# Structural Analysis of Musical Signals for Indexing and Thumbnailing

Wei Chai Barry Vercoe
*MIT Media Laboratory*
*{chaiwei, bv}@media.mit.edu*

## Abstract

*A musical piece typically has a repetitive structure. Analysis of this structure will be useful for music segmentation, indexing and thumbnailing. This paper presents an algorithm that can automatically analyze the repetitive structure of musical signals. First, the algorithm detects the repetitions of each segment of fixed length in a piece using dynamic programming. Second, the algorithm summarizes this repetition information and infers the structure based on heuristic rules. The performance of the approach is demonstrated visually using figures for qualitative evaluation, and by two structural similarity measures for quantitative evaluation. Based on the structural analysis result, this paper also proposes a method for music thumbnailing. The preliminary results obtained using a corpus of Beatles' songs show that automatic structural analysis and thumbnailing of music are possible.*

## 1. Introduction

A musical piece typically has a repetitive structure. For example, a song may have a structure of ABA, indicating a three-part compositional form in which the second section contrasts with the first section, and the third section is a restatement of the first. Methods for automatically detecting the repetitive structure of a musical piece from acoustical signals is valuable for information retrieval systems and digital libraries; for example, the result can be used for indexing the musical content or for music thumbnailing.

There has been some recent research on this topic. Dannenberg and Hu presented a method to automatically detect the repetitive structure of musical signals [6]. The process consists of searching for similar segments in a musical piece, forming clusters of similar segments, and explaining the musical structure in terms of these clusters. Three representations were investigated: monophonic pitch estimation, chroma representation, and polyphonic transcription followed by harmonic analysis. Although the promise of this method was demonstrated in several examples, there was no quantitative evaluation of the method in their paper.

Two topics closely related to structural analysis of music have also been investigated. One is *music thumbnailing* (or *music summarization*), which aims at finding the most representative part (normally assumed to be the most repeated section) of a song. Some research on music thumbnailing deals with symbolic musical data (e.g., MIDI files and scores) [10]. There have also been studies on thumbnailing of musical signals. Logan and Chu attempted to use a clustering technique or Hidden Markov Models to find key phrases of songs. Mel Cepstral features were used to characterize each song [11].

The other related topic is *music segmentation*. Most previous research in this area attempted to segment musical pieces by detecting the locations where a significant change of statistical properties occurs [1]. This method is more appropriate for segmenting different local events rather than segmenting the semantic components of the global structure.

Additionally, Foote proposed a representation called a *similarity matrix* for visualizing and analyzing the structure of audio, including symbolic music, acoustic musical signals or more general audio [7][8]. One attempt using this representation was to locate points of significant change in music (e.g., score analysis) or audio (e.g., speech/music segmentation) [8]. Bartsch and Wakefield used the similarity matrix and chroma-based features for music thumbnailing [4]. A variation of the similarity matrix was also proposed for music thumbnailing [12].

This paper describes research into automatic identification of the repetitive structure of musical pieces from acoustic signals. Specifically, an algorithm is presented that will 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. This assumption requires both robustness and efficiency of the algorithm.

Two novel structural similarity measures are also proposed in this paper to quantitatively evaluate the performance of the algorithm, in addition to the qualitative evaluation presented by figures.

The remainder of this paper is organized as follows. Section 2 illustrates the structural analysis approach. Section 3 presents the experimental results. Section 4 explains how the structural analysis result can be used for music thumbnailing. Section 5 gives conclusions and proposes future work.

## 2. Approach

This section illustrates the structural analysis method, which follows five steps and is also illustrated in Figure 1:

1) Segment the signal into frames and compute the feature of each frame;
2) Segment the feature sequence into overlapped segments of fixed length and compute the repetition property of each segment using dynamic programming;
3) Detect the repetitions of each segment by finding the local minima in the dynamic programming result;
4) Merge consecutive segments that have the same repetitive property into sections and generate pairs of similar sections;
5) Segment and label the repetitive structure.

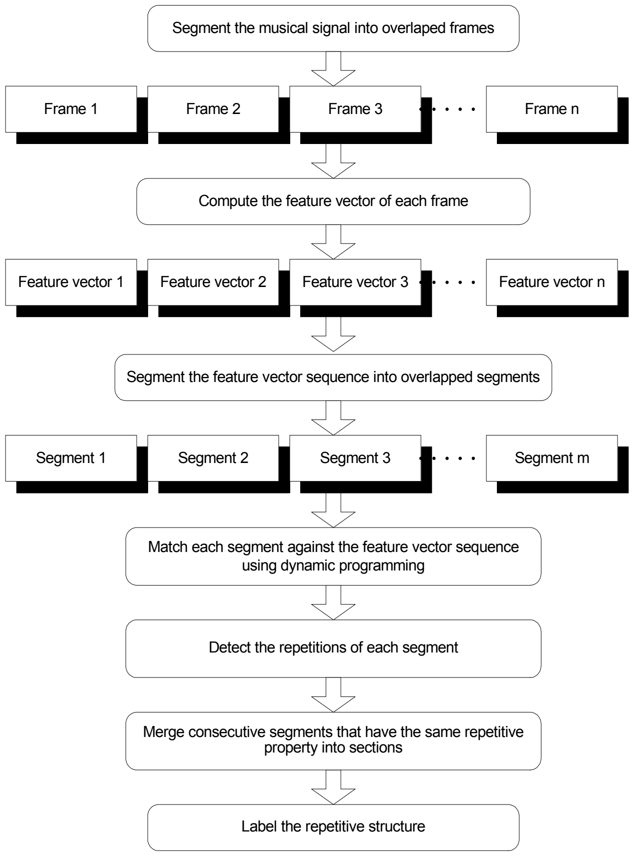

Segment the musical signal into overlaped frames
Frame 1
Frame 2
Frame 3
Frame n
Compute the feature vector of each frame
Feature vector 1
Feature vector 2
Feature vector 3
Feature vector n
Segment the feature vector sequence into overlapped segments
Segment 1
Segment 2
Segment 3
Segment m
Match each segment against the feature vector sequence using dynamic programming
Detect the repetitions of each segment
Merge consecutive segments that have the same repetitive property into sections
Label the repetitive structure

**Figure 1. Overview of the approach.**

The following five sections explain each step in detail. All the parameter configurations are tuned based on the experimental corpus, which is described in Section 3.

### 2.1. Feature extraction

The algorithm first segments the signal into overlapped frames (e.g., 1024-sample window length with 512- sample overlap) and computes the feature of each frame.

Two representations are investigated in this paper. One is the pitch representation, which uses autocorrelation [13] to estimate the main frequency component of each frame. Although all the test data in the experiment are polyphonic, it turns out that, for musical signals with a leading vocal, this feature can still capture much information. The other representation explored is the spectral representation, i.e., FFT magnitude coefficients.

The distance between two pitch features $v_1$ and $v_2$ is defined as

$$d_p(v_1, v_2) = \frac{|v_1 - v_2|}{normalization\ factor} \qquad (1)$$

The distance between two spectral features $\vec{v}_1$ and $\vec{v}_2$ is defined as

$$d_f(\vec{v}_1, \vec{v}_2) = 0.5 - 0.5 \cdot \frac{\vec{v}_1 \bullet \vec{v}_2}{|\vec{v}_1||\vec{v}_2|} \qquad (2)$$

In both cases, a distance value ranges between 0 and 1.

### 2.1. Pattern matching

After computing the feature vector $v_j$ (one-dimensional vector for the pitch representation and N-dimensional vector for the spectral representation) for each frame, the algorithm segments the feature vector sequence $V[1,n] = \{v_j \mid j = 1,...,n\}$ (n is the number of frames) into overlapped segments of fixed length $l$ (e.g., 200 consecutive vectors with 150 vectors overlap). Since previous research has shown that dynamic programming is effective for music pattern matching [9][14], here dynamic programming is used to match each segment (i.e., $s_i = V[j, j+l-1]$) with the feature vector sequence starting from this segment (i.e., $V[j,n]$). The dynamic programming algorithm will fill in a matrix $M_i$ (i.e., the dynamic programming matrix of the $i^{th}$ segment) as shown in Figure 2 based on Equation 3.

$$M[p,q] = \min \begin{cases} M[p-1,q]+e & (p \geq 1) \\ M[p,q-1]+e & (q \geq 1) \\ M[p-1,q-1]+c & (p,q \geq 1) \\ 0 & o.w. \end{cases} \qquad (3)$$

where $e$ is the insertion or deletion cost, $c$ is the distance between the two corresponding feature vectors, which has been defined in Section 2.1. The last row of matrix $M_i$ is defined as function $d_i[r]$ (shown as the shaded area in Figure 2). In addition, the trace-back step of dynamic programming determines the actual alignments (i.e., the locations in $V[j,n]$ matching the beginning of $s_i$) that result in $d_i[r]$. The trace-back result is denoted as $t_i[r]$.

| | | vj | V(j+1) | V(j+2) | ... | ... | ... | ... | ... | vn |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | ... | ... | ... | ... | ... | 0 |
| vj | e | | | | | | | | | |
| V(j+1) | 2e | | | | | | | | | |
| ... | ... | | | | | | | | | |
| ... | ... | | | | | | | | | |
| V(j+l-1) | le | | | | | | | | | |

**Figure 2. Dynamic programming matrix *M*$_i$.**

This step is the most time consuming one in the structural analysis algorithm; its time complexity is $O(n^2)$.

## 2.3. Repetition detection

This step of the algorithm detects the repetition of each segment. To achieve this, the algorithm detects the local minima in the function $d_i[r]$ for each $i$, because normally a repetition of segment $i$ 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 $d_i[r]$, the algorithm slides the window of width $w$ over $d_i[r]$. Assume the index of the minimum within the window is $r_0$ with value $d_i[r_0]$, the index of the maximum within the window but left to $r_0$ is $r_1$ (i.e., $r_1<r_0$) with value $d_i[r_1]$, and the index of the maximum within the window but right to $r_0$ is $r_2$ (i.e., $r_2>r_0$) with value $d_i[r_2]$. If

(1) $d_i[r_1]-d_i[r_0]>h \quad and \quad d_i[r_2]-d_i[r_0]>h$ (i.e., the local minimum is deep enough); and

(2) $\frac{d_i[r_1]-d_i[r_0]}{r_1-r_0}>p \quad or \quad \frac{d_i[r_2]-d_i[r_0]}{r_2-r_0}>p$ (i.e., the local minimum is sharp enough); and

(3) No two repetitions are closer than $d$,

the algorithm adds the minimum into the detected repetition set. In our experiment, we set *w=400, d=5, and p=0.1.* Figure 3 shows the repetition detection result of one segment in the song *Yesterday*.

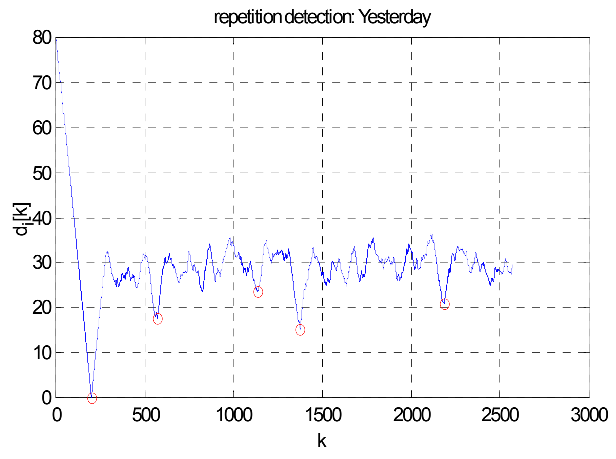


**Figure 3. One-segment repetition detection result of *Yesterday*. The local minima indicated by circles correspond to detected repetitions of the segment.**

The repetitions detected may have add or drop errors, meaning a repetition is falsely detected or missed. For example, in Figure 3, the first, the second, the fourth and the fifth detected local minima correspond to the four restatements of the same melodic segment in the song ("… here to stay …", "… over me …", "… hide away …", "… hide away …"). However, there is an add error occurring at the third detected local minimum. 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 vise versa. For balancing between these two kinds of errors, the algorithm searches within a range for the best value of $h$ (e.g., decreasing from 10 to 5 with step -1 for the pitch representation, and decreasing from 12 to 8 with step -1 for the spectral representation), so that the number of detected repetitions of the whole song is reasonable (e.g., $\#\ detected\ \ repetition\,s\,/\,n \approx 2$).

For each detected minimum $d_i[r^*]$ for $s_i=V[j,j+l-1]$, let $k^*=t_i[r^*]$; thus, it is detected that the segment starting at $v_j$ is repeated at $v_{j+k^*}$. Please note that by the nature of dynamic programming, the matching part may not be of length $l$ due to the variations in the repetition.

## 2.4. Segment merging

The algorithm merges consecutive segments that have the same repetitive property into sections and generates pairs of similar sections in this step.

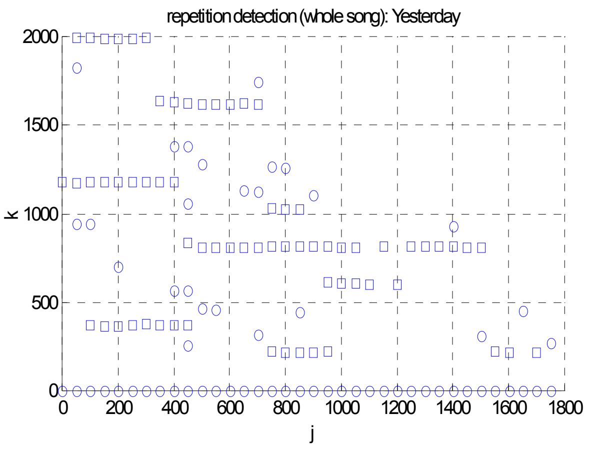


**Figure 4. Whole-song repetition detection result of *Yesterday*. A circle or a square at location (j, k) indicates that the segment starting at $v_j$ is detected to repeat at $v_{j+k}$.**

Figure 4 shows the repetition detection result of the 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 at $v_j$ is repeated at $v_{j+k}$). 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 4, where the squares have the horizontal patterns indicating consecutive segments have roughly the same shifts.

By detecting these horizontal patterns (denoted by squares in Figure 4) and discarding other detected repetitions (denoted by circles in Figure 4) obtained from the third step, the effects of add/drop errors are further reduced.

The output of this step is a set of merged sections in terms of tuples $< j_1, j_2, shift >$, indicating that the segment starting at $v_{j_1}$ and ending at $v_{j_2}$ repeats roughly from $v_{j_1+shift}$ to $v_{j_2+shift}$. Each tuple corresponds to one horizontal pattern in the whole-song repetition detection result. For example, the tuple corresponding to the left-bottom horizontal pattern in Figure 4 is <100, 450, 370>. Since the shifts of repetitions may not be exactly the same for segments in the merged section, the shift of the whole section is the average value.

## 2.5. Structure labeling

Based on the tuples obtained from the fourth 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.). Thus, 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 for labeling is always labeling the most frequently repeated section first. Specifically, the algorithm finds the most frequently repeated section based on the first two columns in the tuples, and labels it and its shifted versions as section A. Then the algorithm deletes the tuples already labeled, repeats the same procedure for the remaining tuples, and labels 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 previous labeled sections will always remain intact and the current section will be truncated.

## 3. Experiment and evaluation

This section presents the experimental results and evaluations of the structural analysis approach.

### 3.1. Data set

The experimental corpus consists of the 26 Beatles' songs in the two CDs *The Beatles* (1962-1966). All these songs have clear repetitive structures and leading vocal. The data were mixed to 8-bit mono and down-sampled to 11kHz.

### 3.2. Measures of structural similarity

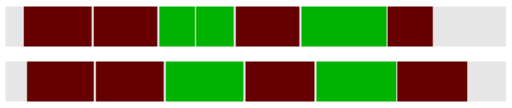


**Figure 5. Comparison of the computed structure (above) using the pitch representation and the ideal structure (below) of *Yesterday*. Sections in the same color indicate restatements of the section. Sections in the lightest grey correspond to the sections with no repetition.**

To qualitatively evaluate the results, figures as shown in Figure 5 are used to compare the structure obtained from the algorithm with the ideal structure obtained by manually labeling the repetition. This paper also proposes two measures of structural similarity to quantitatively evaluate the result. Both of the measures need to be as small as possible, ideally equal to zero.

*Measure 1 (structural measure)* is defined as the edit distance between the strings representing different forms. For the example in Figure 5, the distance between the ideal structure AABABA and the computed structure AABBABA is 1, indicating one insertion. Here how the algorithm labels each section is not important as long as the repetitive relation is the same; thus, this ideal structure is deemed as equivalent (0-distance) to structure BBABAB, or structure AACACA.

*Measure 2 (boundary measure)* is mainly used to evaluate how accurate the boundaries of each section are. It is defined as

$$BM = (1-r)/s \qquad (4)$$

where $r$ is the ratio of the length of parts where both structures have the same labeling to the whole length, and $s$ is the number of the repetitive sections in the ideal structure.

## 3.3. Results

Figure 6 and Figure 7 show the structural and boundary measures of the experimental results using both the pitch representation and the spectral representation. In Figure 7, the baseline results corresponding to labeling the whole song as a single section are also plotted for a comparison.

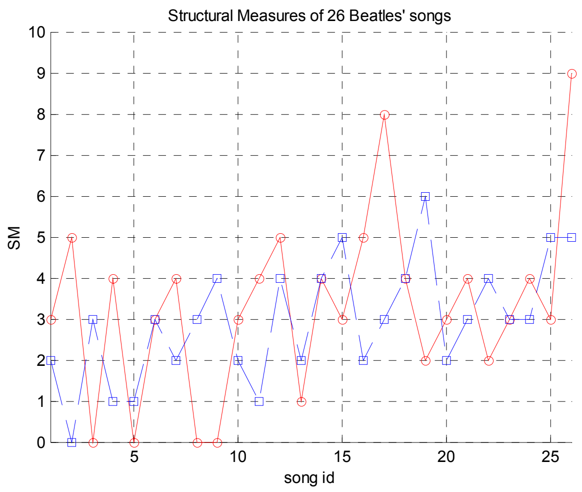
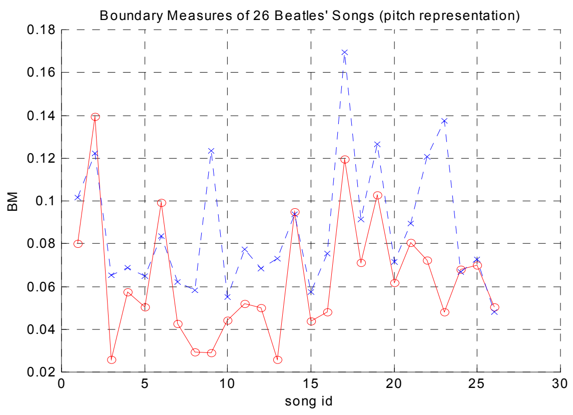


**Figure 6. Structural measures of the 26 Beatles' songs. The solid line with circle markers corresponds to the pitch representation results. The dashed line with square markers corresponds to the spectral representation results.**



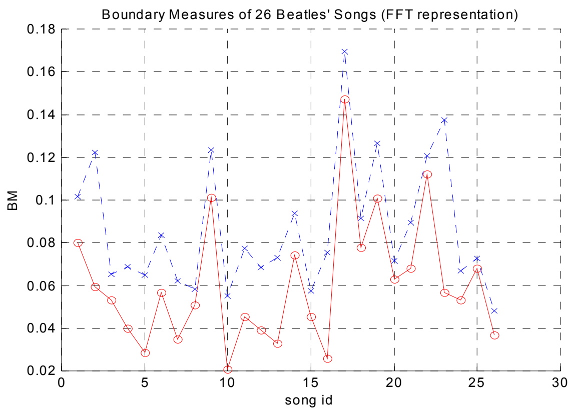


**Figure 7. Boundary measures of the 26 Beatles' songs. The solid lines with circle markers correspond to the computed results (above: pitch representation; below: spectral representation). The dotted lines with x markers correspond to the baseline.**

It is easily seen from the above figures that the performance of the third, the eighth and the ninth song using the pitch representation are the best (the structural measures are 0 and the boundary measures are low). For example, the result of the third song *From me to you* using the pitch representation is shown in Figure 8.

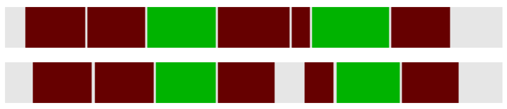


**Figure 8. Comparison of the computed structure (above) using the pitch representation and the ideal structure (below) of *From me to you*.**

The one of the worst performance is the seventeenth song *Day tripper* using the pitch representation, whose result is shown in Figure 9.

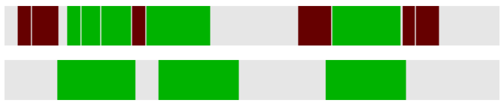


**Figure 9. Comparison of the computed structure (above) using the pitch representation and the ideal structure (below) of *Day tripper*.**

Some interesting results also occur. For example, for the twelfth song *Ticket to ride*, although the computed structure using the spectral representation is different from the ideal structure as shown in Figure 10, it also looks reasonable by seeing section A in the computed structure as the combination of section A and section B in the ideal structure.

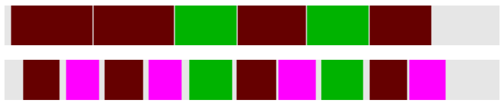


**Figure 10. Comparison of the computed structure (above) using the spectral representation and the ideal structure (below) of *Ticket to ride*.**

### 3.4. Discussions

The experimental result shows that, by either the pitch representation or the spectral representation, the performance of 15 out of 26 songs have structural measures less than or equal to 2 (Figure 6) and the results of all the songs have boundary measures better than the baseline (Figure 11). This demonstrates the promise of the method.

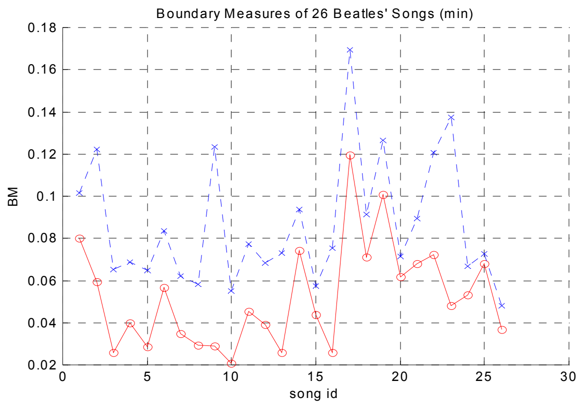


**Figure 11. Boundary measures of the 26 Beatles' songs. The solid line with circle markers corresponds to the best computed result for each song using either the pitch representation or the spectral representation. The dotted line with x markers corresponds to the baseline.**

The result does not show one representation is significantly superior to the other. However, for each song, the result of one representation is often better than the result of the other. This indicates that the representation does play an important role in performance and investigating other feature representations might help improve the accuracy of the algorithm.

One can notice that, even for the song with the best performance, the computed boundaries of each section were slightly shifted from the ideal 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, etc.) can be used to detect local events so as to locate the boundaries accurately. In fact, this problem suggests that computing only the repetitive relation might not be sufficient for finding the semantic structure. "The position of phrase boundaries in tonal melodies relates to a number of interacting musical factors. The most obvious determinants of musical phrases are the standard chord progressions known as cadence. Other factors include 'surface features' such as relatively large interval leaps, change in dynamics, and micropauses ('grouping preference rules'), and repeated musical patterns in terms of harmony, rhythm and melodic contour." [3]

## 4. Music thumbnailing via structural analysis

The problem of music thumbnailing aims at finding the most representative part of a song, which can be used for music browsing and searching. It would be helpful if the song has been segmented into semantically meaningful sections before summarization, because, although what makes a part of a song most memorable is not clear, this part often appears at particular locations within the structure, e.g., the beginning or the end part of each section.

For example, among the 26 Beatles' songs, 6 songs have the song titles in the first sentence of a section; 9 songs have them in the last sentence of a section; and 10 songs have them in both the first and the last sentences of a section. For many pop/rock songs, titles are contained in the "hook" sentences. This information is very useful for music thumbnailing: once we have the structure of a song, a straightforward strategy is to choose the beginning or the end part of the most repeated section as the summary of the music. For example, if ten-second summaries are wanted, Table 1 shows the performance of this strategy based on the criteria proposed by Logan and Chu [11]. The four columns in Table 1 indicate the percentage of summaries that contain a vocal portion, contain the song's title, are the beginning of a section, and are the beginning of a phrase. The algorithm first find the most repeated sections, take the first section among these and truncate the beginning ten seconds of it as the summary of the song. The thumbnailing result using this method highly depends on the accuracy of the structural analysis result. For example, the summary of the song *From me to you* using the pitch representation is “If there's anything that you want; If there's anything I can do; Just call…”; the summary of the song *Yesterday* using the pitch representation is “Yesterday, all my troubles seemed so far away; Now it looks as though…”. Both of the summaries start right at the beginning of the most repetitive section. However, the summary of song *Day tripper* using the pitch representation does not contain any vocal portion due to the poor structural analysis result of this song.

**Table 1: Thumbnailing results of the 26 Beatles' songs**

| | Vocal | Title | Beginning of a section | Beginning of a phrase |
|---|---|---|---|---|
| Pitch | 85% | 23% | 50% | 58% |
| FFT | 88% | 35% | 46% | 58% |

## 5. Conclusions and future work

This paper presents an algorithm for automatically analyzing the repetitive structure of music from acoustic signals. Preliminary results were evaluated both qualitatively and quantitatively, which demonstrate the promise of the proposed method. To improve the accuracy, more representations need to be investigated. The possibility of generalizing this method to other music genres should also be explored.

Additionally, inferring the hierarchical repetitive structures of music and identifying the functionality of each section within the structure would be a more complicated yet interesting topic.

Music segmentation, thumbnailing and structural analysis are three coupled problems. Discovery of effective methods for solving any one of the three problems will benefit the other two. Furthermore, the solution to any of them depends on the study of humans' 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.

Finally, while most previous research on music digital libraries was based on symbolic representations of music (e.g., MIDI, scores), this paper attempts to address the structural analysis problem of acoustic musical data. We believe that current automatic music transcription techniques are still far from robust and efficient, and thus analyzing acoustic musical data directly without transcription is of great application value for indexing the digital music repository, segmenting music at transitions, and summarizing the thumbnails of music, all of which will benefit the users' browsing and searching in a music digital library.

## 6. References

[1] J.J. Aucouturier and M. Sandler. “Segmentation of Musical Signals using Hidden Markov Models,” *In Proc. AES 110$^{th}$ Convention*, May 2001.

[2] J.J. Aucouturier and M. Sandler. “Using Long-Term Structure to Retrieve Music: Representation and Matching,” *In Proc. International Symposium on Music Information Retrieval*, Bloomington, IN, 2001.

[3] M. Balaban, K. Ebcioglu, and O. Laske, Understanding Music with AI: Perspectives on Music Cognition, Cambridge: MIT Press; Menlo Park: AAAI Press, 1992.

[4] 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.

[5] W.P. Birmingham, R.B. Dannenberg, G.H. Wakefield, M. Bartsch, D. Bykowski, D. Mazzoni, C. Meek, M. Mellody, and W. Rand, “MUSART: Music Retrieval via Aural Queries,” *In Proc. International Symposium on Music Information Retrieval*, Bloomington, IN, 2001.

[6] R.B. Dannenberg and N. Hu, “Pattern Discovery Techniques for Music Audio,” *In Proc. International Conference on Music Information Retrieval*, October 2002.

[7] J. Foote, "Visualizing Music and Audio using Self-Similarity," *In Proc. ACM Multimedia Conference*, Orlando, FL, pp. 77-80, 1999.

[8] J. Foote, "Automatic Audio Segmentation using a Measure of Audio Novelty." *In Proc. of IEEE International Conference on Multimedia and Expo*, vol. I, pp. 452-455, 2000.

[9] J. Foote, “ARTHUR: Retrieving Orchestral Music by Long-Term Structure,” *In Proc. International Symposium on Music Information Retrieval*, October 2000.

[10] 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.

[11] B. Logan and S. Chu, “Music Summarization using Key Phrases,” *In Proc. International Conference on Acoustics, Speech and Signal Processing*, 2000.

[12] 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.

[13] C. Roads, The Computer Music Tutorial, MIT Press, 1996.

[14] C. Yang, "Music Database Retrieval Based on Spectral Similarity," *In Proc. International Symposium on Music Information Retrieval*, 2001.