Robust Polyphonic Midi Score Following with Hidden Markov Models

Diemo Schwarz, Nicola Orio, Norbert Schnell
schwarz@ircam.fr, *orio@dei.unipd.it*, *schnell@ircam.fr*
Ircam - Centre Pompidou, 1 pl. Igor Stravinsky, 75004 Paris, France
University of Padova, Dept. of Information Engineering, Via Gradenigo 6/B, 35131 Padova, Italy

Abstract

Although modern audio score following systems work very well with low polyphony performances, they are still too imprecise with highly polyphonic instruments such as the piano, or the guitar. On the other hand, these instruments can easily output Midi information which shows that our work on robust Midi score following is still needed. We propose an adaptation to Midi input of our HMM-based stochastic audio score follower, focusing the attention on the piano as our test instrument. The acoustic salience of the Midi notes is modeled by an amplitude envelope, taking into account the sustain pedal, from which note match and attack probabilities are derived. Tests with a complex piano piece played with many errors showed a very high robustness.

1 Introduction

Score following is the synchronisation of a computer that provides an accompaniment (audio synthesis or sound processing) to a human performer who plays a known musical score. It has a history of about twenty years as a research and musical topic (see Orio, Lemouton, Schwarz, and Schnell (2003) for an overview), and is an ongoing project of Ircam's Real-Time Applications Team (ATR 2004). Research is still very active in this area, see for example the various recent approaches proposed by Loscos, Cano, and Bonada (1999), Raphael (1999), Orio and Déchelle (2001), Shalev-Shwartz et al. (2002), Izmirli, Seward, and Zahler (2003). Early prototypes of score followers were based on Midi instruments, or relied on pitch-to-Midi converters. For this reason, the typical application of a score follower was the accompaniment of a monophonic instrument. More recent score followers directly use the audio signal as their input, without requiring pitch detection, and thus can cope with slightly polyphonic performances, for instance chords played on a violin.

Instruments capable of high polyphony, like the piano, the organ, and the guitar, are still too difficult for audio score following. Yet, Midi output can be easily obtained from these instruments by adding converters. In an inversion of the historical development, we propose a Midi follower that is an adaptation of our audio score following system (Orio and Déchelle 2001), where audio input is replaced by an acoustic salience measure predicted from Midi input, the methodology for on-line alignment being unchanged. Midi score following of highly polyphonic performances introduces a number of interesting problems which have not yet been addressed in sufficient depth, neither on the score nor the performance modeling side.

As for all approaches to score following, the system requires a model of the musical score performed by the musician, which is built from an external score coding and preprocessed to deal with polyphonic performances (section 2). In parallel, the system needs to model the Midi performance in order to match it to the score (section 3). Alignment is then carried out using a statistical approach (section 4), the advantages of which are the possibility to train the system and to model different features from examples of performances and scores. The widely used Hidden Markov Models (HMMs) can deal with the several levels of unpredictability typical of performed music. We tested our system on a real situation, using *La Frontière* by Philippe Manoury as our test case (section 5).

2 Score Modeling

The internal score model is built by parsing an external score file, which is not straightforward in the polyphonic case. As implicitly introduced in (Orio and Schwarz 2001), the result of the score parsing is a time-ordered sequence of *score events* at every change of polyphony, i.e. at each note start and end, as shown in figure 1. The score states are created between the score events. Rest states are created when the polyphony is zero.

Many score files, especially those generated by recording a Midi-performance, contain score events with slight desynchronisations: For instance, the notes of a chord played on a keyboard are in general not triggered perfectly synchronous, but are slightly arpeggiated (figure 2). Equally, notes that should be played with a legato articulation can have short overlap or gaps (figure 3). To avoid generating too many very short states, a *quantisation* is performed that fuses score events within a window of 30 ms into one single event, and

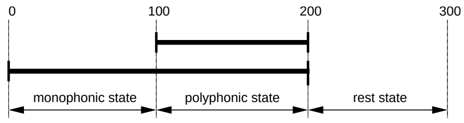

Figure 1: Score parsing into score events and the score states between them.

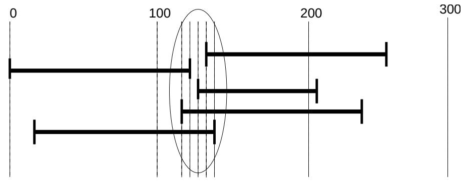

Figure 2: Desynchronised chord

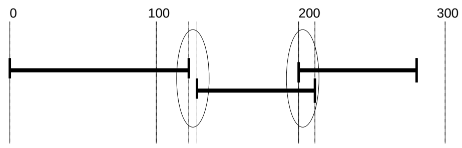

Figure 3: Desynchronised legato notes

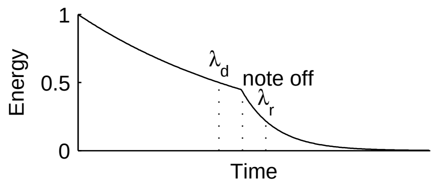

Figure 4: Modeling of the "energy" curve of a Midi note.

eliminates rests shorter than 100 ms altogether. The result is that all circled events in figures 2 and 3 are moved to the time of the earliest event for each circle.

Each polyphonic event is then described by the notes that are expected to be alive in the corresponding score state: We refer to the set of score notes at state i with symbol $S(i)$. Moreover, each event may have a number of notes that are expected to start at the beginning of the corresponding state: We refer to the set of expected attacking notes of state i with symbol $A(i)$.

Non-note objects with a special interpretation can be specified in the score: trills are modeled as one score state, and rests can be explicitly given in the score in order to assign them a cue, i.e. a label to be output to trigger some action.

3 Performance Modeling

A naive interpretation of the Midi input as a gate signal (a note is alive or not) is sufficient for purely monophonic music, but presents severe disadvantages for highly polyphonic piano music, due the use of the sustain pedal: First, sustained notes can be obtained with the pedal or without, depending on the musician. To ignore the pedal and treat only note-on and note-off messages is not possible because of this ambiguity which leads to score–performance mismatch. Second, a long part with constantly held pedal will produce a very high Midi polyphony, although most of the notes are no longer audible.

The solution is to take into account the acoustic prominence of a played note by simulating its audio envelope by an exponentially decaying "energy" curve (we use the term *energy* in this sense from now on) with the two stages of decay and release. This solves a third problem of very short notes in arpeggiated chords that would only match partly because the notes are not present long enough to overlap completely.

At the start of an incoming Midi note with pitch n, its energy $e(n)$ is set to one. After each update period (*tick*) of τ, the energy is multiplied by a decay factor d_d while the note is on, and a release factor d_r after note off. The two factors can be specified by half-life, i.e. the time λ required for reducing the energy to half of its initial value as shown in figure 4. The decay factor d_d is given by

$$d_\mathrm{d} = 2^{-\frac{\tau}{\lambda_\mathrm{d}}}$$

analogously, d_r is given by the release half-life λ_r.

Based on this energy, we can define a *note match* m_n where the current performance notes are compared to the expected notes from the score: For each expected note s in $S(i)$, sum their energy, and normalise by the sum of the energy of all alive performance notes p as $E = \sum_p e(p)$.

$$m_\mathrm{n} = \begin{cases} \frac{\sum_{s \in S(i)} e(s)}{E} & \text{if } E > 0 \\ 0 & \text{otherwise} \end{cases}$$

The *attack match* m_a helps being faster in the detection of arpeggiated chords, where the full note match is reached only when all notes have arrived. It is given by comparing the alive performance notes with the set of notes $A(i)$ that are expected to attack, normalised by their number $||A(i)||$:

$$m_\mathrm{a} = \begin{cases} \frac{\sum_{a \in A(i)} e(a)}{||A(i)||} & \text{if } ||A(i)|| > 0 \\ 0 & \text{otherwise} \end{cases}$$

4 HMM-based Score Following

To use Hidden Markov Models for score following, we identify the observation sequence with the features extracted from the performance (the match values defined in section 3), and the state sequence with the score. Orio and Déchelle (2001) introduced a two-level model, which distinguishes between a low-level note model (section 4.1), and a high-level score model (section 4.3). They also proposed a real time decoding algorithm (section 4.4), differing from the standard HMM Viterbi decoding, that tells us which HMM state is aligned with the current performance frame.

4.1 Note Model

The low-level models the score states, which are either a group of notes or a rest. Each of these note models is a left-to-right HMM consisting of three different kinds of *low-level states*, as depicted in figure 5. A single attack state a, which emits features related to the attack modeling. A sequence of sustain states s, which share the same emissions of features related to note modeling. Their number n and self-transition probability p model the expected duration of the state. A single release state r, which emits features related to the silence or rests and which can be skipped to accommodate legato playing.

It is assumed that observations, that is the parameters extracted from the Midi performance, are emitted only by the low-level states, according to the observation likelihoods described in the following section.

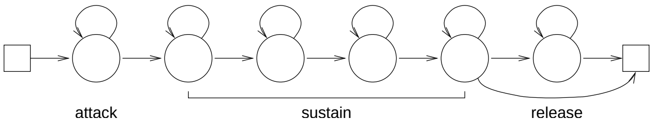

Figure 5: One note model with its low-level states.

4.2 Observation Likelihoods

The HMM's observation likelihoods are given by *probability density functions* (PDFs), which essentially perform a mapping from one of the features described in section 3 to a probability of a low-level HMM state of type attack or sustain or rest. We use thresholded exponential probability density functions (TEPDFs), shown in figure 6, instead of Gaussian because they better model the features we encounter.

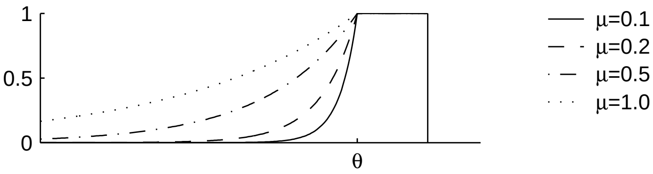

Figure 6: Thresholded exponential probability density function for upper threshold θ and various values of μ.

The note match probability P_{m_n} is given directly by the note match m_n. We derive the probability of attack with the definition of a normalised upper-threshold exponential PDF:

$$P_{\mathrm{attack}} = \begin{cases} e^{-\frac{\theta_a - m_\mathrm{a}}{\mu_a}} & \text{if } m_\mathrm{a} > \theta_a \\ 1 & \text{otherwise} \end{cases}$$

and the probability of sustain as the Gaussian PDF

$$P_{\mathrm{sustain}} = e^{-\left(\frac{\mathrm{m_a} - \theta_s}{\mu_s}\right)^2}$$

The values used for our experiments are τ = 5 ms, θ_a = 0.8, which corresponds to 2 ticks with the default half-life of λ_d = 500 ms, μ_a = 0.3, θ_s = 0.7 (4 ticks), μ_s = 0.3, and θ_r = 0.1, μ_s = 0.4, λ_r = 100 ms.

The final observation likelihoods O_x for the low-level state classes are combined from the note match and the attack/sustain probabilities, and the rest probability is derived by a lower-TEPDF from the total energy E:

$$\begin{aligned} O_a &= P_{m_\mathrm{n}} P_{\mathrm{attack}} \\ O_s &= P_{m_\mathrm{n}} P_{\mathrm{sustain}} \\ O_r &= \begin{cases} e^{-\frac{E-\theta_r}{\mu_r}} & \text{if } E > \theta_r \\ 1 & \text{otherwise} \end{cases} \end{aligned}$$

4.3 Score Model

For each event in the score (as defined in section 2), a note model is created. We call each of these models a *high-level state*. The high level is mostly conceptual and helps us to more easily express a topology for the score modeling.

Together with the sequence of events in the score, which have temporal relationships that are reflected in the left-to-right structure of the HMM, also possible performance mismatches are modeled. As introduced by Dannenberg (1984), there are three possible errors: wrong notes, skipped notes, or inserted notes. The model copes with these errors by introducing error states, or *ghost states*, that model the possibility of a wrong event after each event in the score. Figure 7 shows the possible paths for a correct performance (a) and the three possible errors *wrong event* (b), *skipped event* (c), where the model has to wait for the next correct event to resynchronise itself to distinguish a skipped note from a wrong note, and *extra event* (d). Only one ghost state is shown for clarity.

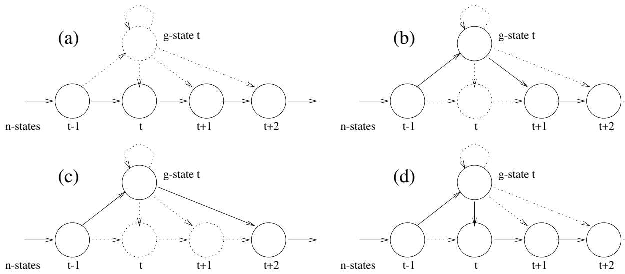

Figure 7: High-level states with different possible errors.

4.4 Decoding

The decoding of the HMM states tells us for each performance frame which state is most probable to be aligned with this frame. The proposed approach is different from the standard Viterbi algorithm, which is not suitable for real-time score following. In particular, as introduced by Orio and Déchelle (2001), at each time t the system computes, for each low-level state q, the probability of being the last state of a path inside the HMM given the observations $o(1)\ldots o(t)$ until time t. This correspond to the computation of the forward variables $\alpha_i(t)$, as described by Rabiner (1989). The most probable alignment between the score and the performance at time t is then computed by the simple maximisation $\hat{q} = \mathrm{argmax}_q\, P(q, o(1)\ldots o(t))$, and the synchronisation is achieved by finding the score event i that contains state $\hat{q}$.

4.5 Training

In the context of our HMM score follower, training means adapting the various probabilities and probability distributions governing the HMM to one or more example performances such as to optimise the quality of the follower. At least two different things can be trained: the transition probabilities between the states of the Markov chain, and the PDF parameters of the observation likelihoods. In our case, the transition probabilities were trained by generating performances with errors, and adapting the transitions so that a good balance between speed of recognition and robustness to errors was reached. The observation probabilities defined in section 4.2 are quite simple in nature for our case, such that the PDF parameters can be set by hand.

5 Results and Conclusion

Tests with simple scores work perfectly well, even in the presence of errors, where the follower waits in the ghost states for the next known notes.

Our real-world test case was Philippe Manoury's chamber opera *La Frontière* for piano, electronics, and orchestra. Fortunately, during the first tests, the piano player was not yet familiar with the score, which provided us with many welcome performance errors, ranging from wrong notes to missed chords, up to complete confusion, where chords from later in the score were inserted.

These tests show a very robust performance of the follower, an example of our informal evaluation view is given in figure 8. We can see the desynchronisation of chords and many wrong notes, despite which the follower reacts very quickly. Even for the very off cases, the follower would not get lost but resynchronise itself at the next known chord in context.

There are still some problems that require further work. In particular, the preciseness of the following depends on the relationship between the expected tempo and the actual tempo of the performance. We are working on a more complex modeling of events durations, which takes into account different modeling of fast and medium-slow events. Problems of different timing becomes crucial when the score is based on the fast repetition of the same note, because in this case the system does not have other clues to state the actual position in the score other than modeling the tempo.

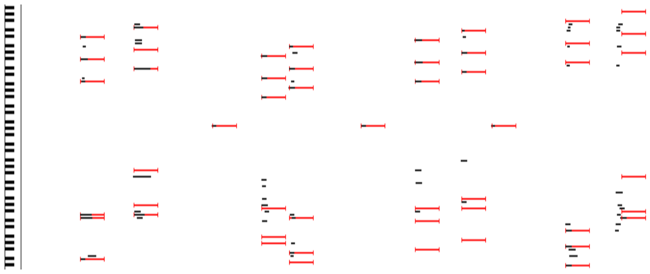

Figure 8: Score following run-time visualisation: vertically aligned chords of grey score notes with begin/end tick marks are overlaid over short black played notes.

The system is running on *j*Max and *Max/MSP* with FTM (*Faster Than Music*) (ATR 2004), and it will be used in future productions at Ircam.

6 Acknowledgements

We'd like to thank Philippe Manoury and Serge Lemouton for their support and our reviewers for their helpful remarks.

References

ATR (2004). Real-Time Applications Team/Équipe Applications Temps-Réel, Ircam—Centre Pompidou. Web page. `http://www.ircam.fr/equipes/temps-reel`

Dannenberg, R. B. (1984). An On-Line Algorithm for Real-Time Accompaniment. In *ICMC*, pp. 193–198.

Izmirli, O., R. Seward, and N. Zahler (2003). Melodic pattern anchoring for score following using score analysis. In *Proc. ICMC*, Singapore.

Loscos, A., P. Cano, and J. Bonada (1999). Score-Performance Matching using HMMs. In *ICMC*, Beijing, pp. 441–444.

Orio, N. and F. Déchelle (2001). Score Following Using Spectral Analysis and Hidden Markov Models. In *ICMC*, Havana.

Orio, N., S. Lemouton, D. Schwarz, and N. Schnell (2003). Score Following: State of the Art and New Developments. In *New Interfaces for Musical Expression (NIME)*, Montreal.

Orio, N. and D. Schwarz (2001). Alignment of Monophonic and Polyphonic Music to a Score. In *ICMC*, Havana.

Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE 77*(2), 257–285.

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

Shalev-Shwartz, S., S. Dubnov, N. Friedman, and Y. Singer (2002). Robust temporal and spectral modeling for query by melody. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 331–338.