Proceedings of the Twentieth National Conference on Artificial Intelligence, Pittsburgh, Pennsylvania, July 9-13, 2005

Modeling Form for On-line Following of Musical Performances

Bryan Pardo[1] and William Birmingham[2]

[1]Computer Science Department, Northwestern University, 1890 Maple Ave, Evanston, IL 60201
[2]Department of Math and Computer Science, Grove City College, 100 Campus Drive, Box 3123, Grove City, PA 16127
pardo@northwestern.edu, wpbirmingham@gcc.edu

Abstract

Automated musical accompaniment of human performers often requires an agent be able to follow a musical score with similar facility to that of a human performer. Systems described in the literature represent musical scores in a way that assumes no large-scale structural variation of the piece during performance. If the performer deviates from the expected path by skipping or repeating a section, the system may become lost. We describe a way to automatically generate a Markov model from a written score that models the score form, and an on-line algorithm to align a performance to a score. The resulting system can follow performances that take alternate paths through the score without losing its place. We compare the performance of our system to that of sequence-based score followers on a melodic corpus of 98 Jazz melodies. Results show that explicitly representing the branching structure of a score significantly improves score following when the branch a performer may take is unknown beforehand.

Introduction

Automated musical accompaniment that reacts naturally to the human performer is a long-standing goal of a number of computer-music researchers (Grubb and Dannenberg 1994, Dannenberg 1984; Bloch and Dannenberg 1985; Toiviainen 1998; Raphael 1999). The ideal is a peer musician that can be integrated into an ensemble of human players with minimal need for the humans to adjust their interaction styles to accommodate the computer performer. For many styles of music, this requires an agent that is able to follow a representation of a written score with similar facility to that of a human performer. Systems that perform this function are called score matchers or score followers.

Figure 1 shows a simple case of score following. The top portion of the figure contains a simple written lead sheet, or score. A musician *performs* a score by translating the note, chord, key, and other symbols into a sequence of performance actions (depress piano key *k* at time *t* with velocity *v*, for example). These actions result in a sequence of events that make the *performance*. In computer score following, the performance is often encoded as MIDI (MIDI-Manufacturers-Association 1996). Figure 1 shows an example MIDI performance of the written score. This is shown in piano roll notation. Here, each bar represents a note. The horizontal placement of the note represents the onset time. The vertical placement of the note represents the pitch. Note duration is indicated by the length of a bar.

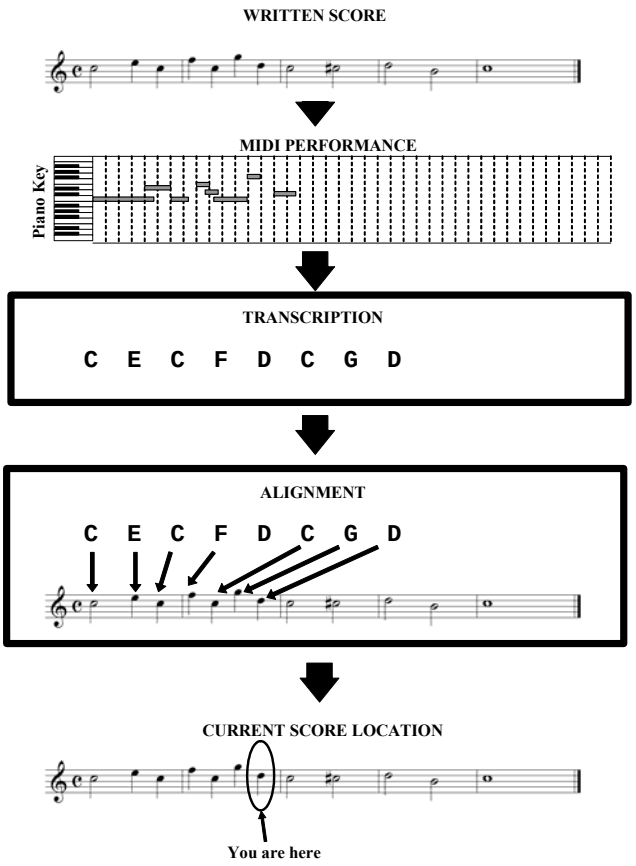

Figure 1. Score Following

Score following can be broken down into *transcription* and *alignment* (also known as *matching*). Transcription involves parsing the performance into a sequence of salient events. In Figure 1, transcription consists of encoding each MIDI note-on event as a simple pitch class, drawn from the set of twelve pitch classes used in Western music.

Matching consists of finding the best alignment between the sequence of events in the performance transcription

and the events in the score. Typically, these events are single notes, although they may also be lyrics, percussion sounds, or groups of notes. A score follower, unless otherwise stated, is presumed to align the performance to the score in real-time as the performance takes place.

Because of the difficulty of dealing with polyphonic MIDI and audio, researchers (Dannenberg 1984; Dannenberg and Mont-Reynaud 1987; Puckette and Lippe 1992; Large 1993; Puckette 1995; Vantomme 1995; Desain, Honing et al. 1997; Grubb and Dannenberg 1997) generally restrict matching to a monophonic score that (nearly) completely specifies the pitch and ordering of every note. For the sake of simplicity, information in the score about key, meter, dynamics, and song structure is ignored, leaving a simple sequence of note on and off events.

The standard practice for score following (Vantomme 1995; Desain, Honing et al. 1997; Grubb and Dannenberg 1997) is to linearize a score by removing structural branch points (e.g., repeats, codas, etc.) before the performance begins. This effectively limits the performance to a single path through the form that is not changeable during performance.

Thus, existing score followers require fixing, *a priori*, how the repeats in a score, like that in Figure 2, would be handled by the musicians. The performers would have to agree on repeating (at the end of measure two) once and then going on to the end. The musicians would then be prohibited from altering the path during performance.

In many live performance situations, musicians repeat or skip a section in response to the needs of the moment. Musicians often extend a piece to let dancers who are enjoying the music continue dancing, or shorten a piece (perhaps by skipping an introductory section) when they are running behind schedule. In these cases, existing score followers cannot adapt to the changing performance situation.

When a score contains repeats and particularly when the form is variable (e.g., the form may be ABA or AABA or any permutation depending on the whim of the performers) a score representation that does not allow branch points is undesirable. To account for variability in form, we need to extend the score model to represent structural score elements that affect the form, such as repeats and codas.

In this paper, we introduce a new method for representing large-scale form in score following. Our representation is based on Markov models, which allow us to both capture the form of a piece implied by the score, as well as reason probabilistically about how a performer is moving through the piece. The system can then model performances that may start anywhere in the form and repeat or skip sections (as specified by the score) a non-predetermined number of times. This greatly expands the types of music amenable to automatic score following.

The remainder of this paper describes Markov models, shows how one may follow a variable-form performance using Markov models, and compares Markov model score followers to string matching based followers, using a corpus of 98 Jazz melodies.

Music Scores as Markov Models

A Markov model describes a process that goes through a sequence of discrete states, such as notes or chords in a lead sheet. The model is a weighted automaton that consists of:

- A set of states, $S = \{s_1, s_2, s_3, \ldots, s_n\}$
- S_E, a subset of S containing the legal ending states. As a default $S_E=S$
- A set of possible emissions, $E=\{e_1, e_2, \ldots, e_n\}$
- A transition function, $\tau(s_i, s_j)$, that specifies the probability of a transition to s_j, from s_i
- A function, $\sigma(s_i)$, that defines the probability of beginning in s_i
- An emission function $\varepsilon(s_j, e_i)$, that defines the probability state s_j will emit e_i.

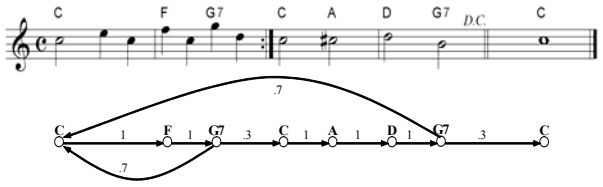

Figure 2. A score as a directed graph (Markov model)

Markov models are generative. A generative model describes an underlying structure able to emit a sequence of observed events. A musical score may be represented as a (hidden) Markov model. The directed graph in Figure 2 shows a Markov model created from the chord labels in the score passage in the figure. Nodes represent chords in the score. Directed edges (arrows) represent transitions. Repeats and skips (codas) in the score are represented by directed edges connecting distant portions of the score model. Transition probabilities are indicated by a value associated with each edge.

An observation sequence, $O = o_1, o_2, \ldots, o_n$, is a sequence of events drawn from the emission alphabet, E. Relating this to music, the sequence of musical events (notes, chords, etc.) generated by the performer is the observation sequence generated in response to the score.

The emission function $\varepsilon(s_j, e_i)$ defines the probability that state s_j will emit e_i. For a music performance, this is equivalent to the probability the jth item in the score (a chord symbol) would elicit the ith performance event (a chord voicing on the piano). A Markov model is called a hidden Markov model, or HMM, when it has at least one state whose emission function is non-zero for multiple elements of the emission alphabet. An example would be a chord symbol that maps onto multiple chord voicings.

In our approach, the emission function is determined beforehand through an empirical study of the likelihood of performance events, given each score state. Good estimation of the emission function lets a system model a variety of possible causes for variable performance output in response to score states, including production errors

(cracked notes, poor pitch control), transcription errors, and intentional variation (alternate voicings).

For example, we calculate a note-based emission probability function $\varepsilon(s_j, e_i)$ for an alto saxophonist by recording the musician and automatically transcribing his performance of an assigned set of chromatic scales and chord arpeggios. The resulting count of associations between performed pitch and transcribed pitch is used to estimate $\varepsilon(s_j, e_i)$ (Pardo and Birmingham 2002). In another study (Pardo 2005), we estimate the likelihood a set of performance notes mapping onto a chord symbol in the score through empirical study of the performances of a Jazz pianist on a known sequence of chord symbols. The associations learned in this training phase can then be used to estimate $\varepsilon(s_j, e_i)$ where the score element is a chord name and the performance emission is a set of notes.

Using state transition values derived from the score and emission probabilities based on prior training allows the construction of score models without the need to train the model on a set of performances of that score prior to use. This lets the system function on performances of scores it has never "heard" before.

Finding the current state in the model

For score following, we want to know the most likely current state in the score model, given the observation sequence. To find the most likely current state, we modify the Forward-Backward algorithm (Rabiner and Juang 1993) for real-time score following. Our approach is distinct from the standard algorithm, in that it is designed to work on an in-progress sequence (a live performance), rather than a completed sequence. Thus, we use only causal information.

The emission function $\varepsilon(s_j, e_i)$ gives the probability that state s_j will emit e_i. We define the observation function, $\phi(s_j, e_i)$ as the probability of being in s_j when observing e_i. Equation 1 follows directly from Bayes' theorem. Here, $P(e_i)$ is the prior expected probability of the ith performance event and $P(s_j)$ is the prior probability of the jth score state.

$$\phi(s_j, e_i) = \frac{P(s_j)\varepsilon(s_j, e_i)}{P(e_i)} \quad (1)$$

For some music styles or performers, it may make sense to develop estimates of performance events and score state likelihood, especially if the style tends to use a subset of the possible pitches. It may, however, be impractical to collect meaningful statistics for the full alphabets of score states and performance events. In this case, one can save significant training time by assuming all emissions in E have equal prior probability and all states in S have equal prior probability. Given this assumption, the ratio of their probabilities is a constant, k, as follows:

$$\phi(s_j, e_i) = k\varepsilon(s_j, e_i) \quad (2)$$

Given an observation sequence, $O = o_1, o_2, \ldots, o_n$, and a starting probability distribution $\sigma(s_j)$, we define the alpha function for the first as follows:

$$\alpha(s_j, o_1) = \phi(s_j, o_1)\sigma(s_j) \quad (3)$$

This captures the probability of starting in any given state in the model. The alpha function for each subsequent observation may be calculated recursively using Equation 4. Here, the summation captures the likelihood of arriving in state s_j over all routes through the Markov model of length i.

$$\alpha(s_j, o_i) = \phi(s_j, o_i)\sum_{s_k \in S}\left(\tau(s_k, s_j)\alpha(s_k, o_{i-1})\right) \quad (4)$$

Equation 4, when implemented, will often generate underflow errors, as the length of the observation sequence increases. We are not interested in the likelihood of the overall sequence up to the current observation. We are only interested in finding the most likely state when we have reached the ith observation in the sequence. Given this, we create a state-value function that normalizes state probabilities at each observation, avoiding underflow, as given in Equation 5.

$$v(s_j, o_i) = \frac{\alpha(s_j, o_i)}{\sum_{s_k \in S}\alpha(s_k, o_i)} \quad (5)$$

Equation 5 requires that we make an adjustment to Equation 4, resulting in Equation 6.

$$\alpha(s_j, o_i) = \phi(s_j, o_i)\sum_{s_k \in S}\left(\tau(s_k, s_j)v(s_k, o_{i-1})\right) \quad (6)$$

The state with maximal value is then taken to be the current location, l, in the model. Thus, the current score location is given by Equation 7.

$$l_i = \arg\max_{s \in S}(v(s, o_i)) \quad (7)$$

The normalization in Equation 5 is not possible if the observation sequence cannot be generated by the Markov model and the probability of being in any state in the model is zero. If, for some observation o_i, all states have an observation probability of zero, the presumption is that the performer has played something not in the score. The score follower then resets and o_{i+1} is treated as the initial observation, Equation 3 is applied, and score following proceeds again from that point.

The Viterbi algorithm (Rabiner and Juang 1993) is a commonly-used alternative to the Forward algorithm. Instead of calculating the likelihood of s_j given all the paths through the model, the Viterbi algorithm estimates the probability of only the most likely path through the model. Our method can use a Viterbi-style estimation by replacing the summation in Equation 6 with the maximization in the Equation 8.

$$\alpha(s_j, o_i) = \phi(s_j, o_i) \cdot \max_{s_k \in S} \left(\tau(s_k, s_j) v(s_k, o_{i-1}) \right) \quad (8)$$

While it is often better to favor the most likely path, since it gives the alignment output continuity, there are subtle differences. For score following, Viterbi must be adapted for on-line use, where the best current state may be asked for at any time. In the on-line case, Viterbi may be more susceptible to "garden-path" errors, where what initially appears to be the correct path proves to be incorrect only after several additional observations have been made. Later in this paper, we compare the on-line performance of the Viterbi and Forward methods on the corpus.

The current model architecture

We wish to create a single model that handles small-scale formal variation (e.g., the performer skips or repeats a note) and large-scale formal variation (e.g., the performer skips or repeats a section of the music). By introducing specific topological features into our Markov model, we cover both situations.

Consider the following, if the performer skips a single score event, this may be modeled with skips in the Markov model. Figure 3 shows a hidden Markov model for the first eight beats of a written score. This model admits skipped or repeated states, as well as allowing for repeats as shown in the written score. Here, each state represents a beat. The arrows represent allowable transitions between states. This model also allows for self loops on every state. The loop from the eighth state back to the first state corresponds to the repeat sign shown after the eighth beat of the written score.

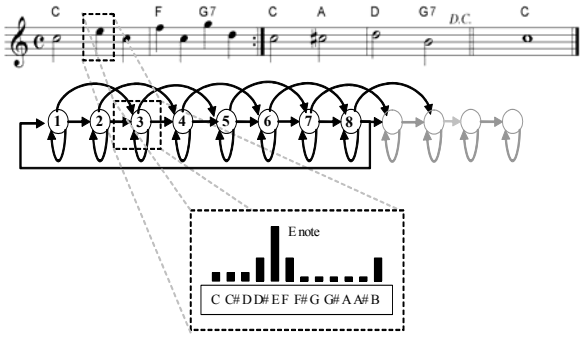

Figure 3. An HMM model allowing skips and repetitions

The histogram in Figure 3 shows the emission probability function for the third state in the model, which corresponds to the "E" in the written score. Here, the height of the bar corresponds to the relative likelihood of observing the given pitch class when in the third state of the model. The emission probability function for each element in the alphabet of possible score states is developed before constructing the score model, by analysis of a corpus of music performances in the style of the piece to be performed (Pardo and Birmingham 2002).

The following section describes a simple score following experiment that compares the use of an HMM like that in Figure 3 with the typical string alignment algorithms, commonly used for score following.

A Simple Experiment

We have asserted that using a Markov model to explicitly model branch points in the written score improves score following where a performer repeats a section an unpredictable number of times. The following experiment illustrates this point.

We created a synthetic corpus, based on well-known Jazz pieces, designed to emphasize the effect of adequate score-structure representation on the ability of a score follower to handle alternate paths through the score that are chosen at run-time. The corpus consists of melodic lines from 98 Jazz pieces. These range from Bossa Nova (*Corcovado*), to Ballad (*What's New*) to Blues (*Blue Monk*), to Swing (*Back Home in Indiana*), to Jazz Waltz (*Alice in Wonderland*) to modal pieces (*Footprints*).

For each piece, the score consisted of the full written melody of the piece as shown in the Real Book[1], truncated to the first 64 beats and encoded as scores in the format of the popular music notation program *Sibelius*. A repeat mark was inserted at the end of the 32nd beat of each score. This, effectively made each score have form AB, where the A section could be repeated an unspecified number of times.

A Markov model with the graph connectivity shown in Figure 3 was automatically generated from each score. All states in the model were set to an equal initial starting probability. Each state represented a single beat. At each beat, the Markov model could either repeat the state, move on to the next state, or skip forward two states. For this experiment, the transition probability for moving forward a single state was 0.5, while those of repeating and skipping forward two states were each 0.25. At the end of state 32 (the 32nd beat), there was an additional connection back to the first state (the repeat of the A section).

For each score, we created four MIDI performances: one that performed the A section once, another that performed it twice, a third that performed it three times and a final performance that skips the section entirely. We then followed the performance using the on-line modification of the Forward algorithm, the on-line modification of the Viterbi algorithm, an on-line local string matcher, and an on-line global string matcher used by a variety of researchers (Dannenberg 1984; Dannenberg and Mont-Reynaud 1987; Puckette 1995; Desain, Honing et al. 1997).

String matchers find the best alignment between two strings (sequences) by finding the lowest cost transformation of one into the other, in terms of operations

[1] The Real Book is a standard, albeit illegal (with no publisher or author information), compendium of Jazz lead sheets, used by professional Jazz musicians.

(insertion or deletion of characters). Such matchers are all based on similar techniques and are the "classic" score following approach. Dynamic-programming based implementations that search for a good alignment of two strings have been used for over 30 years to align gene sequences based on a common ancestor (Needleman and Wunsch 1970). Global String alignment requires every element of the performance to be accounted for in an alignment to the full score. Local string alignment allows matching of a substring of the performance to any portion of the score (Pardo and Birmingham 2002).

The string matchers were unable to use the repeat information in the score file and thus always expected a performance with no repeat. Given this, one would expect the performance of both string matchers to degrade whenever the A section was presented an unexpected number of times, while the Markov models should maintain roughly similar performance, regardless of the number of repeats. The case where the initial section is skipped entirely should favor both Markov model approaches and the local string matcher.

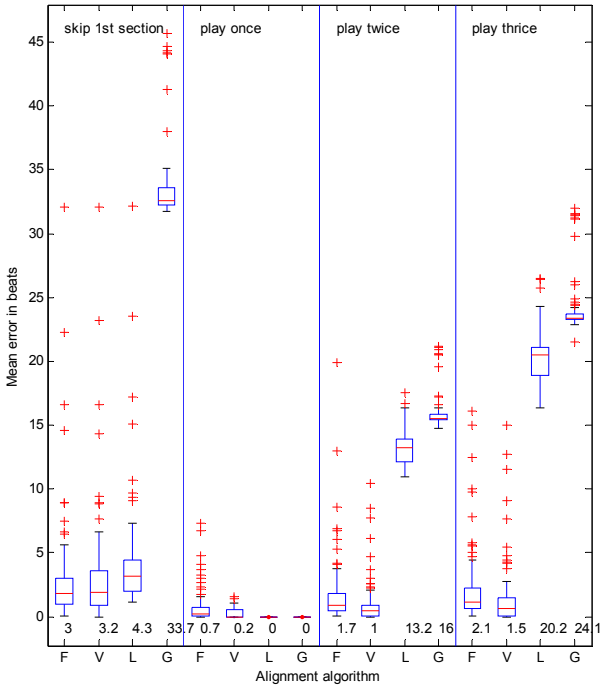

Figure 4. Mean score-follower error, by algorithm

Figure 4 shows the score tracking errors generated in this experiment. Within each group, "F" stands for the Forward algorithm, "V" stands for the Viterbi algorithm, "L" stands for local string alignment, and "G" stands for global string alignment. Each column shows a box plot with lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the box to show the extent of the rest of the data. Outliers are indicated by plus symbols beyond the ends of the whiskers. All values indicate mean distance (in beats) between the correct location in the score and the location reported by the score follower over the course of a performance, or group of performances. Below each box plot is the mean error for all cases.

The "play once" group in Figure 4 corresponds to the case where the number of repetitions is known beforehand. In this case, both string alignment approaches work perfectly (assuming one begins at the start of the piece), while the Markov models occasionally make a wrong choice at a repeat and take some time to recover, with the Forward algorithm performing worse than Viterbi.

Once the number of repeats increases, the benefits of explicit representation of score structure become clear. Both the string matching methods get lost for extended periods in the "play twice" and "play thrice" conditions, resulting in average position estimates that are many beats away from the correct location. Both the Forward and Viterbi-based followers stay within an average of two beats under all conditions, with the Viterbi performing slightly better, on average.

When the first section is skipped entirely, the global string alignment method fails. Both methods based on the HMM and the local string alignment method do significantly better, with methods based on the Markov model doing better than local string alignment.

Summary and Conclusions

We have described a score representation designed to handle the large-scale form variation often found in live performances of many styles of music. Our representation explicitly models those elements of a musical score that indicate repeats and jumps to different sections (coda symbols), and thus possible changes in form. From these elements, we induce a Markov model that allows us to accurately follow a live performance, using the on-line modification of either the Forward or the Viterbi algorithm. The model is generated automatically from the score, and can be used without training on a corpus of performances of the score in question. This is a significant advance in score following technology.

The experimental results in this paper show that explicitly representing branch points in a score significantly improves score following when the form a performer may take is not known beforehand and that the on-line Viterbi algorithm performs best on the performance corpus assembled for this paper.

Acknowledgments

The majority of this research was conducted at The University of Michigan, Ann Arbor, with partial support from the National Science Foundation under grant IIS-0085945. The opinions in this paper are solely those of the authors and do not necessarily reflect the opinions of the funding agency. We also thank Roger Dannenberg for comments on various sections of this work.

References

Bloch, J. and R. D. Dannenberg (1985). Real-Time Computer Accompaniment of Keyboard Performances. *International Computer Music Conference.*

Dannenberg, R. (1984). An On-Line Algorithm for Real-Time Accompaniment. *International Computer Music Conference.*

Dannenberg, R. and B. Mont-Reynaud (1987). Following an Improvisation in Real Time. *International Computer Music Conference.*

Desain, P., H. Honing, et al. (1997). Robust Score-Performance Matching: Taking Advantage of Structural Information. *International Computer Music Conference.*

Grubb, L. and R. Dannenberg (1994), Automated Accompaniment of Musical Ensembles. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, AAAI, pp. 94-99

Grubb, L. and R. Dannenberg (1997). A Stochastic Method of Tracking a Vocal Performer. *International Computer Music Conference.*

Large, E. W. (1993), Dynamic Programming for the Analysis of Serial Behaviors. *Behavior Research Methods, Instruments and Computers* **25**(2): 238-241.

MIDI-Manufacturers-Association (1996). The Complete MIDI 1.0 Detailed Specification. Los Angeles, CA, The MIDI Manufacturers Association.

Needleman, S. B. and C. D. Wunsch (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48: 443-453.

Pardo, B. (2005). Probabilistic Sequence Alignment Methods for On-line Score Following of Music Performances, Doctoral Dissertation, Electrical Engineering and Computer Science. University of Michigan: Ann Arbor, MI.

Pardo, B. and W. Birmingham (2002). Improved Score Following for Acoustic Performances. *International Computer Music Conference (ICMC)*, Goteborg, Sweden.

Puckette, M. (1995). Score following using the sung voice. *International Computer Music Conference.*

Puckette, M. and C. Lippe (1992). Score Following In Practice. *International Computer Music Conference.*

Rabiner, L. and B.-H. Juang (1993). *Fundamentals of Speech Recognition*. Englewood Cliffs, New Jersey, Prentice-Hall.

Raphael, C. (1999). Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21**(4): 360-370.

Toiviainen, P. (1998). An Interactive MIDI Accompanist. *Computer Music Journal* **22**(4): 63-75.

Vantomme, J. (1995). Score Following by Temporal Pattern. *Computer Music Journal* **19**(3): 50-59.