Real Time Tracking and Visualisation of Musical Expression

Simon Dixon, Werner Goebl, and Gerhard Widmer

Austrian Research Institute for Artificial Intelligence, Schottengasse 3, A-1010 Vienna, Austria.
email: {simon,wernerg,gerhard}@ai.univie.ac.at

Abstract. Skilled musicians are able to shape a given piece of music (by continuously modulating aspects like tempo, loudness, etc.) to communicate high level information such as musical structure and emotion. This activity is commonly referred to as expressive music performance. The present paper presents another step towards the automatic high-level analysis of this elusive phenomenon with AI methods. A system is presented that is able to measure tempo and dynamics of a musical performance and to track their development over time. The system accepts raw audio input, tracks tempo and dynamics changes in real time, and displays the development of these expressive parameters in an intuitive and aesthetically appealing graphical format which provides insight into the expressive patterns applied by skilled artists. The paper describes the tempo tracking algorithm (based on a new clustering method) in detail, and then presents an application of the system to the analysis of performances by different pianists.

1 INTRODUCTION

An expert musical performer is able to shape a given piece of music to communicate high level information such as musical structure and emotion. That is, the artist goes beyond what is prescribed in the written score and modifies, gradually or abruptly, the tempo or loudness or other parameters at certain places in the piece in order to achieve certain musical and emotional effects. This activity is commonly referred to as *expressive music performance*. Expressive performance is an element of central importance in art music and especially in classical music, where the performing artists have (or take) a lot of freedom in expressing their interpretation of the music and their individuality. At the same time, expressive performance is still a poorly understood phenomenon, both from a musical and a cognitive perspective. No formal models exist that would explain, or at least quantify and characterise, aspects of commonalities and differences in performance style.

This paper presents a step towards the automatic high-level analysis of this elusive phenomenon with Artificial Intelligence methods. We restrict our attention to two of the most important expressive dimensions: fluctuations in *tempo* and *loudness* (*dynamics*). A system is presented that is able to measure tempo and dynamics of a musical performance and to track their development over time. The system accepts raw audio input (e.g., from a microphone), tracks tempo and dynamics changes in real time, and displays the development of these expressive parameters in an intuitive and aesthetically

appealing graphical format which provides insight into the expressive patterns applied by skilled artists.

Measuring and tracking *dynamics* is rather straightforward. The (perceived) loudness of the music can be derived from the audio signal by applying well-known signal processing techniques and psychoacoustic principles. The difficult part is inferring the basic *tempo*, and tracking changes in tempo in real time. The main problems are detecting the onsets of notes in the raw audio signal (event detection), inferring the basic rate of beats or tempo and the most plausible metrical level (tempo induction), and the real time adaptation of the tempo hypotheses in response to newly incoming information (tempo tracking).

The main technical contribution of this paper is a real time algorithm that finds the tempo of a musical performance, keeping track of multiple hypotheses, rating and updating each of the hypotheses dynamically, and allowing the user to interactively switch between hypotheses (e.g., when the system has obviously chosen a wrong metrical level). At the heart of the tempo induction and tracking system is a fast on-line clustering algorithm for time intervals.

In the following, we describe the tempo tracking and clustering algorithm in detail, and then present an application of the algorithms in a real time visualisation system for expressive music performance. An example visualisation of two pianists playing the same piece demonstrates what kind of direct insight into expressive performance can be facilitated by this kind of system.

2 REAL TIME TEMPO TRACKING

Most music has as its rhythmic basis a series of pulses, spaced approximately equally in time, from which the timing of all musical events can be measured. This phenomenon is called the *beat*, and the individual pulses are also called beats. The rate at which beats occur defines the *tempo*, a value which varies over time. Sometimes a multiple or divisor of the tempo is perceived as an alternative tempo; these different rates are called *metrical levels*.

The task of a tempo induction and tracking system at any moment during its operation is to infer, from the observed inter-note time intervals, possible beat rates and select the one that most likely represents the perceived tempo of the piece. This is performed by a clustering algorithm which groups similar time intervals between note onsets, forming clusters which correspond to musical time units, such as half notes, quarter notes and dotted quarter notes. In a mechanical performance, these time intervals would be precisely integer or simple integer fraction multiples of the time between two consecutive beats. But in expressive performance, the categories are blurred and change over time, so the clustering algorithm must be robust to noise and able to adapt dynamically to drift in the cluster centres.

The architecture of the tempo tracker is shown in figure 1. The input signal is preprocessed in several stages to detect the onsets of musical notes (events), and this information is used by the multiple tempo tracking subsystem to create a set of tempo hypotheses which are updated dynamically as further input data arrives. The highest-ranking tempo hypothesis is given as output, but the ranking can be overridden by the

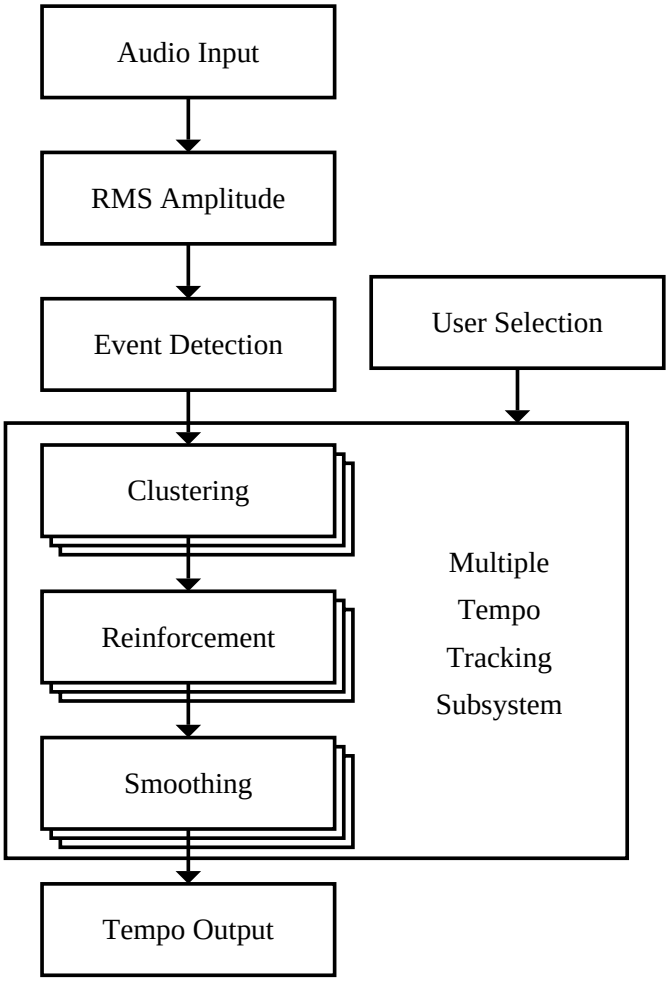

Fig. 1. Architecture of tempo tracking system

user selecting a different metrical level. In the remainder of this section, we describe the successive stages of processing in detail.

2.1 Audio Processing

The audio input is read from the soundcard or from a file, in linear PCM format. If the input has more than one channel, a single channel signal is created by averaging all channels. The resulting signal is denoted $x[n]$ and its sampling rate R.

The audio data is processed in blocks by a smoothing filter which calculates the RMS amplitude $A[k]$, where k is the integer time index expressed as a multiple of the block size. A block size of 10ms ($b = 0.01R$ samples) is used for input and processing of the signal, and this determines the time resolution of the system. The amplitude values are calculated by smoothing across a number ($h = 4$) of blocks and passed to the event detection module.

$$A[k] = (\frac{1}{hb} \sum_{i=kb}^{(k+h)b-1} x[i]^2)^{\frac{1}{2}} \tag{1}$$

The event detection module finds the slope $S[k]$ of the smoothed amplitude using an m-point linear regression. It then calculates the set $P[k]$ of local peaks of $S[k]$ which are above given thresholds T_1 of amplitude and T_2 of slope, where a local peak is defined to be a point which is maximal among the l points either side of it.

$$S[k] = \frac{4 \sum_{i=k-m+1}^{k} \frac{ibA[i]}{R} - \Big(\sum_{i=k-m+1}^{k} A[i] \Big) \Big(\sum_{i=k-m+1}^{k} \frac{ib}{R} \Big)}{4 \Big(\sum_{i=k-m+1}^{k} A[i]^2 \Big) - \Big(\sum_{i=k-m+1}^{k} A[i] \Big)^2} \tag{2}$$

$$\begin{aligned} P[k] = \{\ k-m+1 \quad & | \ S[k] = \max_{i=k-l}^{k+l}(S[i]) \quad \text{and} \\ & A[k] > T_1 \quad \text{and} \quad S[k] > T_2 \ \} \end{aligned} \tag{3}$$

These local peaks are taken to be note onset times, which are the main input to the multiple tempo tracking module, the most complex part of the system. Although a relatively simple time domain algorithm is used for event detection, it has been shown previously [3] that the accuracy is sufficient for successful extraction of tempo.

2.2 Multiple Tempo Tracking Subsystem

Clustering The tempo induction and tracking system calculates the time intervals between pairs of recent events (*inter-onset intervals*, or *IOIs*) and uses a clustering algorithm (figure 2) to find significant clusters of IOIs, which are assumed to represent musical units. These clusters form the bases of the tempo hypotheses generated by the

```
For each new onset
    For times t from 100ms to 2500ms in 10ms steps
        Find pairs of onsets which are t apart
        Sum the mean amplitude of these onset pairs
    Loop until all time points are used
        For times t from 100ms to 2500ms in 10ms steps
            Calculate window size s as function of t (see (6))
            Find average amplitude of IOIs in window [t, t + s]
            Store t which gives maximum average amplitude
        Create a cluster containing the stored maximum window
        Mark the IOIs in the cluster as used
    For each cluster
        Find related clusters (multiples or divisors)
        Combine related clusters using weighted average
    Match combined clusters to tempo hypotheses and update
```

Fig. 2. Algorithm for clustering of inter-onset intervals

system. The clustering algorithm maintains a limited memory ($M = 8$ seconds) of onset times in the set $P'[k]$, and begins processing by calculating all IOIs between pairs of onsets in its memory, weighting the intervals by the geometric mean of the amplitudes of the onsets, and summing across equally spaced onset pairs, to give the sums $I[k, i]$ for each IOI i calculated at time index (block number) k, as shown in figure 3(a).

$$P'[k] = \{j \in P[k] \quad | \quad k - j \leq \frac{MR}{b}\} \tag{4}$$

$$I[k, i] = \sum_{\substack{i_1,\, i_2 \,\in\, P'[k] \\ \text{with } i = i_1 - i_2}} \sqrt{A[i_1 + m - 1]A[i_2 + m - 1]} \tag{5}$$

At each time k, the inter-onset intervals $I[k, j]$, limited to the range 0.1s to 2.5s ($\frac{0.1R}{b} \leq j \leq \frac{2.5R}{b}$) are clustered using an iterative best-first algorithm given in figure 2, which sequentially finds the clusters with the greatest average amplitude, without reusing the data for more than one cluster.

The size $s[i]$ of the windows used in clustering is calculated as a function of the minimum IOI i in the cluster:

$$s[i] = \lfloor \frac{i}{30} + 8 \rfloor \tag{6}$$

The best clusters are given by maxima in the average weight $w[k, i]$, as shown in figure 3(b). The starting indices of the best clusters at time k are denoted $a[k, 1], a[k, 2], ...$

$$w[k, i] = \frac{\sum_{j=i}^{i+s[i]} I[k, j]}{s[i] + 1} \tag{7}$$

$$a[k, 1] = argmax_i w[k, i] \tag{8}$$

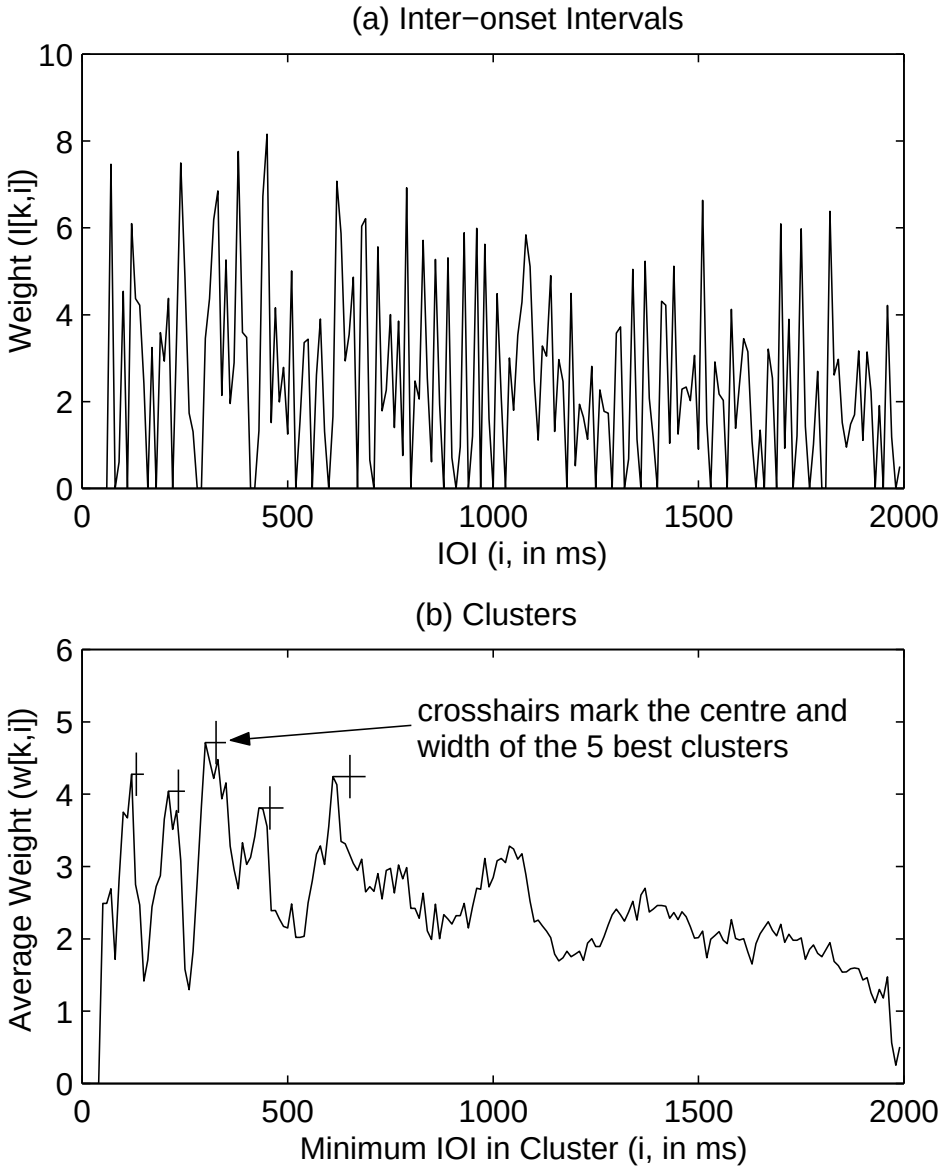

Fig. 3. (a) Example showing weighted inter-onset intervals $I[k,i]$ before clustering, for a fixed time k, and (b) Average weights $w[k,i]$ after clustering, showing the 5 best clusters as crosshairs centred at $(t[k,l], v[k,l])$ for $l = 1, 2, 3, 4, 5$, with leftmost point at $a[k,l]$ and horizontal length $s[a[k,l]]$.

The centroid $t[k,1]$ of the cluster at $a[k,i]$ is calculated as the weighted average of the component IOIs. The initial weight assigned to the cluster is denoted $v[k,1]$.

$$t[k,1] = \frac{b\sum_{j=a[k,1]}^{a[k,1]+s[a[k,1]]} jI[k,j]}{R\sum_{j=a[k,1]}^{a[k,1]+s[a[k,1]]} I[k,j]} \tag{9}$$

$$v[k,1] = w[k,a[k,1]] \tag{10}$$

The next best clusters $t[k,i]$ and their weights $v[k,i]$ for $i = 2,3,...$ are calculated by setting to 0 the values of $I[k,j]$ which have been used in a previous cluster $t[k,j]$ (where $j < i$) and repeating the above calculations.

Reinforcement It is usually the case in traditional Western music that time intervals are approximately related by small integer ratios, and therefore the cluster times $t[k,i]$ are expected also to reflect this property. In other words, the cluster times are not independent; they represent related musical units such as quarter notes and half notes. The tempo inducer exploits this property by recalculating the cluster times and weights based on the combined information given by sets of related clusters. Two clusters are said to be related if the ratio of their time intervals $\frac{t[k,i]}{t[k,j]}$ is close to an integer. More formally, the set of clusters $r[k,i]$ related to $t[k,i]$ is defined as follows.

$$f(x,y) = \begin{cases} y/x & x \leq y \\ x/y & x > y \end{cases} \tag{11}$$

$$d(x,y) = |f(x,y) - round(f(x,y))| \tag{12}$$

$$d'(x,y) = \frac{d(x,y)}{f(x,y)} \tag{13}$$

$$r[k,i] = \Big\{ t[k,j] \mid d'(t[k,i],t[k,j]) < 0.1 \quad \text{and} \\ 2 \leq round(f(t[k,i],t[k,j])) \leq 8 \Big\} \tag{14}$$

The errors of individual cluster times are reduced by bringing the related times closer to the ideal integer ratios. To achieve this, the related clusters are scaled to the same metrical level and a weighted average of the scaled clusters is calculated. The weighting favours longer time intervals, which tend to have a lower relative error. Formally, the updated clusters $t'[k,i]$ (with weights $v'[k,i]$) are calculated as follows.

$$f'(x,y) = \begin{cases} y/x & x \leq y \\ 1 & x > y \end{cases} \tag{15}$$

$$t'[k,i] = \frac{\sum_{t[k,j] \in r[k,i]} t[k,j]v[k,j]/f'(t[k,i],t[k,j])^2}{\sum_{t[k,j] \in r[k,i]} (v[k,j]/f(t[k,i],t[k,j]))} \tag{16}$$

$$v'[k,i] = \sum_{t[k,j] \in r[k,i]} (v[k,j] / f(t[k,i], t[k,j])) \tag{17}$$

Smoothing Tempo as a percept arises from the timing of many notes; local changes in timing do not unambiguously imply a tempo change. In order that the system is not disrupted by local timing irregularities, like the delay of a single note, the tempo tracker performs smoothing on the tempo hypotheses generated above. The tempo hypotheses $t'[k,i]$ at time step k, ranked by the corresponding weights $v'[k,i]$, are combined with historical values from the previous time step $k-1$ to give updated hypotheses $t''[k,i]$. Each current hypothesis $t'[k,i]$ is matched to the nearest $t''[k-1,j]$ (if a sufficiently near one exists), and updated according to a formula which causes old values to decay exponentially as they are replaced by new. If the decay factor is γ, the updated value is:

$$t''[k,i] = \gamma t''[k-1,j] + (1-\gamma) t'[k,i] \tag{18}$$

Although it would be possible to keep track of all hypotheses, it is sufficient for the system to keep track of the best 10 hypotheses; no change in behaviour is noted when more hypotheses are tracked. Sometimes hypotheses are found to be duplicating the same metrical level, in which case they are merged into a single hypothesis.

3 APPLICATION: A WORM WITH EARS

The above algorithm, together with an algorithm that computes the dynamics (loudness) of a performance from the audio signal, has been implemented in a system that tracks the tempo and dynamics in a given performance and shows the parameters (current values plus part of their history) in an animated display. The idea for this kind of visual representation was originally developed by the musicologist Jörg Langner [7]. As with tempo (equation 18), the dynamics trajectory is smoothed over the past via an exponential decay function. The system takes its input from an audio file or directly from the sound card and works in real time. For reasons that are evident (see figure 4), we call it the *Performance Worm*.

The Worm works interactively. The user can dynamically modify presentation parameters (e.g., rescale the axes) and, what is more, switch between tempo hypotheses, i.e., force the tempo tracker to re-weight its clusters and move to a higher or lower metrical level.

In this section, we briefly present an example of the Worm's output to give an impression of the kind of insight offered by such an animation. Figure 4 shows a snapshot of the Worm as it tracks the performances of the same piece by two famous pianists. The piece is W.A. Mozart's piano sonata K.279 (C major), first section of second movement (*Andante*), and the performances analysed are by Daniel Barenboim and András Schiff.[1] In the plots, the x axis represents the tempo in beats per minute (bpm), the y axis the loudness in terms of sound pressure level (measured in decibels). The darkest

[1] Daniel Barenboim, Mozart: The Piano Sonatas, EMI Classics, 767 294-2, recorded 1985; András Schiff, Mozart: The Piano Sonatas, DECCA, 443 717-2, recorded 1980.

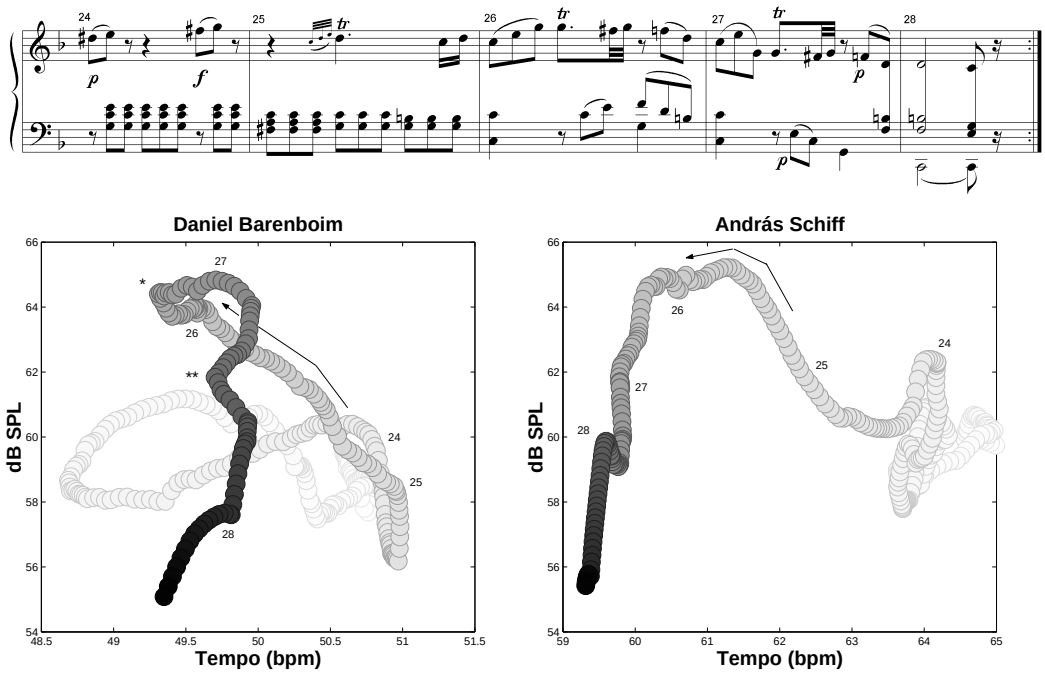

Fig. 4. Expression trajectories over the last bars (mm.24–28) of the Mozart piano sonata K.279, second movement, first section, as played by Daniel Barenboim (left) and András Schiff (right). x axis: tempo in beats per minute; y axis: dynamics ('loudness') in decibel. The darkest point represents the current instant (third beat of m.28), while instants further in the past appear fainter.

point represents the current instant, while instants further in the past appear fainter. We see the respective trajectories as they look when the performers have reached the very end of the section (beat 3 of measure 28). In the following, we will look at what these trajectories tell us about how the pianists played the last five bars of the piece (measures 24–28); the relevant part of the musical score is shown in figure 4 (top), and the positions in the trajectory that correspond to the beginnings of bars (24–28) have been manually marked in the plots, to make the following discussion easier. Note that the Worm knows nothing about the score or bar numbers.

Many interesting patterns emerge that reveal both commonalities and differences between the two performers. Some of these are clearly audible in the recording, others are hard to pinpoint and name when we hear the piece, but clearly emerge from the graphical representation (and contribute to the overall impression of the performance).

The most fundamental *similarity* between the two trajectories is the general leftward tendency in the tempo dimension (i.e., a slowing down) over the last 5 bars, combined with a strong downward movement (a reduction of loudness or *decrescendo*) over the last two bars. This is a well-known performance strategy for closing a piece; everything else would probably sound unmusical to us.

More interesting are the differences. The first striking difference is that Schiff plays the entire section much faster (above 60 bpm) than Barenboim (around 50 bpm). In fact, Schiff's trajectory lingers around the 65 bpm mark for most of the piece and only moves left towards the 60 bpm mark in the final 5 bars of the piece — a very strong way of closing the piece by a long-term, gradual final *ritardando* that clearly sets the ending apart from the rest of the performance. Barenboim, on the other hand, spends most of his time (at least the last 8 or 9 bars) in the range of 50 bpm and also ends the piece there.

Another difference, again in the tempo dimension, is in the structuring of this final slowing down. Starting from m. 24, Schiff applies a continuous *ritardando* with no interruptions — his trajectory moves steadily to the left —, while Barenboim adds micro-structure to his *ritardando* by interjecting two little speedups at two musically similar points: the third beat of m. 26 (*) and the third beat of m. 27 (**).

In the dynamics dimension, an interesting difference is in the placement of the last rise in volume (*crescendo*) before the closing drop: Barenboim performs this *crescendo* during the *trill* in beat 2 of m. 25, while Schiff builds up the most of the volume (and the tension) before the trill (see the arrows in figure 4, which indicate the durations of the trill). On the other hand, both artists combine this buildup in volume with a marked slowing down (the trajectories move from lower right to upper left). This might again represent a common performance strategy.

Many more interesting observations could be made. But this is not the place for an exhaustive musical analysis. What the example is meant to illustrate is how this approach to visualisation provides an intuitive view of a number of high-level aspects of expressive music performance. With a little experience, one immediately sees many interesting and typical features and patterns in a given trajectory. That makes the Worm an extremely useful tool for musical performance analysis.

4 DISCUSSION

We have described a tempo tracking algorithm that extracts potential note onsets from audio input and estimates the current tempo of a piece of music in real time, and the application of this algorithm in a system for the visualisation of musical expression.

Early work in tempo and beat tracking focussed mainly on the converse of expression extraction, that is rhythm parsing, where deviations from metrical time were either not considered (e.g. [9]) or treated as noise (e.g. [10, 2, 11, 1]), and the systems processed symbolic data off-line. More recently, several beat tracking systems have been developed which work with audio input (e.g. [12, 4]) or run in real time (e.g. [8]) or both (e.g. [5, 6]). Compared with the real time audio beat tracking work of [5, 6], our tempo tracking algorithm performs a simpler task, that of finding the tempo but not necessarily the beat. However, our work is not restricted to a particular musical style or tempo, whereas Goto's work is restricted to function only for popular music in 4/4 time, with a tempo range of 61–120 beats per minute, where either the drum patterns or the harmonic changes match assumed rhythmic patterns typical of pop music.

The values of parameters used in this paper were determined empirically, and are not necessarily optimal. In general, optimal values for parameters will depend on the data being processed. For example, the onset detection parameters depend on the instruments used; current values assume a reasonably sharp onset at the beginning of each tone. The tempo tracking parameters can be adjusted to suit different levels of rhythmic complexity and extents of tempo variation; the values cited here appear to work well with small to moderate tempo variations and any level of rhythmic complexity. We note that these two characteristics are interdependent, since by allowing greater variation in tempo, the system becomes more sensitive to rhythmic complexity, that is, more likely to interpret a complex rhythm as a change (or series of changes) in tempo. In further work, we intend to extend the system to allow input of high level (e.g. score) information, to make the system less dependent on parameter settings.

The expression tracking and visualisation system has a variety of interesting applications. For instance, it could be used for didactic purposes, for the visualisation and analysis of students' performances (e.g., in conservatories). Another interesting practical application would be in the automatic synchronisation of the music with other presentation aspects like lighting, videos, animations, etc. in live stage productions. Here, real time capabilities are essential. Note that our algorithms are by no means restricted to classical (or even tonal) music. In fact, they make no assumptions whatsoever regarding the type of music or instruments they are dealing with, except that the notion of 'tempo' has to be applicable (i.e., there has to be some regular, recognisable rhythmic element).

A third application — and that is what we are using it for — is in the visualisation and analysis of the performance style of famous artists. We are currently starting a large-scale study on the typical style of various famous pianists, in an attempt to quantify and characterise at least some aspects of what has so far been discussed by musicologists and music lovers only in rather vague and aesthetic terms: what is it that distinguishes one great artist from another — what makes a Horowitz a Horowitz, to speak with [13]. This, we hope, will make an interesting contribution of AI to the world of music.

ACKNOWLEDGEMENTS

This research is part of the project Y99-INF, sponsored by the Austrian Federal Ministry of Education, Science and Culture (BMBWK) in the form of a START Research Prize. The BMBWK also provides financial support to the Austrian Research Institute for Artificial Intelligence.

References

[1] A.T. Cemgil, B. Kappen, P. Desain, and H. Honing, 'On tempo tracking: Tempogram representation and Kalman filtering', in *Proceedings of the 2000 International Computer Music Conference*, pp. 352–355, San Francisco CA, (2000). International Computer Music Association.

[2] P. Desain and H. Honing, 'Quantization of musical time: A connectionist approach', *Computer Music Journal*, **13**(3), 56–66, (1989).

[3] S. Dixon, 'Automatic extraction of tempo and beat from expressive performances', *Journal of New Music Research*, **30**(1), 39–58, (2001).

[4] S. Dixon and E. Cambouropoulos, 'Beat tracking with musical knowledge', in *ECAI 2000: Proceedings of the 14th European Conference on Artificial Intelligence*, pp. 626–630, Amsterdam, (2000). IOS Press.

[5] M. Goto and Y. Muraoka, 'A real-time beat tracking system for audio signals', in *Proceedings of the International Computer Music Conference*, pp. 171–174, San Francisco CA, (1995). International Computer Music Association.

[6] M. Goto and Y. Muraoka, 'Real-time beat tracking for drumless audio signals', *Speech Communication*, **27**(3–4), 331–335, (1999).

[7] J. Langner and W. Goebl, 'Representing expressive performance in tempo-loudness space', in *Proceedings of the ESCOM 10th Anniversary Conference on Musical Creativity*, Liège, Belgium, (2002).

[8] E.W. Large and J.F. Kolen, 'Resonance and the perception of musical meter', *Connection Science*, **6**, 177–208, (1994).

[9] C.S. Lee, 'The perception of metrical structure: Experimental evidence and a model', in *Representing Musical Structure*, eds., P. Howell, R. West, and I. Cross, 59–127, Academic Press, San Diego CA, (1991).

[10] H.C. Longuet-Higgins, *Mental Processes*, MIT Press, Cambridge MA, 1987.

[11] D. Rosenthal, 'Emulation of human rhythm perception', *Computer Music Journal*, **16**(1), 64–76, (1992).

[12] E.D. Scheirer, 'Tempo and beat analysis of acoustic musical signals', *Journal of the Acoustical Society of America*, **103**(1), 588–601, (1998).

[13] G. Widmer, 'What is it that makes it a Horowitz? Empirical musicology via machine learning', in *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-96)*, Chichester, UK, (1996). Wiley & Sons.