# STRUCTURAL ANALYSIS OF MUSICAL SIGNALS VIA PATTERN MATCHING

*Wei Chai*
MIT Media Laboratory
chaiwei@media.mit.edu

## ABSTRACT

A musical piece typically has repetitive structures. Analysis of this structure can be used for music indexing, thumbnailing or segmentation. The research described here aims at automatically analyzing the repetitive structure of musical signals. First, we detect the repetition of each segment in a piece using dynamic programming. Second, we summarize this repetition information and infer the structure based on some heuristic rules. The performance of our approach is demonstrated visually using figures for qualitative evaluation, and by two structural similarity measures for quantitative evaluation. The experimental results using a corpus of Beatles' songs show that automatic structural analysis of music is possible.

## 1. INTRODUCTION

A musical piece typically has repetitive structures. 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 will be very valuable for information retrieval systems; for example, the result can be used for indexing the musical content or for music thumbnailing and segmentation.

There has been some recent research on this topic. Dannenberg and Hu presented a method to automatically detect the repetitive structure of musical signals [5]. Although promise of their method was demonstrated using several examples, there was no quantitative accuracy evaluation of their method in the paper.

Two topics closely related to structural analysis of music have also been investigated. One is music summarization (or music thumbnailing), which aims at finding the most representative part (normally assumed to be the most frequently repeated section) of a song [9]. The other related topic is music segmentation. Most of previous research in this area attempted to segment musical pieces by detecting the locations where significant change of statistical properties occurs [1].

Additionally, Foote proposed a representation called a *similarity matrix* for visualizing and analyzing the structure of audio [6][7]. Attempts using this representation for music segmentation or thumbnailing have been proposed [7][3].

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 end of each section. We assume that no prior knowledge about musical forms or the length of each section is provided, and the restatement of a section may have some variations. This requires both robustness and efficiency in our method. We also propose two novel structural similarity measures in this paper to quantitatively evaluate the performance of our algorithm, besides the visual representation of the results.

The remainder of this paper is organized as follows. Section 2 illustrates our structural analysis approach. Section 3 presents the experimental results. Section 4 gives conclusions and proposes future work.

## 2. APPROACH

This section explains the structural analysis method including five steps. All the parameter configurations mentioned in this section are based on our experimental corpus, which is described in Section 3.

### 2.1. Feature Extraction

The first step is to segment the signal into overlapped frames (e.g., 1024-sample window length with 512-sample overlap) and compute the feature of each frame.

Two representations are investigated in this paper. One is the pitch representation, which use autocorrelation [10] to estimate the main frequency component of each frame. Although all the test data in our experiment are polyphonic, it turns out that, for musical signals with a leading voice, this feature can still capture much information. The other representation we explored is the frequency representation, i.e. FFT coefficients.

We define the distance between two pitch features $v_1$ and $v_2$ as

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

We define the distance between two frequency features $\vec{v}_1$ and $\vec{v}_2$ 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.

### 2.2. Pattern Matching

After computing the feature vector $v_i$ (one-dimensional vector for the pitch representation and N-dimensional vector for the frequency representation) for each frame, we segment the feature vector sequence $S[1,n]=\{v_i \mid i=1,...,n\}$ (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 have shown that dynamic programming is very effective for music pattern matching [8][11], here we use dynamic programming to match each segment (i.e., $s_i=S[j,j+l-1]$) with the feature vector sequence starting from this segment (i.e., $S[j,n]$). The distance between feature vectors has been defined in Section 2.1. Thus, we obtain a function $d_i[k]$ corresponding to the last row of the dynamic programming matrix, meaning how well the $i^{th}$ segment matches with different locations shifted by $k$ from $j$ in the feature vector sequence. Please note that the corresponding matching part starting from *(j+k)* is not necessarily of length $l$ due to variations in the repetitive part. This step is the most time consuming one in our algorithm; its time complexity is $O(n^2)$.

### 2.3. Repetition Detection

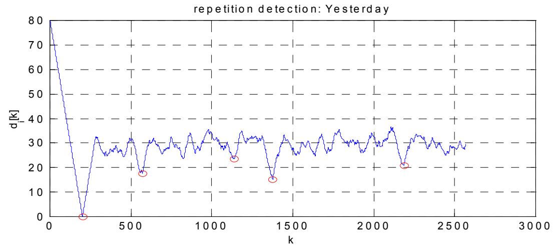

**Figure 1: One-segment repetition detection result of *Yesterday*. The local minima indicated by circles correspond to detected repetitions of the segment.**

In this step of the algorithm, we want to detect the repetition for each segment. To achieve that, we try to detect the local minima in $d_i[k]$ for each $i$, because normally a repetition of segment $i$ will correspond to a local minimum in this function. Figure 1 shows the repetition detection result of one segment in the song *Yesterday*.

The repetitions detected may have add or drop errors. For example, in Figure 1, the first, the second, the fourth and the fifth detected local minima correspond to the four restatements of the same melodic segment ("… 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 a predefined parameter *h*; whenever the local minimum is deeper than height *h*, the algorithm reports a detection of repetition. Thus, when *h* increases, we have more drop errors but less add errors, and vise versa. For balancing between these two kinds of errors, our algorithm searches within a range for the best value of *h*, so that the number of detected repetitions is reasonable (e.g., $\#\,detected\,repetitions/n\approx 2$).

### 2.4. Segment Merging

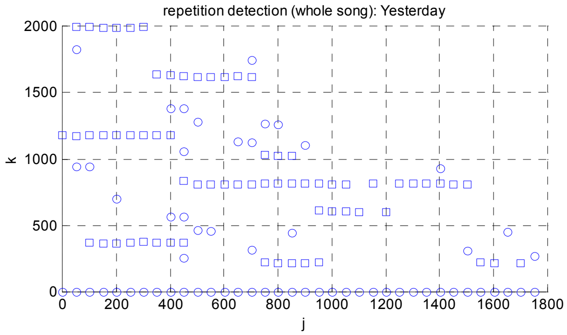

**Figure 2: Whole-song repetition detection result of *Yesterday*. A circle or a square at location (j, k) indicates that the segment starting at $v_j$ is detected to repeat at $v_{j+k}$.**

Figure 2 shows the repetition detection result of the whole song Yesterday. In this figure, each vertical pattern by fixing a particular *j* corresponds to the result as in Figure 1. Since typically one musical phrase consists of multiple segments, 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 2, where the squares have the horizontal patterns indicating consecutive segments have roughly the same shift. By detecting these horizontal patterns and discarding other detected repetitions, we reduce the effects of add/drop errors in step 2.

The output of this step is a set of merged segments 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 2 is <100, 450, 370>. Since the shifts of repetitions may not be exactly the same for segments in the merged one, we use the average of the shifts as the shift of the whole merged segment.

### 2.5. Structure Labeling

Based on the tuples obtained from the fourth step, we segment the whole piece into sections and label each section according to the repetition relation between them.

To solve conflicts that might occur, the rule for labeling is that we always label the most frequently repeated sections first. Specifically, we find the most frequently repeated segment based on the first two columns in the tuples, label it and its shifted versions as section A. We then delete those tuples we have already labeled, repeat the same procedure for the remaining tuples, and label 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), we always remain the previous labeled sections intact and truncate the current sections.

## 3. EXPERIMENT AND EVALUATION

This section presents our experimental results and the evaluation of our approach to structural analysis.

### 3.1. Data Set

Our experiment uses the 26 Beatles' songs in the two CDs The Beatles (1962-1966). We choose this corpus because all those songs have clear repetitive structures and leading voices. In our experiment, all the songs are 8-bit mono and sampled at 11kHz.

### 3.2. Measures of Structural Similarity

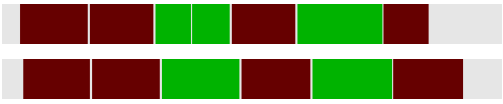

**Figure 3: Comparison of the computed structure (above) and the ideal structure (bottom) of *Yesterday*. Sections in the same color indicate restatements of the section. Sections in the lightest grey correspond to those sections with no repetition.**

To compare the structure obtained from the algorithm with the ideal structure obtained by manually labeling the repetition, we use the structural figures as shown in Figure 3. We also propose two measures here to measure the structural similarity so as 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 of different structures. For the example in Figure 3, the distance between the ideal structure AABABA and the computed structure AABBABA is 1, indicating one insertion. Here we do not care how we label each segment unless the repetition 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

$$BM = (1-r)/s \qquad (3)$$

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

### 3.3. Results

Figure 4 and Figure 5 show the structural and boundary measures of our experimental results. In Figure 5, we also plotted the baseline results corresponding to labeling the whole song as one section.

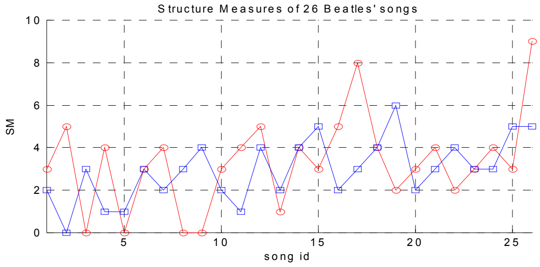

**Figure 4: Structural measures of the 26 Beatles' songs. The line with circle markers corresponds to the pitch representation results. The line with square markers corresponds to the frequency representation results.**

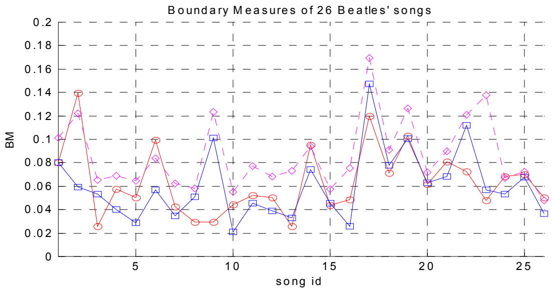

**Figure 5: Boundary measures of the 26 Beatles' songs. The line with circle markers corresponds to the pitch representation results. The line with square markers corresponds to the frequency representation results. The dashed line with diamond markers corresponds to baseline.**

From the two figures, we can see 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 6.

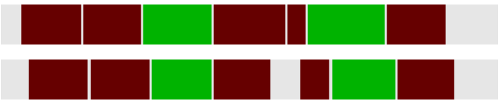

**Figure 6: Comparison of the computed structure (above) and the ideal structure (bottom) of *From me to you*.**

The one of the worst performance is the seventeenth song *Day tripper* using the pitch representation, whose result is shown in Figure 7.

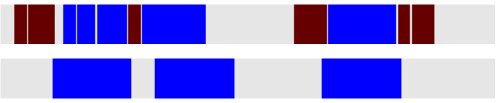

**Figure 7: Comparison of the computed structure (above) and the ideal structure (bottom) of *Day tripper*.**

Some interesting results also occur. For example, for the twelfth song *Ticket to ride*, although the computed structure using the frequency representation is different from the ideal structure as shown in Figure 8, 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.

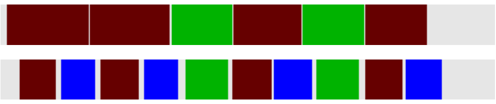

**Figure 8: Comparison of the computed structure (above) and the ideal structure (bottom) of *Ticket to ride*.**

## 4. CONCLUSIONS AND FUTURE WORK

The experimental result shows that, by either the pitch representation or the frequency representation, the performance of 15 out of 26 songs have structural measures less than or equal to 2 and the results of all the songs have boundary measures better than the baseline. This demonstrates the promise of our method. The inaccuracy in our algorithm mainly comes from the inaccuracy of pattern matching, which needs to be improved in the future.

Our result does not show one representation is significantly superior to the other. However, other feature representations should be experimented in the future. We will also explore the possibility of generalizing our method to other music genres. Besides, inferring the hierarchical repetitive structures of music would be a more complicated yet interesting topic.

Local event detection based segmentation may help improve the accuracy of the structural analysis. On the other hand, the result of the structural analysis can be combined with other musical features (e.g., chord progressions, change in dynamics, etc.) to improve the accuracy of music segmentation and music thumbnailing.

## 5. REFERENCES

[1] J.J. Aucouturier and M. Sandler. "Segmentation of Musical Signals using Hidden Markov Models," *In Proc. AES 110th 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.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.

[4] 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.

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

[6] J. Foote, "Visualizing Music and Audio using Self-Similarity," *In Proc. ACM Multimedia Conference*, 1999.

[7] 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.

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

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

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

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