Signal Segmentation: Real-Time Piano Performance

A Degree Thesis submitted to the Faculty of the
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

Bernat Xandri Zaragoza

In partial fulfillment of the requirements for the degree in

BACHELOR'S DEGREE IN TELECOMMUNICATIONS TECHNOLOGIES AND SERVICES ENGINEERING

Advisor: Philippe Salembier Clairon

ETSETB, Barcelona, Spain - Spring 2022

Abstract

Framed in the field of Music Information Retrieval (MIR), there is the widely discussed subject of audio-based Music Structure Analysis (MSA), where the Signal Segmentation Problem would be located. This field of research consists of the study of techniques capable of imitating the human perception of high-level structures that constitute a song. In other words, they aim to be able to identify where one section of the song begins and another ends, in a similar way to how a human listener would perceive it.

In this thesis, we present several implementations in the field of Music Segmentation. The first one consists of a revision of an offline segmentation technique, in which some new features have been implemented, thus improving its performance.
Going one step beyond the conventional form of this problem, thus seeking new lines of development, a novel real-time segmentation algorithm has also been implemented, which allows the exploration of the aforementioned high-level structures of a piece while it is being performed live.

Also, in order to be able to evaluate and analyse the behaviour of the different algorithms, a data-set has been curated to test the different implementations, an evaluation model has been designed to quantify the different solutions, and finally, an automatic optimisation strategy has been implemented to refine the different implementations.

Resum

Emmarcat en l'àmbit de la Recuperació d'Informació Musical (RIM), hi ha el tema àmpliament discutit de l'Anàlisi d'Estructures Musicals (AEM) basat en àudio, on quedaria ubicat el problema de la segmentació del senyal. Aquesta disciplina de recerca consisteix en l'estudi de tècniques capaces d'imitar la percepció humana de les estructures d'alt nivell que constitueixen una cançó. És a dir, pretenen ser capaços d'identificar on comença un passatge de la cançó i on acaba un altre, de manera semblant a com ho percebria un oient humà.

En aquesta tesi, presentem diverses implementacions en el camp de la Segmentació Musical. La primera consisteix en una revisió d'una tècnica de segmentació offline, en la qual s'han implementat algunes novetats, millorant així el seu rendiment.
Anant un pas més enllà del problema en el seu format convencional, també s'ha implementat un innovador algorisme de segmentació en temps real, que permet l'exploració de les esmentades estructures d'alt nivell d'una peça mentre s'està interpretant en directe.

Així mateix, per tal de poder avaluar i analitzar el comportament dels diferents algorismes, s'ha construït un dataset amb el que provar les diferents implementacions, s'ha dissenyat un model d'avaluació per quantificar les diferents solucions i, finalment, s'ha implementat una estratègia d'optimització automàtica per refinar les diferents implementacions.

Resumen

Enmarcado en el campo de la Recuperación de Información Musical (MIR), se encuentra el ampliamente discutido tema del Análisis de Estructuras Musicales (MSA) basado en audio, donde se ubicaría el Problema de la Segmentación de Señales. Este campo de investigación consiste en el estudio de técnicas capaces de imitar la percepción humana de las estructuras de alto nivel que constituyen una canción. Es decir, pretenden ser capaces de identificar dónde empieza un pasaje de la canción y dónde termina otro, de forma similar a cómo lo percibiría un oyente humano.

En este trabajo, presentamos varias implementaciones en el campo de la segmentación musical. La primera consiste en una revisión de una técnica de segmentación offline, en la que se han implementado algunas características nuevas, mejorando así su rendimiento. Yendo un paso más allá del formato convencional de este problema, y buscando así nuevas líneas de desarrollo, se ha implementado también un novedoso algoritmo de segmentación en tiempo real, que permite explorar las mencionadas estructuras de alto nivel de una pieza mientras esta se está interpretando en directo.

Además, para poder evaluar y analizar el comportamiento de los diferentes algoritmos, se ha elaborado un data-set con el que probar los diferentes algoritmos, se ha diseñado un modelo de evaluación para cuantificar las diferentes soluciones y, finalmente, se ha implementado una estrategia de optimización automática para refinar las diferentes implementaciones.

Dedication: *Li dedico aquesta tesi a la meva família, al meu director de TFG per haver-me fet sempre les preguntes adequades, als meus compis de biblioteca, i a totes les persones que m'han acompanyat en aquest projecte.*

Revision History and Approval Record

Revision	Date	Purpose
0	14/05/2022	Document creation
1	19/05/2022	Document revision
2	10/06/2022	Document revision
3	21/06/2022	Document submission

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Bernat Xandri Zaragoza	bernat.xandri@gmail.com
Philippe Salembier Clairon	philippe.salembier@upc.edu

Written by:		Reviewed by:	
Date	17/05/2022	Date	15/06/2022
Name	Bernat Xandri Zaragoza	Name	Philippe Salembier Clairon
Position	Project Author	Position	Project Supervisor

List of Figures

List of Tables

Abbreviations

DCT Discrete Cosine Transform

DFT Discrete Fourier Transform

ETSETB Barcelona School of Telecommunications Engeneering

FFT Fast Fourier Transform

HPCP Harmonic pitch class profiles

Hz Hertz

kHz Kilo-Hertz

MFCC Mel Frequency Cepstral Coefficients

MIR Music Information Retrieval

MSA Music Structure Analysis

RM Recurrence Matrix

SSM Self Similarity Matrix

Contents

1 Introduction

1.1 Statement of Purpose

The goal of this thesis has been to design and implement a set of algorithms capable of recognising relevant changes in the musical intention of a piano player, thus being capable of discerning transitions between homogeneous segments, and estimating the boundaries in-between said segments.
It was also a main requirement of this project that the segmented signal had to be the real-time recording of a live piano performance, and the analysis had to be performed without delays.
Going into further detail, it was necessary for the analyzed musical piece to be part of the recorded collection of live performances by the pianist Marco Mezquida[1], which is characterized by having a contemporary style, with strong influences of the atonal music of the end of the 20st century. This proved to be highly relevant in the decision making during the development of the project.

My motivation towards this thesis was mainly the interest I have developed during my bachelors degree towards the discipline of signal processing. Also, being someone who has studied music since a very young age, the opportunity of developing a project with a multidisciplinary approach, using both knowledge in signal processing and musical analysis, was something that really captivated me.

1.2 Requirements and Specifications

- **Requirements:**
  To be able to face this work, the initial requirements of the project needed to be established. These were devised based on the initial description of this work, so they were only conceived as an initial road map for the project.
  The initial requirements that served as the basic structure for the project are the following:

  1. **A data-set had to be built.** Either using synthetic data, built and designed by us to have specific segments which were annotated, or using real data. This data-set would later on be used to test and evaluate the different implemented algorithms.
  2. **A set of segmentation solutions needed to be implemented.** After having reviewed the current literature about signal segmentation, specifically focused on music segmentation, the more promising solutions had to be selected. The goal was to implement the chosen solutions, and to be able to evaluate these against the previously mentioned database. Thus, drawing conclusions about the different factors that affected the effectiveness, and how these could be improved.

[1] For more information about the pianist - https://marcomezquida.com/

3. **It was essential that a final solution be provided.** At the end of the project, it was expected that a working and optimised solution, capable of detecting changes in a piano performance in real-time, was going to be delivered.

- **Specifications:**
  The specifications from which this project started were relatively flexible. Since this was a project that was being started from scratch, we had the freedom to decide how it would be developed without prior conditionings.
  - The code needed to be implemented using a well known programming language, that was easily usable and readable, to ensure the replicability and adaptability of this project. As a personal choice, the project ended up being developed using Python as the main language.
  - It was required for the implemented solution to be able to detect segments based on some sort of high-level structure.
  - These solutions needed to be computationally efficient to some point, in order to guarantee that a real-time analysis was feasible.
  - Proper results and a qualitative analysis of these, needed to be conducted.

1.3 Project Background

The idea for this project arose from a collaboration between Sonar and UPC in one of its previous editions, but the development of this project started from scratch, taking no previous work or development as a starting point.
It was proposed in this way with the aim that there would be no conditionings when choosing which lines of research were going to be developed.

This project was conducted through the collaboration of the project director, who was the one that proposed the idea for this project, and the student who was be in charge of developing the thesis. The project is not framed within any research team, department or company.

1.4 Work Plan

The methodology used to organize the project was the same used in other courses of this degree. The project was organized in a work breakdown structure with different

work packages. A GANTT diagram is provided to illustrate said work package structure, without going into unnecessary detail.

1.4.1 Work Breakdown Structure

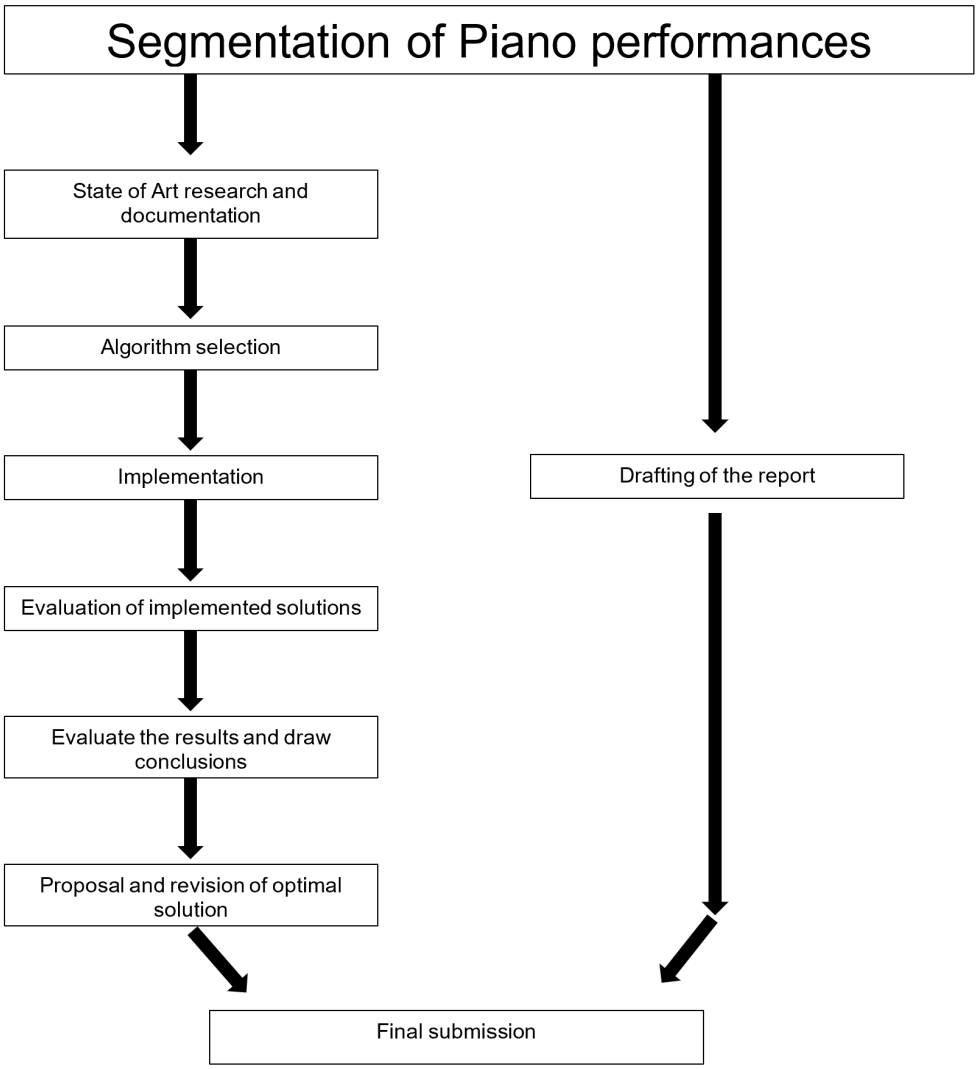


Figure 1: Work Breakdown Structure

1.4.2 Milestones

WP	Short title	Milestone/deliverable	Date
1	State of Art	Collection of Segmentation Techniques	16/03/2022
2	Algorithm selection	Set of techniques to be implemented	20/03/2022
3	Implementation	Set of implemented fully functional solutions	01/05/2022
4	Evaluation	Conclusions on which solution works better	01/05/2022
5	Final solution	Proposal for a final solution for this project	27/05/2022
6	Report	Final report to be submitted	05/06/2022

Table 1: Milestones

1.4.3 GANTT Diagram

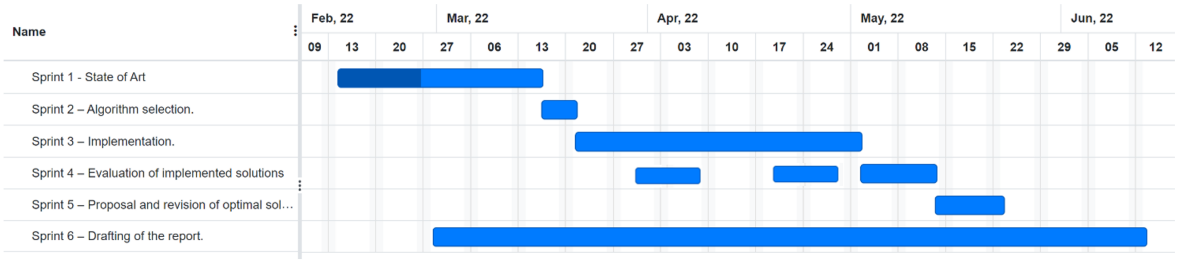


Figure 2: GANTT Diagram

1.5 Deviations of the Original Plan

There were only some small modifications in the planning of this project. The most crucial one being the decision of developing the implementation of the algorithms in parallel with the evaluation model for said solutions.

This was made in such way with the aim of being able to use the conclusions drawn from one implementation, and the intuition developed with this project, in the development of the following solution, therefore avoiding previous mistakes.

2 Overview of the Signal Segmentation Problem

It is usual to find that the signal one may be interested in studying, is actually a time-series which can be represented as a sequence of discrete segments of specific length.
In mathematics, a time-series is described as a set of data points, indexed in a natural temporal ordering, thus being a discrete-time sequences of data. Real-world examples of time-series could be the evolution of water temperature in the process of making a cup of coffee, the output of an electrocardiogram machine connected to a patient, or the audio recording of any musician performing a musical piece.

The so called Signal Segmentation Problem refers to the question of being able to recognise segments that present a certain degree of resemblance within themselves, and differentiate them from others, while detecting the location where this transition occurs (this will be referred as a boundary from now on). Intuition tells us that there must be an element causing this self-similarity in the frames that constitute the so-called 'segment', and also conditioning the differences between the different segments that make up the time-series. From now on, this factor will be referred as an element of homogeneity. It can also be assumed that if it were possible to measure change in this element of homogeneity, it would also be possible to detect where the boundaries are located.

What makes this problem certainly not a triviality, is the difficulty that even describing this element presents.

To illustrate this difficulty using a fairly simple example, when trying to segment a heartbeat signal, it could be defined that a segment aligns with each individual heartbeat, which would give us a segmentation like the one in the following figure 3:

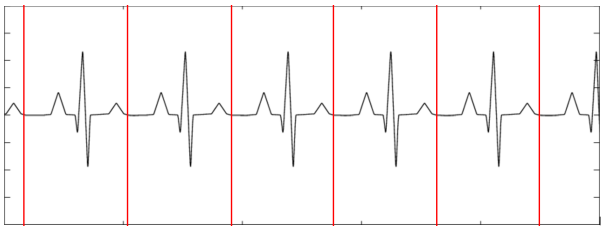


Figure 3: Cardiogram signal segmented in individual heartbeats

However, it may be of interest to rather detect the two periods that integrate a full hearbeat (diastole and systole), therefore obtaining a segmentation such as the one in figure 4, which clearly discerns of the one previously shown in figure 3:

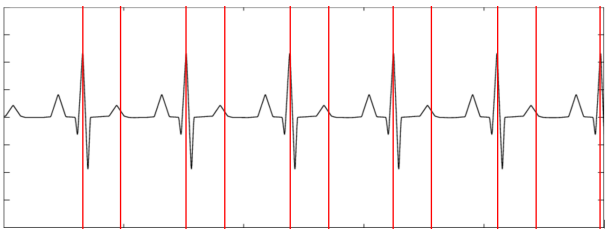


Figure 4: Cardiogram signal segmented by systole and diastole

This brief and oversimplified example, allows us to introduce and illustrate that one of the difficulties faced with this segmentation problem is to get the segments and boundaries estimated by the algorithm to align with the element of homogeneity that we are interested in detecting.

This difficulty becomes especially notable when dealing with music, which is characterized by many remarkably distinct but intertwined elements. This makes music an extreme case in signal segmentation, due to its subjective, ambiguous, and hierarchical nature.
To face this difficulty, an intensive study of the state of the art has been conducted on the different approaches that exist for segmentation in music (see section 2.1), and the different descriptors that exist to characterize the information contained in said signal (see section 3.1).

2.1 The Signal Segmentation Problem applied to Music

Framed in the field of Music Information Retrieval (MIR), there is the widely discussed subject of audio-based Music Structure Analysis (MSA), where the Signal Segmentation Problem would be located.

The basic premise of MSA is that any song can be divided into well defined independent segments, that match the perception of the song by a human listener. This task originates from an old and well extended practice in music theory: analyzing the form of a musical piece by identifying important segments, thus inferring the existence a high-level structure. This segmentation can be conducted either at a short time scale, such as motives or phrases, which are short musical motives that tend to conceal a recurring motive in the piece, or longer parts such as the classical A-B-A structure found in any classical Aria.

While this high-level structure can be very subjective, and can usually be influenced by trends or musical periods, there is usually a broad agreement, both between music experts an non-musician listeners, about which boundaries are more important or well defined, as detailed in [2]. The aim of audio-based MSA (from now on referred as MSA) is to automatize the correct detection of this boundaries, thus segmenting the musical piece and unraveling the high-level structure hidden in the music.

In the past two decades there has been a growing tendency to try to improve and push the state of the art in music segmentation through solutions based on machine learning models. Even though there have been major breakthroughs using these methods, and models based on deep learning using convolutional neural networks seem to be very promising, yielding superior scores in most metrics [3], in this project we will mainly focus in approaches using classic signal processing models, in which there is still a lot of room for improvements and new lines of research.
This was decided mainly due to two reasons: the project requires a solution that is computationally light, and that we want a solution that is designed specifically for the type of music that we want to segment, for which we do not have a database large enough to be able to successfully train a machine learning model.
Finally, also the lack of knowledge of these technologies on the part of the author would have meant a considerable push back.

In the line of research using classical signal processing methods, there are four main approaches when facing this challenge:

2.1.1 Novelty

The novelty approach [1] takes as an *a priori* that music will always be locally homogeneous on both sides of any given boundary. Therefore, the main focus of any novelty technique is to try to quantify the change at any given time, and establish a decision threshold based on which the boundaries are separated from homogeneous segments.

Therefore, the segments that a novelty based algorithm excels in identifying are those that start or end at a point in a given piece where one or more music descriptors (such as rhythm, harmony or timbre) change drastically.
These differences in terms of musical intention can be visualized as blocks in the Self Similarity Matrix (SSM, see section 3.2.1):

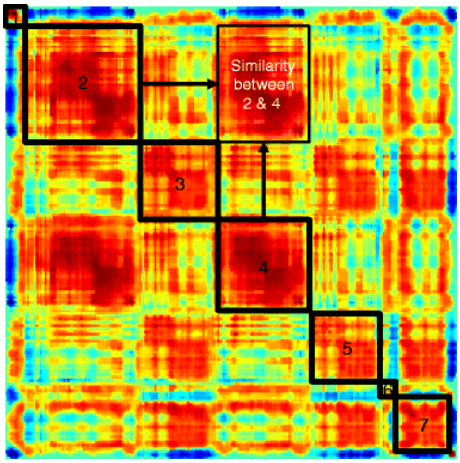


Figure 5: Similarity and cross-similarity between segments identified in a SSM [1]

2.1.2 Homogeneity

The homogeneity approach, also known as State Representation Analysis [4] [5], can be seen as the other side of the same coin when compared with the novelty approach. While novelty focus on detecting change, State Representation methods try to detect the lack of it. Homogeneity based algorithms aim to detect a certain musical aspect (the homogeneity factor previously mentioned in 2.1) which remains constant across the whole segment. This could be found in either a passage where there is an harmonic modulation, a consistent rhythm across some compasses or a passage played by only a certain instrument.

Referencing the SSM, already shown in figure 5 the State Representation methods basically aim to locate blocks of low distance on the main diagonal. These blocks are formed when the used feature descriptor (see section 3.1) remains somewhat similar during an occurrence of a musical part.

2.1.3 Repetition

The repetition approach, also known as Sequence approach [6], is based on the detection of repetitive segments. These segments are those that can be identified due to their re-occurrences in a given piece, regardless of how novel/homogeneous they are on a smaller time scale. Algorithms based in this method detect patterns that correlate with musical structures commonly used in certain styles as indicators for the beginning or closure of a section.

Sequence-based methods aim to locate off-diagonal stripes (a stripe representing low distance of two sequences), as represented in figure 6.

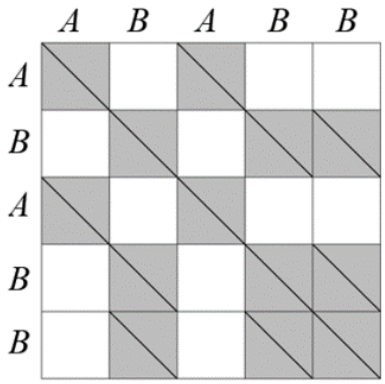


Figure 6: Off-diagonal similarity identified in a SSM

2.1.4 Regularity

The regularity approach [7], is based on the fact that, usually, musical segments hold a certain degree of regularity. Methods based on this principle usually heavily rely on beats and tempo analysis, trying to find structure in the number of beats per segment.

The main difference with respect to the homogeneity-based approach, is that regularity algorithms are more oriented on the detection of more straightforward patterns. For instance, the duration of segments that tend to span an integer number of beats.

3 Theoretical Background for Music Segmentation

In order to properly approach the problem of Music Segmentation, it is necessary to study some of the commonly used techniques and mathematical descriptions. In this section, the most crucial theoretical concepts used in this project will be introduced, so that the reader can later on understand the decisions and processes described in the implementation of this thesis (see section 5 and 6), but leaving more extensive descriptions to Appendix 1 (see section 10).

3.1 Audio Features Descriptors

In this subsection, a brief explanation of the audio feature descriptors that are used throughout this project are presented. Even though the author is aware of other methods such as Tonal Centroids, Pitch-Chromograms or Zero Crossing [8], the analysis of these techniques will be avoided, since they have not been found relevant for our purpose.
In summary, the parameterization itself is a fundamental step in the development of a segmentation algorithm, since it has a direct effect on how the boundaries are detected.

3.1.1 Mel Frequency Cepstral Coefficients - MFCC

The Mel Frequency Cepstral Coefficients, from now on referred as MFCC, is a well known and commonly used feature extraction technique, which consist in the parameterization of each frame of the signal.
The MFCC uses the Mel scale, which is based on the way humans distinguish between frequencies (a non-linear perception), to divide the frequency band into sub-bands, and then extracts the Cepstral Coefficients using the Discrete Cosine Transform (DCT). Since it is based on the way humans perceive sound, it is a good choice for audio feature extraction, specially since our main focus is trying to reflect in our segmentation algorithm the way humans perceive the musical piece.

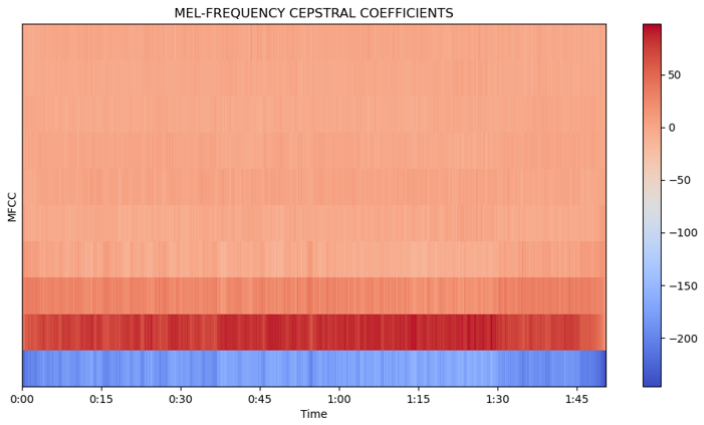


Figure 7: Example of MFCCs extracted from a piano track.

For a detailed explanation of how this technique works and the different nuances it presents, please refer to the section 10.1.1 of the annex.

3.1.2 Harmonic Pitch Class Profiles - HPCP

Harmonic pitch class profiles (from now on, referred as HPCP) is a feature extraction technique, based on a pitch class profile descriptor. HPCP is an improved pitch distribution feature, constituted by sequences of feature vectors that describe tonality, measuring the relative intensity of each of the 12 pitch classes (and its harmonic tones) of the equal-tempered Eastern scale, within a frame. This results in a 12-dimensional profile for each frame, and it is typically visualized as a $12 \times N$ matrix

This feature extraction technique differs from the one explained in the section above (see section 3.1.1) mainly in that it is strongly based on pitch and tonal structures, as opposed to MFCC that focus more on timbre analysis.

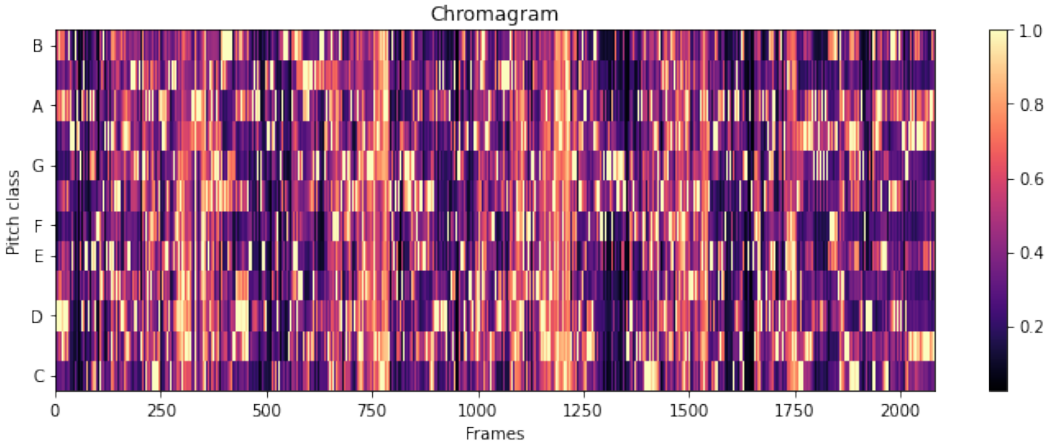


Figure 8: Example of normalized HPCPs extracted from a piano track.

For a detailed explanation on this technique, please see section 10.1.2

3.2 Tools for Structure Discovery

Once the features of the studied signal have been extracted, and therefore it is possible to work with a parameterization or characterization of the audio (as explained in section 3.1), it is proper to proceed with the structure discovery analysis.
For this purpose, there are different techniques and approaches, which will be described below.

3.2.1 Self Similarity Matrix - SSM

One of the pillars for the structure discovery analysis is the Self Similarity Matrix [9] [10]. The construction of this matrix (from now on referred as SSM) consists in the comparison of all pairwise combinations of frames (as illustrated in figure 9a), using a quantitative similarity metric (further detail in section 10.3).

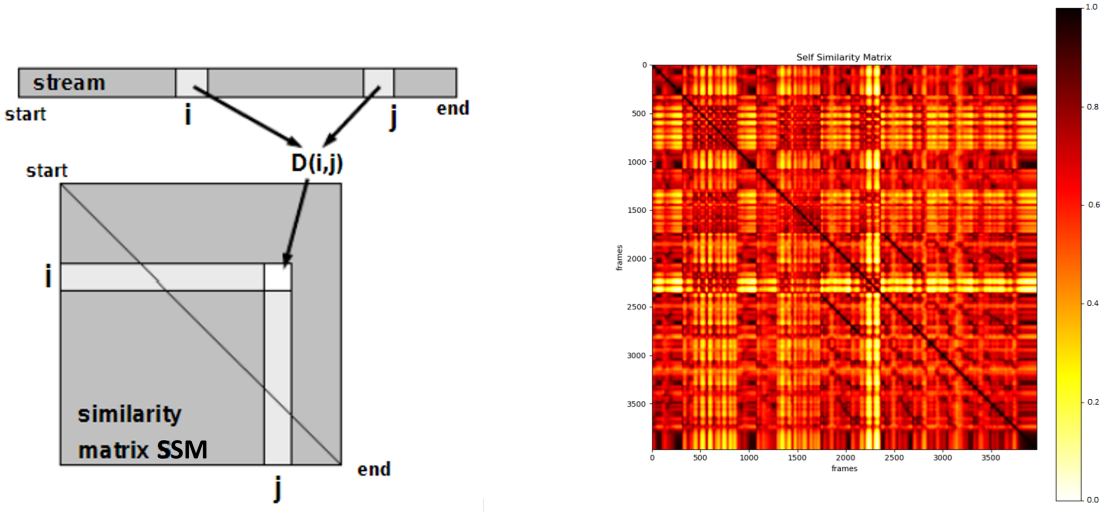
In practice, an SSM can be useful to obtain an overview of the parts of a piece that recur at least once, and the degree of similarity between all frames. The process to calculate an SSM is as described below [10].

> Being the B-dimensional spectral data computed for the N frames of a digital audio file (as described in sections 3.1.1 and 3.1.2) represented by the vectors $\{v_i : i = 1, ..., N\} \subset \mathbb{R}^B$.
> The SSM matrix, normalized such that its maximum value is 1, which, along with its symmetrical properties, can be characterized as $S(i, i) = 1$ and $S(i, j) = S(j, i)\ \forall i, j \in [1 : N]$, can be then calculated as follows.

$$\mathbf{SSM}[\mathbf{i}, \mathbf{j}] = 1 - d(v_i, v_j); \tag{1}$$

> Where d in (1) is a distance metric (further detail in section 10.3), and v is the feature vector (as the MFCCs or HPCPs described in sections 3.1.1 or 3.1.2).

A representation of the result obtained after calculating the described method to obtain a SSM can be observed in figure 9b, where can observed the diagonal stripes pointing out area of similarity, and therefore, possibly high-level structures.



(a) SSM structure illustration

(b) SSM calculated from MFCC features

Figure 9: Normalized Self Similarity Matrix representation, where

The decision on which distance metric to use depends on what is the intended use for the SSM, as well as what feature extraction technique is being used, therefore a few options should be taken into study.

For a detailed explanation on the various distance measures that will be considered in this work, please see section 10.3 in annex.

3.2.2 Recurrence Matrix

Essentially the Recurrence Matrix [11] can be considered a specific case of the more generic SSM (see section 3.2.1), but in this case the K-neighbours technique is used and the matrix is filled with binary values only instead of using distances to express similarities.

> This Recurrence Matrix (from now on referred as RM) consists of a square matrix RM whose elements $R_{i,j}$ indicate pairwise resemblance between the frames composing the analysed characterization, at times i and j. Formally, for the same B-dimensional vector v_i already described in section 3.2.1, taking $\{v_i : i = 1, ..., N\}$, it is set:
>
> $$\mathbf{RM_{i,j}} = \theta(\epsilon_{i,j} - ||v_i, v_j||); \qquad \text{for} \quad \forall i, j \in [1 : N] \tag{2}$$
>
> Where $\theta(z)$ in (2) represents the Heaviside step function (yielding 1 if $z > 0$ and 0 otherwise), $||v_i, v_j||$ can be any norm, even though usually the Euclidean norm is used, and $\epsilon_{i,j}$ describes a suitable threshold for each cell (i, j), which can be dynamically computed as follows.
> For each frame $v_i, \forall i \in [1, N]$, its K neighbors $v_j, \forall j \in [1, N]$ are searched. Then, mutuality between frames is forced by setting $R_{i,j} = 1$ only if v_i is neighbour of v_j and, at the same time, v_j is a neighbor of v_i.

This method is usually found to be more robust against noise than other variants such as the already mentioned SSM, mainly because of its more restrictive strategy. Still, it has the downside that this representation suppresses relevant information. This makes us aware of the need to seek balance between robustness against noise (where MR excels), and the ability to reflect all available information (as is the case with SSM).

3.3 Novelty Indicators

Once the mathematical description of the recording we are working with has been constituted, which allows us to explore the existence of high-level structures, it is needed to study how to quantify the evolution of the studied piece in the temporal domain.

The classic and more broadly used approach to quantify the temporal-evolution and identify boundaries, is to apply a Checkerboard Kernel, as described below, over the main diagonal of the used used matrix, thus obtaining a Novelty Curve from which the boundaries can be extracted by identifying its more prominent peaks [9] [12].

3.3.1 Checkerboard Kernel

When working with any of the mentioned Structure Discovery Analysis (see section 3.2), it is required to be able to locate the instant when the change is happening. In order to accomplish that, the so-called Checkerboard Kernel is correlated with the matrix we are working with [9] [13].

The simplest way to illustrate the logical function of this kernel, is by demonstrating its behaviour in a 2×2 dimension kernel.

$$\mathbf{C} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; \tag{3}$$

Where C in (3) represents the checkerboard kernel, which can be decomposed into coherence and anti-coherence kernels. The coherence term, being the first one, measures the self-similarity on either side of the center point; thus being higher when the regions are self-similar. The anti-coherence term measures the cross-similarity between two regions; this will be high when the regions are substantially similar, thus with little difference across the center point.
The difference of the two decomposed terms works as a novelty estimator of the signal. This will take maximum values when the two regions are self-similar but different from each other, therefore detecting what is considered a transition (boundary between segments).

Also, it is possible to construct larger kernels by means of the Kronecker product ($\otimes$) of C with a matrix of ones, as seen in Equation 4.

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix}; \tag{4}$$

This Kernel can be smoother to avoid edge distortion using windows to taper the edges towards zero. This is usually done by using a radially-symmetric Gaussian function.

> This smoothing factor can be obtained by multiplying two Gaussian windows, let them be called g_1 and g_2, with sizes s_1 and s_2, which defines the amount of past and future features being taken into account (and it should be adjusted depending on the music piece to be analyzed).

$$\mathbf{G} = g_1 g_2^T; \tag{5}$$

In figure 10, a 3D representation of the kernel after being smoothed with Gaussian tapering can be observed.

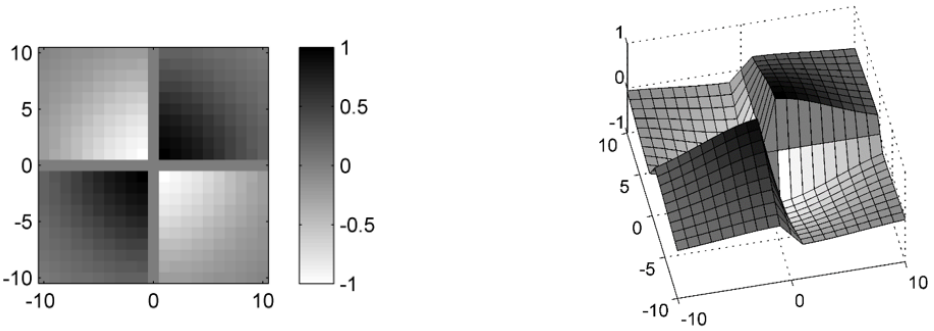


Figure 10: Checkerboard kernel with Gaussian taper

3.3.2 Novelty Curve

Correlating the Checkerboard Kernel (see section 3.3.1) with the similarity matrix (see section 3.2) results in obtaining of a measure of novelty. The representation of this measure of novelty in a time axis is referred as Novelty curve.
When correlating the kernel with the similarity matrix, if the kernel C is over a relatively uniform region, the coherence and anti-coherence regions will tend to add up to zero; by contrast, when C is positioned in an irregular region the overall sum will be larger. Thus, obtaining a time-aligned measure of novelty (Novelty curve) in the recording.

> Being M the similarity matrix we are working with, the Novelty curve for this can be obtained as described below [13].

$$\mathbf{N(i)} = \sum_{m=-L/2}^{L/2} \sum_{n=-L/2}^{L/2} C(m,n)M(i+m,i+n); \quad \text{for} \quad \forall i \in [L+1 : N-L] \tag{6}$$

> Being L the width (lag) of the kernel, and being $N(i)$ in (6) the representation of the novelty frame-aligned vector (where i represents the frame-index). It is important to notice that, because of how L is defined, the correlation is only computed for the interior of the signal where the kernel overlaps M completely.

Lastly, it is important to mention that the width L of the kernel C directly affects the properties of the novelty measure and its estimations. A smaller kernel would only measure novelty on shorter time periods, such as individual musical notes. Instead, wider kernels are able to detect high-level structures such as musical transitions, key modulations, or symphonic movements. Thus, it is obvious to conclude that the value of L will not be trivial, and shall be thoroughly studied.

A representation of the novelty curve can be observed in figure 11.

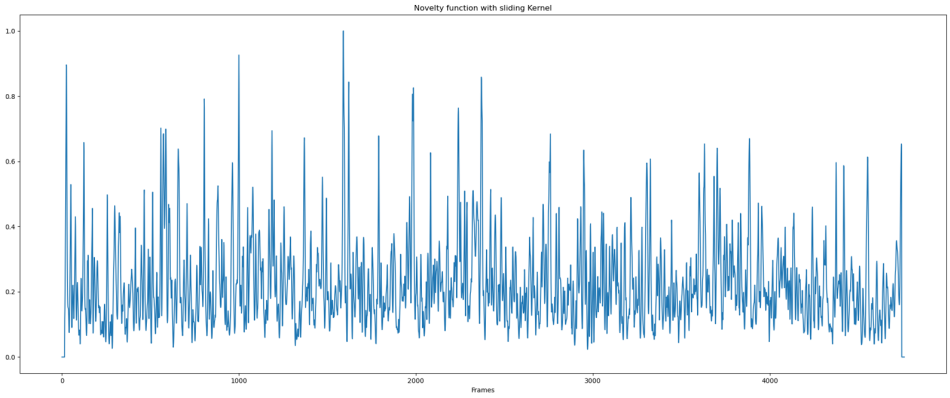


Figure 11: Novelty curve representation

3.4 Peak Detection Algorithm

Once the Novelty curve is obtained, it is still necessary to design a method to decide which peaks should be considered an estimation for a boundary, and which ones should be dismissed.
This difficulty can be approached from many perspectives with different levels of complexity. The approach used in this thesis, is to consider a boundary the peaks whose score exceeds a threshold function, which is calculated according to the processed signal, and normalizing the value such that the maximum score is one and the minimum is zero.

During the implementation, different algorithms have been tested, so the details of the different methods studied will be addressed in the implementation section (see section 5.2).

4 The Segmentation Problem as a Binary Classification Problem

In order to be able to evaluate the different solutions studied and proposed in this thesis, as well as draw consistent conclusions across models, it was considered necessary to design and implement an evaluation model. To approach this need, it was decided by the author to evaluate this problem as a binary classification problem, therefore relying on classical metrics to evaluate the different classification schemes.

4.1 Confusion Matrix and Qualitative Evaluation

In order to evaluate and analyse the results of an algorithm in the framework of binary classification problems, it must be considered that the outputs of the algorithm are predictions, and therefore that these can be considered either correct or incorrect predictions. For this purpose, the use of a confusion matrix is usually resorted to.

A confusion matrix [14] is a matrix with two dimensions ('real' and 'predicted'), and an identical set of parameters on both dimensions. This way, it is easy to represent whether a prediction is either correct or the result of confusion (positive versus negative).

Total Population P + N		Predicted Condition	
		Positive (PP)	Negative (PN)
Real Condition	Positive (P)	True Positive (TP)	False Positive (FP)
	Negative (N)	False Negative (FN)	True Negative (TN)

Table 2: Confusion Matrix

In our specific case, it allows us to determinate either if a boundary classification is correct (True Positive) or if it is a mistake (False negative).

4.2 Dataset Curation

In order to evaluate the prediction obtained, a dataset on which to run the different versions of the code is needed. To solve this need, and given that there are few live performance recordings of the pianist we are working with, and for whom we want the segmentation algorithm to be optimised, it is concluded that the most suitable solution is to create a dataset of synthetic annotations.

For the creation of this synthetic dataset, it was decided to take the different recordings available (a set of 12 tracks of variable length), and by means of an automated script, segment all these tracks into fragments of a given duration, to then join them all together in a single sequence, thus constituting a single audio with a set of synthetic boundaries.

Subsequently, and once the synthetic track was obtained, human listening was carried out in order to complete the automatic annotations with those organic boundaries that could be found in the initial tracks. That is to say, all those organic transitions that could be found in the initial tracks, and that should also be segmented.

This set of boundaries (both synthetic and organic) were then annotated, for later use in the evaluation model, as a binary vector composed of segments of 250ms duration, in which each of these segments are annotated as True if they contain a boundary, or False if they contain a homogeneity period. The use of 250 ms periods was chosen so that boundaries would be centred on these intervals, thus maximising the correctness of the evaluations.

4.3 Evaluation Model

For the evaluation model it was decided to opt for a binary detection model (as described above), based on the evaluation of predictions as correct or incorrect relying on classical metrics. Once the predictions of the implemented algorithm in question are obtained, in order to evaluate and discern whether each of the predictions is correct or incorrect (True Positive or False Positive), it is necessary to also model the vector of predictions following the same structure as described in the previous section. This is as a binary vector composed of segments of 250ms duration, in which each of these segments are annotated as True if they contain the prediction for a boundary, or False if they contain the prediction for a homogeneity segment.

Once these two binary vectors are available, the computation of the different components of the confusion matrix can be obtained by means of the calculation of logical operations (see section 4.1).

With the values composing the confusion matrix, it is possible to compute different classical metrics which can be used to quantify the efficiency and effectiveness of the different iterations of the algorithm. After reviewing the available literature on this matter [15], it was decided to use the F-score as it balances the precision and the recall and allows easy identification of the optimum parameter value, as described in (7).

$$F_{score} = \frac{2TP}{2TP + FP + FN}; \tag{7}$$

5 Offline Music Segmentation

As a first approach to this complex problem, and with the intention of gaining some comfort with the mathematical tools involved in this development, it was decided to select the first algorithm without taking into consideration the requirement of it being capable of running in real time.
For this reason, the author (advised by the thesis director) decided to base the first implementation on one of the most relevant papers in this discipline, dealing with an algorithm that, even after two decades, continues to be referenced in the most recent publications and continues to show state-of-the-art results [16] [17].

The paper introduced and implemented in this section is 'Automatic Audio Segmentation Using A Measure of Audio Novelty' by Jonathan Foote, published in 2000 in the framework of the 2000th IEEE International Conference on Multimedia. This publication introduces a method of estimating the instantaneous audio novelty by analyzing frame-to-frame similarity. The ground rules for detecting significantly novel points, which therefore can be considered predictions for a boundary, is that those will have high self-similarity in the past and future, but low cross-similarity. It is also important to notice that this method presents a very adjustable model, since the extent of the "past" and "future" can be adjusted in order to change the scale of the analysis.

5.1 Proposed Algorithm

The backbone structure presented in the publication can be defined as follows:

Algorithm 1 Foote Automatic Audio Segmentation Algorithm

Require: : $\mathcal{X}, Sr, \mathcal{V}, \mathcal{D}, \mathcal{S}, Novelty_c$
Ensure: : $\mathcal{X} = [x_1, ..., x_N]$ with $N > 0$ and $Sr = 44.1kHz$
for $\forall x_n \in \mathcal{X}$ **do**
    Each frame is tapered with a Hamming window (9)
    $\mathcal{X}' \longleftarrow FFT(\mathcal{X})$
    $\mathcal{V} \longleftarrow FeatureDescriptor(\mathcal{X}')$
for $\forall v_i \in \mathcal{V}$ **do**
    for $\forall v_j \in \mathcal{V}$ **do**
        $\mathcal{S}[i,j] = \mathcal{D}(v_i, v_j)$
$Novelty_c \longleftarrow$ Correlation described in equation (6)
OUTPUT $PeakSelection(Novelty_c)$

Being $\mathcal{X}$ in Algorithm 1 a time-series input containing the music signal studied, Sr the sample rate of said recording and $\mathcal{D}$ a distance metric such as 10.3.
Also V being the vector with the signal characterisation, and $Novelty$ the novelty function calculated by the algorithm.

5.2 Implementation

For the implementation of this first algorithm, the author decided to use the programming language Python, mainly due to the ease that this programming language presents when it comes to being read and adapted to other programming languages, thus making the replicability and revision of this project easier.

Python offers innumerable packages in which computationally efficient implementations of almost all the mathematical tools necessary for this development can be found. Going into further detail regarding the packages used for the implementation of this algorithm, it is worth mentioning NumPy[2] and SCIPY[3], both being common packages for mathematical programming, and Librosa[4] [18], a Python package specially designed for music information retrieval systems and audio analysis.
Lastly, and this proved to be an important factor in the development of the project, python offers many cloud execution platforms, which facilitates the execution of scripts that require long and computationally expensive executions. In this work, most of the development has been carried out on Google Colab, so that the hardware that was available for the author was not a limiting factor for the implementations.

It is important to mention that despite taking the 'idea' for the algorithm from a published paper, this only offered a non-exhaustive mathematical description of the algorithm, therefore having to take decisions and make novel implementations to circumvent the gaps in the paper.

An example of these novel implementations is the peak detection algorithm that was implemented, which is explained in more detail in the following sections of this section.

5.2.1 New Peak Detection Algorithm

The main improvement that was made to the proposed algorithm, apart from slight adjustments necessary for its implementation, and parts that were implemented freely due to the lack of detail in the publication, is the prediction decision block, in which it is selected which peaks of the novelty curve should be considered as predictions of a boundary. The technique originally proposed in Foote's paper consisted of an excessively simple method, which was based on considering as boundary the local maximum within a given time elongation segment. In this way, it did not take into account cases where there could be a rapid succession of segments, or on the contrary, areas of greater homogeneity in which there are no transitions to be detected. Because of these factors, the author worked to improve this aspect and implement a new peak selection block.

The design of this new algorithm was based on the homogeneity principle already described in this thesis, according to which a transition point between two segments can be described as a point where adjacent zones have a high degree of self-similarity, but have a very low score on cross-similarity between them.

[2]For more information about the package - https://numpy.org/
[3]For more information about the package - https://scipy.org/
[4]For more information about the package - https://librosa.org/doc/latest/index.html

Following this rule, an algorithm was implemented that calculates a threshold function by applying a median filter in the form of a sliding window of size F on the novelty curve, which is then adjusted by adding the median value of the novelty curve multiplied by an offset, (these are optimisable values).
The general structure of this algorithm is detailed below.

Algorithm 2 Peak Detection Algorithm

```
for Frame in NC vector do
    if NC[Frame - 1] < NC[Frame] < NC[Frame + 1] then
        It is considered to be a local max for this function.
        if NC[Frame] > threshold[Frame] then
            It is consided a true peak, and therefore a boundary prediction.
            BoundaryList ⟵ append(Frame)
OUTPUT BoundaryList Estimation
```

5.3 Optimization of the Algorithm

In order to be able to adjust the different parameters involved in this implementation to their optimal value, an automatic optimization process was deemed necessary. This step can also be considered an addendum to Foote's initial paper (Algorithm 1), and to the current trends in this line of research, since usually (or as far as the author is aware of from the various papers studied during the course of this thesis) the parameters are usually adjusted manually and without further consideration of the effect they may have on the results obtained.

To carry out this optimization, first it is necessary to narrow down which parameters should be optimized:

- **FFT size:** Which determines the size of the computed FFT (see section 10.1.1.3), thus also conditioning the size of the frames.
- **Mel Filter-Bank size:** This parameter is what determines into how many bands is the Mel spectrum segmented (see section 10.1.1.4). Thus, it is also the number of filters is the Mel Filter-Bank. This affects the resolution of the computed MFCCs.
- **Number of Cepstral Coefficients computed:** As the name indicates, it determines the number of coefficients to calculate for each frame (see section 10.1.1.5).
- **Checkerboard Kernel size:** This determines the size of the kernel (see section 3.3.1).
- **Threshold Offset:** This parameter is a float that acts as a multiplier parameter for the median value of the novelty curve, in the calculation of the threshold function.
- **Threshold Median Filter size:**This variable explicitly determines the size of the Median filter applied when computing the threshold function.

Once the metrics for quantifying the optimization process were defined, and counting with a robust evaluative model (see section 4.3), a suitable optimisation strategy for the described algorithm had to be defined and implemented.

This algorithm consists of an iterative optimisation of individual parameters, in which initially all parameters are instantiated with a random value (or pseudo-random value within a given interval), and from there with those values set, the different parameters are optimised one by one.

To illustrate this process, once the first parameter has been iterated for all the cases considered, the value considered optimal (the one that has achieved the highest F-score) is selected, and this value is fixed for the parameter in the optimisation of all following variables. This same process is repeated for all other parameters to be optimised, until the first optimisation cycle is finished. Once this first round is finished, all parameters are set to the optimal value found so far.

At this point, and before starting the next optimisation cycle, the range of values to be tested for each parameter is narrowed down and centered to the maximum value found in the previous cycle.
This process is then repeated until the maximum F-score achieved at the end of a cycle remains stable.

This algorithm is described in more detail below.

Algorithm 3 Multi-Parameter Optimization Algorithm

```
1: while F-score[end] > F-score[end - 1] do
2:     for Parameter in ParametersList do
3:         for value in RangeOfValues do
4:             The Algorithm runs with Parameter = ParameterRangeOfValues[value]
5:             Fscore ⟵ evaluation of this prediction
6:             Results ⟵ append(Fscore)
7:         Parameter ⟵ max(Results)                ▷ It's set to the optimal value
8:     After the a round of optimization, each variable is set to its local optimal value
9:     RangeOfValues ⟵ bounded around the maximum found
10: OUTPUT Optimal Values for ParametersList
```

5.4 Results and Analysis

The first result to be commented is the one obtained once the segmentation algorithm is executed. Once the piece of music to be segmented is loaded, and after executing the code, the output that can be observed in Figure 12 is obtained.

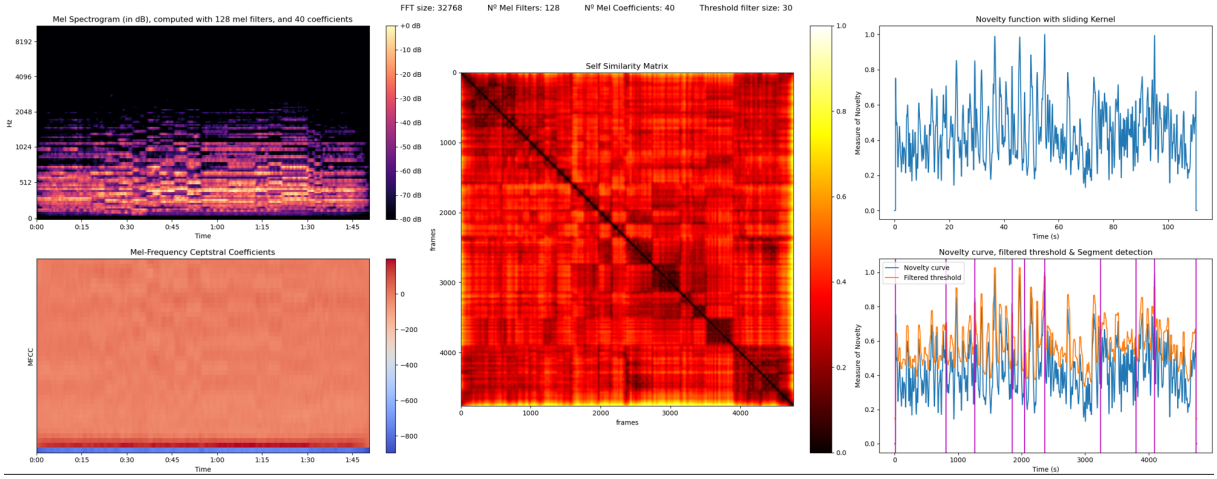


Figure 12: Output when running the final solution for this implementation

In this view it can be observed a set of plots and values, which represent the set of steps followed in the execution.
At the top left, it can be seen a plot containing the calculated Mel Spectogram, and below it, and still on the left, a graph representing the vector of cepstral coefficients obtained by the MFCC characterisation technique.
In the centre of said panel, there is the graphical representation of the computed SSM. Finally, on the right side of this view it can be observed, at the top, the graphic for the Novelty Curve, and at the bottom, a graph containing different plotted functions. In this last graph, it can be observed in blue the same Novelty Curve already seen in the upper plot, but this time accompanied in orange by the calculated Threshold, all together with the predicted Boundaries in purple. This is seen in more detail on Figure 14.

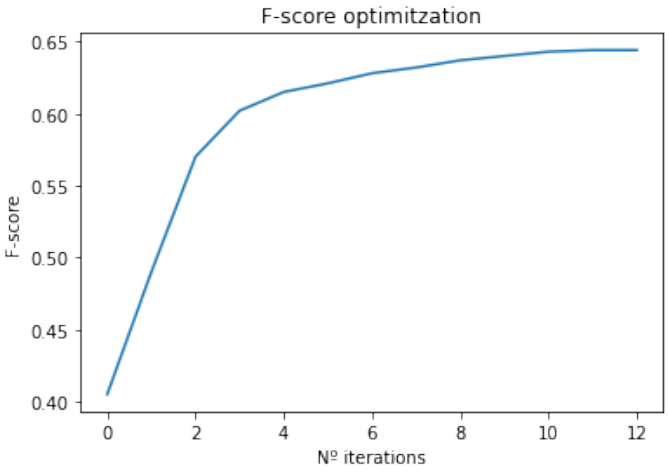


Figure 13: F-score Optimization

By executing the optimization algorithm (see Algorithm 3), it was possible to optimize the different parameters that act as variables in our code. It can be seen in Figure 13 the evolution of the F-score through out the different cycles of optimization.

It should be noted that this is not an exhaustive optimization algorithm, so there are no guarantees that the observed convergence corresponds to the absolute maximum. Even so, the results obtained can be considered conclusive, since the optimisation algorithm was run in different cycles with different random initialization values as a starting point, and always ending at the same maximum, therefore concluding that there is no reason to believe that a better F-score can be obtained through optimization.

The final values for all the optimized parameters, obtained through out this optimization process are:

$$
\begin{aligned}
FFT_{size} &= 8192; \\
k_{MelFilters} &= 40; \\
C_{CepstralCoefficients} &= 20; \\
Kernel_{size} &= 256; \\
Offset &= 0.45; \\
MedianFilter_{size} &= 83;
\end{aligned}
$$

Running the algorithm using this set of parameters, which are the optimal parameters, it is obtained the prediction observed in Figure 14. This prediction scores a maximum F-score value of 0.644068, with a correct detection of 41 out of 61 true boundaries (True Positives) and 1079 out of the 1099 homogeneous segments (True Negatives), therefore detecting 67.213% of all boundaries, and scoring a precision of 0.655.

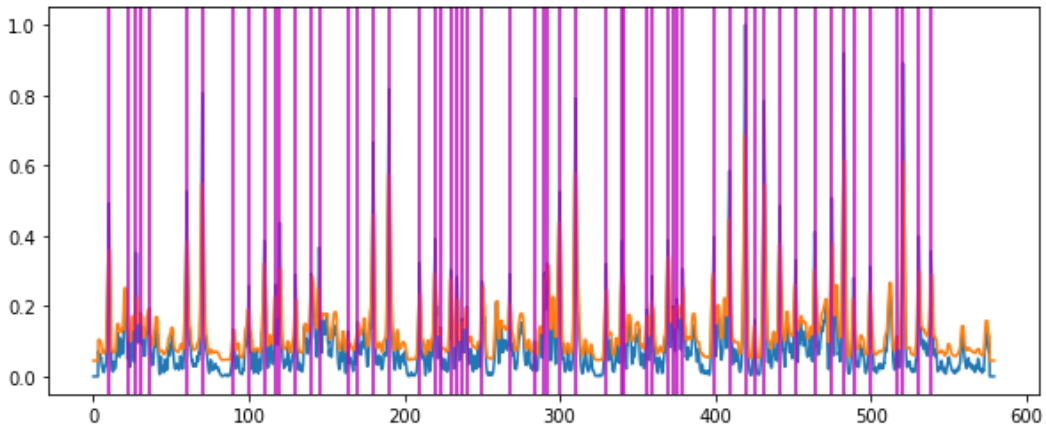


Figure 14: Optimized Prediction

5.5 Discussion

From the results obtained in this section, several conclusions can be obtained.

First of all, it can be observed that the implementation presented in this section works remarkably well, managing to segment the signal in a way that is considerably similar to what a human listener would do.
This conclusion can be drawn from the F-score measure, which is rather good, but it is also reinforced by human listening to the results of the segmentation algorithm. A review of the segmentation proposed by the algorithm shows that there is a certain coherence in the results, where the most marked transitions are almost always detected as a boundary, and where even certain motifs that are repeated throughout the track are detected and segmented repeatedly.

In this aspect, and taking as a reference point the available literature on this type of implementation, it is worth mentioning that the results obtained (without having carried out a rigorous comparison, it should be noted) are quite in line with current trends.

It is also concluded that the design, implementation and consolidation of both an evaluation model and an optimization strategy for the various parameters on which the algorithm depends, have been achieved. Both models have been tested experimentally, proving their correct behaviour, as well as their versatility in adapting to different formats.

With these conclusions the next implementation is started, which takes into consideration the requirement that the offered solution must be one capable of running in real time, therefore capable of processing the audio and offering predictions of its segmentation in real time.

6 Real-time Music Segmentation

In this second implementation, the main focus of attention was to make it work in real time, so that as the music was 'listened to', the same signal was segmented. To achieve this, new factors such as computation time and decision delay had to be taken into consideration. Going into further detail, in the field of real-time processing it is necessary to distinguish two notions; First, the CPU load. In average the computer should be able to process the incoming sample as fast as it is 'read'. This is generally what is referred as real-time processing. If your sampling frequency is Fm, your CPU should allow you to process Fm sample per second.
A second (and different issue) is the delay in which you will be able to generate an answer, and to decide whether or not there was a detection. This is referred as decision delay, since it is not only dependant on the CPU times, but also in the complexity of the process.

These factors had to be taken into consideration, with the code-level optimisations this required, as well as a modelling of the behaviour of the algorithms when constrained to a system with finite memory.

In order to address the requirement to implement a real-time segmentation model, the author was hampered by the fact that there are currently not a large number of publications that study music segmentation in the specific context of real-time processing, and of these, most of them rely on neural network-based models or machine learning models. As these disciplines are outside the intended scope of this thesis, it was necessary to look for a classical signal processing oriented alternative.

For this purpose, it was decided to build an implementation based on the paper "Real-time unsupervised music structural segmentation using dynamic descriptors" [19], published as an academic paper by the university of São Paulo. The segmentation model described in this paper consists on an algorithm based on the study of homogeneity and dissimilarity of the audio studied, but presents considerable differences with respect to the paper used as a reference for the first implementation.
The implementation described in another paper [5] was also followed, adapting it so that it could work in real time. However, this implementation did not reach the optimisation stage due to poor early results.

6.1 Proposed Algorithm

The models described in both papers used as starting point for these implementations, broadly speaking, consist of the following steps.
First of all, and once the signal has been 'listened', it is necessary to proceed to extract the feature descriptors (in this case, dealing with dynamic descriptors, since it is a real-time implementation). Specifically, in both implementations contemplated in this thesis, different descriptors have been used. In the first one the already used MFCCs were extracted, since they already showed a great performance in signal characterization. In the second implementation, and with the aim to confirm the initial intuition that MFCCs were more suitable signal descriptors than other options based on chromatic structures, HPCPs were

used. Once these descriptors have been extracted, and therefore have a parameterisation of the music we are dealing with is available, it is proper to proceed to constitute the structure discovery matrix, which can vary. Again, different structures were used in both implementations, in the first one the already mentioned SSM was used, and in the second one the Recurrence Matrix, with its corresponding Lag Matrix (see section 10.2.1 in annex), were used.

Once the structure exploration matrix is constituted, the novelty curve can be computed, therefore, and once this function is available, the only thing left to do being to select and obtain the predictions for the segmentation by means of a peak detection algorithm. For this last step, two different peak detection algorithms have been used, mainly with the intention of, once again, verifying that the decisions made by the author were indeed the right ones, and thus, discarding possible alternatives.

A broad description for the algorithms implemented can be seen below.

Algorithm 4 Real-Time Structural Segmentation

Require: : $I; Inoutdata, w_{min}, w_{max}$
Ensure: : $S; boundarylocations, L; labelledsections$

$p \longleftarrow 1, T \longleftarrow \{\}, x \longleftarrow read(I)$
while $x \neq NULL$ **do**
  $T \longleftarrow T \cup \{x\}$
  if $|T| + 1 \geq w_{min}$ **then**
    $relative \longleftarrow LocateChangePoint(T)$
    if $relative = NULL$ **then**
      if $|T| + 1 > w_{min}$ **then**
        $T \longleftarrow \{t_i; t_i \in T; i \geq w_{min}\}$
    else
      $absolute \longleftarrow (p-1) - |T| + relative$
      $T \longleftarrow \{t_i; t_i \in T; i > relative\}$
      $S \longleftarrow S \cup absolute$
    $p \longleftarrow p + 1$
  $L \longleftarrow Label(S), x \longleftarrow read(I)$

6.2 Implementation of the Real-Time Solution

Once again, these implementations have been carried out from the ground up, and without any previous basis, by the author, developing the code using Python as the main programming language.
The aforementioned papers were taken as a starting point, but in the author's implementations modifications were made with the intention of solving problems or improving efficiency.

During the development of this real time solution, some challenges had to be faced. The main challenge that characterises the real-time implementation as opposed to the previous implementation is that, being a model that requires processing the signal as it listens to it, processing times must be taken into consideration, because if these times are excessively long, delays are formed that hinder real-time processing.

To solve this situation, the author models the problem by means of two variables called t_{step}, which refers to how many seconds of music should be 'listened to' before making the prediction recalculation, and the variable L_{past}, which determines the number of seconds prior to the current instant that are taken into consideration for processing at any given time.

These two parameters have a relationship, since the greater the value of the parameter L_{past}, the greater the number of data-points to be calculated at any given time, and therefore the greater the processing time, in turn conditioning how small t_{step} can be. This relationship can be illustrated as seen in equation 8.

It is important to notice that the parameter t_{step} also governs with how many seconds of delay the predictions are accepted. To illustrate this idea, if for example the t_{step} takes a value of 5 seconds, this implies that every 5 seconds of music listened the algorithm will recalculate and new predictions will be made, but it also implies that these predictions can be found in those last 5 seconds listened, thus implying a delay in the predictions of up to 5 seconds (the size of the t_{step}).

$$t_{step} \geq t_{decision}(L_{past}); \tag{8}$$

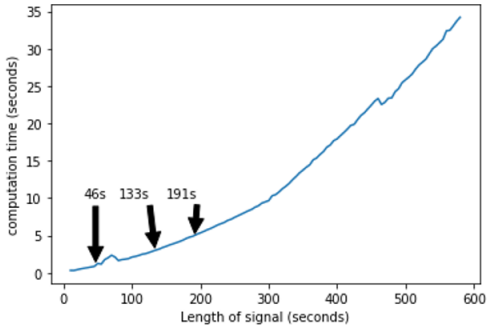
It is also worth mentioning that the processing time is considerably affected by the distance metric used (see section 10.3), so this parameter has to be studied and limited in line with the others already described.

For the second alternative algorithm implementation, in order to confirm the initial hypothesis that MFCC was the most appropriate choice for the type of music we are trying to segment, and to also explore possible ways to improve the effectiveness and efficiency of the different algorithms implemented, it was decided to use HPCP (see section 3.1.2) descriptors to characterise the signal. Taking advantage of this second independent implementation, we took the opportunity to test different peak detection algorithms as well as other tools for structure discovery, as already mentioned. However, due to the results described in section 6.4, this second implementation was not taken to the optimisation stage due to poorer than expected results.

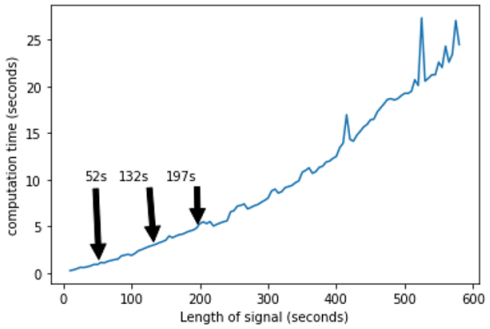
6.3 Optimization

The same optimization model already described in section 4 was used in this implementation, as it proved to be consistent and robust, so it was considered that it did not need to be modified beyond the necessary adaptations to make it compatible with this new implementation.

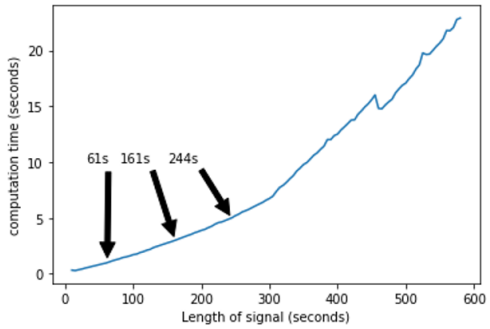
For this new implementation, the main focus of the optimisation process is the relationship described in Equation 8. Therefore, the first condition to study in this relationship is the effect of the different metrics on the processing time. For this study, the algorithm was tested with numerous different metrics, which were obtained from different papers, to then be compared and observe what effect they had.



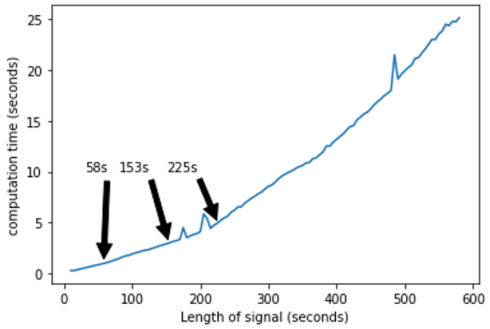
(a) Canberra Distance (see Equation 20)



(b) Euclidean Distance



(c) Correlation Distance (see Equation 22)



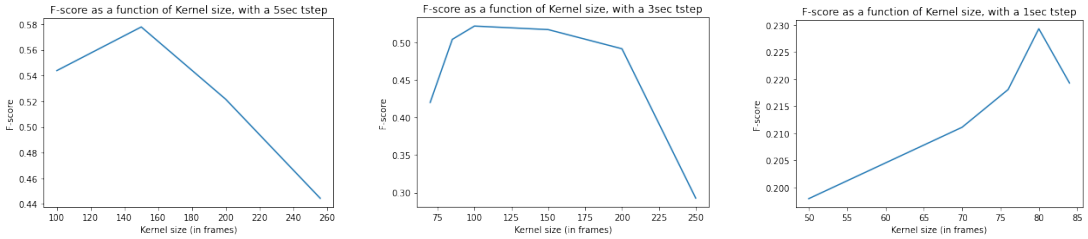
(d) Cosine Distance (see Equation 21)

Figure 15: Evolution of the Decision Delay as a function of the length of the calibrated sample, for different distance metrics.

In Figure 15 it can be observed the graphs representing the sample lengths corresponding to 1, 3 and 5 seconds of decision delay. These are also the different t_{step} which will be used later.

It was also necessary to study the effect of the Kernel size. As explained in section 3.3.1, the size of the kernel influences what kind of structures are detected, but also given how it is implemented, it can make it impossible to predict boundaries at the ends of the SSM diagonal. Therefore, the effect of modifying this parameter on the F-score should also be studied.

For this purpose, the exact relationship between the F-score (in the different t_{step}) and the kernel size was experimentally studied, thus making it possible to conclude what can be seen below.



(a) 5 seconds t_{step} (b) 3 seconds t_{step} (c) 1 seconds t_{step}

Figure 16: Evolution of F-score as a function of the Kernel size, according to the t_{step}.

Here it can be observed that there is a clear relation between kernel size and F-score, but that this is a function of the t_{step} used.
Therefore, it can be concluded that depending on the t_{step} being used, there is an optimal value for the kernel size to be used, and that depending on this the predictions can be improved. It can also be clearly observed that if kernel too large is used, predictions are no longer made in the time interval comprising the t_{step} used, and similarly if a kernel size too small is used, the sensitivity decreases and the predictions become too abundant, thus lowering the F-score.

6.4 Results and Analysis

The first result to be addressed is the view that is generated during the execution of this real-time solution. This can be seen in Figure 17.
In this view it can seen how after each new t_{step} seconds of the listened music, the algorithm computes new predictions, and in case a boundary prediction is made in the last t_{step} seconds of the sample, this boundary is printed on the screen as shown in the figure below, therefore indicating the segmentation in the last few seconds of listened music.
This way, it is possible to make a real time use of the segmentation algorithm, obtaining the information in real time according to the parameters with which it has been configured.

```
Length analyzed: 69.0    Time location: 69       Execution time: 0.7809851169586182

Length analyzed: 70.0    Time location: 70       Execution time: 0.7555372714996338

Length analyzed: 71.0    Time location: 71       Execution time: 0.7661635875701904
Boundrie estimatation at: 70.02485284499673sec
```

Figure 17: Real time execution, detecting a boundary using a t_{step} of 1 second

It is also possible to obtain a representation of the final segmentation performed by the algorithm, thus obtaining a global view of the proposed segmentation. This can be observed below.

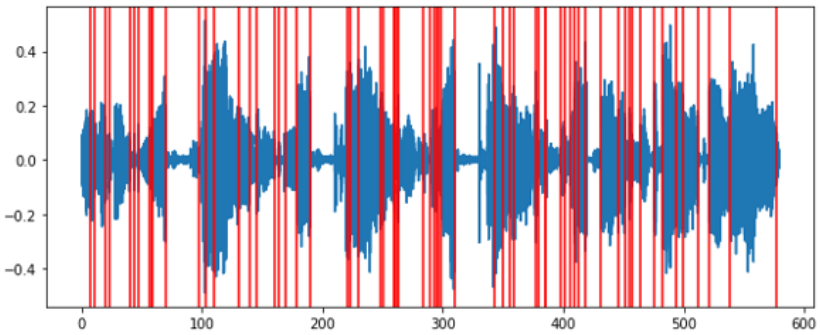


Figure 18: Final view of a real time execution

The second result to be discussed is the one shown in Figure 19. In it, it can be seen the different F-score obtained for the different implementations, and according to the t_{step} used.

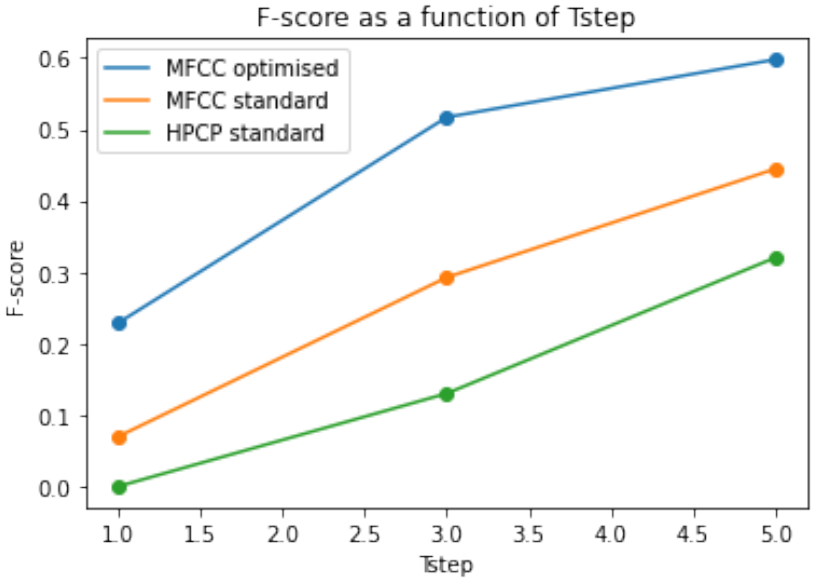


Figure 19: F-score as a function of t_{step}, for different implementations

In this graph, it can be seen; in green the F-score obtained with the standard configuration (this consists of initialising all the parameters with the values obtained in the optimisation of the offline solution) of the HPCP implementation, for the different t_{step} of 1, 3 and 5 seconds. It is worth mentioning that in this implementation the values are significantly lower than in the others, which is coherent with the initial hypothesis that the HPCP feature descriptors are worse given the style of music dealt with.
Also, in orange, it can be seen the F-score for the same t_{steps} of the real-time implementation using MFCC in its standard configuration, and in blue the same for its optimised version. It is worth highlighting that the higher the t_{steps}, the higher the effectiveness of the segmentation algorithm. This is due to the relationship described in Equation 8, which describes that the smaller the t_{steps}, the smaller the memory taken into considera-

tion when recalculating the predictions for each new sample, which consequently hinders the good performance of the algorithm.
It is also remarkable the improvement observed once the optimisation process has been applied, reaching the most optimal case studied to obtain an F-score of 0.597.

6.5 Discussion

None of the different results obtained in this section are unexpectedly new, or in other words, they are all aligned with the initial hypothesis.

It has been confirmed that indeed, given the type of music dealt with, the most suitable feature extractors to use were MFCC, since it was observed that actually HPCP gives significantly worse predictions.

It has also been shown that the algorithm working in real time, and therefore having restrictions on the size of the memory (the length of the music sample size to be used), suffers from a decrease in the performance of the algorithm. This can be clearly observed in that, even after subjecting the algorithm in real time to the proper optimisation process, the F-scores obtained are still lower than those obtained in the offline execution. This result was to be expected and is consistent with what was expected, so it can be extrapolated that it is not a consequence of our algorithm, but of the very nature of what is a real-time execution.

In summary, it can be concluded that although the results are lower than those observed in the section on non-real-time implementation, the results obtained are consistent and within expectations, thus being positive results.

7 Conclusions

As a final conclusion of this work, it can be concluded that indeed, in the field of Real-Time Music Segmentation, framed in the field of audio-based Music Structure Analysis (MSA), there is still much room for the developments of new techniques, as well as improvements on existing models, without having to resort to machine learning solutions.

It has also been observed in this thesis how by using classical signal processing algorithms and techniques, and at the same time improving and complementing some of the techniques already described in other publications, remarkably good results can be obtained.

It is also worth highlighting the different results obtained, which are summarised in the following table.

Implementation	**F-score**	**% correct predictions**	**Precision**
Offline Optimised Segmentation	0.644	67.21 %	0.655
HPCP Standard Real-Time Solution	0.323	26.67 %	0.400
MFCC Standard Real-Time Solution	0.444	43.30 %	0.448
MFCC Optimised Real-Time Solution	0.597	65.00 %	0.513

Table 3: Summary of the results from all implementations

It can be observed that, although the offline segmentation algorithm still performs better than the real-time version, the real-time version has been refined to come closer to the scores of the offline version. Thus, concluding that the main objective of the work, which was to develop and implement a real-time music segmentation solution, has been successfully achieved.

8 Future Work

The main line of improvement that this author considers that would be interesting to investigate, but that due to the limited time offered in a final degree thesis could not be carried out, would be the implementation of a mixed segmentation model in which the predictions made by different feature extracting techniques are taken into consideration at the same time.
That is, using MFCC and HPCP simultaneously, in such a way that either the processing scheme of both models is executed separately, and once the predictions of the two models are obtained separately, a final decision is taken according to a balancing of these two models by means of assigned statistical weights, or establishing the structure exploration matrix by constructing it by means of the distances calculated in both characterisation vectors, and balancing these distances also by means of assigned weights. In this way, a more robust model that can take into account more factors of the processed signal in question is likely to be achieved.
Of course, the weights assigned to each of the vectors or predictions of the various feature extraction techniques should be subjected to a rigorous optimisation process, otherwise the model could be undermined and perform worse than the current model.

9 Economic and Environmental Impact

9.1 Budget

This section will try to model and discuss the budget needed to develop a project of this scale.
For this purpose, it has been considered that the developer would earn 11€/hour and that the whole project needs around 585 hours in order to be completely finished. The number of hours dedicated to each part of the project can be seen in the following table.

Work Plan	**Required Hours**
WP1 : State of Art	95
WP2 : Algorithm selection	10
WP3 : Implementation	320
WP4 : Evaluation and optimization	90
WP5 : Final Solution	25
WP6 : Report	45
TOTAL	585

Table 4: Hours worked break down

After performing an analysis of the total hours dedicated to the project, a total of 585 hours have been obtained. In the following table the rest of the costs necessary to develop the project are discussed.

Concept	**Amount** (in eur)
Developer Salary	11€/h x 585h = 6435€
Computer	750€
Cloud Computing Platform	9.25€/month x 5 month = 46.25€
Electricity and Others	200€
TOTAL	7431.25€

Table 5: Costs break down

As detailed in Table 4, the estimated total budget required for the development of this project is 7431.25€.

9.2 Environmental Impact

In this section we will detail the environmental impact corresponding to the development of this project. Mainly, as this is a software project, the environmental impact corresponds to the CO2 emitted due to electricity consumption, which may or may not come from clean or renewable energy sources. The electricity consumption and the mass of CO_2 emitted into the atmosphere are detailed below.

Causer of the Consumption	**Hourly consumption**	**Total Consumption**	**CO_2 Consumption**
Computer	200 W	90 kWh	20.983 Kg of CO_2
Cloud Computing	300 W	55.5 kWh	0[5] Kg of CO_2
Others	150 W	87.75 kWh	20.458 Kg of CO_2
TOTAL	700 W	233.25 kWh	41.441 Kg of CO_2

Table 6: CO_2 consumption break down

We can see that the total emissions generated by this project would be around 41.5Kg of CO_2.

[3]Google Colab has a carbon footprint of zero
[4]Google Colab has a carbon footprint of zero
[5]Google Colab has a carbon footprint of zero

References

[1] Seungmin Rho Sanghoon Jun and Eenjun Hwang. *Music structure analysis using self-similarity matrix and two-stage categorization.* Multimedia Tools and Applications, 2013.

[2] M. F. McKinney M. J. Bruderer and A. Kohlrausch. *The perception of structural boundaries in melody lines of Western popular music.* Musicæ Scientiæ, 2009.

[3] J. Schlüter K. Ullrich and T. Grill. *Boundary detection in music structure analysis using convolutional neural networks.* 15th International Society for Music Information Retrieval Conference, 2014.

[4] Jouni Paulus and Anssi Klapuri. *Music Structure Analysis Using a Probabilistic Fitness Measure and a Greedy Search Algorithm.* IEEE Transactions on Audio, Speech, and Language Processing, 2009.

[5] Peter Grosche Joan Serrà, Meinard Müller and Josep Ll. Arcos. *Unsupervised Music Structure Annotation by Time Series Structure Features and Segment Similarity.* IEEE TRANSACTIONS ON MULTIMEDIA, 2014.

[6] Perfecto Herrera Joan Serra, Emilia Gomez and Xavier Serra. *Chroma Binary Similarity and Local Alignment Applied to Cover Song Identification.* IEEE Transactions on Audio, Speech, and Language Processing, 2008.

[7] F. Bimbot G. Sargent and E. Vincent. *A regularity-constrained Viterbi algorithm and its application to the structural segmentation of songs.* Proceedings of the International Conference on Music Information Retrieval, 2011.

[8] Bee Suan Ong. *STRUCTURAL ANALYSIS AND SEGMENTATION OF MUSIC SIGNALS.* UNIVERSITAT POMPEU FABRA, 2006.

[9] Jonathan Foote. *Visualizing music and audio using self-similarity.* MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia, 1999.

[10] Jonathan Foote. *Media segmentation using self-similarity decomposition.* Proceedings Volume 5021, Storage and Retrieval for Media Databases 2003, 2003.

[11] Marco Thiel Norbert Marwan, M. Carmen Romano and Jürgen Kurths. *Recurrence Plots for the Analysis of Complex Systems.* Physics Reports, 2007.

[12] K. Noland M. Mauch and S. Dixon. *Using Musical Structure to Enhance Automatic Chord Transcription.* 10th International Society of Music Information Retrieval, 2009.

[13] Jonathan Foote. *Automatic Audio Segmentation Using A Measure of Audio Novelty.* FX Palo Alto Laboratory Publications, 2000.

[14] Stephen V. Stehman. *Selecting and interpreting measures of thematic classification accuracy.* Remote Sensing of Environment., 1997.

[15] Nathalie Japkowicz Marina Sokolova and Stan Szpakowicz. *Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation.* Advances in Artificial Intelligence Vol. 4304, 2006.

[16] Cheng-i Wang Oriol Nieto, Gautham J. Mysore et al. *Audio-Based Music Structure Analysis: Current Trends, Open Challenges, and Applications.* TISMIR - Transactions of the International Society for Music Information Retrieval, 2020.

[17] Mark Levy Matthias Mauch, Robert M. MacCallum and Armand M. Leroi. *Selecting and interpreting measures of thematic classification accuracy.* Royal Society Open Science, 2015.

[18] Dawen Liang McFee, Colin Raffel et al. *Librosa; Audio and music signal analysis in python.* In Proceedings of the 14th python in science conference, 2015.

[19] André Pires and Marcelo Queiroz. *Real-time unsupervised music structural segmentation using dynamic descriptors.* University of São Paulo, 2011.

[20] Jordan B. L. Smith. *A comparison and evaluation of approaches to the automatic formal analysis of musical audio.* McGill University - Montreal, Quebec, Canada, 2010.

[21] Mumtaj Begam Lindasalwa Muda and I. Elamvazuthi. *Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques.* JOURNAL OF COMPUTING, 2010.

[22] E. Newman S. S. Stevens and J. Volkmann. *A Scale for the Measurement of the Psychological Magnitude Pitch.* The Journal of the Acoustical Society of America, 1937.

[23] J.W. Piconen. *Signal Modeling Techniques in Speech Recognition.* Proceedings of the IEEE, 1993.

[24] J.O. Smith. *Spectral Audio Signal Processing.* W3K Publishing, 2010.

[25] Oriol Nieto. *DISCOVERING STRUCTURE IN MUSIC: AUTOMATIC APPROACHES AND PERCEPTUAL EVALUATIONS.* Doctor of Philosophy in the Steinhardt School of Culture, Education, and Human Development - New York University, 2015.

[26] M. Goto. *A Chorus-section Detecting Method for Musical Audio Signals.* International Conference on Acoustics, Speech, and Signal Processing, 2003.

10 Appendix 1: In-Depth Theoretical Background

This appendix provides more detailed explanations of some of the mathematical tools already included in the body of the paper, as well as other mathematical descriptions not previously mentioned, but necessary to get a better grasp of the background of this complex problem.

10.1 Audio Features Descriptors

In this subsection the description and explanation of the audio feature descriptors that will be used throughout this project are presented. Even though the author is aware of other methods such as Tonal Centroids, Pitch-Chromograms or Zero Crossing [8], with the purpose of not extending this section beyond what is necessary, the analysis of these techniques will be avoided, since they have not been found relevant for our purpose.
In summary, the parameterization itself is a fundamental step in the development of a segmentation algorithm, since it has a direct effect on how the boundaries are detected.

10.1.1 Mel Frequency Cepstral Coefficients - MFCC

The Mel Frequency Cepstral Coefficients, from now on referred as MFCC, is a well known and commonly used feature extraction technique, which consist in the parameterization of each frame of the signal.
The MFCC uses the Mel scale, which is based on the way humans distinguish between frequencies (a non-linear perception), to divide the frequency band into sub-bands, and then extracts the Cepstral Coefficients using the Discrete Cosine Transform (DCT). The fact that is it based on the human perception of sound, makes it a good choice for audio feature extraction, specially since our main focus is trying to reflect in our segmentation algorithm the way humans perceive the musical piece.
Going into further detail, the process can be broken down into the following steps:

10.1.1.1 Pre-Processing

Before beginning the process of feature extraction, which would be later on analysed, there are a few steps that need to be taken in order to ensure the quality of the signal we are working with. In order to avoid aliasing in the signal, which would negatively impact the feature extraction, we first need to ensure that the recording we are working with is sampled at a sample rate of 44.1kHz. According to the Nyquist principle, as long as we are working with a sample rate of 44.1kHz, theoretically we can obtain a total bandwidth of 22.05kHz.

This way, since the signal we are working with has a sample rate of 44.1kHz, and therefore a bandwidth of 22.05kHz, this implies that the signal is contained only within the first Nyquist zone (from 0Hz to 22.05kHz), therefore ensuring there is no overlap with the second Nyquist zone, which would cause aliasing.

10.1.1.2 Frame Windowing

Being music recording a slowly time-varying (or quasi-stationary) signal, which can be considered a non stationary process, it is known that the Fast Fourier Transform (FFT) would produce distortions. In order to extract stable acoustic characteristics, we must assume the audio behaves as a stationary process locally (for short periods of time). Therefore, we conclude that, in order to carry out the signal analysis, it is needed to fragment the signal into shorter frames [20].
To accomplish this, we need to taper the signal. A Hamming window is used with this purpose, as described below:

> Let $X = [x_1, ..., x_N]$ be the vector representing the recorded music, where N is the number of samples in the signal.
>
> Using the Hammer window described as:
>
> $$\mathbf{w_H}[\mathbf{n}] = 0.54 - 0.46\cos\left(\frac{2\pi}{N}n\right); \qquad \text{for} \quad 0 \leq n \leq N \tag{9}$$
>
> We obtain $X'[n] = [x'_1, ..., x'_N]$ being the vector containing the windowed frames (column vectors x'_n) of the signal.

The decision to use a Hamming window, instead of other frame blocking and windowing techniques, was made with the intention of accomplishing smooth edges, enhancement of the harmonics, and to reduce the edge effect while computing the Discrete Fourier Transform (DFT) on the signal. It is also important to mention that the size of this window (and consequently, the size of the FFT) can not be chosen randomly, and a thorough study of its effects on the results of the segmentation algorithms must be conducted (see section 5.3).
In addition, we need to consider the overlapping between frames, therefore a step-size needs to be studied and detailed in the process. The purpose of the overlapping between samples is that, as an inherited condition from the use of a Hamming window, all frames present some lose of information in the edges. The best way to avoid the lose of information is through the use of overlapping [21].

10.1.1.3 Discrete Fourier Transform Spectrum

In this step, each frame is converted into magnitude spectrum by applying and calculating its DFT:

$$\mathbf{X}[\mathbf{k}] = \sum_{n=0}^{M-1} x[n]e^{-j\frac{2\pi kn}{M}}; \qquad \text{for} \quad 0 \leq n \leq N-1 \tag{10}$$

> Being M the number of points used to compute the DFT

10.1.1.4 Mel Spectrum

The Mel spectrum is computed by passing the Fourier transformed signal through a set of band-pass filters known as Mel filter-bank. The Mel scale can be described as having a linear frequency spacing until 1 kHz, and a logarithmic spacing above 1 kHz [22].

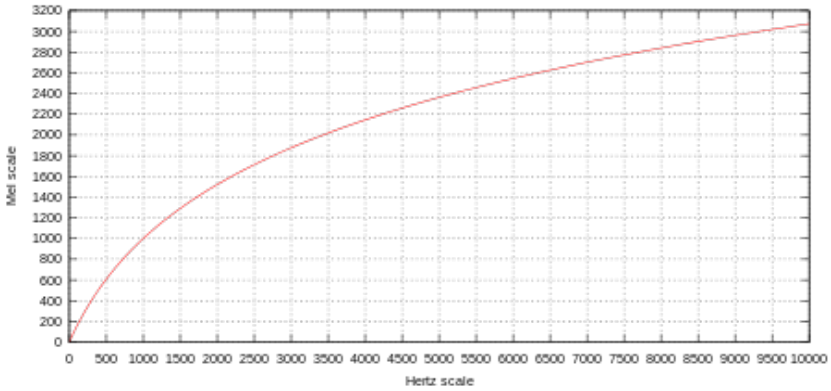


Figure 20: Graph of the relation between Hertz frequency and Mel scale

The relation between the conventional frequency and the Mel spacing, can be described as:

$$\mathbf{f_{Mel}} = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{11}$$

Being f the Hertz frequency, and f_{Mel} the representation of the human perception of said frequency.

To build the filter-bank necessary to obtain the Mel spectrum, first the center-points for each filter need to be calculated. The process followed to obtain said frequencies is illustrated in figure 21.

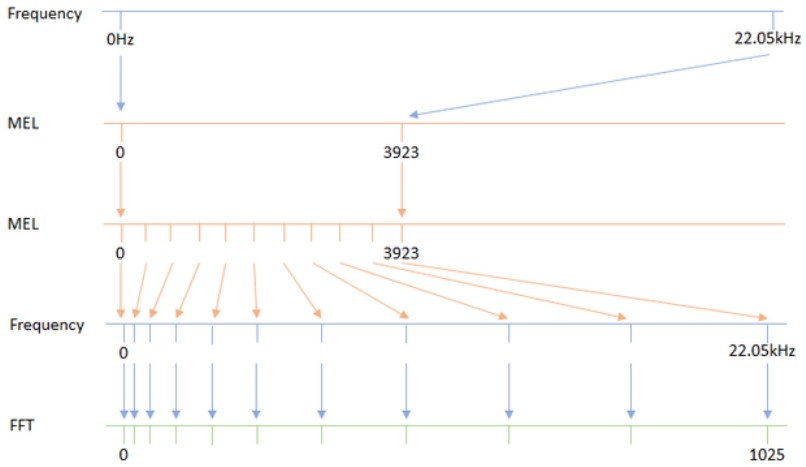


Figure 21: Description of the process followed to obtain the center-points for the filter-bank

Once the previously described values are obtained, we can compute the filter bank as described below.

$$\mathbf{H_m[k]} = \begin{cases} 0, & k < f(m-1) \\ \frac{2(k-f(m-1))}{f(m)-f(m-1)}, & f(m-1) \leq k \leq f(m) \\ \frac{2(f(m+1)-k)}{f(m+1)-f(m)}, & f(m) < k \leq f(m+1) \\ 0, & k > f(m-1) \end{cases} \quad ; \qquad \text{for} \quad 0 \leq m \leq M-1 \tag{12}$$

Where $H_m[k]$ in (12) is the triangular filter for the k^{th} energy spectrum bin.

Therefore, after area-normalizing the filter-bank by dividing the amplitude of the triangle by the width of its Mel band, the following representation is obtained:

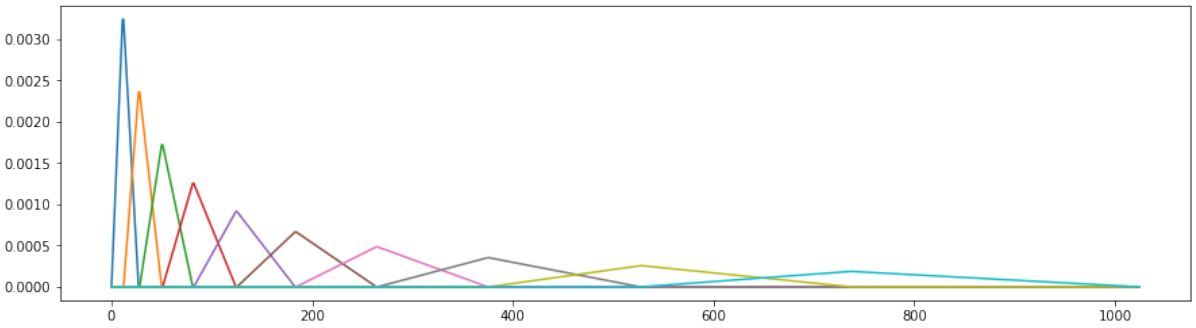


Figure 22: Normalized Mel filter-bank

After that, the Mel spectrum of each frame is computed. This can be done by multiplying the magnitude spectrum of each frame $X[k]$ by each of the of the triangular Mel weighting filters $H_m[k]$, as described below.

$$\mathbf{S[m]} = \sum_{k=0}^{N-1} \left[|X[k]|^2 H_m[k] \right]; \qquad \text{for} \quad 0 \leq m \leq M-1 \tag{13}$$

Where $H_m[k]$ in (13) is the triangular filter for the k^{th} energy spectrum bin as described in (12).

Finally, we obtain $S[m] = [x_{mel_1}, ..., x_{mel_N}]$, being a vector containing each frame x_{mel_k} of the music signal as a vertical vector of k values, each representing the audio power for the k filters in the filter-bank. This is referred as Mel Frequency Coefficients

10.1.1.5 Cepstral Coefficents

The final step in generating the MFCC is to use the Discrete Cosine Transform (DCT) applied to the transformed Mel frequency coefficients in order to obtain the Cepstral Coefficients.
Prior to computing the DCT, the Mel spectrum is usually represented on a log scale. This results in a signal in the Cepstral domain, with a quefrequency (the time scale of the Cepstral domain is usually referred as quefrequency) peak corresponding to the pitch of the signal, and a number of formants representing low quefrequency peaks [23].

$$\mathbf{c}[\mathbf{n}] = \sum_{m=0}^{M-1} \log_{10}(S[m]) \cos\left(\frac{\pi n(m-0.5)}{M}\right); \qquad \text{for} \quad n = 0, 1, 2, ..., C-1 \tag{14}$$

Where $c[n]$ in (14) is the calculated C Cepstral Coefficients for each frame. Therefore, obtaining C MFCCs.

The representation of the C MFCC obtained in 14 looks as illustrated in figure 24.

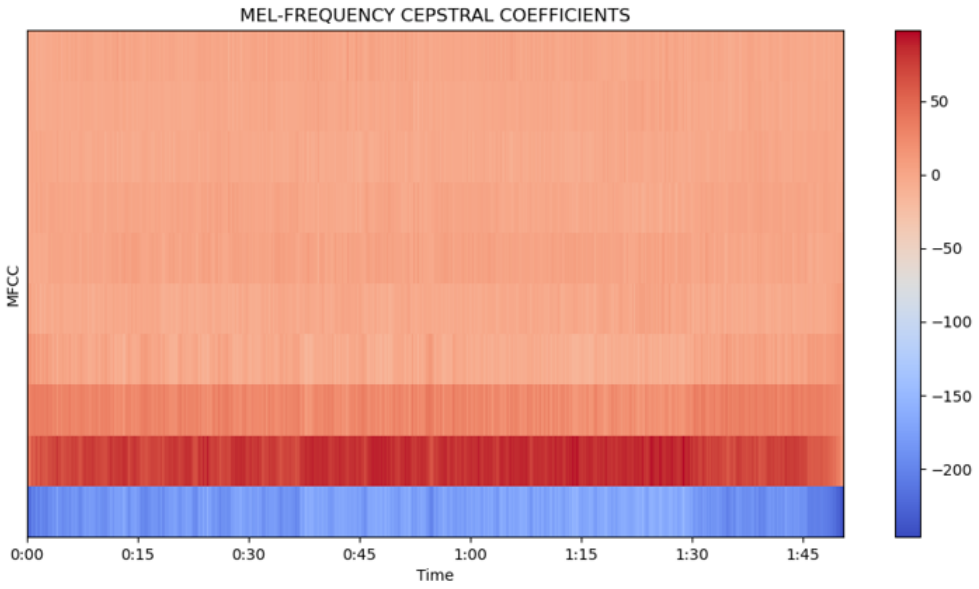


Figure 23: Example of Cepstral Coefficients extracted from a piano track.

It is important to mention a few last details at this point, and before initiating another section. First of all, it is important to understand that the zeroth coefficient of the MFCC can sometimes be excluded, since it represents the average log-energy of the input signal, which only carries volume-specific information [23].

Also, it should be mentioned that the value of C is not trivial, and it needs to be studied and optimized (see section 5.3) as it has an important effect on the effectiveness of the algorithm

10.1.2 Harmonic Pitch Class Profiles - HPCP

Harmonic pitch class profiles (from now on, referred as HPCP) is a feature extraction technique, based on a pitch class profile descriptor. HPCP is an enhanced pitch distribution feature, constituted by sequences of feature vectors that describe tonality, measuring the relative intensity of each of the 12 pitch classes (and its harmonic tones) of the equal-tempered scale within a frame. This results in a 12-dimensional profile for each frame, and it is typically visualized as a $12 \times N$ matrix

Short-Time Fourier Transform

After following the same steps of pre-processing and windowing as described in sections 10.1.1.1 and 10.1.1.2, we convert the framed signal$X[n]$ in a time series of length N'. To do that, we compute the Short-Time Fourier Transform [24] as described below:

$$\mathbf{X_m[k]} = \sum_{n=0}^{M-1} x[n + mH]w[n]e^{-j2\pi n \frac{k}{N_{DFT}}}; \tag{15}$$

Where $x[n]$ in (15) is a frame of the recorded signal, w $w[n]$ is the analysis window, H is the hop size for the sliding window, M is the size of the window and lastly N_{DFT}. The time frame index would be $m \in [0 : N-1]$ and the frequency index $k \in \left[0 : \frac{N_{DFT}}{2} - 1\right]$

The process described below require the absolute value of the complex frequency domain representation $|Xm(k)|$, so that the magnitude spectrum of the signal can be obtained.

HPCPs

To capture the amount of energy contained by each of the twelve notes of the conventional scale, across a specific number of octaves, in one of the frames that constitute the music signal, the following process must be computed [25].

$$\mathbf{HPCP_m[p]} = \sum_{k=f_0}^{f_c} (\varphi(k) = p)|X_m(k)|^2; \qquad \text{for} \quad p \in [0 : 11] \tag{16}$$

Where f_0 and f_c are the start and end frequencies that delimit the bin in which HPCPs are computed. p represents the 12 notes, and φ in (16) is a frequency mapping function defined in (17):

$$\varphi(k) = \left[12 \log_2 \left(\frac{f_s}{f_{ref}} \frac{k}{N_{DFT}}\right)\right] \mod 12; \qquad \text{for} \quad p \in [0 : 11] \tag{17}$$

Where f_s represents the sample rate of the recorded signal, and f_{ref} the frequency of the first bin, corresponding to the note $p = 0$.

Being as they are HPCPs usually normalized, we obtain the following expression, describing the whole process as in (16) and (18):

$$\mathbf{HPCP'_m[p]} = \frac{HPCP_m[p]}{\arg\max_p HPCP_m[p]} = \frac{\sum_{k=f_0}^{f_c} \left(\left[12 \log_2 \left(\frac{f_s}{f_{ref_p}} \frac{k}{N_{DFT}} \right) \right] \mod 12 \right) |X_m(k)|^2}{\arg\max_p HPCP_m[p]} \tag{18}$$

Through this process, we obtain a vector $x[n] = [x_1, ..., x_{N'}]$ which contains a vertical vector $x_n[p]$ (dimension 12x1) corresponding to each of the frames that make up the initial signal, and in which the information of the HPCPs is contained.

As a result of the execution of this audio characterization technique, we obtain the following type of plot, which can be described as a matrix in which the power corresponding to each note can be observed, for each instant of the signal (frame).

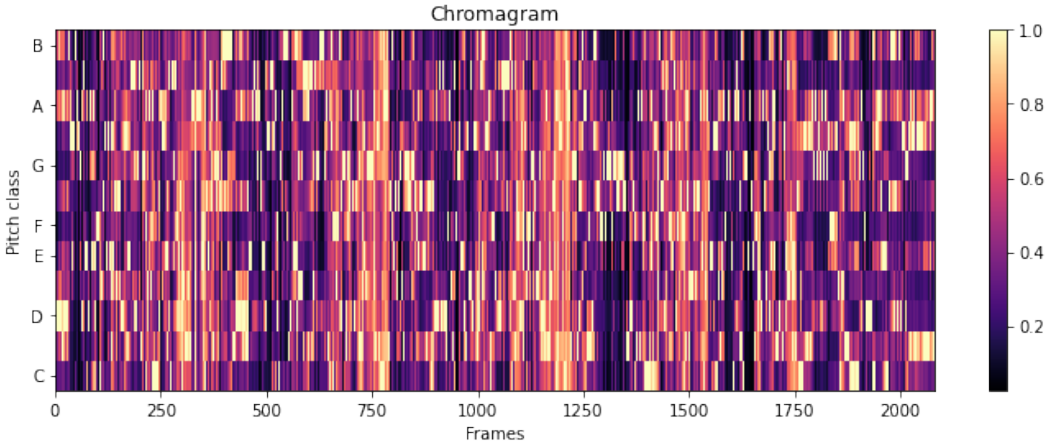


Figure 24: Example of normalized HPCP features extracted of a piano track.

10.2 Tools for Structure Discovery

In this section, we describe another structure of lesser relevance, but necessary to fully understand this thesis.

10.2.1 Circular Time-Lag Matrix

The Circular Time-Lag Matrix is in fact a modified version of the already explained SSM (see section 3.2.1), but a certain post-processing is added to the steps [26].
The Circular Time-Lag Matrix (from now on referred as CLM) represents the similarity of each frame to each of the K frames. Graphically, it implies that it rows correspond

to K appropriately padded diagonals above the main diagonal of an SSM. This allows to detect repetitions (appearing as horizontal lines, instead of the diagonals in the SSM) within a limited context.

Such process requires the circularly shifting of the rows of the previously constructed matrix (this being either a SSM or a RM) such that.

$$\mathbf{CLM_{i,j}} = M_{k+1,j}; \qquad \text{for} \quad \forall i,j \in [1:N] \tag{19}$$

Where M in (19) represents a previously computed square matrix, of dimension $N \times N$, and $k = (i + j - 2) \mod N$.

10.3 Distance Metrics

Which metric is decided to use can have various effects on the segmentation algorithm of which it is a part, since it must be understood that not all distance metrics quantify the same factors.
Therefore, this section will describe some of the most common metrics that have proven to be useful in the course of this thesis.

- **Canberra Distance**

  The Canberra distance is a numerical measure of the distance between pairs of points in a vector space. It can be understand as a weighted version of the more commonly known Manhattan distance.

$$d_{canb}(v_i, v_j) = \sum_{k=1}^{n} \frac{|v_{i_n} - v_{j_n}|}{|v_{i_n}| - |v_{j_n}|}; \tag{20}$$

  Where v_i and v_j in (20) are characterization vectors.

- **Cosine Distance or Similarity**

  The cosine similarity is a measure of similarity between two frames. It is defined as vectors in an inner product space, and the cosine similarity is defined as the cosine of the angle between them, that is, the dot product of the vectors divided by the product of their lengths.

$$d_{cos}(v_i, v_j) = \frac{< v_i, v_j >}{||v_i||||v_j||}; \tag{21}$$

- **Correlation Distance**

  The Correlation distance is a measure of dependence between two paired random vectors of arbitrary, not necessarily equal, dimension. In our case, the correlation distance coefficient is zero if and only if the vectors are independent. Thus, distance correlation measures both linear and nonlinear association between two frames.

$$d_{corre}(v_i, v_j) = \frac{(v_i - \mu_{v_i}) \cdot (v_j - \mu_{v_j})}{||v_i - \mu_{v_i}||^2 ||v_j - \mu_{v_j}||^2}; \tag{22}$$

  Where μ_{v_i} in (22) represents the mean of the vector v_i , and $||v_i - \mu_{v_i}||^2$ represents the l_2-norm, also known as Euclidean norm.