Music Understanding by Computer[1]

Roger B. Dannenberg

ABSTRACT

Although computer systems have found widespread application in music production, there remains a gap between the characteristicly precise and mechanical information processing of the computer and the more subtle and expressive processing performed by humans. In order for computer systems to become more useful to musicians, the human-computer interface must be raised from the mechanical and syntactic level of present systems toward a high-level concept-based dialog. The term ‘‘music understanding’’ is introduced to describe the recognition of pattern and structure in musical information. Applications of music understanding include music transcription and music performance systems. The general state of the art in music understanding and two specific interactive real-time performance systems are described.

1. Introduction

Computers are used in many ways to enhance music composition and performance. In many cases, computers can be used to make traditional music more efficiently. In this role, computers can edit, store, print, transmit and receive musical scores, control music synthesizers, synthesize musical tones, control audio mixers and processors, and operate audio and video tape recorders. Virtually every aspect of traditional music making and recording has been automated to some extent.

One of the most common applications of computers in music is the use of the computer as a controller for a music synthesizer. In this application, the computer specifies notes to be played by a synthesizer, very much as a piano-roll mechanism controls a player piano. The advantage of the computer is the flexibility with which control information can be entered, edited, stored and even electronically transmitted. Unfortunately, the use of computer-controlled synthesizers in place of human performers often seems to sacrifice human and esthetic qualities for expedience and efficiency.

[1]Published as: Dannenberg, ‘‘Music Understanding,’’ ***1987/1988 Computer Science Research Review***, Carnegie Mellon School of Computer Science, pp. 19-28.

Is this an intrinsic problem with computer systems? Since stored control information can be edited with arbitrary precision, the only intrinsic limits of musicality are those of the composer, and this freedom to specify with precision is what attracts many composers to work with computers. Unfortunately, detailed specification of control information can be quite time-consuming, and there seems to be a choice between economy of production and the subtlety or artistic qualities of the performance. It seems clear that today's computer music systems are quite primitive, and there is room for much improvement.

There are many avenues that lead to extended capabilities of computers in music. One possible direction is toward improved user interfaces, perhaps incorporating more musical knowledge to raise the level of abstractions that the musician has to deal with. Another direction is toward musical interpretation of information to make up for the fact that there may be no human performer. Finally, one might explore how to use computer music systems in live performances.

All of these directions require a more sophisticated approach to music than that provided by ''player piano'' systems. In particular, some degree of musical knowledge is required to work with concepts like phrasing and texture, concepts which are not supported by current systems. To obtain a higher degree of sophistication, computer music systems must be able to understand aspects of the musical information being processed. This line of reasoning has led to a new area of study which I call ''music understanding'', the study of methods by which computer music systems can recognize pattern and structure in musical information.

The purpose of this report is three-fold: to examine the potential of music understanding, to survey the state of the art, and to report on recent progress in the field at Carnegie Mellon. In the next section, various forms of music understanding will be discussed. After that, details of two music understanding projects at Carnegie Mellon will be presented. The goal of the first project is to build a competent computer accompaniment system, and the goal of the second is to follow an improvisation in real time. Many other tasks have been pursued or are current research topics [13, 9, 7, 1], but space limitations prohibit an exhaustive treatment of this related activity.

2. Music Understanding Tasks

There are a variety of tasks that one might perform with a music understanding system. Tasks can be categorized according to whether they are accomplished in real-time during a performance, or not in real-time using recorded performance information. Non-real-time systems have the advantage of being able to examine a piece of music from beginning to end before drawing any conclusions, while real-time systems must output results while the performance is in progress. Music understanding tasks can also be categorized according to whether the input is an audio signal or a sequence of note descriptors. As described below, the conversion from audio signals to the note level of abstraction is itself a music understanding task.

2.1. Signal Processing and Music Understanding

A number of music understanding tasks involve processing sound directly to extract various higher-level attributes. One such task is pitch recognition. Since pitch is a primary dimension of most musical sounds, it is important to be able to automate this task. There are a variety of techniques for pitch recognition, and some operate in real time (with a slight delay). Most techniques assume that the input is a signal consisting of a single musical tone. The analysis of multiple (polyphonic) tones to derive a set of pitches is much more difficult and forms an area of active research. There are no general purpose real-time polyphonic pitch recognition systems, but some non-real-time systems have obtained limited success [10, 4]. Electronic keyboards can be interfaced directly to computers so that performance information can be sensed without signal processing. Given the state of the art in polyphonic pitch recognition, it is no surprise that keyboards form the most common computer interface for the musician.

Pitch recognition for musical applications is usually accomplished in two steps. First, a time-varying ‘‘instantaneous’’ pitch is determined, and then the pitch is rounded to the nearest scale step. In traditional Western music, there is an approximately 6% frequency change between adjacent scale steps. Intentional and unintentional variations are to be expected, but these are usually within 3%, so simple rounding can be used. Vocal music presents a special challenge because vocalists commonly deviate more than 3% from the nominal frequency of the sung pitch. This occurs during the onset of the sung note and also when vibrato is used. Although good techniques exist for tracking the fundamental periodicity of speech sounds, there are few if any studies on identifying musical pitches produced vocally.

Another signal processing task is the recognition of note onsets, that is, the beginnings of notes. This task is difficult because the beginning of a note is often masked by the decaying sound of the previous note and by room reverberation. In polyphonic music, onsets may be masked by other notes. Also, slight variation due to air or bow pressure can give the effect of a note onset unintentionally. Detecting note onsets reliably is still an area of ongoing research [8, 12].

Another area in which human listeners are skilled is sound source location. This task is complicated by the presence of sound reflections and reverberation in typical listening spaces, and little if any research has been done to accomplish this task, although the psychoacoustics of human sound source location is an area of active research. Other interesting tasks are the recognition of instrument type, for example detecting the difference between an oboe and a clarinet, and the recognition of performer, for example detecting the difference between Dizzy Gillespie and Miles Davis. The latter is likely to be accomplished by a combination of signal processing and analysis at higher levels of abstraction. To date, instrument and performer recognition have not been studied. Presumably, researchers who would do this work are devoting their energies to polyphonic pitch recognition and psychoacoustics.

2.2. Music Understanding for Transcription

A long-standing goal of many researchers has been the creation of a music transcription system that accepts audio input and produces printed music notation as output. There are actually many subtasks involved in music transcription, including the recognition of beat, pitch, key, meter, phrase, dynamics, and articulation.

Beat recognition is the detection of the pulse or beat by which musical time intervals are measured. This is somewhat similar to pitch recognition in that the goal is to find periodicity. Pitch periodicity is on the order of 1 to 10ms, while beats last on the order of 200ms to 2s. Detecting and following beats is difficult because (1) there can be a fair amount of variation in the timing of note onsets with respect to the beat, (2) beats can be subdivided in various ways, and (3) note onsets often occur off the beats in order to add musical variety.

In addition to recognizing pitch, which was discussed above, a music transcription system must recognize keys. In traditional tonal music, only a subset of the 12 possible pitches per octave are emphasized. This pitch subset, combined with a distinguished pitch called the *tonic*, is called a *key*. Keys are notated, and therefore key recognition is an important aspect of transcription.

Meter refers to the organization of beats into a hierarchy. For example, 6/8 meter organizes beats into a repeated pattern of 2 groups of 3 illustrated by the sequence ‘‘311211311211...’’. Meter is also notated and reflects important structural information about the music.

Phrases are sequences of notes that are heard as a unit. Typically, a melody or tune is constructed from a series of phrases, and phrases are separated by a slight pause, an important change in harmony, or an important metrical boundary. Phrases are often but not always indicated in music notation.

Dynamics in music refers to changes in loudness that occur primarily at the note and phrase level. Articulation generally refers to variation in loudness and other effects within a given note. For example, *forte*, a dynamic notation, means to play loudly, while *sforzando*, an articulation, means to accent the beginning of a note by playing it loudly. Recognizing dynamic levels and articulation is difficult and requires a deep understanding of the music being notated as well as signal processing to extract the loudness contours of the music to be notated.

It should now be apparent that music transcription involves music understanding along many dimensions. Commercial music transcription systems avoid many of these dimensions. For example, beat and meter recognition is avoided by having the user explicitly give the system a tempo and meter. The transcription program then outputs an audible pulse to which the user must synchronize his performance. Alternatively, at least one system allows the user to tap beats on a pedal as he or she performs. Key changes are typically entered manually and little or no effort is made to notate phrases, articulation, or dynamics. In spite of these simplifications, transcription programs cannot reliably distinguish between performed triplets and sixteenth notes.

An ambitious music transcription program was described by Chafe, et. al. [3]. This system used musical knowledge to achieve a much higher level of performance than that of commercial systems described above. Particular emphasis was placed on completely automating the transcription process, and interesting techniques were developed for recognizing beats, meter, and keys.

The tasks described here have many applications outside of music transcription. For example, the study of ethnic music and folk songs might be considerably aided by systems that can recognize rhythms and phrases. Music understanding at the level of music transcription is also the first step toward harmonization, arranging, and orchestration. While these tasks are usually described as creative tasks rather than understanding tasks, it is still necessary to understand the structure of a given melody or other materials in order to use them creatively.

2.3. Music Understanding in Performance

While transcription is normally considered a non-real-time task, music understanding is also important in real-time music performance tasks. The primary application is in situations where computer systems and human performers are simultaneously and interactively performing music.

One such task is called *computer accompaniment*. In this task, a computer system is provided with a machine-readable score that contains music to be played by an ensemble consisting of one or more human players. The score also contains music to be performed by the computer. The task is to listen to the live performance by human musicians and play along in synchrony. The task is complicated by the fact that the human performance will vary somewhat from the score in that human performers might vary tempo and articulation according to acoustical conditions and the response from an audience. Also, both humans and pitch detectors make mistakes.

Accompaniment systems can be classified according to their inputs. A *monophonic* accompaniment system follows an input performance that consists of a sequence of single non-simultaneous notes. A *polyphonic* accompaniment system follows an input performance that consists of a sequence of single or compound (simultaneous) notes. (Simultaneous notes are usually called *chords*.) Polyphonic input is difficult to handle in that the order in which simultaneous notes are detected may vary from one performance to the next, and it can be ambiguous whether a group of notes that are detected in rapid succession are intended to be a chord or a sequence of individual notes. An *ensemble* accompaniment system accepts input from multiple human players. The difficulty here is that the players may be not be synchronized among themselves. The state of the art in computer accompaniment will be discussed in detail in the next section.

Computer accompaniment systems are intended to deal with music in which all notes are notated in advance. Music is often improvised, however, and pitch sequences are not available in advance. Even fairly conventional melodies are often played with considerable leeway in terms of added ornamentation and variation. For this reason, there is interest in music understanding systems that deal with improvised music. One such system constructed by the author and Bernard Mont-Reynaud is discussed in Section 4.

Even when music is improvised, there is still generally some organization, structure and pattern which can be recognized. Music understanding systems can be classified according to what sort of input they accept and what sort of structure they look for. As with accompaniment systems, input can be monophonic or polyphonic from a single musician, or the input may be obtained from an ensemble. (One could subdivide ensemble input according to whether each performer produces monophonic or polyphonic music, but this distinction seems relatively unimportant.) The structure that a music understanding system looks for can be of many types. Some interesting structures include beat and meter detection (as in the transcription task), recognition of chord sequences (either recognizing a known sequence or discovering arbitrary patterns), recognition of keys, scales, and pitch organization, identification of phrases, and recognition of large forms (for example, recognizing the climax of a composition).

No system today performs any of these tasks well. Some beat detection systems are in commercial use, but these devices seem to be effective only when the input is very simple. Various systems have been constructed to interact with and respond to performers in real-time. The response is often a function of pitch and timing, but the level of pattern recognition and understanding is generally low. Recently, several interactive performance software packages of this type have appeared in the commercial marketplace.

Although the field of music understanding appears to be in its infancy, one can imagine many future applications. Some of these are described in the concluding section. The next section will present a detailed look at techniques for computer accompaniment.

3. Computer Accompaniment

As described above, the goal of computer accompaniment is to follow a live performance and synchronize a computer performance according to a precomposed score. Computer accompaniment systems are implemented as several cooperating subtasks that handle various aspects of accompaniment. Information flow is generally in one direction from each subtask to the next, and the separation into subtasks greatly simplifies the system design. The tasks are *input processing*, *matching*, *accompaniment performance*, and *music synthesis* (see figure 3-1). These will be described in sequence.

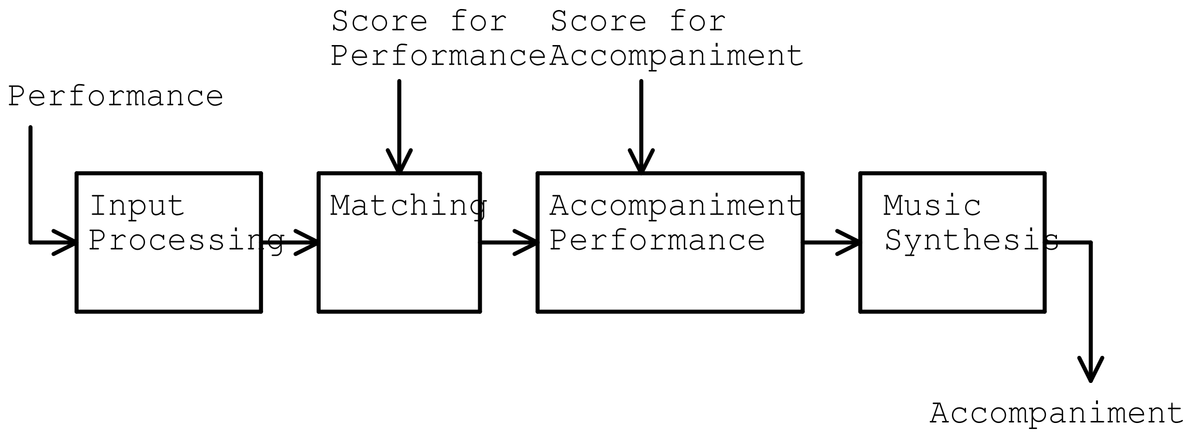

Figure 3-1: Block diagram of a computer accompaniment system.

3.1. Input Processing

The input processing subtask takes the human performance as its input. Various transformations are performed on this input to produce an abstract symbolic representation of the input for use by the matching subtask. This representation consists of the pitch and onset time of each note.

If the input is audio, then the first step is to track the pitch of the input. At present, this is only possible with monophonic input. The pitch is tracked continuously and quantized to the nearest scale step. Note onsets are assumed to occur when either the nearest scale step changes or when the signal crosses some energy threshold in the positive direction.

Various heuristics can be applied to raw signal processing data in an attempt to increase the reliability of the information. For example, the input processor -- making pitch readings hundreds of times per second -- might not report a pitch change until it obtains three consecutive identical pitch readings. In cases where onset time detection is error-prone, it is sometimes profitable to filter out all consecutive repetitions of a pitch. In effect, a pitch-change detector is substituted for an onset detector. In many cases, the reduction in information is made up for by its reliability.

3.2. Matching

The *matching* subtask receives input from the input processor and attempts to find a correspondence between the real-time performance and the score. The matcher loads the entire score before the performance begins. Then, as each note is reported by the input processor, the matcher looks for a corresponding note in the score. Whenever a match is found, it is reported to the accompaniment performance subtask. The information needed by the accompaniment performance subtask is just the real-time occurence of the note performed by the human and the designated time of the note according to the score.

The matcher must meet several criteria. First, it must be a semi-on-line algorithm; that is, it is allowed to look ahead in the score, but it must process performance information as it is received. Second, it must be a real-time algorithm: the processing time allowed for each note must be small and bounded by a constant so that the matcher can keep up with the performance. Third, the matcher must be tolerant of variations in timing as well as wrong notes, extra notes, and omitted notes. Errors will inevitably occur in the performance as well as in the input processor. Fourth, the matcher must have a low rate of false-positive outputs, although the rate of false-negatives can be high. In other words, the matcher should only report matches when there is a very high probability of an actual match.

Since the matcher must be tolerant of timing variations (in fact, the whole purpose of accompaniment is to synchronize in the face of timing variation), matching is performed on sequences of pitches only. This decision makes the matcher completely time-independent. One problem raised by this pitch-only approach is that each pitch is likely to occur many times in a composition. A typical melody might have 170 notes, but only 13 distinct pitches. The most frequent pitch might occur 32 times, or almost one out of every five notes. Whenever this note is played there will be many candidates for a match.

Fortunately, an almost ideal matcher has been developed for computer accompaniment [5]. The matcher is derived from the dynamic programming algorithm for finding the longest common subsequence of two strings. Imagine starting with two strings and eliminating arbitrary characters from each string until the the remaining characters (subsequences) match exactly. If these strings represent the performance and score, respectively, then a common subsequence represents a potential correspondence between performed notes and the score (see Figure 3-2.) If we assume that most of the score will be performed correctly, then the longest possible common subsequence should be close to the ‘‘true’’ correspondence between performance and score. This is merely a heuristic, but it works very well in practice.

The longest common subsequence (LCS) algorithm [11] operates by solving the problem on all initial substring pairings of the two input strings. Let P and S be the input strings, and let $P_{<1,c>}$ represent the first r elements of string P. The LCS algorithm constructs a two-dimensional array L in which each element $L_{r,c}$ represents the length of the LCS of $P_{<1,c>}$ and $S_{<1,r>}$. It can be shown (ignoring boundary conditions) that if S_r *does not match* P_c then $L_{r,c} = max(L_{r-1,c}, L_{r,c-1})$

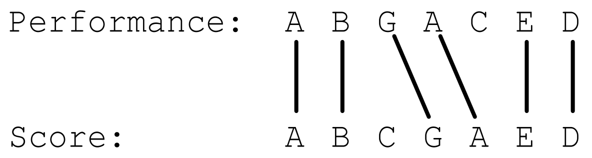

Figure 3-2: The correspondence between a score and a performance.

and if S_r *matches* P_c then $L_{r,c} = L_{r-1,c-1} + 1$. Notice that each cell in the array L can be computed in terms of the neighboring cells in the previous row and column. This allows the algorithm to run in time proportional to the product of the lengths of P and S. Figure 3-3 illustrates the array constructed from two strings.

Score:	Performance: A	B	G	A	C	E	D
A	1	1	1	1	1	1	1
B	1	2	2	2	2	2	2
C	1	2	2	2	3	3	3
G	1	2	3	3	3	3	3
A	1	2	3	4	4	4	4
E	1	2	3	4	4	5	5
D	1	2	3	4	4	5	6

Figure 3-3: Computing the length of the longest common substring.

The LCS algorithm is not on-line; the longest common subsequence is computed by working backward from the lower right cell of the completed array to construct one or more longest common substrings. The first modification for computer accompaniment is to change the nature of the output. What we want to know is not necessarily the full LCS, but to which element of *S* (the score) does each element of *P* (the performance) correspond, if any. A good guess is sufficient, since the optimal guess depends upon the future of the performance. A good heuristic is to report the element of *S* whenever a new value of *L* is computed which is larger than any previously computed values of *L*. (Computation order is important here: the array is computed one column at a time and columns are computed from the first row to the last.) Figure 3-4 has underlined the cells corresponding to matches that would be reported by this heuristic.

The modified LCS algorithm is now on-line, but not real-time, because one entire column of *L* must be computed for each input symbol of *P*. The height of the column is the size of the score. To avoid a potentially long computation, only a portion of *L*, called a window, is computed. The window is centered around the row of the expected corresponding score note. Even though most of the array is not computed, identical results will be obtained as long as the ultimate LCS is contained within the window. Figure 3-5 illustrates the use of windows. The window is centered around the last match and advanced by one row per column until the next match is found. For the sake of illustration, a very small window is used; typical window sizes range from 10 to 40 notes.

One more modification is necessary for use in computer accompaniment. With the algorithm

$$\begin{array}{ll|ccccccc}
 & \texttt{Performance:} & \texttt{A} & \texttt{B} & \texttt{G} & \texttt{A} & \texttt{C} & \texttt{E} & \texttt{D} \\
\hline
\texttt{Score:} & & & & & & & & \\
 & \texttt{A} & \underline{1} & 1 & 1 & 1 & 1 & 1 & 1 \\
 & \texttt{B} & 1 & \underline{2} & 2 & 2 & 2 & 2 & 2 \\
 & \texttt{C} & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\
 & \texttt{G} & 1 & 2 & \underline{3} & 3 & 3 & 3 & 3 \\
 & \texttt{A} & 1 & 2 & 3 & \underline{4} & 4 & 4 & 4 \\
 & \texttt{E} & 1 & 2 & 3 & 4 & 4 & \underline{5} & 5 \\
 & \texttt{D} & 1 & 2 & 3 & 4 & 4 & 5 & \underline{6}
\end{array}$$

Figure 3-4: In the on-line version, matches are reported when a new maximum is found as shown by underlining.

$$\begin{array}{ll|ccccccc}
 & \texttt{Performance:} & \texttt{A} & \texttt{B} & \texttt{G} & \texttt{A} & \texttt{C} & \texttt{E} & \texttt{D} \\
\hline
\texttt{Score:} & & & & & & & & \\
 & \texttt{A} & \underline{1} & 1 & & & & & \\
 & \texttt{B} & 1 & \underline{2} & 2 & & & & \\
 & \texttt{C} & 1 & 2 & 2 & & & & \\
 & \texttt{G} & & & \underline{3} & 3 & & & \\
 & \texttt{A} & & & & \underline{4} & 4 & 4 & 4 \\
 & \texttt{E} & & & & 4 & 4 & \underline{5} & 5 \\
 & \texttt{D} & & & & & 4 & 5 & \underline{6}
\end{array}$$

Figure 3-5: In the on-line version, matches are reported when a new maximum is found as shown by underlining.

as presented so far, a strange behavior can take place when a wrong note is played. If the wrong note matches some future note in the score but still within the window, then the note will be reported as matching by the heuristic discussed above. Without knowledge of the future, it is impossible to say whether the performer has played a wrong note or skipped ahead in the score. In practice, it is rare to skip ahead, so the heuristic makes many mistakes. This is particularly problematic because if one looks far enough ahead, one is almost sure to find a match. In the LCS algorithm, these ‘‘false’’ matches are not a problem because eventually a better match is found, and the ‘‘false’’ match is disregarded. It is for this reason that LCS is not an on-line algorithm. For computer accompaniment, however, this problem must be solved differently in order to retain the on-line property.

The solution lies in the recognition that the LCS algorithm is a particular instance of a more general algorithm for finding a string that optimizes an evaluation function. In LCS, the evaluation function is the length of the common substring, but for computer accompaniment, we can modify the function to compute length minus the number of skipped score notes. The new equations for L are: if S_r *does not match* P_c then $L_{r,c} = max(L_{r-1,c} - 1, L_{r,c-1})$ and if S_r *matches* P_c then $L_{r,c} = L_{r-1,c-1} + 1$ Other functions have been considered, but this one seems to work best in practice.

The matcher is an interesting combination of algorithm design, use of heuristics, and outright ad-hoc decisions. Much of the challenge in designing the matcher was to model the matching problem in such a way that good results could be obtained efficiently. This involved compromises in both the musical objective and the formalism.

Polyphonic matchers have also been explored. One approach is to group individual notes that occur approximately simultaneously into structures called *compound events*. A single isolated note is considered to be a degenerate form of compound event. Now by modifying the definition of ‘‘matches’’, the monophonic matcher can be used to find a correspondence between two sequences of compound events. Another approach processes each incoming performance event as it occurs with no regard to its timing relationship to other performed notes. It is important in this case to allow notes within a chord (compound event) in the score to arrive in any order. (Note that the LCS algorithm disallows reordering.) To allow partial reordering, each row of the array L represents a compound score event, while each column represents a single performed note. The resulting algorithm is time-independent. This study was performed with Joshua Bloch [2] and the reader is referred to our paper for further details.

The matcher performs a fairly low-level recognition task where efficiency is important and relatively little knowledge is required. When matches are found, they are reported to the accompaniment performance subtask, which uses knowledge about musical performance to control a synthesizer.

3.3. Accompaniment Performance

At first, one might think that accompaniment is a simple task once the matcher is present to provide the current location in the score. In fact, things are not so simple because (1) the matcher may not find or report all matches, (2) accompaniment notes may occur in between matched notes, so accompaniment is not simply a matter of waiting for accompaniment notes to be triggered by the matcher, and (3) the accompaniment must deal with match reports that indicate the accompaniment is not presently playing at the proper score location or at the correct tempo.

In order to deal with the problems listed above, the accompaniment performance subtask operates fairly autonomously from the matcher. In essence, the accompaniment advances through the score at a fixed rate, playing the notes it encounters until it receives a message from the matcher. When the message arrives, the accompaniment performer will generally be either ahead or behind the human soloist. The accompaniment performer chooses a strategy to handle the discrepancy and carries out the strategy until the next message from the matcher.

Playing the score at a fixed rate does not mean playing the score at a fixed tempo. In fact, the music is not encoded with respect to beats but in absolute time units. Thus, tempo changes can be an integral part of the score and invisible to the accompaniment performer. The accompaniment performer assumes that variations between the score and the actual performance will be such that a locally linear mapping exists from score time to real time. The accompaniment performer must constantly reevaluate the parameters of this mapping to keep the accompaniment in synchrony with the soloist.

The accompaniment performer has two primary concerns. The first is calculating a good value for the accompaniment rate, the ratio of real time to score time. The second is moving to the correct score location in a musically appropriate way.

To compute the rate, a history of recent matches is maintained. Recall that the matcher reports the real time and the score time of each match, exactly the data needed to calculate a best fitting linear approximation. The slope of the best fit line is used as the rate. To maintain robustness, the rate is only changed when the data seem reliable, and several heuristics are used to judge reliability.

The second concern of the accompaniment performer is adjustment to errors in score location. For example, if the matcher reports that the current score time should be 0.3s ahead of the current location of the accompaniment, what should be done? One approach is to increase the rate so as to catch up with the soloist. Another is to jump ahead to catch up immediately. The best strategy depends upon the magnitude of the time difference. If the time difference is very small, then the difference can be attributed to performance interpretation and no adjustment is necessary. Small errors are ultimately corrected by the rate adjustment described above. If the error is significant but still small, say less than 1s, a reasonable approach is to perform the accompaniment very rapidly to catch up or to simply stop the accompaniment and allow the soloist to catch up to the accompaniment. This maintains important musical continuity in the accompaniment. Finally, if the error is large, it is unmusical either to run ahead at high speed or to stop for a long time. In this case, the best strategy is to skip forward or backward in the score. These strategies are encoded as a set of condition-action style rules in the accompaniment performance subtask. More elaborate rules are certainly possible, but these give plausible behavior over a wide range of circumstances.

3.4. Synthesis

The accompaniment performance subtask deals with music at the control level. The output of this subtask is a stream of high-level, real-time commands which specify when to start and stop notes. The commands can also specify many details such as loudness, timbre, vibrato, and articulation if desired. The purpose of the synthesis subtask is to produce sounds according to the commands. This can be achieved through digital synthesis or by mechanically controlling an acoustic instrument such as a piano. The widespread use of a common command interface for music, called MIDI, greatly simplifies this task, and the synthesis subtask is reduced to the translation of commands into MIDI commands.

3.5. Enhancements

The accompaniment system described here has been implemented on several machines including microprocessors and personal computers. In one system, score following is used to display music notation and turn pages appropriately in addition to performing an accompaniment. One can imagine computer accompaniment being used to control other computer graphics and lighting systems, selecting organ stops, and even silencing or correcting wrong notes.

Recently, several enhancements have been made to the accompaniment system described above in order to handle a wider range of input and to improve the matcher. The new system can deal with grace notes, trills, and glissandi, musical ornaments that are often not precisely notated and therefore problematic to the matcher. The system also overcomes the most common cases of matcher failure in which the performer plays outside of the window, making it impossible for the matcher to find the proper score location.

Computer accompaniment systems work remarkably well due to the robust pattern matcher and the high information content of each note. The next section considers improvisation, where less is known about the performance and consequently the music understanding task is more difficult.

4. Following Improvisations

Knowledgeable listeners can often identify a popular song even when the melody is not played. This is possible because harmonic and rhythmic structures are present even without the melody. Even the improvisations of a single monophonic instrument can contain enough clues for a listener to discern the underlying harmonic and rhythmic structure. Can a computer system exhibit this level of music understanding?

Many different problems involving improvisation might be posed. For example, one could attempt to recognize a new harmonic structure or simply match against a known one, and one could work with polyphonic or monophonic input. An initial task was chosen somewhat arbitrarily: to construct a system that listens to a real-time performance of a 12-bar blues improvisation by a monophonic instrument. The goals of the system are two-fold: to detect the underlying beat of the improvisation and to locate the start of the cyclical chord progression. This is enough information to, for example, join in the performance with a synthesized rhythm section consisting of piano, bass, and drums.

This improvisation understanding task can be divided into two subtasks: finding the beat and finding the harmonic progression. After lengthy discussions with Bernard Mont-Reynaud, who developed beat-tracking software for the automatic music transcription system described earlier [3], we decided to collaborate in the design and implementation of a ''blues follower'' program [6]. Bernard designed the beat follower or ''foot tapper'', and I designed the harmonic analysis software.

The ''foot tapper'' operates by predicting where the next beat will occur. When a note onset occurs near the predicted beat, the estimated tempo is adjusted in the direction of the note onset. The magnitude of the adjustment is carefully weighted so that onsets far from the predicted beat have little effect while onsets very close to the predicted beat have a strong correcting effect. To initialize the ''foot tapper'' with a good estimate of the initial tempo, various starting criteria are checked in the data. The primary conditions are that three consecutive note onsets are detected with inter-onset times that are approximately equal and within a prescribed range. This fairly simple approach was developed after exploring a number of more complicated approaches.

The first approaches to harmonic analysis were similarly complex and unsuccessful. One of the difficulties with improvisations is that virtually any pitch can occur in the context of any harmony. However, given a harmonic context many notes would only be expected in a particular form such as a chromatic passing tone. This led to the idea that by searching for various features, one might assign functions to different notes. Once labeled with their function, it might be possible after a few notes to unambiguously determine the harmonic context by the process of elimination.

So far, this approach has not been fruitful, so a more statistical approach was tried. In this approach, it is assumed that even though any pitch is possible in any context, there is a certain probability distribution associated with each time position in the 12-bar blues form. For example, in the key of C, we might expect to see a relatively frequent occurrence of the pitch B in measure 9 where B forms the important major third interval to the root of the dominant chord (G). We can calculate a ''correlation'' between the expected distribution and the actual solo to obtain a figure of merit. This not a true numerical correlation but a likelihood estimate formed by the product of the probabilities of each note of the solo. Since we wish to find where the 12-bar blues form begins in the solo, we compute this estimate for each possible starting point. The point with the highest likelihood estimate indicates the most likely true starting point.

Figure 4-1 illustrates a typical graph of this likelihood estimate vs. starting point. (Since only the relative likelihood is of interest, the computed values are not normalized to obtain true probabilities, and the plotted values are the direct result of integer computations. See Dannenberg and Mont-Reynaud [6] for details.) Both the graph and the 12-bar blues form are periodic with a period of 96 eighth notes. Slightly more than one period is plotted so that the peak at zero is repeated around 96. Thus the two peaks are really one and the same, modulo 12 bars. The peak does in fact occur at the right place. There is also a noticeable 4-bar (32 eighths) secondary periodicity that seems to be related to the fact that the 12-bar blues form consists of 3 related 4-bar phrases.

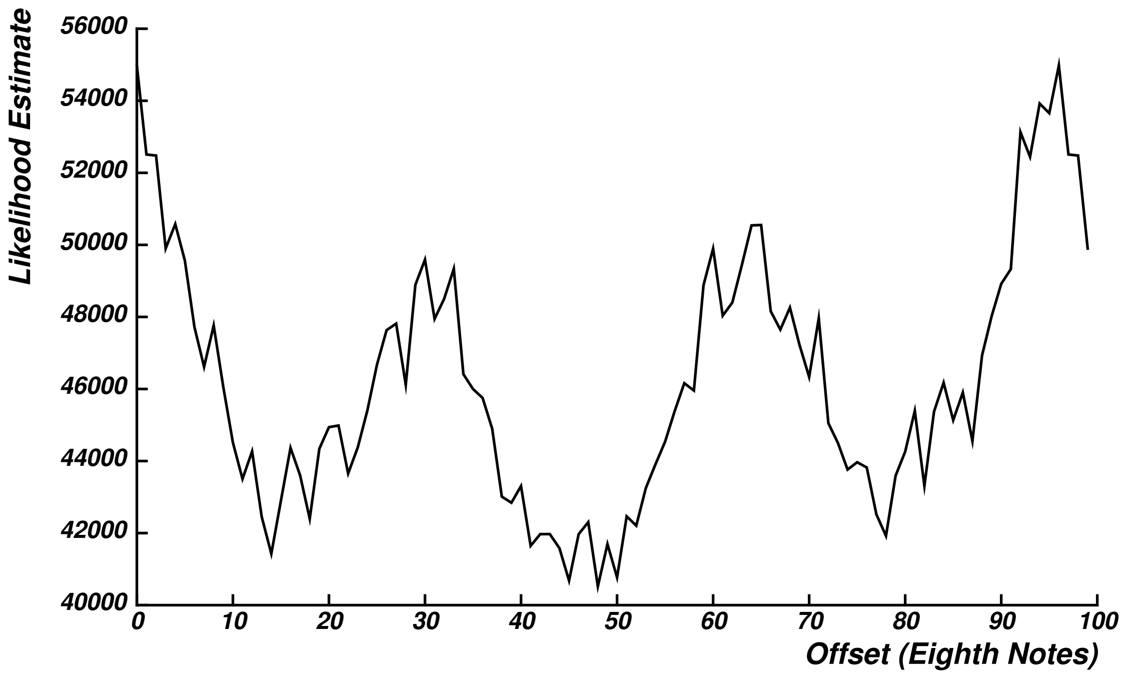

Figure 4-1: Likelihood estimates of the solo starting at different offsets in a 12-bar blues progression.

The probability distribution used for the ''correlation'' can be obtained from actual performances. The beat and starting point of the 12-bar form are recorded along with the notes of the performance or are manually indicated later. The distribution used in Figure 4-1 was obtained in this way and combines the pitches of about 40 repetitions of the 12-bar chord progression. The data were obtained from the recorded output of a real-time pitch detector. An interesting question is whether it matters if the distribution and the solo are created by the same soloist. If so, can this technique be used to identify a soloist? These questions have not yet been studied.

The ‘‘foot tapper’’ and a real-time implementation of the correlation approach were integrated into a real-time improvisation understanding program for further experimentation. The results are interesting, but not up to the level required for serious applications. The tempo estimation software tends to start reliably but eventually loses synchronization with the performance unless the performance is very precise. The ‘‘correlation’’ software can locate the beginning of the blues form only when the performance is very obvious in outlining the harmonic structure. This is less of a problem when the likelihood estimation is performed over many measures, but then the technique becomes totally unsuited to real-time performances.

Even though some of the simplest approaches tried thus far have been the most successful, it seems obvious that human listeners bring together a panoply of techniques and knowledge in order to follow a blues solo and interpret it correctly. Further research is needed to explore new approaches and to examine ways in which results from different approaches can be integrated.

Another direction for future research is to examine a new problem. (If you can’t win, change the rules!) The author is currently studying the problem of following a polyphonic improvisation. It is hoped that the additional information available in polyphonic (chordal) performances will indicate more directly the underlying harmonic structure and enable more accurate tracking. It is also hoped that a more obvious harmonic structure will make it possible for the harmonic analysis subtask to provide reliable clues to a beat analysis subtask running in parallel.

5. Summary and Conclusions

The field of ‘‘music understanding’’ has been introduced as the study of methods by which computer music systems can recognize pattern and structure in musical information. Various music understanding tasks have been described, including different forms of signal analysis, tasks related to music transcription, and performance-related tasks.

Two performance systems were described in detail. One system follows a performer in a score and plays a composed accompaniment in synchrony. The other system attempts to identify the tempo and locate the beginning of a 12-bar blues form while listening to a monophonic soloist in real time.

There are many tasks which might be performed by future music understanding systems. One task is providing very high-level interfaces for music composition and editing systems. For example, a composer might ask a computer-based editing system to play the second occurrence of a particular theme. Music understanding is also the first step toward the automatic interpretation of a traditionally notated score to obtain a musical performance. Recognition of musical style, rhythmic patterns and phrases is a prerequisite to a competent performance, and such information could be used to enhance systems that attempt to obtain musical performances from notated scores [14]. Music understanding might be used in musical databases to classify compositions and to allow searching for phrases or for pieces that meet certain criteria. In the area of analysis, music understanding might lead to new theories or a better understanding of music by enabling scholars to search for certain abstract patterns or structures. In the performance area, music understanding can help make computer systems easier to use and less error-prone in live performances by carrying out pre-stored plans rather than requiring extensive manual control. There is also the possibility of building more expressive and/or easier-to-learn instruments that are based on an understanding of the gestures and musical goals of the performer. Making instruments easier to learn is particularly important for experimental music where computer music instruments are rapidly evolving.

Present-day systems illustrate that even low-level music understanding can have a profound effect on the way musicians and computers interact. As the level of understanding increases, the applicability of computers to musical problems will grow. With the flexibility and adaptability made possible by music understanding, computer-based systems seem likely to take an increasingly important role in music composition, performance, and in the development of musical esthetics.

References

[1] Berger, Jonathan.
Theory and Practice: Chicken and Egg.
In *The Arts and Technology II: A Symposium*, pages 24-30. Connecticut College, 1989.

[2] Bloch, J. J. and R. B. Dannenberg.
Real-Time Computer Accompaniment of Keyboard Performances.
In B. Truax (editor), *Proceedings of the International Computer Music Conference 1985*, pages 279-290. International Computer Music Association, 1985.

[3] Chafe, Chris, Bernard Mont-Reynaud, and Loren Rush.
Toward an Intelligent Editor of Digital Audio: Recognition of Musical Constructs.
Computer Music Journal 6(1):30-41, Spring, 1982.

[4] Chafe, Chris, David Jaffe, Kyle Kashima, Bernard Mont-Reynaud, and Julius Smith.
Techniques for Note Identification in Polyphonic Music.
In *1985 Proceedings of the International Computer Music Conference*, pages 399-405. International Computer Music Association, 1985.

[5] Dannenberg, R. B.
An On-Line Algorithm for Real-Time Accompaniment.
In W. Buxton (editor), *Proceedings of the International Computer Music Conference 1984*, pages 193-198. International Computer Music Association, 1984.

[6] Dannenberg, R. B. and B. Mont-Reynaud.
Following an Improvisation in Real Time.
In J. Beauchamp (editor), *Proceedings of the 1987 International Computer Music Conference*, pages 241-248. International Computer Music Association, San Francisco, 1987.

[7] Desain, Peter and Henkjan Honing.
Quantization of Musical Time: a connectionist approach.
Computer Music Journal 13(3):56-66, 1989.

[8] Foster, Scott, Schloss, W. Andrew, and Rockmore, A. Joseph.
Toward an Intelligent Editor of Digital Audio: Signal Processing Methods.
Computer Music Journal 6(1):42-51, Spring, 1982.

[9] Longuet-Higgins, H. C. and C. S. Lee.
The Perception of Musical Rhythms.
Perception 11:115-128, 1982.

[10] Moorer, J. A.
On the Segmentation and Analysis of Continuous Musical Sounds by Digital Computer.
PhD thesis, Stanford University Department of Computer Science, 1975.

[11] Sankoff, David and Joseph B. Kruskal, editors.
Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.
Addison-Wesley, Reading, Mass., 1983.

[12] Schloss, A.
On the Automatic Transcription of Percussive Music.
PhD thesis, Department of Speech and Hearing, Stanford University, June, 1985.
Department of Music Technical Report STAN-M-27.

[13] Simon, H. A.
Perception du Pattern Musical par AUDITEUR.
Sciences de l'Art 2:28-34, 1968.

[14] Sundberg, J., A. Askenfelt, L. Fryden.
Musical Performance: A Synthesis-by-Rule Approach.
Computer Music Journal 7(1):37-43, Spring, 1983.