ECAI 2008
M. Ghallab et al. (Eds.)
IOS Press, 2008

doi:10.3233/978-1-58603-891-5-241

Automatic Page Turning for Musicians via Real-Time Machine Listening

Andreas Arzt[(1)], Gerhard Widmer[(1,2)], and Simon Dixon[(3)] [1]

Abstract. We present a system that automatically turns the pages of the music score for musicians during a performance. It is based on a new algorithm for following an incoming audio stream in real time and aligning it to a music score (in the form of a synthesised audio file). Precision and robustness of the algorithm are quantified in systematic experiments, and a demonstration using an actual page turning machine built by an Austrian company is described.

1 Introduction

The musicians among the readers may be familiar with the problem of having to turn the pages of the music score while playing a piece of music on an instrument. This is not so much a problem in a live concert, where the artist either plays the piece by heart or has a (semi-)professional page turner by her side, but it is very bothersome during practicing (where it has to be done over and over again). In many cases, having to turn the page requires the musician to remove one hand from the instrument and, thus, to stop playing and then continue after the page has been turned. Such a forced disruption is annoying and frustrating, both musically and from a practicing point of view. An intelligent system that automatically 'knows' when to turn the pages and does that in a reliable manner would be highly useful.

The Austrian Company *Quidenus GmbH* (www.qidenus.com) has developed an electro-mechanical device that turns the pages of books, music scores, etc. via two physical 'fingers' (see Fig. 5, center). The device is operated by the musician via a foot switch. The musician may thus play without interruption, but is still forced to focus her thoughts on the act of page turning as the respective point in the piece approaches. Our idea was to make this device decide and act completely autonomously, without the musician having to do anything, by 'listening' to the musician in real time, comparing the ongoing performance to some internal representation of the sheet music, and automatically turning the page at the appropriate time. This is a very challenging AI task, which involves real-time machine listening and adaptivity. The contribution of this paper to Artificial Intelligence is thus a general method for tracking ('listening to') audio streams in real time, on-line, with high robustness and flexibility.

2 Requirements and Related Work

Technically, what is required is an algorithm that is capable of automatically listening to a live music performance (in the form of a raw audio stream, in our case) and tracking the current position in the score. This task is known as *automatic score following*. There has been quite some work on score following in the AI and computer music communities, starting as early as 1984 [3, 9] and intensifying in recent years (e.g., [2, 4, 7, 8]). Many of these algorithms require practising sessions with the same musician [9], during which the system learns a predictive model of the expected tempo and timing deviations applied by the musician, represented, e.g., as a Graphical Model or Bayesian Network [7], or a Hidden Markov Model [8].

Our goal is to avoid this laborious training phase and develop a system that adapts to the musician currently playing without any training. An additional goal is to provide robustness in the face of structural changes (see below), which has mostly been ignored in previous approaches (with the notable exception of [6], where HMMs are used to model the high-level structure of the music). In more detail, our system should have the following properties:

On-line tracking: The artificial page turner must 'listen to' and follow a musician's performance in real time. Specifically, in a real scenario, we cannot assume that the musician plays a MIDI instrument (in which case we would have conveniently processable symbolic input data), but, rather, a 'regular' instrument whose sound is recorded by a microphone. The problem is thus to track a raw audio stream and to align it to the music score in real time.

Robustness against changes in tempo and timing: In classical music, musicians deliberately vary the tempo (among other parameters) to add expression to a piece; this phenomenon comes under various names like 'agogics', 'rubato', or 'expressive timing'. In fact, this is an indispensable part of (classical) music performance [10]. Such tempo changes can be very abrupt and large (e.g., a slowing down of 50% within one beat). Musical scores do contain some rough indications (like a *ritardando* prescription), but these are neither precise and quantitative nor complete. A music tracking system must be able to accommodate such changes without becoming confused.

Robustness in the face of structural changes: In some cases, the musician may choose to follow structural indications in the score (specifically, *repeats*), but may also decide to ignore repeated sections. A page turning system should be able to automatically recognise the performer's decisions.

Error tolerance: Musicians make mistakes (particularly in the practise phase); they may omit notes or whole segments, play erroneous or superfluous notes, or spontaneously restart at a particular point of a piece after having made a mistake. Clearly, a page turning system should be robust to such errors.

Adaptivity: Initially, it is unclear how and how fast the musician is going to play. The system has to be able to adapt to the specific circumstances of a live performance, without prior training or information.

[1] [(1)]Department of Computational Perception, Johannes Kepler University Linz, Austria; [(2)]Austrian Research Institute for Artificial Intelligence, Vienna, Austria; [(3)]Department of Electronic Engineering, Queen Mary, University of London

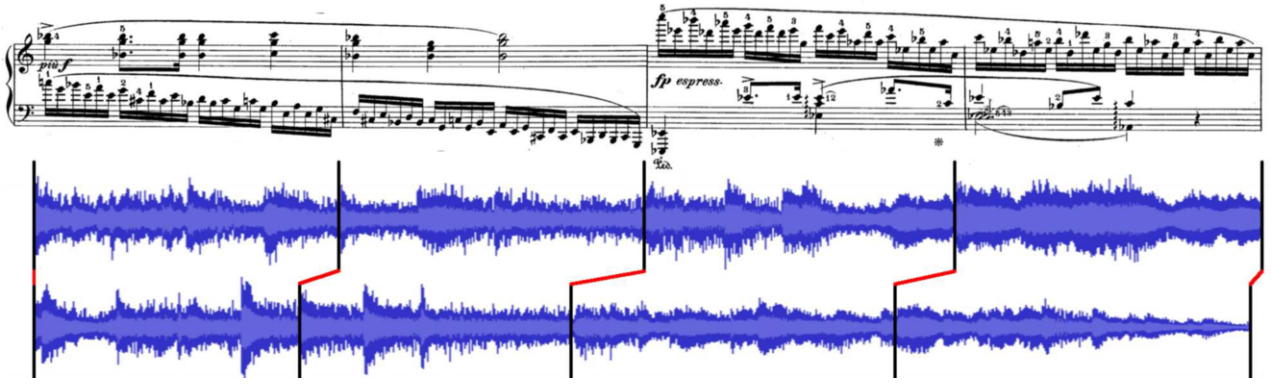

Figure 1. Excerpt (bars 43-46) of the Etude Op.25 No.11 in A minor by Frédéric Chopin: notated score (top); audio signal of synthesised reference score (middle); real performance (bottom); the correct time alignment produced by the algorithm is indicated by connecting lines. To avoid clutter, only the alignment of points at bar lines is shown.

3 On-line Audio Following

The solution we are going to adopt is the following. Rather than trying to identify individual notes from the incoming audio stream and trying to associate these with the corresponding notes in the notated score (the 'sheet music'), we first convert (a MIDI version of) the given score into a sound file, by using any available software synthesiser. That gives an audio rendition of the piece in poor sound quality, without any expressive aspects (the piece will sound mechanical), and, in the case of the piano, without any pedalling. In the live tracking process, the incoming audio stream must be aligned, on-line, to the synthesised audio file. Figure 1 shows a short excerpt from the Etude Op.25 No.11 in A minor by Frédéric Chopin, with the corresponding excerpts from the synthesised score audio file, and from an actual performance.

The algorithm to be described here builds on our on-line time-warping (OLTW) method presented in [5], and adds a number of new strategies to make it more robust and adaptive. We first recapitulate the basic algorithm and then present our new method.

3.1 The Basic Audio Alignment Algorithm

In [5], we presented an algorithm for the online alignment of two audio streams that is based on Dynamic Time Warping (DTW). The streams are given as sequences of short (46 ms) audio frames. The important differences between this algorithm and standard DTW are linear time and space complexity, and the fact that the optimal alignment is computed incrementally. The algorithm works as follows:

Given two sequences $U = u_1, ..., u_m$ and $V = v_1, ..., v_n$, an alignment between U and V is a path $W = W_1, ..., W_i$ (through a cost matrix) where each W_k is an ordered pair (i_k, j_k) such that $(i, j) \in W$ means that the points u_i and v_i are aligned. W is constrained to be monotonic and continuous. An $m \times n$ matrix represents a local cost function $d(i, j)$ which assigns costs to the alignment of each pair (u_i, v_i). The cost of a path W is the sum of the local alignment costs along the path. The $m \times n$ path cost matrix D is computed using the recursion:

$$D(i,j) = d(i,j) + min \left\{ \begin{array}{c} w_a * D(i, j-1) \\ w_a * D(i-1, j) \\ w_b * D(i-1, j-1) \end{array} \right\} \quad (1)$$

$D(i, j)$ is the cost of the minimum cost path from $(1, 1)$ to (i, j), $D(1, 1) = d(1, 1)$, $w_a = 1$ and $w_b = 2$. The weights w_a and w_b are used to normalise paths of different lengths to make them comparable. The alignment algorithm computes a quasi-optimal solution (a *'forward path'*) by incrementally constructing this cost matrix in real time. During the initial phase, as long as less than $s = 500$ elements of each series have been processed, columns and rows are calculated alternately and the path follows the diagonal of the matrix. Calculating a row (column) means incrementing the pointer to the next element of the respective time series, calculating the new local distances, and updating the cost matrix D by using formula 1.

After this initial phase the number of cells to be calculated is given by a search width parameter $c = 500$, e.g. for a new column i the local distances $d(i, j-(c-1)), d(i, j-(c-2)), ..., d(i, j)$ are calculated, where j is the index of the current row. The calculation of the minimum cost paths using formula 1 is restricted to using only calculated cells. In this way, only a sub-band of the cost matrix of constant width is computed (see Fig. 2), which reduces time and space complexity from quadratic to linear.

To decide if a row or a column should be computed (i.e., which of the two time series to advance), the minimum path cost for each cell in the current row j and column i is found. If this occurs in the current position (i, j) both a new row and column are calculated. If this occurs elsewhere in row j a new row is calculated and if this occurs elsewhere in column i a new column is calculated. If one time series has been incremented more than $MaxRunCount = 3$ times, the other series is incremented. In our musical setting, this embodies the assumption that a given performance will not be more than 3 times faster or slower than the reference score, and prevents the alignment algorithm from 'running away' too far.

The audio streams to be aligned are represented as sequences of analysis frames, using a low-level spectral representation computed via a windowed FFT of the signal with a hamming window of size 46ms and a hop size of 20ms. The data is mapped into 84 frequency bins which are spread linearly up to 370Hz and logarithmically above, with semitone spacing, and then normalised to sum up to 1. In order to emphasise *note onsets* – the most important indicators of musical timing – only the increase in energy in each bin relative to the previous frame is stored. The cost of aligning two such 84-dimensional vectors is computed as the Euclidean distance between the two vectors.

3.2 Steps Towards Intelligent Audio Following

Experiments with the original OLTW algorithm showed that it works relatively well with professional performances (e.g., recordings by famous pianists), when there are no serious performance errors – though even there we encountered some substantial alignment errors, especially in situations of rapidly changing tempo. With less perfect performances, the algorithm has severe problems.

In this paper, we propose three strategies for making on-line audio following more effective. They will be evaluated experimentally in Section 4.1. The strategies are presented here in the context of music alignment, but – with the exception of the second one – they are completely general and should prove useful in many other domains that require robust on-line sequence alignment. In the following, the completely known time series representing the score sits on the y axis of the cost matrix, the live audio stream on the x axis.

Strategy 1: The Backward-Forward Strategy

The first strategy consists in using the present hypothesis plus the information from which it was constructed, in order to re-consider past decisions and then, in turn, using the revised decisions to improve the present hypothesis.

More precisely, the method works as follows: After every 2 frames of the live input a smoothed backward path is computed, starting at the current position (i, j) of the forward path. By following this path b steps backwards on the y-axis (the score) one gets a new point which lies with a high probability nearer to the globally optimal alignment than the corresponding point of the forward path (because this backward computation takes into account information from the 'future' that was not available when computing the original forward path). Starting at this new point another forward path is computed until a border of the current matrix (either column i or row j) is reached. If this new path ends in (i, j) again, this can be seen as a confirmation of the current position. If the path ends in a column $k < i$, new rows are calculated until the current column i is reached again. If the path ends in a row $l < j$, the calculation of new rows is stopped until the current row j is reached. In our specific implementation, two different backtracking lengths are used: after 4 short backtrackings of length $b = 10$ a longer one of length $b = 50$ is performed.

The main effect of this strategy is *increased robustness against tempo changes* and *improved error tolerance*: If there are extreme tempo changes in the performance, or the performer makes large errors – plays wrong notes and repeats or omits a whole bar – the forward-backward strategy permits the system to correct the error faster by waiting for the musician or jumping forward in the score. This is because the re-computation of the backward path is not limited by the $MaxRunCount$ constraint that governs the on-line forward path computation. A situation where the system 'waits for' the performer to catch up after a serious mistake is shown in Fig. 2.

Strategy 2: Utilising Musical Information

Given that the reference audio file to which a performance is aligned was synthesised from a written score, we have additional information – beyond the pure audio representation – that can be exploited. In particular, for each note, we know precisely where it starts, i.e., we know the precise *onset times* in the score audio; this is something that is not at all obvious from the audio itself (cf. Fig. 1). The information can be used to bias the path to pass though points where the performance signal is particularly similar to the sound expected at note onsets, as follows: For every frame of the incoming live input

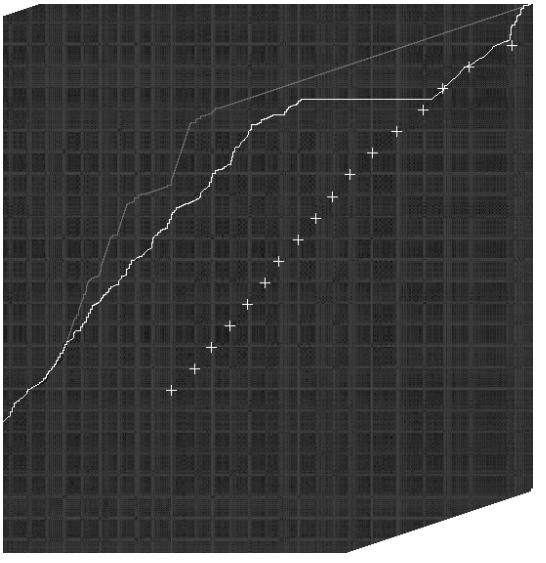

Figure 2. Part of a cost matrix (note that not the complete matrix, but only a sub-band around the diagonal is computed). This particular situation shows the system reacting to an additional bar of music (not present in the score) erroneously played by the pianist. The live performance is on the x axis, the score representation on the y axis. Crosses show the correct note onsets according to the score. The grey path is calculated by Dixon's original algorithm, the white path is our performance tracker. Note how our algorithm effectively 'waits' for the pianist (the horizontal segment) after having noticed the error. This is made possible by the Backward-Forward Strategy.

a heuristic measure M is computed that tries to capture the likelihood that the current frame corresponds to the next onset expected according to the score. If M exceeds a given threshold, the current frame is aligned with the corresponding onset frame on the score axis; otherwise, forward path computation continues as usual. The measure M combines three components: the sound similarity between the current audio frame and the score audio frame representing the next onset; a measure of "onset-ness" of the current frame (this is computed by a simple onset detection measure based on spectral differences to the previous frame); and the distance, on the y axis, of the forward path to the score coordinate of the next onset (the closer, the more likely). The details of the function are too complex to explain here (they can be found in [1]), but the idea is fairly intuitive.

The main effect of this strategy is an *increase in alignment precision*. In addition, strategy 2 also helps improve the *robustness of alignment*, particularly during hard-to-track tempo changes: the search for onsets adds the capability to catch onsets correctly even if the forward path went wrong for some frames.

Strategy 3: Maintaining Multiple Hypotheses

The third strategy is aimed directly at the 'structural changes' problem, i.e., the possibility that a musician may choose to ignore repeat signs or repeat or skip entire sections. This problem is solved in a straightforward way: Instead of using just one instance of the algorithm, up to 3 instances are started simultaneously at predefined positions and work on different parts of the piece. After 500 frames of the score representation (y) the path costs, normalised by the number of frames processed, are compared and the instance of the algorithm with the least cost is selected. The positions where this 'forking' is triggered are the boundaries of major sections as given in the score. At each section boundary, one instance of the alignment

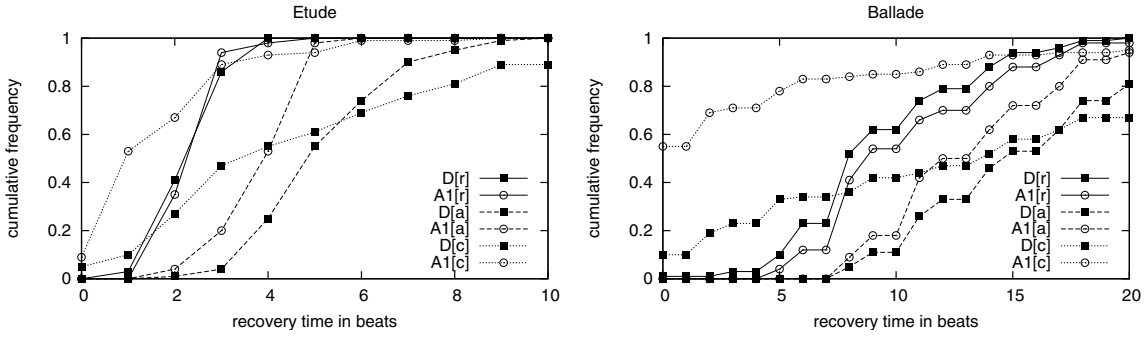

Figure 3. Recovery time in beats after big mistakes by the musician (removing a bar [r], adding a bar [a] and changing a bar [c]) as cumulative frequencies ("$y\%$ of the errors are below x beats"). The evaluation is based on 88 alignments of the Etude (4 bars changed) and 132 alignments of the Ballade (6 bars changed), for each of the cases [r,a,c].

algorithm assumes that the musician will repeat the previous section and skips back to the beginning of the section; one instance assumes the musician will continue with the next section; and a third instance assumes that the musician will skip the new section and jumps ahead to the next one. As our experiments show (see Section 4.1), this strategy works extremely well (and is still computationally feasible).

Other Improvements

Other changes that led to some improvement concern the optimisation of a few parameters. The weights in the path cost recursion were set to $w_a = 1.3$ while leaving $w_b = 2$, which makes diagonal steps cheaper. The original algorithm showed problems – e.g. uncontrolled expansions of the path in one single direction – especially between note onsets where there is no data for a reasonable alignment. That was mostly solved by this reweighting. As a consequence, the path computation now proved to be so stable that $MaxRunCount$ could be set to 6, which leads to more freedom in path computation and the possibility to catch even large differences in tempo.

And finally, with regard to the *adaptivity* problem, the described algorithm works entirely without pre-training (in contrast to, e.g., HMM-based approaches). The only information used is the score represented as a (constant tempo) MIDI file. The only change relative to Dixon's original algorithm was to shorten the initial phase (where the path is forced to follow the diagonal) to 1 second instead of 10. As a consequence, the algorithm adapts much faster to the general playing speed of the musician. From that point onwards, the above strategies ensure that the system adapts very effectively to tempo changes, delays and even insertions and deletions in the performance.

4 Experiments

4.1 Quantitative Experiments

A quantitative evaluation requires correct reference alignments. For practical reasons, the systematic experiments were performed off-line. The results are the same as for on-line alignment, except for a small latency that would occur in real-time processing. In the following we refer to Dixon's original algorithm, which serves as a reference, as *D*; to the new algorithm that uses all our improvements except strategy 2 (thus not relying on any music-specific information) as *A1*; and to the new algorithm including strategy 2 as *A2*.

The algorithms were evaluated on 2 sets of 22 piano recordings of the Etude in E major, Op.10 no.3, bars 1–21 and the Ballade Op.38, bars 1–45 by Frédéric Chopin, played on a computer-monitored grand piano by skilled pianists. The audio recordings were aligned to synthesised score audio files with constant tempo. As the computer-monitored piano also records the precise ('true') note onset times, the alignment error could then be calculated.

	Etude			Ballade		
	D	**A1**	**A2**	**D**	**A1**	**A2**
Mean Error	0.23	0.10	0.07	0.32	0.19	0.15
1st Quartile	0.02	0.02	0.02	0.04	0.02	0.02
2nd Quartile	0.08	0.04	0.02	0.08	0.04	0.02
3rd Quartile	0.26	0.12	0.04	0.26	0.10	0.06
Largest Error	3.06	2.02	2.12	9.82	7.56	7.24

Table 1. Alignment errors of the algorithms on the Etude and the Ballade. The results are based on the alignment of 3564 notes in the Etude and 4422 notes in the Ballade. The errors are given in seconds.

As Table 1 shows, both new algorithms *A1* and *A2* outperform *D* by far. Further evaluations showed that especially the reweighting towards cheaper diagonal steps improved the accuracy of *A1*. The further improvement of *A2* is due to the fact that strategy 2 is very effective at correcting errors between 0.02 and 0.2 seconds.

The excerpt of the ballade ends at a phrase boundary, which due to extreme variations in tempo and discontinuities in timing are the most problematic parts in score following. This explains the large errors on the Ballade in Table 1. After a phrase boundary the algorithm recovers easily. Nonetheless if a page-turning mark happens to be in the area of a phrase boundary this could cause a delayed or premature page-turning. Improvements on handling those boundaries are among the main goals of future work.

The new algorithms not only increased the accuracy but also decreased the variability of the results, as can be seen in Figure 4. Furthermore, there was no performance which was better aligned by *D* than by *A1* or which was better aligned by *A1* than by *A2*.

Further tests were performed to evaluate the robustness against performance errors. As it is not possible to change the performances, the score representation was changed instead. For the case of the musician leaving out notes, notes are repeated in the score, for playing additional notes, notes are deleted from the score, and for playing wrong notes, score notes are replaced by an augmented fourth.

As Figure 3 shows, the new path computation recovers much faster from mistakes than the old one, especially in the cases of adding

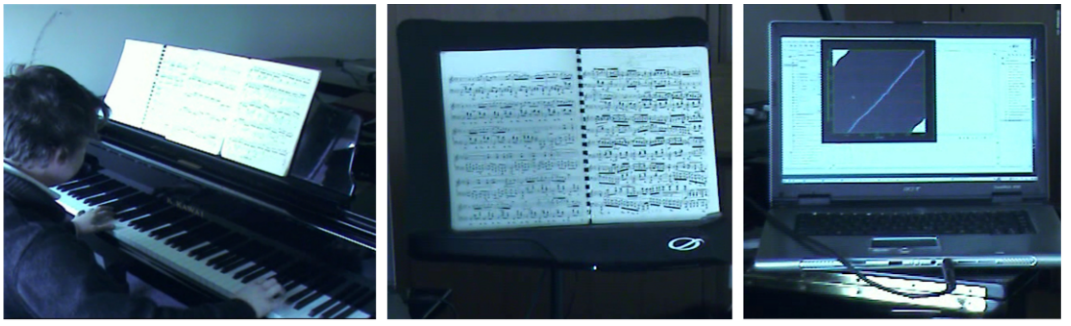

Figure 5. Some impressions from the second live experiment. Center panel: the page turning device.

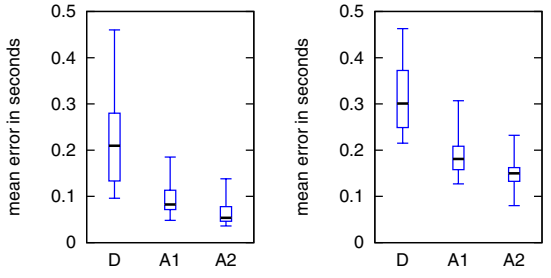

Figure 4. Variability, among 22 performances of the etude (left) and the ballade (right), of the mean errors of the alignments (shown as boxplots).

notes (due to the capability of 'waiting' for the musician, see Figure 2) and playing false notes (due to both the 'waiting' and the reweighting towards the diagonal). In correcting alignment errors due to omitted notes the performance of the algorithms is about equal.

Strategy 3, 'Considering Alternatives', was evaluated on altered scores of the Etude and the Ballade where a repeated section was inserted. For all 22 performances of both pieces, the omitted repetition was recognised and the correct path through the piece was found.

4.2 Qualitative Evaluation

To evaluate the system under realistic conditions, two live experiments were performed (for some impressions see Fig. 5). One was done with a simple electronic piano, one with a grand piano. The audio signal was recorded over the air with a single microphone. A trained pianist from our research group played two Chopin pieces: the Ballade Op.52 in A♭ major and the Etude Op.25 No.11 in A minor (cf. Fig. 1). In both tests the system worked very reliably, even in the presence of errors (and even some re-starts) by the pianist. It turned out that the more onsets are played (the faster the piece is), the better the alignment. So even the very fast etude was aligned perfectly (or at least well enough for correct page turning).

5 Conclusions and Future Work

The paper has presented a general algorithm for robust, effective online tracking of audio streams in real time, and has demonstrated its usefulness for an interesting task: automatic page turning for musicians. The algorithm may prove useful in many other score following tasks, e.g. live visualisation and automatic accompaniment.

On the technical side there are some clear directions for future work. The first concerns improvements in handling large, discontinuous tempo changes as they occur at phrase boundaries, or at the ends of pieces. This may require explicit recognition and modeling of musical structure. As the concept of multiple matchers is currently limited to fixed parts of the piece, such flexible structure models might also lead to more intelligent tracking of the performer (e.g., re-starting at a musically suitable place after a mistake).

So far, the system has only been tested on piano music. There are no fundamental results why it should not perform well on other kinds of music, though non-percussive instruments (like the violin) could be more problematic because our audio features are strongly related to note onsets. Investigations in this direction will be carried out.

ACKNOWLEDGEMENTS

This work was supported by the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung (FWF) under project number P19349-N15. Many thanks to Sebastian Flossmann for playing the piano in the live experiments.

REFERENCES

[1] A. Arzt, *Score Following with Dynamic Time Warping: An Automatic Page Turner*, Master's Thesis, University of Technology, Vienna, 2008.
[2] A. Cont, D. Schwarz, N. Schnell, and C. Raphael, 'Evaluation of Real-time Audio-to-score Alignment', in *Proc. 8th International Conference on Music Information Retrieval (ISMIR 2007)*, Vienna, (2007).
[3] R. Dannenberg, 'An On-line Algorithm for Real-time Accompaniment', in *Proceedings of the International Computer Music Conference (ICMC 1984)*, Paris, France, (1984).
[4] R. Dannenberg and N. Hu, 'Polyphonic Audio Matching for Score Following and Intelligent Audio Editors', in *Proceedings of the International Compter Music Conference (ICMC 2003)*, Singapore, (2003).
[5] S. Dixon, 'An On-line Time Warping Algorithm for Tracking Musical Performances', in *Proceedings of IJCAI 2005*, pp. 1727–1728, Edinburgh, Scotland, (2005).
[6] B. Pardo and W. Birmingham, 'Modeling Form for On-line Following of Musical Performances', in *Proceedings of the 20th National Conference on Artificial Intelligence*, Pittsburgh, Pennsylvania, (July 2005).
[7] C. Raphael, 'A Bayesian Network for Real Time Music Accompaniment', in *Neural Information Processing Systems*, 14, (2001).
[8] D. Schwarz, N. Orio, and N. Schnell, 'Robust Polyphonic Midi Score Following with Hidden Markov Models', in *Proc. of the International Computer Music Conference (ICMC 2004)*, Miami, Florida, (2004).
[9] B. Vercoe and M. Puckette, 'Synthetic Rehearsal: Training the Synthetic Performer', in *Proceedings of the International Computer Music Conference (ICMC 1985)*, Vancouver, Canada, (1985).
[10] G. Widmer, S. Dixon, W. Goebl, E. Pampalk, and A. Tobudic, 'In Search of the Horowitz Factor', *AI Magazine*, **24**(3), 111–130, (2003).