# Improving Acoustic Features for Structural Segmentation of Music

João Afonso Raposo
Instituto Superior Técnico

***Abstract*—Structural segmentation of music is the task of identifying the temporal boundaries of each structural segment within a song and assigning these a label, using the same label for repeated segments. Although several structural segmentation algorithms have been developed in the last two decades, not much attention has been given to the underlying feature extraction model, specially to the combination of multiple features. We are mainly focused on the feature extraction model, while making use of an already available segmentation algorithm. We propose the creation of two embeddings, Generalized Canonical Analysis (GCCA) and identity vectors (i-vectors), to be used as the input feature vectors for the structural segmentation algorithm. Results from both approaches reveal improvements regarding some of the evaluation metrics, when compared with three standard feature models, namely Mel Frequency Cepstral Coefficients (MFCC), Constant-Q Transform (CQT) and Chromagram (Chroma).**

## I. Introduction

Structural segmentation of music is the task of identifying the temporal boundaries of each structural segment (the instants where it begins and ends) within a song and assigning these a label, using the same label for repeated segments. Perceptually, each segment has a homogeneous structure, which shifts significantly on the transition between different segments. This homogeneous structure is mostly encoded in three musical dimensions, namely: rhythm, timbre and tonality. A structural segment is usually representative of a given musical section, such as a chorus or a verse. There can be multiple segments, within a song, of the same musical section. The labelling of the segments is often made by a generic notation, such as *ABCBCDBDE*, where each letter corresponds to any musical section, thus putting aside the semantic requirement of classifying meaningful musical sections.

Broadly speaking, to perform segmentation, one relies on the intrinsic repetitive nature of music. This is what allows us to identify patterns and swift changes in music, making it possible to extract the desired segments. Most of the available systems make use of data sets consisting mainly on popular music genres (rock, pop, etc.), where the aforementioned characteristics are more evident and more easily identified.

There are several applications for our task, namely audio thumbnail generation and the ability to skip between the beginning and ending of the different segments of a song. Besides this, a good structural segmentation is helpful to perform other tasks within MIR (Music Information Retrieval), such as Music Recommendation and Chord Extraction [1].

### A. Goals

Feature extraction plays a big role on every MIR task, and Structural Segmentation is no exception. Although this task has been around for some years, over which several algorithms have been developed, few attention has been given to the feature extraction model. Moreover, most methods developed so far to combine multiple features have failed to achieve significant improvements in performance. Hence, this document focuses on exploring the impact of the feature extraction model, leaving aside most of the concerns regarding the structural segmentation algorithm. We will make use of a plugin called QMSegmenter [2] (based on a state of the art structural segmentation algorithm [3]) and ”feed” it with embeddings that, to our knowledge, have never been used for our task. These embeddings are extracted using two different models, namely GCCA and i-vectors. Our goal is to improve and analyze the performance impact by using the aforementioned models when compared with standard baseline features, namely MFCC, CQT and Chroma.

### B. Document structure

This document is organized as follows: Sections II and III revise several evaluation metrics, feature models and algorithms developed throughout the years to perform our task; Section IV details our experimental setup by fist describing our approach and then our dataset, which features are used and finally how do we evaluate our results; Section V compares, discusses and analyzes the results obtained from the different experiments made; Section VI concludes this document by summarizing the most relevant details of this work.

## II. Background

Feature extraction plays a major role throughout the overall performance of a model. There are several algorithms available in literature for the the extraction of audio features. Most of these try to capture three major musical dimensions, namely rhythm, timbre and tonality. Some of the most widely adopted feature extraction models are explained in the following subsections.

### A. Mel-frequency cepstral coefficients (MFCC)

The underlying idea is based on human perception experiments. It has been found that the human ear predominantly focuses on some frequency components, meaning that its perception follows a non-linear frequency spectrum. Roughly

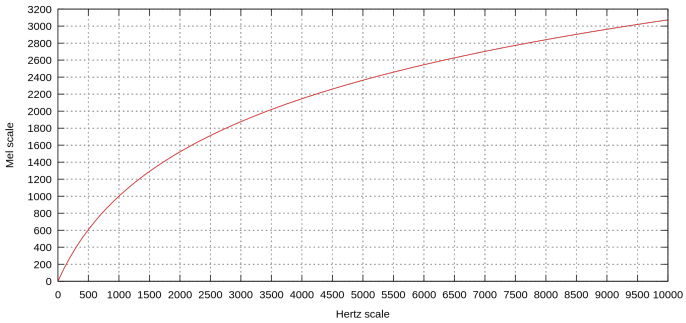


Fig. 1: Mapping of the frequency of a signal from hertz ($x$ axis) to mels ($y$ axis).

speaking, low frequencies are more relevant than high frequencies. To address this, a *mel* scale was created, mapping the frequency spectrum into the mel spectrum in a way to model the hearing perception of humans. Such mapping is approximately logarithmic (fig. 1).

The MFCCs extraction is comprised of 4 main steps:

1) Compute the Discrete Fourier Transform (DFT) of a signal
2) Map the power spectrum into the mel scale as explained above
3) Take the log of the resulting power spectrum
4) Apply Discrete Cosine Transform on the previous spectrum

The MFCCs are the amplitudes of the final cepstrum. MFCCs are considered to be closely related to the timbre dimension, making its coefficients specially good at capturing dis(similarities) between different instruments playing the same frequencies.

### B. Chromagram (Chroma)

The underlying idea it that humans perceive two frequencies shifted apart by an octave from each other as similar. Let us consider the equal-tempered scale (Western's most commonly adopted scale since the 18th century), which divides each octave into 12 different parts or chroma values, namely {C, C#, D, D#, E, F, F#, G, G#, A, A#, B}. A pitch class is the set containing all pitches sharing the same chroma, which means that two notes shifted apart by n $\in \mathbb{N}$ octaves share the same chroma. Chroma features are then a set of 12 coefficients, where each coefficient represents the aggregated information of its corresponding chroma into a single value representing the intensity of the corresponding pitch class. The aggregation can be performed by any given function such as the average.

### C. Constant-Q Transform (CQT)

Many approaches rely on DFT which, roughly speaking, decomposes the audio signal into equally-spaced frequencies and provides the corresponding intensities or amplitudes. However, some systems choose to adopt the CQT instead. CQT can be seen as a sequence of logarithmically-spaced filters, where the width of each filter $k$ is a multiple of the previous $(k-1)$ filter:

$$\delta f_k = 2^{1/n} \cdot \delta f_{k-1} \quad (1)$$

| | MFCC | Chroma | Both |
|---|---|---|---|
| F-Measure (%) | 58.6 | 50 | 53.6 |
| Precision (%) | 58.1 | 46.5 | 49 |
| Recall (%) | 61.9 | 52.2 | 55 |

TABLE I: Results from the "TUT Beatles" dataset, consisting of 174 songs from The Beatles.

| | MFCC | Chroma | Both |
|---|---|---|---|
| F-measure (%) | 59 | 61 | 63 |

TABLE II: Results from a dataset consisting of 180 songs from The Beatles.

where $\delta f_k$ is the bandwidth of the $k$-th filter and $n$ is the number of filters per octave.

The logarithmic spacing of bins of CQT is well suited for musical data since it provides a higher frequency resolution for low frequencies using many bins, while using fewer bins for higher frequencies, thus better modelling the hearing perception of humans. Moreover, the harmonics of musical notes form characteristic patterns of the timbre of the instrument being played, easing the identification of different instruments in a given signal. Compared to the DFT, CQT is slower and more complex to implement, although there are already techniques available to overcome some of these limitations.

### D. Multiple features

There are also some systems that tried to integrate multiple feature models in order to develop a more robust solution [4]. Such integration is far from trivial, and if not properly modelled can lead to worse results than using just one feature model. In [5], the authors provide results with the use of separate feature models (MFFC and Chroma) and the combination of both:

As can be observed from the table depicted above, the use of MFCC alone provided better results than the the combination of MFCC and Chroma features.

However, there are already some systems that managed to increase their performance by combining multiple features. In [6], combining Chroma and MFCC features using Non-Negative Matrix Factorization (detailed in **??**) led to a slight increase on performance:

### E. GCCA

CCA is a statistical method that relies on cross-covariance matrices to infer information. Having two sets of variables $X = (x_1, x_2, ...x_n)$ and $Y = (y_1, y_2, ...y_m)$, CCA will find linear combinations of $X$ and $Y$ that maximize the correlation between them. The cross-covariance matrix can be defined as $\sum_{XY} \quad (n \times m)$, where the entry $(i, j)$ corresponds to the covariance $cov(x_i, y_j)$. The covariance $cov(x, y)$ between any two variables $x$ and $y$ is a measure of the joint variability of $x$ and $y$. Its magnitude is positive when the variables have the same behaviour (increases and decreases in $x$ correspond to increases and decreases in $y$, respectively) and negative when the variables have opposite behaviours (increases and decreases in $x$ correspond to decreases and increases in $y$,

respectively). GCCA is identical to CCA, except it can be applied to $n \in \mathbb{N}$ sets of variables rather than just 2.

### F. i-vectors

The concept of *i-vectors* (identity vectors) [7] was developed based on Joint Factor Analysis (JFA) [8] and was initially proposed for the speaker verification task. One of its other applications is Acoustic Event Detection (AED) which is the task of identifying acoustic events. Regarding the speaker verification task, the JFA model assumes a Gaussian distribution on the speaker and channel dependent super-vectors (stacked means of the mixture components) in which most of the variance of the population can be described by a small number of hidden variables referred as channel/session and speaker factors [9]. Hence, the channel and speaker factors are assumed to only model the channel and speaker effects, respectively, meaning that the values of the speaker factors remain constant for different recordings of the same speaker while the channel factors vary from one recording to another. However, it was later observed that the channel dependent super-vector also models the speaker factors. To account for this, the i-vectors were proposed as a solution where there is no distinction between speaker and channel variabilities but rather a single low dimensional total variability space. Roughly speaking, an i-vector of a segment (set of frames) is extracted by estimating the differences between the posterior model and the original model after observing the segment, and concatenating these differences into a super-vector:

$$\mu - m = Vy, \tag{2}$$

where $\mu$ and $m$ are the posterior and original model super-vectors, respectively, $V$ is the eigenmatrix and $y$ is the i-vector.

The original (or background) model is a Gaussian Mixture Model (GMM) with C components assumed to describe the whole audio features space. This model is called an Universe Background Model (UBM) and its corresponding super-vector $m$ can be formed by concatenating the mean vectors of its Gaussian components. The posterior model is derived after observing an audio segment and its corresponding super-vector $\mu$ can be computed in the same way as the original model. It is claimed that an audio segment is highly related to only a subset of the Gaussian components, reflected by the posterior model *responsabilities* of certain mixtures for generating the resulting feature vectors. Hence, it is expected that the posterior mean statistics in the super-vector space are sparse. Thus, one can take advantage of such *sparseness* by performing dimensionality reduction (one of the main goals of using i-vectors) and increase the discriminative power of the i-vectors.

## III. Structural Segmentation of Music

### A. Evaluation Metrics

Several evaluation metrics have been developed for our task, such as mutual information between sequences of segment labels for each frame [10], a segment-wise Hamming distance [10] and a string edit distance between sequences of labels for each segment [11][12]. It is worth noting that the ground truth annotations available for this task are rather ambiguous, meaning that annotations from different musical experts are not always unanimous, leading to a not so consistent ground truth labelling of the segments within a song. Besides this, different models often use and publish results originated from non-standard datasets or a subset of these, thus hindering comparisons. Collective efforts have been made to mitigate these issues, such as the framework Music Information Retrieval Evaluation eXchange (MIREX) managed by a devoted community, with the purpose of sharing and comparing systems that solve tasks within MIR through standard procedures (standard datasets and evaluation metrics). According to MIREX, one can divide evaluation into two major evaluation measures that address the main aspects of our task. These evaluations measures are Boundary Retrieval and Frame Clustering. The former addresses the timestamps of the boundaries of the segments, considering a boundary to be correctly identified if it is within a given temporal threshold (e.g. 0.5 seconds [13] or 3 seconds [3]) regarding the corresponding boundary of the ground truth. Given the number of correctly identified boundaries $p$, the number of falsely identified boundaries $f$ and the number of total ground truth boundaries $t$ we can calculate Precision, Recall and F-measure as:

- Precision: $P = \frac{p}{p+n}$
- Recall: $R = \frac{p}{t}$
- F-measure: $F = \frac{2 \cdot P \cdot R}{P+R}$

Two more metrics are used regarding Boundary Retrieval, namely: Median Guess-to-True (MGTT) and Median True-to-Guess (MTTG) which address, respectively, the median deviation (measured in seconds) from the extracted boundaries to the closest ground truth boundary and vice-versa.

Frame Clustering addresses the correctness of the labels for each frame of the segmentation produced by a given system regarding a ground truth: considering $P_S$ and $P_{GT}$ to be the the sets containing the pairs of frames with the same label for the system and ground truth, respectively, we can calculate pairwise Precision, Recall and F-measure as:

- Precision: $P = \frac{|P_S \cap P_{GT}|}{|P_S|}$
- Recall: $R = \frac{|P_S \cap P_{GT}|}{|P_{GT}|}$
- F-measure: $F = \frac{2 \cdot P \cdot R}{P+R}$

### B. Hidden Markov Models (HMM)

HMMs are extensively used in several systems proposed to solve this task. A HMM is a statistical Markov model to be applied on a system assumed to be a Markov process containing unobserved (hence *hidden*) states. A Markov process is a random process satisfying the Markov property which states that, considering a sequence of events (observations), the probability of each event is only dependent on the single previous event.

Roughly speaking, a HMM is a set of hidden states with given probabilities of transiting between each other. Such probabilities are learnt by training the model over a sequence of observations. A widely adopted training algorithm is the Expectation Maximization (EM) algorithm.

The probability of a sequence of states $X = (x_1, x_2, ..., x_n)$ given a sequence of observations $Y = (y_1, y_2, ..., y_n)$ is given by:

$$P(X|Y) \propto P(y_1|x_1) = \prod_{i=2}^{n} P(y_n|x_n)p(x_n|x_{n-1}) \quad (3)$$

where $P(y_n|x_n)$ is the likelihood of observing $y_n$ while in state $x_n$ and $p(x_n|x_{n-1})$ is the probability of transiting between states $x_n$ and $x_{n-1}$.

Regarding our task, the usual approach is to consider each observation as the feature vector of a frame. Taking all frames as the sequence of observations one can train a HMM for a song. Afterwards, it is applied the Viterbi algorithm which finds the most probable sequence of hidden states given a sequence of observations.

Ideally, we would want that each state would be able to represent each segment label so that after the Viterbi algorithm application we would already have the final segmentation. However tempting this approach might be, it does not work well in practice, producing overly temporal fragmented results. One reason for this is that the states tend to model low-level musical elements such as individual sounds, rather than longer musical elements.

To circumvent this issue, a HMM with a fairly large number of states (e.g. 40,60,80) can be created as in [3][10][14]. This implies one more step to size down the number of states to the number of underlying musical sections in the song so that it can produce a final segmentation.

This step is usually addressed by creating histograms of the states. Using a sliding window with size $l_1$ and a hop-size $l_2$, histogram vectors $h$ are created where for each state $i$, $h(i)$ is a coefficient representing the number of states $i$ observed within its window. These histograms are then clustered to determine the segment they belong to. Such clustering might still become too fragmented, hence failing to model the desired segmentation.

### C. Constrained clustering

In [3], the clustering is constrained in a way to better model the temporal continuity of music, avoiding overly temporal fragmented segmentations. They define a set of observations (the set of histograms of states $X = \{x_1, x_2, ...x_n\}$) and a corresponding set of hidden labels $Y = \{y_1, y_2, ...y_n\}$ representing the segment label for each histogram. Thus, $y_i$ is the the cluster index of $x_i$ taking its value from a set of cluster indices $S = \{1, ...K\}$, where $K$ is the number of segment labels. Observations are constrained to belong to the same cluster according to a set of observable constraints $C = (c_{12}, c_{13}, ...c_{n-1n})$ where $c_{ij} = 1$ forces the pair $(x_i, x_j)$ to share the same cluster and $c_{ij} = 0$ means that the pair is unconstrained. A parameter $d_{ML}$ acts as a threshold for the minimum neighbourhood size, and hence, the minimum length of the segments. Such length differs significantly between different songs, suggesting the need of learning/estimating $d_{ML}$ for each song. Results with $d_{ML} = 8$ and $d_{ML} = 16$ are shown in table III, where the columns *Recent* and *Beatles* refer to 30 Beatles songs and 30 pop/rock/rap songs from 1980-2008, respectively.

| | | Recent | Beatles | Recent+Beatles |
|---|---|---|---|---|
| $d_{ML} = 8$ | label pairwise f-measure (%) | 56.7 | 58.8 | 54.7 |
| | boundary f-measure (%) | 66.2 | 58.1 | 62.2 |
| $d_{ML} = 16$ | label pairwise f-measure (%) | 60.5 | 60.4 | 60.3 |
| | boundary f-measure (%) | 64.0 | 54.0 | 59.0 |

TABLE III: Results regarding Frame Clustering f-measure (detailed in section III-A).

### D. Self-similarity Matrix (SSM)

Another popular method used in several systems (e.g. [15][16][1][17]) is the Self-similarity matrix (SSM) of a set of feature vectors of a song.

A SSM is a symmetric matrix where each index reflects how similar each pair of data points of a data series is. Similarity can be defined by different measures such as spatial distance and correlation coefficients.

To build a SSM, one must first partition the data into feature vectors. For this, a feature extraction model takes place beforehand originating a sequence of $n$ feature vectors $V = (v_1, v_2, ..., v_n)$ from the input data series. Thus, the SSM $S$ is constructed by computing the similarity of all possible pairs in $V$:

$$S(j, k) = s(v_j, v_k) \quad j, k \in \{1, ..., n\} \quad (4)$$

where $s(v_j, v_k)$ is the similarity *score* of the pair $(v_j, v_k)$ for a given similarity function. Independently of the similarity function chosen, large similarity values (relative to the domain of the function) emerge from comparing two similar objects (feature vectors) and vice versa.

The graphical representation of a SSM (fig. 2) can provide a rather intuitive understanding of the similarities present in the data series being analyzed. Let us consider identical pairs to display a white pixel on the matrix while the most dissimilar ones display a black pixel. Values in between display a gray colour with a lighter/darker pitch corresponding to its closeness to the maximum/minimum values, respectively. By looking at the output matrix, light pixels and diagonal light stripes might be present.

Single light pixels or squares arise from a pair of similar feature vectors. The most meaningful phenomena for our task is the presence of diagonal stripes parallel to the main diagonal. These stripes are potential repeated segments, representing a repeated sequence of feature vectors. A repeated segment such as a chorus appearing multiple times in a song, would originate light diagonal stripes at the corresponding time intervals.

*1) Novelty Function:* One popular approach designed in [15] and later adopted by other systems such as [17], is to originate a novelty measure function $\eta$ over time from $S$ by applying a filter known as a kernel. For explanation purposes, let us suppose a very simple $2 \times 2$ kernel $C$ (thus comparing a pair of single frames at a time) which can be decomposed into two terms:

$$C = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad (5)$$

The first term measures the self-similarity of either regions while the second measures their cross-similarity. The difference of both terms estimates the novelty score at the center

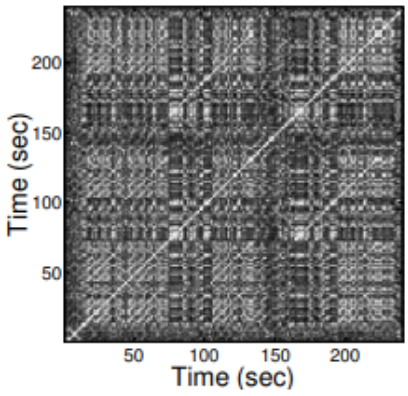


Fig. 2: SSM

(boundary between regions). Thus, this score is high when the two regions have high self-similarity and low cross-similarity.

By correlating a kernel $C$ with $S$ we can obtain a measure of novelty. It is as if we make $C$ "slide through" $S$, acting as filter that produces high novelty scores when crossing contrasting similarity regions (regions containing sudden changes of similarity scores) and low novelty scores for uniform regions. The Novelty measure $\eta(i)$ is computed as follows:

$$\eta(i) = \sum_{m=-L/2}^{L/2} \sum_{n=-L/2}^{L/2} C(m,n)S(i+m, i+n) \quad (6)$$

where $L$ is the width or lag of the kernel. The parameter $L$ has a significant impact on the novelty measure function. Small kernels narrow down the novelty detection to short-time fragments, like single notes, while large ones are able to detect novelty at a higher scale, like musical transitions and symphonic movements. We then need to choose an appropriate kernel width, large enough to be able to detect the transitions of segments while neglecting single notes onsets. In [15], they define a $64 \times 64$ checkerboard kernel with a Gaussian Taper for their experiments, but provide no quantitative results. They also describe an easy way to construct larger kernels by computing the Kronecker product of the aforementioned $2 \times 2$ kernel C with a matrix of ones:

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} \quad (7)$$

### *E. Deep Learning Methods*

The limited amount of annotated (hence, training) data for our task makes it impractical to utilize supervised deep learning methods. For this reason, it was recently developed an approach for solving our task based on unsupervised audio feature embeddings [18], avoiding the need to handle annotated data and thus making it possible to work with much larger datasets. Such approach consists on an unsupervised training of CNNs (Convolutional Neural Networks) in order to obtain a more meaningful and discriminative representation of the features for the task. Considering that musical segments

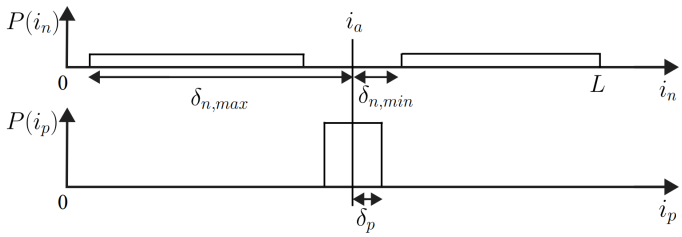


Fig. 3

form contiguous sections in a song and that each distinct segment occurs for a short duration, the authors developed a sampling framework that relies on the time proximity between frames to determine if they belong to the same segment. Given this premise, features sampled close together or at a minimum distance apart relative to an anchor beat index are labelled as positive and negative examples, respectively. The anchors are chosen uniformly from the set of previously extracted beat indices in a song, $\{0..L-1\}$, where $L$ is the total number of beats in a song. The positive and negative beat indices, $i_p$ and $i_n$, are then sampled uniformly from regions within some given positive and negative thresholds, respectively. The uniform probability distributions for the positive and negative examples given the thresholds $\delta_p$ and $\delta_n$, respectively, are depicted in fig. 3. The loss function associated to the CNN is the triplet-loss function [19] which, roughly speaking, simultaneously minimizes/maximizes the distance between the anchor and the positive/negative examples. Ideally, positive examples will belong to the same segment as the anchor, while negative examples will belong to distinct segments. Hence, once the CNN is trained, features could be transformed into a space where distinct clusters (with respect to a given distance metric) would represent distinct structural segments. Although there is no exact labelling of the data, the sampling technique described above provides a rather good alternative. To test the aforementioned embeddings, the authors made use of a simple approach developed by [15] and detailed in III-D. The corresponding SSM is computed as follows:

$$S[i,j] = ||f(x_i[q,k]) - f(x_j[q,k])||_2^2 \quad (8)$$

where $x_i[q,k]$ and $x_j[q,k]$ correspond to beat synchronous CQT frames centered at beats $i$ and $j$ and $f$ is the transformation function of the CNN. To detect the boundaries, a checkerboard kernel is convolved along the diagonal of the SSM, producing a novelty function. Finally, values exceeding a given threshold are selected as boundaries.

## IV. Experiments

### *A. Approach*

To the best of our knowledge, few research has been made in the last years for solving our task, probably because new methods provided no significant improvements on performance nor new ways of approaching the task. We believe that a learned combination of multiple features might lead to better results. Although some work has been made to combine multiple features (detailed in section II-D), the methods provided no

considerable increase on performance. As an example on how naive methods of combining multiple features often cannot produce better results, let us consider the most trivial example where we simply concatenate two different feature vectors into one single vector: we are neglecting the fact that the different features might have values belonging to completely different ranges and the fact that increasing the number of dimensions might lead to a very sparse final space.

Similarities between the same type of segments and dissimilarities between different ones originate from several aspects of the audio, such as rhythm, timbre and pitch. As stated in section II, different feature models capture these characteristics differently, making some of them better than the others in some scenarios. Attending to the homogeneity within segments, we might better identify the boundaries and the similarities between segments by taking into account the joint variability of multiple features, instead of just one. This suggested the creation of a multi-feature embedding by creating a common space between previously extracted acoustic features with GCCA (detailed in section II-E), using well-known features such as MFCCs and CQT, in order to produce a possibly better representation of music data for our task.

In [20], the authors proposed a blind audio segmentation approach for AED using i-vectors. In our scenario, the boundaries of each structural segment could be the events to be identified. Unlike most previous approaches using i-vectors, they segmented the audio stream into equal-length chunks or segments. Then, one i-vector per segment is extracted and used as the input feature vector for the segment. Finally, they train a binary SVM classifier for each label or event (”dog”, ”speech”, ”beep”, ...) to determine whether a segment contains the respective event. This suggested the possibility of using i-vectors for our task. However, since our labels have no semantic meaning outside the context of each song, we shall disconsider classifying the segments. Hence, we made experiments using i-vectors directly as the input feature vectors for the structural segmentation algorithm, for possibly better capturing changes/similarities in the audio stream.

Having the baseline feature models (MFCC, CQT and Chroma) and the feature representations aforementioned (GCCA multi-feature embeddings and i-vectors), we will use them as the input feature vectors for the software QMSegmenter [2] to perform the structural segmentation algorithm. Experiments and respective results will be compared and analyzed in section V.

### B. Dataset

The chosen dataset consists of a collection of 123 songs belonging to 9 different albums (*Abbey Road, A Hard Day's Night, Beatles for Sale, Help, Let it Be, Please Please Me, Magical Mistery Tour, Revolver, Rubber Soul*) from the rock band *The Beatles*. We chose this dataset because it is widely used in most of the experiments for this task and due to the pop/rock genre having a well defined structure.

### C. Features

Three well-know acoustic features, detailed in section II, were used: MFCC, Chroma, and CQT. In order to produce directly comparable results with the baseline, the features were extracted with a plugin belonging to the same software used for the structural segmentation.

Moreover, projections of pairs of features ({MFCC, Chroma} and {MFCC, CQT}) using GCCA were used as embeddings belonging to a common space between the input features. Specifically, for each pair, we project each of the features into the common space, thus producing two embeddings, and then we take the mean value of the embeddings as the final embeddings. The pair {CQT, Chroma} was disconsidered for the nature of the features being too similar.

As stated in section IV-A, attending to the fact that we have no classes for the labels of the segments but simply generic labels, we did not perform any training nor classification of the labels. Hence, we use the i-vectors (using MFCC and CQT features) directly as the final representation of the input feature vectors for the segmentation algorithm.

MFCC features were extracted with frame/hop sizes of 0.6 and 0.2 seconds, respectively. The number of coefficients was set to 20: 19 plus the $c_0$ coefficient (which indicates the average power of the input signal).

CQT features were extracted with frame/hop sizes of 0.6 and 0.2 seconds, respectively. The number of bins was set 64 (8 bins per octave using 8 octaves). PCA (Principal Component Analysis [21]) was then applied to reduce the number of dimensions from 64 to 20.

Chroma features were extracted with frame/hop sizes of 0.4 and 0.2 seconds, respectively. 8 octaves were considered and the number of ”chromas” per octave was set to 12.

Regarding GCCA, we used MFCC, CQT and Chroma features with the same parameters used for the baseline. The model was trained song by song, separately. We varied the number of dimensions of the final space from 5 to 20 for {MFCC, CQT} and from 3 to 12 for {MFCC, Chroma} attending to the fact that the number of dimensions of the final space must be less or equal to the number of dimensions of the view with fewer dimensions. The maximum values of 20 and 12 for the pairs {MFCC, CQT} and {MFCC, Chroma} arise, respectively, from the number of dimensions of MFCC/CQT (both 20) and Chroma (12).

Regarding i-vectors, the UBM is comprised of the whole dataset (123 songs), assuming it is enough to describe the audio features space. We used MFCC and CQT features (with the same parameters used for the baseline) as two different representations of the audio features. Regarding the selection of the number of gaussian components of the UBM, we ranged its values from 20 to 80. We also varied the size of the segment containing the ”bag of statistics” to be extracted into an i-vector. This size can be measured as the number of audio frames and its value ranges from 1 to 10 frames. Finally, we varied the number of dimensions of the i-vectors from 2 to 19.

### D. Evaluation

We decided to employ the evaluation of MIREX (detailed in section III-A) seeing that it is applied on most of the systems performing our task. Regarding Boundary Retrieval,

| | $\delta$ | MTTG(s) | MGTT(s) | Precision | Recall | F-measure |
|---|---|---|---|---|---|---|
| CQT | **5** | 1.965 | 3.951 | 0.498 | 0.705 | **0.571** |
| | 7 | 1.720 | 4.060 | 0.485 | 0.720 | 0.568 |
| | 10 | 1.741 | 4.129 | 0.481 | 0.734 | 0.569 |
| Chroma | **5** | 2.234 | 4.186 | 0.450 | 0.640 | **0.519** |
| | 7 | 2.004 | 4.359 | 0.436 | 0.668 | 0.519 |
| | 10 | 1.934 | 4.473 | 0.426 | 0.686 | 0.517 |
| MFCC | **5** | 1.955 | 3.677 | 0.509 | 0.694 | **0.579** |
| | 7 | 1.881 | 3.887 | 0.499 | 0.721 | 0.577 |
| | 10 | 1.721 | 3.983 | 0.485 | 0.732 | 0.573 |

TABLE IV: Baseline: Boundary Retrieval

| | $\delta$ | Precision | Recall | F-measure |
|---|---|---|---|---|
| CQT | **5** | 0.611 | 0.482 | **0.526** |
| | 7 | 0.663 | 0.446 | 0.521 |
| | 10 | 0.706 | 0.421 | 0.515 |
| Chroma | **5** | 0.536 | 0.466 | **0.484** |
| | 7 | 0.572 | 0.408 | 0.463 |
| | 10 | 0.613 | 0.374 | 0.451 |
| MFCC | **5** | 0.608 | 0.459 | **0.511** |
| | 7 | 0.665 | 0.413 | 0.499 |
| | 10 | 0.709 | 0.374 | 0.478 |

TABLE V: Baseline: Frame Clustering

we consider a boundary to be correctly identified if it is within 3 seconds of a ground truth boundary. All results in section V will be provided using the metrics aforementioned, explicitly divided into two evaluation sets:

- Boundary Retrieval: Precision, Recall, F-value, MGTT, MTTG
- Frame Clustering: Precision, Recall, F-value

All results shown are relative to the whole dataset, resulting from the mean values of the corresponding metrics for all the songs in the dataset.

## V. Results

### A. Baseline

The QM Segmenter software allows us to set some parameters such as the maximum number of different segment labels ($\delta$) for the segmentation. Since it proved to have a significant impact on performance, tables IV and V compare the results obtained by varying $\delta$. For simplicity reasons, even though the f-measure metric does not entirely determine which instance is the best, we selected it as the tiebraker to pick the "best" $\delta$ value. For all features, the best f-measure score was obtained with $\delta = 5$. Hence, all experiments done throughout the development of this document were made with $\delta = 5$.

### B. GCCA

GCCA was performed over 2 pairs of features, namely (MFCC, CQT) and (MFCC, Chroma). Sections V-B1 and V-B2 compare the performance impact by varying the number of dimensions of the final space for both pairs. Overall, the best performance was achieved with 20 dimensions (better scores in all metrics except MTTG(Boundary Retrieval)).

*1) GCCA using MFCC and CQT:* From tables VI and VII, we can see a comparison between the performance of GCCA(MFCC, CQT) with 5, 10, 15 and 20 dimensions.

| #dims | MTTG(s) | MGTT(s) | Precision | Recall | F-Measure |
|---|---|---|---|---|---|
| 5 | 2.236 | 3.983 | 0.494 | 0.675 | 0.558 |
| 10 | **2.012** | 3.870 | 0.508 | 0.686 | 0.572 |
| 15 | 2.402 | 3.870 | 0.505 | 0.687 | 0.571 |
| 20 | 2.374 | **3.780** | **0.513** | **0.702** | **0.580** |

TABLE VI: Results for Boundary Retrieval metrics, of GCCA using MFCC and CQT, by varying the number of dimensions of the final space between 5 and 20.

| #dims | Precision | Recall | F-Measure |
|---|---|---|---|
| 5 | 0.579 | 0.463 | 0.500 |
| 10 | 0.601 | 0.471 | 0.515 |
| 15 | 0.615 | 0.474 | 0.522 |
| 20 | **0.617** | **0.478** | **0.525** |

TABLE VII: Results for Frame Clustering metrics, of GCCA using MFCC and CQT, by varying the number of dimensions of the final space between 5 and 20.

Overall, the best performance was achieved with 20 dimensions (better scores in all metrics except MTTG (Boundary Retrieval)).

*2) GCCA using MFCC and Chroma:* From tables VIII and IX, we can see a comparison between the performance of GCCA(MFCC, Chroma) with 3, 6, 9 and 12 dimensions. Overall, the best performance was achieved with 12 dimensions (better scores in all metrics except MGTT (Boundary Retrieval) and Recall (Frame Clustering)).

### C. i-vectors

Beforehand, we performed several experiments in order to analyze the performance impacts by varying all the three parameters separately, namely: gaussian components, size of the segments and the number of dimensions of the i-vector, respectively. Combining all possible values for the three parameters, a total of 210 instances were ran. From tables X and XI we can see which is the best instance for each one of the 8 evaluation metrics, regarding MFCC and CQT features, respectively.

| #dims | MTTG | MGTT | Precision | Recall | F-Measure |
|---|---|---|---|---|---|
| 3 | 2.451 | 4.092 | 0.467 | 0.637 | 0.529 |
| 6 | **1.995** | 3.904 | 0.482 | 0.677 | 0.554 |
| 9 | 2.020 | **3.830** | 0.492 | 0.684 | 0.561 |
| 12 | 2.005 | 3.862 | **0.494** | **0.687** | **0.565** |

TABLE VIII: Results for Boundary Retrieval metrics, of GCCA using MFCC and Chroma, by varying the number of dimensions of the final space between 3 and 12.

| #dims | Precision | Recall | F-Measure |
|---|---|---|---|
| 3 | 0.534 | 0.464 | 0.483 |
| 6 | 0.572 | 0.463 | 0.499 |
| 9 | 0.582 | **0.474** | 0.508 |
| 12 | **0.591** | 0.470 | **0.512** |

TABLE IX: Results for Frame Clustering metrics, of GCCA using MFCC and Chroma, by varying the number of dimensions of the final space between 3 and 12.

| i-vectors (MFCC) | Metric | Components | Seg.size | Dimensions | Value |
|---|---|---|---|---|---|
| Boundary Retrieval | MTTG | 20 | 1 | 11 | 2.478(s) |
| | MGTT | 20 | 5 | 11 | 3.944(s) |
| | Precision | 50 | 3 | 19 | 0.486 |
| | Recall | 60 | 2 | 15 | 0.656 |
| | F-measure | 20 | 5 | 15 | 0.544 |
| Frame Clustering | Precision | 20 | 1 | 11 | 0.576 |
| | Recall | 50 | 7 | 3 | 0.467 |
| | F-measure | 30 | 3 | 11 | 0.485 |

TABLE X: Results for both Boundary Retrieval and Frame Clustering metrics, of the instance that achieved the best score regarding each metric, separately.

| i-vectors (CQT) | Metric | Components | Seg.size | Dimensions | Value |
|---|---|---|---|---|---|
| Boundary Retrieval | MTTG | 30 | 10 | 19 | 1.755(s) |
| | MGTT | 30 | 10 | 11 | 3.740(s) |
| | Precision | 20 | 1 | 19 | 0.518 |
| | Recall | 80 | 3 | 15 | 0.706 |
| | F-measure | 80 | 3 | 15 | 0.572 |
| Frame Clustering | Precision | 30 | 5 | 15 | 0.606 |
| | Recall | 20 | 1 | 19 | 0.512 |
| | F-measure | 30 | 5 | 15 | 0.527 |

TABLE XI: Results for both Boundary Retrieval and Frame Clustering metrics, of the instance that achieved the best score regarding each metric, separately.

| | MTTG | MGTT | Precision | Recall | F-Measure |
|---|---|---|---|---|---|
| MFCC | **1.955** | **3.677** | 0.509 | 0.694 | 0.579 |
| CQT | 1.965 | 3.951 | 0.498 | **0.705** | 0.571 |
| GCCA | 2.374 | 3.780 | **0.513** | 0.702 | **0.580** |

TABLE XII: GCCA vs baseline: Boundary Retrieval

| | Precision | Recall | F-Measure |
|---|---|---|---|
| MFCC | 0.608 | 0.459 | 0.511 |
| CQT | 0.611 | **0.482** | **0.526** |
| GCCA | **0.617** | 0.478 | 0.525 |

TABLE XIII: GCCA vs baseline: Frame Clustering

| | MTTG | MGTT | Precision | Recall | F-Measure |
|---|---|---|---|---|---|
| MFCC | **1.955** | **3.677** | **0.509** | **0.694** | **0.579** |
| Chroma | 2.234 | 4.186 | 0.450 | 0.640 | 0.519 |
| GCCA | 2.005 | 3.862 | 0.494 | 0.687 | 0.565 |

TABLE XIV: GCCA vs baseline: Boundary Retrieval

| | Precision | Recall | F-Measure |
|---|---|---|---|
| MFCC | **0.608** | 0.459 | 0.511 |
| Chroma | 0.536 | 0.466 | 0.484 |
| GCCA | 0.591 | **0.470** | **0.512** |

TABLE XV: GCCA vs baseline: Frame Clustering

### *D. GCCA vs Baseline*

In this section, we are going to compare the results obtained with GCCA (using the pairs of features (MFCC, CQT) and (MFCC, Chroma)) with the results obtained using only the baseline features that make up the pairs. Experiments were done beforehand, to determine the best value for the number of dimensions of the final space, for each pair of features.

*1) (MFCC, CQT):* Tables XII and XIII compare the best instances from the baseline with MFCC and CQT features and the best instance from GCCA with the pair (MFCC, CQT). Results of MFCC and CQT are quite similar. Additionally, GCCA results are also very similar, providing no significant performance improvement.

Tables XII and XIII compare the best instances from the baseline with MFCC and CQT features and the best instance from GCCA with the pair (MFCC, CQT). Results of MFCC and CQT are quite similar. Additionally, GCCA results are also very similar, providing no significant performance improvement.

*2) (MFCC, Chroma):* From tables XIV and XV, we can see that Chroma presented the worst results for all metrics. Regarding Boundary Retrieval's MTTG and MGTT, MFCC performed slightly better than GCCA, whereas performances regarding Precision, Recall and F-value were led by GCCA. Regarding Frame Clustering, GCCA provided a slight increase in Precision, a slight decrease in Recall and the same F-value when compared to MFCC.

### *E. i-vectors vs Baseline*

As stated in sections IV-A and V-C, we varied the number of gaussian components of the UBM, the size of the segment to be extracted as an i-vector and the number of dimensions of the i-vectors. By analyzing the corresponding results, we were able to pick the best (overall) instances, for both MFCC and CQT features. The following two subsections compare these instances with the best instance from the baseline.

*1) MFCC:* The i-vectors instance reported in tables XVI and XVII was ran with 20 gaussian components, 11 dimensions and with a segment size of 1. From tables XVI and XVII, we can see that results from the baseline are significantly better compared to the i-vectors, with an average percentage difference of 2,5% and 2,7% regarding Precision, Recall and F-measure metrics of Boundary Retrieval and Frame Clustering, respectively.

*2) CQT:* The i-vectors instance reported in tables XVIII and XIX was ran with 30 gaussian components, 15 dimensions

| | MTTG | MGTT | Precision | Recall | F-Measure |
|---|---|---|---|---|---|
| Baseline | **1.955** | **3.677** | **0.509** | **0.694** | **0.579** |
| i-vectors | 2.478 | 3.970 | 0.479 | 0.652 | 0.542 |

TABLE XVI: i-vectors vs baseline(MFCC): Boundary Retrieval

| | Precision | Recall | F-Measure |
|---|---|---|---|
| Baseline | **0.608** | **0.459** | **0.511** |
| i-vectors | 0.576 | 0.444 | 0.483 |

TABLE XVII: i-vectors vs baseline(MFCC): Frame Clustering

and with a segment size of 5. Contrasting with the performance of i-vectors using MFCC features (reported in section V-E1), results from i-vectors when using CQT features have proven to be at least as good as the baseline (slight decreases of performance on some of the evaluation metrics and slight increases of performance on others).

### *F. Discussion*

We experimented GCCA in an attempt to create a denser audio space with multiple features, and i-vectors to enhance the discriminative power of the potential boundaries between different segments. Results from GCCA led to slight increases in performance regarding some evaluation metrics, but overall it was as good as the baseline. Results from i-vectors using CQT features also achieved slight improvements regarding some evaluation metrics, while results regarding the use of MFCC features led to significant decreases in performance. One possible cause for the poorer performance of i-vectors using MFCC versus i-vectors using CQT, is the low audio quality of our dataset together with the tendency of MFCC lacking robustness in the presence of additive noise [22]. The lack of significant improvements when using GCCA reveals that the covariance between the different views was not so well captured. Possible reasons for this issue are: linear projections are not able to handle the input features, song-specific models do not contain enough data, frame/hop sizes of the input features are not adequate and/or the projection of the final embeddings causes a loss of meaningful information from the input features. Tables XX and XXI summarize the best results for each one of the eight different input feature vectors.

## VI. Conclusions and Future Work

This document reviewed several feature extraction models and structural segmentation algorithms. Our experiments were focused on applying different feature models using the same structural segmentation algorithm and observe the corresponding performance impacts. We first ran a baseline with three well-known acoustic features (MFCC, CQT and Chroma) and then we applied two different techniques acting as feature models (which, to our knowledge, have never been used for our task), namely GCCA and i-vectors. Afterwards, we compared the baseline with the novel approaches and observed slight performance improvements regarding some evaluation metrics.

Future work might include the use of different framing parameters for the input features, different feature models, larger datasets and non-linear CCA variants, such as Deep Canonical Correlation Analysis [23].

| | MTTG | MGTT | Precision | Recall | F-Measure |
|---|---|---|---|---|---|
| Baseline | 1.965 | 3.951 | **0.498** | **0.705** | **0.571** |
| i-vectors | **1.871** | **3.908** | 0.493 | 0.686 | 0.563 |

TABLE XVIII: i-vectors vs baseline(CQT): Boundary Retrieval

| | Precision | Recall | F-Measure |
|---|---|---|---|
| Baseline | **0.611** | 0.482 | 0.526 |
| i-vectors | 0.606 | **0.484** | **0.527** |

TABLE XIX: i-vectors vs baseline(CQT): Frame Clustering

## References


[1] M. Mauch, K. C. Noland, and S. Dixon, "Using musical structure to enhance automatic chord transcription," in *Proceedings of the 10th International Conference on Music Information Retrieval, Kobe, Japan*, 2009, pp. 231–236.

[2] C. Cannam, E. Benetos, M. Mauch, M. E. Davies, S. Dixon, C. Landone, K. Noland, and D. Stowell, *Mirex 2015: Vamp plugins from the centre for digital music*, 2015.

[3] 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.

[4] A. Klapuri, J. Paulus, and M. Müller, "Audio-based music structure analysis," in *ISMIR, in Proc. of the Int. Society for Music Information Retrieval Conference*, 2010.

[5] F. Kaiser and T. Sikora, "Music structure discovery in popular music using non-negative matrix factorization.," in *ISMIR*, 2010, pp. 429–434.

[6] R. Chen and M. Li, "Music structural segmentation by combining harmonic and timbral information.," in *ISMIR*, 2011, pp. 477–482.

[7] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[8] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1435–1447, 2007.

[9] P. Kenny, "Joint factor analysis of speaker and session variability: Theory and algorithms," *CRIM, Montreal,(Report) CRIM-06/08-13*, vol. 14, pp. 28–29, 2005.

[10] S. Abdallah, K. Noland, M. Sandler, M. A. Casey, and C. Rhodes, "Theory and evaluation of a bayesian music structure extractor," in *International Conference on Music Information Retrieval*, J. Reiss and G. Wiggins, Eds., London, 2005, pp. 420–425. [Online]. Available: http://research.gold.ac.uk/2349/.

| | MTGG | MGTT | Precision | Recall | F-Measure |
|---|---|---|---|---|---|
| MFCC | 1.955 | **3.677** | 0.509 | 0.694 | 0.579 |
| Chroma | 2.234 | 4.186 | 0.450 | 0.640 | 0.519 |
| CQT | 1.965 | 3.951 | 0.498 | **0.705** | 0.571 |
| GCCA(Chroma, MFCC) | 2.005 | 3.862 | 0.494 | 0.687 | 0.565 |
| GCCA(CQT, MFCC) | 2.374 | 3.780 | **0.513** | 0.702 | **0.580** |
| i-vectors(MFCC) | 2.478 | 3.970 | 0.479 | 0.652 | 0.542 |
| i-vectors(CQT) | **1.871** | 3.908 | 0.493 | 0.686 | 0.563 |

TABLE XX: Results for Boundary Retrieval metrics, of the best instance of each one of the eight feature models used.

| | Precision | Recall | F-Measure |
|---|---|---|---|
| MFCC | 0.608 | 0.459 | 0.511 |
| Chroma | 0.536 | 0.466 | 0.484 |
| CQT | 0.611 | 0.482 | 0.526 |
| GCCA(Chroma, MFCC) | 0.591 | 0.470 | 0.512 |
| GCCA(CQT, MFCC) | **0.617** | 0.478 | 0.525 |
| i-vectors(MFCC) | 0.576 | 0.444 | 0.483 |
| i-vectors(CQT) | 0.606 | **0.484** | **0.527** |

TABLE XXI: Results for Frame Clustering metrics, of the best instance of each one of the eight feature models used.


[11] W. Chai and B. Vercoe, "Music thumbnailing via structural analysis," in *Proceedings of the eleventh ACM international conference on Multimedia*, ACM, 2003, pp. 223–226.

[12] J. Paulus and A. Klapuri, "Music structure analysis by finding repeated parts," in *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, ACM, 2006, pp. 59–68.

[13] D. Turnbull, G. R. Lanckriet, E. Pampalk, and M. Goto, "A supervised approach for detecting boundaries in music using difference features and boosting.," in *ISMIR*, 2007, pp. 51–54.

[14] M. Levy and M. Sandler, "New methods in structural segmentation of musical audio," in *Signal Processing Conference, 2006 14th European*, IEEE, 2006, pp. 1–5.

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

[16] 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*, ser. MIR '04, New York, NY, USA: ACM, 2004, pp. 275–282, ISBN: 1-58113-940-3. DOI: 10.1145/1026711.1026756. [Online]. Available: http://doi.acm.org/10.1145/1026711.1026756.

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

[18] M. C. McCallum, "Unsupervised learning of deep features for music segmentation," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 346–350.

[19] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[20] Z. Huang, Y.-C. Cheng, K. Li, V. Hautamäki, and C.-H. Lee, "A blind segmentation approach to acoustic event detection based on i-vector.," in *INTERSPEECH*, 2013, pp. 2282–2286.

[21] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[22] X. Zhao and D. Wang, "Analyzing noise robustness of mfcc and gfcc features in speaker identification," in *2013 IEEE international conference on acoustics, speech and signal processing*, IEEE, 2013, pp. 7204–7208.

[23] G. Andrew, R. Arora, J. Bilmes, and K. Livescu, "Deep canonical correlation analysis," in *International conference on machine learning*, 2013, pp. 1247–1255.