Discovering Musical Structure in Audio Recordings

Roger B. Dannenberg and Ning Hu

Carnegie Mellon University, School of Computer Science, Pittsburgh, PA 15217, USA
{rbd, ninghu}@cs.cmu.edu

Abstract. Music is often described in terms of the structure of repeated phrases. For example, many songs have the form AABA, where each letter represents an instance of a phrase. This research aims to construct descriptions or explanations of music in this form, using only audio recordings as input. A system of programs is described that transcribes the melody of a recording, identifies similar segments, clusters these segments to form patterns, and then constructs an explanation of the music in terms of these patterns. Additional work using spectral information rather than melodic transcription is also described. Examples of successful machine "listening" and music analysis are presented.

1 Introduction

Machine recognition of music is an important problem, with applications in databases, interactive music systems, and musicology. It would not be an exaggeration to say that music recognition and understanding is a central problem of computer music research.

Recognition and understanding are not well-defined terms in the field of music, largely because we have no general theory of semantics for music. Music often seems to have an internal logic and certainly exhibits rich structures, but we have no formal descriptions of meaning as in many other domains. For example, a speech understanding system or a natural language translation system can be evaluated objectively in terms of how well it preserves semantic information across a change of representation. Alternatively, internal representations of meaning can be examined. In contrast, music has no accepted "meaning" or semantic representation. Without a formal semantic theory, the concept of "music understanding" is on shaky ground.

One path for research is to pursue analogies to semantic theories from other fields. For example, if "recognition" is defined in terms of translation, as in most speech recognition, then the analogy to music might be music transcription, including pitch recognition, beat tracking, bass-line extraction, etc. If understanding is defined in terms of parsing, then we can pursue the parsing of music into phrases, voices, sections, etc.

Another path is to take music more on its own terms. When we listen to music, there is no question that we identify repeated patterns. It is also clear that music is full of relationships and analogies. For example, a melodic fragment can be repeated at some tonal or chromatic transposition, a melody can be repeated by a different instrument or with different orchestration, and a passage can be echoed with a change in dynamics. Multiple relationships are common within a composition, and they can also occur between elements of a composition and elements outside of the composition. For example, when a melody contains a scale, we can easily recognize the scale even if we have not heard the melody before.

Structures formed by relationships are themselves recognizable. For example, 12-bar blues or an AABA song forms in popular music and jazz say nothing about the specific content of the music, but only about where we expect to find similarities and

relationships. The fact that we describe music in this way is an important clue about the very nature of music.

We believe that musical relationships form the basis of music, and that an important component of "music understanding" is the identification of these relationships. The simplest and most salient relationship is a literal repetition of a segment of music at a later time. Other relationships involve repetition with a change of one or more aspects of the music, such as pitch or tempo. Higher order transformations are also possible.[13] Consider a phrase that is transposed upward by a step, and then the result is transposed upward by another step. In this case, it is not just the intervals but also the transposition that is repeated.

In this research, we begin with the premise that an important part of music understanding is the identification of repetition within the music. Repetition generates structure, but the recovery of structure is not trivial. We describe some new methods designed for this purpose. One important aspect of the work described here is that it starts with audio recordings of music rather than symbolic ones. While symbolic representations would be natural for this kind of work, we believe that symbolic representations oversimplify the problem. Much of our interest lies in discovering how it is possible to recover structure from the least-structured representation of music: sound itself. We are especially interested in applications to audio databases, another reason to work with audio representations.

This work began using simple monophonic pitch estimation to extract data from audio where a single voice is prominent. More recently, we have had some success with a chroma-based representation on more polyphonic or chordal music, where (current) monophonic pitch estimation techniques are not useful.

In the next section, we highlight some related work. Following that, we describe a system of programs, SAM1, that performs a structural analysis of music from audio input. In the section "Polyphonic Music," we describe experiments to use spectra rather than melodies to analyze music, and we show some early results. We finish with a discussion and conclusions.

2 Related Work

The idea that similarity in music is an important aspect of musical structure is not new. Any standard text on music theory will introduce the ideas of repetition at different time scales, and of common forms such as rondo or sonata allegro, and these forms will be described largely in terms of what and when elements of the music are repeated or transformed.

Finding repeated phrases or sequences for various purposes is the goal of numerous studies. David Cope has searched music for short patterns, or *signatures*, as a way to identify characteristics of compositions and composers.[5] Conklin and Anagnostopoulou describe a pattern-discovery program for music that uses a probabilistic approach to separate significant patterns from chance ones. [4] Stammen and Pennycook used dynamic programming to search for repeated contours in jazz improvisations [19], and Rolland and Ganascia describe work using dynamic programming to find patterns in musical scores.[17]

Simon and Sumner described music listening as pattern formation, and introduced the idea that music can be encoded in terms of operations and transformations.[18] They suggest that listeners construct encodings when they listen to music, and that the encoding chosen by a listener will tend toward the shortest encoding possible. This idea is related to data compression, and more recent studies have used techniques from data compression to encode and generate music.[8]

Michael Leyton has written about structure and form as generative; that is, structures are perceived as a series of generative transfers from one space to another. [9] This suggests that the "content" of music is not melody, harmony, and rhythm but

the transfer (by Leyton's *control* groups) of these elements (Leyton's *fiber groups*) within a composition, forming relationships and therefore structure.

This work is an instance of music analysis by computer, of which there are many examples in the literature. Foote and Cooper [6] constructed a similarity matrix from midi and audio data, and Wakefield and Bartsch created a similar structure using spectral data.[1, 2] This has inspired some of our work.

3 Structure from Melody

A set of programs was designed to apply these ideas to actual music content. These programs, which we will collectively call SAM1, for "Structural Analysis of Music – version 1," perform of a number of steps:

- Read or record digital audio,
- Extract a pitch contour,
- Compute a similarity matrix,
- Find clusters of similar sequences,
- Build an explanation of the music in terms of structural relationships.

These steps are described in the following subsections, using a particular recording as an illustration. Results with other recordings will be described later.

3.1 Acquire Digital Audio

An audio CD recording of John Coltrane's "Naima" [3] was used as input to the program. The digital audio was copied directly from the CD. One of the two stereo channels contained the strongest saxophone signal, so it was extracted to a mono sound file and down-sampled to 22,050Hz for the next step.

3.2 Pitch Extraction and Segmentation

Pitch estimation was performed using an autocorrelation technique on overlapping windows. Autocorrelation [14, 15] computes the correlation between the signal and a copy of the signal shifted by different amounts of time. When the shift is equal to a multiple of fundamental periods, the correlation will be high. In theory, one simply finds the first peak in the autocorrelation. In practice, there are small peaks corresponding to strong partials above the fundamental, and there may be higher peaks at multiples of the fundamental due to noise or subharmonics. The pitch estimation algorithm employs several simple heuristics to find the peak that corresponds to the fundamental.

In some cases there is no clear peak or fundamental, or the amplitude is low indicating there may be no signal present. Pitch estimates at these places are ignored. We also attempted to identify note boundaries using amplitude information, but found this to be unreliable, even in "Naima" where most notes are clearly articulated with clear attacks. Spectral techniques might also be used for note onset detection [16], but this could be a problem in a polyphonic context. We settled on using pitch estimates to segment the signal into notes.

After pitches have been estimated, the program labels regions of relatively stable pitch as notes. The program uses median filters to reject outliers from the sequence of pitch estimates and then searches for intervals where pitch varies within a limited range of 70 cents (0.7 semitones). The overall pitch of the note is based on the median value of up to 1 second of pitch estimates. This is more reliable than estimating pitch based on the very beginning of the note where pitch is less stable.

Note durations are represented in seconds, and no attempt is made to identify beats or transcribe rhythm.

3.3 Similarity Matrix

After obtaining a sequence of notes, we need to find melodic fragments that repeat within the composition. Imagine comparing the melody starting on the first note to the melody starting on the second note, then the third note, etc., until the initial melody is compared to every other location in the piece. Then, the procedure is repeated starting with the second note and comparing that to every other location, etc. This approach gives rise to a 2-dimensional matrix we call the *similarity matrix*.

The similarity matrix contains elements $\mathbf{s}_{i,j}$, representing the length (in seconds) of a melodic sequence starting at note *i* that matches a melodic sequence starting at note *j*. Matches are determined by a relatively simplistic algorithm that works as follows: Starting at positions *i* and *j*, look for a match between note *i* and note *j*. If they match in both pitch and approximate duration, advance to the next notes and look for a match. If not, consider different rules to construct a match:

- Consolidate notes *i* and *i*+1 if they are near in pitch and duration to match note *j*,
- Consolidate notes *j* and *j*+1 to match note *i*,
- Consolidate notes *i* and *i*+1 and match to the consolidation of notes *j* and *j*+1, or
- Omit *i* and *j* if they are short and match *i*+1 to *j*+1.

A "greedy" algorithm without backtracking is used. If there is no match at all, the length of the match is reported as zero. In general, these rules are intended to identify similar melodic contours. Consolidation [11] helps to match transcriptions with spurious note onsets. The transcription also tends to skip short notes, so we allow matches to ignore short durations. Other melodic similarity metrics could certainly be substituted for this one.

The diagonal of the matrix is uninteresting because we already know a melodic sequence is similar to itself, so the diagonal is filled with zeros. Because similarity is symmetric, only half of the matrix must be computed. When similar sequences are found, the length of one is stored in the upper half and the length of the other is stored in the lower half of the matrix.

3.4 Find Clusters of Similar Phrases

The similarity matrix identifies similar phrases, but there is no explicit indication that a phrase occurs more than once. The purpose of this next step is to form clusters from pairs of similar phrases.

Unfortunately, similarity is not transitive. If phrase A is similar to phrase B, and phrase B is similar to phrase C, it is not necessarily the case that phrase A is similar to C. This is because thresholds are used in comparing pitches and durations.

To form clusters, we scan across rows of the similarity matrix (or equivalently, down columns). If there are, say, two non-zero elements in row *r* in columns *m* and *n*, and if these all have approximately the same values (durations), it means that there are similar melodic sequences beginning at locations *r, m,* and *n*. These three locations and their durations form a cluster.

When a cluster is identified, it implies other similarities. E.g. we expect to find a similarity between *m* and *n*, so when *m* and *n* are added to a cluster, zeros are written to locations (*m*,*n*) and (*n*,*m*). The cluster implies other similarities as well. For example, if *r* and *n* are similar and span more than one note each, we expect to find a similarity between *r*+1 and *n*+1. These implied similarities are also zeroed in the matrix to avoid further consideration of these elements.

The result of this step is a list of clusters consisting of sets of time intervals in the overall piece. Within each cluster, all time intervals refer to similar melodic sequences at different temporal locations.

3.5 Building an "Explanation"

The final step of the program is to describe the music in terms of structural relationships, the overall goal of the program. Intuitively, we view this as an "explanation" of the music. For each moment of music, we would like to have an explanation of the form "this is an instance of *phrase-name,*" where *phrase-name* is an arbitrary name that denotes one of the clusters identified in the previous step.

To build such an explanation, we proceed from first note to last. At each point, if the note has not already been "explained," search the clusters to find an interval that includes the note. Name the cluster, e.g. using the next name in the sequence "A," "B," "C," etc. Then, for each unexplained note included in an interval in the cluster, mark the note with the cluster name.

3.6 Results from SAM1

Figure 1 illustrates intermediate and final results of the program applied to "Naima." The input waveform is shown at the top. Next, a piano-roll transcription is shown. Because the recording contains bass, drums, and piano along with the saxophone, and because the note identification and segmentation requires many fairly stable pitch estimates, some of the short notes of the performance (which are clearly audible) are not captured in this transcription. Also, the piano solo is missed completely except for a few spurious notes. This is because the piano notes are often in chords and because the notes decay quickly and therefore do not meet the stability criterion.

Below the note are shown clusters. A thin line connects the time intervals of each cluster, and the time intervals are shown as thick lines. Finally, the bottom of the figure shows the final result. Here, we see that "Naima" begins with an AABA form, where the B part is actually $b_1b_1b_2$, so the flattened structure should be $AAb_1b_1b_2A$, which is what we see in Figure 1. Following that, the recording contains a piano solo that was mostly missed by the transcription. After the piano solo, the saxophone enters, not by repeating the entire AABA form, but on the "bridge," the "B" part. This aspect of the structure can be seen clearly in the figure. Following the final $b_1b_1b_2A$, the last 2 measures are repeated 3 times (as shown in the figure) and followed by a rising scale of half notes that does not appear elsewhere in the performance.

Overall, the analysis is excellent, and it was an encouraging surprise to see such a clear form emerge solely from audio input without the benefit of polyphonic transcription, beat tracking, or other analyses.

To be fair, "Naima" was used to debug and refine the program, so there is some danger that the program is actually tuned to fit the data at hand. To further evaluate the program, we applied it to monophonic recordings of some simple melodies. These are the familiar Christmas tune "We Three Kings," and a standard jazz tune "Freddie the Freeloader" composed by Miles Davis. An amateur violinist performed the first of these, and an experienced jazz trumpet player played the second (without improvisation.)

The program was applied to these recordings without any further tuning or refinement. The results of these experiments are also quite good, as shown in Figures 2 and 3. One bit of explanation is in order for Figure 3. "Freddie the Freeloader," basically a 12-bar blues, has a first and second ending for the last two bars, so it was performed with the repeat for this experiment. The program found the similarity between the first and second halves of the performance, so the overall form might have been simply AA. However, the program also found and reported substructure

within each A that corresponds to what we would ordinarily describe as the "real" structure of the piece. (It was an unexpected "feature" that the program managed to report both explanations.) In the future, it seems essential to construct hierarchical explanations to avoid similar problems.

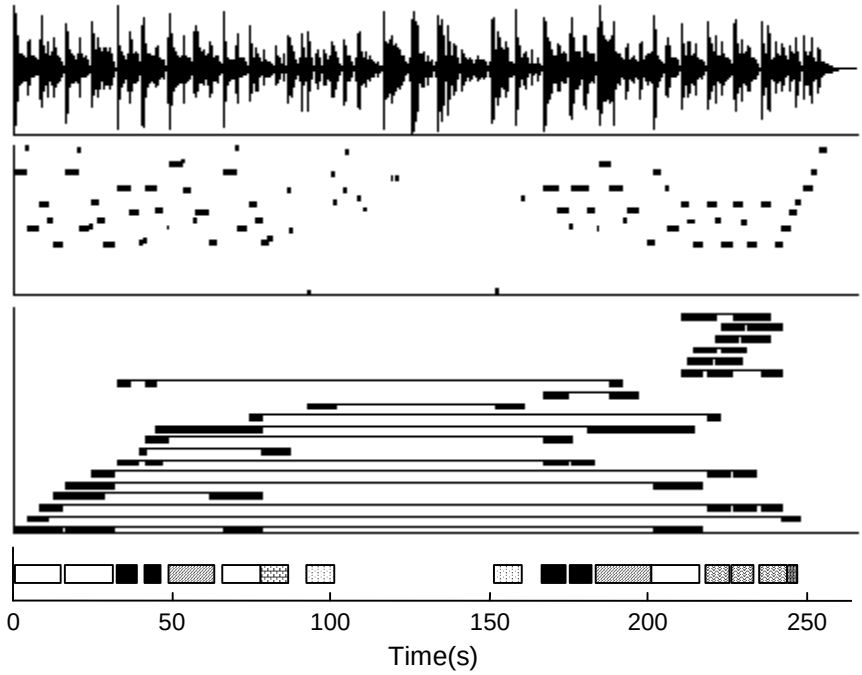

Fig. 1. Analysis of "Naima." From top to bottom: input audio waveform, transcription in "piano roll" format, clusters of similar melodic material, analysis or "explanation" of the music. In the display of clusters, the vertical axis has no meaning except to separate the clusters. Each cluster is a thin horizontal line joining the elements of the cluster indicated by heavy lines.

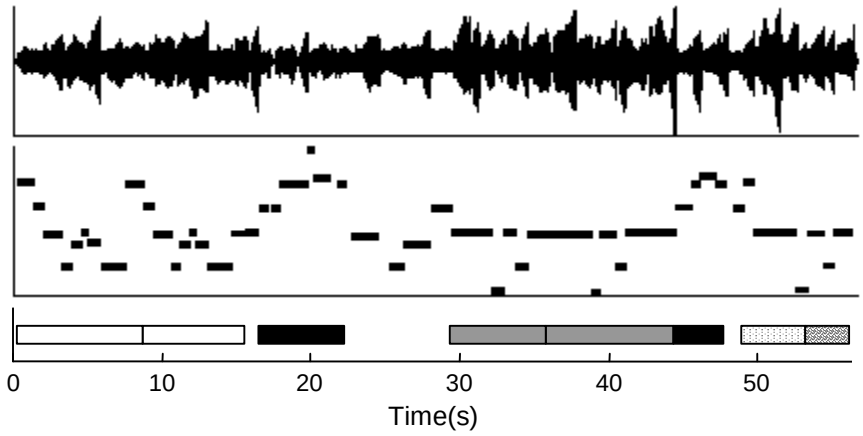

Fig. 2. Analysis of "We Three Kings." The structure is AABCDDED, but the program found some similarity between B and E. The final D section was not matched with the other D sections.

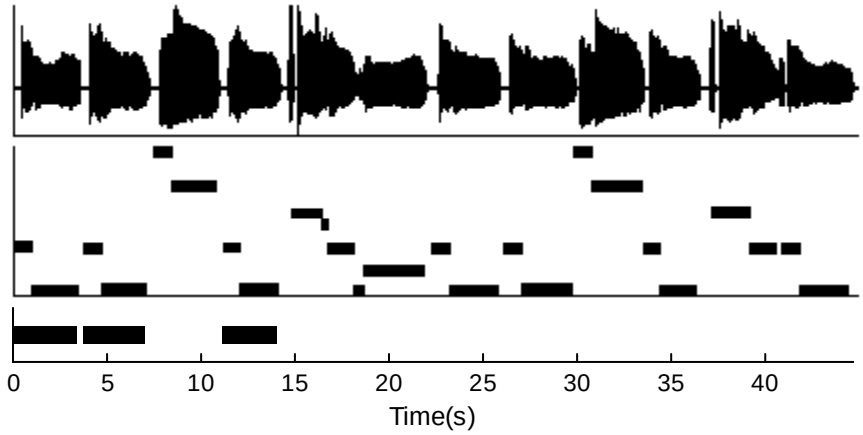

Fig. 3. Analysis of "Freddie the Freeloader," showing three occurrences of the opening figure. The program also detected the similarity between the first and second halves of the piece, but the non-hierarchical descriptions and graphics output do not capture this aspect of the structure.

4 Analysis of Polyphonic Music

In terms of transcription, music is either monophonic or polyphonic. Pitch estimation from monophonic sources is relatively simple, while polyphonic transcription is characterized by high error rates, limited success, and continuing active research. While "Naima" is certainly polyphonic music in the sense of there being a quartet of instruments, we were able to treat it as a monophonic signal dominated by the tenor saxophone lines. This approach cannot work for many types of music.

In principle, there is no need to use melodic contour or monophonic pitch estimates for music analysis. Any local attribute or set of attributes could be used to judge similarity. There are many alternatives to using monophonic transcription, although research is still needed to test and evaluate different approaches. We have begun to explore the use of spectra, and in particular, the use of a spectrum-based vector called "chroma." [20]

The hope is that we can identify similar sections of music by comparing short-term spectral content. With spectra and particularly with polyphonic material, it is unlikely that we will be able to separate the music into segments in the way that we obtained notes from pitch estimates. Instead, we take spectral frames of equal length as the units of comparison, inspired by earlier work on melodic matching.[10]

4.1 Chroma

Rather than using spectra directly, we use *chroma*, which are similar to spectra, but reduced in dimensionality to a vector of length 12. The elements of the vector represent the energy associated with each of the 12 pitch classes in an equal-tempered chromatic scale. Chroma are computed in a straightforward manner from spectra. Our choice of chroma was influenced by earlier and successful work in which chroma were used to locate recurring passages of music.

Recall in the earlier work that a similarity matrix was computed to locate similar sequences of notes. With chroma, we also construct a similarity matrix to locate similar sequences of chroma. We use chroma frames of 0.25s duration. To compute the distance between two chroma, we first normalize the chroma vector to have a mean value of zero and a standard deviation of 1. Then we compute the Euclidean distance between the two vectors.

Our goal is to find whether there is a match beginning with each pair of frames *i* and *j*. Dynamic programming, e.g. time warping algorithms, could be used to find an optimal sequence alignment starting at each pair. Since dynamic programming costs $O(n^2)$ and there are n^2 pairs, the brute force search costs $O(n^4)$, although this can be optimized to $O(n^3)$ by computing an entire row of pairs at once. A song will often be divided into hundreds of chroma frames, and the $O(n^3)$ cost is prohibitive. Instead, we use a greedy algorithm to increase the computational speed, and we compute a little less than half of the similarity matrix (the diagonal is ignored), because the matrix is symmetric. (Note that the name "similarity" may be confusing because here, increasing values represent greater distance and less similarity.)

The goal of the algorithm is to find pairs of intervals ($S_{i,j}$, $S_{m,n}$) of frames such that $S_{i,j}$ is "similar" to $S_{m,n}$. Interval similarity is based on the distance of chroma at two locations. Call chroma distance as described earlier $D(i, j)$. Define interval $S_{i,j}$ to be a range of frame numbers from *i* to *j,* inclusive. The similarity of two intervals is the quotient of a distance function $d(S_{i,j}, S_{m,n})$ and a length function $l(S_{i,j}, S_{m,n})$. The distance function is defined in terms of a path from (*i, m*) to (*j, n*). A path consists of a sequence of adjacent (diagonal, horizontal, or vertical) cells in the matrix. Given a path *P*, we can define distance d_P as the sum of the chroma distance along path *P*:

$$d_P = \sum_{(i,j) \in P} D(i, j)$$

The distance function *d* is the minimum over all paths:

$$d(S_{i,j}, S_{m,n}) = \text{argmin}_P(d_P)$$

The length function is defined as the total path length, where a diagonal step has length 1 and a horizontal or vertical step has length $\sqrt{2}/2$. This is equivalent to:

$$l(S_{i,j}, S_{m,n}) = \min(j\text{-}i, n\text{-}m) + (\sqrt{2}/2)(\max(j\text{-}i, n\text{-}m) - \min(j\text{-}i, n\text{-}m))$$

The distance *M* between two intervals is:

$$M(S_{i,j}, S_{m,n}) = d(S_{i,j}, S_{m,n}) / l(S_{i,j}, S_{m,n})$$

In general, we want to find interval pairs such that *M* falls below some threshold. In practice, there are many overlapping intervals that satisfy any reasonable threshold value, so we really only want to find large intervals, ignoring intervals that are either very similar or contained within larger similar pairs.

Thus, the algorithm we use is a heuristic algorithm that searches for the longest paths starting as promising starting points where $S(i, j)$ is already below a threshold. The algorithm computes several values for each element of a matrix: the distance function *d*, the length *l*, and starting point *p*. The matrix cells are scanned along diagonals of constant *i+j*, moving from upper left to lower right (increasing values of *i+j*). At each point, we compute *d/l* based on the cell to the left (*i, j*-1), above (*i*-1, *j*), and diagonally left and above (*i*-1, *j*-1). The cell giving the minimum distance is used to compute the new values of *d, l,* and *p* for location *i, j*. Cells are only computed if the value of *d/l* is below threshold.

This calculation identifies whole regions of ending points for a set of good starting points for matching intervals. To determine good ending points, this search process is also executed in reverse (decreasing *i+j*) and with paths going in the opposite direction, computing a second matrix. The starting points of this reverse matrix become the ending points of the forward matrix. It is easy to find the ending point corresponding to a starting point using the *p* values of the reverse matrix.

Figure 4 shows pairs obtained from an analysis of "Minuet in G."

4.2 Clusters

After computing similar segments of music in pairs, the next step is to construct clusters from the pairs. The clustering algorithm is different from the note-based clustering described earlier. When matching notes, similar sequences match at

discrete locations corresponding to note onsets, but with chroma frames, matches do not occur at clearly defined starting and ending points. Thus, the technique of scanning along a row to find all members of a cluster might not work unless we search at least several rows for near matches. The following clustering algorithm deals with near-matches in a different way.

Fig. 4. Finding pairs of similar material in Beethoven's Minuet in G. The horizontal thin lines connect elements of a pair.

To form clusters from pairs, iteratively remove clusters from the set of pairs as follows: On each iteration, remove a pair from the set of pairs (this gives the first two elements of a new cluster) and search the remaining pairs for matches. Two pairs match if one element of the first pair is approximately the same as either element of the second (recall that elements here are simply time intervals). If a match is found, add the unmatched element of the pair to the cluster and remove the pair from the set of pairs. Continue outputting new clusters until the set of pairs is empty.

This algorithm has one problem. Sometimes, there are pairs that differ in size and therefore do not match. Consider the pairs in Figure 5. Segment A matches B, which is identical to the first part of C. Since C matches D, the first part of D must also match A. It is clear that there are three segments that match A, but since element C is much larger than B, the pairs do not match. The algorithm is extended by handling this as an alternate case. Since C is larger than B but contains it, we compute which portion of C matches B, then add the corresponding portion of D to the cluster.

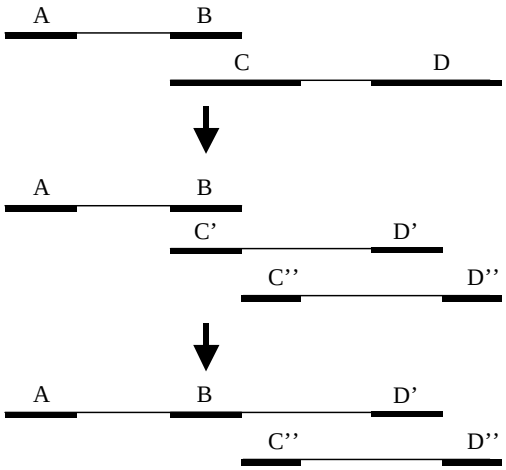

Fig. 5. When forming clusters, two pairs may share overlapping elements (B and C) that do not match because of length. To form a cluster of 3 elements that all match A, we split C to match B (which in turn matches A) and spit D in proportion to C.

Hierarchical clustering techniques might be useful in dealing with approximate matches and provide a more theoretically sound basis for clustering melodic segments. An interesting twist here is that data can be modified as in Figure 5 to form

better clusters. One might also get better results using an iterative approach where matches suggested by clustering provide evidence to be used in the signal analysis and similarity phases.

4.3 Results Using Chroma

Once clusters are computed, we can perform the explanation step as before. Figure 6 illustrates an explanation (or structural analysis) of Beethoven's Minuet in G, analyzed from an audio recording of an acoustic piano performance. The analysis clearly illustrates the structure of the piece.

Other pieces, with more variety and less exact repetition, have not produced such definitive results. For example, a pop song with a great deal of repetition is shown in Figure 7. It is encouraging that much of the repetitive structure emerged, but those repetitions are rather exact. Where the repetition includes some variation or improvisation, our system missed the similarity because even the chroma representation is quite different in each variation. This is due to changes in timbre, orchestration, and vocal improvisation. Better representations will be needed to recognize these variations in polyphonic music. Further work is also needed to deal with the fact that similar sections do not have definitive starting and ending points. When a segment in one cluster overlaps a segment in another cluster, it is not clear whether the two segments represent the same phrase of music or whether one is a sub-phrase of the other. This has implications for the structure of the music.

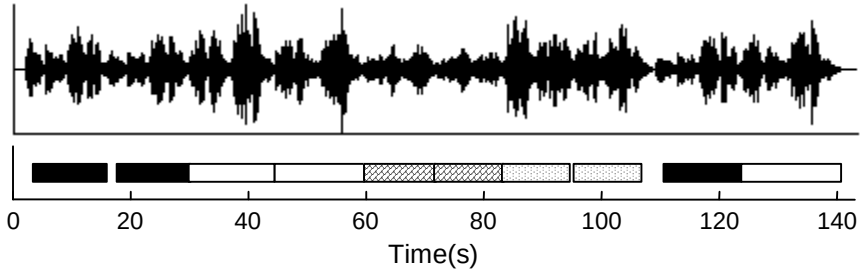

Fig. 6. Analysis of Beethoven's Minuet in G is shown below the input waveform. Notice the clear analysis at the bottom: AABBCCDDAB.

5 Discussion

It seems quite clear that an important aspect of music listening is to identify relationships and structure within the music. The most basic, salient, and common relationship is the repetition of a phrase. Our goal is to build automated "music listeners" that identify musical structure by finding relationships. There are several motivations behind this work. First, building machines that mimic human behavior is intrinsically interesting. As music understanding seems to be a uniquely human experience, automating aspects of this skill is particularly fascinating. Second, we can learn more about music and about music perception by studying listening models. Our models are not intended as faithful perceptual models, but they can still tell us something about structure, coding, redundancy, and information content in musical signals. For example, this work sheds some light on whether general musical knowledge is required to understand musical structure, or is the structure implied by the music itself? Third, our original motivation was the problem of music meta-data creation. Audio databases have very little structure and there is very little that can be searched unless the audio is augmented with descriptions of various kinds. Text fields naming composers, performers, titles, instrumentation, etc. are a standard way to

make audio searchable, but there are many "musical" aspects of music that cannot easily be described by text. A program that "listens" to music seems much more likely to find attributes that are of interest to humans searching a database. Attributes including common themes, salient or surprising chord progressions, and overall form might be derived automatically. These attributes should also be useful to obtain or validate other information about tempo, genre, orchestration, etc.

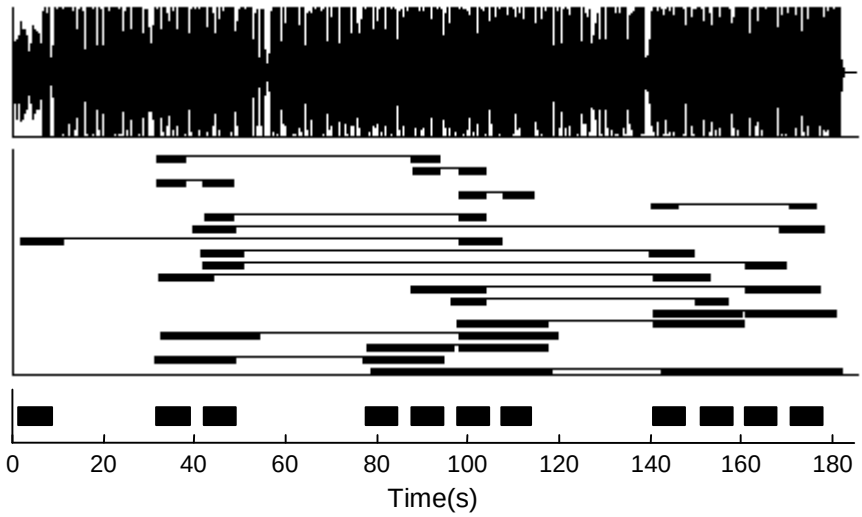

Fig. 7. Analysis of a pop song. There is frequent repetition of a short pattern within the song as shown in the analysis. Note the ambiguity of starting and ending times in the pairs (middle of figure). Our clustering and analysis failed to find much structure other than many repetitions of one motive. For example, the recording has a melody from about 10 to 20s and a variation at approximately 50 to 70s.

The significance of our work is that we have demonstrated through examples how an automated process can find pattern and structure in music. We believe that a primary activity in music listening is the identification of pattern and structure, and that this program therefore exhibits a primitive form of music listening. The input to the program is audio, so there is no "hidden" information provided through symbolic music notation, pre-segmented MIDI representation, or even monophonic audio where sources are cleanly separated.

The quality of our analyses is variable. We suspect that is true for human listeners as well, but we have no metric for difficulty or experimental data on human listeners. Furthermore, the "correct" analysis is often ambiguous. There is certainly room for much experimentation in this area. Given the inherently analog and "noisy" nature of audio input, we are pleasantly surprised that the analysis of at least some music corresponds so clearly with our intuition about what is correct.

Although some form of quantitative evaluation of our approach might be interesting, our goal thus far has been to identify techniques that show promise and to demonstrate that some analysis procedure can actually succeed using audio input. As indicated in Section 3.6, we performed a successful analysis of "Naima" after tuning various parameters and refining our algorithms. We also successfully analyzed two other pieces with no further tuning to demonstrate that the algorithms are robust enough to handle different pieces and instruments. On the other hand, we have run into difficulties with polyphonic transcription data, and it is clear that our techniques are not general enough for a wide spectrum of musical styles. Rather than evaluate this early stage of research, we believe it is more appropriate to experiment with different techniques and compare them.

In the future, we plan to look at other techniques for analysis. Polyphonic transcription is unable to *reliably* transcribe typical audio recordings, but the results

seem good enough to determine if passages are similar or dissimilar, and the discrete nature of the output might be easier to work with than continuous spectra. Intermediate levels of transcription, such as Goto's [7] work on bass and melody extraction could also be used.

Another direction for future work is a better theoretical model of the problem. How do we know when an "explanation" is a good one, and how do we evaluate ambiguous choices? Ideally, we would like a probabilistic framework in which evidence for or against different explanations could be integrated according to a sound (no pun intended) theory.

So far, we have only looked for repetition. In many of our examples, there are instances of transposition and other relationships. These are also important to music understanding and should be identified. It appears that these relationships are often less redundant than outright repetition. After all, transposition can occur at different intervals, and transposition may be tonal or chromatic. Also, transpositions often involve just two or three notes in sequence, i.e. one or two intervals. A good model will be needed to decide when a descending second is a "motive" and when it is just a descending second. The distinction often depends upon rhythmic and harmonic context, which means more sophisticated analysis is required to detect transpositional structure. It would also be interesting to look for rhythmic motives, as in the work of Mont-Reynaud. [12] This is an area for further research.

6 Conclusions

We have shown how automated systems can listen to music in audio form and determine structure by finding repeated patterns. This work involves the conversion of audio into an intermediate form that allows comparisons. We have used monophonic pitch estimation and transcription as well as spectral information as an intermediate representation. With transcription, the intermediate form is a sequence of notes, and we developed a note-based pattern discovery algorithm to find similar subsequences. With spectra, the intermediate form is a sequence of frames of equal duration, and the pattern search uses a "ridge following" algorithm to find correspondences in the music starting at two different points in time. In both cases, pairs of music segments found to be similar are formed into clusters. These clusters represent patterns that recur in the music. The music is then "explained" by identifying how the music can be constructed from a sequence of these patterns. In spite of the fact that the music is originally in audio form, complete with noise, reverberation, drums, and other complications, our algorithms are able to do a very good job of extracting structure in many cases. Familiar structures such as AABA and AABBCC… are found in our examples. The structural description also reports the location of the patterns, so it is a simple matter for a user to locate all occurrences of the "A" pattern, for example.

Our results so far are quite encouraging. Many inputs are analyzed almost without any error. As would be expected, there are plenty of examples where analysis is not so simple. In the future, we hope to develop better theoretical models, refine our analysis techniques, and identify other relationships such as rhythmic patterns and transpositions.

7 Acknowledgements

This work is supported in part the National Science Foundation, award number #0085945.

8 References

[1] Bartsch, M. and Wakefield, G.H., 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*, (2001), IEEE.

[2] Birmingham, W.P., Dannenberg, R.B., Wakefield, G.H., Bartsch, M., Bykowski, D., Mazzoni, D., Meek, C., Mellody, M. and Rand, W., MUSART: Music Retrieval Via Aural Queries. in *International Symposium on Music Information Retrieval*, (Bloomington, Indiana, 2001), 73-81.

[3] Coltrane, J. Naima *Giant Steps*, Atlantic Records, 1960.

[4] Conklin, D. and Anagnostopoulou, C., Representation and Discovery of Multiple Viewpoint Patterns. in *Proceedings of the 2001 International Computer Music Conference*, (2001), International Computer Music Association, 479-485.

[5] Cope, D. *Experiments in Musical Intelligence*. A-R Editions, Inc., Madison, Wisconsin, 1996.

[6] Foote, J. and Cooper, M., Visualizing Musical Structure and Rhythm via Self-Similarity. in *Proceedings of the 2001 International Computer Music Conference*, (Havana, Cuba, 2001), International Computer Music Association, 419-422.

[7] Goto, M., A Predominant-F0 Estimation Method for CD Recordings: MAP Estimation using EM Algorithm for Adaptive Tone Models. in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, (2001), IEEE, V-3365-3368.

[8] Lartillot, O., Dubnov, S., Assayag, G. and Bejerano, G., Automatic Modeling of Musical Style. in *Proceedings of the 2001 International Computer Music Conference*, (2001), International Computer Music Association, 447-454.

[9] Leyton, M. *A Generative Theory of Shape*. Springer, Berlin, 2001.

[10] Mazzoni, D. and Dannenberg, R.B., Melody Matching Directly From Audio. in *2nd Annual International Symposium on Music Information Retrieval*, (2001), Indiana University, 17-18.

[11] Mongeau, M. and Sankoff, D. Comparison of Musical Sequences. in Hewlett, W. and Selfridge-Field, E. eds. *Melodic Similarity Concepts, Procedures, and Applications*, MIT Press, Cambridge, 1990.

[12] Mont-Reynaud, B. and Goldstein, M., On Finding Rhythmic Patterns in Musical Lines. in *Proceedings of the International Computer Music Conference 1985*, (Vancouver, 1985), International Computer Music Association, 391-397.

[13] Narmour, E. Music Expectation by Cognitive Rule-Mapping. *Music Perception*, *17* (3). 329-398.

[14] Rabiner, L. On the use of autocorrelation analysis for pitch detection. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *ASSP-25* (1). 24-33.

[15] Roads, C. Autocorrelation Pitch Detection. in *The Computer Music Tutorial*, MIT Press, 1996, 509-511.

[16] Rodet, X. and Jaillet, F., Detection and Modeling of Fast Attack Transients. in *Proceedings of the 2001 International Computer Music Conference*, (2001), International Computer Music Association, 30-33.

[17] Rolland, P.-Y. and Ganascia, J.-G. Musical pattern extraction and similarity assessment. in Miranda, E. ed. *Readings in Music and Artificial Intelligence*, Harwood Academic Publishers, 2000, 115-144.

[18] Simon, H.A. and Sumner, R.K. Pattern in Music. in Kleinmuntz, B. ed. *Formal Representation of Human Judgment*, Wiley, New York, 1968.

[19] Stammen, D. and Pennycook, B., Real-Time Recognition of Melodic Fragments Using the Dynamic Timewarp Algorithm. in *Proceedings of the 1993 International Computer Music Conference*, (Tokyo, 1993), International Computer Music Association, 232-235.

[20] Wakefield, G.H., Mathematical Representation of Joint Time-Chroma Distributions. in *International Symposium on Optical Science, Engineering, and Instrumentation, SPIE'99*, (Denver, 1999).