4 discusses MSA beyond the DSA approach.

## 2.1 The DSA Pipeline

Below, each step that is part of the pipeline of distance-based segmentation and annotation is individually discussed, including their implementations.

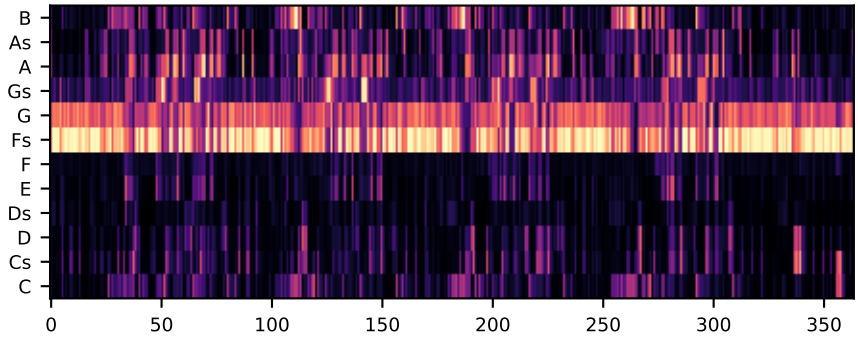
### 2.1.1 Feature Vector selection

The first step is determining what information or features will be used in the analysis. A signal as complex as an audio signal holds a lot of information, but extracting this information can be hard.

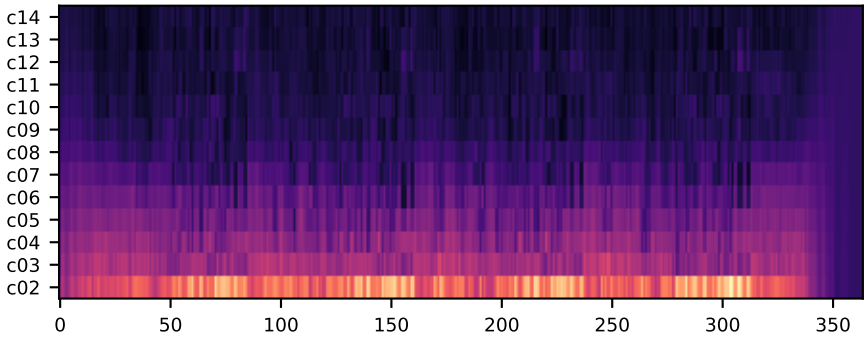
Before actual features are extracted, it's common to down-sample the input wave file to a monaural 11025Hz sampling rate. In the end, the goal of the feature extraction is to extract an $n \times m$ feature matrix

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \dots & x_{1,m} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \dots & x_{n,m} \end{bmatrix} \tag{2.1}$$

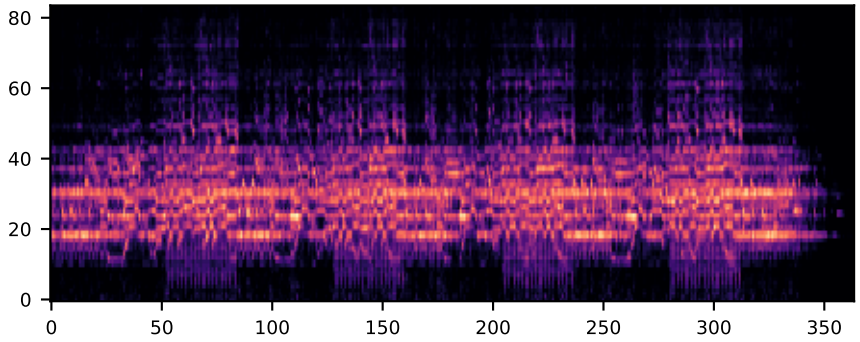
such that the row vector $\mathbf{x}_i$ of length $m$ represents the $i$'th feature vector, for $i = 1, ..., n$. All different feature extraction methods define a way to convert a multiple of audio samples, or *frames*, into one such feature vector. The dimensionality of the vector ($m$) depends on the feature extraction method. Below, some of the most common feature types that are used for music structure analysis are described. These feature types can be subdivided into three categories: *chroma*, *timbre* and *rhythm*.



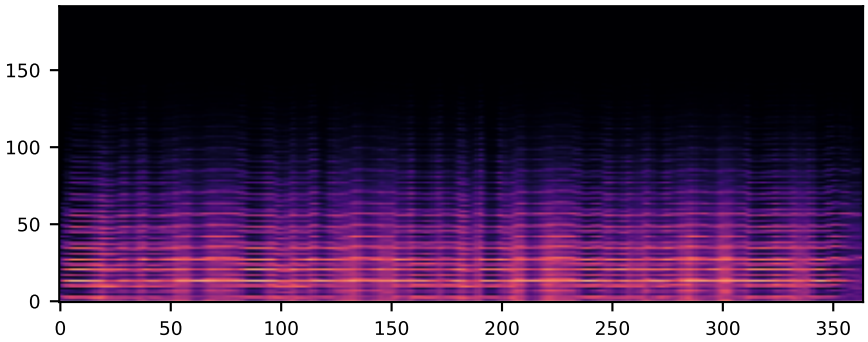

(a) PCP




(b) MFCC (c01 omitted)




(c) CQT




(d) Tempogram

Figure 2.1: Feature matrices (transposed) from 'Mrs Robinson' by Simon & Garfunkel.

### Chroma

Chroma ("color") vectors are vectors that represent information about the pitch, or frequency, of a track over time. The first step that all chroma features have in common, is the conversion from the raw audio signal to the intensity of all audible frequencies used over time. This matrix is called a *spectrogram*, and can be obtained by applying the discrete fourier transform [73] on a *window*, or slice of the audio data. By repeating this whilst 'sliding' the window through the audio data, a representation of frequency intensity over time is made. This is called the *Short Time Fourier Transform*, or *STFT* [63].

The *Pitch Class Profile* [19] or *PCP* converts a STFT chromagram into a measure of intensity of the 12 pitch classes on the equal-tempered scale ($C$, $C\sharp$, $D$, etc.). The deviation from the ground frequency (440Hz) is estimated for all peaks in intensity in the chromagram, and these are mapped to the right pitch class. The total amount of bins is equal to a multiple of 12, so that each bin represents a fraction of a semitone. For instance, if the amount of bins is equal to 36, a value can be mapped on one third of a semitone. Refer to Figure 2.1a for the PCP of 'Mrs Robinson'. We can see that in the track the notes $G$ and $F\sharp$ are used most often. Additionally, from the plot a 'cone' structure emerges that is repeated four times. This probalby indicates a repeated section in the track. As can be seen when comparing this figure to Figure 1.1, these cones roughly correspond to the chori in the track.

### Timbre

Whenever two instruments play the same note, they play at the same frequency. This means that a chroma feature as described above wouldn't be able to separate these two instruments. Some information is therefore lost during chroma extraction, as humans generally don't have many problems identifying what instruments play a certain note. Timbre, sometimes described as "the way it sounds", tries to capture the information that is lost in the chromatic representation.

One feature type that describes timbre is called *Mel-Frequency Cepstral Coefficients* or *MFCC*. MFCCs make use of the mel-scale [67], which is a non-linear subjective scale of pitches that multiple listeners judged to be of equal distance to each other. This use of mel-scale makes MFCCs more in accordance with our own perception, which is why it is a common feature type for both speech recognition and music structure analysis [38]. Refer to Figure 2.1b for the MFCCs of 'Mrs Robinson'. Not that the first ceptsrum coefficient, 'c01', is ommited from this plot.[1] We can see that 'c02' has the highest intensity, and the most changes in intensity as well. This 'expressiveness' of the coefficients becomes less and less as we go to higher coefficients, indicating that the lower coefficients capture more information about the track than higher ones.

Another feature type that describes timbre is the *Constant Q-Transform* [5, 6, 3] or *CQT*. CQT is based on the assumption that subjective pitch scales logarithmically with frequency [14], and it was used in the context of music structure analysis [39, 53].

Refer to Figure 2.1c for the CQT features of 'Mrs Robinson'. In this plot, we can clearly see some patterns that recur. For instance, the snippet from 25s to 50s recurs at 100s, around 170s and at 250s. Similarly, these snippets all are succeeded by parts that are more bright (i.e. have higher values) accross the scale. In this case, the brighter parts represent the verses, indicating that more tonic variety is present in these sections as compared to the preceding choruses. These choruses, however, do employ highly characteristic melodies, explaining why they are so clearly spotted in the CQT matrix.

[1] This is because the first cepstrum coefficient represents track loundess, which is represented in negative decibels. As all the other coefficients lie between 0 and 1, adding 'c01' to the plot would mean that changes in the other coefficients would become hardly noticeable in the plot.

#### Rhythm

All the feature types up until this point represent musical harmony. However, rhythmic information of a track could also be of use for music structure analysis. All verses in a track, for example, commonly have a very similar rhythmic pattern, whereas verses tend to diverge a bit more in their rhythm.

A *tempogram* provides information about the variation in tempo and general 'pulse' in the track. For the construction, a sense of audio *novelty* is used, as, intuitively, a beat occurs at places where something new happens and thus where audio novelty peaks. This is commonly called an 'onset'.

Refer to Figure 2.1d for the tempogram (extracted from the local autocorrelation of the onset strength envelope [24]) of 'Mrs Robinson'. We can see that the bright lines are slightly ripply, indicating that the tempo in this track doesn't change all that much. We can see that around the verses (that onset at 50s, 125s, 200s and 280s), the tempogram is slightly more diffused and unclear as compared to the choruses that preceed them. This indicates that the instrumentation in the choruses is a lot clearer an more percussive, resulting in a clearer tempogram picture. This is in contrast to the verses as they are more 'loose', resulting in a less clear tempogram image.

Tempograms and tempogram-based features are less common in music structure analysis [58], however there have been more recent attempts at producing more tempogram-based audio features [68, 29, 44].

#### Beat synchronization

The features as described above all use a *sliding window approach*, where, as discussed in Section 2.1.1, the window slides through the input data. The final length $n$ of the feature matrix is then defined in terms of the window size and the *hop size*, or the amount of samples that the window 'hops' forward each time.

In music structure analysis, it is common to use beat-synchronized feature vectors [44, 53], which means that $\mathbf{x}_i$ represents the feature vector of the $i$'th beat. To this end the beat of the music piece is tracked with a dynamic programming algorithm [12]. By taking the mean of the feature vectors that span a beat, the feature vector of that beat is extracted.

### 2.1.2 The Self-Similarity Matrix

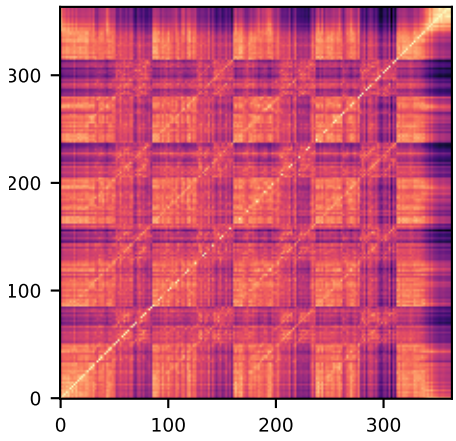
Given an $n \times m$ feature matrix $\mathbf{X}$, we can now construct a *self-similarity matrix*, or *SSM* from it. The notion of an SSM was first introduced by Foote [17] and is the foundation for most boundary and labelling algorithms.

Given a self-similarity matrix $\mathbf{S}$ and two frames $i, j \in [1 : n]$, the value of $S_{i,j} \in [0, 1]$ defines the similarity between the column vectors $\mathbf{x}_i$ and $\mathbf{x}_j$. We define $S_{i,j} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are exactly the same, or if $i = j$. $\mathbf{S}$ will thus be a diagonally symmetric $n \times n$ matrix.

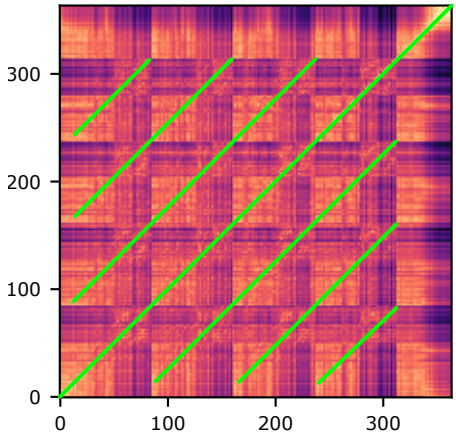
There are a couple variations in the distance metric that is used to calculate all values in the SSM. The most common metric is the regular euclidean distance, however cosine distance and the correlation coefficients are also used.

Refer to Figure 2.2 for an example of a self-similarity matrix of 'Mrs. Robinson' from Simon & Garfunkel, using euclidean distance and using CQT as features. The bright line that goes from the bottom-left to the top-right, is the line where $i = j$ and thus $\mathbf{S}_{i,j} = 1$. Additionally, in parallel to the main diagonal, six other bright lines can be spotted. These lines indicate a repetition in the track, and the offset of the lines (in $x$ or $y$ direction, not horizontally) tells us the offset of the repetition with relation to the first occurrence.

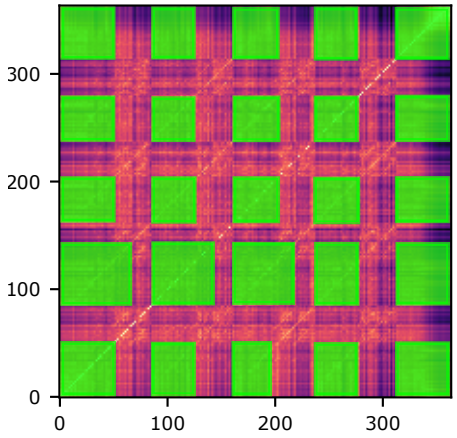
Another pattern that often merges from self-similarity matrices, are the occurrence of blocks that are consistently brighter than their surroundings. An occurrence of a block on the main diagonal tells us that the track has consistent features during this period. This is therefore intuitively a good indication of a musical section that spans such a block.




(a) Regular SSM




(b) SSM with enhanced lines

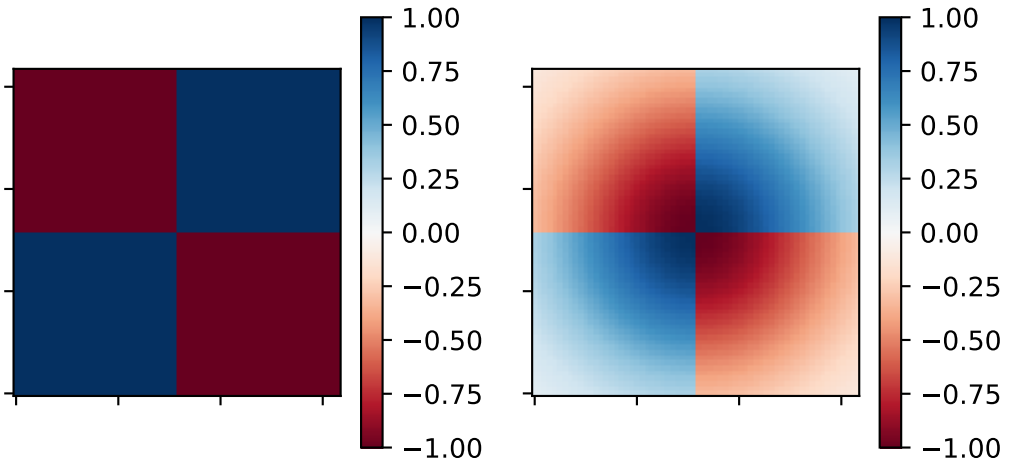



(c) SSM with enhanced blocks

Figure 2.2: Self-similarity matrix extracted from 'Mrs. Robinson' by Simon & Garfunkel, using CQT as features.

This means that, when a transition is made from one block to another block on the main diagonal, it has good odds of being a transition from one musical section to another.

Using the patterns as described above, we can categorize MSA approaches by means of what kind of patterns they try to discover in the music. The common categorization [58, 48] used for MSA approaches is as follows. *Repetition-based* approaches try to find the repetition of similar patterns, represented by the lines in the SSM (Figure 2.2b). *Homogeneity-based* approaches try to find parts in the track that remain constant for a while, and thus are homogeneous. This is represented in the SSM by the bright block (Figure 2.2c). Lastly, *Novelty-based* approaches try to find parts where a pattern transitions into another contrasting pattern. In the SSM, this is represented as the locations along the main diagonal where one block ends, and a new one starts. Even though some approaches can be fully accounted for by one of these categories, a lot of approaches incorporate information from more than one category in their evaluation.




(a) Normal checkerboard kernel (b) Gaussian tapered checkerboard kernel

Figure 2.3: An example of a checkerboard kernel

### 2.1.3 DSA Approaches for Music Structure Analysis

As already discussed in Section 1.2.2, the extraction of music structure from a track can be subdivided into two subproblems: structural segmentation and segment clustering. Some approaches that have been proposed tackle both; however some approaches are specifically designed for one of the two subproblems. The most notable of these approaches are discussed below, and will be compared and categorized as discussed above.

#### Gaussian Checkerboard Kernel

One of the very first structural music segmentation algorithms was introduced by Foote [16] and is a novelty-based approach that tries to detect the locations in the SSM where one square transitions into another.

To detect such transitions, this approach makes use of a checkerboard kernel, that is convolved over the SSM's main diagonal. A checkerboard kernel is a square matrix that looks a bit like a checkerboard, for instance the most simple $2 \times 2$ checkerboard kernel looks like this:

$$\mathbf{K} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.2}$$

In music structure analysis however, we're more interested in changes over a longer period of time. This is why larger matrices such as a $64 \times 64$ checkerboard kernel are commonly used. Refer to Figure 2.3 for a visualization of such a larger checkerboard kernel.

Given an $n \times n$ self-similarity matrix $\mathbf{S}$ and an $\ell \times \ell$ checkerboard kernel $\mathbf{K}$, the novelty vector $\mathbf{c}$ is extracted, such that $c_i$ represents the novelty of beat $i$ with relation to what came before it. The novelty curve is calculated as follows:

$$c_i = \sum_{x=-\frac{\ell}{2}}^{\frac{\ell}{2}} \sum_{y=-\frac{\ell}{2}}^{\frac{\ell}{2}} S_{i+x,i+y} \cdot K_{x,y} \tag{2.3}$$

For $i \in [1 : n]$. The center point of $\mathbf{K}$ is set to $(0, 0)$ by convention, as this makes the math easier to comprehend. Furthermore $\mathbf{K}$ is assumed to have an even width and length. At every step $i$ in $\mathbf{c}$, $\mathbf{K}$ is correlated to the sub-matrix $\mathbf{S}$, from $\frac{1}{2}\ell - i$ to $\frac{1}{2}\ell + i$, both in $x$ and

$y$ direction. As can be seen in Figure 2.3a, $\mathbf{K}$ exactly represents the pattern of a transition from square to another. Therefore, if the value of $\mathbf{c}_i$ is high, the music transitions into a novel section. To find the segment boundaries, one only has to extract the locations of peaks from $n$. This is often done using an adaptive thresholding method. An example of a thresholding method that has been used in MSA is introduced by Jacobson [28].

To avoid edge cases, the checkerboard kernel is smoothed using a radially-symmetric Gaussian filter. Refer to Figure 2.3b for the visualization of such a kernel.

The gaussian checkerboard kernel method is one of the most common approaches to structural segmentation because of its simplicity. It is therefore commonly used as a critical step in more complicated approaches [57, 7, 68, 56, 32]

### Non-negative Matrix Factorzation

Non-negative Matrix Factorization or NMF [34] is a form of matrix factorization, like Principal Component Analysis or Vector Quantization. These methods are used to compress a dataset, in such a way that as much information as possible is retained. NMF however adds the constraint of non-negative values. Consider the non-negative matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$, for certain $n, m \in \mathbb{N}$, where each row vector of $\mathbf{A}$ represents an obervation. An observation is a vector of data that is indipendent of the rest of the vectors in the dataset. For instance, if the dataset is a feature matrix, each feature vector is an observation.

NMF now tries to describe $\mathbf{A}$ in terms of $\mathbf{F} \in \mathbb{R}^{n \times r}$ and $\mathbf{G} \in \mathbb{R}^{r \times m}$ for a certain rank $r \in [1 : \min(n, m)]$, such that:

$$\mathbf{A} \approx \mathbf{FG} \tag{2.4}$$

To achieve data compression, $r$ is commonly chosen such that $nr + rm < nm$. Now consider the matrices $\mathbf{A}^k$, for $k \in [1 : r]$ such that

$$\mathbf{A}^k = \mathbf{f}_k \mathbf{g}^k \tag{2.5}$$

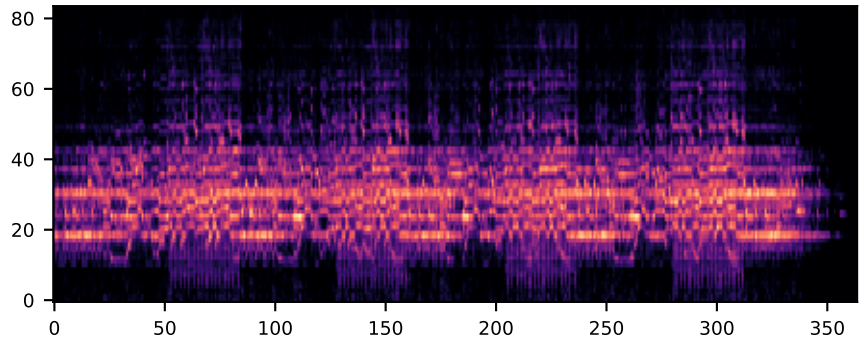
Where $\mathbf{f}_k$ denotes the $k$th row vector of $\mathbf{F}$, and $\mathbf{g}^k$ the $k$th column vector of $\mathbf{G}$. These matrices $\mathbf{A}^1$ to $\mathbf{A}^r$, referred to as *decomposition matrices*, provide a separation of the data of $\mathbf{A}$. In Kaiser and Sikora [32], it was shown that this separation relates to the structural segmentation of a piece of music, if a feature matrix matrix is decomposed with NMF. Refer to Figure 2.4 for an example of NMF decomposition on Simon & Garfunkel's 'Mrs. Robinson'. We can see that the decomposition matrices retain information of specific parts in the track. In this case, $\mathbf{A}^2$ represent all verses, $\mathbf{A}^3$ the choruses (and the intro) whereas $\mathbf{A}^3$ only represents the outro. This property makes NMF interesting for music structure analysis, and Kaiser and Sikora [32] proposed a segment clustering method that uses NMF.

In this segment clustering method, actually the self-similarity matrix of a feature matrix is decomposed using NMF. This results in square $n \times n$ decomposition matrices. Then, a new feature space $\mathbf{D}$ was considered that contains all the diagonals of the $r$ decomposition matrices.

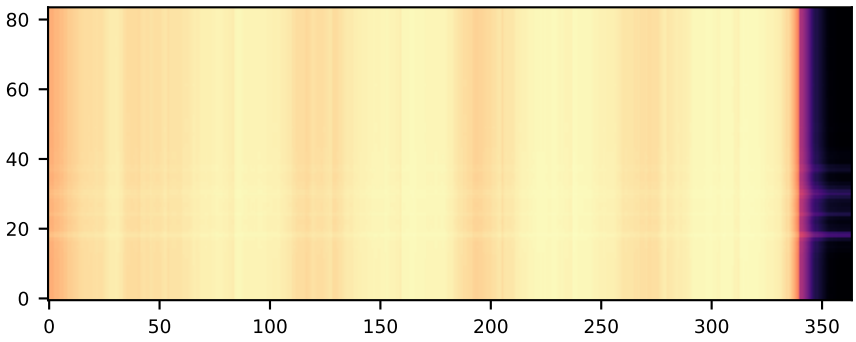
$$\mathbf{D} = \begin{bmatrix} A^1_{0,0} & A^1_{1,1} & \dots & A^1_{n,n} \\ \vdots & \ddots & \ddots & \vdots \\ A^r_{0,0} & A^r_{1,1} & \dots & A^r_{n,n} \end{bmatrix} \tag{2.6}$$

Then a classical hierarchical clustering was done on $\mathbf{D}$, where both the Bayesian Information Criterion [50] (BIC) and the Mahalanobis distance [41] were tried as distance measures. The Mahalanobis distance was shown to provide the best results when paired with the aforementioned Gaussian checkerboard kernel for the structural segmentation.

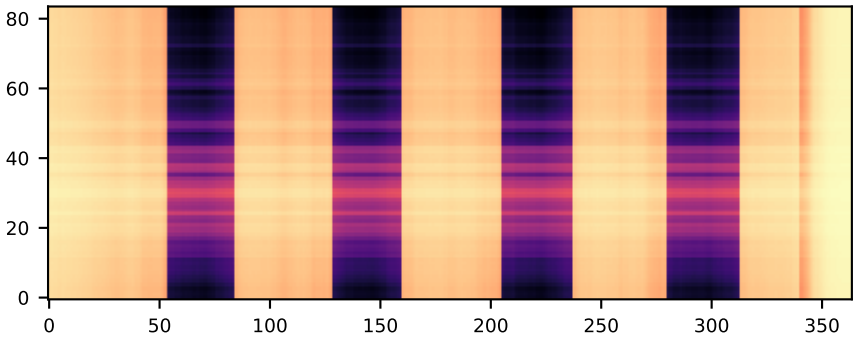
Nieto and Jehan [53] were the first to use NMF for the structural segmentation subproblem, however they added a constraint that the columns of $\mathbf{F}$ need to be convex combinations of the observations in $\mathbf{A}$. This means that, for an arbitrary weight matrix



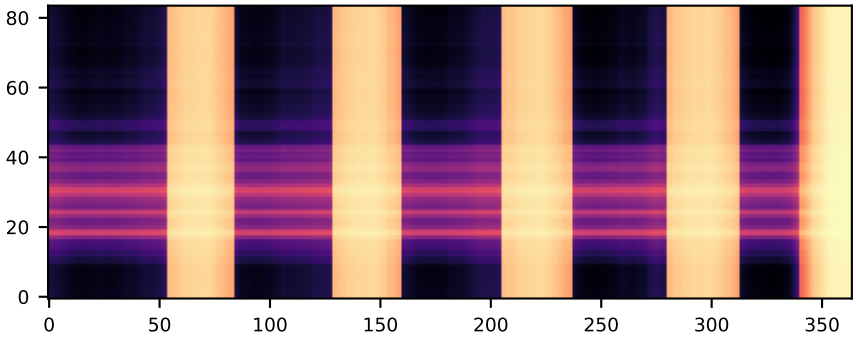

(a) Feature matrix $\mathbf{X}$




(b) Decomposition matrix $\mathbf{A}^1$




(c) Decomposition matrix $\mathbf{A}^2$




(d) Decomposition matrix $\mathbf{A}^3$

Figure 2.4: Example of NMF on the CQT feature matrix from 'Mrs. Robinson'

$\mathbf{W} \in [0,1]^{m \times r}$ such that each column $\mathbf{w}^j$ of $\mathbf{W}$ sums to one, the following equation holds for all $j \in [1:r]$:

$$\mathbf{f}^j = \mathbf{a}^1 \cdot W_{1,j} + ... + \mathbf{a}^m \cdot W_{m,j} = \mathbf{A}\mathbf{w}^j \tag{2.7}$$

This means that, in addition to Equation 2.4, the following equation holds:

$$\mathbf{F} = \mathbf{AW} \tag{2.8}$$

As $\mathbf{F}$ now is a set of convex combinations of the input matrix $\mathbf{A}$ and the input matrix is a feature matrix, each row $\mathbf{f}_i$ now contains information of the time frame $i$ across the entire song. As Nieto and Jehan found, this greatly improves the clarity of segment boundaries in the decomposition matrices.

In the method proposed by Nieto and Jehan, segment boundaries are finally found by running $k$-means clustering with $k = 2$ (i.e. is there a boundary at this frame or not) over each of the decomposition matrices, where each row represents an observation. The $r$ binary vectors that are returned by the clustering process are combined using a distance window, so that boundaries that are close to each other are averaged into a single boundary.

There are more approaches that use a form of NMF-based approach to DSA. Most notably, Weiss and Bello [72] proposed to use a shift-invariant form of *Probabilistic Latent Component Analysis* or SI-PLCA, where a vector $\mathbf{z}$ of mixing weights is added to the regular NMF estimation, such that $\mathbf{A} \approx \mathbf{FZG}$ where $\mathbf{Z}$ is the diagonal matrix of $\mathbf{z}$. Cheng, Smith, and Goto [7] proposed a method using *Non-negative Matrix Factor 2D Deconvolution* or NMF2D [47], which convolves $\mathbf{A}$ in both the $x$-axis and the $y$-axis.

The NMF-based approaches can be seen as novelty-based, as the $k$-means clustering aims to directly find the locations of segment boundaries, in other words the start of a novel section. The approach can however also be seen as homogeneity-based, as NMF describes the track in terms of individually homogeneous decomposition matrices.

### Structure Features

The following approach is introduced by Serra et al. [64] and aims to describe the structure features of the track, by using a time-lag matrix derived from the self-similarity matrix. A time-lag matrix describes, for a certain frame, the amount of time (in frames) it takes before the musical structure in the frame repeats itself. This is in contrast with a self-similarity matrix, that describes for a certain frame what frames have a high similarity to it.

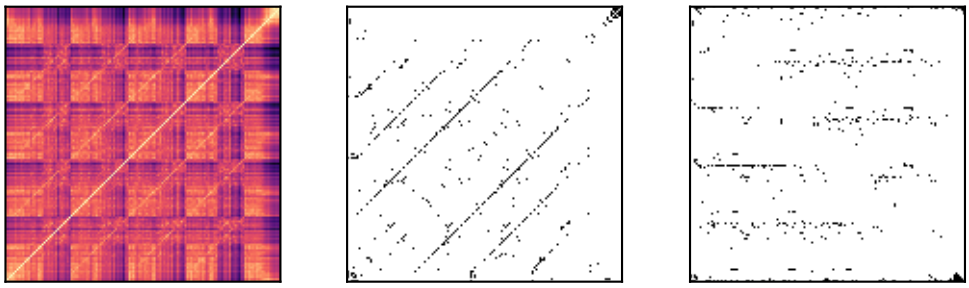
A time-lag matrix is binary ($L_{i,j} = 1$ if frame $i$ is repeated after $(i-j)$ frames, $L_{i,j} = 0$ if not), so in order to convert the self-similarity matrix to a time-lag matrix, the SSM is first converted into a binary representation. This is called a connectivity matrix, here denoted as $\mathbf{R}$,[2] and can be made by thresholding the SSM:

$$R_{i,j} = \begin{cases} 1 & \text{if } S_{i,j} \geq \tau_{i,j} \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

for certain threshold parameters $\tau_{i,j}$. However, Serra et al. showed that using an adaptive thresholding strategy yields better performance. They used the following threshold:

$$R_{i,j} = \begin{cases} 1 & \text{if } j \in \mathcal{N}_\kappa(i) \wedge i \in \mathcal{N}_\kappa(j) \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

[2] At first sight, it might be counterintuitive to denote a connectivity matrix with $\mathbf{R}$ and not $\mathbf{C}$. However a connectivity matrix (and, indeed a SSM) is a form of a recurrence matrix which is why it is denoted as $\mathbf{R}$. This is consistent with notation in literature.



(a) Self-similarity matrix (b) Connectivity matrix (c) Time-lag matrix

Figure 2.5: Visualization of a self-similarity matrix, recurrence plot and time-lag matrix of 'Mrs. Robinson' from Simon & Garfunkel

where $\mathcal{N}_\kappa(i)$ denotes the set of frame indices that belong to the $\kappa$ nearest neighbors of frame $i$ as measured by $\mathbf{S}$. $\mathcal{N}_\kappa(i)$ can be extracted from $\mathbf{S}$ by taking the positions of the $\kappa$ highest values from $\mathbf{s}_i$. The connectivity matrix that is extracted from the SSM of Simon & Garfunkel's 'Mrs. Robinson' can be seen in Figure 2.5b.

The conversion from the connectivity matrix to the time-lag matrix $\mathbf{L}$ is done as follows:

$$L_{i,j} = R_{i,i+j-2 \mod n} \tag{2.11}$$

The resulting time-lag matrix can be seen in Figure 2.5c. Intuitively, the conversion step transforms the diagonal lines into horizontal lines. The next step is to apply a 2-dimensional Gaussian filter over $\mathbf{L}$, however this Gaussian filter isn't symmetrical; it is made by multiplying a one-dimensional Gaussian window with the transpose of another one-dimensional Gaussian window. Both windows have the same variance, but different sizes.

Serra et al. [64] observed that the structural boundaries in the feature matrix $\mathbf{X}$ correspond to relative changes in the sequence of this smoothed time-lag matrix. To extract these changes, a novelty curve $\mathbf{c}$ similar to that of the Gaussian checkerboard kernel is constructed:

$$c_i = ||\mathbf{l}_{i-1} - \mathbf{l}_i||^2 \tag{2.12}$$

Where $||\cdot||$ denotes the Euclidean norm. Similar to the Gaussian Checkerboard kernel, the locations of the segment boundaries can be extracted from $\mathbf{c}$ by using an adaptive thresholding method.

As this approach primarily focuses on the diagonal lines in the self-similarity matrix, it can be seen as novelty-based. However, the connectivity matrix not only describes audio novelty, but audio homogeneity as well. Therefore this approach is both homogeneity-based and novelty-based.

### 2D Fourier Magnitude Coefficients

2D Fourier Magnitude Coefficients, or 2D-FMCs, are coefficients that can be used to reflect segment similarity. The 2D Fourier transform is a common operation in image processing, and the proposal to use only the magnitude of it in music processing has been first introduced by Ellis and Thierry [13]. This study showed that 2D-FMCs are invariant to rotation – not only in key (key shift invariance) but also in time (phase shift invariance). This property makes the 2D-FMCs very suitable for music information retrieval.

Using 2D-FMCs for the MSA subtask of segment clustering was proposed by Nieto and Bello [51]. Assuming some way of extracting segment boundaries, this study proposed to

construct a 2D-FMC 'patch' (matrix of coefficients) of equal size for every such segment. These patches were then clustered using $k$-means clustering. The resulting clustering was shown to have an accurate correlation with the ground truth segment clustering.

#### Eigenvalue Decomposition

The last technique that will be discussed is that of the Eigenvalue decomposition of self-similarity matrices. Eigenvalue decomposition has been used in MSA, specifically in audio segmentation [18, 10]. Eigenvalue decomposition in the context of structural segmentation [45, 23] was used not only for dimensionality reduction like NMF, but also to convert line structures in the self-similarity matrix into block structures. This makes it possible to convert a repetition-based problem, that needs to make use of the line structures, into a more traditional homogeneity-based problem that makes use of block structures. The process of MSA using eigenvalue decomposition is also called Laplacian segmentation.

## 2.2 Evaluating MSA Algorithms

It is a challenging problem to evaluate structural analysis algorithms. One reason for this is the complexity of the data that needs to be evaluated on its precision, which is especially true for the evaluation of the structural segmentation subtask. Another reason lies in properties of the ground truth that the algorithms will be compared to. As briefly explored in Section 1.1, humans include a measure of subjectivity in their judgement of music, and their structural annotations could reflect that. The following sections will discuss both problems, and its solutions that have been used in previous work.

### 2.2.1 Ground Truth: Different Datasets for MSA

In the history of music structure analysis, a lot of different datasets were made to evaluate the approaches developed through the years. These datasets varied a lot: in genre, recording quality, types of annotations and the quality and consistency thereof.

Datasets include, for a set of songs, at least one human annotation per song of its structure. Some datasets include annotations at multiple hierarchical levels. The Beatles TUT is a dataset of 174 annotations, published by the Tampere University of Technology. As the dataset is made out of entirely song from The Beatles, its genre is entirely pop. This is a dataset that is very commonly used for evaluating MSA algorithms [64, 32, 51, 42, 53, 72].

Another common dataset that is especially known for its size, is the SALAMI [66] or Structural Analysis of Large Amounts of Music Information[3] dataset. It contains 769 songs – from multiple sources – that all have annotations by at least two human annotators, and it contains high-profile annotations that correspond to the musical sections as defined in this thesis, but also lower-level annotations that describe patterns within these sections. Some of the tracks that are included in the SALAMI dataset are live recordings of slightly lower quality, which makes MSA based on these files more challenging. Noteworthy of this dataset is that a large subset (477 tracks) of the audio is publicly available through the Internet Archives.

Other datasets for MSA include the Structural Poly-Annotations of Music[4] or SPAM [52] that include 50 particularly challenging tracks with 5 annotations per track; and the INRIA semiotic music structure annotation [2] that include annotations of multiple music datasets[5].

[3] https://github.com/DDMAL/salami-data-public
[4] https://github.com/urinieto/msaf-data/tree/master/SPAM
[5] http://musicdata.gforge.inria.fr/structureAnnotation.html

### 2.2.2 Evaluation Approaches

If a dataset with structural annotations of tracks has been selected, the next challenge is to evaluate the algorithm's performance with it. As the MSA problem is divided into two subproblems and both subproblems are commonly solved individually, both subproblems are often evaluated individually as well.

#### Evaluating the Structural Segmentation Subtask

For comparing two structural segmentations of a track, it is unreasonable to qualify a certain boundary as a true positive only if the ground-truth contains a segment boundary at exactly the same location, and a false positive otherwise. This is firstly because this will result in a precision, recall and $F$-measure values of near zero as the state space is too sparse. A vector that contains at location $i$ for $i \in [1:n]$ a one if a segment boundary is expected at beat $i$ or a zero otherwise, would result in a vector with a length of around 500 to 800, while around one percent of that will contain ones thus denoting segment boundaries. Another reason that evaluating structural segmentation algorithms like this is unreasonable, is because, even if a segment boundary generated by an automated method would be slightly offset with relation to a segment boundary in the ground truth, this segment boundary could still perceptually make sense. Transitions in music are almost never instantaneous, so a segment boundary anywhere in this transitional period would describe that transition as a whole accurately.[6]

A common way to represent the performance of a structural segmentation algorithm is to use a 'hit-rate' value, where a segment boundary is considered a 'hit' if it falls within a certain period around a segment boundary location in the ground truth. The use of hit rates yields us precision, recall, and F-measure values that can be used for comparison. Periods that are most commonly used are 3 seconds and 0.5 seconds. In Jensen [29], the F-measure performance was measured with relation to the window period, and it showed that at a tolerance of around 3 to 4 seconds the increase in $F$-measure flattened. Additionally, Serra et al. [64] observed that a tolerance of 0.5 seconds yielded inaccurate results, and the 3 second window seems to be the most used overall [7, 37, 53, 57, 68].

In Nieto et al. [54] empirical evidence was presented that the traditional $F$-measure has a limited resemblance to the perception of segment boundaries. It was found that high precision is perceptually more important than high recall, and the traditional $F$-measure doesn't reflect that. Nieto et al. [54] proposed a more perceptually resembling metric for the evaluation of music structure. The proposed metric is the $F_{0.58}$ measure, where

$$F_\alpha = (1+\alpha^2) \cdot \frac{P \cdot R}{(\alpha^2 \cdot P) + R} \tag{2.13}$$

Where $P$ and $R$ denote the Precision and Recall respectively. Note that the value of $\alpha$ determine the relative importance of $P$ and $R$. If $\alpha < 1$, the value of $P$ will have a greater impact on the value of $F_\alpha$. However, if $\alpha > 1$, more importance is given to $R$. The traditional $F$-measure is thus only a single case of the $F_\alpha$-measure, where $\alpha = 1$ so as to make sure $R$ and $P$ are of equal importance.

#### Evaluating the Segment Labelling Subtask

For the segment labelling subtask evaluation, two different metrics are commonly used. The harmonic $F$-measure ($\alpha = 1$) is one of them, but *entropy scores*, as proposed in Lukashevich [40] are also often used.

The entropy scores consists of two different values: $S_o$ or the over-segmentation score, and $S_u$ or the under-segmentation score. These two scores have the advantage that they

[6] Or as accurately as possible using this 'flat' representation of musical structure. Refer to Section 5.3 for the discussion of flat versus hierarchical clustering.

tell something about the hierarchical level of the labelling. For instance, if $S_o$ is high but $S_u$ is low, this means that under-segmentation has occurred and that the estimated music structure most probably represents a higher hierarchical level than the ground-truth. The precision and recall values used for calculation of the (harmonic) $F$-measure has no such ability, which can be seen in the examples in Lukashevich [40]. A drawback of using entropy scores is that they can't be used directly as an accuracy rate, as the functions used to calculate the entropy scores aren't linear. There is therefore no conclusive answer as to what metric is the single best for flat structure analysis.

Note that the estimated labelling is done on the segment produced by the segment boundary detection algorithm, while the ground-truth labelling of course uses the annotated segment boundaries. This leads to the fact that, when the performance of such a labelling algorithms is evaluated, the performance of the underlying segment boundary detection algorithm is evaluated too. To evaluate the performance of a segment labelling algorithm individually, one could use the grount-truth segmentation as input for the labelling algorithm. However, when the performance of a MSA system as a whole needs to be evaluated, a metric for the segment labelling algorithm can be used as the 'one number'.

### 2.2.3 Frameworks for Approach Comparison

As can be read above, there is a large variety in how different parts of the music structure analysis pipeline can be evaluated. As every study uses a different combination of databases and evaluation metrics, it can be hard to perform a comparison between different approaches, solely based on the performances listed in the articles.

In Nieto and Bello [52], the *Music Structure Analysis Framework* or MSAF was introduced to allow for direct comparison of the performance different algorithms, including those that are discussed in Section 2.1, using different features, and evaluated on different datasets. Refer to Appendix A.1 for links to the MSAF source-code.

The MSAF allows for comprehensive comparison of different approaches. Regarding the structural segmentation subproblem, Nieto and Bello [52] found that Structure Features performed the best when looking at a harmonic $F$-measure of the hit-rate with a 3s window, however Laplacian Segmentation performed better when looking at the hit-rate with a 0.5s hit-rate. Regarding the segment labelling subproblem, Laplacian Segmentation performed marginally better when evaluated using the (harmonic) $F$-measure. However, 2D-FMC seemed to perform better when evaluated with entropy scores.

The score distribution of all different algorithms not only depends on evaluation metric, but it also depends heavily on the feature type that was used, the dataset that it was compared to, and to what annotator in the dataset it was compared. Additionally, the performance of segment labelling algorithms depends on the structural segmentation algorithm that was used to extract the segment boundaries.

## 2.3 Feature Fusion in DSA

The idea of using multiple feature types in DSA has seen some exploration. Some studies used a simple form of feature fusion, either by performing a summation of the self-similarity matrices [64] or by stacking multiple feature matrices [56, 57]. Kaiser and Peeters [31] introduced a method that fuses information from structure features and novelty kernels for structural segmentation.

In Tralie and McFee [69], a method was proposed to combine self-similarity matrices from different feature matrices using *Similarity Network Fusion*, or SNF. This method allows for a theoretically unlimited amount of features to be fused together, however it was used on hierarchical music segmentation which falls beyond the scope of this thesis.

As of yet, it's impossible to properly evaluate the possible advantages that some kind of feature fusion might yield with relation to using a single feature type. This is because the MSAF as described in Section 2.2.3 currently doesn't support any kind of feature fusion.

## 2.4 Approaches beyond DSA: Deep Neural Networks

Distance-based Segmentation and Annotation has been the traditional approach to Music Structure Analysis. However, there exist successful MSA methods that take another approach than DSA. These techniques are discussed below. For a more extensive report on the SbA-approach and Machine Learning in the context of Music Information Retrieval and Music Structure Analysis in particular, please refer to the thesis of Van Boven [4].

Machine Learning, and Deep Neural Networks or DNNs, have seen a lot of successes in Music Information Retrieval. Fields in MIR that DNNs are used for include Automatic Audio Segmentation or AAS [59] where audio is segmented and the segments classified as being *speech*, *music*, or some kind of *noise*, and Musical Onset Detection [74, 1] where the onsets (starting points) of musical events in a track are determined.

Especially for music onset detection, the best performing algorithm included the use of a Convolutional Neural Network or CNN (e.g. [61, 21]), and because of the similarity between musical onset detection and structural segmentation, CNNs were also introduced for use in MSA [70]. A state-of-the-art CNN-based MSA approach (e.g. [21]) scores considerably better (taken into account the variability of the evaluation metrics as discussed in Section 2.2.3) than a state-of-the art DSA-based approach, however DSA has as big advantage that it is completely explainable. The lack of explainability is still one of the big pitfalls of machine learning, which is why it's still desirable to explore possible improvements of explainable techniques as DSA.

Segmentation by Annotation, which is the main focus of the thesis of Van Boven [4], is as of yet an unexplored technique in the field of MSA. It has been used for AAS [20] with good results. SbA approaches commonly employ deep neural networks as CNNs or Long-short term memory (LSTM) networks. It is interesting to evaluate its feasibility in MSA.

# Chapter 3

# Methodology of Adding Feature Fusion

This chapter attempts to answer research question 1b: How can feature fusion be added to such approaches? This chapter presents several ways of feature fusion. Additionally, this chapter will introduce an experiment that will be used to answer research question 1: Can the performance of state-of-the-art DSA-based approaches to MSA be improved with feature fusion?

Section 3.1 proposes three methods of feature fusion. Section 3.2 discusses the way these methods will be implemented, while Section 3.3 discusses how the performance of these methods will be evaluated.

## 3.1 Feature Fusion

Feature fusion is, as previously described, the practice of combining different music features such as CQT, PCP and MFCC into a single representation. This representation can then be used in DSA-based MSA algorithms, replacing a single feature matrix. Fusion can occur at multiple points in the processing pipeline, for instance before or after SSM generation, or as a final step of multiple iterations of the same algorithm. Three implementations of feature fusion that this thesis proposes are discussed in the sections below.

### 3.1.1 Matrix Stack

Matrix Stack is the simplest way of feature fusion, and as discussed in Section 2.3, has been attempted in combination with certain algorithms. Given a set of $F$ $N \times M$ feature matrices $\left\{\mathbf{X}^f\right\}_{f=1}^{F}$ where $\mathbf{X}^f = \left[\mathbf{x}_1^f, \mathbf{x}_2^f, \ldots, \mathbf{x}_N^f\right]^\top$, the resulting fused feature matrix is calculated as follows:

$$\hat{\mathbf{X}} = \begin{bmatrix} \mathbf{x}_1^1 & \ldots & \mathbf{x}_1^F \\ \vdots & \ddots & \vdots \\ \mathbf{x}_N^1 & \ldots & \mathbf{x}_N^F \end{bmatrix} \tag{3.1}$$

Note that every value of $\mathbf{x}_i^f$ is a row vector, and thus every row of $\hat{\mathbf{X}}$ becomes a concatenation of the row vectors $\mathbf{x}_i^1$ to $\mathbf{x}_i^F$. $\hat{\mathbf{X}}$ thus becomes a $N \times (FM)$ matrix. $\hat{\mathbf{X}}$ can now be used directly for the calculation of the self-similarity matrix.

### 3.1.2 Novelty-curve Summation

This fusion approach is tailored to segment boundary detection algorithms that make use of a novelty curve, such as the Gaussian checkerboard and Structure Features. For every

feature, the SSM is calculated and the novelty curve is too, according to the algorithm that is used. We now have a set of $F$ novelty curve vectors of length $N$, $\left\{\mathbf{c}^f\right\}_{f=1}^{F}$ where $\mathbf{c}_i^f \in [0,1]$ denotes the novelty of beat $i$, according to feature $f$. We now calculate a fused novelty curve $\hat{\mathbf{c}}$, where for each $i$:

$$\hat{c}_i = \sum_{f=1}^{F} c_i^f \tag{3.2}$$

We can now extract the locations of the maxima from $\hat{\mathbf{c}}$ to get the segment boundaries. This can be done using the same adaptive thresholding method as would normally be used by the segment boundary algorithm.

### 3.1.3 Similarity Network Fusion

The last method of feature fusion that is proposed in this thesis, is Similarity Network Fusion, or SNF [71]. As discussed in Section 2.3, SNF was proposed in Tralie and McFee [69] for the hierarchical segmentation of music structure, however the technique hasn't yet been used together with flat segmentation techniques. The method defines a way to convert the set of $F$ $N \times N$ self-similarity matrices $\left\{\mathbf{S}^f\right\}_{f=1}^{F}$ into a single SSM-like representation $\hat{\mathbf{S}}$. This fused self-similarity matrix can the be used in any regular DSA algorithm.

The first step is to normalise the self similarity matrices into $\left\{\mathbf{W}^f\right\}_{f=1}^{F}$ such that

$$\mathbf{W}^f = \exp\left(-(\mathbf{S}^f / \mathbf{B}^f)^2\right) \tag{3.3}$$

Where $\cdot \ / \ \cdot$ denotes pairwise matrix division. $\mathbf{B}^f$ is a matrix containing 'bandwidth' coefficients and scales values of $\mathbf{S}$ relative to the values of its nearest neighbors. It is defined by

$$B_{i,j}^f = \frac{1}{6} \cdot \left(\frac{1}{\kappa} \cdot \left( \sum_{k \in \mathcal{N}_\kappa^f(i)} S_{i,k}^f + \sum_{k \in \mathcal{N}_\kappa^f(j)} S_{k,j}^f \right) + S_{i,j}^f\right) \tag{3.4}$$

Where $\mathcal{N}_\kappa^f(i)$ denotes the set of indices of the $\kappa$ nearest neighbors of of $i$, as measured by $\mathbf{S}^f$. From this, SNF constructs the doubly-stochastic transition probability matrices $\left\{\mathbf{P}^f\right\}_{f=1}^{F}$ of $\left\{\mathbf{W}^f\right\}_{f=1}^{F}$, which is calculated as follows.

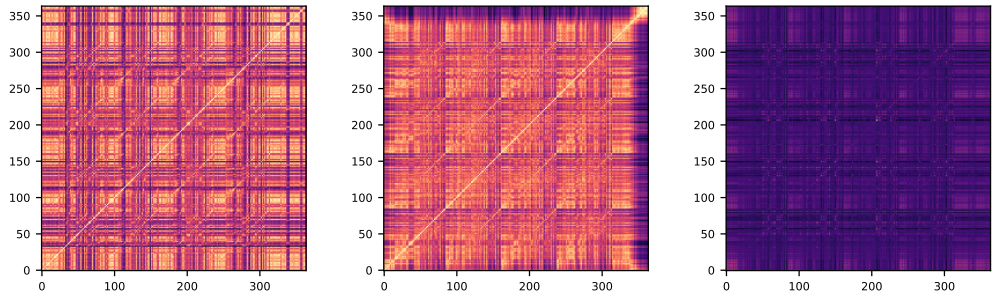
$$P_{i,j}^f = \begin{cases} \frac{1}{2} \cdot \frac{W_{i,j}^f}{\sum_{k \neq j} W_{i,k}^f} & j \neq i \\ \frac{1}{2} & j = i \end{cases} \tag{3.5}$$

$\mathbf{P}^f$ defines the distance of each value with relation to its row, *without* taking into account the matrix' diagonal. This is the reason we use $\mathbf{P}^f$ instead of $\mathbf{W}^f$ directly, as the similarity of a vector to itself firstly is irrelevant, and secondly results in the normalization of Equation 3.3 having to take into account these high values.

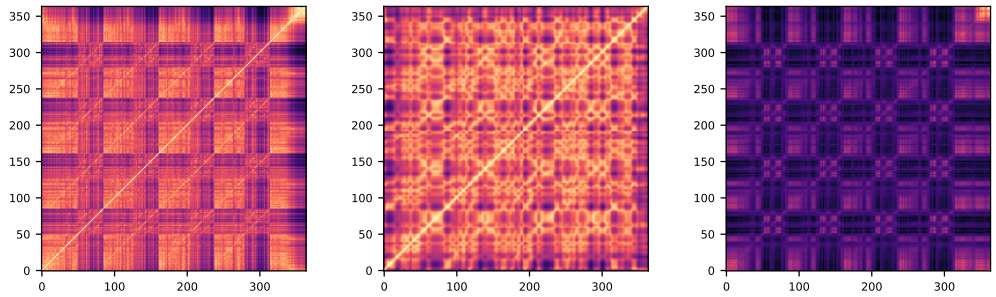
The next step is based on the assumption that close (i.e. highly similar) neighbors are more reliable than distant (i.e. only slightly similar) neighbors. We therefore want, for each value in $\mathbf{P}^f$, to let the distant neighbors to be of less importance with relation to near neighbors. We therefore define the nearest neighbor thresholded versions $\left\{\mathbf{Q}^f\right\}_{f=1}^{F}$ of $\mathbf{P}^f$:

$$Q_{i,j}^f = \begin{cases} \frac{1}{2} \cdot \frac{W_{i,j}^f}{\sum_{k \in \mathcal{N}_\kappa^f(i)} W_{i,k}^f} & j \in \mathcal{N}_\kappa^f(i) \\ 0 & j \notin \mathcal{N}_\kappa^f(i) \end{cases} \tag{3.6}$$

$\mathbf{Q}^f$ calculates the same values as $\mathbf{P}^f$, however only takes into account the $\kappa$ nearest neighbors of $\mathbf{W}^f$. The actual fusion is done by using the following iterative assignment.



(a) PCP (b) MFCC (c) PCP + MFCC fused



(d) CQT (e) Tempogram (f) All fused

Figure 3.1: Applying SNF to the self-similarity matrices from Simon & Garfunkel's 'Mrs. Robinson'.

We define $\mathbf{P}_t^f$ as $\mathbf{P}^f$ at iteration $t$, with $\mathbf{P}_0^f = \mathbf{P}^f$. The iteration is then defined as follows, for $t = 1, 2, \ldots, T$:

$$\mathbf{P}_t^f = \mathbf{Q}^f \times \frac{\sum_{k \neq f} \mathbf{P}_{t-1}^k}{F - 1} \times (\mathbf{Q}^f)^\top \tag{3.7}$$

The final representation can then be calculated as

$$\hat{\mathbf{S}} = \frac{1}{F} \sum_{f=1}^{F} \mathbf{P}_T^f \tag{3.8}$$

The result of applying Similarity Network fusion to Simon & Garfunkel's 'Mrs. Robinson' can be seen in Figure 3.1. We can see that the fusion of all four features (Figure 3.1f) results in quite a clear matrix, where for instance a Gaussian checkerboard matrix would have no problem to extract segment boundares from the blocks along the diagonal. The picture is quite a bit clearer than the fusion of only MFCC and PCP (Figure 3.1c), indicating that CQT and Tempogram together provide additional information that isn't present in MFCC and PCP alone.

As SNF is shown [69] to converge very quickly, I chose to use $T = 10$. I chose $\kappa = 3$ as this performed best in some small experiments and in Tralie and McFee [69] too. Refer to appendix C for the implementation of SNF in Python.

## 3.2 Means of Implementation

To implement the feature fusion methods above, an alteration to the MSAF is proposed. The Music Structure Analysis Framework, discussed in 2.2.3, is a framework that allows

for direct comparison of different DSA-based algorithms for MSA. The links to the source code and documentation of the MSAF can be found in Appendix A.1. The MSAF doesn't support feature fusion out-of-the box, which is why this thesis presents an altered version of the MSAF to allow for the inclusion of multiple features. A regular call to the MSAF looks like this:

```
msaf.process( song_dir
            , feature = "cqt"
            , boundaries_id = "sf"
            , labels_id = "fmc2d"
            )
```

The altered version of MSAF accepts a call like the following:

```
msaf.process( song_dir
            , feature = ["cqt", "pcp", "tempogram"]
            , boundaries_id = "sfncs"
            , labels_id = "fmc2d"
            , feat_def = "pcp"
            )
```

Where `feature` now has the ability to accept a list of feature types that will be fused together. The added parameter `feat_def` is the feature type that the MSAF will fall back to if a chosen algorithm doesn't support feature fusion.

## 3.3 Means of Evaluation

I will evaluate the performance of all the fusions that are possible with feature types: Pitch-class Profiles, Constant-Q Transform, Mel-frequency Cepstral Coefficients and Tempogram. I will use all possible combinations of these features, and to compare the performance of the feature fusions, I will also use every feature non-fused.

To evaluate the performance and possible increase in performance by using one of the ways of feature fusion, I will implement the three methods discussed above to the two structural segmentation algorithms: structure features (in the MSAF denoted with 'sf') and the Gaussian checkerboard kernel (in the MSAF denoted with 'foote'). I chose the Structure Features as this was deemed one of best performing structural segmentation algorithms in [52], and I chose the Gaussian checkerboard kernel as this is one of the simplest yet adequately performing algorithms. The Gaussian checkerboard kernel can therefore act as a good 'baseline' test. For the specific parameters that were used for all the algorithms, refer to Table 3.1

As evaluation metric, I will use the trimmed $F_{0.58}$-measure of the hit-rate with a 3s time window. I will use the $F_{0.58}$ measure as this metric is shown to be more in correspondence with human perception, as discussed in Section 2.2.2. The trimming means that the song start and song end boundaries won't be taken into account in the evaluation. As the detection of these boundaries are trivial, they aren't relevant for evaluation and comparison.

To evaluate the impact of performance of the structural segmentation algorithm on the segment labelling algorithm and the impact of the use of feature fusion thereon, I will use the (non-fused) 2D-Fourier magnitude coefficients (denoted as 'fmc2d') using the feature PCP. This combination is among the best performing segment labelling methods, as shown in [52]. I will measure its performance with relation to the kind of input boundaries that the labelling method will take as input. I will run this method both on the boundaries as extracted by all the different – fused and non-fused – combinations of structural segmentation algorithms with all different combinations of features, and on the ground truths. The performance will be measured using the harmonic mean of the entropy scores $S_o$ and $S_u$, denoted $S_f$.

| Algorithm | Parameter | Value | Explanation |
|---|---|---|---|
| **sfms/sfncs** | M_gaussian | 27 | Size of the 1D Gaussian filter in beats |
| | m_embedded | 3 | Number of embedded dimensions |
| | k_nearest | 0.04 | Multiplier for nearest neighbors of recurrence plot |
| | Mp_adaptive | 28 | Size of adaptive threshold for peak picking |
| | offset_thresh | 0.05 | Offset threshold for peak picking |
| | norm_type | np.inf | Type of normalisation |
| **sfsnf** | T | 10 | Amount of SNF iterations |
| | k_snf | 3 | Omount nearest neighbors, $\kappa$ in literature |
| | ssm_metric | seuclidean | Distance metric used for SSM calculation |
| | embed_next | 4 | Amount of vectors from the past to embed |
| | embed_prev | 1 | Amount of vectors from the future to embed |
| | norm_type | min_max | The normalisation type for SSM calculation |
| | k_nearest | 0.05 | See sfms / sfncs |
| | M_gaussian | 27 | See sfms / sfncs |
| | Mp_adaptive | 28 | See sfms / sfncs |
| | offset_thresh | 0.05 | See sfms / sfncs |
| **footems / footencs** | M_gaussian | 66 | Size of 2D Gaussian kernel |
| | m_median | 12 | Size of median filter |
| | L_peaks | 64 | Size of median filter for peak picking |
| | norm_type | min_max | Normalisation type |
| **footesnf** | T | 10 | Amount of SNF iterations |
| | k_snf | 3 | Amount nearest neighbors, $\kappa$ in literature |
| | ssm_metric | seuclidean | Distance metric used for SSM calculation |
| | embed_next | 1 | Amount of vectors from the past to embed |
| | embed_prev | 1 | Amount of vectors from the future to embed |
| | norm_type | min_max | See footems / footencs |
| | M_gaussian | 66 | See footems / footencs |
| | m_median | 12 | See footems / footencs |
| | L_peaks | 66 | See footems / footencs |

Table 3.1: Parameters for structural segmentation algorithms

I will use the tracks from the SALAMI [66] dataset that are publicly available through the Internet Archives. Links to the database and to the list of songs that I used can be found in Appendix A.2. The specific list of tracks that are used in this thesis can be found in Appendix B. The songs that are used include annotations from three different human annotators, yielding three different ground-truths.

# Chapter 4

# Evaluating Feature Fusion

In this chapter, the results of the experiments that are introduced in the previous chapter are discussed. With these results, an attempt will be made to answer research question 1: Can the performance of state-of-the-art DSA-based approaches to MSA be improved with feature fusion? In Section 4.1, the addition of feature fusion to the structural segmentation algorithms is evaluated. In Section 4.2, this is evaluated for segment labelling. In Section 4.3, an attempt to answer the research question is made and the results are discussed.

## 4.1 Performance of Adding Feature Fusion to Structural Segmentation Algorithms

### 4.1.1 Results

The mean trimmed $F_{0.58}$ measures of the un-fused algorithms and the best fusion on that algorithm can be seen in Figure 4.1 for the Gaussian checkerboard kernel and Figure 4.2 for the structure features algorithm. The algorithm names of this figure are consistent with MSAF, meaning that 'foote' represents the Gaussian checkerboard kernel, and 'sf' represents the structure features algorithm. The suffixes '-ms', '-ncs' and '-snf' denote the use of matrix stack, novelty curve summation and similarity network fusion respectively. The color represents the amount of features that are fused, and the alpha denotes to which annotator the boundaries are compared.

Matrix stack and novelty curve summation behave the same as their non-fused counterpart when combined with a single feature type, which is why we can use the performance of these to evaluated at the non-fused algorithms. For the non-fused version of the Gaussian checkerboard kernel, the best performing feature are MFCCs (M= .4776, SD= .1732, 95% CI=[.4571, .4981]). The difference compared to the second best performing feature, PCP, is statistically significant ($t(276) = 4.6973$, $p < .001$). For the non-fused version of the structure features algorithm, the best performing feature is CQT (M= .4901, SD= .1732, 95% CI=[.4695, .5106]) with a statistically significant difference compared to the second best performing feature, MFCCs ($t(276) = 5.5909$, $p < .001$). The difference between the best non-fused Gaussian checkerboard algorithm and the best non-fused structure features algorithm is insignificant ($t(276) = 1.2814$, $p = .2011$).

Looking at feature fusions for the Gaussian checkerboard kernel, matrix stack with PCP and MFCCs combined performs the best (M= .4955, SD= .1728, 95% CI=[.4751, .5160]). Novelty curve summation with PCP and MFCCs performs second best, and the difference between these two algorithms is insignificant ($t(276) = -.2638$, $p = .7921$). These algorithms perform better than the best non-fused checkerboard algorithm, however the difference in performance between these is but marginally significant ($t(276) = 3.1174$, $p = .0020$).

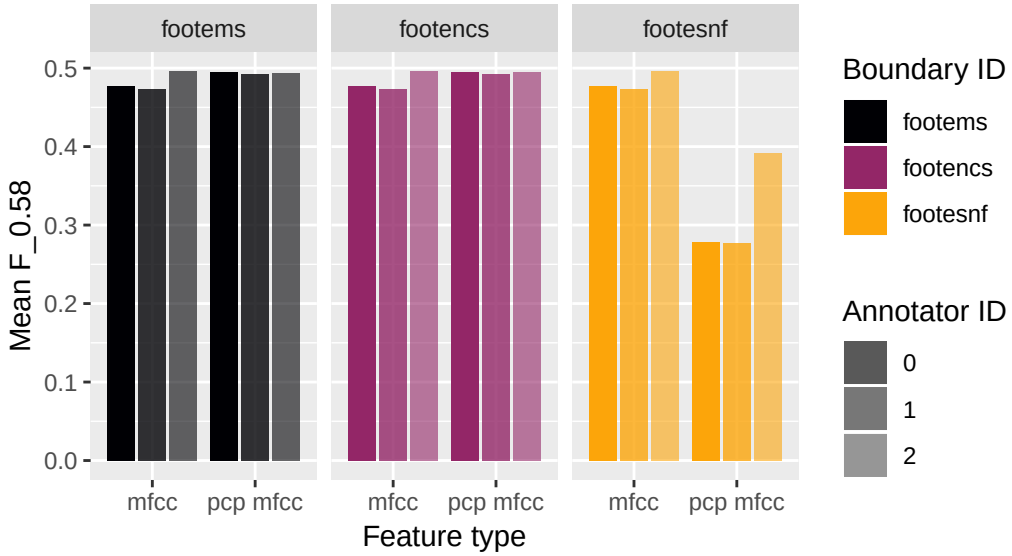



Figure 4.1: Mean $F_{0.58}$-measures of the Gaussian checkerboard kernel s versus fused or non-fused features. The suffixes '-ms', '-ncs' and '-snf' denote the use of matix stack, novelty curve summation and similarity network fusion respectively.

Looking at feature fusions for the algorithms based on structure features, the best performing combination is matrix stack with a combination of CQT and PCP (M= .4903, SD= .1730, 95% CI=[.4698, .5107]). However, the performance difference between all structure features-based algorithms with matrix stack that use CQT and at least one other feature in fusion, is insignificant (ANOVA: $F(7) = .136$, $p = .996$). The difference in performance between the best fused structure features algorithm and the best non-fused structure feature algorithm is insignificant ($t(276) = .3848$, $p = .7007$).

The best fused algorithm (the Gaussian checkerboard using matrix stack with PCP and MFCCs) performs better than the best non-fused algorithm (structure features with CQT), but the difference in performance is insignificant $t(276) = .5855$, $p = .5586$.

All other combinations of algorithms and features had such a perceptually significant decrease in performance, that it was deemed unnecessary to perform any statistical tests on them. Refer to Figure 4.6 for the mean trimmed $F_{0.58}$-measure of all structural segmentation algorithms with all feature combinations.

### 4.1.2 Conclusion

In conclusion, adding feature fusion yields a marginally significant increase in performance on the Gaussian checkerboard algorithm. This increase is only notable using matrix stack or novelty curve summation, and when timbre (MFCCs) is fused with chroma (PCP). This increase is not enough to outperform the structure features algorithm un-fused with CQT. Fusing CQT with another feature in a structure features algorithm does not yield a significant in- or decrease in performance.

Similarity network fusion yields a significant decrease in performance compared to both the un-fused 'base' algorithms, and the best fused combinations.

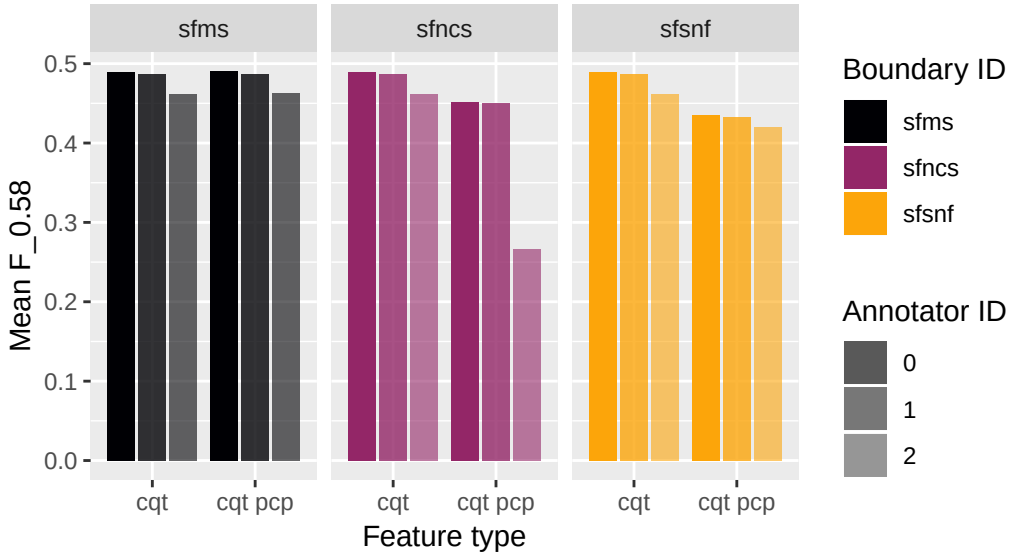



Figure 4.2: Mean $F_{0.58}$-measures of the structure features algorithm versus fused or non-fused features. The suffixes '-ms', '-ncs' and '-snf' denote the use of matix stack, novelty curve summation and similarity network fusion respectively.

## 4.2 Performance of Adding Feature Fusion to Segment Labelling Algorithms

### 4.2.1 Results

In the previous section, the performance increase of the structural segmentation algorithms when adding feature fusion is measured. In this section, we'll measure the performance of the segment labelling algorithms, when using segment boundaries that are produced by such fused algorithms and see whether an increase of performance can be significantly measured.

The labelling algorithm performs very good when paired with matrix stack and novelty curve summation versions of structure features, together with most feature combinations. Looking at the best performing structural segmentation combination, which is the Gaussian checkerboard kernel with MFCCs and PCP, we can see that the labelling performance paired with both the novelty curve summation and matrix stack variants are relatively good (resp. M= .5305, SD= .1007, 95% CI=[.5187, .5425] and M= .5331, SD= .0995, 95% CI=[.5214, .5449]) and the difference in performance between them is insignificant ($t(276) = .7365$, $p = .4621$).

The best total package makes use of structure features using novelty curve summation and CQT and PCP (M= .5459, SD= .1119, 95% CI=[.5327, .5591]), but this increase is insignificant with relation to the best un-fused combination, structure features with CQT $t(276) = 0.7727$, $p = .4404$. The comparison of the best un-fused package and the two best fused packages can be seen in Figure 4.3. Refer to Figure 4.7 for a comparison of all possible algorithms packages.

### 4.2.2 Conclusion

One of the best performing combinations are the labelling algorithm with the boundaries from the Gaussian checkerboard kernel with matrix stack and MFCCs and PCP as features, and the structure features algorithm with novelty curve summation and CQT and PCP.

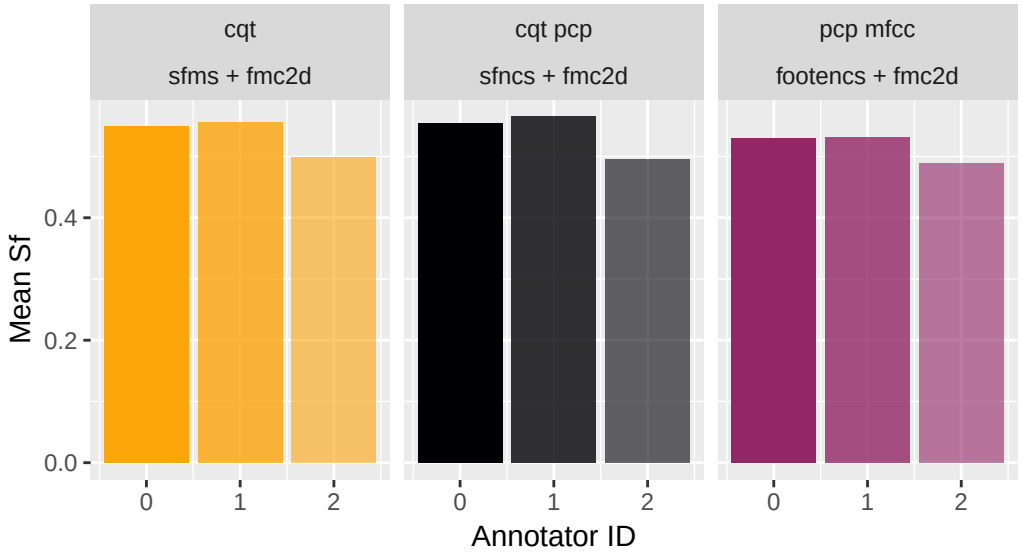



Figure 4.3: Mean $S_f$ of best performing algorithm combinations. The leftmost combination is an un-fused combination, whereas the middle and rightmost combinations are fused.

This is consistent with the findings concerning feature fusion in the structural segmentation algorithms, where was found that these combinations perform the best.

Noteworthy is the fact that the performance of the labelling algorithm together with all fused structure features based algorithms is marginally better than its non-fused counterparts, even though the performance of the underlying structural segmentation algorithms vary a lot in performance.

The performance increase of the best combination of the labelling algorithm with a fused algorithm, which is 'fmc2d + sfncs' with CQT and PCP as features, is insignificant when compared to a non-fused combination.

## 4.3 Discussion

Even though there is no significant increase in performance observable in the whole DSA package when adding feature fusion, it is hard to conclusively answer the first research question. This is firstly because there might be some improvements for feature fusion possible that could increase its performance, and secondly because of the quality of the dataset. These points are explained in the following paragraphs.

### 4.3.1 Observations from the Experiments

It is expected that the improvement that feature fusion can bring only (though marginally) emerges when a timbre feature is combined with a chroma feature (i.e. CQT + PCP with structure features and MFCCs + PCP with Gaussian checkerboard kernel). This is because these two feature types provide very different types of information about the track. The fact that the inclusion of the tempogram didn't provide any improvement in performance is unexpected however, as rhythm information isn't inherently present in chroma and timbre combined.

It is also quite interesting to see that similarity network fusion didn't provide any increase in performance to the structural segmentation algorithms – in fact the performance of the Gaussian checkerboard decreased significantly across the board when using similarity network fusion. In [69], it was shown that similarity network fusion yields an increase

in performance when compared to its non-fused counterpart, so there is some discrepancy between that work and this thesis. This could be due to the fact that [69] used similarity network fusion for hierarchical clustering and this thesis focuses on flat clustering. It could also be a possibility that the similarity network fusion as implemented in this thesis needs some parameter tuning.

During the SNF experiments, it was observed that the resulting fused matrix seems to lack some contrast in comparison to regular un-fused SSMs. This is visible in Figure 3.1f, where only the bottom-left and top-right of the plot show some orange (high) spots. This could be due to the fact that some feature types tend to contradict each other, and that the iterative averaging of SNF is unsuitable to represent such contradicting information. A careful selection of features that are complementing instead of contradicting could solve this problem, but it might also be the case that SNF is not the best technique for feature fusion in DSA.

The lack of contrast of SNF shouldn't be an issue to describe music structure, however the implications of this fact for the algorithms using SNF needs to be properly explored. This could for instance have an impact on the steepness of novelty curves, and thus on the boundary extraction from the novelty curve.

Structural segmentation shows a marginally significant improvement when using feature fusion. However, this improvement is more or less 'smoothed out' when paired with the segment labelling algorithm that uses 2D Fourier Magnitude Coefficients. Nowhere is the performance increase of feature fusion statistically significant in comparison to the best non-fused counterparts. This is probably due to the fact the labelling algorithm has a certain robustness to the performance of its underlying structural segmentation algorithm. This robustness has its limits though, which can be seen by the dips in performance that are present in Figure 4.7 in subplot 'fmc2d + footencs' that corresponds to dips in performance that are present in the same subplot in Figure 4.6. Additionally, as can be seen in Figure 4.4, the performance of the labelling algorithm does decrease when presented estimated segment boundaries instead of annotated (ground-truth) ones.

### 4.3.2 Assessment of the Quality of the Dataset

Another reason that is hard to conclusively answer the first research question, is because of the dataset. Firstly, the subset of SALAMI that is used in this thesis are recordings from live performances and are of particularly poor quality. These kinds of tracks are among the hardest to analyze, as the algorithm will have to cope with poor recording of the instruments, parts of silence, and applause and other noises from the audience. These are also present as labels in the ground-truth, meaning the algorithms will have to classify them too. The un-fused reference was measured on the same dataset, but using such a hard dataset makes possible differences in performance less visible, as the deltas become smaller.

Additionally, in Figures 4.1, 4.2, 4.3 and 4.4 it is clearly visible that there is a significant yet consistent discrepancy between the algorithm's performances and the annotator ID. This discrepancy is most probably due to a discrepancy between the different annotator's music representation. This would be a prime example of the inherent subjectivity of music representation as discussed in Section 1.1. To see whether the different annotators were working on different levels in the musical hierarchy, the mean $S_o$ and $S_u$ are plotted against annotator ID in Figure 4.5. We can see that the over- and under-segmentation scores of Annotator 1 lie closer together when compared to the difference in scores of Annotator 0, indicating that the evaluated algorithms are more in correspondence to Annotator 1. This is consistent with the patterns that can be observed in Figure 4.7, where the algorithm combinations tend to perform best when compared to Annotator 1.

Looking at the scores of Annotator 2, we can see that the over-segmentation score decreases just a little bit in comparison to Annotator IDs 0 and 1. However, the under-segmentation score of Annotator 2 is drastically lower than the scores of the other annota-

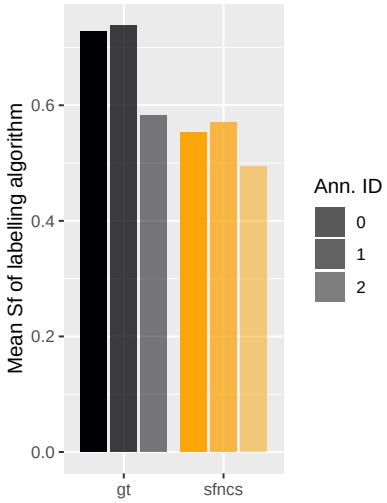



Figure 4.4: Performance of 2D-FMC when using ground truth as input boundaries.

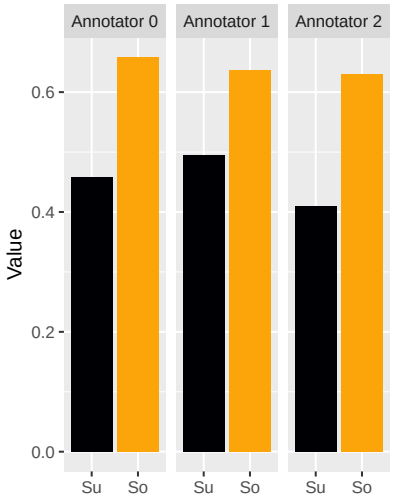



Figure 4.5: Mean over-segmentation and under-segmentation scores of all evaluated algorithms plotted against annotator ID.

tors. This gives an indication that the algorithms tend to under-segment when compared to the ground-truth of Annotator 2. This means that Annotator 2 made their annotations on a higher hierarchical level compared to both the evaluated algorithms and the other annotators.

The fact that there is such a disagreement between different annotators, means that these ground-truths are quite ambiguous. This raises the question whether we can use them as a good reference for evaluating MSA algorithms, or whether an ingenious combination of ground-truths may be needed. Maybe other datasets such as those discussed in Section 2.2.1 cope with annotator ambiguity better than this subset of the SALAMI dataset. It is also possible that the high ambiguity of this 'flat' kind of music structure representation makes it simply unsuitable for MSA algorithm evaluation.

### 4.3.3 Possible Improvements: Future Research in DSA Feature Fusion

The final algorithm (thus including segment labelling) doesn't yield a performance increase in this experiment when adding feature fusion, however there are still plenty of ways to improve upon. For instance, the segment labelling algorithms don't make any use of the feature fusion as of yet. As the structural segmentation subtask saw a marginal improvement from feature fusion, this could well be the case for segment labelling algorithms too.

Additionally, the feature fusion techniques themselves could be improved or altered. For instance, the novelty curve summation technique currently isn't robust against boundary offsets. Consider two novelty curves extracted from two different feature types, that both found the same boundary and reproduce that as a (local) maximum. However, both found the same boundary in a slightly different location. If the offset is large enough, this could result in two or more local maxima in the summation and thus, depending on the thresholding algorithm, in multiple boundaries where only a single boundary is expected.

This problem could possibly be solved if the un-fused novelty curves first get converted into multi-modal Gaussian distributions, where every gaussian distribution represents a possible boundary location. Such a representation has some resemblance to the practice of *Simultaneous Localization And Mapping* [11] or SLAM, where an autonomous robot

attempts to combine observations made by its exteroceptive sensors (i.e. its surroundings) with observations made by its proprioceptive sensors (i.e. internal, e.g. how many revolutions did each wheel spin). These sensor observations are commonly modeled with a Gaussian distribution too. Because of this resemblance, it might be possible to solve our problem using SLAM techniques, such as the *Extended Kalman Filter* or the *Markov Filter*. One of first challenges to overcome is then the method of determining what boundaries from all feature types are seen as describing the same boundary. One might use a windowing technique for this, such that when multiple boundaries fall within a window of a certain amount of time, they are treated as describing a single boundary.

Additional research on the SNF technique of DSA might also be of use. For instance, the SNF technique employ a lot of parameters, and an extensive experiment on the interaction of these parameters and SNF as a whole with the subsequent segmentation (or clustering) algorithm could yield improvements in performance. As discussed in Section 4.3.1, it could just be the case that SNF isn't suitable for DSA. Perhaps other methods of information combination such as voting ensembles of classifiers [43], that saw some succes in automatic genre classification [65, 35] could be of use for DSA.

This thesis used fusions of the most common feature types PCP, MFCC, CQT and Tempogram. However, there exist more feature types that might be of use for fused DSA. For instance, Tonnetz [25], denoting tonal centroids, and chroma variant *Chroma Energy Normalized* (CENS) [49] could be good contenders. Additionally, the chromagram and timbregram introduced in [29] make use of a Gaussian weighted version of the STFT, which was shown to be more robust against noise. This feature would be particularly interesting to use on datasets such as was used in this thesis.

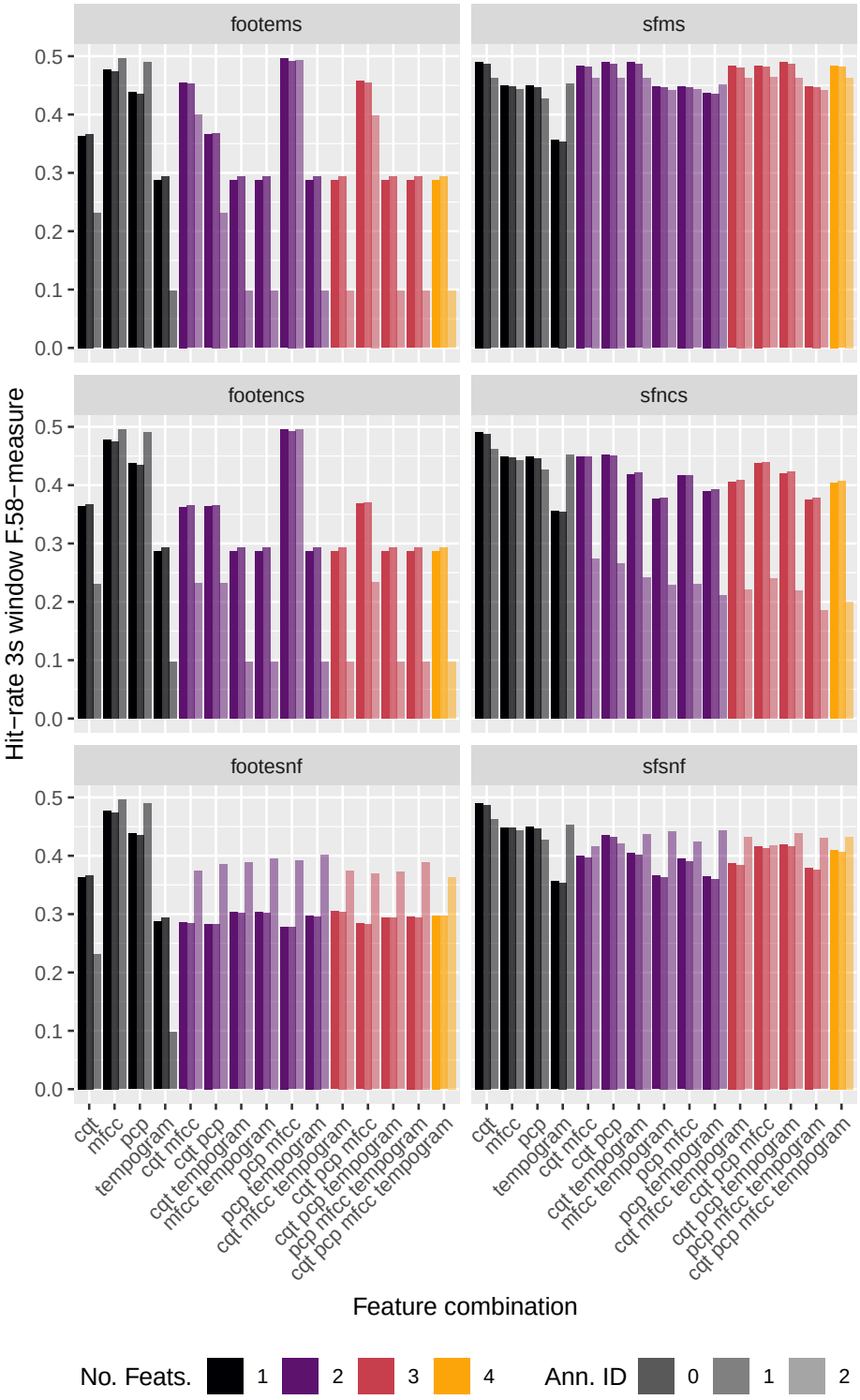



Figure 4.6: Mean $F_{0.58}$-measures of the tested structural segmentation algorithms versus feature combinations. The suffixes '-ms', '-ncs' and '-snf' denote the use of matix stack, novelty curve summation and similarity network fusion respectively.

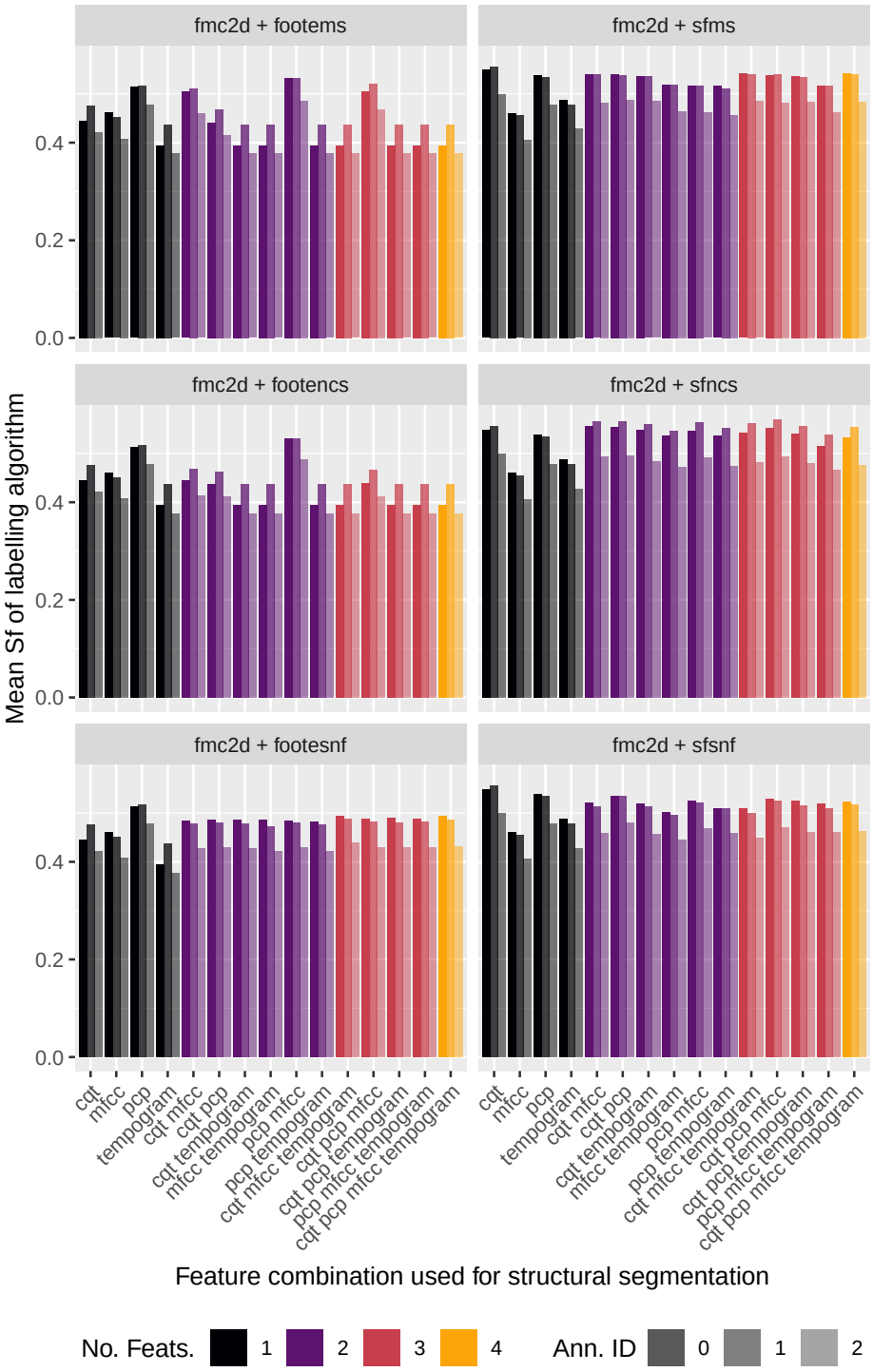



Figure 4.7: Mean $S_f$ values of the 2D-FMC segment labelling algorithm paired with different structural segmentation algorithms.

# Chapter 5

# DSA In Context

In this chapter, an attempt will be made to answer research question 2: How do DSA-based approaches compare to SbA-based or other MSA approaches? To this end, the general approach of distance-based segmentation and annotation will be discussed and put into context with other approaches to music structure analysis.

Section 5.1 compares the performance of different state-of-the-art MSA approaches thus attempting to answer research question 2a: How does the performance of DSA-based approaches compare to the performance of other approaches? Section 5.2 compares the approaches in general thus providing an answer to research question 2b: What are other properties of DSA-based approaches that might be advantageous or disadvantageous in comparison to other approaches? Section 5.3 looks forwards to the future of music structure analysis.

## 5.1 Comparing Performance

This section attempt to answer sub-question 2a: How does the performance of DSA-based approaches compare to the performance of other approaches? To this end, the performance of the DSA approaches as discussed in this thesis will be compared to the performances of the SbA-approach as proposed in the thesis of Leander van Boven [4] and another approach [22] that represents the current state of the art. In Chapter 4, the trimmed $F_{0.58}$-measure $\pm$3s tolerance was used as it provides better comparison across DSA algorithms. However, with the inter-approach comparison I will use the $F_1$-measure to provide the most complete comparison. Furthermore, I will compare entropy scores using the $S_o$ and $S_u$ values as introduced in Section 2.2.2, and the harmonic mean of these scores denoted $S_f$.

In Table 5.1, performances of different algorithms are listed. From this thesis the Gaussian checkerboard kernel with novelty curve summation and PCP and MFCC fused is listed as 'GCK'. 'SF' represents the Structure Features algorithm with novelty curve summation and CQT and MFCC fused together. SbA CNN and LSTM refer to two SbA approaches from the thesis of Van Boven [4]. SbA CNN denotes Segmentation by Annotation implemented using convolutional neural networks, whereas SbA LSTM denotes

| **Algorithm** | **$F_1$ $\pm$3s** | **$F_1$ $\pm$.5s** | **$S_f$** | **$S_u$** | **$S_o$** |
|---|---|---|---|---|---|
| GCK | 0.565 | 0.273 | 0.530 | 0.537 | 0.552 |
| SF | 0.49 | 0.320 | 0.557 | 0.523 | 0.644 |
| SbA CNN | 0.340 | 0.145 | 0.527 | 0.479 | 0.601 |
| SbA LSTM | 0.349 | 0.147 | 0.528 | 0.473 | 0.614 |
| Grill (2015)[1] | N/A | 0.558 | N/A | N/A | N/A |

Table 5.1: Performance of different algorithms compared

SbA using long-short term memory networks. The final algorithm, listed as Grill (2015) is proposed in Grill and Schlüter [22] and builds upon algorithms proposed in [62, 61, 70, 21]. This is a structural segmentation algorithm that makes use of convolutional neural networks and is the current state of the art in music structure analysis. Of note here is that this algorithm was evaluated on a slightly different subset of the SALAMI dataset, however, given the fact that it is also a subset from the publicly available Internet Archives, it is of the same quality as the subset used in this thesis and the thesis of Van Boven [4].

We can see that the algorithm proposed by Grill and Schlüter performs the best by a long shot – The $F_1$-measure ±3s of the second best algorithm (GCK) is about the same as the $F_1$-measure ±.5s of Grill and Schlüter's. The SbA approaches perform the worst in this direct comparison, however the models used were trained on a modified version of the ground-truth, where certain rare segment labels were substituted to the most equal label that occurs more often. This was done to cope with some of the quality issues of the dataset (i.a. those discussed in Section 4.3.2) that was used, and was done because of the fact that SbA directly uses the segment labels 'chorus', 'verse' etc. for its training. Refer to the thesis of Van Boven [4] for the full explanation of this decision.

Regarding the two best performing algorithms from this thesis, it is interesting to see that GCK yields a better $F_1$-measure ±3s tolerance, whereas SF yields a better $F_1$-measure ±.5 tolerance. So, even though using GCK results in more boundaries being in the neighborhood of the ground-truth, SF yields more boundaries that are close to the ground-truth. This is in line with the fact that SF is slightly more under-segmented than GCK, as this means that SF probably employs a higher certainty threshold to trigger a segment boundary. We can see that both SbA algorithms are slightly under-segmented too, which is in line with expectations as the networks were trained on less labels and thus less boundaries. The entropy scores of GCK are roughly equal to each other, meaning that it operates on the same hierarchical level as the annotators. As the algorithm of Grill and Schlüter only performs structural segmentation, there are no entropy scores available for it.

In conclusion, the state-of-the art approach of Grill and Schlüter has the best 'raw' performance on structural segmentation. In combination with a good performing segment labelling algorithm such as the 2D-Fourier Magnitude Coefficients or Spectral Clustering, it will probably perform very well as an MSA algorithm too. This is because the performance of such segment labelling algorithms are mostly reliant on the performance of the underlying structural segmentation algorithm, as observed in Chapter 4. This raises the question whether 'traditional' SSM approaches like the DSA approaches still have any relevance in the current state of the art. This will be explored in the following section.

## 5.2 Comparison of MSA approaches

### 5.2.1 DSA in comparison with SbA

The performance difference between DSA and SbA can't yet be accurately determined. Comparing results from this thesis, and the thesis of Leander van Boven, the DSA approach yields better results using the standard SALAMI ground-truth. However, as already mentioned, SbA was trained on a simplified version of the SALAMI ground-truth, and validation on that ground-truth yields significantly higher results. This is why it isn't possible to write off the one or the other approach in advance.

SbA has as a disadvantage that it is unable to detect two consecutive repeating sections. As SbA first splits the song into small pieces and classifies them individually, these sections will (if the method works correctly) all be classified as having an equal label. Next, SbA merges consecutive pieces with the same label into a single segment. This has as implication that the recurring sections are incorrectly merged together into one segment. Naturally, repetition-based DSA methods don't have this disadvantage.

A property of SbA is that it immediately returns segment labels, 'verse', 'chorus', et cetera. In contrast, the DSA approach returns clustered segments where each segment cluster represents a certain section type. This means that, if the end goal is to retrieve those meaningful segment labels, this needs to be done after the fact if a DSA algorithm is used whereas an SbA model (once trained) does it immediately. However, such labels are not only highly ambiguous with relation to genre, but prone to subjectivity as well. The DSA approach therefore is more context-agnostic than the SbA approach. In addition, the SbA approach as introduced in Van Boven [4] is based on machine learning. This means that during its training, the model fits to the data that is supplied to it. However, machine learning models can be quite prone to over-fitting, which adds to the danger of context-reliance of the SbA approach.

Another disadvantage of the current implementation of SbA for MSA, is that the model doesn't make use of the location of the small pieces in the track. However, this information can be of use for the annotation of the segment labels. For instance, the label 'intro' will of course only be present somewhere at the start of a song, and the label 'pre-chorus' will not often occur after a 'chorus'. As DSA does away with these meaningful segment labels right until (and only if) it is needed, DSA-based approaches don't have this problem.

### 5.2.2 DSA Compared to State of the Art

Both the algorithms described in this thesis, as (non-SbA) algorithms that employ CNNs such as the algorithm introduced by [22], make use of SSM-like representations. The difference lies in the patterns that they extract from them to extract boundaries. In the algorithms discussed in this thesis, we define the patterns we're looking for ourselves. CNN-based algorithms however learn the patterns from the training data. This gives CNN-based algorithms the big advantage that they are extremely adaptive and, indeed, can perform very well. However, this property has some disadvantages as well.

Firstly, the behavior of the algorithm will be directly influenced by the training data. In the context of MSA; if the training data consists of the ground-truth of only a single human annotator, the algorithm will fit to this annotator. However, as already discussed in this thesis and visible in the results in Chapter 4, different humans tend to annotate tracks quite differently. No single ground-truth can therefore be accurately seen as the absolute ground-truth. This means that if a CNN-based MSA algorithm is trained on a single annotator, it will only reflect the subjective interpretation of that annotator.

A second disadvantage of CNNs is the fact that the patterns that it learns during training are partly hidden in the complex non-linear operations that make up a neural network. This lack of explainability has as result that, even if a trained network is able to accurately perform structural segmentation, we don't gain any insight as to how the model came to this result. This is part of the general discussion of symbolic (e.g. DSA) approaches versus sub-symbolic (i.e. machine learning, SbA) approaches, where the sub-symbolic approach's lack of explainability is a big selling point of symbolic approaches. This debate in context of MSA is discussed in the following section.

### 5.2.3 The Debate of Explainability

Using the notions of explainability introduced in [9], methods using convolutional neural networks such as the SbA approach and the approach proposed by [61], are *comprehensible*. In their process, comprehensible methods show symbols, in the case of CNNs visualizations of the receptive fields. The user itself needs make inferences from these symbols in order to comprehend them. In contrast, DSA is *interpretable*. Interpretable systems are transparent, in the sense that the mathematical function that is employed can be understood by the user. The user needs technical knowledge to do this, but then the entire system can be analyzed. Deep neural networks such as CNNs are such complex processes that they can't be seen as interpretable. Approaches that use other types of deep neural networks,

such as Long-short term memory, are increasingly opaque; they don't emit any kind of comprehensible symbols, and the internal mapping is of such complexity that they can't be interpreted.

As discussed in 1.2.1, one of the goals of music structure analysis is to achieve machine music apperception. Machine music apperception can be described as a machine being able to fathom the internal patterns that underlie a piece of music. However, this would only be of use if that machine would be able to explain these patterns to humans. A neural network that hold some internal representation of a piece of music is therefore not enough.

According to [9], a truly explainable AI needs to be able to formulate a *line of reasoning*, or explanation as to why the AI made a certain decision. They argue that an algorithm that is only comprehensible or interpretable, only enable humans to formulate such a line of reasoning.

However, the practice of making Machine Learning more explainable is an active field [27, 60]. This means that the use of CNNs for MSA might in the future not only yield better results, but also be more explainable.

## 5.3 The Future of MSA

This thesis has focused on flat segmentation, meaning that the segmentation is done in one dimension, and so segment is 'subordinate'. However, as discussed in section 1.1, human music representation is most probably hierarchical instead of flat. In addition, we've seen results in Chapter 4 that support this, as the third annotator of the SALAMI subset used in this thesis seemed to work on a lower hierarchical level as compared to the first and second annotator. The fact that humans use a hierarchical structure for their music representation is further supported by [33], where was shown that humans respond differently to the same song with different hierarchical structures (keeping local structure intact). As doing flat segmentation on a structure that is inherently hierarchical leads to subjectivities, it might be a good idea to move to hierarchical segmentation instead. There is a little research in hierarchical music structure analysis. For instance, the feature fusion model [69] that was used in this thesis was used with hierarchical clustering. The SALAMI [66] dataset already includes two-level hierarchical ground-truths. McFee, Nieto, and Bello [46] introduced an evaluation metric for hierarchical segment boundary detection. Additionally, the MSAF already supports hierarchical segmentation and evaluation.

There have been some DSA variants for hierarchical segmentation, such as the spectral clustering algorithm of [69]. Some algorithms employ flat clustering, however can alter the hierarchical level the algorithm works on using a parameter [29, 37]. It might be hard to alter SbA to employ hierarchical clustering, as an entire song's context is needed to say something about the hierarchical level of a certain piece. This knowledge isn't present in SbA implementations as in the thesis of Van Boven.

To achieve true machine music apperception, the step needs to be taken to truly explainable artificial intelligence, or XAI. Given the fact that there is a lot of work done in making deep neural networks more explainable, it might prove easier to make the switch to XAI with CNN-based approaches to MSA.

Ideally, the interpretability of symbolic DSA approaches and the comprehensibility of sub-symbolic SbA or otherwise DNN-based approaches are combined into a single, explainable, model for music structure analysis.

# Bibliography


[1] Juan Pablo Bello et al. “A tutorial on onset detection in music signals”. In: *Transactions on speech and audio processing* 13.5 (2005), pp. 1035–1047.

[2] Frédéric Bimbot et al. “Methodology and conventions for the latent semiotic annotation of music structure”. In: *Publications Internes de l’Institut de Recherche en Informatique et Systèmes Aléatoires* (2012).

[3] Benjamin Blankertz. “The constant Q transform”. In: (2001). URL: http://doc.ml.tu-berlin.de/bbci/material/publications/Bla%5C_constQ.pdf.

[4] Leander Van Boven. “On Music Structure Analysis: Machine learning implementations of the Segmentation by Annotation approach”. Utrecht University, June 2020.

[5] Judith Brown. “Calculation of a constant Q spectral transform”. In: *The Journal of the Acoustical Society of America* 89.1 (1991), pp. 425–434.

[6] Judith Brown and Miller Puckette. “An efficient algorithm for the calculation of a constant Q transform”. In: *The Journal of the Acoustical Society of America* 92.5 (1992), pp. 2698–2701.

[7] Tian Cheng, Jordan Smith, and Masataka Goto. “Music structure boundary detection and labelling by a deconvolution of path-enhanced self-similarity matrix”. In: *International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2018, pp. 106–110.

[8] Arnie Cox. “Tripartite Subjectivity in Music Listening”. In: *Indiana Theory Review* 30.1 (2012), pp. 1–43. ISSN: 02718022. URL: http://www.jstor.org/stable/24045414.

[9] Derek Doran, Sarah Schulz, and Tarek Besold. “What does explainable AI really mean? A new conceptualization of perspectives”. In: *arXiv preprint arXiv:1710.00794* (2017).

[10] Shlomo Dubnov and Ted Apel. “Audio Segmentation by Singular Value Clustering.” In: *International Computer Music Conference*. 2004.

[11] Hugh Durrant-Whyte and Tim Bailey. “Simultaneous localization and mapping: part I”. In: *Robotics & Automation Magazine* 13.2 (2006), pp. 99–110.

[12] Daniel Ellis. “Beat tracking by dynamic programming”. In: *Journal of New Music Research* 36.1 (2007), pp. 51–60.

[13] Daniel Ellis and Bertin-Mahieux Thierry. “Large-scale cover song recognition using the 2D Fourier transform magnitude”. In: (2012).

[14] Robert Elmasian and Michael Birnbaum. “A harmonious note on pitch: Scales of pitch derived from subtractive model of comparison agree with the musical scale”. In: *Perception & Psychophysics* 36.6 (1984), pp. 531–537.

[15] Richard Evans et al. “Making sense of sensory input”. In: *arXiv preprint arXiv:1910.02227* (2019).

[16] Jonathan Foote. "Automatic audio segmentation using a measure of audio novelty". In: *International Conference on Multimedia and Expo*. Vol. 1. IEEE. 2000, pp. 452–455.

[17] Jonathan Foote. "Visualizing music and audio using self-similarity". In: *ACM international conference on Multimedia*. 1999, pp. 77–80.

[18] Jonathan Foote and Matthew Cooper. "Media segmentation using self-similarity decomposition". In: *Storage and Retrieval for Media Databases 2003*. Vol. 5021. International Society for Optics and Photonics. 2003, pp. 167–175.

[19] Takuya Fujishima. "Real-time chord recognition of musical sound: A system using common lisp music". In: *International Computer Music Conference* (Oct. 1999), pp. 464–467.

[20] Pablo Gimeno et al. "Multiclass audio segmentation based on recurrent neural networks for broadcast domain data". In: *European Association for Signal Processing Journal on Audio, Speech, and Music Processing* 2020.1 (2020), pp. 1–19.

[21] Thomas Grill and Jan Schluter. "Music boundary detection using neural networks on spectrograms and self-similarity lag matrices". In: *European Signal Processing Conference*. IEEE. 2015, pp. 1296–1300.

[22] Thomas Grill and Jan Schlüter. "Music Boundary Detection Using Neural Networks on Combined Features and Two-Level Annotations." In: *International Society of Music Information Retrieval*. 2015, pp. 531–537.

[23] Harald Grohganz et al. "Converting path structures into block structures using eigenvalue decompositions of self-similarity matrices." In: *International Society of Music Information Retrieval*. 2013, pp. 209–214.

[24] Peter Grosche, Meinard Müller, and Frank Kurth. "Cyclic tempogram—A mid-level tempo representation for musicsignals". In: *International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2010, pp. 5522–5525.

[25] Christopher Harte, Mark Sandler, and Martin Gasser. "Detecting harmonic change in musical audio". In: *ACM Workshop on Audio and Music Computing Multimedia*. 2006, pp. 21–26.

[26] Dorien Herremans. "MorpheuS: Automatic music generation with recurrent pattern constraints and tension profiles". In: (2016). ISSN: 2043-0167.

[27] Andreas Holzinger. "From machine learning to explainable AI". In: *World Symposium on Digital Intelligence for Systems and Machines*. IEEE. 2018, pp. 55–66.

[28] M. Jacobson. "Auto-threshold peak detection in physiological signals". In: *International Conference of the IEEE Engineering in Medicine and Biology Society*. Vol. 3. IEEE. 2001, pp. 2194–2195.

[29] Kristoffer Jensen. "Multiple scale music segmentation using rhythm, timbre, and harmony". In: *European Association for Signal Processing Journal on Advances in Signal Processing* 2007 (2006), pp. 1–11.

[30] Nanzhu Jiang and Meinard Müller. "Towards efficient audio thumbnailing". In: *International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2014, pp. 5192–5196.

[31] Florian Kaiser and Geoffroy Peeters. "A Simple Fusion Method of State And Sequence Segmentation for Music Structure Discovery." In: *International Society of Music Information Retrieval*. 2013, pp. 257–262.

[32] Florian Kaiser and Thomas Sikora. "Music Structure Discovery in Popular Music using Non-negative Matrix Factorization." In: *International Society of Music Information Retrieval*. 2010, pp. 429–434.

[33] Stefan Koelsch et al. "Processing of hierarchical syntactic structure in music". In: *National Academy of Sciences* 110.38 (2013), pp. 15443–15448.

[34] Ju-Hong Lee et al. "Automatic generic document summarization based on non-negative matrix factorization". In: *Information Processing & Management* 45.1 (2009), pp. 20–34.

[35] Pedro Ponce de León, José Inesta, and Carlos Pérez-Sancho. "Classifier ensembles for genre recognition". In: *Pattern recognition: Progress, directions and applications, Centre de Visió per Computador-Universitat Autónoma de Barcelona* (2006), pp. 41–53.

[36] Fred Lerdahl and Ray Jackendoff. "An overview of hierarchical structure in music". In: *Music Perception: An Interdisciplinary Journal* 1.2 (1983), pp. 229–252.

[37] Mark Levy and Mark Sandler. "Structural segmentation of musical audio by constrained clustering". In: *transactions on audio, speech, and language processing* 16.2 (2008), pp. 318–326.

[38] Beth Logan. "Mel frequency cepstral coefficients for music modeling." In: *International Society of Music Information Retrieval*. Vol. 270. 2000, pp. 1–11.

[39] Lie Lu, Muyuan Wang, and Hong-Jiang Zhang. "Repeating pattern discovery and structure analysis from acoustic music data". In: *6th ACM Special Interest Group on Multimedia international Workshop on Multimedia information retrieval*. 2004, pp. 275–282.

[40] Hanna Lukashevich. "Towards Quantitative Measures of Evaluating Song Segmentation." In: *International Society of Music Information Retrieval*. 2008, pp. 375–380.

[41] Roy De Maesschalck, Delpine Jouan-Rimbaud, and Désiré Massart. "The mahalanobis distance". In: *Chemometrics and intelligent laboratory systems* 50.1 (2000), pp. 1–18.

[42] Akira Maezawa. "Music boundary detection based on a hybrid deep model of novelty, homogeneity, repetition and duration". In: *International Conference on Acoustics, Speech, and Signal Processing*. IEEE. 2019, pp. 206–210.

[43] Ofer Matan. "On voting ensembles of classifiers". In: *Association for the Advancement of Artificial Intelligence Workshop on integrating multiple learned models*. Citeseer. 1996, pp. 84–88.

[44] Matthias Mauch, Katy Noland, and Simon Dixon. "Using Musical Structure to Enhance Automatic Chord Transcription." In: *International Society of Music Information Retrieval*. 2009, pp. 231–236.

[45] Brian McFee and Dan Ellis. "Analyzing Song Structure with Spectral Clustering." In: *International Society of Music Information Retrieval*. 2014, pp. 405–410.

[46] Brian McFee, Oriol Nieto, and Juan Pablo Bello. "Hierarchical Evaluation of Segment Boundary Detection." In: *International Society of Music Information Retrieval*. 2015, pp. 406–412.

[47] Morten Mørup and Mikkel Schmidt. *Sparse Non-negative Matrix Factor 2-D Deconvolution*. 2006.

[48] Meinard Müller. *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.

[49] Meinard Müller and Sebastian Ewert. "Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features". In: *International Conference on Music Information Retrieval*. Citeseer. Oct. 2011.

[50] Andrew Neath and Joseph Cavanaugh. "The Bayesian information criterion: background, derivation, and applications". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 4.2 (2012), pp. 199–203.

[51] Oriol Nieto and Juan Pablo Bello. "Music segment similarity using 2d-fourier magnitude coefficients". In: *International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2014, pp. 664–668.

[52] Oriol Nieto and Juan Pablo Bello. "Systematic exploration of computational music structure research." In: *International Society of Music Information Retrieval*. 2016, pp. 547–553.

[53] Oriol Nieto and Tristan Jehan. "Convex non-negative matrix factorization for automatic music structure identification". In: *International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 236–240.

[54] Oriol Nieto et al. "Perceptual analysis of the f-measure for evaluating section boundaries in music". In: *International Society for Music Information Retrieval Conference*. 2014, pp. 265–270.

[55] Francois Pachet and Jean-Julien Aucouturier. "Improving timbre similarity: How high is the sky". In: *Journal of negative results in speech and audio sciences* 1.1 (2004), pp. 1–13.

[56] Jouni Paulus and Anssi Klapuri. "Music structure analysis by finding repeated parts". In: *ACM Workshop on Audio and music computing multimedia*. 2006, pp. 59–68.

[57] Jouni Paulus and Anssi Klapuri. "Music structure analysis using a probabilistic fitness measure and a greedy search algorithm". In: *Transactions on Audio, Speech, and Language Processing* 17.6 (2009), pp. 1159–1170.

[58] Jouni Paulus, Meinard Müller, and Anssi Klapuri. "State of the Art Report: Audio-Based Music Structure Analysis." In: *International Society of Music Information Retrieval*. Utrecht. 2010, pp. 625–636.

[59] Ewald Peiszer, Thomas Lidy, and Andreas Rauber. "Automatic audio segmentation: Segment boundary and structure detection in popular music". In: *Proceedings of the International Workshop on Learning the Semantics of Audio Signals*. Vol. 106. Learning the Semantics of Audio Signals Paris. 2008, pp. 45–59.

[60] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models". In: *arXiv preprint arXiv:1708.08296* (2017).

[61] Jan Schlüter and Sebastian Böck. "Improved musical onset detection with convolutional neural networks". In: *International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2014, pp. 6979–6983.

[62] Jan Schlüter and Sebastian Böck. "Musical onset detection with convolutional neural networks". In: *International Workshop on Machine Learning and Music*. 2013.

[63] Ervin Sejdić, Igor Djurović, and Jin Jiang. "Time–frequency feature representation using energy concentration: An overview of recent advances". In: *Digital signal processing* 19.1 (2009), pp. 153–183.

[64] Joan Serra et al. "Unsupervised detection of music boundaries by time series structure features". In: *Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*. 2012.

[65] Carlos Silla Jr, Celso Kaestner, and Alessandro Koerich. "Automatic music genre classification using ensemble of classifiers". In: *International Conference on Systems, Man and Cybernetics*. IEEE. 2007, pp. 1687–1692.

[66] Jordan Smith et al. “Design and creation of a large-scale database of structural annotations”. In: *International Society of Music Information Retrieval* (2011), pp. 555–560.

[67] Stanley Stevens, John Volkmann, and Edwin Newman. “A scale for the measurement of the psychological magnitude pitch”. In: *The Journal of the Acoustical Society of America* 8.3 (1937), pp. 185–190.

[68] Mi Tian et al. “On the use of the tempogram to describe audio content and its application to music structural segmentation”. In: *International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2015, pp. 419–423.

[69] Christopher Tralie and Brian McFee. “Enhanced hierarchical music structure annotations via feature level similarity fusion”. In: *International Conference on Acoustics, Speech, and Signal Processing*. IEEE. 2019, pp. 201–205.

[70] Karen Ullrich, Jan Schlüter, and Thomas Grill. “Boundary Detection in Music Structure Analysis using Convolutional Neural Networks.” In: *International Society of Music Information Retrieval*. 2014, pp. 417–422.

[71] Bo Wang et al. “Similarity network fusion for aggregating data types on a genomic scale”. In: *Nature methods* 11.3 (2014), p. 333.

[72] Ron Weiss and Juan Pablo Bello. “Identifying Repeated Patterns in Music Using Sparse Convolutive Non-negative Matrix Factorization.” In: *International Society of Music Information Retrieval*. 2010, pp. 123–128.

[73] Shmuel Winograd. “On computing the discrete Fourier transform”. In: *Mathematics of computation* 32.141 (1978), pp. 175–199.

[74] Ruohua Zhou and Josh Reiss. “Music Onset Detection”. In: *Machine Audition: Principles, Algorithms and Systems*. IGI Global, 2011, pp. 297–316.

# Appendices

# Appendix A

# Links

## A.1 Music Structure Analysis Framework

- Source code: https://github.com/urinieto/msaf
- Documentation: https://msaf.readthedocs.io/en/latest/

## A.2 Salami database

- Salami database:
  https://github.com/DDMAL/salami-data-public
- List of the songs used in this thesis:
  https://github.com/DDMAL/salami-data-public/blob/master/metadata/id_index_internetarchive.csv
  Refer to Appendix B for the song identifiers (corresponding to columns 'SONG_ID')
  that are used in this thesis.

# Appendix B

# Track Identifiers

All the songs are extracted from the Internet Archives, and the annotations are extracted from the SALAMI datase. Below is a listing of all track identifiers that were used in this thesis. The IDs correspond to the numbers in columns 'SONG_ID' in the table that is listed in Appendix A.2.

```
956,  958,  960,  962,  964,  968,  970,  972,  974,  976,
978,  980,  982,  984,  986,  988,  990,  992,  994,  996,
998,  1000, 1004, 1006, 1008, 1012, 1014, 1016, 1018, 1020,
1022, 1024, 1026, 1028, 1032, 1034, 1036, 1038, 1042, 1044,
1046, 1048, 1050, 1054, 1056, 1058, 1060, 1062, 1064, 1066,
1068, 1070, 1072, 1074, 1076, 1078, 1080, 1082, 1084, 1086,
1088, 1090, 1092, 1094, 1096, 1098, 1100, 1102, 1104, 1106,
1108, 1110, 1112, 1114, 1116, 1118, 1120, 1122, 1124, 1128,
1130, 1132, 1134, 1136, 1138, 1142, 1144, 1146, 1148, 1150,
1152, 1154, 1156, 1158, 1160, 1162, 1164, 1166, 1168, 1170,
1172, 1174, 1176, 1180, 1182, 1184, 1186, 1188, 1190, 1192,
1194, 1196, 1198, 1200, 1202, 1204, 1206, 1208, 1210, 1212,
1214, 1216, 1218, 1220, 1222, 1224, 1226, 1228, 1230, 1232,
1234, 1236, 1238, 1240, 1242, 1244, 1246, 1248, 1250, 1254,
1256, 1258, 1260, 1262, 1264, 1266, 1268, 1270, 1272, 1274,
1276, 1278, 1280, 1282, 1284, 1286, 1288, 1290, 1292, 1294,
1296, 1298, 1300, 1302, 1304, 1306, 1308, 1310, 1312, 1314,
1316, 1318, 1322, 1324, 1326, 1328, 1330, 1332, 1334, 1336,
1338, 1340, 1342, 1346, 1348, 1350, 1352, 1354, 1356, 1358,
1360, 1362, 1364, 1366, 1368, 1370, 1372, 1374, 1376, 1378,
1380, 1382, 1384, 1386, 1387, 1388, 1390, 1391, 1392, 1394,
1395, 1396, 1399, 1400, 1402, 1403, 1404, 1406, 1407, 1408,
1412, 1414, 1415, 1416, 1418, 1419, 1420, 1422, 1423, 1424,
1427, 1428, 1431, 1432, 1434, 1435, 1436, 1438, 1439, 1442,
1443, 1444, 1446, 1447, 1448, 1450, 1451, 1452, 1454, 1455,
1456, 1458, 1459, 1460, 1462, 1464, 1467, 1468, 1470, 1472,
1474, 1475, 1476, 1478, 1479, 1482, 1483, 1484, 1487, 1488,
1490, 1491, 1492, 1494, 1495, 1496, 1498
```

This is a total of 277 tracks.

# Appendix C

# Implementation of Similarity Network Fusion

```python
import numpy as np
from scipy.spatial import distance

def compute_ssm(X, metric="seuclidean"):
    """
    Computes the self-similarity matrix of X using the
    given metric.

    Parameters
    ----------
    X: feature matrix as numpy array

    Returns
    ----------
    Self-similarity matrix as numpy array
    """
    D = distance.pdist(X, metric=metric)

    if np.any(np.isnan(D)):
        D = distance.pdist(X, metric="euclidean")

    D = distance.squareform(D)
    D /= D.max()

    return 1 - D

def embedded_space(X, b, f):
    """
    Return X with embedded space, such that for each row,
    the b previous rows and the f next rows are stacked
    horizontally

    Parameters
    ----------
    X: Feature matrix as numpy array
    b: number of rows to embed before
    f: number of rows to embed after

    Returns
    ----------
    Embedded feature matrix as numpy array
    """
    back = [np.roll(X, -(i+1), axis=0) for i in range(0,b)]
    front = [np.roll(X, i+1, axis=0) for i in range(0,f)]
    Y = np.hstack(tuple(back + [X] + front))
    return Y[b:-f,:] if f > 0 else Y[b:,:]
```

```
47
48 def fuse(self, X):
49     """
50     Performs Network Similarity Fusion on the given
51     input matrices.
52
53     Parameters
54     ----------
55     X: list of input matrices as numpy arrays
56
57     Returns
58     ----------
59     Fused self-similarity matrix as numpy array
60     """
61
62     F = len(X)
63
64     Ps = []
65     Qs = []
66
67     # Get parameters from 'config' dict
68     T = self.config["T"]
69     k = self.config["k_snf"]
70     p_emb = self.config["embed_prev"]
71     n_emb = self.config["embed_next"]
72     metric = self.config["ssm_metric"]
73
74     for f in range(F):
75
76         # Embed space, such that the 'p_emb' vectors before
77         # and the 'n_emb' vectors after each vectors get
78         # concatenated as a single vector in the matrix
79         X[f] = embedded_space(X[f], p_emb, n_emb)
80
81         # Compute the self-similarity matrix from it
82         S = compute_ssm(X[f], metric=metric)
83
84         # Let N be an array, such that N[j] is equal to the sum
85         # of the k nearest neighbors of column S[:,j]. SSM
86         # is symmetrical, therefore N[j] is also equal to
87         # the sum of the k nearest neighbors of row S[j,:].
88         N = np.partition( S - np.eye(S.shape[0])
89                         , S.shape[0] - k
90                         , axis = 1
91                         ) [:,-k:] \
92                             .sum(axis = 1)
93
94         # Construct N, such that N[i,j] is equal to the sum
95         # of the k nearest neighbors of row S[i,:], and
96         # column D[:,j]
97         N = N[:,None] + N[None,:]
98
99         # Construct bandwidth matrix
100        B = (S + N/k) / 6
101
102        # Construct affinity matrix
103        W = np.exp(-((D / B)**2))
104
105        # Create P
106        P = W.copy()
107
108        # Calculate case i!=j in eqn. 3.5, divide P[i,j] by
109        # 2 * the sum of row P[i,:] without P[i,j]
110        np.fill_diagonal(P, 0)
111        P = P / (2 * np.sum(P, axis=0)[:, None])
112
113        # Add case i==j in eqn. 3.5 (diagonal), which is
114        # a fixed value of 1/2
```

```
115         np.fill_diagonal(P, 0.5)
116
117         # Create W, sized the same as W. We immediately
118         # have all values for case 2 in eqn. 3.6
119         Q = np.zeros_like(W)
120
121         # Get the indices of the k nearest neighbors, such
122         # that for all x in N_ids[i,:], x is a neighbor
123         # of i as measured by S
124         N_ids = \
125             np.argpartition ( S - np.eye(S.shape[0])
126                             , S.shape[0] - k
127                             , axis=1
128                             ) [:,-k:]
129
130
131         # Get the values of the k nearest neighbors
132         # from W
133         W_nn = np.take_along_axis(W, N_ids, axis=1)
134
135         # Summation of each row, and make a column
136         # vector
137         W_nn_sum = W_nn.sum(axis=1)[:, None]
138
139         # Calculate the values of Q for the first
140         # case in eqn. 3.6
141         Q_c1 = W_nn / (2*W_nn_sum)
142
143         # Put these values in Q corresponding to the
144         # indices of N_ids.
145         np.put_along_axis(Q, N_ids, Q_c1, axis=1)
146
147         # Append resulting matrices to the lists
148         Ps.append(P)
149         Qs.append(Q)
150
151     # End for f in range(F)
152
153     # Iteration
154     for _ in range(T):
155
156         # Create a copy of all Ps for the next iteration
157         Ps_n = Ps.copy()
158
159         # Calculate eqn. 3.7 for each f in F
160         for f in range(F):
161
162             M = np.sum(Ps[:f] + Ps[f+1:], axis=0)
163             M /= (F - 1)
164
165             Ps_n[f] = Qs[f] @ M @ Qs[f].T
166
167         # Replace old Ps
168         Ps = Ps_n
169
170     # Calculate final fused matrix
171     fuse = sum(Ps) / F
172
173     # Min-max normalisation
174     fuse += fuse.min()
175     fuse /= fuse.max()
176
177     return fuse
```