ADAPTIVE STRUCTURAL ANALYSIS OF MUSIC RECORDINGS

Aggelos Pikrakis and Sergios Theodoridis

Department of Informatics
University of Piraeus
80 Karaoli and Dimitriou Str., 18534, Piraeus, Greece
phone: + (30) 2104142128, fax: + (30) 2104142264
email: pikrakis@unipi.gr
web: www.unipi.gr

Dept. of Informatics and Telecommunications
University of Athens
Panepistimioupolis, 15784, Athens, Greece
phone: + (30) 2107275328, fax: + (30) 2107275214
email: stheodor@di.uoa.gr
web: www.di.uoa.gr

ABSTRACT

This paper presents a structure mining scheme for music recordings. The term adaptive refers to the fact that the method relies on an adaptive scheme to detect similarity on the diagonals of the self-similarity matrix of the recording *and* removes the need for hard thresholds during this processing stage. Structural analysis is subsequently cast in a clustering framework. The output of the adaptive scheme is used to initialize a hierarchical data clustering algorithm whose output is a representation of the recording in terms of non-overlapping repeating patterns. The proposed method has been evaluated on a corpus of popular music recordings and various performance measures have been computed.

1. INTRODUCTION

Automated structural analysis of music recordings has attracted significant research effort over the last few years due to the need for fast browsing techniques for music recordings and efficient content-based music retrieval and indexing. In the literature, variations of the problem of structural analysis [1, 2, 5] are encountered under different names, including music summarization [7, 11, 13, 14], repeated (or repeating) pattern finding [12] and audio thumbnailing [4]. A survey of research activity reveals that the terms audio thumbnailing and summarization are used interchangeably and refer to the same task, i.e., the extraction of the most representative part of a music recording. Thumbnailing has a direct commercial interest, because it is very common among music vendors on the Internet to allow visitors to their sites to listen to the most representative extract of a music recording. Therefore, it does not come as a surprise that a lot of research effort has so far been invested on music thumbnailing.

In this paper, the term structural analysis refers to the more general problem of locating pattern repetition in raw music recordings (i.e., not MIDI), while providing the means to associate repeating patterns in such a way so as to highlight how longer patterns can be formed from smaller building blocks. From the point of view of problem definition, the proposed method falls in the same category with [1], [12], [9], [5] and [11]. A common starting point in most methods is the self-similarity matrix of the recording, which is processed so as to reveal pattern repetition. This is usually achieved by means of Dynamic Programming techniques, e.g., [1], [5]. The problem of redundancy in the resulting patterns has been tackled with various techniques, including a heuristic iterative scheme in [1] and a RP-tree in [11]. It is also interesting to note, that in most of the above methods certain assumptions (in the form of hard thresholds) need to be made at various processing stages, e.g., the allowable similarity between short-term frames. The goal of this paper is twofold:

- To remove the need for hard thresholds at various processing stages of the self-similarity matrix. To this end, an adaptive scheme is introduced that takes as input the self-similarity matrix of the recording and converts it to a binary map, where sequences of 1's on the diagonals indicate pattern repetition. This adaptive scheme determines the values of two parameters, namely the similarity threshold and the minimum pattern length, starting from some initial values. The two parameters control crucial properties of pattern repetition, i.e., the redundancy and the fraction of the audio stream that has been covered with repeating patterns.
- To cast the problem of structural analysis in a pure hierarchical data clustering framework. To this end, the binary map (i.e., the output of the previous stage), serves to initialize a hierarchical data clustering scheme, where each repeating pattern is treated as a cluster. A merging operator is introduced that determines which pair of clusters will be merged at each iteration of the clustering scheme.

The output of our method is the structure of the recording as a tree of clusters, where the highest level is the most compact representation of the music recording. It should be emphasized that the proposed method is not feature dependent. Any self-similarity matrix can be used. The paper is organized as follows: the next section describes the feature extraction stage and Section 3 presents the adaptive scheme. The hierarchical data clustering algorithm is presented in Section 4. The evaluation of the proposed method is carried out in Section 5 and conclusions are drawn in Section 6.

2. FEATURE EXTRACTION

The goal of this stage is to extract the self-similarity matrix of the recording. To this end, the chroma vector is computed on a short-term frame basis using non-overlapping frames. Although the length of the moving window is not crucial, the recommended value is 1 sec. The chroma vector is computed as in [4] except for the following modifications, which in our study have shown to increase performance:

- All DFT coefficients take part in the computation
- The mean value of the vector is not subtracted from the vector elements and each chroma element is normalized to unity by dividing it with the sum of elements of the vector.

To compute the self-similarity matrix, the Euclidean function is used as the distance metric. Therefore, low values indicate high similarity.

Let $S_{M\times M}$ be the resulting self-similarity matrix, where M is the number of short-term frames. Therefore,

$$S = \{S(i,j); S(i,j) = ||\underline{c}_i, \underline{c}_j||, i,j = 1,\dots,M\} \quad (1)$$

where $||.||$ is the Euclidean distance metric and $\underline{c}_i$ is the i-th chroma vector. By its definition, S is symmetric around the main diagonal and it thus suffices to focus on its upper triangle only.

Matrix S is then smoothed with a moving average filter. The filter that we have used is the identity matrix, i.e., $I_{K\times K}$, where K is odd. For a short-term frame length equal to 1 sec, K is set equal to 3, i.e, the equivalent of K in seconds is 3 secs. Due to the fact that the filter's mask is diagonal, the averaging operation only takes into account neighboring elements on the diagonal.

3. ADAPTIVE BINARIZATION

The self-similarity matrix, S, is fed as input to an adaptive binarization algorithm, which is subject to two parameters, namely, the similarity threshold an the minimum pattern length. The algorithm determines the values of these two parameters, starting from some initial conditions and outputs a binary map. The key idea behind this algorithm is that an element of the resulting binary map is equal to 1, if:

- *Condition (a)*: Its value in the self-similarity matrix is "sufficiently" low.
- *Condition (b)*: Its value in the self-similarity matrix is lower than the values of its neighboring elements.
- *Condition (c)*: It is part of a "sufficiently" long sequence of 1 s on a diagonal of the binary map.

The proposed algorithm is iterative. At each iteration, the value of a cost function is computed. The algorithm terminates when the cost ceases to increase.

3.1 Cost function

In order to proceed, we first describe the adopted cost function. Let B_k, T_k and L_k be the binary map, the similarity threshold and the minimum pattern length respectively, at the k-th iteration. Clearly, the size of B_k is $M\times M$, where M is the number of feature vectors. The first two conditions stated above suggest that:

$$B_k(i,j) = \begin{cases} 1, & i \le j,\ S(i,j) \le T_k \quad and \\ & S(i,j) < S(i-1,j) \quad and \\ & S(i,j) < S(i+1,j) \quad and \\ & S(i,j) < S(i,j-1) \\ 0, & otherwise \end{cases} \quad (2)$$

Equivalently, $S(i,j)$ has to be lower than its northern, southern and eastern neighbors.

Matrix B_k is then"cleaned" by removing isolated 1's. This can be easily achieved by correlating B_k with the following rectangular mask, $BM_{3\times3}$:

$$BM = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (3)$$

As a result, isolated 1's will yield zero correlation.

Each diagonal of B_k is then processed separately in order to detect adjacent 1's, i.e., diagonal segments solely consisting of 1's. We only keep segments whose length exceeds L_k (condition (c)). Any detected diagonal segment relates a pair of corresponding patterns, where each pattern is a sequence of adjacent frame indices. Therefore, if the i-th detected diagonal segment starts at (P_{i_1}, P_{i_2}) and is l_i pixels long (i.e., it consists of l_i 1's), then the two corresponding patterns are:

$$\{P_{i_1}, P_{i_1}+1, \dots, P_{i_1}+l_i-1\}$$

and

$$\{P_{i_2}, P_{i_2}+1, \dots, P_{i_2}+l_i-1\}$$

i.e., P_{i_1} and P_{i_2} are the frame indices that mark the beginning of each pattern. In addition, let PU_i be the union of the above two frame sets, i.e.,

$$PU_i = \{P_{i_1}, P_{i_1}+1, \dots, P_{i_1}+l_i-1, P_{i_2}, P_{i_2}+1, \dots, P_{i_2}+l_i-1\}$$

It is important to note that, in most cases, the detected segments exhibit certain redundancy, i.e., the respective PU_i's overlap (see Fig. 1). This is not desirable and our next goal

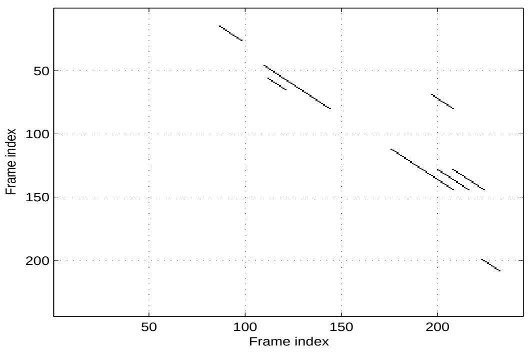

Figure 1: Binary map at the output of the adaptive scheme. Upon initialization $T_0 = 0.01$ and $L_0 = 10$ frames (seconds).

is to quantify redundancy. To this end, let P_D be the union of all PU_i 's, i.e.,

$$P_D = \bigcup_{i=1}^{D} PU_i,$$

where D is the number of detected segments. By the definition of union of sets, each frame index appears only once in P_D, even if it is a member of more than one PU_i 's. If V_{P_D} is the cardinality of P_D, then the fraction C_{P_D}, of short-term frames that have been "covered" by all detected patterns is defined as

$$C_{P_D} = \frac{V_{P_D}}{M}, \quad (4)$$

where M is the total number of frames. *Redundancy*, R_{P_D}, is then defined as

$$R_{P_D} = 1 - \frac{V_{P_D}}{\sum_{i=1}^{D} l_i}, \quad (5)$$

where $l_i, i = 1,\dots,D$ is the length of the i-th diagonal segment. If the PU_i's do not overlap at all, then R_{P_D} is zero.

The proposed *cost function*, $Cf(.)$, is then defined as :

$$Cf(P_D) = C_{P_D}(C_{P_D} - C_{P_D}R_{P_D}) \quad (6)$$

If C_{P_D} is close to 1 (the detected patterns cover most of the recording) and R_{P_D} is close to zero (low redundancy), then $Cf(P_D)$ will be close to 1. In the extreme case of zero redundancy, the cost function only depends on C_{P_D}. If, on the other hand, both C_{P_D} and R_{P_D} are large, the expression in parenthesis will be close to zero. This indicates that although a large fraction of the recording has been covered with patterns, this has been achieved to the expense of high pattern redundancy. Equation (6) suggests that the higher the value of the cost function the better the "quality" of the binary map.

Having introduced the cost function the adaptive scheme can now be presented as two separate iteration schemes; the first one determines the similarity threshold and the second one the minimum segment length.

3.2 First scheme

1. *Initialization*: Choose initial values, T_0 and L_0, for the similarity threshold and the minimum pattern length respectively. Generate the binary map and compute $Cf(P_D)$.
2. *Iteration*: Increase the similarity threshold by a given step size, i.e, $T_k = T_{k-1} + \Delta T$. Generate the new binary map and compute the new $Cf(P_D)$. If $Cf(P_D)$ has increased, repeat the iteration, otherwise terminate and keep the similarity threshold that has yielded the maximum value of $Cf(P_D)$.

Note that this iterative procedure does not affect the minimum segment length.

3.3 Second scheme

1. *Initialization*: Choose as the initial value for the similarity threshold the value that has resulted from the previous scheme and use the same L_0 as before (note that minimum pattern length remained unaltered during the previous scheme). Generate the binary map and compute $Cf(P_D)$.
2. *Iteration*: Decrease the minimum segment length by a given step size, i.e., $L_k = L_{k-1} + \Delta L$. Generate the new binary map and compute $Cf(P_D)$. If $Cf(P_D)$ has increased, repeat the iteration, otherwise terminate and keep the segment length that has yielded the maximum value of $Cf(P_D)$

This adaptation of the second scheme is subject to the minimum segment length only. In Section 5, we show how the initial conditions affect the performance of the adaptive scheme and also discuss issues related to the number of iterations that are needed to achieve convergence.

In the end of this iterative scheme, the values for similarity threshold and minimum segment length have been decided. The resulting binary matrix is then fed as input to a hierarchical clustering scheme that yields the structure of the music recording in terms of non-overlapping, repeated patterns. Figure 1 presents the binary map at the output of the adaptive scheme for the music track "Animal" by "Def Leppard".

4. HIERARCHICAL DATA CLUSTERING

The adaptive scheme converges to a solution that can be interpreted as a trade-off between pattern redundancy and the fraction of the music recording that has been covered with repeating patterns. Our next goal is to process the binary matrix B at the output of the adaptive scheme, so as to produce non-overlapping repeating patterns, i.e., remove redundancy and produce the structure of the recording in terms of repeating patterns. To this end, we have chosen a hierarchical clustering scheme.

In our problem definition, *a cluster is a set of similar patterns, all of which have the same length*. It is therefore necessary to define how clusters are initialized, when two clusters should be merged and what the outcome of such a merging action will be.

4.1 Initialization

At a first step, matrix B is scanned row-wise. By the definition of B, if the j-th element on the i-th row is equal to 1, then the i-th frame can be considered to be similar to the j-th frame. To this end, let

$$Cr_i = \{i\} \cup \{j; B(i,j) = 1, j = 1,\dots,M, j \neq i\},$$

where $i = 1,\dots,M$. In other words, Cr_i is the set of frames which are similar to the i-th frame following the inspection of the i-th row.

Matrix B is then scanned column-wise. Similarly, let

$$Cc_i = \{i\} \cup \{j; B(j,i) = 1, j = 1,\dots,M, j \neq i\},$$

where $i = 1,\dots,M$. Equivalently, Cc_i contains the indices of frames which are similar with the i-th frame, following the inspection of the i-th column. In addition, let

$$C_i = Cr_i \cup Cc_i,\ i = 1,\dots,M$$

Obviously, C_i is the set of all frames which are similar with the i-th frame (including the frame itself). By the definition of union of sets no repetitions are encountered in C_i.

The C_i's are then examined pairwisely. If $C_i \cap C_k \neq \emptyset$ then C_i is replaced by $C_i \cup C_k$, C_k is disgarded and this is repeated until no more merging is possible. Let K be the number of sets that have survived when this procedure has finished. Each $C_i, i = 1,\dots,K$ is considered to be an *initial cluster*, which contains at least two frame indices.

For notational purposes, let $N_i, i = 1,\dots,K,$ be the number of frames in the i-th cluster. Each frame is considered to be an *elementary* pattern. Therefore, the i-th cluster containes N_i patterns, all of which have the same length, equal to one frame. For generalisation purposes, in the sequel we will refer to this pattern length with the symbol $L_i, i = 1,\dots,K$. In addition, due to the fact that, in the general case, each pattern is a sequence of adjacent frames, it is convenient to represent the pattern by means of the pair of indices that mark its endpoints. Therefore, the i-th cluster can be written as a set of pairs of indices, i.e.,

$$C_i = \{(i_1,i_2),(i_3,i_4),\dots,(i_{2N_i-1},i_{2N_i}),\} \qquad (7)$$

where $i_1 \leq i_2 \leq \dots \leq i_{2N_i-1} \leq i_{2N_i}$. Due to the equalities, this definition accounts for the possibility that a pattern consists of a single frame, as is the case at the output of the initialization stage,

4.2 Iteration - Merging clusters

Having adopted a hierarchical clustering scheme, at each iteration two clusters are chosen and merged based on a merging criterion. The pair of clusters to be merged is determined by computing a merging cost for all possible pairs of clusters. This process is repeated until no more merging can take place. For convenience of presentation, the merging cost along with the merging operator are first explained by means of a real example drawn from the music track "People are People" of "Depeche Mode". For this track, at some point of processing, the following two clusters have been formed:

$C_5 = \{(3,4),(19,20)\}$ and $C_6 = \{(5,5),(21,21)\}$, i.e., $N_5 = N_6 = 2$, $L_5 = 2$ and $L_6 = 1$. It can be observed that pattern $(3,4)$ can be expanded by attaching pattern $(5,5)$ to its right end and similarly, pattern $(19,20)$ can be expanded by attaching $(21,21)$ to its right end. In other words, the two clusters can be merged to form cluster $\{(3,5),(19,21)\}$. In this example, the merging cost equals zero because no patterns have been left out of the merging operation. Furthermore, it can be seen that that a pattern from one cluster can be merged with a pattern from another cluster, if the latter can be attached to the former's right or left end.

To continue the example, if

$$C_6 = \{(5,5),(21,21),(37,37)\}$$

then the merging cost would be equal to one frame, because pattern $(37,37)$ will have to be left out. The result of the merging operation would be the same as before, i.e., the pattern that was omitted is disgarded. If it is not disgarded, it should make a cluster of its own, but this would not make sense, because a cluster has to contain at least two patterns.

As a third case, consider

$$C_5 = \{(3,4),(19,20),(42,43),(100,101))\}$$

and

$$C_6 = \{(5,5),(21,21)\}$$

The merging cost is now equal to 4 frames, because the last two patterns from the first cluster cannot take part in the merging operation (but can make a cluster of their own). The merging result will be two clusters, i.e., $\{(3,5),(19,21)\}$ and $\{(42,43),(100,101)\}$.

In the general case, two clusters, say A and B, can be merged, if $L_A \geq L_B$, i.e., the number of patterns in A is larger or equal than the number of patterns in B and one of the following holds (*mc* stands for the merging cost and *ws* for the short-term processing step measured in seconds):

1. If $N_A = N_B$ and all patterns in B can be attached to the right end of an equal number of patterns in A (or all patterns in B can be attached to the left end of an equal number of patterns in A), then a single cluster is formed. The cluster contains the expanded patterns of A, B vanishes and $mc = 0$;
2. If $N_A - N_B = 1$, $ws \times L_A \leq 2secs$ and all patterns in B can be attached to the right end of an equal number of patterns in A (or all patterns in B can be attached to the left end of an equal number of patterns in A) then a single cluster is formed. The cluster contains the expanded patterns of A, B vanishes and also the pattern that has been left out from A vanishes. In this case, $mc = ws \times L_A$.
3. If $N_A - N_B \geq 2$ and all patterns in B can be attached to the right end of an equal number of patterns in A (or all patterns in B can be attached to the left end of an equal number of patterns in A) then two clusters are formed. The first cluster contains the expanded patterns of A, B vanishes and the second cluster contains all the patterns that have been left out from A. This is because the remaining patterns are more than one and it still makes sense to let them form a cluster. To compute the merging cost in this case, we compute the cumulative length of patterns in the second cluster and multiply it by the moving window length.

Following the above, at each step the pair of clusters that yields the lowest merging cost is selected as the winner and the two clusters are merged. If more than one pairs of clusters yield the same merging cost, the winner is selected randomly. The clustering is repeated until no more merging is possible. Note that the merging cost for a pair of clusters can be computed with $O(N_A + N_B)$ complexity, if the patterns of the two clusters are sorted in ascending order according to their left endpoints.

5. EXPERIMENTS

The proposed method has been evaluated on a corpus of popular music recordings consisting of 104 music tracks. The track listing can be accessed at http://www.unipi.gr/faculty/pikrakis/cor.html. The corpus consists of pop music tracks including mainstream pop, alternative pop and a limited number of hard-rock tracks. The tracks were chosen based on the fact that repetition is more evident in this type of music, making it easier to evaluate the structure returned by our method per track. We first present issues related to the initialization of the adaptive scheme and then proceed by defining a number of performance measures related to the complexity of the structure returned by our method. In the end, we discuss how close the results are to the human perception of structure.

5.1 Initialization of the adaptive scheme

Given that the Euclidean distance is used as the distance metric, the adaptive scheme is initialized with a threshold value (T_0) close to zero. After experimentation, we have found that any value in the range $0.006 - 0.01$ is a good candidate. If a smaller value is chosen, then the number of elements in B that survive binarization is too limited. The recommended value for the threshold step, ΔT, is 0.001. Concerning the initialization of the minimum segment length (L_0), a value of 10 seconds is recommended, with $\Delta L = 1$ sec. We have observed that small values of L_0 (around 5) result into oversegmentation in most cases (i.e., large number of clusters where pattern length is close to one). In the vast majority of music tracks, the scheme converges in less than 10 iterations.

5.2 Performance statistics related to the hierarchical clustering scheme

Following our study, the average number of clusters at the initialization stage of the clustering scheme is ≈ 38. In 80% of the clusters, the pattern length is shorter than 5 secs and in 95% of the clusters it is shorter then 10 secs. This reveals that one expects a large number of clusters at the initialization stage and due to the fact that the clusters do not overlap, the average pattern length is inevitably small. Note that in 40% of the music tracks the number of detected segments on the diagonals is less than 10 and in 70% of the tracks is less than 20.

Furthermore, the average number of clusters at the end of the hierarchical clustering scheme is 9. In 4 tracks out of 104 a single cluster was formed in the end, containing instances of the thumbnail of the track. In 70% of the tracks the number of clusters was less than 10. The average pattern length at the end of the clustering stage was found to be ≈ 16 secs and in 60% of the clusters it was larger than 10 secs.

Figure 2 presents in piano-roll format the repeating patterns at the end of the clustering scheme for the track “Animal” of “Def Leppard”. If a capital letter is assigned to each cluster, then the track can be written as *B_EA_B_EA_ACDC_CD*, where _ stands for a gap, i.e., a part of audio that has not been assigned to any pattern. For this particular recording, the music recall is ≈ 62%. On average, ≈ 65% of a music track is covered by the extracted repeating papers and for the majority of tracks (≈ 80%), the patterns in the largest cluster practically coincide with instances of the chorus, given certain tolerance is acceptable around the chorus endpoints.

Details on the relevance of the extracted clusters with each track’s thumbnail can be accessed at http://www.unipi.gr/faculty/pikrakis/cor.html.

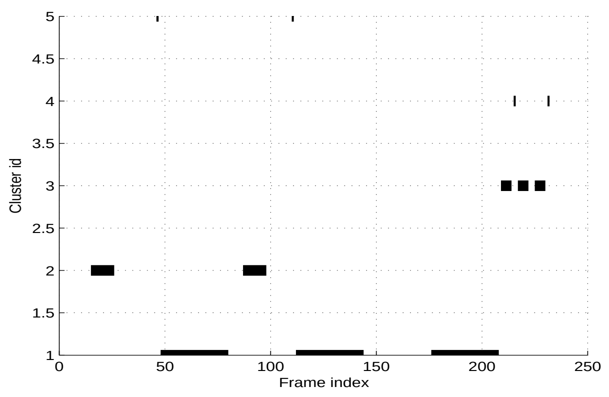

Figure 2: The track “Animal” of “Def Leppard” in piano-roll format. Five clusters have been formed, the 4-th and 5-th of which contain two small patterns. The largest cluster contains three repetitions of the song’s chorus.

6. CONCLUSIONS

This paper has presented a structure mining scheme for music recordings. An adaptive scheme for detecting similarity on the diagonals of the self-similarity matrix is first employed and it is followed by a clustering scheme that produces a hierarchy of non-overlapping clusters, i.e., repeating patterns. The proposed method removes the need for hard thresholds during the processing stages of the self-similarity matrix and in addition formulates the problem of structure analysis in the context of a hierarchical data clustering scheme. As a result, the proposed solution can be treated as a general, i.e., a method that can be applied on any self-similarity matrix, irrespective of the short-term features being used, as long as the distance is bounded.

REFERENCES

[1] R. B. Dannenberg and Ning Hu, “Discovering Musical Structure in Audio Recordings, ” in *Proc. Second International Conference on Music and Artifical Intelligence (ICMAI 02)*, Edinburgh, Scotland, 2002, pp. 43–57.

[2] M. Sandler and M. Levy, “Structural Segmentation of Musical Audio by Constrained Clustering”, *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16(2), pp. 318–326, 2008

[3] M.A. Bartsch and G.H. Wakefield, “To catch a chorus: Using chroma-based representations for audio thumbnailing ” in *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2001, pp. 15–18.

[4] M.A. Bartsch and G.H. Wakefield, “Audio Thumbnailing of Popular Music Using Chroma-Based Representations”, *IEEE Transactions on Multimedia*, vol. 7(1), pp. 96–104, 2005.

[5] W. Chai, “Structural analysis of musical signals via pattern matching ” in *Proc. of ICASSP -03*, 2003, vol. 5, pp. 549–552.

[6] W.Chai and B. Vercoe, “Music thumbnailing via structural analysis ” in *Proc. of the 11th ACM International Conference on Multimedia*, 2003, pp. 223–226.

[7] M. Cooper and J. Foote, “Visualizing music and audio using self-similarity ” in *Proc. of ACM Multimedia*, 1999, pp. 77–80.

[8] Geoffroy Peeters, “Toward automatic music audio summary generation from signal analysis, ” in *Proc. International Conference on Music Information Retrieval*, 2002, pp. 94–100.

[9] C. Xu, Y. Zu and Qi Tian, “Automatic Music Summarization based on temporal, spectral and cepstral features, ” in *Proc. IEEE ICME’02*, 2002, pp. 117–120.

[10] J. Wellhausen and M. Hoynck, “Audio thumbnailing using MPEG-7 low level audio descriptors, ” in *Internet Multimedia Management Systems IV*, 2003, pp. 65–73.

[11] C. Liu P. and Yao, “Automatic summarization of MP3 music objects ” in *Proc. of ICASSP’04*, 2004, vol. 5, pp. 921–924.

[12] M. Wang, L. Lu and H. Zhang, “Repeating pattern discovery from acoustic musical signals ” in *Proc. of the 6th ACM SIGMM International workshop on Multimedia Information Retrieval*, 2004, vol. 3, pp. 2019–2022.

[13] C. Xu, X. Shao, N.C. Maddage, M.S. Kankanhalli and Q. Tian, “Automatically summarize musical audio using adaptive clustering ” in *Proc. of IEEE ICME’04*, 2004, vol. 3, pp. 2063–2066.

[14] S. Kim, S. Kwon and H. Kim, “A music summarization scheme using tempo tracking and two stage clustering ” in *Proc. of the 8th IEEE Workshop on Multimedia Signal Processing*, 2006, pp. 225–228.