

# Semantic Segmentation and Summarization of Music

Wei Chai

MIT Media Laboratory

chaiwei@media.mit.edu

## ABSTRACT

Automatic segmentation and summarization of music is a key issue in music browsing, searching and recommendation. This article presents methods for segmenting music based on its tonality and recurrent structure, and summarizing music based on its structure. Experimental results are evaluated quantitatively to demonstrate the promise of the proposed methods.

## 1. INTRODUCTION

Listening to music and perceiving its structure is a fairly easy task for humans, even for listeners without formal musical training. However, building computational models to mimic this process is a hard problem. Furthermore, the amount of digital music has already become unfathomable. How to efficiently store and retrieve the digital content has become an important issue. This article presents our research on automatic music segmentation and summarization from audio signals. It will inquire scientifically into the nature of human perception of music, and offer a practical solution to difficult problems of machine intelligence for automated multimedia content analysis and information retrieval. Specifically, three problems will be addressed in this article: *segmentation based on tonality analysis*, *segmentation based on recurrent structural analysis* and *summarization (or music thumbnailing)*.

Successful solutions to the above problems can be used for web browsing, web searching and music recommendation. Some previous research already attempted to solve some of the similar problems. For segmentation, some research attempted to segment musical signals by detecting the locations where significant changes of statistical properties occur [2], which typically has nothing to do with the high-level structure. There has also been some research trying to consider semantic musical structure for segmentation.

For example, Sheh [9] proposed to use EM-based HMM for chord-based segmentation. For summarization, Dannenberg [8] presented a method to automatically detect the repeated patterns of musical signals using self-similarity analysis and clustering. Logan [13] attempted to use a clustering technique or HMM to find key phrases of songs. Bartsch [1] used the similarity matrix proposed by Foote [10][11] and chroma-based features for music thumbnailing. A variation of the similarity matrix was also proposed for music thumbnailing [15]. Previous research typically assumes that the most repeated pattern is the most representative part of music. There has been little research aiming at generating a global recurrent structure of music and a semantic segmentation based on it.

## 2. CHROMAGRAM REPRESENTATION

The chromagram, also called the Pitch Class Profile feature (PCP), is a frame-based representation of audio, very similar to Short-time Fourier Transform (STFT). It combines the frequency components in STFT belonging to the same pitch class (i.e., octave folding) and results in a 12-dimensional representation, corresponding to C, C#, D, D#, E, F, F#, G, G#, A, A#, B in music, or a generalized version of 24-dimensional representation for higher resolution and better control of noise floor [6].

Specifically, for the 24-dimensional representation, let  $X_{STFT}[K, n]$  denote the magnitude spectrogram of signal  $x[n]$ , where  $0 \leq K \leq NFFT - 1$  is the frequency index,  $NFFT$  is the FFT length. The chromagram of  $x[n]$  is

$$X_{PCP}[\tilde{K}, n] = \sum_{K: P(K)=\tilde{K}} X_{STFT}[K, n]. \quad (1)$$

The spectral warping between frequency index  $K$  in STFT and frequency index  $\tilde{K}$  in PCP is

$$P(K) = [24 \cdot \log_2(K / NFFT \cdot f_s / f_1)] \bmod 24, \quad (2)$$

where  $f_s$  is the sampling rate,  $f_1$  is the reference frequency corresponding to a note in the standard tuning system, for example, MIDI note C3 (32.7Hz). For the following two segmentation tasks, chromagram will be employed as the representation.

### 3. MUSIC SEGMENTATION BASED ON TONALITY ANALYSIS

This section describes an algorithm for detecting the key (or keys) of a musical piece. Specifically, given a musical piece (or part of it), the system will segment it into sections based on key change and identify the key of each section. Note that here we want to segment the piece and identify the key of each segment at the same time. A simpler task could be: given a segment of a particular key, detect the key of it.

In the following, the task of key detection will be divided into two steps: (1) Detect the key without considering its mode. For example, both C major and A minor will be denoted as key 1, C# major and A# minor will be denoted as key 2, and so on. Thus, there could be 12 different keys in this step; (2) Detect the mode (major or minor).

The task is divided in this way because diatonic scales are assumed and relative modes share the same diatonic scale. Step 1 attempts to determine the height of the diatonic scale. And again, both steps involve segmentation based on key (mode) change as well as identification of keys (modes).

The model used for key change detection should be able to capture the dynamic of sequences, and to incorporate prior musical knowledge easily since large volume of training data is normally unavailable. We propose to use Hidden Markov Models for this task, because HMM is a generative model for labeling structured sequence and satisfying both of the above properties. The hidden states correspond to different keys (or modes). The observations correspond to each frame represented as 24-dimensional chromagram vectors. The task will be decoding the underlying sequence of hidden states (keys or modes) from the observation sequence using Viterbi approach [16].

The parameters of HMM that need to be configured include:

- The number of states  $N$  corresponding to the number of different keys (=12) or the number of different modes (=2), respectively, in the two steps.
- The state transition probability distribution  $\mathbf{A} = \{a_{ij}\}$  corresponding to the probability of changing from key (mode)  $i$  to key (mode)  $j$ . Thus,  $\mathbf{A}$  is a  $12 \times 12$  matrix in step 1 and a  $2 \times 2$  matrix in step 2, respectively.

- The initial state distribution  $\Pi = \{\pi_i\}$  corresponding to the probability at which a piece of music starts from key (mode)  $i$ .
- The observation probability distribution  $\mathbf{B} = \{b_j(v)\}$  corresponding to the probability at which a chromagram  $v$  is generated by key (mode)  $j$ .

Due to the small amount of labeled audio data and the clear musical interpretation of the parameters, we will directly incorporate the prior musical knowledge by empirically setting  $\Pi$  and  $\mathbf{A}$  as follows:

$$\Pi = \frac{1}{12} \cdot \mathbf{1}, \quad (3)$$

where  $\mathbf{1}$  is a 12-dimensional vector in step 1 and a 2-dimensional vector in step 2. This configuration denotes equal probabilities of starting from different keys (modes).

$$\mathbf{A} = \begin{bmatrix} \textit{stayprob} & b & \dots & b \\ b & \textit{stayprob} & \dots & b \\ b & b & \dots & b \\ b & b & \dots & \textit{stayprob} \end{bmatrix}_{d \times d}, \quad (4)$$

where  $d$  is 12 in step 1 and is 2 in step 2. *stayprob* is the probability of staying in the same state and  $\textit{stayprob} + (d - 1) \cdot b = 1$ . For step 1, this configuration denotes equal probabilities of changing from a key to a different key. It can be easily shown that when *stayprob* gets smaller, the state sequence gets less stable (changes more often). In our experiment, *stayprob* will be varying within a range (e.g., [0.9900 0.9995]) in step 1 to see how it impacts the performance and be empirically set to  $1 - 10^{-20}$  in step 2.

For observation probability distribution, instead of Gaussian probabilistic models, commonly used for modeling observations of continuous random vectors in HMM, the cosine distances between the observation (the 24-dimensional chromagram vector) and pre-defined template vectors were used to represent how likely the observation was emitted by the corresponding keys or modes, i.e.,

$$b_j(v) = \frac{v \cdot \theta_j}{\|v\| \cdot \|\theta_j\|}, \quad (5)$$

where  $\theta_j$  is the template of state  $j$  (corresponding to the  $j^{\text{th}}$  key or mode). The advantage of using cosine distance instead of Gaussian distribution is that the key (or mode) is more correlated with the relative amplitudes of different frequency components rather than the absolute values of the amplitudes.

The template of a key was empirically set corresponding to the diatonic scale of that key. For example, the template for key 1 (C major or A minor) is  $\theta_{1,odd} = [1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 1]^T$ ,  $\theta_{1,even} = \mathbf{0}$ , where  $\theta_{1,odd}$  denotes the sub-vector of  $\theta_1$  with odd indexes (i.e.,  $\theta_1(1:2:23)$ ) and  $\theta_{1,even}$  denotes the sub-vector of  $\theta_1$  with even indexes (i.e.,  $\theta_1(2:2:24)$ ). This means we ignore the elements with even indexes when calculating the cosine distance. The templates of other keys were set simply by rotating  $\theta_1$  accordingly:

$$\theta_j = r(\theta_1, 2 \cdot (j-1)) , \quad (6)$$

$$\beta = r(\alpha, k), \text{ s.t. } \beta[i] = \alpha[(k+i) \bmod 24] , \quad (7)$$

where  $j=1, 2, \dots, 12$  and  $i, k=1, 2, \dots, 24$ . Let us also define  $24 \bmod 24 = 24$ .

For step 2, the templates of modes were empirically set as follows:  $\theta_{major,odd} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$ ,  $\theta_{minor,odd} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0]^T$ ,  $\theta_{major,even} = \theta_{minor,even} = 0$ . This setting comes from musical knowledge that typically in a major piece, the dominant (G in C major) appears more often than the submediant (A in C major), while in a minor piece, the tonic (A in A minor) appears more often than the subtonic (G in A minor). Note the templates need to be rotated accordingly (Equation 6, 7) based on its key detected from step 1.

#### 4. MUSIC SEGMENTATION BASED ON RECURRENT STRUCTURAL ANALYSIS

Music typically has a recurrent structure. This section describes research into automatic identification of the recurrent structure of music from acoustic signals. Specifically, an algorithm will be presented to output structural information, including both the form (e.g., AABABA) and the boundaries indicating the beginning and the end of each section. It is assumed that no prior knowledge about musical forms or the length of each section is provided, and the restatement of a section may have variations (e.g., different lyrics, tempos). This assumption requires both robustness and efficiency of the algorithm.

#### 4.1 Representation for Self-similarity Analysis

For visualizing and analyzing the recurrent structure of music, Foote [10,11] proposed a representation called *self-similarity matrix*. Each cell in the matrix denotes the similarity between a pair of frames in the musical signal. Here, instead of using similarity, we will use distance between a pair of frames, which results in a *distance matrix*. Specifically, let  $V = v_1 v_2 \dots v_n$  denote the feature vector sequence of the original musical signal  $x$ . It means we segment  $x$  into overlapped frames  $x_i$  and compute the feature vector  $v_i$  of each frame (e.g., chromagram). We then compute the distance between each pair of feature vectors according to some distance metric and obtain a matrix DM, which is the distance matrix. Thus,

$$DM(V) = [d_{ij}] = [\|v_i - v_j\|], \quad (8)$$

where  $\|v_i - v_j\|$  denotes the distance between  $v_i$  and  $v_j$ .

Since distance is typically symmetric, i.e.,  $\|v_i - v_j\| = \|v_j - v_i\|$ , the distance matrix is also symmetric.

One widely used definition of distance between vectors is based on cosine distance:

$$\|v_i - v_j\| = 0.5 - 0.5 \cdot \frac{v_i \bullet v_j}{\|v_i\| \|v_j\|}, \quad (9)$$

where we normalized the original definition of cosine distance to range from 0 to 1 instead of 1 to  $-1$  to be consistent with the non-negative property of distance. If we plot the distance matrix, we can often see the diagonal lines in the plot, which typically correspond to repetitions. Some previous research attempted to detect these diagonal patterns for identifying repetitions. However, not all repetitions can be easily seen from this plot due to variations of the restatements.

#### 4.2 Dynamic Time Warping for Music Matching

The above section showed that when part of the musical signal repeats itself nearly perfectly, diagonal lines appear in the distance matrix or its variation representations. However, if the repetitions have various variations (e.g., tempo change, different lyrics), which are very common in all kinds of music, the diagonal patterns will not be obvious. One solution is to consider approximate matching based on the self-similarity representation to allow flexibility of repetitions, especially tempo flexibility. Dynamic time warping was

widely used in speech recognition for similar purposes. Previous research has shown that it is also effective for music pattern matching [18]. Note that dynamic time warping is often mentioned in the context of speech recognition, where similar technique is cited as dynamic programming for approximate string matching, and the distance between two strings based on it is often called edit distance.

Assume we have two sequences and we need to find the match between the two sequences. Typically, one sequence is the input pattern ( $U = u_1u_2\dots u_m$ ) and the other ( $V = v_1v_2\dots v_n$ ) is the one in which to search for the input pattern. Here, we allow multiple appearances of pattern U in V. Dynamic time warping utilizes dynamic programming approach to fill in an m-by-n matrix WM based on Equation 10. The initial condition ( $i=0$  or  $j=0$ ) is set based on Figure 1.

$$DM[i, j] = \min \begin{cases} DM[i-1, j] + c_D[i, j], & (i \geq 1, j \geq 0) \\ DM[i, j-1] + c_I[i, j], & (i \geq 0, j \geq 1) \\ DM[i-1, j-1] + c_S[i, j], & (i, j \geq 1) \end{cases} \quad (10)$$

where  $c_D$  is cost of deletion,  $c_I$  is cost of insertion, and  $c_S$  is cost of substitution. The definitions of these parameters are determined differently for different applications. For example, we can define  $c_S[i, j] = \|u_i - v_j\|$  and  $c_D[i, j] = c_I[i, j] = 1.2 \cdot c_S[i, j]$  to penalize insertion and deletion based on the distance between  $u_i$  and  $v_j$ . We can also define  $c_D$  and  $c_I$  to be some constant.

		$\leq$	$\leq$	$\leq$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\leq$
	0	0	0	0	...	...	...	...	0
U <sub>1</sub>	e								
U <sub>2</sub>	2e								
...	...								
...	...								
U <sub>m</sub>	me								

**Figure 1: Dynamic time warping matrix WM with initial setting.  $e$  is a pre-defined parameter denoting the deletion cost.**

The last row of matrix WM (highlighted in Figure 1) is defined as a matching function  $r[i]$  ( $i=1, 2, \dots, n$ ). If there are multiple appearances of pattern U in V, local minima corresponding to these locations will

occur in  $r[i]$ . We can also define the overall cost of matching  $U$  and  $V$  (i.e., edit distance) to be the minimum of  $r[i]$ , i.e.,  $\|U - V\|_{DTW} = \min_i \{r[i]\}$ . In addition, to find the locations in  $V$  that match pattern  $U$  we need a trace-back step. The trace-back result is denoted as a trace-back function  $t[i]$  recording the index of the matching point. The time complexity of dynamic time warping is  $O(nm)$ , corresponding the computation needed for filling up matrix  $WM$ .

### 4.3 Recurrent Structural Analysis

Assuming that we have computed the feature vector sequence and the distance matrix  $DM$ , the algorithm follows four steps, which will be explained in the following four sections. All the parameter configurations are tuned based on the experimental corpus that will be described in Section 6.

#### 4.3.1 Pattern Matching

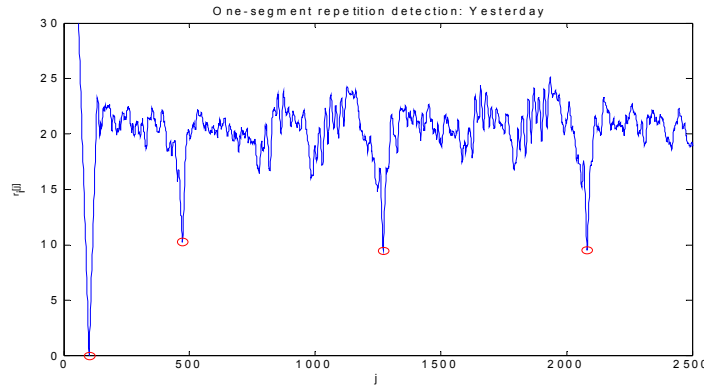
In the first step, we segment the feature vector sequence (i.e.,  $V = v_1 v_2 \dots v_n$ ) into overlapped segments of fixed length  $l$  (i.e.,  $S = S_1 S_2 \dots S_m$ ;  $S_i = v_{k_i} v_{k_i+1} \dots v_{k_i+l-1}$ ; e.g., 200 consecutive vectors with 150 vectors overlap) and compute the repetitive property of each segment  $S_i$  by matching  $S_i$  against the feature vector sequence starting from  $S_i$  (i.e.,  $V_i = v_{k_i} v_{k_i+1} \dots v_n$ ) using dynamic time warping. We define the cost of substitution  $c_S$  to be the distance between each pair of vectors. It can be obtained directly from the distance matrix  $DM$ . We also define the costs of deletion and insertion to be some constant:  $c_D[i, j] = c_I[i, j] = a$  (e.g.,  $a = 0.7$ ). For each matching between  $S_i$  and  $V_i$ , we obtain a matching function  $r_i[j]$ .

#### 4.3.2 Repetition Detection

This step detects the repetition of each segment  $S_i$ . To achieve this, the algorithm detects the local minima in the matching function  $r_i[j]$  for each  $i$ , because typically a repetition of segment  $S_i$  will correspond to a local minimum in this function.

There are four predefined parameters in the algorithm of detecting the local minima: the width parameter  $w$ , the distance parameter  $d$ , the height parameter  $h$ , and the shape parameter  $p$ . To detect local minima of

$r_i[j]$ , the algorithm slides the window of width  $w$  over  $r_i[j]$ . Assume the index of the minimum within the window is  $j_0$  with value  $r_i[j_0]$ , the index of the maximum within the window but left to  $j_0$  is  $j_1$  (i.e.,  $j_1 < j_0$ ) with value  $r_i[j_1]$ , and the index of the maximum within the window but right to  $j_0$  is  $j_2$  (i.e.,  $j_2 > j_0$ ) with value  $r_i[j_2]$ . If the following conditions are all satisfied: (1)  $r_i[j_1] - r_i[j_0] > h$  and  $r_i[j_2] - r_i[j_0] > h$  (i.e., the local minimum is deep enough); (2)  $\frac{r_i[j_1] - r_i[j_0]}{j_1 - j_0} > p$  or  $\frac{r_i[j_2] - r_i[j_0]}{j_2 - j_0} > p$  (i.e., the local minimum is sharp enough); (3) No two repetitions are closer than  $d$ , then the algorithm adds the minimum into the detected repetition set.



**Figure 2: One-segment repetition detection result of Beatles song *Yesterday*. The local minima indicated by circles correspond to detected repetitions of the segment.**

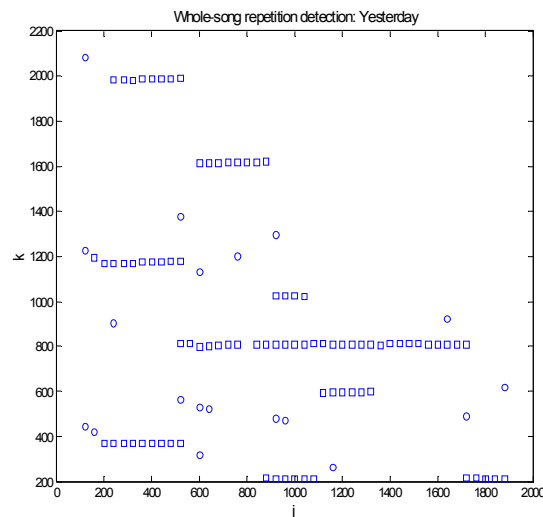
Figure 2 shows the repetition detection result of a particular segment for Beatles song *Yesterday*. In Figure 2, the four detected local minima correspond to the four restatements of the same melodic segment in the song (“Now it looks as though they are here to stay ...”, “There is a shadow hanging over me ...”, “I need a place to hide away ...”, “I need a place to hide away ...”). However, the repetitions detected may have add- or drop-errors, meaning a repetition is falsely detected or missed. The number of add-errors and that of the drop-errors are balanced by the predefined parameter  $h$ ; whenever the local minimum is deeper than height  $h$ , the algorithm reports a detection of repetition. Thus, when  $h$  increases, there are more drop-errors but fewer add-errors, and vice versa. For balancing between these two kinds of errors, the algorithm can

search within a range for the best value of  $h$ , so that the number of detected repetitions of the whole song is reasonable (e.g., # *total detected repetitions* /  $n \approx 2$ ).

For each detected minimum  $r_i[j^*]$  for  $S_i$ , let  $k^* = t_i[j^*]$ ; thus, it is detected that segment  $S_i = v_{k_i} v_{k_i+1} \dots v_{k_i+l-1}$  is repeated in  $V$  from  $v_{k_i+k^*}$ . Note that by the nature of dynamic programming, the matching part in  $V$  may not have length  $l$  due to the variations in the repetition.

### 4.3.3 Segment Merging

This step merges consecutive segments that have the same repetitive property into sections and generates pairs of similar sections. Figure 3 shows the repetition detection result of the Beatles song *Yesterday* after this step. In this figure, a circle or a square at  $(j, k)$  corresponds to a repetition detected in the last step (i.e., the segment starting from  $v_j$  is repeated from  $v_{j+k}$ ). Since typically one musical phrase consists of multiple segments, based on the configurations in previous steps, if one segment in a phrase is repeated by a shift of  $k$ , all the segments in this phrase are repeated by shifts roughly equal to  $k$ . This phenomenon can be seen from Figure 3, where the squares form horizontal patterns indicating consecutive segments have roughly the same shifts.



**Figure 3: Whole-song repetition detection result of Beatles song *Yesterday*.**

By detecting these horizontal patterns (denoted by squares in Figure 3) and discarding other detected repetitions (denoted by circles in Figure 3), add- or drop-errors in repetition detection are further reduced. The output of this step is a set of sections consisting of merged segments and the repetitive relation among these sections in terms of section-repetition vectors  $[j_1 \ j_2 \ shift_1 \ shift_2]$ , indicating that the segment starting from  $v_{j_1}$  and ending at  $v_{j_2}$  repeats roughly from  $v_{j_1+shift_1}$  to  $v_{j_2+shift_2}$ . Each vector corresponds to one horizontal pattern in the whole-song repetition detection result. For example, the vector corresponding to the left-bottom horizontal pattern in Figure 3 is [200 520 370 370].

#### 4.3.4 Structure Labeling

Based on the vectors obtained from the third step, the last step of the algorithm segments the whole piece into sections and labels each section according to the repetitive relation (i.e., gives each section a symbol such as “A”, “B”, etc.). This step will output the structural information, including both the form (e.g., AABABA) and the boundaries indicating the beginning and the end of each section. To solve conflicts that might occur, the rule is to always label the most frequently repeated section first. Specifically, the algorithm finds the most frequently repeated section based on the first two columns in the section-repetition vectors, and labels it and its shifted versions as section A. Then the algorithm deletes the vector already labeled, repeats the same procedure for the remaining section-repetition vectors, and labels the sections produced in each step as B, C, D and so on. If conflicts occur (e.g., a later labeled section has overlap with the previous labeled sections), the previously labeled sections will always remain intact, and the currently labeled section and its repetition will be truncated, so that only the unoverlapped part will be labeled as new.

## 5. MUSIC SUMMARIZATION

Music summarization (or, thumbnailing) aims at finding the most representative part of a musical piece. For example, for pop/rock songs, there are often catchy and repetitious parts (called the “hooks”), which can be implanted in your mind after hearing the song just once. This section analyzes the correlation between the representativeness of a musical part and its location within the global structure, and proposes a method to automate music summarization. Results will be evaluated both by objective criteria and human experiments.

In general, it would be helpful if the song has been segmented into meaningful sections before summarization for locating structurally accented locations, e.g., the beginning or the ending of a section, especially a chorus section. Once we have the recurrent structure of a song, we can have different music summarization strategies for different applications or for different types of users. In the following, the methods we present will find the most representative part of music (specifically, hooks of pop/rock music) based on the result of recurrent structural analysis.

### 5.1 Section-beginning Strategy (SBS)

The first strategy assumes that the most repeated part of the music is also the most representative part and the beginning of a section is typically essential. Thus, this strategy, illustrated by Figure 4 chooses the beginning of the most repeated section as the thumbnail of the music. The algorithm first finds the most repeated sections based on the structural analysis result, takes the first section among these and truncates its beginning (20 seconds in this experiment) as the thumbnail.

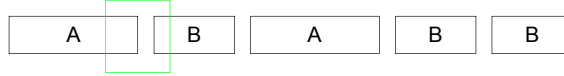


**Figure 4: Section-beginning strategy.**

### 5.2 Section-transition Strategy (STS)

We also investigated the music thumbnails at some commercial music web sites for music sales (e.g., Amazon.com, music.msn.com) and found that the thumbnails they use do not always start from the beginning of a section and often contain the transition part (end of section A and beginning of section B). This strategy assumes that the transition part can give a good overview of both sections and is more likely to capture the hook (or, title) of the song, though it typically will not give a thumbnail right at the beginning of a phrase or section.

Based on the structural analysis result, the algorithm finds a transition from section A to section B; and then it truncates the end of section A, the bridge and the beginning of section B (shown in Figure 5). The boundary accuracy is not very important for this strategy.



**Figure 5: Section-transition strategy.**

To choose the transition for summarization, three methods were investigated:

- STS-I: Choose the transition such that the sum of the repeated times of A and those of B is maximized; if there is more than one such transition, the first one will be chosen. In the above example, since there are only two different sections, either  $A \rightarrow B$  or  $B \rightarrow A$  satisfies the condition; thus the first transition from A to B will be chosen.

- STS-II: Choose the most repeated transitions between different sections; if there is more than one such transition, the first one will be chosen. In the above example,  $A \rightarrow B$  occurs twice,  $B \rightarrow A$  occurs once; thus the first transition from A to B will be chosen.

- STS-III: Choose the first transition right before the most repeated section. In the above example, B is the most repeated section; thus the first transition from A to B will be chosen.

Although in the above example, all these three methods will choose the same transition for summarization, we can come out with various other forms where the three methods will choose different transitions.

## 6. EXPERIMENT AND EVALUATION

### 6.1 Evaluation of Segmentation

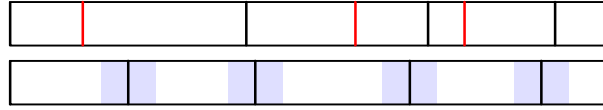
To evaluate segmentation results, two aspects need to be considered: label accuracy (how the computed label of each frame is consistent with the actual label) and segmentation accuracy (how the detected locations of transitions are consistent with the actual locations).

Label accuracy is defined as the proportion of frames that are labeled correctly, i.e.,

$$\text{Label accuracy} = \frac{\# \text{ frames labeled correctly}}{\# \text{ total frames}}. \quad (11)$$

Two metrics were proposed and used for evaluating segmentation accuracy. *Precision* is defined as the proportion of detected transitions that are relevant. *Recall* is defined as the proportion of relevant transitions

detected. Thus, if  $B=\{\text{relevant transitions}\}$ ,  $C=\{\text{detected transitions}\}$  and  $A=B \cap C$ , from the above definition,  $Precision = \frac{A}{C}$  and  $Recall = \frac{A}{B}$ .



**Figure 6: An example for measuring segmentation performance (above: detected transitions; below: relevant transitions).**

To compute precision and recall, we need a parameter  $w$ : whenever a detected transition  $t_1$  is close enough to a relevant transition  $t_2$  such that  $|t_1 - t_2| < w$ , the transitions are deemed identical (a *hit*). Obviously, greater  $w$  will result in higher precision and recall. In the example shown in Figure 6, the width of each shaded area corresponds to  $2w - l$ . If a detected transition falls into a shaded area, there is a *hit*. Thus, the precision in this example is  $3/6=0.5$ ; the recall is  $3/4=0.75$ . Given  $w$ , higher precision and recall indicates better segmentation performance. In our experiment (512 window step at 11kHz sampling rate),  $w$  will vary within a range to see how precision and recall vary accordingly: 10 frames ( $\sim 0.46s$ ) to 80 frames ( $\sim 3.72s$ ).

It can be shown that, given  $n$  and  $l$ , precision increases by increasing  $w$  (i.e., increasing  $m$ ); recall increases by increasing  $k$  or  $w$ .

For recurrent structural analysis, besides label accuracy, precision and recall, one extra metric - *formal distance* - will be used to evaluate the difference between the computed form and the true form. It is defined as the edit distance between the strings representing different forms. For example, the formal dissimilarity between structure AABABA and structure AABBABBA is 2, indicating two insertions from the first structure to the second structure (or, two deletions from the second structure to the first structure; thus this definition of distance is symmetric). Note that how the system labels each section is not important as long as the repetitive relation is the same; thus, structure AABABA is deemed as equivalent (0-distance) to structure BBABAB, or structure AACACA.

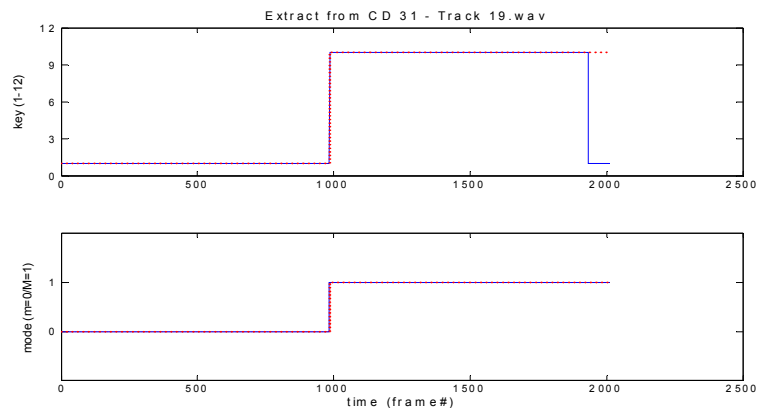
## 6.2 Evaluation of Thumbnailing

Based on the previous human experiments, five criteria for pop/rock music are considered for evaluating the summarization result. These criteria include: (1) The percentage of generated thumbnails that contain a vocal portion; (2) The percentage of generated thumbnails that contain the song’s title; (3) The percentage of generated thumbnails that start at the beginning of a section; (4) The percentage of generated thumbnails that start at the beginning of a phrase; (5) The percentage of generated thumbnails that capture a transition between different sections.

## 6.3 Experimental Results

### 6.3.1 Performance of Key Detection

Ten classical piano pieces were used in the experiment of key detection, since the chromagram representation of piano music has a good mapping between its structure and its musical interpretation. These pieces were chosen randomly as long as they have fairly clear tonal structure (relatively tonal instead of atonal). The “truth” was manually labeled by the author based on the score notation for comparison with the computed results. The data were mixed into 8-bit mono and down-sampled to 11kHz. Each piece was segmented into frames of 1024 samples with 512 samples overlap.

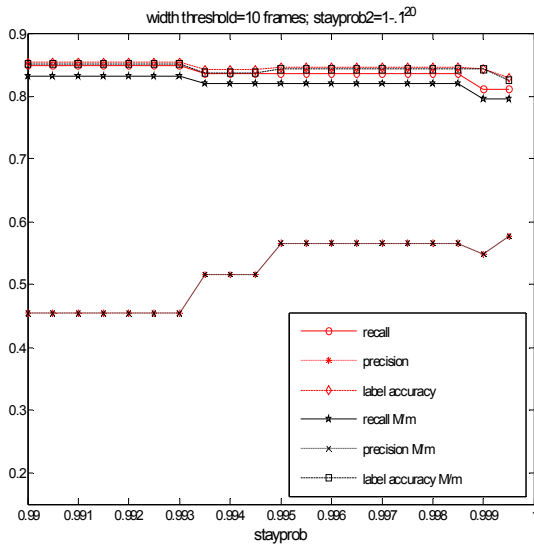


**Figure 7: Detection of key change in “Mozart: Sonata No. 11 In A ‘Rondo All Turca, 3<sup>rd</sup> movement”” (solid line: computed key; dotted line: truth)**

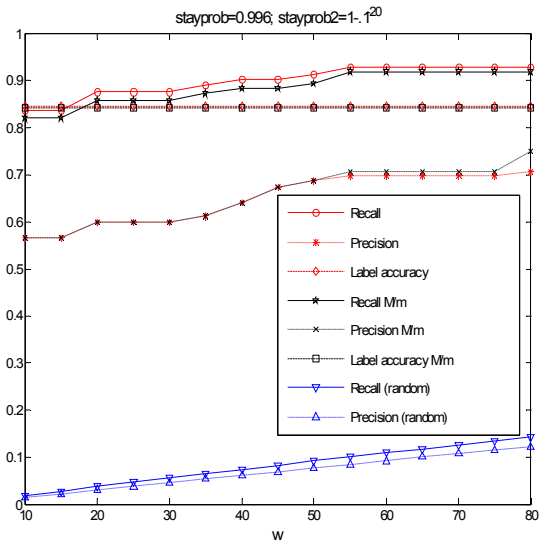
Figure 7 shows key detection result of Mozart's piano sonata No. 11 with  $stayprob=0.996$  in step 1 and  $stayprob2=1-1^{-20}$  in step 2. The figure above presents the result of key detection without considering mode (step 1) and the figure below presents the result of mode detection (step 2).

To show label accuracy, recall and precision of key detection averaged over all the pieces, we can either fix  $w$  and change  $stayprob$  (Figure 8-a), or fix  $stayprob$  and change  $w$  (Figure 8-b). In Figure 8-a, two groups of results are shown: one corresponds to the performance of step 1 without considering modes; the other corresponds to the overall performance of key detection with mode into consideration. It clearly shows that, when  $stayprob$  increases, precision also increases while recall and label accuracy decrease. In Figure 8-b, three groups of results are shown: one corresponds to the performance of step 1 without considering modes; one corresponds to the overall performance of key detection with mode taken into consideration; and one corresponds to recall and precision based on random segmentation.

Additionally, label accuracy based on random should be around 8%, without considering modes. It clearly shows that when  $w$  is increasing, the segmentation performance (recall and precision) is also increasing. Note that label accuracy is irrelevant to  $w$ .



(a)



(b)

Figure 8: Performance of key detection. (a) varying  $stayprob$  ( $w=10$ ); (b) varying  $w$  ( $stayprob=0.996$ ).

### 6.3.2 Performance of Recurrent Structural Analysis

Two experimental corpora were tested. One corpus is piano music same as the one used for key detection. The other consists of the 26 Beatles songs in the two CDs *The Beatles* (1962-1966). All of these musical pieces have clear recurrent structures, so that the true recurrent structures were labeled easily for comparison. The data were mixed into 8-bit mono and down-sampled to 11kHz.

To qualitatively evaluate the results, figures as shown in Figure 9 are used to compare the structure obtained from the algorithm to the true structure obtained by manually labeling the repetitions. Sections in the same color indicate restatements of the section. Sections in the lightest gray correspond to the parts with no repetition.



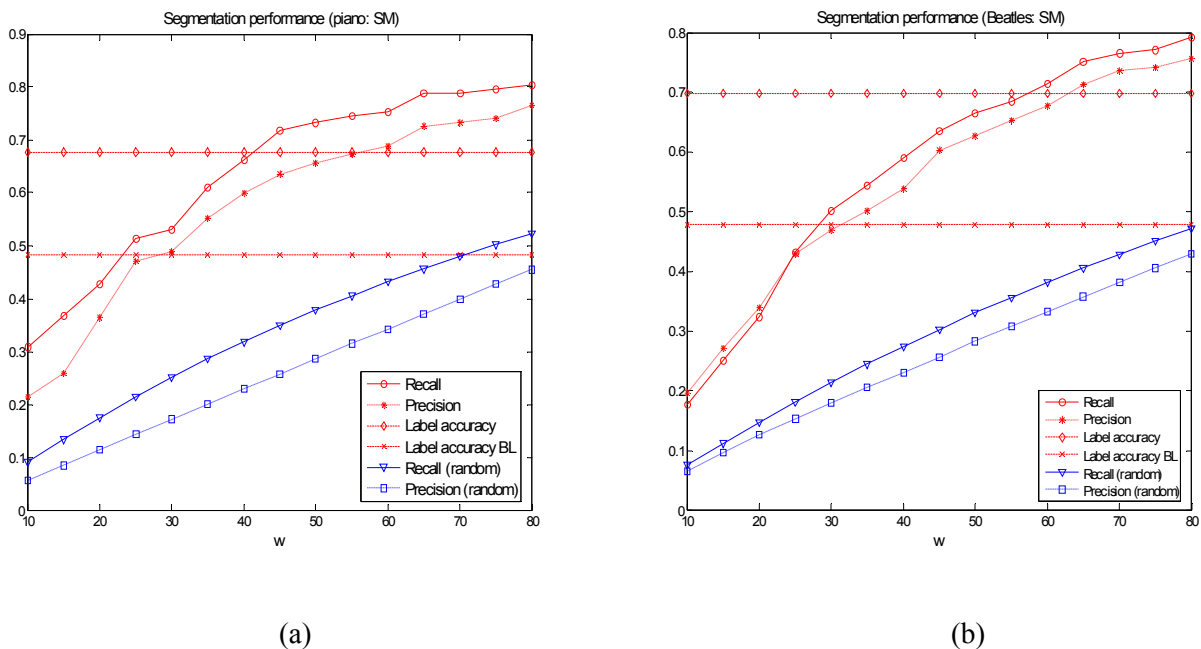
**Figure 9: Comparison of the computed structure using DM (above) and the true structure (below) of *Yesterday*.**

Figure 10 shows the segmentation performances of the two data corpora, respectively, with varying  $w$ . In each plot, the bottom two curves correspond to upper bounds of recall and precision based on random segmentation. The bottom horizontal line shows the baseline label accuracy of labeling the whole piece as one section.

The experimental result shows that the performance of 7 out of 10 piano pieces and 17 out of 26 Beatles songs have formal distances less than or equal to 2. The label accuracy is significantly better than the baseline and the segmentation performance is significantly better than random segmentation. This demonstrates the promise of the method.

We also found that the computed boundaries of each section were often slightly shifted from the true boundaries. This was mainly caused by the inaccuracy of the approximate pattern matching. To tackle this

problem, other musical features (e.g., chord progressions, change in dynamics) should be used to detect local events so as to locate the boundaries accurately.



**Figure 10: Segmentation performance of recurrent structural analysis. (a) Classical piano music; (b) Beatles songs.**

### 6.3.3 Performance of Thumbnailing

Human experiments (not covered in this paper) have shown that using the beginning of a piece is a fairly good summarization strategy for classical music. Here we will only consider pop/rock music for evaluating summarization results. Table 1 shows the performance of all the strategies (SBS, STS-I, STS-II and STS-III) presented in Section 5 using the 26 Beatles songs. For evaluating transition criterion (5<sup>th</sup> column), only the 22 songs in our corpus that have different sections were counted.

The comparison of the thumbnailing strategies clearly shows that the section-transition strategies generate a lower percentage of thumbnails starting at the beginning of a section or a phrase, while these thumbnails are more likely to contain transitions. SBS has the highest chance to capture the vocal and STS-I has the highest chance to capture the title. It is possible, though, to achieve better performance using this strategy, if we can improve the structural analysis accuracy in the future.

**Table 1: 20-second music summarization result.**

	Vocal	Title	Beginning of a section	Beginning of a phrase	Transition
SBS	100%	65%	62%	54%	23%
STS-I	96%	73%	42%	46%	82%
STS-II	96%	62%	31%	46%	91%
STS-III	96%	58%	31%	50%	82%

## 7. CONCLUSION AND FUTURE WORK

This paper presents our research into segmenting music based on its semantic structure such as key change, and recurrent structure, and summarizing music based on its structure. Experimental results were evaluated quantitatively, which demonstrate the promise of the proposed methods. Future directions include inferring the hierarchical structures of music and incorporating more music knowledge to achieve better accuracy. Furthermore, a successful solution to any of these problems depends on the study of human perception of music, for example, what makes part of music sounds like a complete phrase and what makes it memorable or distinguishable. Human experiments are always necessary for exploring such questions.

## 8. REFERENCES

- [1] M.A. Bartsch and G.H. Wakefield, "To Catch a Chorus: Using Chroma-based Representations for Audio Thumbnailing," *In Proc. Workshop on Applications of Signal Processing to Audio and Acoustics*, 2001.
- [2] A.L. Berenzweig, and D. Ellis, "Locating Singing Voice Segments within Music Signals," *Proceedings of Workshop on Applications of Signal Processing to Audio and Acoustics*, NY, 2001.
- [3] G. Burns, "A typology of 'hooks' in popular records," *Popular Music*, 6/1, pp. 1-20, January 1987.
- [4] W. Chai and B. Vercoe, "Music Thumbnailing via Structural Analysis," *In Proc. of ACM Multimedia Conference*, 2003.
- [5] W. Chai, "Structural Analysis of Musical Signals via Pattern Matching," *In Proc. International Conference on Acoustics, Speech and Signal Processing*, 2003.
- [6] W. Chai and B.L. Vercoe, "Structural Analysis of Musical Signals for Indexing and Thumbnailing," *In Proc. Joint Conference on Digital Libraries*, 2003.

- [7] C. Chuan and E. Chew, "Polyphonic Audio Key-Finding Using the Spiral Array CEG Algorithm," *Proceedings of International Conference on Multimedia and Expo*, Amsterdam, Netherlands, July 6-8, 2005.
- [8] R.B. Dannenberg and N. Hu, "Pattern Discovery Techniques for Music Audio," *In Proc. International Conference on Music Information Retrieval*, October 2002.
- [9] A. Sheh and D. Ellis, "Chord Segmentation and Recognition using EM-Trained Hidden Markov Models," *4th International Symposium on Music Information Retrieval ISMIR-03*, Baltimore, October 2003.
- [10] J. Foote, "Visualizing Music and Audio using Self-Similarity," *In Proc. ACM Multimedia Conference*, 1999.
- [11] J. Foote, "Automatic Audio Segmentation using a Measure of Audio Novelty." *In Proc. of IEEE International Conference on Multimedia and Expo*, vol. WE, pp. 452-455, 2000.
- [12] J.L. Hsu, C.C. Liu, and L.P. Chen, "Discovering Nontrivial Repeating Patterns in Music Data," *IEEE Transactions on Multimedia*, Vol. 3, No. 3, pp. 311-325, September 2001.
- [13] B. Logan and S. Chu, "Music Summarization using Key Phrases," *In Proc. International Conference on Acoustics, Speech and Signal Processing*, 2000.
- [14] T. Kemp, M. Schmidt, M. Westphal, and A. Waibel, "Strategies for Automatic Segmentation Audio Data," *In Proc. International Conference on Acoustics, Speech and Signal Processing*, 2000.
- [15] G. Peeters, A.L. Burthe and X. Rodet, "Toward Automatic Music Audio Summary Generation from Signal Analysis," *In Proc. International Conference on Music Information Retrieval*, October 2002.
- [16] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, 77(2):257—286, 1989.
- [17] N. Scaringella, G. Zoia, and D. Mlynek, "Automatic Genre Classification of Music Content: a Survey," *IEEE Signal Processing Magazine, Special Issue on Semantic Retrieval of Multimedia*, 2006.
- [18] C. Yang, "MACS: Music Audio Characteristic Sequence Indexing for Similarity Retrieval," *Proceedings of Workshop on Applications of Signal Processing to Audio and Acoustics*, 2001.