REAL-TIME UNSUPERVISED MUSIC STRUCTURAL SEGMENTATION USING DYNAMIC DESCRIPTORS

André Pires
Department of Computer Science
University of São Paulo
andrespires@gmail.com

Marcelo Queiroz
Department of Computer Science
University of São Paulo
mqz@ime.usp.br

ABSTRACT

This paper presents three approaches for music structural segmentation, i.e. intertwined music segmentation and labelling, using real-time techniques based solely on dynamic sound descriptors, without any training data. The first method is based on tracking peaks of a sequence obtained from a weighted off-diagonal section of a dissimilarity matrix, and uses Gaussian models for labelling sections. The second approach is a multi-pass method using Hidden Markov Models (HMM) with Gaussian Mixture Models (GMM) in each state. The third is a novel approach based on an adaptive HMM that dynamically identifies and labels sections, and also sporadically reevaluates the segmentation and labelling, allowing redefinition of past sections based on recent and immediate past information. Finally, a method to evaluate results is presented, that allows penalization both of incorrect section boundaries and of incorrect number of detected segments, if so desired. Computational results are presented and analysed both from quantitative and qualitative points-of-view.

1. INTRODUCTION

Music structural segmentation is a task that underlies many important audio processing applications, including genre classification, audio summarization and music search [1]. Finding the temporal borders and labelling music sections according to common properties provides a means to approach such applications. Besides, real-time music structural segmentation may also be used in musical performances, for instance to control interactive processes responding to timbre variations.

In the literature we may find several methods for music segmentation. Cooper and Foote [2] propose a segmentation method based on finding peaks of a dissimilarity sequence built from a similarity matrix, whose main diagonal is traversed (and dot-multiplied) by a radial smoothing checkerboard kernel; additionally, they cluster similar sections via a statistical method. Aucouturier and Sandler [3] propose a segmentation method based on HMMs, using the sequence of observations as training data. Peeters, la Burthe and Rodet [4] propose a multi-pass approach using an HMM which is initialized after finding the centroids of the sections using k-means. The main difference between the latter and the former method lies in the way the HMM is initialized, where HMM parameters are expected to converge to a local optimum that better represents the real music sections, avoiding over-segmentation due to fine-grained changes in spectral content, aiming for long-term structures [4].

The methods proposed in [2, 3, 4] are not specifically meant to be used in real-time, and they specifically require training data for initializing key structures used in segmentation. As opposed to that, we look specifically toward real-time unsupervised techniques, i.e. methods that operate promptly on an audio stream and do not require any *a priori* information about the stream content. In section 3.2 and 3.3 we present variations of methods found in [2, 4], and in section 3.5 we propose a novel approach based on an adaptive HMM that identifies section boundaries and labels sections in real-time, but also sporadically reevaluates the output of the segmentation, allowing a complete redefinition of past sections using more recent information, possibly information that was not yet available at the time a specific past section was identified. This allows the correction of errors imposed by the real-time output requirement, both in terms of redefining section boundaries as well as relabelling sections according to the most recent statistical models of the identified sections.

According to [9], all segmentation methods described in this paper could be categorized as *homogeneity-based* approaches, as they first depart from a *novelty-based* procedure to finally produce clusters based on the similarity between section models.

The goal of this paper is both to present a novel real-time method for the music structural segmentation problem, and to assess the results of segmentation using real-time unsupervised techniques, both from quantitative and qualitative points-of-view. We also aim to compare segmentation results using instantaneous and dynamic descriptors within such methods. Dynamic descriptors [4] provide an alternative temporal modelling for describing audio frames, combining information obtained from several audio frames, as opposed to instantaneous descriptors, which refer to a single audio frame.

The structure of this paper is as follows; in section 2 we present techniques for defining dynamic descriptors; in section 3, we develop the aforementioned real-time unsupervised techniques for automatic music structural segmentation; and in section 4 we present a method to evaluate the results obtained by these techniques, which is used to compare their performances using instantaneous MFCC descriptors and several dynamic MFCC descriptors.

2. DYNAMIC DESCRIPTORS

Choosing the right sound features can strongly influence the segmentation's success or failure. As pointed by [4], Mel Frequency Cepstral Coefficients (MFCC) can be considered *static* features, as they represent sound restricted by an analysis frame (usually 30 to 100 ms), as opposed to *dynamic* features, that can model the temporal evolution of sound.

One way to model dynamic signal evolution is shown in [4], where temporal evolution is modelled after the spectral shape of the signal energy, divided in Mel sub-bands. Another approach for generating dynamic descriptors is based on the idea of modelling temporal evolution from any descriptors extracted from the audio signal [5], using a smoothing function that condenses several past observations into a single value. This attenuates abrupt changes in the sequence of observations at the frame-rate temporal level, which might easily confuse segmentation techniques, but hopefully it preserves timbre changes at larger-scale temporal levels.

Let $X = \{x_n\} \subset \mathbb{R}^d$, $n = 1, \ldots, N$ be the observation sequence extracted from the audio signal, where n is a frame index. Let L be the number of past observations, we define a dynamic descriptor sequence $Y = \{y_n\} \subset \mathbb{R}^\alpha$, $n = L+1, \ldots, N$ as

$$y_n = D(x_{n-L}, x_{n-L+1}, \ldots, x_n), \qquad (1)$$

where D is a smoothing function that takes into account L observations and α is the new dimension after the transform. This smoothing function D can be defined in several ways; we considered four strategies: statistical moments, Euclidean norm, exponential decay and FFT coefficients.

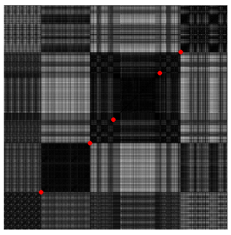

Figure 1. Similarity Matrix generated using raw MFCC descriptors. The red dots are the real transitions between sections, manually defined.

To illustrate these strategies, consider the similarity matrix of Figure 1, generated using MFCC. Figures 2a to 2h display two similarity matrices for each strategy, with dynamic descriptors using temporal memories of 1 and 10 seconds, corresponding in our experiments to $L = 20$ and $L = 200$, respectively. In those images the *red dots* are the real transitions between sections, which were manually

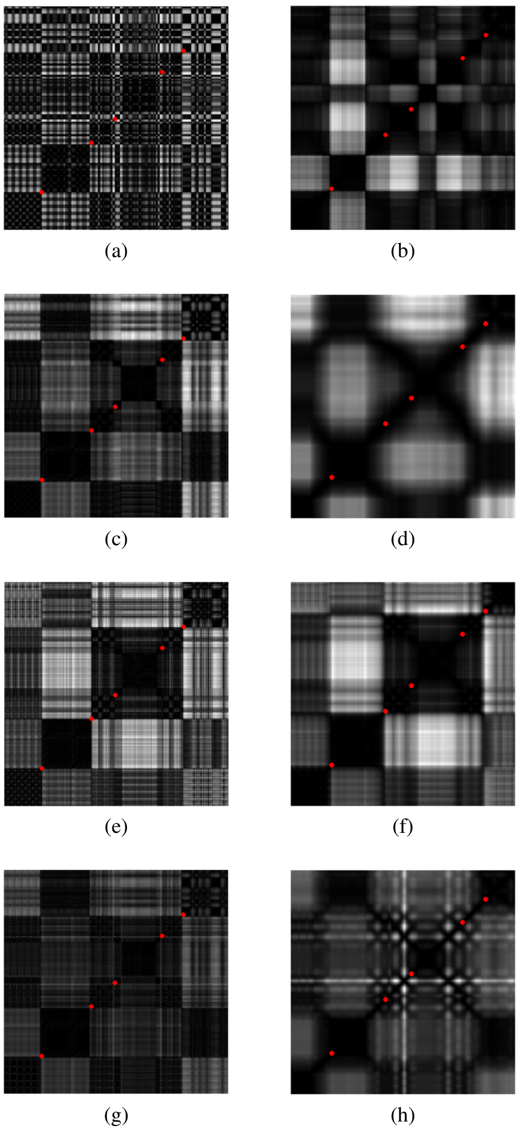

Figure 2. Similarity matrices generated using dynamic descriptors with memory parameters of 1 and 10 seconds, shown in left and right column, respectively. Figures (2a) and (2b) use *Statistical Moments*, figures (2c) and (2d) use *Euclidean Norm*, figures (2e) and (2f) use *Exponential Decay* and figures (2g) and (2h) use *FFT coefficients*. The red dots are the real transitions between sections, which were manually defined.

defined. The similarities were calculated using the cosine similarity measure.

Thus, for each $x_{k,t}$ (i.e. the dimension k of x at instant time t) we can generate the descriptors as follows.

Statistical Moments. In the dynamic descriptors generated with statistical moments, each $x_{k,t}$ provides four new values: mean, variance, skewness and kurtosis.

Euclidean Norm. This dynamic descriptor represents the temporal series corresponding to $x_{k,t}$ as the Euclidean norm of L previous values, similarly to what RMS amplitude does with respect to instantaneous amplitude values.

Exponential Decay. With this dynamic descriptor, temporal memory is represented by an exponential decay function $w(n) = e^{-2\pi n}$, which is used to attenuate past observations.

FFT coefficients. With this method, each dimension k of x_t is segmented in windows of size L, from which Fast Fourier Transform (FFT) coefficients are calculated. Only the first 7 coefficients are selected, corresponding to the slow variation of the spectrum of that descriptor.

3. REAL-TIME STRUCTURAL SEGMENTATION

To define objectively and above all suspicion what a *musical section* is might be a harsh task. Putting aside the music theoretic issues and focusing only on the properties of sound (more specifically the *polyphonic timbre* as defined by [3]), we may define a section as an audio fragment with a contiguously varying global timbre description. Pitch, rhythm and amplitude might give some clues to where section boundaries are, but in this work we aim to capture section changes only by identifying contiguous sections of global timbre stability.

In real-time unsupervised music structural segmentation there is no available training data, and the input data are flowing continuously. One issue we have to deal with is to define the right moment when the identified sections are going to be labelled. As the input data keeps flowing in, labels and section boundaries might also change, and until the music ends it is not possible to define all musical sections and labels. Despite this theoretical impossibility, it might be interesting to consider musical applications of incomplete or provisional segmentation and labelling, for instance, for switching on and off interactive sound-processing units depending on variation of global timbre. Defining an incomplete segmentation and labelling at time t as the result of those processes assuming the music had literally ended at time t is at least theoretically sound.

Algorithm 1 outlines a general solution for real-time structural segmentation. At first, it continuously reads the descriptor vector and accumulates these observations until it reaches a minimum section size allowed. The core of the algorithm is formed by the function that finds section changes (line 5) and the function that labels sections according to timbre proximity (line 17). In the following sections we will discuss the techniques used to solve these two problems. In section 3.1 we will present a first technique to automatically label the sections; alternative techniques will be presented in sections 3.3 and 3.5, which uses Viterbi's algorithm on an HMM to label the sections.

Algorithm 1 Real-time structural segmentation

```
Require: $I$ ; Input data
         $w_{\min}$ ; minimum window size
         $w_{\max}$ ; maximum window size
Ensure:  $S$ ; section change locations
         $L$ ; labelled sections
 1: $p \leftarrow 1, T \leftarrow \{\}, x \leftarrow \mathrm{read}(I)$
 2: while $x \neq$ NULL do
 3:   $T \leftarrow T \cup \{x\}$
 4:   if $|T| + 1 \geq w_{\min}$ then
 5:     $relative \leftarrow \mathrm{LocateChangePoint}(T)$
 6:     if $relative =$ NULL then
 7:       if $|T| + 1 > w_{\max}$ then
 8:         $T \leftarrow \{t_i : t_i \in T,\, i \geq w_{\min}\}$
 9:       end if
10:     else
11:       $absolute \leftarrow (p-1) - |T| + relative$
12:       $T \leftarrow \{t_i : t_i \in T,\, i > relative\}$
13:       $S \leftarrow S \cup absolute$
14:     end if
15:     $p \leftarrow p + 1$
16:   end if
17:   $L \leftarrow \mathrm{Label}(S), x \leftarrow \mathrm{read}(I)$
18: end while
```

3.1 Labelling

The primary task in labelling is to identify which musical sections have the same characteristics, i.e. share a common global timbre so they may receive the same label. For this purpose we need a method to compare all sections found and to calculate some measure of section separability. This measure should determine how similar is a section model ω_i to another section model ω_j, and a threshold on this measure would tell us whether or not to label them equally. For didactic purposes let us consider only two section models: ω_i and ω_j; this labelling method will extend to a set of P sections in a straightforward fashion. This method assumes that section models are normally distributed.

The Bhattacharyya distance was developed to estimate the classifier error, but it is also used as a class separability measure, and provides an upper bound to the classifier error, similarly to the Chernoff Bound [6]. This distance between section models $\omega_i = \mathcal{N}(\mu_i, \Sigma_i)$ and $\omega_j = \mathcal{N}(\mu_j, \Sigma_j)$ is given by

$$d_{ij} = \frac{1}{8}(\mu_i - \mu_j)^T \left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1} (\mu_i - \mu_j) + \frac{1}{2} \ln \frac{\left|\frac{\Sigma_i + \Sigma_j}{2}\right|}{\sqrt{|\Sigma_i||\Sigma_j|}}.$$

Once we have calculated the distance for each pair of section models, those who have very small distances shall be clustered, sharing the same label. In order to cluster similar sections, we have used average linkage clustering [7], and the clusters are given by calculating the inconsistency coefficient ξ_i for each link i, established by the clustering algorithm, and cutting the dendrogram at any point between the two links having the largest inconsistency values [8]. The inconsistency coefficient is a measure of how similar the links below each link are, calculated as $\xi_i = \frac{z_i - \mu_{z_i}}{\sigma_{z_i}}$, where z_i is the i-th link height, μ_{z_i} is the mean of all link's heights at the same depth and level and σ_{z_i} is the standard deviation of all link's heights at the same depth and level. Figure 3a shows a dissimilarity matrix using the Bhattacharyya distance between section models before the clustering algorithm. Light shades indicate small distances between models, meaning that a cluster can possibly be formed. Depending on the threshold we might say that the second and third section models are candidates to form a cluster, as well as the sixth, seventh and eight models. Figure 3b shows the section models after the clustering algorithm. Note that the clustering algorithm may join not only temporally adjacent sections, but also temporally dis-

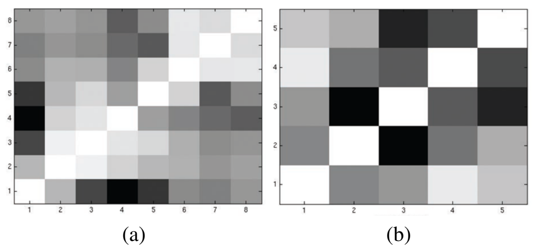
Figure 3. Dissimilarity between section model pairs, calculated before (figure 3a) and after (figure 3b) clustering, using Bhattacharyya distance.

tant sections, according to the similarity of their statistical models.

The above method is akin to the one in [2], but there a Singular Value Decomposition of the similarity matrix is used, and section models are clustered according to the Kullback-Leibler distance.

3.2 Dissimilarity Matrix Peaks

In the Dissimilarity Matrix Peaks (DISSM-PEAKS) method, transitions are detected through the rapid and abrupt changes in the temporal observation sequence. To this end, we calculate a dissimilarity sequence using the information given by the dissimilarity matrix and select the peaks of this sequence, which are the section transitions. The dissimilarity matrix is built using the cosine distance. After we locate the transitions, each section is mapped into a normal distribution and we label them according to subsection 3.1. The details of this method are as follows.

1. Let N be the number of observations. The dissimilarity sequence, defined as $\Delta = \{\delta_i : 1 \leq i \leq N\}$, is calculated by scanning a certain neighborhood of the dissimilarity matrix M from each diagonal point M_{ii}. This neighborhood can be characterized by its size $h \ll N$ and a vector ν_i, $1 \leq i \leq N$ containing the neighboring observations $\nu_i = \{M_{i+k,i-k} : 1 \leq k \leq h,\ 1 \leq i-k,\ i+k \leq N\}$. The dissimilarity is defined as the inner product $\delta_i =< W, \nu_i >$ of the observation vector and a exponential decay vector $W = \{w_j : 1 \leq j \leq h,\ w_j = e^{-2\pi j(h-1)^{-1}}\}$.

2. Detect the peaks of the vector Δ using a noise tolerant peak finder algorithm [1] . Peak magnitudes must be ordered in descending order, in order to select candidate transition points with highest dissimilarity. The final transition points are obtained after the parameter $w_{\min}$ (minimum section size) is used to exclude peaks that lie too close to other candidate transition points with higher dissimilarity. Temporal indices $T = \{t_i : 1 \leq i \leq P\}$ of the P selected peaks are considered transitions points for $P+1$ sections $\{[x_{t_0}, x_{t_1}), [x_{t_1}, x_{t_2}), \ldots, [x_{t_P}, x_{t_{P+1}})\}$, where $t_0 = 1$ and $t_{P+1} = N+1$.

3. Each section $[x_{t_k}, x_{t_{k+1}})$ is mapped to a normal distribution $\mathcal{N}(\mu_k, \Sigma_k)$, where the mean vector μ_k and the covariance matrix Σ_k are estimated from the observation data contained in $[x_{t_k}, x_{t_{k+1}})$.

[1] In our experiments we used a MATLAB® code by Nathanael C. Yoder, peakfinder.m

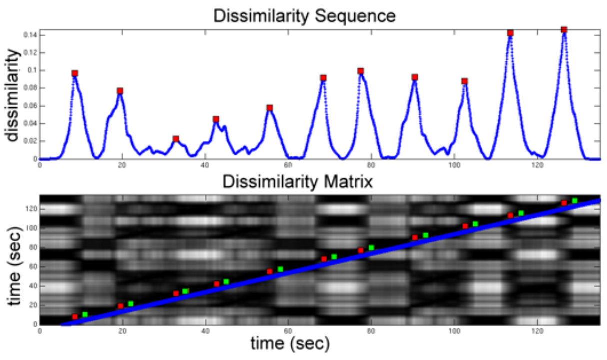

Figure 4. Transitions between sections found with the Dissimilarity Matrix Peaks method. The red dots in dissimilarity matrix are the peaks found before the adjustment, the green dots are the peaks after the adjustment, and the blue strip is the neighborhood used to build the dissimilarity sequence.

4. Label the sections using the method described in subsection 3.1.

When the dissimilarity matrix M is generated using dynamic descriptors, the observation sequence gets smoother as we increase the temporal memory L, and so we need to navigate farther away from the diagonal of the dissimilarity matrix to be able to detect peaks. Thus, it is convenient to change the neighborhood definition so that it starts with an offset from the diagonal, as $\nu_i = \{M_{i+k,i-k} : 1+\rho \leq k \leq h+\rho,\ 1 \leq i-k \leq i+k \leq N\}$, where the parameter $\rho = L/2$ ensures that the method will exclude smoothed observations that are affected by $M_{i,i}$.

Furthermore, also due to temporal smoothing, temporal transitions T found by the method are anticipated of $L/2$ observations, which means that we also need to adjust all points to $\hat{t_i} = t_i + L/2$. In our experiments we have used a 1 second neighborhood, corresponding to $h = 20$, which fits very well for our dataset, but it must be estimated according to a given musical context/genre.

Figure 4 shows the dissimilarity sequence and the transition points found calculated using exponential decay dynamic descriptors, where L corresponds to 5 seconds.

3.3 Multi-pass GMM-HMM

The Multi-Pass GMM-HMM (MPS-GHMM) method is based on the multi-pass algorithm by [4]; the motivation for this variant is modelling the states (potential, initial and final) as mixtures of Gaussian distributions. Using our database set in another running tests, this variation improved the method described in [4], leading to an error of 6.8% against 15.25%. The method can be summarized in the following steps:

Build potential states. First, we identify the boundaries of each section. This is accomplished by the method described in subsection 3.2. To build potential states, we select the observations within each section and estimate the parameters of the Gaussian Mixture Models (GMM) using the Expectation Maximization (EM) algorithm [10]; the potential states are modelled as $s_i(x) =$

$\sum_{m=1}^{M} c_{im} p_{im}(x)$, where i is the i-th state, c_{im} is the m-th mixture coefficient, and the m-th *pdf* is $p_{im}(x) = \frac{1}{2\pi^{d/2}|\Sigma_{im}|^{1/2}} e^{-0.5*(x-\mu_{im})^T \Sigma_{im}^{-1}(x-\mu_{im})}$ with mean vector μ_{im} and covariance matrix Σ_{im}. See Figure 5a.

Reduce potential states. The number of potential states is reduced in a similar way to what is done in [4], i.e. potential states having similarity $> .99$ are grouped together. In [4], the potential states are represented by the mean observation vector within each section, and the cosine distance is used to measure similarity. In our case, the potential states are GMMs having m mean vectors and m covariance matrices, so we used the Bhattacharyya distance to measure similarity and summing up the distances for each Gaussian of the mixture. *Initial states* are generated after grouping potential states; initial states are also modelled as GMMs, whose parameters are estimated using the EM algorithm and the observations of each potential state. In other words, if there is a group $\{s_a, s_b\}$, then GMM parameters of each initial state are estimated through the observations $X_a \subset X$ and $X_b \subset X$ restricted to each potential state section. At the end of this step, K initial states are built. See Figure 5b.

Introduce time constraints. The timing constraints are applied similarly as in [4], except that we can use a mixture of normal distributions instead of a simple normal distribution. The K initial states are used to initialize a continuous ergodic HMM [10]. The parameters of the HMM are reestimated using the Baulm-Welch algorithm with the training data set X, and so we obtain the *final states*. The final music labels are obtained using Viterbi's algorithm, given the HMM and the descriptor vectors X. See Figure 5c.

3.4 Bayesian Information Criterion

The Bayesian Information Criterion (BIC) is a very well-known hypothesis test method in statistics, but to our knowledge it has not yet been used specifically for Music Information Retrieval. Nonetheless, there are works [11, 12] using BIC to detect acoustic changes in an audio signal, e.g. to detect changes from one speaker to another.

BIC is based on the log likelihood ratio between two Gaussian models, and in our case is used to compare two competing hypotheses: either an audio fragment has (at least) one possible splitting point defining (at least) two sections, or else the whole fragment belongs to a single section. Thus, let $X_{i,j} = x_i, \ldots, x_j$ be the observation sequence. We'd like to test the hypothesis of $X_{i,j}$ being adequately modelled by a single normal distribution $\mathcal{N}(\mu, \Sigma)$,

$$H_0 : x_i, \ldots, x_j \sim \mathcal{N}(\mu, \Sigma) \tag{2}$$

versus the hypothesis of $X_{i,j}$ being split in two parts adequately modelled by two different normal distributions

$$\begin{aligned} H_1 : \quad & x_i, \ldots, x_k \sim \mathcal{N}(\mu_1, \Sigma_1) \\ & x_{k+1}, \ldots, x_j \sim \mathcal{N}(\mu_2, \Sigma_2). \end{aligned} \tag{3}$$

Parameters for each normal distribution are computed from the corresponding observation sequence, and depend on the splitting point k. The log-likelihood ratio is given by $R(k) = N \log|\Sigma| - N_1 \log|\Sigma_1| - N_2 \log|\Sigma_2|$ where $N = |X_{i,j}|$, $N_1 = |X_{i,k}|$ and $N_2 = |X_{k+1,j}|$, i.e. the number of observations within each set. The BIC values for each index k is

$$BIC(k) = R(k) - \lambda\Pi \tag{4}$$

which is a function of the maximum likelihood ratio and a penalty factor $\Pi = \frac{1}{2}(d + \frac{1}{2}d(d+1)) \log N$, depending on the number of random variables d in the observation vector X, weighted by λ (in our experiments we have used $\lambda = 1$). The hypothesis H_1 with two different Gaussian models offers a better model (compared to H_0) whenever the result of this equation is positive. Thus the transition point that provides the best splitting point (corresponding to the maximum likelihood ratio) is given by

$$k = \underset{k}{\operatorname{argmax}}\{BIC(k) \mid BIC(k) > 0\}. \tag{5}$$

In real-time segmentation with BIC, after locating the section boundaries, constrained to w_{min} (see algorithm 1) in order to prevent over-segmentation, the labelling is done using the method described in subsection 3.1.

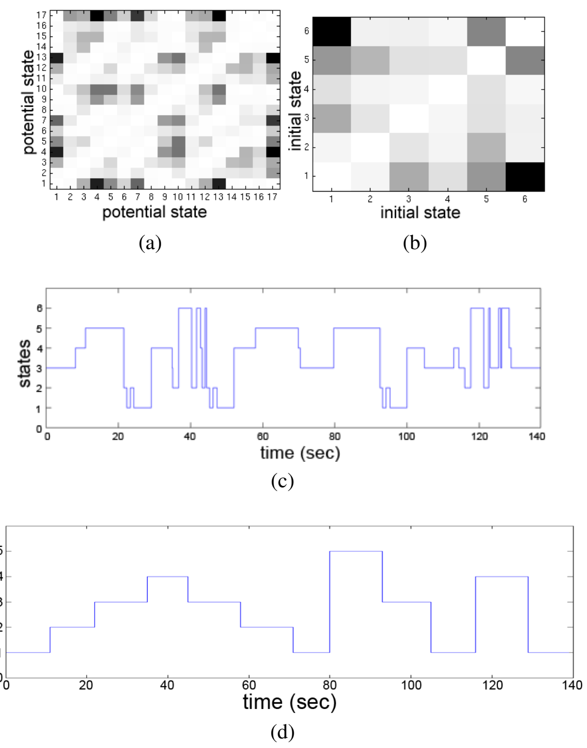

Figure 5. Potential states found using the dissimilarity sequence method – Figure 5a. Initial states after grouping – Figure 5b. Final states and labelling after Viterbi's algorithm – Figure 5c. Reference Segmentation, for comparison – Figure 5d.

3.5 Adaptive HMM

The Adaptive HMM (AHMM) is a real-time method which aims at labelling sections as each transition point between sections is identified, and not afterwards, as in the BIC method. In other words, labelling is done simultaneously with real-time segmentation. The algorithm works in two stages: a cache stage and a reevaluate stage.

3.5.1 *Cache Stage*

Each transition point t_k found provides a new section $T = [x_{t_{k-1}}, x_{t_k})$, and triggers the labelling routine. Let $B = \{b_j\}$ be the set of existing section labels of the real-time routine. The model of the new section is compared to each model corresponding to $b_j \in B$, and the model having higher similarity (mean log-likelihood) is updated with the observations of the new section T, provided that this similarity is higher than a minimum threshold α; if this is not the case, a new label is added to B, with a model built from T. Note that each state is modelled as a mixture of Gaussian distributions

$$b_j(x) = \sum_{m=1}^{M} c_{jm} p_{jm}(x), \tag{6}$$

where M is the number of Gaussians per state and $p_{jm}(x)$ is the Gaussian density function (see Section 3.3).

The details of the first stage of this method are as follows:

1. Initialize HMM $\lambda(A, B, \pi)$ with 0 states, where $A = \{a_{ij}\}$ is the state transition probability matrix and $\pi = \{\pi_j\}$ is the initial state probability distribution.

2. Find a transition point t_k defining a section $T = [t_{k-1}, t_k)$, as described in algorithm 1 (section 3). For instance, using the BIC method 3.4.

3. Find a label b_j with model closest to T:

$$j = \underset{i}{\operatorname{argmax}} \left\{ \theta_i = \frac{1}{|T|} \sum_{t_k \in T} \log b_i(t_k) \right\} \tag{7}$$

   If $\theta_j \leq \alpha$ then a new section model b_{S+1} is added to B, with parameters estimated by the EM algorithm.

   If $\theta_j > \alpha$ then the observations in T are added to the model of b_j, whose parameters are re-estimated using the EM algorithm.

4. Update the state transition matrix A, reinforcing the transition between the previously detected state model and b_j (or b_{S+1}, if $\theta_j \leq \alpha$). Return to step 2.

Instead of simply applying the label with higher similarity to the observations of each section found, observations are labelled according to λ, using Viterbi's algorithm.

3.5.2 *Reevaluate Stage*

As time passes, the HMM model represented by λ accumulates errors due to the requirement of real-time operation, in the sense that models that were defined and made sense for segmentation at time t are carried on and keep influencing labelling at times $t + 1$, $t + 2$ and so on. This shows the need for a model refinement stage, where up-to-date knowledge about the input is used to reevaluate all previously defined sections and labels. The triggering of this refinement routine can be done by several different means. For instance, refinement may be triggered by user intervention, by a threshold on the number of HMM states allowed, or it might be automatically triggered every K iterations for a pre-defined K. The refinement stage of this method is described as follows.

1. Cluster similar section models using hierarchical cluster and Bhattacharyya distance as similarity measure (section 3.1).

2. Update the state transition matrix A by summing up the grouped state transitions.

3. Reestimate the HMM parameters $\hat{\lambda}(\hat{A}, \hat{B}, \pi)$, using the Baulm-Welch algorithm. The corresponding state sequence for the observations X is obtained by Viterbi's algorithm.

Figure 6a shows all states (labels) found in a musical piece just before the refinement stage. Figure 6b shows the same piece after the refinement stage, where we can see a reduction from 26 to 8 states.

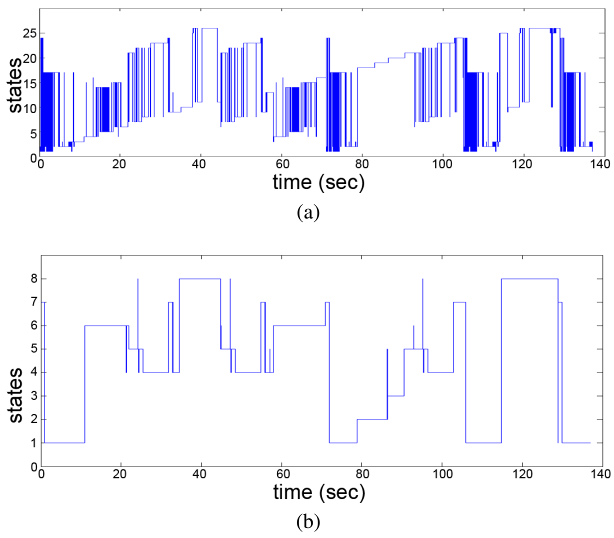

Figure 6. State sequence before (6a) and after (6b) the clustering step in Adaptive HMM. Refer to Figure 5d for reference segmentation.

This method is sensitive to the α parameter, which must be estimated. In our experiments we under-estimated its value (for the record, $\alpha = -0.016138$), so the algorithm tended to produce a larger number of states, which would hopefully be fixed by the second stage. An alternative is to let the user decide which level of similarity is suitable for a given musical context.

4. RESULTS

Having defined the segmentation and labelling techniques, we would like to compare the probability error of applying these methods to a set of musical pieces. We built a musical database containing 41 pieces of different genres with up to 7 different musical parts each, manually segmented and labelled. For each piece, we extracted MFCC descriptors and four groups of dynamic descriptors as in section 2: *moments*, *Euclidean*, *exponential decay* and *FFT coefficients*. Each group was computed using four different values for temporal memory: 0.5, 1, 5 and 10 seconds, totalling 697 executions for each method described in the previously section.

To evaluate segmentation and labelling performances, a measure of probability error is defined in subsection 4.1, which corresponds to a lower bound on the number of temporal positions wrongly labelled. Results are collected in groups of dynamic descriptors, from which we compare the performance of different values of the temporal memory parameter.

4.1 Evaluation

The segmentation and labelling evaluation depends on two factors: the labelling error probability (i.e. a measure of whether the labels in the test sequence are consistent with the labels in the reference sequence), and the segmentation error probability (i.e. a measure of the temporal errors in detected section boundaries). There are many metrics for evaluating the segmentation results, some of which do not consider the temporal order of observations, but the overall frames [9]. We propose here a measure that takes as input the temporal sequence of manually-assigned (correct) labels for each analysis frame, and the corresponding sequence produced by the algorithm. Thus, temporal discrepancies in the localization of boundaries will be translated into the (relative) number of analysis frames wrongly labelled.

Let N be the number of observations in X, p the number of labels manually defined in the reference sequence $f(t)$, such that $f(t) : \mathbb{N} \rightarrow \{1, \ldots, p\}$, and let q be the number of labels found in the test sequence $g(t)$, such that $g(t) : \mathbb{N} \rightarrow \{1, \ldots, q\}$. We wish to find a relabelling function $m : \{1, \ldots, q\} \rightarrow \{1, \ldots, p\}$, corresponding to a translation of the algorithmically produced labels to the manually defined labels, in such a way that the relabelled sequence $h(t) = m(g(t))$ is closest to the reference sequence in the sense of minimizing the error ratio

$$\epsilon(m) = \frac{1}{N} \sum_{t=1}^{N} \delta(f(t), h(t)), \qquad (8)$$

where δ is the Kronecker Delta. This is necessary due to the fact that the labelling method does not necessarily provide the same symbols adopted by the reference sequence.

Although this relabelling function can be constrained in different ways according to different interpretations, in this work we will adopt a measure that focuses on the temporal accuracy of section boundaries, and only implicitly penalizes differences in the number of reference labels p and produced labels q. Thus we will consider as candidate relabelling functions all p^q sequences of size q generated by the elements $\{1, \ldots, p\}$. Instead of a brute force search, we used an association matrix $s_{f(t),g(t)}$, from which the optimal labelling function according to $\epsilon(m)$ is given by the Hungarian algorithm [13].

One interesting aspect of this measure is the fact that it does not over-penalize a segmentation strategy that would subdivide reference sections because of minor timbre variations, but it does penalize incorrectly detected section boundaries. Theoretically, this measure would not penalize a degenerate labelling that associated a different label to every frame, but no honest structural segmentation method would produce such a degenerate solution. An alternative set of relabelling functions that do not suffer from this theoretical problem is the set of injections from $\{1, \ldots, q\}$ to $\{1, \ldots, p\}$, where the second set is enlarged with artificial reference labels whenever $q > p$. This alternative would severely penalize over-segmentations but would be proportionately less stringent on sloppy section boundary detection.

4.2 Comparative Results

To evaluate the impact of the temporal memory parameter on the segmentation and labelling methods, we grouped the results by dynamic descriptor type. Besides executing all methods described previously, we added the K-Nearest Neighbors (K-NN) method for the sake of comparison. As a well-known supervised classification technique, meaning it has access to training data prior to classification, K-NN is expected to surpass every unsupervised competitor, but the obtained error values may be used as a baseline. For the sake of simplicity, we have used a 1-NN configuration, and a 10-fold cross-validation for evaluation.

In our experiments we used 13 MFCC extracted from 100 ms analysis windows with a feature extraction rate of 20 Hz. Comparing the results within each dynamic descriptor type (Figure 7) we see that the segmentation errors generally increase with temporal memory, except for the MPS-GHMM technique, where we can clearly see that smaller values of temporal memory (or no memory, i.e raw MFCC) also produce large errors.

Table 1 displays the best error results for each technique separately, i.e. which dynamic descriptor (if any) worked best for each technique. We may see from this table that only the *Exponential Decay* dynamic descriptors were able to improve error bounds with respect to raw MFCC, whereas *FFT coefficients*, *Euclidean* and *moments* dynamic descriptors followed similar patterns, but with slightly worse error values.

	Error	**Descriptor**
BIC	14.81%	MFCC
AHMM	20.11%	MFCC
MPS-GHMM	5.47%	MFCC Exp. Decay, 1 sec
DISSM-PEAKS	15.41%	MFCC Exp. Decay, 0.5 sec
K-NN	2.57%	MFCC Exp. Decay, 1 sec

Table 1. Minimum error rate for each technique and descriptors that provided best results.

5. CONCLUSIONS

In this paper, we have presented three unsupervised methods for real-time music structural segmentation, including a novel one, and we compared them to a fourth method [11, 12]. We also presented a method to evaluate competing hypotheses for segmenting the same piece, aiming primarily at capturing temporal location errors of section boundaries. We have discussed the generation of dynamic descriptors based on the temporal modelling of MFCC.

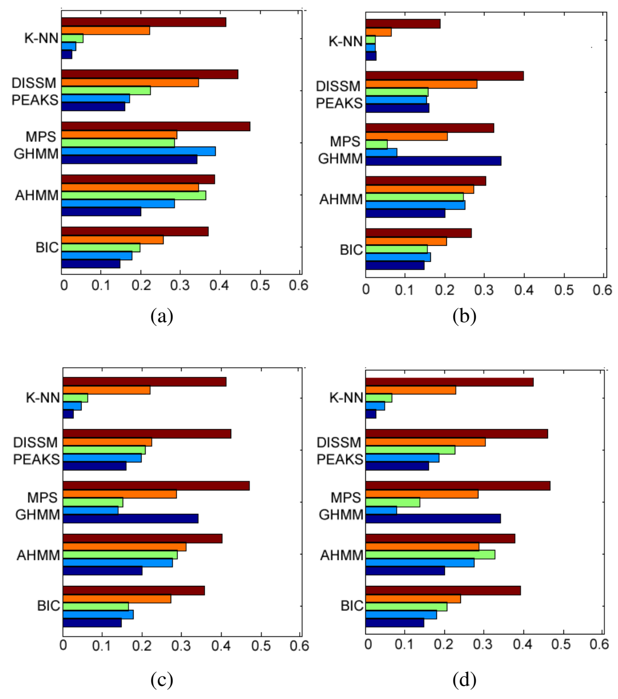

Figure 7. Error rate for each of the technique using *Moments* (Figure 7a), *Exponential Decay* (Figure 7b), *Euclidean* (Figure 7c) and *FFT coefficients* (Figure 7d) dynamic descriptors. Bar colors refer to temporal memory values for the dynamic descriptors: dark blue = raw MFCC values, light blue = 0.5 sec, green = 1 sec, orange = 5 sec and brown = 10 sec.

A number of experiments were performed to compare different methods using dynamic descriptors with different temporal memory values. The primary results are promising, showing that it is possible to perform such tasks in real-time, without any training data. The results show that temporal modelling of static music descriptors can lead to improved results, and that the best dynamic descriptors are those which do remember past information but do not overemphasize it, as is in the case of raw MFCC and Exponential Decay temporal memory.

Further work will consider using other evaluation techniques and a standard musical database, as described in [9].

Acknowledgments

We have used GMM algorithms from Netlab [2], HMM algorithms from PMTK3 [3], and the K-NN algorithm from Weka [14].

This work has been supported by CAPES, CNPq and FAPESP (grant 2008/08632-8). We would also like to thank Nina Hirata, Roberto Hirata, Airlane Alencar and Miguel Ramirez for valuable discussions, and the Brazilian company E-BIZ Solution.

[2] http://www1.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/. March 20th, 2011

[3] http://code.google.com/p/pmtk3. March 20th, 2011

6. REFERENCES

[1] R. Dannenberg and M. Goto, "Music structure analysis from acoustic signals," *Handbook of Signal Processing in Acoustics*, pp. 305–331, 2009.

[2] M. Cooper and J. Foote, "Summarizing popular music via structural similarity analysis," in *Applications of Signal Processing to Audio and Acoustics, IEEE Workshop on.*, 2003, pp. 127–130.

[3] J. Aucouturier and M. Sandler, "Segmentation of musical signals using hidden Markov models," *110th Convention of the Audio Engineering Society*, 2001.

[4] G. Peeters, A. La Burthe, and X. Rodet, "Toward automatic music audio summary generation from signal analysis," in *Proc. International Conference on Music Information Retrieval*, 2002, pp. 94–100.

[5] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the CUIDADO project," *Project Report*, 2004.

[6] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Fourth Edition.* Academic Press, 2008.

[7] F. Murtagh, "A survey of recent advances in hierarchical clustering algorithms," *The Computer Journal*, vol. 26, no. 4, p. 354, 1983.

[8] T. Korenius, J. Laurikkala, M. Juhola, and K. Jarvelin, "Hierarchical clustering of a finnish newspaper article collection with graded relevance assessments," *Information Retrieval*, vol. 9, no. 1, pp. 33–53, 2006.

[9] J. Paulus, M. Müller, and A. Klapuri, "Audio-based music structure analysis," in *Proc. 11th International Conference on Music Information Retrieval*, 2010.

[10] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[11] S. Chen and P. Gopalakrishnan, "Speaker, environment and channel change detection and clustering via the Bayesian Information Criterion," *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, 1998.

[12] M. Omar, U. Chaudhari, and G. Ramaswamy, "Blind change detection for audio segmentation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, Philadelphia, USA*, 2005.

[13] H. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

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