The Performance Worm: Real Time Visualisation of Expression based on Langner's Tempo-Loudness Animation

Simon Dixon, Werner Goebl and Gerhard Widmer
Austrian Research Institute for Artificial Intelligence, Vienna
email: {simon,wernerg,gerhard}@oefai.at

Abstract

In an expressive performance, a skilled musician shapes the music by continuously modulating aspects like tempo and loudness to communicate high level information such as musical structure and emotion. Although automatic modelling of this phenomenon remains beyond the current state of the art, we present a system 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.

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*. Despite its importance in art music, 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. We restrict our attention to two of the most important expressive dimensions: *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.

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 quantity 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 infor-

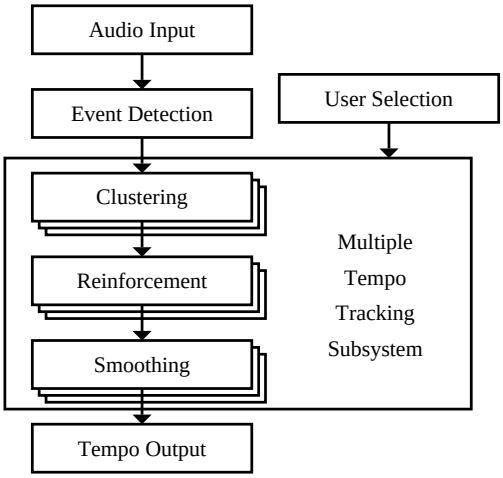

Figure 1: Architecture of tempo tracking system

```
For each new onset
    For IOI 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 IOI time points are used
        For times t from 100ms to 2500ms in 10ms steps
            Calculate window size s as function of t
            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
```

Figure 2: Algorithm for clustering of inter-onset intervals

mation 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 user selecting a different metrical level. In the remainder of this section, we describe the successive stages of processing. Full details can be found in Dixon et al. (2002).

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 data is then processed by a smoothing filter which calculates the RMS amplitude for 40ms windows with 75% overlap, giving the system a 10ms time resolution in all subsequent calculations.

The event detection module then finds the slope of the smoothed amplitude using a linear regression, and calculates the set of local peaks which are above given thresholds of amplitude and slope. 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 (Dixon, 2001) that this approach is sufficient for successful extraction of tempo.

2.2 Multiple Tempo Tracking Subsystem

2.2.1 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 system. The clustering algorithm maintains a limited memory (5 seconds) of onset times, and begins processing by calculating all IOIs between pairs of onsets in its memory. The intervals are weighted by the geometric mean of the amplitudes of the onsets, and summed for each time interval, to give the weighted IOIs shown in figure 3(a).

At each time interval, the inter-onset intervals (limited to the range 0.1s to 2.5s) are clustered using an iterative best-first algorithm given in figure 2, which sequentially finds the clusters with the greatest average amplitude, using a sliding window which grows proportionally with IOI size. For each of the best clusters, the time interval represented by the cluster (*cluster time*) is calculated as the weighted average of the component IOIs (figure 3(b)).

2.2.2 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 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 adjusting 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 is close to an integer.

Thus the error in each individual cluster time can be reduced by bringing the cluster time and related cluster 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.

2.2.3 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

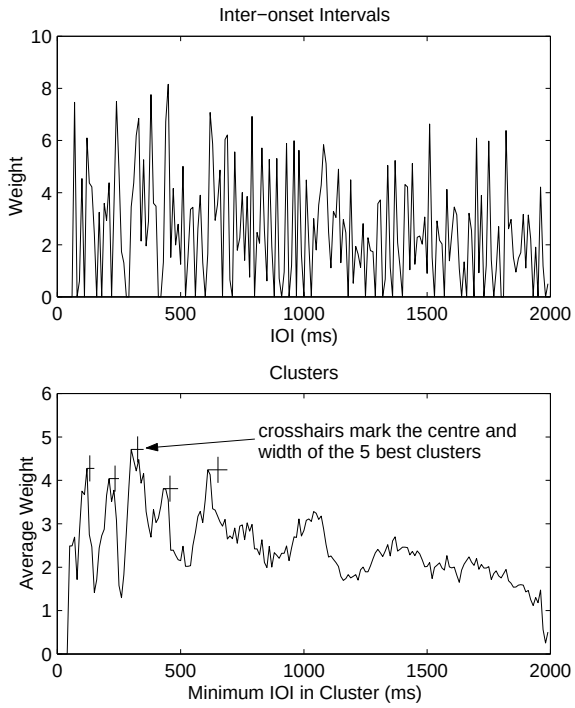

Figure 3: (a) Weighted inter-onset intervals before clustering and (b) results of clustering showing the 5 best clusters

tempo tracker performs smoothing on the tempo hypotheses generated above. At each time step, the hypotheses from the previous time step are updated by combining with new values. This is done by matching each current hypothesis to the nearest previous hypothesis (if a sufficiently near one exists), and updating using a recursive (one-pole) filter, which causes old values to decay exponentially as they are replaced by new.

3 THE PERFORMANCE WORM

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. As with tempo, 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*. This representation is based on an offline tempo-loudness animation developed by musicologist Jörg Langner (Langner and Goebl, 2002).

The Worm works interactively. The user can dynamically modify presentation parameters (e.g., rescaling the axes) and switch between tempo hypotheses, i.e., force the tempo tracker to re-weight its clusters and move to a higher or lower metrical level. Figure 4 shows a snapshot of the Worm as it tracks a performance (by Barenboim) of Mozart's

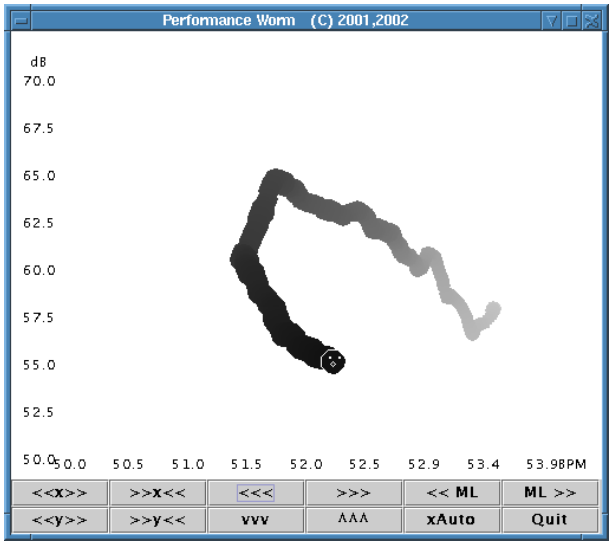

Figure 4: Screen shot of the Performance Worm, showing an expression trajectory with x axis: tempo in beats per minute; y axis: dynamics ('loudness') in decibel. The darkest point represents the current instant, while instants further in the past appear fainter.

piano sonata K.279 (C major, 2nd movt). 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 point represents the current instant, while instants further in the past appear fainter. In the example, we see a simultaneous crescendo and ritardando, followed by a decrescendo, with the ritardando continuing for a short time into the decrescendo before the tempo starts to increase towards its previous rate.

Observing trajectories such as in figure 4, many interesting patterns emerge that reveal characteristics of performances. 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). What this 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 musical expression visualisation system.

Previous work in tempo and beat tracking has focussed primarily on the converse of expression extraction, that is rhythm parsing, where deviations from metrical time were

either not considered (e.g. Lee, 1991) or treated as noise (e.g. Desain and Honing, 1989; Rosenthal, 1992; Cemgil et al., 2000), and the systems processed symbolic data offline. More recently, several beat tracking systems have been developed which work with audio input (e.g. Scheirer, 1998; Dixon, 2001) or run in real time (e.g. Large and Kolen, 1994) or both (e.g. Goto and Muraoka, 1995, 1999). Compared with the real time audio beat tracking work of Goto and Muraoka, 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 approach to visualising expressive performance in the form of an animation in a 2-D tempo-loudness space comes from Langner and Goebl (2002). The system differs from Langner's in several important ways. The primary difference is that it works in real time, which means that the Worm does not use information from the future. Langner's system smooths the tempo and dynamics using a large Gaussian window, which requires looking several seconds into the future. This has the side effect of the display sometimes anticipating the performer. On the other hand, the Performance Worm displays a more local (i.e. less smoothed) representation, which, due to the smoothing with data only from the past, sometimes lags behind the performance. Langner's system also requires knowledge of the musical score and the precise onset times of each note, information which is not available to the Performance Worm.

The Performance Worm 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 music with other presentation aspects like lighting, videos, animations, etc. in live 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 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 Widmer (1996).

ACKNOWLEDGMENTS

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

REFERENCES

Cemgil, A., B. Kappen, P. Desain, and H. Honing (2000). On tempo tracking: Tempogram representation and Kalman filtering. In *Proceedings of the 2000 International Computer Music Conference*, San Francisco CA, pp. 352–355. International Computer Music Association.

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

Dixon, S. (2001). Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research 30*(1).

Dixon, S., W. Goebl, and G. Widmer (2002). Real time tracking and visualisation of musical expression. Technical Report 2002–04, Austrian Research Institute for Artificial Intelligence.

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

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

Langner, J. and W. Goebl (2002). Representing expressive performance in tempo-loudness space. In *ESCOM 10th anniversary conference on Musical Creativity*, Liège, Belgium.

Large, E. and J. Kolen (1994). Resonance and the perception of musical meter. *Connection Science 6*, 177–208.

Lee, C. (1991). The perception of metrical structure: Experimental evidence and a model. In P. Howell, R. West, and I. Cross (Eds.), *Representing Musical Structure*, pp. 59–127. San Diego CA: Academic Press.

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

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

Widmer, G. (1996). 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. Wiley & Sons.