Music Structure Analysis from Acoustic Signals

Roger B. Dannenberg and Masataka Goto

Abstract

Music is full of structure, including sections, sequences of distinct musical textures, and the repetition of phrases or entire sections. The analysis of music audio relies upon feature vectors that convey information about music texture or pitch content. Texture generally refers to the average spectral shape and statistical fluctuation, often reflecting the set of sounding instruments, e.g. strings, vocal, or drums. Pitch content reflects melody and harmony, which is often independent of texture. Structure is found in several ways. Segment boundaries can be detected by observing marked changes in locally averaged texture. Similar sections of music can be detected by clustering segments with similar average textures. The repetition of a sequence of music often marks a logical segment. Repeated phrases and hierarchical structures can be discovered by finding similar sequences of feature vectors within a piece of music. Structure analysis can be used to construct music summaries and to assist music browsing.

Introduction

Probably everyone would agree that music has structure, but most of the interesting musical information that we perceive lies hidden below the complex surface of the audio signal. From this signal, human listeners perceive vocal and instrumental lines, orchestration, rhythm, harmony, bass lines, and other features. Unfortunately, music audio signals have resisted our attempts to extract this kind of information. Researchers are making progress, but so far, computers have not come near to human levels of performance in detecting notes, processing rhythms, or identifying instruments in a typical (polyphonic) music audio texture.

On a longer time scale, listeners can hear structure including the chorus and verse in songs, sections in other types of music, repetition, and other patterns. One might think that without the reliable detection and identification of short-term features such as notes and their sources, that it would be impossible to deduce any information whatsoever about even higher levels of abstraction. Surprisingly, it *is* possible to automatically detect a great deal of information concerning music structure. For example, it is possible to label the structure of a song as AABA, meaning that opening material (the "A" part) is repeated once, then contrasting material (the "B" part) is played, and then the opening material is played again at the end. This structural description may be deduced from low-level audio signals. Consequently, a computer might locate the "chorus" of a song without having any representation of the melody or rhythm that characterizes the chorus.

Underlying almost all work in this area is the concept that structure is induced by the repetition of similar material. This is in contrast to, say, speech recognition, where there is a common understanding of words, their structure, and their meaning. A string of unique words can be understood using prior knowledge of the language. Music, however, has no language or dictionary (although there are certainly known forms and conventions). In general, structure can only arise in music through repetition or systematic transformations of some kind.

Repetition implies there is some notion of similarity. Similarity can exist between two points in time (or at least two very short time intervals), similarity can exist between two sequences over longer time intervals, and similarity can exist between the longer-term statistical behaviors of acoustical features. Different approaches to similarity will be described.

Similarity can be used to segment music: contiguous regions of similar music can be grouped together into segments. Segments can then be grouped into clusters. The segmentation of a musical work and the grouping of these segments into clusters is a form of analysis or "explanation" of the music.

Features and Similarity Measures

A variety of approaches are used to measure similarity, but it should be clear that a direct comparison of the waveform data or individual samples will not be useful. Large differences in waveforms can be imperceptible, so we need to derive features of waveform data that are more perceptually meaningful and compare these features with an appropriate measure of similarity.

Feature Vectors for Spectrum, Texture, and Pitch

Different features emphasize different aspects of the music. For example, mel-frequency cepstral coefficients (MFCCs) seem to work well when the general shape of the spectrum but not necessarily pitch information is important. MFCCs generally capture overall "texture" or timbral information (what instruments are playing in what general pitch range), but some pitch information is captured, and results depend upon the number of coefficients used as well as the underlying musical signal.

When pitch is important, e.g. when searching for similar harmonic sequences, the chromagram is effective. The chromagram is based on the idea that tones separated by octaves have the same perceived value of *chroma* (Shepard 1964). Just as we can describe the *chroma* aspect of pitch, the short term frequency spectrum can be restructured into the *chroma spectrum* by combining energy at different octaves into just one octave. The *chroma vector* is a discretized version of the *chroma spectrum* where energy is summed into 12 log-spaced divisions of the octave corresponding to pitch classes (C, C#, D, … B). By analogy to the spectrogram, the *discrete chromagram* is a sequence of chroma vectors.

It should be noted that there are several variations of the chromagram. The computation typically begins with a short-term Fourier transform (STFT) which is used to compute the magnitude spectrum. There are different ways to "project" this onto the 12-element chroma vector. Each STFT bin can be mapped directly to the most appropriate chroma vector element (Bartsch and Wakefield 2001), or the STFT bin data can be interpolated or windowed to divide the bin value among two neighboring vector elements (Goto 2003a). Log magnitude values can be used to emphasize the presence of low-energy harmonics. Values can also be averaged, summed, or the vector can be computed to conserve the total energy. The chromagram can also be computed by using the Wavelet transform.

Regardless of the exact details, the primary attraction of the chroma vector is that, by ignoring octaves, the vector is relatively insensitive to overall spectral energy distribution and thus to timbral variations. However, since fundamental frequencies and lower harmonics of tones feature prominently in the calculation of the chroma vector, it is quite sensitive to pitch class content, making it ideal for the detection of similar harmonic sequences in music.

While MFCCs and chroma vectors can be calculated from a single short term Fourier transform, features can also be obtained from longer sequences of spectral frames. Tzanetakis and Cook (1999) use means and variances of a variety of features in a one second window. The features include the spectral centroid, spectral rolloff, spectral flux, and RMS energy.

Peeters, La Burthe, and Rodet (2002) describe "dynamic" features, which model the variation of the short term spectrum over windows of about one second. In this approach, the audio signal is passed through a bank of Mel filters. The time-varying magnitudes of these filter outputs are each analyzed by a short term Fourier transform. The resulting set of features, the Fourier coefficients from each Mel filter output, is large, so a supervised learning scheme is used to find features that maximize the mutual information between feature values and hand-labeled music structures.

Measures of Similarity

Given a feature vector such as the MFCC or chroma vector, some measure of similarity is needed. One possibility is to compute the (dis)similarity using the Euclidean distance between feature vectors. Euclidean distance will be dependent upon feature magnitude, which is often a measure of the overall

music signal energy. To avoid giving more weight to the louder moments of music, feature vectors can be normalized, for example, to a mean of zero and a standard deviation of one or to a maximum element of one.

Alternatively, similarity can be measured using the scalar (dot) product of the feature vectors. This measure will be larger when feature vectors have a similar direction. As with Euclidean distance, the scalar product will also vary as a function of the overall magnitude of the feature vectors. If the dot product is normalized by the feature vector magnitudes, the result is equal to the cosine of the angle between the vectors. If the feature vectors are first normalized to have a mean of zero, the cosine angle is equivalent to the correlation, another measure that has been used with success.

Lu, Wang, and Zhang (Lu, Wang, and Zhang 2004) use a constant-Q transform (CQT), and found that CQT outperforms chroma and MFCC features using a cosine distance measure. They also introduce a "structure-based" distance measure that takes into account the harmonic structure of spectra to emphasize pitch similarity over timbral similarity, resulting in additional improvement in a music structure analysis task.

Similarity can be calculated between individual feature vectors, as suggested above, but similarity can also be computed over a window of feature vectors. The measure suggested by Foote (1999) is vector correlation:

$$S_w(i,j) = \frac{1}{w}\sum_{k=0}^{w-1}(V_{i+k} \bullet V_{j+k}) \quad (1)$$

where w is the window size. This measure is appropriate when feature vectors vary with time, forming significant temporal patterns. In some of the work that will be described below, the detection of temporal patterns is viewed as a processing step that takes place after the determination of similarity.

Evaluation of Features and Similarity Measures

Linear prediction coefficients (LPC) offer another low-dimensional approximation to spectral shape, and other encodings such as moments (centroid, standard deviation, skewness, etc.) are possible. Aucouturier and Sandler (2001) compare various approaches and representations. Their ultimate goal is to segment music according to texture, which they define as the combination of instruments that are playing together. This requires sensitivity to the general spectral shape, and insensitivity to the spectral details that vary according to pitch. They conclude that a vector of about 10 MFCCs is superior to LPC and discrete cepstrum coefficients (Galas and Rodet 1990).

On the other hand, Hu, Dannenberg, and Tzanetakis (2003) compare features for detecting similarity between acoustic and synthesized realizations of a single work of music. In this case, the goal is to ignore timbral differences between acoustic and synthetic instruments, but to achieve fine discrimination of pitches and harmonies. They conclude that the chroma vector is superior to pitch histograms and MFCCs.

Segmentation

One approach to discovering structure in music is to locate segments of similar musical material and the boundaries between them. Segmentation does not rely on classification or the discovery of higher order structure in music. However, one can envision using segmentation as a starting point for a number of more complicated tasks, including music summarization, music analysis, music search, and genre classification. Segmentation can also assist in audio browsing, a task that can be enhanced through some sort of visual summary of music and audio segments.

Segmentation Using Texture Change

Tzanetakis and Cook (1999) perform segmentation as follows: Feature vectors V_i are computed as described above. A feature time differential, Δ_i, is defined as the Mahalanobis distance:

$$\Delta_i = (V_i - V_{i-1})^T \Sigma^{-1} (V_i - V_{i-1}) \quad (2)$$

where Σ is an estimate of the feature covariance matrix, calculated from the training data, and i is the frame number (time). This measure is related to the Euclidean distance but takes into account the variance and correlations among features. Next, the first order differences of the distance, $\Delta_i - \Delta_{i-1}$, are computed. A large difference indicates a sudden transition. Peaks are picked, beginning with the maximum. After a peak is selected, the peak and its neighborhood are zeroed to avoid picking another peak within the same neighborhood. Assuming the total number of segments is given *a priori*, the neighborhood is 20% of the average segment size. Additional peaks are selected and zeroed until the desired number of peaks (segment boundaries) has been obtained.

Segmentation by Clustering

Logan and Chu (2000) describe a clustering technique for discovering music structure. The goal is to label each frame of audio so that frames within similar sections of music will have the same labels. For example, all frames within all occurrences of the chorus should have the same label. This can be accomplished using bottom-up clustering to merge clusters that are similar. Initially, the feature vectors are divided into fixed-length contiguous segments and each segment receives a different label. The following clustering step is iterated:

> Calculate the mean μ and covariance Σ of the feature vectors within each cluster. Compute a modified Kullback Leibler (KL) distance between each pair of clusters, as described below. Find the pair of clusters with the minimum *KL2* distance, and if this distance is below a threshold, combine the clusters. Repeat this step until no distance is below the threshold.

The *KL2* distance between two Gaussian distributions A and B is given by:

$$KL2(A,B) = KL(A;B) + KL(B;A) \quad (3)$$

$$= \frac{\Sigma_A}{\Sigma_B} + \frac{\Sigma_B}{\Sigma_A} + (\mu_A - \mu_B) \cdot \left(\frac{1}{\Sigma_A} + \frac{1}{\Sigma_B} \right) \quad (4)$$

Segmentation and Hidden Markov Models

Another approach to segmentation uses a hidden Markov model (HMM). In this approach, segments of music correspond to discrete states Q and segment transitions correspond to state changes. Time advances in discrete steps corresponding to feature vectors, and transitions from one state to the next are modeled by a probability distribution that depends only on the current state. This forms a Markov model that generates a sequence of states. Note that states are "hidden" because only feature vectors are observable. Another probability distribution, $p(V_i | q_i)$, models the generation of feature vector V_i from state q_i. The left side of Figure 1 illustrates a 4-state ergodic Markov model, where arrows represent state transition probabilities. The right side of the figure illustrates the observation generation process, where arrows denote conditional probabilities between variables.

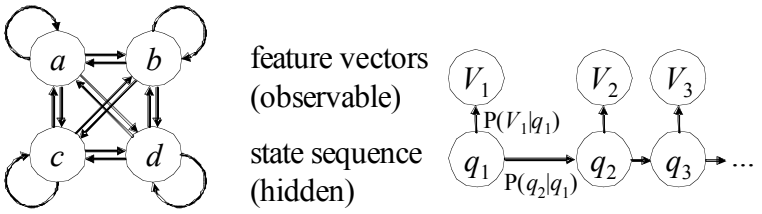

Figure 1. Hidden Markov model with four hidden states *a, b, c,* and *d*. As shown, feature vectors depend only upon the current state, which depends only upon the previous state.

The HMM has advantages for segmentation. In general, feature vectors do not indicate the current state (segment class) unambiguously, so when a single feature vector is observed, one cannot assume that it was generated by particular state. However, some features are more likely to occur in one state than another, so one can observe the *trend* of feature vectors, ignoring the unlikely outliers and guessing the state that is most consistent with the observations. If transitions are very unlikely, one may have to assume many outliers occur. On the other hand, if transitions are common and segments are short, one can change states rapidly to account for different feature vectors. The HMM formalism can determine the segmentation (the hidden state sequence) with the maximum likelihood given a set of transition probabilities and observations, thus the model can formalize the tradeoffs between minimizing transitions and matching features to states. Furthermore, HMM transition probabilities can be estimated from unlabeled training data, eliminating the need to guess transition probabilities manually.

Aucouturier and Sandler (2001) model the observation probability distribution $P(V_i|q_j)$ as a mixture of Gaussian distributions over the feature space:

$$P(V_i \mid q_j) = \sum_{m=1}^{M} c_{j,m} \cdot \mathbf{N}(V_i, \mu_{j,m}, \Gamma_{j,m}) \tag{5}$$

where **N** is a Gaussian probability density function with mean $\mu_{i,m}$, covariance matrix $\Gamma_{j,m}$, and $c_{j,m}$ is a mixture coefficient. Here, *i* indexes time and *j* indexes state. They train the HMM using the Baum-Welch algorithm using the sequence vectors from the single song chosen for analysis. The Viterbi algorithm is then used to find the sequence of hidden states with the maximum likelihood, given the observed feature vectors.

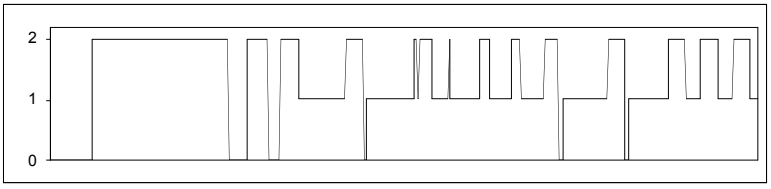

Figure 2. Segmentation of 20 seconds of a song. State 0 is silence, State 1 is voice, accordion, and accompaniment, and State 2 is accordion and accompaniment.

One potential drawback of this approach is that the HMM will segment the signal according to fine-grain changes in spectral content rather than long-term elements of musical form. For example, in one of Aucoturier and Sandler's test cases (see Figure 2), the HMM segmentation appears to isolate individual words of a singer rather than divide the song according to verses and instrumental interludes. (Aucouturier, Pachet, and Sandler 2005) In other words, the segments can be quite short when there are rapid changes in the music. Although this might be the desired result, it seems likely that one could detect longer-term, higher-level music structure by averaging features over a longer time span or applying further processing to the state sequence obtained from an HMM.

Peeters, La Burthe, and Rodet (2002) approach the problem of clustering with a two-pass algorithm. Imagine a human listener hearing a piece of music for the first time. The range of variation of music features becomes apparent, and templates or classes of music are formed. In the second hearing, the structure of the music can be identified in terms of the previously identified templates.

An automated system is inspired by this two-pass model. In the first pass, texture change indicates segment boundaries, and "potential" states are formed from the mean values of feature vectors within segments. In the second pass, potential states that are highly similar are merged by using the *K*-means algorithm. The resulting *K* states are called the "middle" states. Because they represent clusters with no regard for temporal contiguity, a hidden Markov model initialized with these "middle" states is then used to inhibit rapid inappropriate state transitions by penalizing them. The Baum-Welch algorithm is used to train the model on the sequence of feature vectors from the song. Viterbi decoding is used to obtain a state sequence. Figure 3 shows the result of an analysis using this smoothing technique.

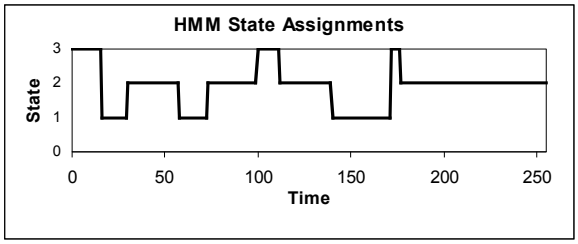

Figure 3. Classification of states in "Head Over Feet" from artist Alanis Morisette. (Adapted from Peeters, La Burthe, and Rodet, 2002).

The Similarity Matrix

A concept used by many researchers is the similarity matrix. Given a sequence of feature vectors V_i and a measure of similarity $S(i, j)$, one can simply view $S(i, j)$ as a matrix. The matrix can be visualized using a grayscale image where black represents dissimilar vectors and white represents similar vectors. Shades of gray represent intermediate values. Since any vector is similar to itself, the diagonal of the similarity matrix will be white. Also, assuming the similarity measure is symmetric, the matrix will be symmetric about the diagonal. The interesting information in the matrix is in the patterns formed off the diagonal.

In very general terms, there are two interesting sorts of patterns that appear in the similarity matrix, depending on the nature of the features. The first of these appears when features correspond to relatively long-term textures. The second appears when features correspond to detailed short-term features such as pitch or harmony and where similar sequences of features can be observed. These two types of patterns are considered in the next two sections.

Texture Patterns

First, consider the case where features represent the general texture of the music, for example whether the music is primarily vocal, drum solo, or guitar solo. Figure 4 shows an idealized similarity matrix for this case. The white diagonal appears because feature vectors along the diagonal are identical. Notice that wherever there are similar textures, the matrix is lighter in color (more similar), so for example, all of the feature vectors for the vocals (V) are similar to one another, resulting in large light-colored square patterns both on and off the diagonal. Where two feature vectors correspond to different textures, for example drums and vocals, the matrix is dark.

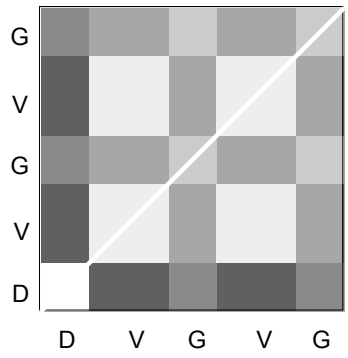

Figure 4. An idealized similarity matrix for segments of drum (D), vocal (V), and guitar (G) texture.

Notice that along the diagonal, a checkerboard pattern appears at segment boundaries, with darker regions to the upper left and lower right, and lighter regions to the lower left and upper right. Foote (2000) proposes the correlation of the similarity matrix S with a kernel based on this checkerboard pattern in order to detect segment boundaries. The general form of the kernel is:

$$C = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} \tag{6}$$

(Note that in Equation 6, row numbers increase in the downward direction whereas in the similarity matrix images, the row number increases in the upward direction. Therefore the diagonal in Equation 6 runs from upper left to lower right.) The kernel image in Figure 5 represents a larger checkerboard pattern with radial smoothing. The correlation $N(i)$ of this kernel along the diagonal of a similarity matrix S can be considered to be a measure of novelty (Foote 2000):

$$N(i) = \sum_{m=-L/2}^{L/2} \sum_{n=-L/2}^{L/2} C(m,n)S(i+m,i+n) \tag{7}$$

A graph of $N(i)$ for the similarity matrix in Figure 4 is shown in Figure 5. A peak occurs at each transition because transition boundaries have the highest correlation to the checkerboard pattern.

Checkerboard Kernel

Checkerboard Kernel with Radial Smoothing

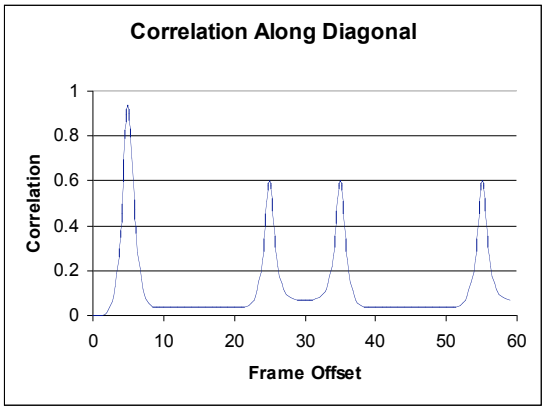

Figure 5. The correlation of the kernel shown at lower left with a similarity matrix.

Cooper and Foote (2003) extend this technique for finding segment boundaries with a statistical method for clustering segments.

Repeating Sequence Patterns

While the texture patterns described above are most useful for detecting transitions between segments, the second kind of pattern can be used to discover repetition within a song. For these patterns to appear, it is important that features reflect short-term changes. Generally, features should vary significantly with changes in the pitch of a melody or with changes in harmony. If this condition is satisfied, then there will not be great similarity within a segment, and there will not be a clear pattern of light-colored squares as seen in Figure 4. However, if a segment of music repeats with an offset of j, then $S(i, i)$ will equal $S(i, i+j)$, generating a diagonal line segment at an offset of j from the central diagonal. This is illustrated schematically in Figure 6, where it is assumed that the vocal sections (V) constitute three repetitions of very similar music, whereas the two guitar sections (G) are not so similar. Notice that each non-central diagonal line segment indicates the starting times and the duration of two similar sequences of features. Also, notice that since each pair of similar sequences is represented by two diagonal line segments, there are a total of six (6) off-central line segments in Figure 6.

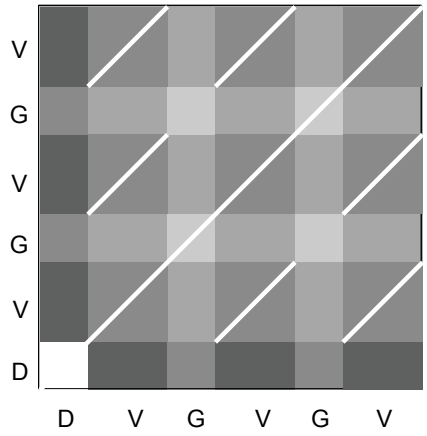

Figure 6. When sections of music are repeated, a pattern of diagonal line segments is generated.

Although not shown in Figure 6, the similarity matrix can also illustrate hierarchical relationships. For example, if each vocal section (V) consists of a phrase that is repeated, the similarity matrix would look like the one in Figure 7.

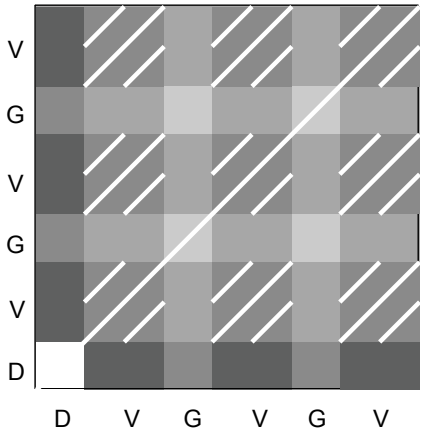

Figure 7. The vocal segments (V) in this similarity matrix contain a repetition, generating additional pattern that is characteristic of music structure.

Figure 8 illustrates both texture patterns and repeated sequence patterns from the song “Day Tripper” by the Beatles. The bridge is displayed, starting with three repetitions of a two-measure guitar phrase in the first 11 seconds, followed by six measures of vocals. Notice how a checkerboard pattern appears due to the timbral self-similarity of the guitar section (0 to 11s) and the vocal section (11 to 21s). Finer structure is also visible. A repeated sequence pattern appears within the guitar section as parallel diagonal lines. This figure uses the power spectrum below 5.5kHz as the feature vector, and uses the cosine of the angle between vectors as a measure of similarity.

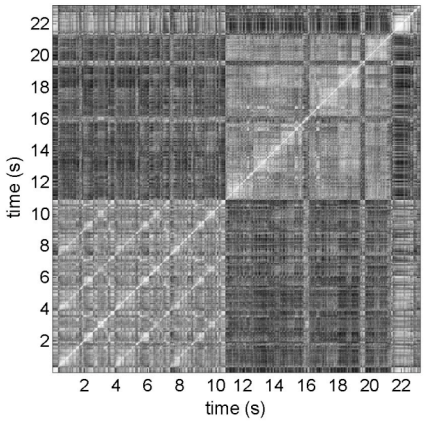

Figure 8. Similarity matrix using spectral features from the bridge of "Day Tripper" by the Beatles.

The Time-Lag Matrix

When the goal is to find repeating sequence patterns, it is sometimes simpler to change coordinate systems so that patterns appear as horizontal or vertical lines. The *time-lag* matrix r is defined by:

$$r(t, l) = S(t, t-l), \text{ where } t-l \geq 0 \quad (8)$$

Thus, if there is repetition, there will be a sequence of similar frames with a constant lag. Since lag is represented by the vertical axis, a constant lag implies a horizontal line. The time-lag version of Figure 7 is shown in Figure 9. Only the lines representing similar sequences are shown, and the grayscale has been reversed, so that similarity is indicated by black lines.

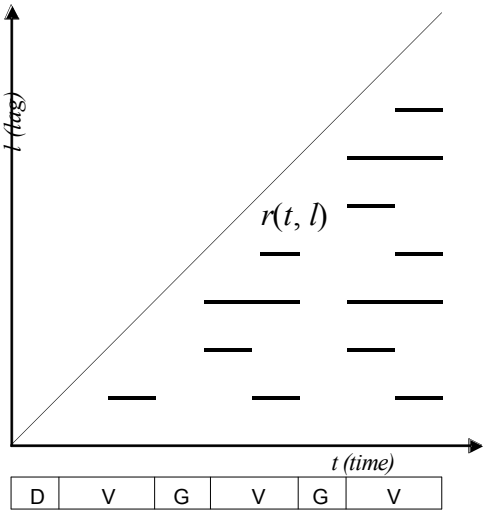

Figure 9. Time-lag matrix representation of the similarity matrix in Figure 7.

Finding Repeating Sequences

Of course, with audio data obtained from real audio recordings, the similarity or time-lag matrix will be full of noise and ambiguity arising from spurious similarity between different frames. Furthermore, repetitions in music are rarely exact; variation of melodic, harmonic, and rhythmic themes is an essential characteristic of music. In order to automate the discovery of musical structure, algorithms must be developed to identify the structure that lies within the similarity matrix.

Melodic Sequence Matching

One way to find repetition is to transcribe the melody and perform matching on the resulting symbolic transcription. While extracting the melody from a polyphonic recording (Goto 2004) is very difficult in general, an approximate transcription from an instrumental recording or from a monophonic (melody only) recording is relatively easy. Dannenberg (2002) describes a simple transcription system based on the enhanced autocorrelation algorithm (Tolonen and Karjalainen 2000) applied to a ballad recorded by John Coltrane. The transcription results in a quantized integer pitch value p_i and real inter-onset interval d_i for each note (inter-onset intervals are typically preferred over note duration in music processing). This sequence is processed as follows:

1) First, a similarity matrix is constructed where rows and columns correspond to notes. This differs from the similarity matrix described above where rows and columns correspond to feature vectors with a fixed duration.

2) Each cell of the similarity matrix $S(i, j)$ represents the duration of similar melodic sequences starting at notes i and j. A simple "greedy" algorithm is used to match these two sequences. If note i does not match note j, $S(i, j) = 0$.

3) Simplify the matrix by removing redundant entries. If a sequence beginning at i matches one at j, then there should be another match at i+1 and j+1. To simplify the matrix, find the submatrix $S(i{:}u, j{:}v)$ where the matching sequences at i and j end at u and v. Zero every entry in the submatrix except $S(i, j)$. Also, zero all entries for matching sequences of length 1.

4) Now, any non-zero entry in S represents a pair of matching sequences. By scanning across rows of S we can locate all similar sequences. Sequences are clustered: the first non-zero element in a row represents a cluster of two sequences. Any other non-zero entry in the row that roughly

matches the durations of the clustered sequences is added to the cluster. After scanning the row, all pair-wise matches are zeroed so they will not be considered again.

The result of this step is a set of clusters of similar melodic segments. Because repetitions in the music are not exact, there can be considerable overlap between clusters. It is possible for a long segment to be repeated exactly at one offset, and for a portion of that same segment to be repeated several times at other offsets. It may be desirable to simplify the analysis by labeling each note with a particular cluster. This simplification is described in the next section.

Simplification or Music Explanation

The goal of the simplification step is to produce *one* possible set of labels for notes. Ideally, the labels should offer a simple "explanation" of the music that highlights repetition within the music. The AABA structure common in songs is a typical explanation. In general, longer sequences of notes are preferable because they explain more, but when sequences are too long, interesting substructure may be lost. For example, the structure AABAAABA could also be represented as AABA repeated, i.e. the structure could be labeled AA, but most theorists would consider this to be a poor explanation. Hierarchical explanations offer a solution, but there is no formal notion as yet of the optimal simplification or explanation.

Dannenberg uses a "greedy" algorithm to produce reasonable explanations from first note to last. (Dannenberg and Hu 2002) Notes are initially unlabeled. As each unlabeled note is encountered, search the clusters from the previous section to find one that includes the unlabeled note. If a cluster is found, allocate a new label, e.g. "A", and label every note included in the cluster accordingly. Continue labeling with the next unlabeled note until all notes are processed.

Figure 10 illustrates output from this process. Notice that the program discovered substructure within what would normally be considered the "bridge" or the B part, but this substructure is "real" in the sense that one can see it and hear it. The gap in the middle of the piece is a piano solo where transcription failed. Notice that the program correctly determines that the saxophone enters on the bridge (the B part) after the piano solo. The program also identifies the repeated 2-measure phrase at the end. It fails to notice the structure of ascending pitches at the very end because, while this is a clear musical gesture, it is not based on the repetition of a note sequence.

Figure 10. A computer analysis of "Naima" by John Coltrane. The automatic transcription appears as a "piano roll" at the top, the computer analysis appears as shaded bars, where similar shading indicates similar sequences, and conventional labels appear at the bottom.

Finding Similar Sequences in the Similarity Matrix

Typically, transcription of a music signal into a sequence of notes is not possible, so similar sequences must be detected as patterns in the similarity or time-lag matrix. For example, Bartsch and Wakefield (2001) filter along diagonals of a similarity matrix to detect similarity. This assumes nearly constant tempo, but that is a good assumption for the popular music used in their study. Their objective was not to identify the beginnings and endings of repeating sequences but to find the chorus of a popular song for use as an "audio thumbnail" or summary. The thumbnail is selected as the maximum element of the filtered similarity matrix, with the additional constraints that the lag is at least one tenth of the length of the song and the thumbnail does not appear in the last quarter of the song.

Peeters and Rodet (2003) suggest using a 2D structuring filter on the lag matrix to detect similar sequences. Their filter counts the number of values in the neighborhood to the left and right of a point that

are above a threshold. To allow for slight changes in tempo, which results in lines that are not perfectly horizontal, neighbor cells above and below are also considered. Lu, Wang, and Zhang (Lu, Wang, and Zhang 2004) suggest erosion and dilation operations on the lag matrix to enhance and detect significant similar sequences.

Dannenberg and Hu (2003) use a discrete algorithm to find similar sequences and is based on the idea that a path from cell to cell through the similarity matrix specifies an alignment between two subsequences of the feature vectors. If the path goes through $S(i, j)$, then vector i is aligned with j. This suggests using a dynamic time warping (DTW) algorithm (Rabiner and Juang 1993), and the actual algorithm is related to DTW.

The goal is to find alignment paths that maximize the average similarity of the aligned features. A partial or complete path P is defined as a set of pairs of locations and is rated by the average similarity along the path:

$$q(P) = \frac{1}{|P|} \sum_{(i,j) \in P} S(i, j) \tag{9}$$

where $|P|$ is the path length using Euclidean distance. Paths are extended as long as the rating remains above a threshold. Paths are constrained to move up one cell, right one cell, or diagonally to the upper right as shown in Figure 11 (and adopting the orientation of the similarity matrix visualizations where time increases vertically and to the right). Therefore, every point that is on a path can be reached from below, from the left, or from the lower left. Each cell (i,j) of an array is computed by looking at the cell below, left, and below left to find the (previously calculated) best path (highest $q(P)$) passing through those cells. Three new ratings of r are computed by extending each of the three paths to include (i,j). The path with the highest rating is remembered as the one passing through (i,j).

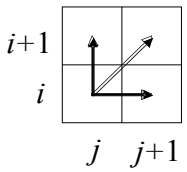

Figure 11. Extending a path from $S(i, j)$.

Because cells depend on previously computed values to the lower left, cells are computed along diagonals of constant $i+j$, from lower left to upper right (increasing $i+j$). When no path has a rating above some fixed threshold, the path ends. A path may begin wherever $S(i, j)$ is above threshold and no previous paths exist to be extended.

Forming Clusters

After alignment paths are found, they are grouped into clusters. So if sequence A aligns to sequence B, and sequence A also aligns to sequence C, then A, B, and C should be grouped in a single cluster. Unfortunately, it is unlikely that the alignments of A to B and A to C use exactly the same frames. It is more likely that A aligns to B and A′ aligns to C, where A and A′ are mostly overlapping. This can be handled simply by considering A to equal A′ when they start and end within some fraction of their total length, for example within 10 percent. Once clusters are formed, further simplification and explanation steps can be performed as described above.

Isolating Line Segments from the Time-Lag Matrix

If nearly constant tempo can be assumed, the alignment path is highly constrained and the alignment path approach may not work well. Taking advantage of the fact that similar sequences are represented by

horizontal lines in the time-lag matrix, Goto (2003a) describes an alternative approach to detecting music structure. In this work, the time-lag matrix is first normalized by subtracting a local mean value while emphasizing horizontal lines. In more detail, given a point *r(t,l)* in the time-lag matrix, six-directional local mean values along the right, left, upper, lower, upper-right, and lower-left directions starting from *r(t,l)* are calculated, and the maximum and minimum are obtained. If the local mean along the right or left direction takes the maximum, *r(t,l)* is considered a part of a horizontal line and emphasized by subtracting the minimum from *r(t,l)*. Otherwise, *r(t,l)* is considered a noise and suppressed by subtracting the maximum from *r(t,l)*; noises tend to appear as lines along the upper, lower, upper-right, and lower-left directions. Then, a summary is constructed by integrating over time:

$$R_{all}(t,l) = \int_l^t \frac{r(\tau,l)}{t-l} d\tau \tag{10}$$

R_{all} is then smoothed by a moving average filter along the lag. The result is sketched in Figure 12. R_{all} is used to decide which lag values should be considered when searching for line segments in the time-lag matrix. A thresholding scheme based on a discriminant criterion is used. The threshold is automatically set to maximize the following between-class variance of the two classes established by the threshold:

$$\sigma_B^2 = \omega_1 \omega_2 (\mu_1 - \mu_2)^2 \tag{11}$$

where ω_1 and ω_2 are the probabilities of class occurrence (the fraction of peaks in each class), and μ_1 and μ_2 are the means of peak heights in each class.

Each peak above threshold determines a lag value, l_p. For each peak, the one-dimensional function $r(\tau, l_p)$ is searched over $l_p \le \tau \le t$. A smoothing operation is applied to this function and the discriminant criterion of Equation 11 is again used to set the threshold. The result is the beginning and ending points of line segments that indicate repeated sections of music.

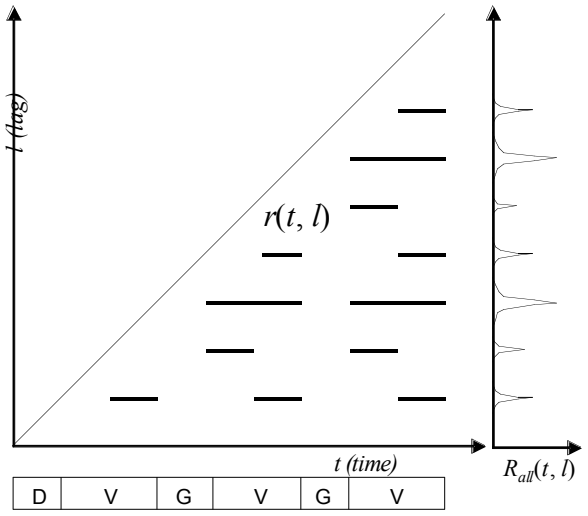

Figure 12. The summary $R_{all}(t, l)$ indicates the possibility that there are similar segments at a lag of l.

Modulation Detection

A common technique in pop music when repeating a chorus is to change the key, typically modulating upward by half steps. (Note that "modulation" in music is not related to amplitude modulation or frequency modulation in the signal processing sense.) Since modulation changes all the pitches, it is unlikely that a feature vector that is sensitive to pitch sequences could detect any similarity between musical passages in different keys. To a first approximation, a modulation in music corresponds to

frequency scaling, as if changing the speed of a vinyl record turntable or changing the sample rate of a digital recording. On a logarithmic frequency scale, modulation is simply an offset, and when the scale is circular as with pitch classes and chroma, modulation is a rotation. To rotate a vector by ζ, the value of the i^{th} feature is moved to become feature $(i + \zeta)$ *mod* 12. One would expect the chroma vectors for a modulated passage of music to be quite similar to a rotation of the chroma vectors of the unmodulated version.

Goto (2003a) exploits this property of the chroma vector by extending the time-lag matrix to incorporate chroma vector rotation by a transposition amount ζ. Denoting V_t^{ξ} as a transposed (rotated) version of a chroma vector V_t, $r_{\zeta}(t, l)$ is the similarity between V_t^{ξ} and the untransposed vector V_{t-l}^{0}. Since we cannot assume the number of semitones at the modulation in general, the line segment detection is performed on each of 12 versions of $r_{\zeta}(t, l)$ corresponding to the 12 possible transpositions (this usually does not increase harmful false matches). The segments from all 12 versions are combined to form the set of repeated sections of music, and the transposition information can be saved to form a more complete explanation of the music structure.

Chorus Selection after Grouping Line Segments

Since each line segment indicates just a pair of repeated contiguous segments, it is necessary to organize into a cluster the line segments that have mostly overlapping frames. When a segment is repeated n times ($n \geq 3$), the number of line segments to be grouped in a cluster should theoretically be $n(n-1)/2$ in the time-lag matrix. Aiming to exhaustively detect all the repeated segments (choruses) appearing in a song, Goto (2003a) describes an algorithm that redetects missing (hidden) line segments to be grouped by top-down processing using information on other detected line segments. The algorithm also appropriately adjusts the start and end times of line segments in each cluster because they are sometimes inconsistent in the bottom-up line-segment detection. Lu, Wang, and Zhang (Lu, Wang, and Zhang 2004) describe another approach to obtain the best overall combination of segment similarity and duration by adjusting segment boundaries.

A cluster corresponding to the chorus can be selected from those clusters. In general, a cluster that has many and long segments tends to be the chorus. In addition to this property, Goto (2003a) uses heuristic rules to select the chorus with a focus on popular music; for example, when a segment has half-length repeated sub-segments, it is likely to be the chorus. The *choruslikeness* (chorus possibility) of each cluster is computed by taking these rules into account, and the cluster that maximizes the choruslikeness is finally selected.

Texture Sequences

Detecting repeating patterns in the similarity matrix is equivalent to finding sequences of similar feature vectors. An alternative is to find sequences of similar texture classes. Aucouturier and Sandler (2002) perform a segmentation using hidden Markov models as described earlier. The result is a "texture score," a sequence of states, e.g. 11222112200, in which patterns can be discovered. They explore two methods for detecting diagonal lines in the similarity matrix. The first is kernel convolution, similar to the filter method of Bartsch and Wakefield (2001). The second uses the Hough Transform (Leavers 1992), a common technique for detecting lines in images. The Hough Transform uses the familiar equation for a line: $y = mx+b$. A line passing through the point (x,y) must obey the equation $b = -mx+y$, which forms a line in the (m,b) space. A series of points along the line $y = m_0x+b_0$ can be transformed to a series of lines in (m,b) space that all intersect at (m_0,b_0). Thus, the problem becomes one of finding the intersection of lines. This can be accomplished, for example, by making a sampled two-dimensional image of the (m,b) space and searching for local maxima. It appears that the Hough Transform could be used to find patterns in the similarity matrix as well as in the "texture score" representation.

One of the interesting features of the texture score representation is that it ignores pitch to a large extent. Thus, music segments that are similar in rhythm and instrumentation can be detected even if the pitches do not match. For example, "Happy Birthday" contains 4 phrases of 6 or 7 notes. There are obvious parallels between these phrases, yet they contain 4 distinct pitch sequences. It seems likely that pitch sequences, texture sequences, rhythmic sequences, and other feature sequences can provide complementary views that will facilitate structure analysis in future systems.

Music Summary

Browsing images or text is facilitated by the fact that people can shift their gaze from one place to another. The amount of material that is skipped can be controlled by the viewer, and in some cases, the viewer can make a quick scan to search for a particular image or to read headlines. Music, on the other hand, exists in time rather than space. Listeners cannot time-travel to scan a music performance, or experience time more quickly to search for musical "headlines." At best, one can skip songs or use fast-forward controls with recorded music, but even this is confusing and time-consuming.

One application of music structure analysis is to enable the construction of musical "summaries" that give a short overview of the main elements of a musical work. Summaries can help people search for a particular piece of music they know or locate unfamiliar music they might like to hear in full. By analogy to low-resolution versions of images often used to save space or bandwidth, summaries of music are sometimes called "music thumbnails."

Cooper and Foote describe a simple criterion for a music summary of length L: the summary should be maximally similar to the whole. In other words, a summary can be rated by summing the similarity between each feature vector in the summary with each feature vector in the complete work. The rating for the summary beginning at feature vector i is:

$$Q_L(i) = \frac{1}{NL} \sum_{m=i}^{i+L} \sum_{n=1}^{N} S(m,n) \tag{12}$$

The best summary is then the one starting at the value of i the maximizes $Q_L(i)$. The formula can be extended by weighting $S(m,n)$ to emphasize earlier or louder sections of the song.

Other approaches to summary construction are outlined by Peeters, La Burthe, and Rodet (2002). Assume that music has been segmented using one of the techniques described above, resulting in three classes or labels A, B, and C. Some of the interesting approaches to musical summary are:

- use the most common class, which in popular music is often the chorus. Some research specifically aims to determine the chorus as described earlier. (Bartsch and Wakefield 2001; Goto 2003a);
- use a sample of music from each class, i.e A, B, C.
- use examples of each class transition, i.e. A→B, B→A, A→C.

In all cases, audio segments are extracted from the original music recording. Unfortunately, artificially and automatically generated transitions can be jarring to listeners. Music structure analysis can help to pick logical points for transitions. In particular, a cut from one phrase of music to a repetition of that phrase can be inaudible. When a cut must be made to a very different texture, it is generally best to make the cut at an existing point of strong textural change. In most music, tempo and meter create a framework that is important for listening. Cuts that jump from the end of one measure to the beginning of another preserve the short-term metrical structure of the original music and help listeners grasp the harmonic and melodic structure more easily. Segments that last 2, 4, or 8 measures (or some duration that relates to the music structure) are more likely to seem "logical" and less disruptive. Thus, music structure analysis is not only important to determine what sections of music to include in a summary, but also to organize those sections in a way that is "musical" and easy for the listener to comprehend.

An alternative to the construction of "music thumbnails" is to provide a "smart" interface that facilitates manual browsing of entire songs. The SmartMusicKIOSK music listening station (Goto 2003b) displays a time-line with the results of an automatic music structure analysis. In addition to the common stop, pause, play, rewind, and fast forward controls, the SmartMusicKIOSK has controls labeled "next chorus," "next section," and "prev section." (See Figure 13.) These content-based controls allow users to skim rapidly through music and give a graphical overview of the entire music structure, which can be understood without listening to the entire song.

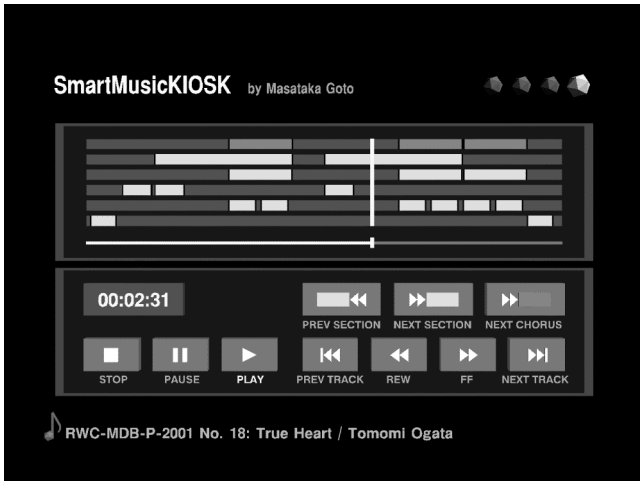

Figure 13. The SmartMusicKIOSK user interface showing music structure and structure-related controls.

Evaluation

Most research in this area has been exploratory, with no means to evaluate whether computer-generated structures and segments are "correct." In most cases, it is interesting simply to explore what types of structures can be uncovered and what methods can be used. Quantitative evaluations will become more important as problems are better understood and when competing methods need to be compared.

Tzanetakis and Cook conducted a pilot study (1999) to compare their automatic segmentation with human segmentation. They found most human subjects agreed on more than half of the segments, and their machine segmentation found more than half of the segments that humans agreed upon.

Bartsch and Wakefield (2001) hand-selected "true audio thumbnails" from 93 popular songs and measured "recall," the fraction of true frames labeled by their program as the chorus, and "precision," the fraction of labeled chorus frames that are true frames. With the chorus length set to around 20 to 25 seconds, the average recall and precision is about 70%, compared to about 30% for a chorus interval selected at random.

Goto (Goto 2003a) also used hand-labeled choruses in 100 popular songs from the RWC Music Database, a source that enables researchers to work with common test data. (Goto et al. 2002) Goto judged the system output to be correct if the F-measure was more than 0.75. The F-measure is the harmonic mean of recall rate (R) and precision rate (P): F-measure = 2RP/(R+P). The system dealt correctly with 80 out of 100 songs.

Evaluating music structure descriptions is difficult. Structure exists at many levels and often exhibits hierarchy. The structure intended by the composer and perhaps determined by a music theorist may not correspond to the perception of the typical listener. Nevertheless, one can ask human subjects to identify pattern and structure in music, look for consistency between subjects, and then compare human descriptions to machine descriptions of music. One can also evaluate the impact of music structure detection upon tasks such a browsing, as in SmartMusicKIOSK (Goto 2003b).

Summary and Conclusions

Knowledge of musical structure can be used to construct music summaries, assist with music classification, provide high-level interfaces for music browsing, and offer high-level top-down guidance for further analysis. Automatic analysis of music structure is one source of music meta-data, which is important for digital music libraries.

High-level music structure is generally represented by partitioning the music into segments. Sometimes, segments are labeled to indicate similarity to other segments. There are two main principles used to detect high-level music structure. First, segment boundaries tend to occur when there is a substantial change in musical texture. In other words, this is where the music on either side of the boundary is self-similar, but the two regions differ from each other. Secondly, segments can be located by detecting patterns of repetition within a musical work.

It should be noted that the music signal, viewed as a time-domain waveform, is not directly useful for analysis because repetition in music is never exact enough to reproduce phase and amplitude relationships. Therefore, the signal is processed to obtain features that capture useful and more-or-less invariant properties. In the case of texture analysis, features should capture the overall spectral shape and be relatively insensitive to specific pitches. Low-order MFCCs are often used to measure texture similarity. To detect music repetition, features should capture changes in pitch and harmony, ignoring texture which may change from one repetition to the next. The chroma vector is often used in this case.

The similarity matrix results from a comparison of all feature vector pairs. The similarity matrix offers an interesting visualization of music, and it has inspired the application of various image-processing techniques to detect music structure. Computing the correlation with a “checkerboard” kernel is one method for detecting texture boundaries. Using filters to detect diagonal lines is one method for detecting repetition.

Detecting segment boundaries or music repetition generates individual segments or pairs of segments. Further processing can be used to merge segments into clusters. Hidden Markov models, where each hidden state corresponds to a distinct texture, have been applied to this problem. When music is analyzed using repetitions, the structure can be hierarchical, and the structure is often ambiguous. Standard clustering algorithms assume a set of distinct, fixed items, but with music analysis, the items to be clustered are possibly overlapping segments whose start and end times might be adjustable.

Music structure analysis is a rapidly-evolving field of study. Future work will likely explore the integration of existing techniques, combining texture-based with repetition-based segmentation. More sophisticated features including music transcription will offer alternative representations for analysis. Finally, there is the possibility to detect richer structures, including hierarchical patterns of repetition, rhythmic motives, harmonic progressions and key changes, and melodic phrases related by transposition.

Acknowledgments

The authors wish to thank Jonathan Foote for Figure 8. Jean-Julien Aucotourier, Mark Bartsch, Jonathan Foote, Geoffroy Peeters, Ning Hu, Xavier Rodet, Mark Sandler, George Tzanetakis, and Greg Wakefield have made contributions through their work, discussions, and correspondence.

References

Aucouturier, J.-J., F. Pachet, and M. Sandler. 2005. ""The Way It Sounds": Timbre Models for Structural Analysis and Retrieval of Music Signals." *IEEE Transactions on Multimedia*, (to appear).

Aucouturier, J.-J., and M. Sandler. 2001. "Segmentation of Musical Signals Using Hidden Markov Models." In *Proceedings of the 110th Convention of the Audio Engineering Society*. Audio Engineering Society.

Aucouturier, J.-J., and M. Sandler. 2002. "Finding Repeating Patterns in Acoustic Musical Signals: Applications for Audio Thumbnailing." In *AES22 International Conference on Virtual, Synthetic and Entertainment Audio*. Audio Engineering Society, pp. 412-421.

Bartsch, M., and G. H. Wakefield. 2001. "To Catch a Chorus: Using Chroma-Based Representations For Audio Thumbnailing." In *Proceedings of the Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New York: IEEE, pp. 15-18.

Cooper, M., and J. Foote. 2003. "Summarizing Popular Music via Structural Similarity Analysis." In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New York: IEEE, pp. 127-130.

Dannenberg, R. B. 2002. "Listening to "Naima": An Automated Structural Analysis from Recorded Audio." In *Proceedings of the 2002 International Computer Music Conference (ICMC 2002)*. San Francisco: International Computer Music Association, pp. 28-34.

Dannenberg, R. B., and N. Hu. 2002. Discovering Musical Structure in Audio Recordings. In Anagnostopoulou, C*., et al.* eds. *Music and Artificial Intelligence, Second International Conference, ICMAI 2002*. Berlin: Springer-Verlag, pp. 43-57

Dannenberg, R. B., and N. Hu. 2003. "Pattern Discovery Techniques for Music Audio." *Journal of New Music Research, 32*(2), 153-164.

Foote, J. 1999. "Visualizing Music and Audio Using Self-Similarity." In *Proceedings of ACM Multimedia '99*. New York: Association for Computing Machinery, pp. 77-80.

Foote, J. 2000. "Automatic Audio Segmentation Using a Measure of Audio Novelty." In *Proceedings of the International Conference on Multimedia and Expo (ICME 2000)*. New York: IEEE, pp. 452-455.

Galas, T., and X. Rodet. 1990. "An Improved Cepstral Method for Deconvolution of Source-Filter Systems with Discrete Cepstral: Application to Musical Sounds." In *1990 International Computer Music Conference (ICMC 1990)*. San Francisco: International Computer Music Association, pp. 82-84.

Goto, M. 2003a. "A Chorus-Section Detecting Method for Musical Audio Signals." In *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing Proceedings (ICASSP 2003)*. New York: IEEE, pp. V-437-440.

Goto, M. 2003b. "SmartMusicKIOSK: Music Listening Station with Chorus-Search Function." In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology (UIST 2003)*. New York: Association for Computing Machinery, pp. 31-40.

Goto, M. 2004. "A Real-time Music Scene Description System: Predominatn-F0 Estimation for Detecting Melody and Bass Lines in Real-world Audio Signals." *Speech Communication (ISCA Journal), 43*(4), 311-329.

Goto, M., T. Nishimura, H. Hashiguchi, and R. Oka. 2002. "RWC Music Database: Popular, Classical, and Jazz Music Databases." In *ISMIR 2002 Conference Proceedings*. Paris: IRCAM, pp. 287-288.

Hu, N., R. B. Dannenberg, and G. Tzanetakis. 2003. "Polyphonic Audio Matching and Alignment for Music Retrieval." In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New York: IEEE, pp. 185-188.

Leavers, V. F. 1992. *Shape Detection in Computer Vision Using the Hough Transform*. Berlin: Springer-Verlag.

Logan, B., and S. Chu. 2000. "Music Summarization Using Key Phrases." In *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing Proceedings (ICASSP 2000)*. New York: IEEE, pp. II-749-752.

Lu, L., M. Wang, and H.-J. Zhang. 2004. "Repeating Pattern Discovery and Structure Analysis from Acoustic Music Data." In *Proceedings of the 6th ACM SIGMM International Workshop on Multimedia Information Retrieval*. New York: Association for Computing Machinery, pp. 275-282.

Peeters, G., A. L. Burthe, and X. Rodet. 2002. "Toward Automatic Audio Summary Generation from Signal Analysis." In *ISMIR 2002 Conference Proceedings*. Paris: IRCAM, pp. 94-100.

Peeters, G., and X. Rodet. 2003. "Signal-based Music Structure Discovery for Music Audio Summary Generation." In *Proceedings of the 2003 International Computer Music Conference (ICMC 2003)*. San Francisco: International Computer Music Association, pp. 15-22.

Rabiner, L., and B.-H. Juang. 1993. *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice Hall.

Shepard, R. 1964. "Circularity in Judgements of Relative Pitch." *Journal of the Acoustical Society of America, 36*(12), 2346-2353.

Tolonen, T., and M. Karjalainen. 2000. "A computationally efficient multi-pitch analysis model." *IEEE Transactions on Speech and Audio Processing, 8*(6), 708-716.

Tzanetakis, G., and P. Cook. 1999. "Multifeature Audio Segmentation for Browsing and Annotation." In *Proceedings of the Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New York: IEEE.