AUTOMATIC STRUCTURAL SEGMENTATION OF MUSIC

INSIGHTFULLY CLUSTERING THE BEATS IN A GIVEN PIECE OF MUSIC TO REFLECT IT'S MUSICAL STRUCTURE

by

Lyndon D Quadros

A masters project report submitted in partial fulfillment of
the requirements for the degree of

Master of Science

(Electrical Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

December 18, 2015

Acknowledgements

I am extremely grateful to my Graduate Adviser and Master's Project guide Prof. William Sethares for his constant support, assistance, encouragement and patience during the course of this project and my Master's Degree as a whole. Being under his tutelage was utterly enjoyable and as delightful as it was enlightening.

I wish to extend my gratitude to all the Professors at the Department of Electrical and Computer Engineering shared their knowledge and experience with me during the various courses I took towards the completion of my degree. The administrative staff of the department have also been ever helpful and willingly so.

I wish to thank the Department of Physics for providing me funding and monetary support in the way of a Teaching Assistant-ship. In particular, I wish to convey my heartfelt gratitude to Dr. James Reardon, who has been a great friend, mentor and guide.

My sincere thanks to Dr. Mark Levy and Ruofeng Chen whose papers and assistance during the initial stages of this project were instrumental in shaping it.

I wish to thank my friends and colleagues for helping me through difficult and stressful times, both in and out of school.

Most importantly, the completion of this stage of my life and career would never have been possible without the patience, love and never-ending support of my parents, my sister and my family. Last, but not the least, I would like to thank my late grandmother, Mrs. Cecilia Fernades for her blessings as she continues to be a major influence and a constant pillar of support in my life.

ABSTRACT

This Project posits elementary analogies of existing Probabilistic and Machine Learning models that have been used to find solutions to the problem of the Structural Segmentation of Musical audio. I have tried to use the idea that the chord of a given beat or frame of a song is an analogous representation of the *states* generated by trained Hidden Markov Models in generating feature vectors for the aforementioned problem; and that the knowledge of the temporal boundaries within which, a group of frames lie, can be used as constraints in creating the feature vectors that are eventually clustered to identify the pattern in which the various segments of a song repeat.

DISCARD THIS PAGE

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Problem definition and Related Work

The Structural Segmentation of Musical Audio has been a long-existing and challenging problem in Music Information Retrieval and the solutions to this problem vary from using matrix completion to systematic clustering of feature vectors of a song. One of the major reasons for developing Structural Segmentation techniques for musical audio is to achieve Audio Thumbnailing, which is a very useful tool for Music researchers, composers and listeners alike. Knowledge of the structure of music can also help make tasks such as Automatic Genre Classification and Recommendation algorithms more accurate; since structure or form, as it is sometimes called, is a very important characteristic of a song.

The structure or form of a song typically describes the sequence of various parts in the song or the arrangement of the song. One of the most common is the verse-chorus form; a typical example of which would be a song with the following structure:
Intro-verse-chorus-verse-chorus-refrain-chorus.

This project aims at finding the right sequence in which these segments repeat; but does not actually concern itself with the *correct human labeling* of the segments. For example, for a song having the structure mentioned above, the output of the method employed in this project will be as follows:

A-B-C-B-C-D-C

This representation of the structure does not cause any loss of information as compared to the actual *verse-chorus-* type of representation. Also, what one person would identify as the *refrain* of the song might be what some other person identifies as the *bridge*. Due to such ambiguity, the standard representation for the structure of a song in already existing literature has been of the *A-B-* type mentioned above.

Early work on structural segmentation has been largely based on assuming that the form of a song can be identified as being one out of a few listed canonical forms (for example, the intro cannot be at the end). In this case, a brute-force matching of audio features to possible structural templates [1] or a heuristically constrained search across certain elements of structure [2], [3] have been the techniques that have been employed.However, such assumptions might not be very practical for a lot of the songs. In fact, of the Beatles 210 recorded songs, only 23 would follow the introversechorusversechorus sequence; some of those in fact dispense with the intro, and they continue in several different ways [4].The newer approaches therefore aim at basing the estimation of structure on a variety of features that humans perceive and analyze when deciding the structure of music which include the differences in melody, overall feel of the section, rhythm and beat, changes of instrumentation etc. In [5], the authors used Non-Negative Matrix factorization, while in [6], the authors developed a system which combined harmonic and timbral information and then used multi-level clustering and non-negative matrix factorization to get competitive results.

This project builds primarily on the state labeling and histogram clustering approach in [7], while also incorporating ideas from [8],[9] and [10]. I have used the Chroma feature vector in conjunction with Boundary Detection and Histogram Clustering in order to accomplish the aforementioned task.

The following subsection contains an overview of the approach employed in this project. Chapter 2 gives details about the extraction of the Chroma feature vector and the state labeling of each beat of the song. Chapter 3 describes the boundary detection method used. Chapter 4 gives details about the creation of the histogram of states, which incorporates the knowledge about the boundaries obtained in the previous step. It also contains some detail of the clustering methods used to cluster these histograms of states. Chapter 5 enlists the tests conducted, gives the metric for evaluating the performance of the algorithm, and gives the results of these tests. Chapter 6 gives a conclusion to the project and includes possible future development based on the content of this project.

1.2 Employed Approach

Figure 1.1 shows the bock diagram of the employed approach in this project.

In [7], the authors pose the transitions in a song as a change in the sequence and probability of the occurrence of *states* that each beat in the song belongs to. These *states* are learned from a Hidden Markov Model. The authors use the Audio Projection Envelope descriptor from the MPEG-7 standard as a feature vector and assign state labels to each of these vectors in the vector sequence using a trained Hidden Markov Model. One of the first tasks I undertook in this project was to be able to make an educated guess of what these states could be *or* find an analog of these states that can be computed without the use of Hidden Markov Models. The *guess* was that these states could correspond to *a certain chord or some kind of musical spectral information*. In order to compute this *information/chord*, I used the Chroma feature vector and assigned states accordingly. Section 2.2 contains the details of the state labeling approach.

The system then detects the *approximate* number of boundaries in a given song using the method described in [10]. The boundary detection aids in the creation of feature vectors *(A histogram of state labels)* used in the clustering part and the smoothing of the cluster indices.

The next step is to create a histogram of state labels for each grouping of d_{ml} beats. These histogram vectors are then clustered in a semi-supervised framework using a few clustering approaches available in literature.

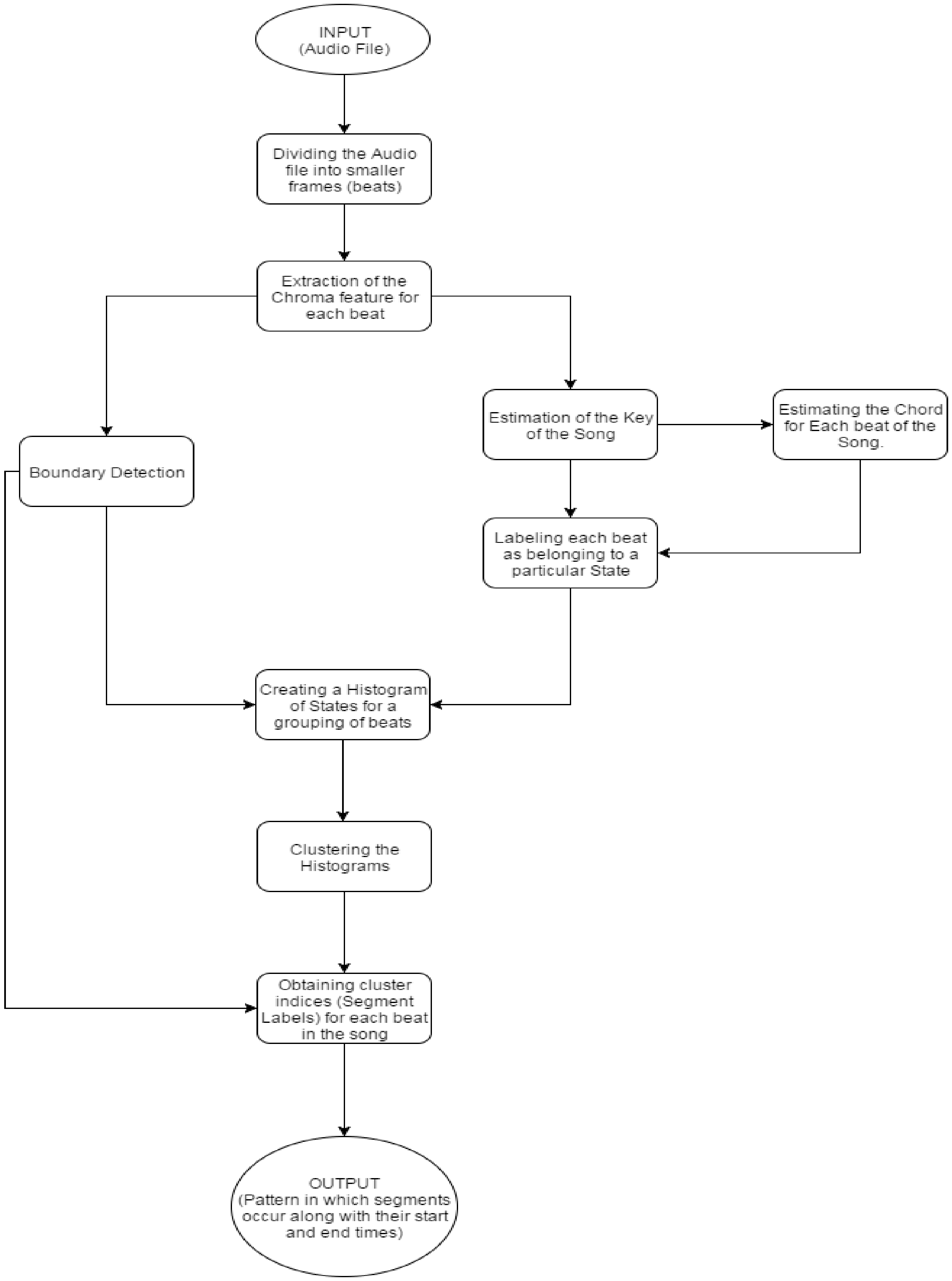
INPUT
(Audio File)
Dividing the Audio file into smaller frames (beats)
Extraction of the Chroma feature for each beat
Estimation of the Key of the Song
Estimating the Chord for Each beat of the Song.
Boundary Detection
Labeling each beat as belonging to a particular State
Creating a Histogram of States for a grouping of beats
Clustering the Histograms
Obtaining cluster indices (Segment Labels) for each beat in the song
OUTPUT
(Pattern in which segments occur along with their start and end times)

Figure 1.1 The Master Block Diagram of the employed approach.

The cluster indices thus obtained correspond to the different sections in the song. The indices obtained from each histogram can then be assigned to the respective individual beats that made up each histogram, thereby obtaining the segment to which each beat belongs to. The indices obtained usually have a little sporadic nature to them and can be smoothed using a *Commonest Filter* approach and the information about the boundaries of the song.

1.3 Sample Input and Sample Output

The code of this project takes an audio file in the **.WAV** format as input. Another very helpful input argument may be the number of segments the user would like the to split the song into. Typically, the determination of the right number of clusters in the conventional k-means kind of approach is a NP-hard problem. As noted in [8], it is not impractical to expect an input about the number of segments from the user. The output is a sequence of *start times* and *end times* for each segment accompanied with the segment label. The system's output for the segmentation of *'Creep'* by *Radiohead* is given below:

```
Start Time   End Time  Label
0.000000  11.152000    A
11.152000  53.136000   E
53.136000  65.600000   D
65.600000  86.592000   C
86.592000  107.584000  C
107.584000  122.016000  A
122.016000  143.008000  A
143.008000  164.000000  A
164.000000  175.808000  C
175.808000  191.552000  A
191.552000  205.984000  B
205.984000  226.976000  A
```

Chapter 2

Feature Extraction and State Labeling

2.1 The Chroma Feature Vector

The Chroma feature vector basically organizes the spectral information of the given audio signal into 12 pitch class bins based on the semitones in Western-classical music. In musical applications, and especially for the task of structural segmentation, it is more useful to compute the Chroma feature for each *frame* of time of the audio signal. Hence, before one begins computing the Chroma, one must first divide the audio signal into smaller *frames*.

2.1.1 Pre-Processing

I chose the duration of one beat of the song as the length of one frame. This can be done by using a tempo estimation and beat detection algorithm. In this project, I have used the tempo estimation algorithm developed in [11]. Once the tempo (beats per minute) is estimated, the audio signal can be divided into n beats depending on the length of the audio signal and it's sampling rate. This is followed by the computation of the *Discrete Fourier Transform* for each beat. This will result in n number of M-dimensional vectors, where M is the number of frequency bins and each entry in a vector is the Fourier co-efficient of the corresponding frequency. For musical applications, the typical choices for the M bins are logarithmically spaced points from $20Hz$ to $20kHZ$, which is the range of the frequencies a human being can perceive.

2.1.2 Computing the Chroma Feature

The Chroma feature organizes the magnitudes (averaged over all octaves) of the co-efficients in the DFT into frequency bins according to the semitones used in Western Music theory. In a simplified manner, it can be said that the Chroma feature gives the relative amplitude of each musical *note* in a given Audio signal.

Let's consider the *C* note for example. The frequency of the *C* note in the first octave is $32.70Hz$. It's frequency in the second octave is $65.40Hz$, in the third octave is $96.10Hz$ and so on. The entry corresponding to this note in the Chroma vector will be the average of the magnitudes of the corresponding frequencies in each octave. More specifically, it is the average of the *log-magnitudes*, since the human ear perceives sound amplitudes logarithmically.

Let S_k be the set of all frequencies belonging to pitch-class k. Here, *pitch class* is analogous to a *note* in music. Hence, if we are talking about the *C* note, $S_c = \{32.70, 65.40, 96.10, \cdots\}Hz$ Let $A_j(k)$ be the log-magnitude of the Fourier co-efficient of the j^{th} element in S_k. Let N_k be the cardinality of S_k. Then, the Chroma value c_k for the pitch class k [12] is given by:

$$c_k = \sum_{j \in S_k} \frac{A_j(k)}{N_k} \tag{2.1}$$

Once the Chroma is extracted for each of the 12 pitch classes for each beat in the song, the result is a $n \times 12$ matrix, C that contains the sequence of the Chroma vectors of each beat of the song.

2.2 State Labeling

As stated in section 1.2, I try to find an analog, using elementary music theory, of the states generated by the Hidden Markov Models in [7] and posit that these states could be analogous to a *chord* or representative of *spectral uniqueness*. Once the beats have been assigned these state labels, similar state transitions should occur in similar segments of a song as shown in [7].

2.2.1 Estimating the *Chord*

In order to estimate the amount of presence of a chord in a beat, I computed the sum of the Chroma values for each pitch class belonging to the triad of a chord. For example, if I want to compute the presence of the *C major* chord in a beat, I would compute the sum of the Chroma values of the *C, E* and *G* bins of the beat.

The triad that yields the maximum presence when computed as above comprises the chord of the given beat. I then assigned a state number or state index to each chord depending on the key of the given piece of music.

2.2.2 Elementary Approximation of the Key of the song

One can use already existing key-detection algorithms in order to compute the key of the given piece of music. However, these algorithms are usually computationally expensive. I use a simple approach to only approximate the key of the song using the chroma values for each bin, again incorporating very elementary knowledge of music theory.

Every key in Western classical music is characterized by a set of seven pitch classes or notes. Let Γ_v be set of pitch classes belonging to the key of v. Let $c_k(v)$ be the chroma value of the k^{th} pitch class in Γ_v. Then the key v can be estimated as:

$$v = argmax_v \sum_{k \in \Gamma_v} c_k(v) \tag{2.2}$$

That is, the key of the music is the one whose corresponding set of pitch class profiles yields the maximum sum of Chroma values. Again, this might not give the key very accurately; but it does a decent job at approximating the key of the song to the nearest relative major/minor. That is, if the song is in the key of *C major*, one could expect the above formulation to yield a key of *C major* or *A minor*. However, this will not affect the state indexing of the chord and consequently, the state labeling of the beats in any major way. This will be shown in the next subsection.

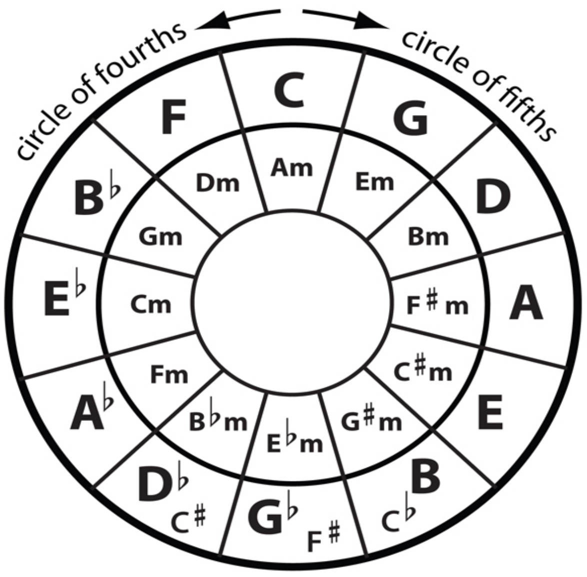

Figure 2.1 A diagram illustrating the sequence of notes in the Circle of Fifths (taken from *http://www.evirtuoso.com*).

2.2.3 State Indexing of the Chords and State Labeling of the Beats

By state indexing of a chord, I mean that I would like to assign a number (an integer from 1 to 24) to a chord (considering only the 12 major and the 12 minor chords). These numbers can then be assigned as state labels to the beats depending on the chord with the maximum presence in a given beat.

An easy way to index each chord would be to assign it a constant value that does not change depending on the song. For example the *A major* chord could always be *state 1*, *B major* could always be *state 2* and so on. However, when these are assigned as state labels to the beats of the song, it's not trivial to detect any pattern in their transitions in order to figure out any similarity or dissimilarity between the different sections of a song.

However, once the key of the song has been approximated, the chord that directly corresponds to the key of the song can be indexed as state 12 or state 13 (the middle of 24), and all the other chords indexed around it according to the circle of fifths (Figure 2.1). An illustration of such a state indexing is shown in Table 2.1 and the resulting state labeling of the beats is

State Number	**Chord**
1	$G^{\#}$ major
2	F minor
3	$D^{\#}$ major
4	C minor
5	$A^{\#}$ major
6	G minor
7	F major
8	D minor
9	C major
10	A minor
11	G major
12	E minor
13	D major
14	B minor
15	A major
16	$F^{\#}$ minor
17	E major
18	$C^{\#}$ minor
19	B major
20	$G^{\#}$ minor
21	$F^{\#}$ major
22	$D^{\#}$ minor
23	$C^{\#}$ major
24	$A^{\#}$ minor

Table 2.1 State numbers for chords belonging to the **D major** scale

shown. Figures 2.2 through 2.4 show that state indexing according to the circle of fifths yields interpretive and useful results.

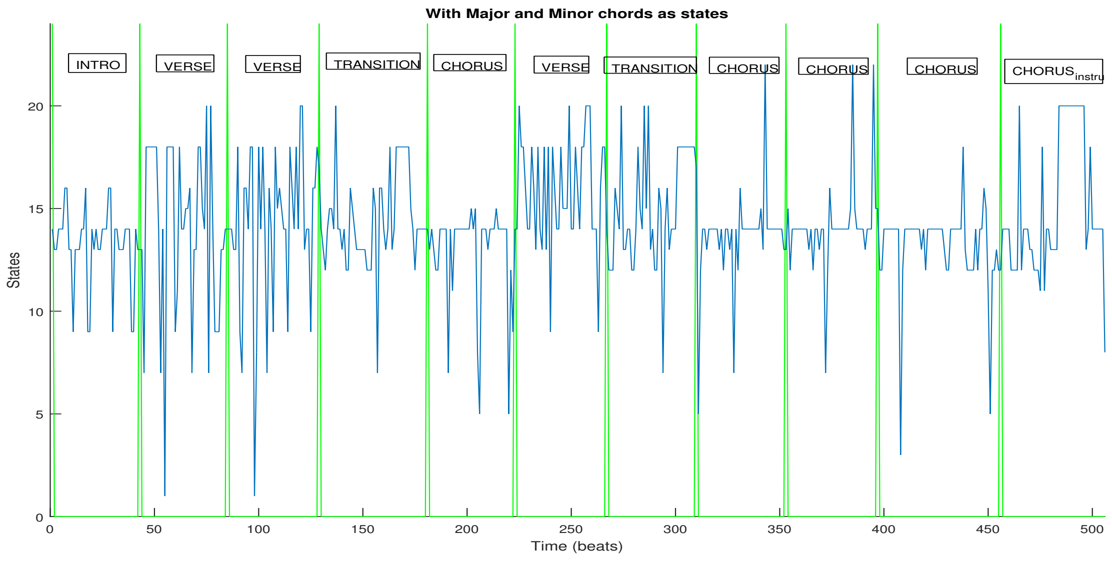

Figure 2.2 State labeling of the beats in *'Wonderwall'* by *Oasis*. The similarity of state transitions between the choruses and the verses is very evident.

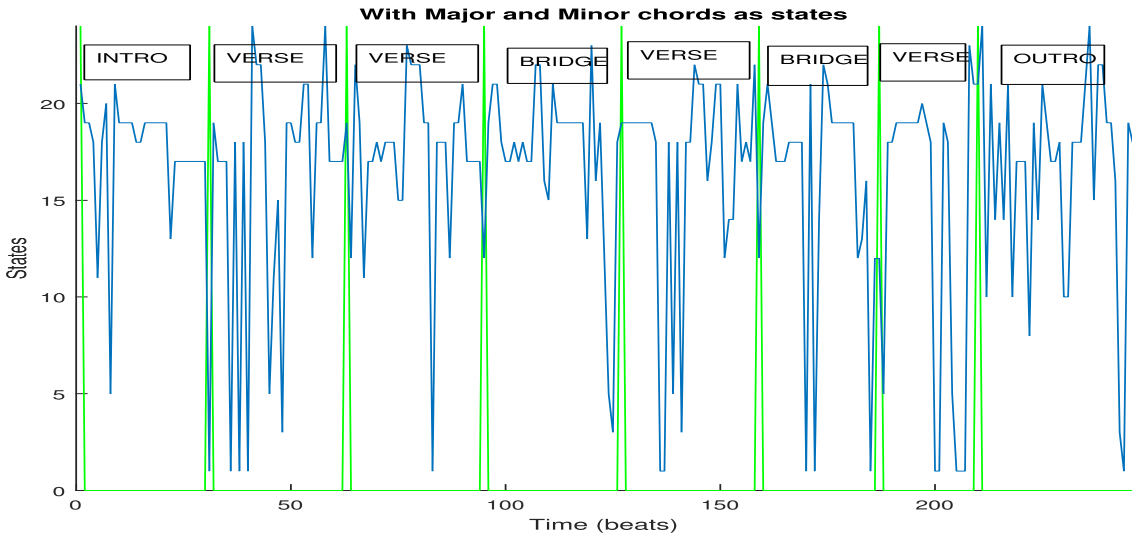

Figure 2.3 State labeling of the beats in *'Misery'* by *The Beatles*.

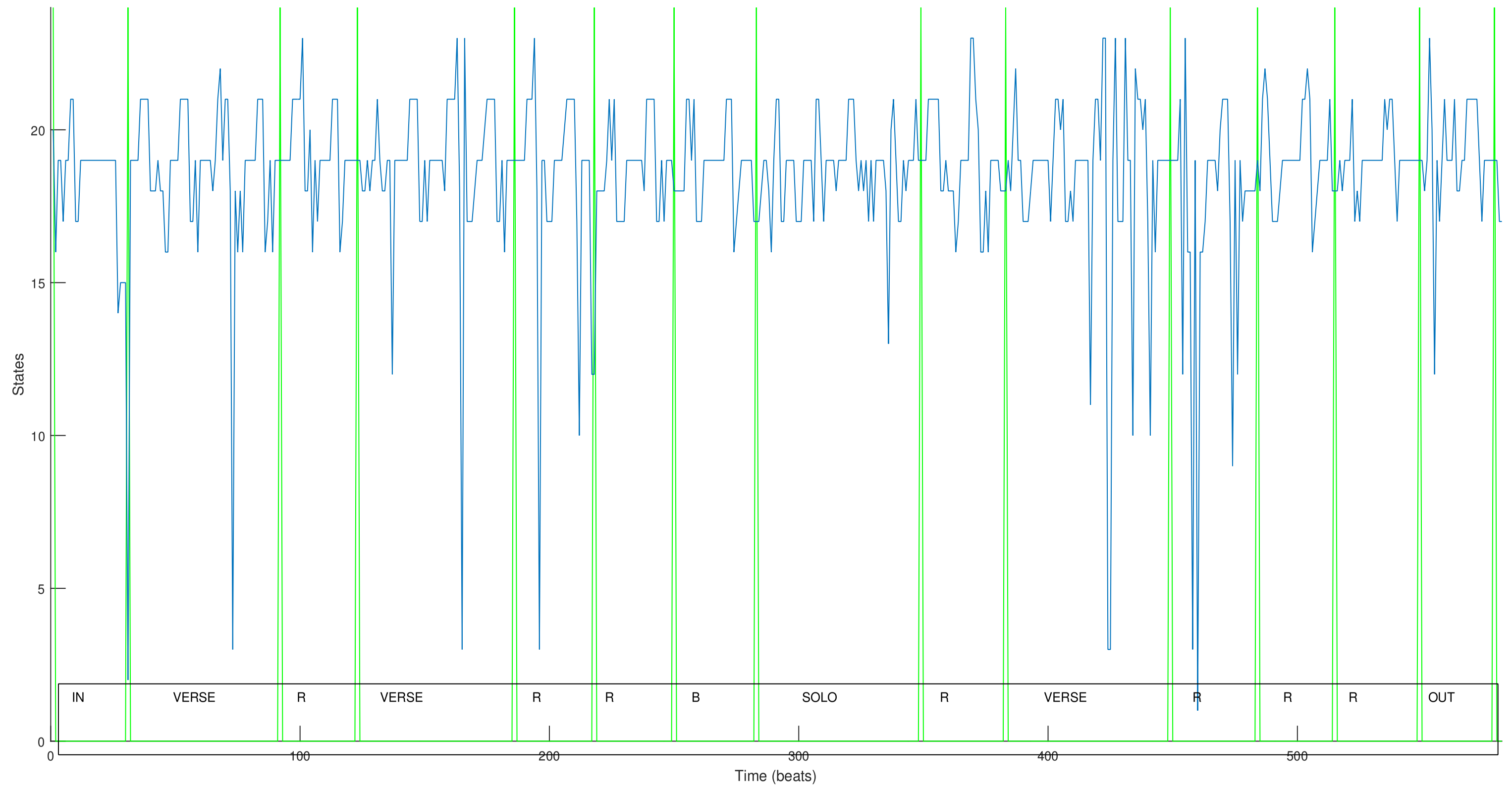

Figure 2.4 State labeling of the beats in *'Let It Be'* by *The Beatles*.

Chapter 3

Boundary Detection

3.1 Preliminary Boundary Detection and Smoothing

The boundary detection approach used in this project was developed in [9]. I used the Chroma feature computed in the previous chapter for this task. In this method, first, a self-similarity matrix is computed for the sequence of the Chroma features for a given song. This can be computed as follows:

$$S(i,j) = \rho(c_i, c_j), i, j = 1, 2, 3, ..., n \tag{3.1}$$

where $\rho(c_i, c_j)$ denotes the distance between the Chroma vectors for the i^{th} and the j^{th} beat. A typical choice for $\rho(.,.)$ is the Euclidean distance.

Once the Self-similarity matrix is obtained, a novelty measure $\eta(i)$ is computed for each beat using the following:

$$\eta(i) = \sum_{\alpha=-k_c/2}^{k_c/2} \sum_{\beta=-k_c/2}^{k_c/2} G_{k_c}(\alpha,\beta) S(i+\alpha, i+\beta) \tag{3.2}$$

where G_{k_c} is a symmetric Gaussian tapered checkerboard kernel of dimensions $k_c \times k_c$., the details of which are given in [9]. For the purpose of this project, I chose k_c = 96 and a Gaussian tapering with standard deviation $k_c/6$, which in this case, turns out to be 16. The choice of k_c was made empirically.

When $\eta(i)$ is computed as above, it could yield some *noisy* results and these can be made smoother using a moving average filter operation. I chose a window length of 8 beats for the implementation of the moving average filter on the sequence of the η_is.

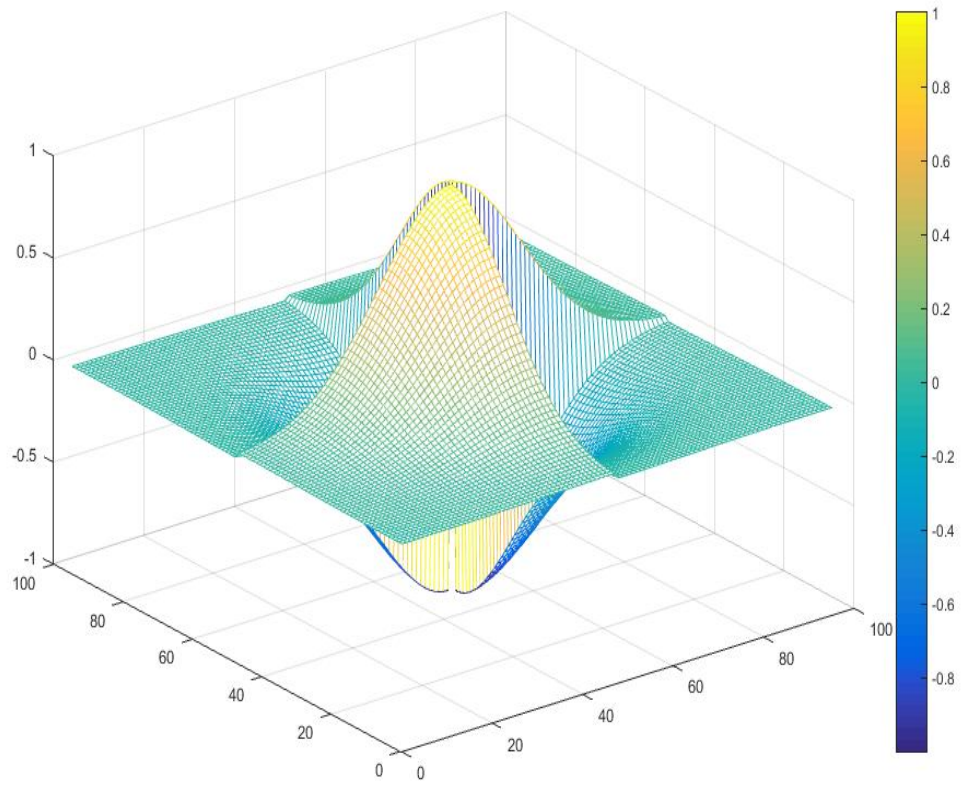

Figure 3.1 The Gaussian tapered Checkerboard Kernel.

Even after the smoothing, the number of boundaries detected could be very high. In order to combat this, a two-step process explained below can be used.

3.2 Temporal and Amplitude Thresholds to reduce the number of detected boundaries

One of the ways in which the number of boundaries can be reduced is by setting a temporal threshold t_{ml} in which two boundaries cannot occur. That is, if two of the detected boundaries are very close to each other, one can surely say from empirical evidence that these boundaries might not necessarily indicate two different sections. I chose t_{ml} = 16 beats for the temporal threshold between which two boundaries cannot occur.

One can further reduce the number of boundaries by comparing the magnitude of the novelty measure for each boundary with the magnitudes of the novelty measures of the other boundaries in the song. That is, one can choose a certain value for threshold magnitude η_{thresh} and any boundaries that have a $\eta(i)$ value lesser than η_{thresh} can be deleted from the set of all the boundaries. I chose the following value of η_{thresh}:

$$\eta_{thresh} = \bar{\eta} - \sigma_{\eta(n_b)} \tag{3.3}$$

$\forall n_b \in \mathcal{B}$ where $\mathcal{B}$ is the set of all the boundaries obtained after setting the temporal thresholds, $\bar{\eta}$ is the mean of the magnitudes of the novelty measures of the boundaries in $\mathcal{B}$ and $\sigma_{\eta(n_b)}$ is their standard deviation. I also tried using the difference between the mean and the variance of the $\eta(n_b)$ in order to impose a tighter threshold, but this leads to a loss of some important boundaries towards the end of the song. In the song *Wonderwall* by *Oasis*, for example, the tighter threshold leads to a loss in the boundary towards the end of the song where it transitions from a vocal chorus to an instrumental chorus, a transition which is noted in the reference segmentaions.

The resulting boundaries detected for a few of the songs in the data set along with the ground truth reference boundaries are shown in the figures 3.2 through 3.4. As can be seen from these figures, there are boundaries detected by the algorithm that are not present in the ground truth data, but the ones that are present, are very close to and many times, at the same locations as the ones detected by the algorithm.

In general, the method described above detects an average of 33% more boundaries as compared to the ground truth number of boundaries. This, however, does not seem to adversely affect the final segmentations when compared to the ground truth.

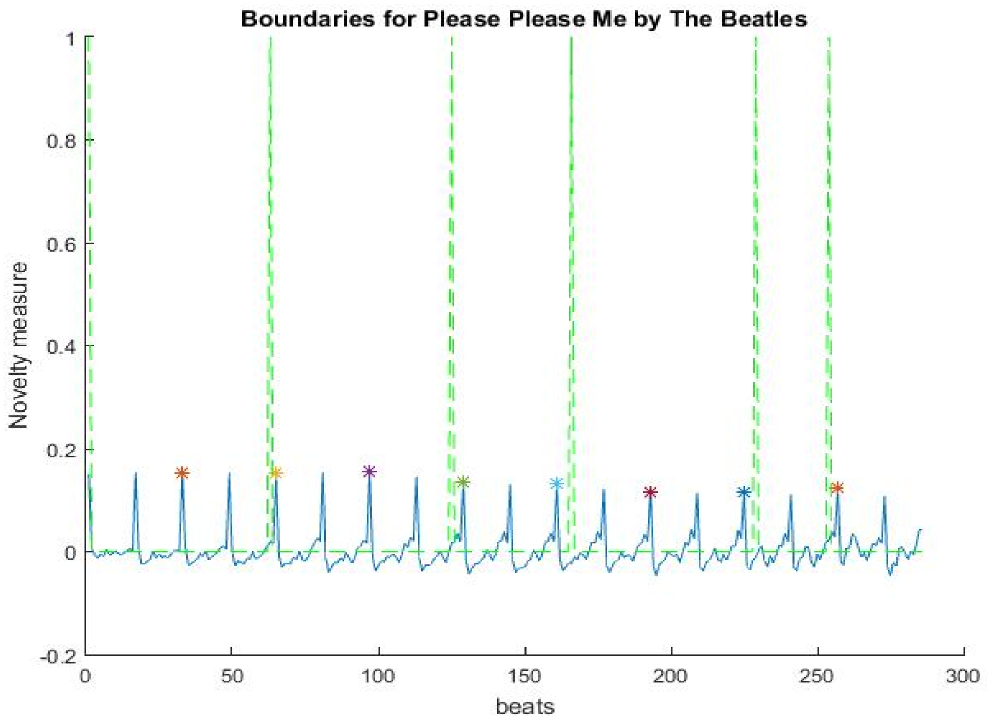

Figure 3.2 The peaks marked with a '*' are the boundaries located by the Algorithm. The green dotted lines indicate the ground truth boundaries

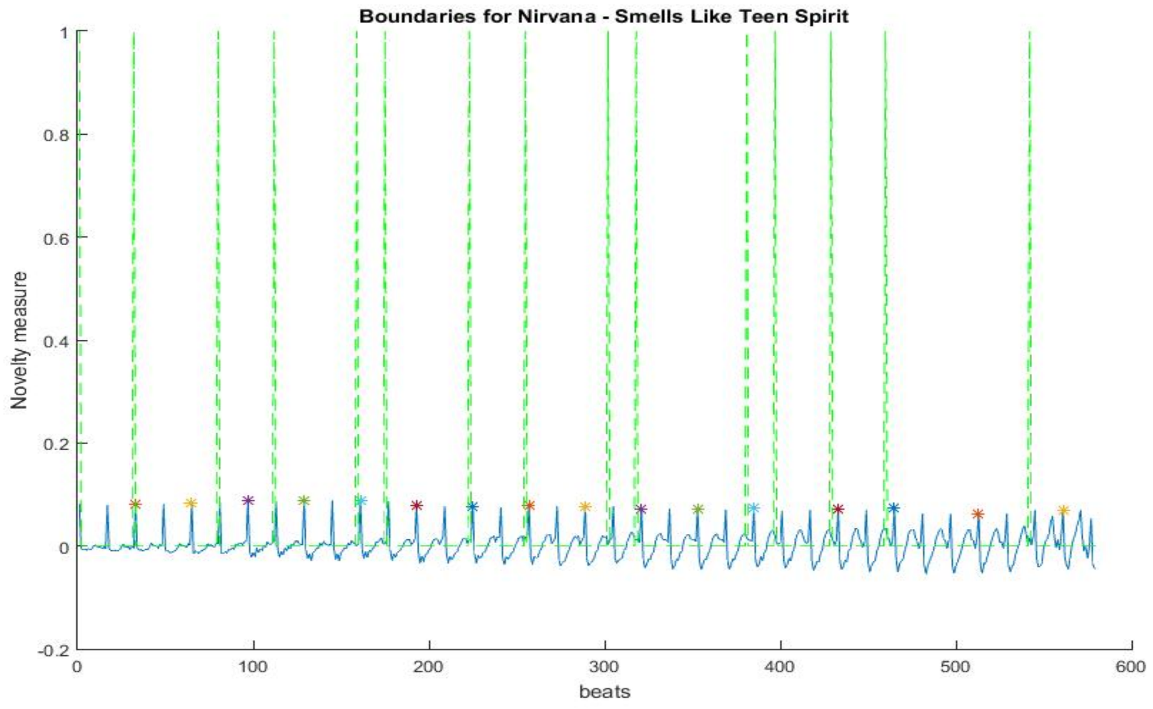
Boundaries for Nirvana - Smells Like Teen Spirit
Novelty measure
beats
1
0.8
0.6
0.4
0.2
0
-0.2
0
100
200
300
400
500
600

Figure 3.3

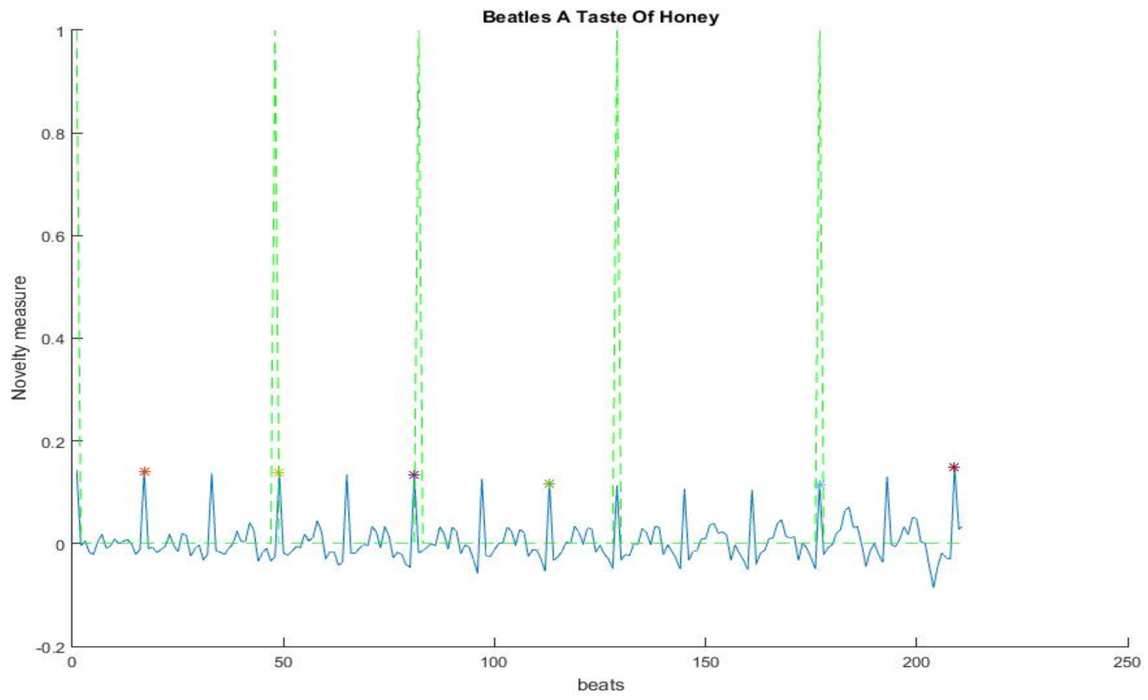
Beatles A Taste Of Honey
Novelty measure
beats
1
0.8
0.6
0.4
0.2
0
-0.2
0
50
100
150
200
250

Figure 3.4

Chapter 4

Clustering

This chapter deals with the clustering approaches employed in the experimentation of the project. Before setting up the clustering task, a sequence of histograms of the state labels in each beat is computed and these histograms are then clustered using four clustering models.

4.1 Histogram of States

I constructed a histogram of the states present in a grouping of d_{ml} beats. In [7], these histograms are created and are then clustered using temporal constraints. Constrained clustering, as developed in [13], uses some background knowledge of the points to be clustered and imposes a set of *must-link* and *cannot-link* constraints, while performing the clustering. The background knowledge is incorporated as a matrix $\mathcal{C}$ of binary constraints, where $\mathcal{C}(i,j) = 1$ if the points i and j must be grouped in the same cluster and $\mathcal{C}(i,j) = 0$ if the two points must lie in separate clusters. The authors of [7] use temporal continuity as a set of must-link constraints.

In this project, I chose a more direct, almost brute-force approach towards imposing these constraints and instead of using a matrix of binary constraints in the final clustering module, I used the location of the boundaries while creating the histograms and then clustered the histograms without any constraints.

The histograms were created using the following steps and pseudo-code:

1. Initialize the index H denote the matrix containing the sequence of the histogram of states and let b_k be the location of the k^{th} detected boundary and let $b_0 = 0$. Let n_b be the total number of detected boundaries. Let X be the $n \times 24$ matrix of state labels for each beat obtained from the state labeling in 2.2.3, where n is the number of beats in the song.

2.
```
for $k = 1 : (n_b - 1)$
  for $i = b_k + 1 : b_{k+1} - d_{ml}$
    for $j = 1 : 24$
      $H(i,j) = \sum_{l=i}^{i+d_{ml}} X(i,j)$
    end
  end
end
```

3.
```
for $i = (b_{n_b} + 1) : (n - d_{ml})$
    $H(ij) = \sum_{l=i}^{i+d_{ml}} X(i,j)$
end
```

The above process will yield a $m \times 24$ matrix H of the histogram of states with $m < n$. Thus, each histogram only contains the total number occurences of each state within windows of length d_{ml} that only lie within a given temporal region specified by the boundaries enclosing the region. This incorporates the temporal constraints of the data while creating the histograms itself. These histograms are then used for clustering.

4.2 Clustering the Data

The sequence of histograms obtained in the above section were clustered using the three different approaches:

1. k-means clustering

2. Hierarchical Agglomerative Clustering (HAC)

3. EP-means : A modified version of k-means that uses the Earth Mover's Distance [10]

Of these, the k-means and the agglomerative clustering approaches are well known, long-existing clustering algorithms in Machine Learning. However, the EP-means clustering algorithm is a newer algorithm that incorporates the Earth Mover's Distance to cluster probability distributions.

4.2.1 The Earth Mover's Distance (EMD)

The Earth Mover's Distance is a metric to measure the distance between two Probability Distributions. Informally, if the distributions are interpreted as follows:

Given a certain amount of dirt (*earth*), one can pile up the dirt over the region $\mathcal{D}$ in many different ways. The EMD between any two such piles is the minimum cost of turning one pile of dirt into the other; where the cost is assumed to be amount of dirt moved times the distance by which it is moved. Hence the name *Earth Mover's Distance*.

The histograms of states obtained in 4.1 can be normalized to reflect the probability of the occurrence of each state in a given window of beats. Hence, clustering these histograms using the Earth Mover's distance would entail a better notion of distance between them in the clustering process.

For one dimensional distributions like the histograms that have been obtained, the computation of the Earth Mover's Distance has a simple closed form solution [14]:

$$\rho_{emd}(h_i, h_j) = \sum_p \delta_{i,j}(p) \tag{4.1}$$

where h_i and h_j are p-dimensional histograms and

$$\delta_{i,j} = |cdf(h_i) - cdf(h_j)|$$

is also a p-dimensional vector and $cdf(h)$ denotes the vector containing the cumulative distribution of h.

4.2.2 The EP-means Clustering Algorithm

This algorithm can be thought of as a version of the k-means algorithm. However, it should be noted that the k-means algorithm has been traditionally defined using the squared Euclidean distance metric and hence, when updating the centers at each iteration so that the sum of the *within cluster point-to-center* distances is minimized, the centers turn out to be the *means* of the points in the respective clusters. Hence, the name *k-means*.

The EP-means algorithm [10] follows a similar approach, that is:

1. Choose an initial number of cluster centers, then assign points in the feature space to the nearest cluster using the distance between the points and the cluster centers.

2. Update the cluster centers according to the points assigned.

3. Assign the points to the nearest cluster using the new cluster centers

4. Repeat steps 2 and 3 till convergence.

In this project, I have used the *k-means++* technique of choosing the initial cluster centers, but by computing the EMD instead of the squared Euclidean distance. Once the initial centers are chose, the histograms are assigned to the cluster containing the nearest cluster center.

Now, in the next stage, which is the updating of the cluster centers, one cannot use the mean of the existing points in a cluster as the new cluster center since that minimizes the within cluster distance only if we are using the squared Euclidean distance. The steps for updating the center for the k^{th} cluster are listed below.
Let $F(x)$ denote the cumulative distribution value for bin x and $F_{h_i}(x)$ denote the cumulative distribution value for bin x of the histogram h_i. Let $\mathcal{P}_k$ be the set of all the points in the cluster k. Then, the center of cluster k can be updated as follows:

1. Start with $F(x)$ = 0

2. Find $\hat{x_0} = mean\{x|F_{h_i}(x) \geq 0\}$ for $h_i \in \mathcal{P}_k$. $\hat{x_0}$ will be the first bin of the cluster center with cumulative distribution value 0.

3. Find $\alpha_1 = min\{F_{h_i}(x)|F_{h_i}(x) > 0\}$ for $h_i \in \mathcal{P}_k$

4. Find $\hat{x_1} = mean\{x|F_{h_i}(x) \geq \alpha_1\}$ for $h_i \in \mathcal{P}_k$. $\hat{x_1}$ will be the second bin of the center with cumulative distribution value equal to α_1

5. At the t^{th} step, find $\alpha_t = min\{F_{h_i}(x)|F_{h_i}(x) > \alpha_{t-1}\}$ for $h_i \in \mathcal{P}_k$ and find $\hat{x_t} = mean\{x|F_{h_i}(x) \geq \alpha_t\}$. This will be the t^{th} bin of the center with cumulative distribution value α_t

6. Repeat step 5 till $\alpha_t = 1$.

It is very important that the histograms be normalized so that the value in each bin is in [0,1] to reflect a probability distribution. Once the centers are computed, the points can be assigned to their nearest clusters.

4.2.3 Sub-Optimality of the EP-means Algorithm for the given case

In [10], the authors have developed the EP-means algorithm for *Continuous Valued, one-dimensional probability distributions*. However, in this project, I have tried to extrapolate the idea to discrete set of probability distributions that are the histograms of states. This yields sub-optimal, particularly noisy results, mainly due to non-discrete values that can be obtained for the α_ts in the computation of the centroids because these are averages. Rounding them off empirically yields noisy results as illustrated in 4.1

Still, it can be seen from the clustering output of the EP-means algorithm that, if smoothed efficiently, it could replicate the k-means and the HAC outputs very closely.

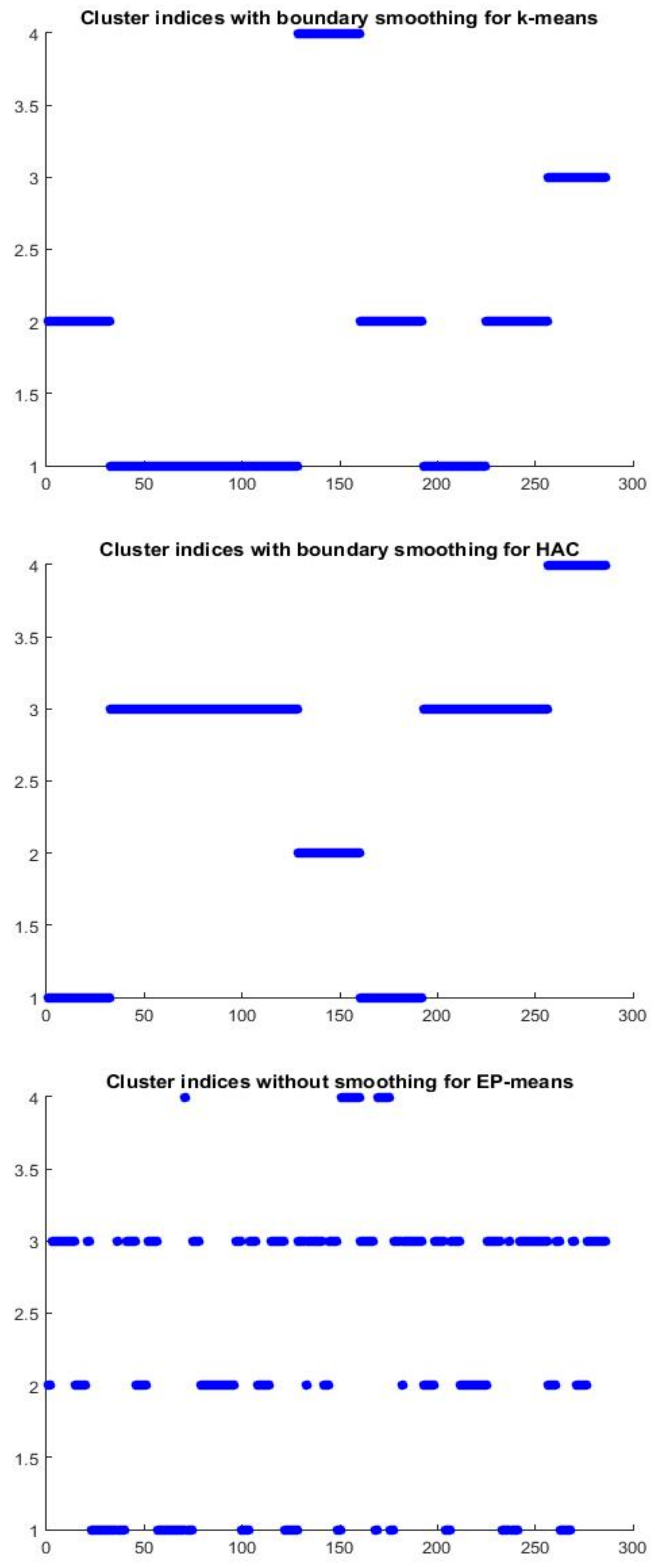

Figure 4.1 The smoothed versions of the cluster indices for k-means and HAC undergo similar transitions (similar structure). For the EP-means clustering, there is a huge presence of cluster number 3. Smoothing this would lead to most of the beats in the song being labeled as belonging to segment 3. (The *x-axis* contains the beat number of the song)

Chapter 5

Testing and Evaluation

5.1 Data Set

I used the publicly available ground truth segmentations of Beatles' songs available [1] under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 License, that were first created by Allan Pollack in his *"Notes On..."* series and have been corrected at the Audio Research Group, Institute of Signal Processing at the Tampere University of Finland and Pompeu Fabra University simultaneously. I also tested the algorithm on a few non-Beatles songs that were used for testing in [7], the ground truth segmentations [2] for which are also available. In all, the system was tested on a total of 70 songs for whom the ground truths were available in the aforementioned data sets.

5.2 Evaluation Metric

I used the metric proposed in [7], called the Pairwise F-measure, sometimes abbreviated as pwf or simply, F. This metric has been most widely used in the evaluation of the structural segmentation task. The metric is computed as follows:
Let $\mathcal{P}_s$ be the set of similarly labeled pairs of beats in a given song according to the artificial system. Let $\mathcal{P}_r$ be the set of similarly labeled pairs of beats in a given song according to the reference segmentations. Define two quantities, $precision$ and $recall$, as follows:

$$precision = \frac{|\mathcal{P}_s \cap \mathcal{P}_r|}{|\mathcal{P}_s|}$$

[1]`http://www.cs.tut.fi/sgn/arg/paulus/structure.html`
[2]`http://www.elec.qmul.ac.uk/digitalmusic/downloads`

$$recall = \frac{|\mathcal{P}_s \cap \mathcal{P}_r|}{|\mathcal{P}_r|}$$

Then the pwf is given as:

$$pwf = \frac{2 \times precision \times recall}{precision + recall} \tag{5.1}$$

Due to the above formulation, $0 \leq pwf \leq 1$, with a larger value of pwf implying a better segmentation.

5.3 Results

As seen in Section 4.2.3, the use of the EP-means algorithm for discrete histograms could yield results that are very sub-optimal. Hence, I tried the conventional *k-means* and *Hierarchical Aglomerative Clustering* approaches to try to cluster the histograms created in Section 4.1. I used the number of total segments as input from the ground truth data and computed the average pwf for each clustering method. These results are shown in Table 5.1. I have also provided the average of the pwf obtained for the 25 songs on whom the algorithm performed the best in Table 5.2 and the best case results in Table 5.3. One notable thing that can be seen from the output for the tracks which yield low pwf values is that; since the features are chords, a song containing vocals that sound identical to the chord on which they are set will usually cause the deletion of one or more segments in the final output even though the actual number of segments are fed in the input. This defect, however, is not pronounced in most of the songs. It should be noted that songs like ”Dig It” by *The Beatles* yield better results (*pwf* of 0.87 for both k-means and HAC) than the ones mentioned in Table 5.3, but were not considered as being the best results due to the simplistic nature and short duration of the songs (usually just a prolonged segment with a small, slightly different segment in the end).

Clustering Method	**Average Precision**	**Average Recall**	**Average *pwf***
k-means	0.44	0.42	0.51
HAC	0.49	0.57	0.47

Table 5.1 Average of the Evaluation metrics for all the 70 songs

Clustering Method	**Average Precision**	**Average Recall**	**Average *pwf***
k-means	0.48	0.40	0.63
HAC	0.58	0.59	0.59

Table 5.2 Average Evaluation metrics for the 25 songs with best results

Clustering Method	**Precision**	**Recall**	***pwf***	**Song Title**
k-means	0.56	0.44	0.75	*Twist and Shout- The Beatles*
HAC	0.70	0.62	0.79	*There's a Place- The Beatles*

Table 5.3 Evaluation metrics for the songs with the best results

Chapter 6

Conclusion and Possible Future Development

The results obtained for the best performing songs and the 25 best performing songs indicate that the features used and the constraints imposed during the creation of feature vectors could provide some insight to existing models and scope for development of structural segmentation models:

1. (a) A lot of existing literature and methods have developed and used the idea that in order to be able to develop a notion of difference between different segments, the transition of the states to which the frames in a song belong to and the probability of the occurrence of these states in a given window of frames can be taken into consideration and in fact, gives fruitful results.

   By representing these states-which are generally generated using Hidden Markov Models-as elementary versions of a chord, and seeing that they do not perform very poorly, useful insight can be gained into the physical meaning of these states.A way of being able to directly compare the performance of the *chord states* with the HMM states in the same setting could help determine the strength of the aforementioned claim.

   (b) Better chord detection and key detection techniques can be used in the initial phase of the module to obtain better representations of the sequence of states; thereby leading to better clustering.

2. Constrained clustering is a type of semi-supervised learning that has many practical uses. The technique of incorporating the constraints while creating the feature vector, as has

been done in this project, and then clustering these features in an unsupervised setting can be developed and investigated further to possibly make the clustering and segmentation better.

3. The development of a more robust, discrete version of the EP-means algorithm used in this project can enable the use of the Earth Mover's Distance as a distance metric for the clustering of discrete histograms, possibly aiding in the performance of models that use histogram clustering.

LIST OF REFERENCES

[1] N. C. Maddage, C. Xu, M. S. Kankanhalli, and X. Shao, "Content-based music structure analysis with applications to music semantics understanding," in *Proceedings of the 12th annual ACM international conference on Multimedia*. ACM, 2004, pp. 112–119.

[2] L. Lu, M. Wang, and H.-J. Zhang, "Repeating pattern discovery and structure analysis from acoustic music data," in *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*. ACM, 2004, pp. 275–282.

[3] M. Goto, "A chorus-section detecting method for musical audio signals," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 5. IEEE, 2003, pp. V–437.

[4] K. Johansson, "The harmonic language of the beatles," *STM-Online*, vol. 2, no. 1999, 1999.

[5] R. W. J. Bello, "Identifying repeated patterns in music using sparse convolutive non-negative matrix factorization," 2010.

[6] R. Chen and M. Li, "Music structural segmentation by combining harmonic and timbral information." *ISMIR*, pp. 477–482, 2011. [Online]. Available: http://ismir2011.ismir.net/papers/PS4-1.pdf

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

[8] E. Peiszer, T. Lidy, and A. Rauber, "Automatic audio segmentation: Segment boundary and structure detection in popular music," *Proc. of LSAS*, 2008.

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

[10] K. Henderson, B. Gallagher, and T. Eliassi-rad, "Ep-means : An efficient nonparametric clustering of empirical probability distributions," 2015.

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

[12] T. Giannakopoulos and A. Pikrakis, *Introduction to Audio Analysis: A MATLAB® Approach*. Academic Press, 2014.

[13] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl *et al.*, "Constrained k-means clustering with background knowledge," in *ICML*, vol. 1, 2001, pp. 577–584.

[14] Y. Rubner, C. Tomasi, and L. J. Guibas, "A metric for distributions with applications to image databases," in *Computer Vision, 1998. Sixth International Conference on*. IEEE, 1998, pp. 59–66.