On the analysis of musical expression in audio signals

Simon Dixon

Austrian Research Institute for Artificial Intelligence, Vienna, Austria

ABSTRACT

In western art music, composers communicate their work to performers via a standard notation which specificies the musical pitches and relative timings of notes. This notation may also include some higher level information such as variations in the dynamics, tempo and timing. Famous performers are characterised by their expressive interpretation, the ability to convey structural and emotive information within the given framework. The majority of work on audio content analysis focusses on retrieving score-level information; this paper reports on the extraction of parameters describing the performance, a task which requires a much higher degree of accuracy. Two systems are presented: BeatRoot, an off-line beat tracking system which finds the times of musical beats and tracks changes in tempo throughout a performance, and the Performance Worm, a system which provides a real-time visualisation of the two most important expressive dimensions, tempo and dynamics. Both of these systems are being used to process data for a large-scale study of musical expression in classical and romantic piano performance, which uses artificial intelligence (machine learning) techniques to discover fundamental patterns or principles governing expressive performance.

Keywords: Beat tracking, musical expression, content analysis, digital audio

1. INTRODUCTION

This paper describes recent work in extracting the expressive aspects of musical performances from digital audio. That is, we seek to measure how a skilled performer shapes a given piece of music in order to create emotive effect beyond the expressive content due to the composition itself, which does not vary from performance to performance. In non-improvised music, the score provides a framework constraining but not fully determining the performance, which the musicians then instantiate by continuously adjusting various parameters of the music in order to convey high-level information such as musical structure and emotion. Although expression is necessarily contained in the physical features of the audio signal, such as amplitudes, frequencies and onset times, it is better understood when viewed from a higher level of abstraction, that is, in terms of musical constructs such as tempo, rhythm, pitch and timbre, which are not directly measurable. This paper addresses the problem of automatically computing these expressive dimensions from audio data, as a first step towards the understanding of expressive performance with artificial intelligence techniques. We describe two systems: BeatRoot,[1,2] a beat tracking system, which finds the times of musical beats in a performance, and the Performance Worm,[3] a system for automatic real-time visualisation of musical expression. These systems have been developed to extract expressive information from recordings, and as such have been found to be useful in musicological research.[3–5]

One of the most important expressive dimensions in musical performance is time. In most Western music, timing is expressed in terms of a (quasi-)regular pulse, the beats of which are like landmarks from which all other timing is derived. Musical time can be considered in terms of the rate of the pulse (tempo) and the placement of individual notes relative to the pulse (timing), and expressive effect is created by lengthening, shortening, delaying or anticipating notes relative to their expected time and duration based on the current pulse. The extraction of this pulse from an audio signal, known as beat tracking, is a non-trivial problem. Several systems have been proposed to perform this task, and varying degrees of success are reported for particular types of music. BeatRoot uses a multiple agent architecture which simultaneously considers several different hypotheses concerning the rate and placement of musical beats, resulting in accurate tracking of the beat, quick recovery from errors, and graceful degradation in cases where the beat is only weakly implied by the data.

Address for correspondence: ÖFAI, Schottengasse 3, A-1010 Vienna, Austria. Email: `simon@oefai.at`

In a second system, the Performance Worm, tempo is displayed simultaneously with another important expressive dimension, dynamics, in an intuitive and aesthetically appealing graphical format which provides insight into the expressive patterns applied by skilled artists. Unlike the tempo calculation, it is a trivial process to calculate the physical energy of a signal at any time, and several implementations of psychoacoustic models are available to compute the perceived loudness of the signal, if this is considered more relevant than its physical correlate. In recent work, the German musicologist Langner developed an animation for visualising the interactions between tempo and loudness.[6] These parameters are plotted in a two-dimensional space, which is animated in time with the music, the plotted points describing a performance trajectory which reflects the musician's expressive choices. The Performance Worm provides an implementation of this idea, which can work off-line with prepared data, or in real-time, taking audio input directly from a microphone or line input. The real-time system is based on a real-time tempo induction algorithm using a new clustering algorithm, which keeps track of multiple tempo hypotheses, updating them dynamically, and allowing the user to select the desired metrical level at any time.

In the remainder of the paper, we provide a brief review of research in audio analysis relating to music, and then describe the BeatRoot and Performance Worm systems, concluding with a description of our current and planned future work in the area of music signal analysis.

2. BACKGROUND

Musical audio analysis has a long research history, which is reviewed quite thoroughly by Roads.[7] The problems that have received the most attention are pitch detection, spectral analysis and rhythm recognition, areas which correspond respectively to the three most important features of music: melody, harmony and rhythm.

Pitch detection is the estimation of the fundamental frequency of a signal, usually assuming it to be monophonic. Methods include: time domain algorithms such as counting of zero-crossings and autocorrelation; frequency domain methods such as Fourier analysis and the phase vocoder; and auditory models which combine time and frequency domain information based on an understanding of human auditory processing. Many of these methods are not immediately relevant to music signals, since they assume the signal is monophonic, but the recent surge of interest in music information retrieval has revived research into finding the most salient pitch components in signals.

Spectral analysis has been researched in great depth by the signal processing community, and many algorithms are available which are suitable for various classes of signals. The short time Fourier transform is the best known of these, but other techniques such as wavelets and more specialised time-frequency distributions are also used.

The problem of extracting rhythmic content from a musical performance, and in particular finding the rate and temporal location of musical beats, has attracted considerable interest in recent times.[1,8–17] Previous work had concentrated on rhythmic parsing of musical scores, lacking the tempo and timing variations that are characteristic of performed music, but in the last few years, these restrictions have been lifted, and tempo and beat tracking systems have been developed that work successfully on a wide range of performed music.

Despite these developments, the field of performance research is yet to experience much benefit from automatic analysis of audio; in most cases, general purpose signal visualisation tools combined with human judgement have been used to extract performance parameters from audio data. Only recently are systems being developed which automatically extract performance data from audio signals.[18,19] The main problem in music signal analysis is the development of algorithms to extract sufficiently high level content from audio signals. The low level signal processing algorithms are well understood, but they produce inaccurate or ambiguous results, which can be corrected given sufficient musical knowledge, such as that possessed by a musically literate human listener. This type of musical intelligence is difficult to encapsulate in rules or algorithms that can be incorporated into computer programs.

3. BEATROOT

Compared with complex cognitive tasks such as playing chess, beat tracking (identifying the basic rhythmic pulse of a piece of music) does not appear to be particularly difficult, as it is performed by people with little

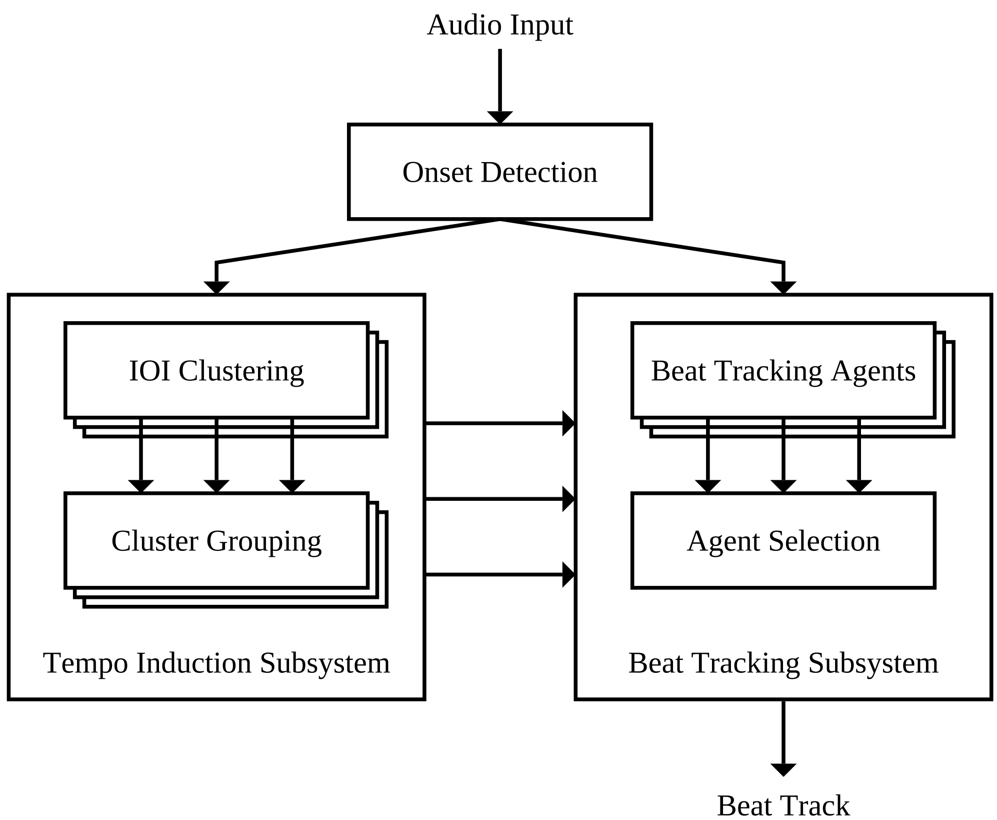

Figure 1. System architecture of BeatRoot

or no musical training, who tap their feet, clap their hands or dance in time with music. However, while chess programs compete with world champions, no computer program has been developed which approaches the beat tracking ability of an average musician, although recent systems are approaching this target. In this section, we describe BeatRoot, a system which estimates the rate and times of musical beats in expressively performed music. A full description of the system can be found elsewhere.[1, 2]

BeatRoot models the perception of beat by two interacting processes: the first finds the rate of the beats (*tempo induction*), and the second synchronises a pulse sequence with the music (*beat tracking*). A clustering algorithm finds the most significant metrical units, and the clusters are then compared to find reinforcing groups, and a ranked set of tempo hypotheses is computed. Based on these hypotheses, a multiple agent architecture is employed to match sequences of beats to the music, where each agent represents a specific tempo and alignment of beats with the music. The agents are evaluated on the basis of the regularity, continuity and salience of the onsets corresponding to hypothesised beats, and the highest ranked beat sequence is returned as the solution. The user interface presents a graphical representation of the music and the extracted beats, and allows the user to edit and recalculate results based on the editing.

BeatRoot takes as input either digital audio or symbolic music data such as MIDI. This data is processed off-line to detect salient rhythmic events, and the timing of these events is analysed to generate hypotheses of the tempo at various metrical levels. The stages of processing for audio data are shown in Figure 1, and will be described in the following subsections.

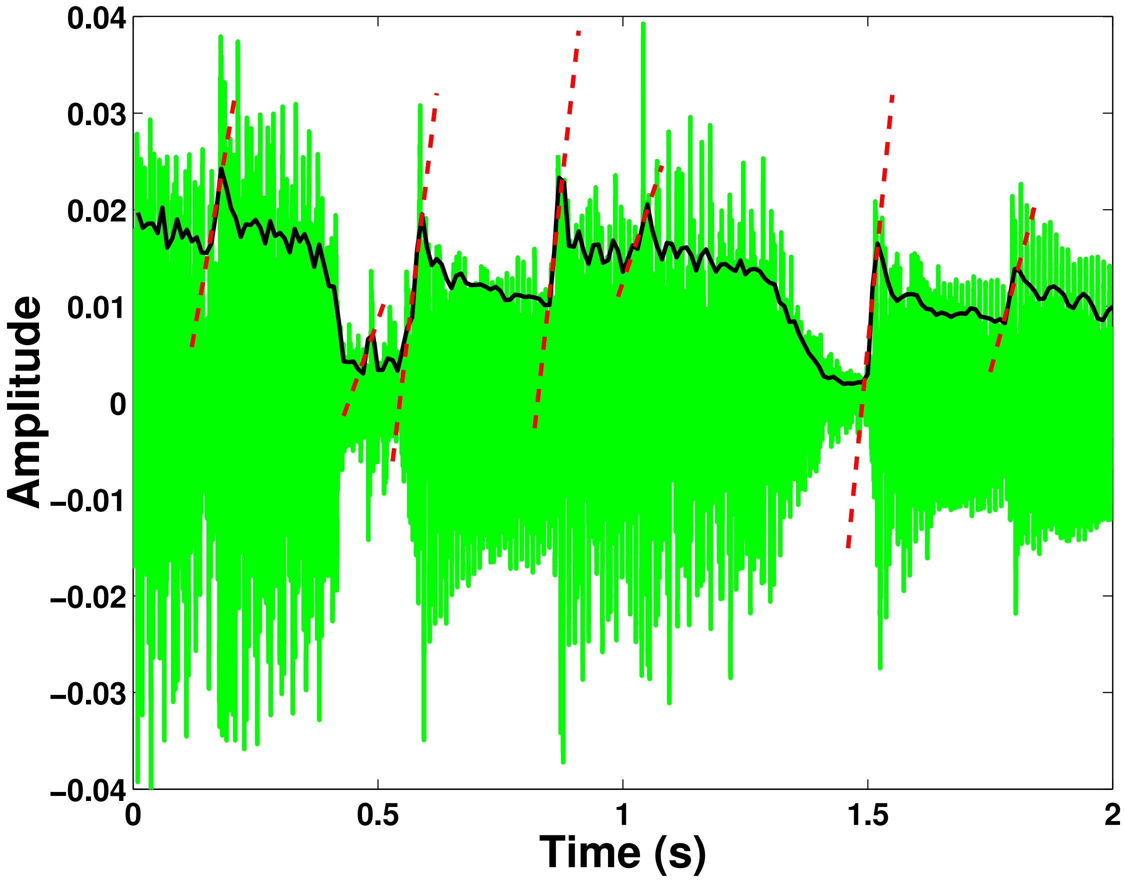

Figure 2. "Surfboard" method of onset detection, showing the audio signal in light grey, the smoothed signal (amplitude envelope) in black, and the detected onsets as dashed dark grey lines

3.1. Onset Detection

Initial processing of the audio signal is concerned with finding the onsets of musical notes, which are the primary carriers of rhythmic information. Earlier versions of BeatRoot used a time-domain onset detection algorithm, which finds local peaks in the slope of a smoothed amplitude envelope. This method is particularly well suited to music with drums, but less reliable at finding onsets of other instruments in a polyphonic setting. Nevertheless, it has been shown that this method is sufficient in many cases for finding the beat (which does not require knowledge of every onset time). More recently, this method was extended by combining features from various frequency bands using a method which was optimised with a genetic algorithm.[20] This improved the accuracy of note onset detection considerably, at the cost of greater processing time. We now describe these two methods.

For many instruments, the note onset can be indentified by a sharp increase in energy in the frequency bands associated with the note and its harmonics. For percussion instruments such as piano, guitar and drums, the attack is sharp enough that it can often be detected in the time domain signal, making possible an extremely fast onset detection algorithm. This algorithm is based on the "surfboard" method of Schloss,[8] which involves smoothing the signal to produce an amplitude envelope and finding peaks in its slope using linear regression. Figure 2 shows the original signal with the smoothed amplitude envelope drawn in bold over it, and the peaks in slope shown by dotted lines tangential to the envelope.

This method is lossy, in that it fails to detect the onsets of many notes which are masked by simultaneously sounding notes. Also, occasional false onsets are detected, such as those caused by amplitude modulation in the signal. The tempo induction and beat tracking algorithms, which are designed to be robust to noise, cope

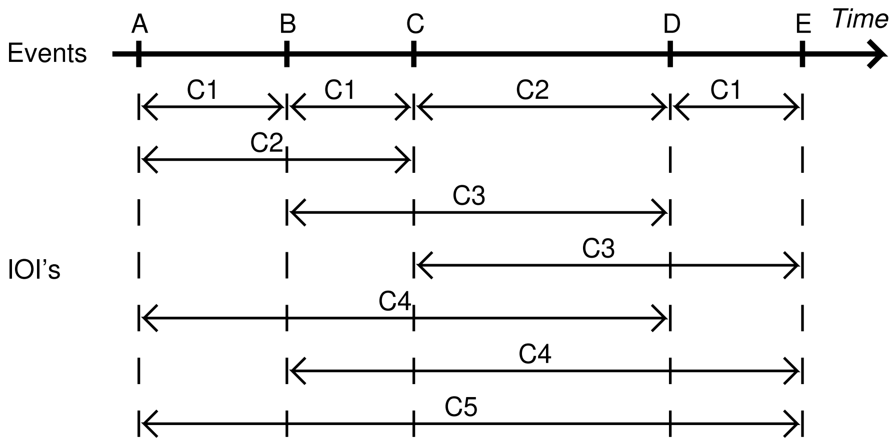

Figure 3. Clustering of inter-onset intervals: each interval between any pair of events is assigned to a cluster (C1, C2, C3, C4 or C5)

well with these errors, since the onsets which are hardest to detect are usually those which are least important rhythmically, whereas rhythmically important events tend to have an emphasis which makes them easy to detect.

The second method improves accuracy by separating the signal into frequency bands, and looking for onsets in each band. The signal was analysed in 256 frequency bands spaced linearly from 0–6000Hz, and local maxima in the log power (smoothed at a 10ms frame rate) were found. These maxima were then summed across frequency bands, and local peaks were found. The surfboard method was then used on each frequency band to find peaks in the slope, which were also summed across frequency. Finally the features were combined with a linear prediction function which had been optimised by a genetic algorithm.

For solo piano music (10 complete Mozart sonatas, over 3 hours of music), this onset detection method found over 91% of onsets, with an average error of 11ms. Since the frame rate used was 10ms, it is unlikely that a better resolution can be obtained with this method.

3.2. Tempo Induction

The tempo induction algorithm uses the calculated onset times to compute clusters of inter-onset intervals (IOIs). An IOI is defined to be the time interval between any pair of onsets, not necessarily successive. In most types of music, IOIs corresponding to the beat and simple integer multiples and fractions of the beat are most common. Due to fluctuations in timing and tempo, this correspondence is not precise, but by using a clustering algorithm, it is possible to find groups of similar IOIs which represent the various musical units (e.g. half notes, quarter notes).

This first stage of the tempo induction algorithm is represented in Figure 3, which shows the events along a time line (above), and the various IOIs (below), labelled with their corresponding cluster names (C1, C2, etc.). The next stage is to combine the information about the clusters, by recognising approximate integer relationships between clusters. For example, in Figure 3, cluster C2 is twice the duration of C1, and C4 is twice the duration of C2. This information, along with the number of IOIs in each cluster, is used to weight the clusters, and a ranked list of tempo hypotheses is produced and passed to the beat tracking system.

3.3. Beat Tracking

The most complex part of BeatRoot is the beat tracking subsystem, which uses a multiple agent architecture to find sequences of events which match the various tempo hypotheses, and rates each sequence to determine the most likely sequence of beat times. The music is processed sequentially from beginning to end, and at any particular point, the agents represent the various hypotheses about the rate and the timing of the beats up to that time, and make predictions of the next beats based on their current state.

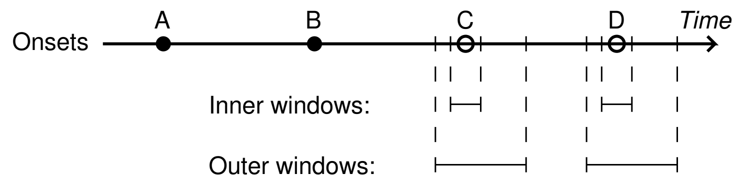

Figure 4. Tolerance windows of a beat tracking agent after onsets A and B have been determined to correspond to beats

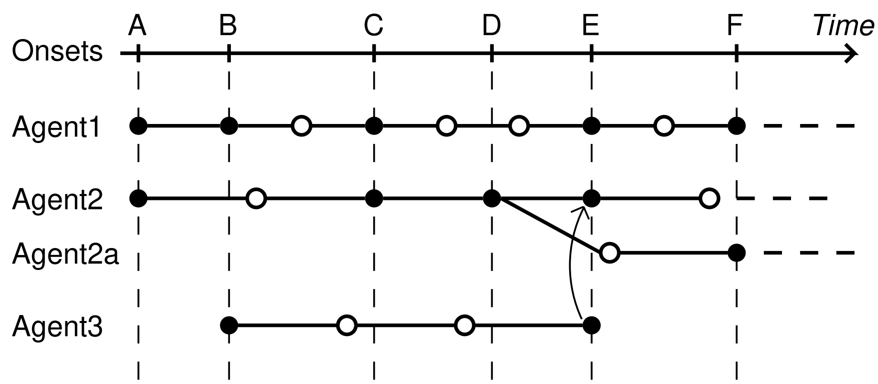

Figure 5. Beat tracking by multiple agents (see text for explanation)

Each agent is initialised with a tempo (rate) hypothesis from the tempo induction subsystem and an onset time, taken from the first few onsets, which defines the agent's first beat time (phase). The agent then predicts further beats spaced according to the given tempo and first beat, using tolerance windows to allow for deviations from perfectly metrical time (see Figure 4). Onsets which correspond with the inner window of predicted beat times are taken as actual beat times, and are stored by the agent and used to update its rate and phase. Onsets falling in the outer window are taken to be possible beat times, but the possibility that the onset is not on the beat is also considered.

Figure 5 illustrates the operation of beat tracking agents. A time line with 6 onsets (A to F) is shown, and below the time line are horizontal lines marked with solid and hollow circles, representing the behaviour of each agent. The solid circles represent predicted beat times which correspond to onsets, and the hollow circles represent predicted beat times which do not correspond to onsets. The circles of Agent1 are more closely spaced, representing a faster tempo than that of the other agents.

Agent1 is initialised with onset A as its first beat. It then predicts a beat according to its initial tempo hypothesis from the tempo induction stage, and onset B is within the inner window of this prediction, so it is taken to be on the beat. Agent1's next prediction lies between onsets, so a further prediction, spaced two beats from the last matching onset, is made. This matches onset C, so the agent marks C as a beat time and interpolates the missing beat between B and C. Then the agent continues, matching further predictions to onsets E and F, and interpolating missing beats as necessary.

Agent2 illustrates the case when an onset matches only the outer prediction window, in this case at onset E. Because there are two possibilities, a new agent (Agent2a) is created to cater for the possibility that E is not a beat, while Agent2 assumes that E corresponds to a beat.

A special case is shown by Agent2 and Agent3 at onset E, when it is found that two agents agree on the

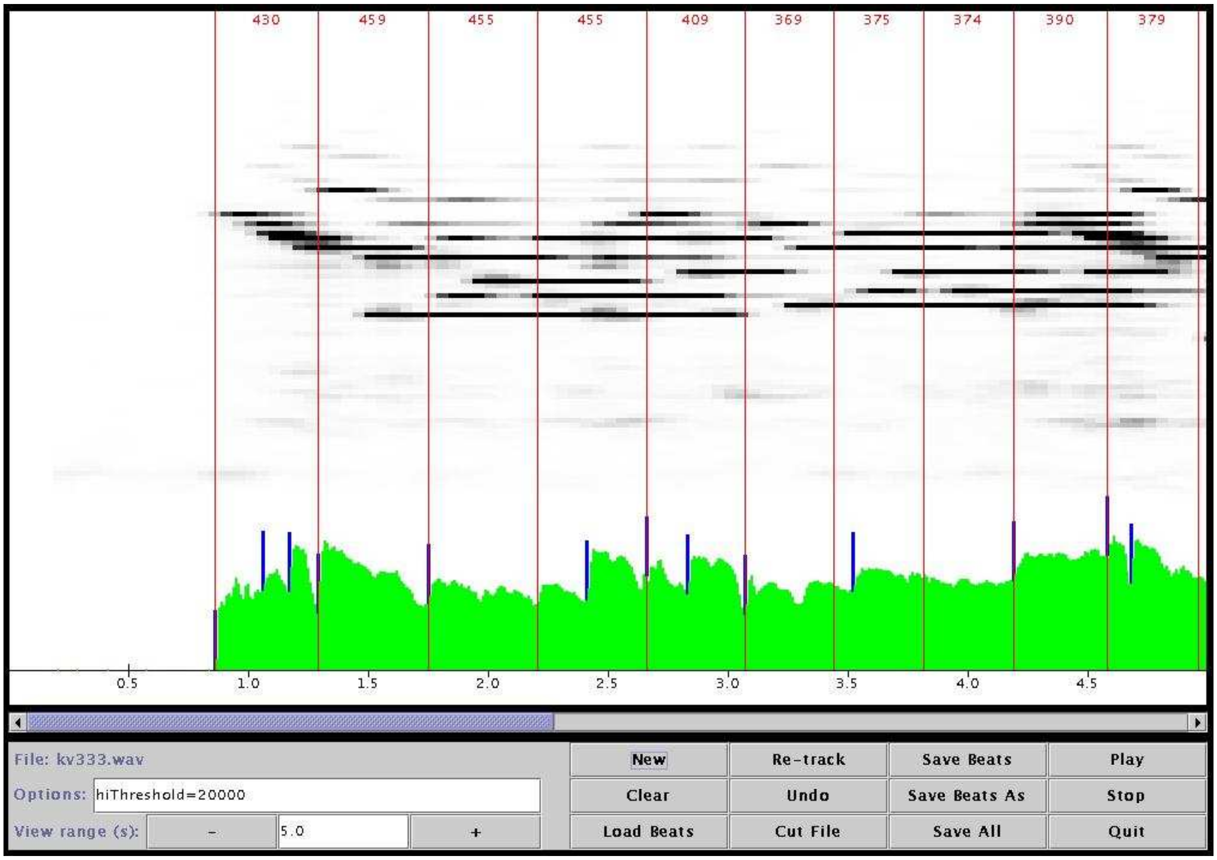

Figure 6. Screen shot of BeatRoot processing the first 5 seconds of a Mozart piano sonata, showing the inter-beat intervals in ms (top), calculated beat times (long vertical lines), spectrogram (centre), waveform (below) marked with detected onsets (short vertical lines) and the control panel (bottom)

time and rate of the beat. Rather than allowing the agents to duplicate each others' work for the remainder of the piece, one of the agents is terminated. The choice of agent to terminate is based on the evaluation function described in the following paragraph. In this case, Agent3 is terminated, as indicated by the arrow. A further special case (not illustrated) is that an agent can be terminated if it finds no events corresponding to its beat predictions (it has lost track of the beat).

Each agent is equipped with an evaluation function which rates how well the predicted and actual beat times correspond. The rating is based on how evenly the beat times are spaced, how many predicted beats correspond to actual events, and the salience of the matched events, which is calculated from the signal amplitude at the time of the onset. At the end of processing, the agent with the highest score outputs its sequence of beats as the solution to the beat tracking problem.

3.4. Implementation

The system described above has been implemented with a graphical user interface which allows playback of the music with the beat times marked by clicks, and provides a graphical display of the signal and the beats with editing functions for correction of errors or selection of alternate metrical levels. The audio data can be displayed as a waveform and/or a spectrogram, and the beats are shown as vertical lines on the display (Figure 6).

The main part of BeatRoot is written in C++ for the Linux operating system, comprising about 10000 lines of code. The user interface is about 1000 lines of Java code. Although it would be desirable to have a cross-platform implementation (e.g. pure Java), this was not possible at the time the project was commenced (1997), as the JavaSound API had not been implemented, and the audio analysis would have made the software too slow. Neither of these problems are significant now, so a pure Java version is in future plans. BeatRoot is open source software (under the GNU Public Licence), and is available from: `www.oefai.at/~simon/beatroot`

3.5. Testing and Applications

The lack of a standard corpus for testing beat tracking makes an objective evaluation of any such system difficult. BeatRoot has been tested on a range of different musical styles, including classical, jazz, and popular works with a variety of tempi and meters. The test data consisted of a set of 13 complete piano sonatas, a large collection of solo piano performances of two Beatles songs and a small set of popular, jazz and latin songs. In each case, the system found an average of over 90% of the beats,[1] and compared favourably to another state of the art tempo tracker.[21] Tempo induction is in most cases correct, with the most common error being the choice of a musically related metrical level such as double or half the subjectively chosen primary rate. The calculation of beat times is also robust; when the system loses synchronisation with the beat, it recovers quickly to resume correct beat tracking, despite the fact that the system has no high level knowledge of music to guide it. Interested readers are referred to the sound examples at: `www.oefai.at/~simon`

As a fundamental part of music cognition, beat tracking has practical uses in performance analysis, perceptual modelling, audio content analysis (such as for music transcription and music information retrieval), and the synchronisation of musical performance with computers or other devices. The implementation of BeatRoot includes a graphical interface which allows interactive monitoring of the beat tracking, selection of alternative metrical levels and correction of errors. Presently, BeatRoot is being used in a large scale study of interpretation in piano performance,[5] to extract symbolic data from audio CDs for automatic analysis.

4. THE PERFORMANCE WORM

Skilled musicians communicate high level information such as musical structure and emotion when they shape the music by the continuous modulation of aspects such as tempo and loudness. That is, artists go beyond what is prescribed in the score, and express their interpretation of the music and their individuality by varying certain musical parameters within acceptable limits. This is referred to as expressive music performance, and is an important part of western art music, particularly classical music. Expressive performance is a poorly understood phenomenon, and there are no formal models which explain or characterise the commonalities or differences in performance style. The Performance Worm[3] is a real time system for tracking and visualising the tempo and dynamics of a performance in an appealing graphical format which provides insight into the expressive patterns applied by skilled artists. This representation also forms the basis for automatic recognition of performers' style.[5]

The system takes input from the sound card (or from a file), and measures the dynamics and tempo, displaying them as a trajectory in a 2-dimensional performance space.[6] The measurement of dynamics is straightforward: it can be calculated directly as the RMS energy expressed in decibels, or, by applying a standard psychoacoustic calculation,[22] the perceived loudness can be computed and expressed in sones. The difficulty lies in creating a tempo tracking system which is robust to timing perturbations yet responsive to changes in tempo. This is performed by an algorithm which tracks multiple tempo hypotheses using an online clustering algorithm for time intervals. We describe this algorithm and then the implementation and applications of the Performance Worm.

4.1. Real Time Tempo Tracking

The tempo tracking algorithm is an adaptation of the tempo induction section of the BeatRoot system, modified to work in real time by using a fast online clustering algorithm for inter-onset intervals to find clusters of durations corresponding to metrical units. Onset detection is performed by the time domain "surfboard" algorithm from BeatRoot (see section 3.1), and inter-onset intervals are again used as the basis for calculating tempo hypotheses. The major difference is in the clustering algorithm, since it can only use the musical data up to the time of processing, and must immediately output a tempo estimate for that time. Another feature which is different is that the Performance Worm permits interactive selection of the preferred metrical level.

The tempo induction algorithm proceeds in 3 steps after onset detection: clustering, grouping of related clusters, and smoothing. The clustering algorithm finds groups of IOIs of similar duration in the most recent

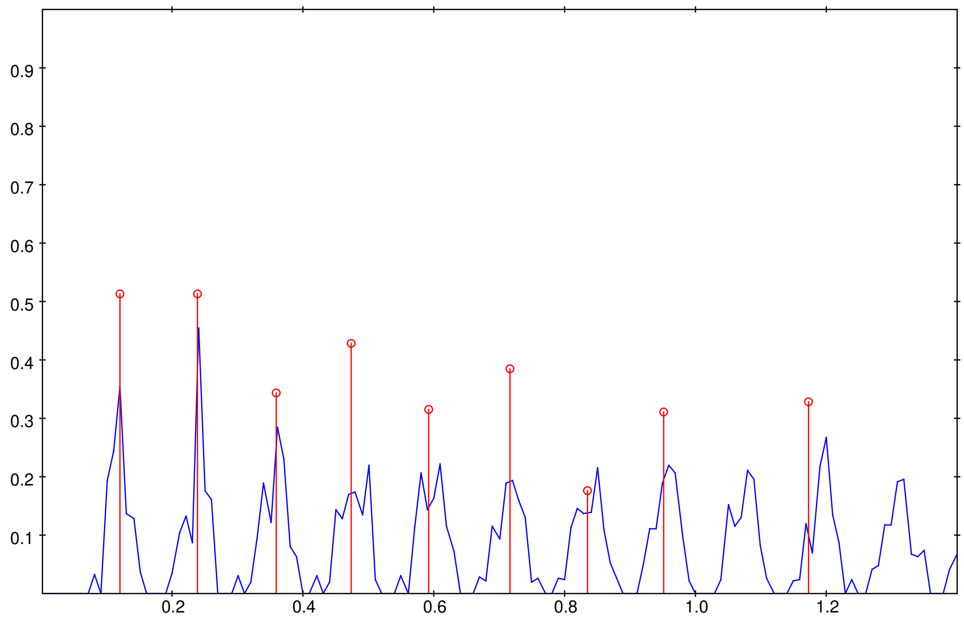

Figure 7. Screen shot of a weighted IOI histogram and the adjusted cluster centres (shown as vertical bars with height representing cluster weight) for part of the song Blu-bop by Béla Fleck and the Flecktones. The horizontal axis is time in seconds, and the vertical axis is weight.

8 seconds of music. Each IOI is weighted by the geometric mean of the amplitudes of the onsets bounding the interval. The weighted average IOI defines the tempo represented by the cluster, and the sum of the weights is calculated as the weight of the cluster.

In many styles of music, the time intervals are related by simple integer ratios, so it is expected that some of the IOI clusters also have this property. That is, the tempos of the different clusters are not independent, since they represent musical units such as half notes and quarter notes. To take advantage of this fact, each cluster is then grouped with all related clusters (those whose tempo is a simple integer multiple or divisor of the cluster's tempo), and its tempo is adjusted to bring the related groups closer to precise integer relationships.

The final step in tracking tempo is to perform smoothing, so that local timing irregularities do not unduly influence the output. The 10 best tempo hypotheses are stored, and they are updated by the new tempo estimates using a first order recursive smoothing filter. The output of the tempo tracking algorithm is a set of ranked tempo estimates, as shown (before smoothing) in Figure 7, which is a screen shot of a window which can be viewed in real time as the program is running.

4.2. Implementation and Applications

The Performance Worm is implemented as a Java application (about 4000 lines of code), and requires about a 400MHz processor on a Linux or Windows PC in order to run in real time. The graphical user interface provides buttons for scaling and translating the axes, selecting the metrical level, setting parameters, loading and saving files, and playing, pausing and stopping the animation. A screen shot of the main window of the Worm is shown in Figure 8.

Apart from the real time visualisation of performance data, the Worm can also load data from other programs, such as the more accurate beat tracking data produced by BeatRoot. This function enables the accurate comparison of different performers playing the same piece, in order to characterise the individual interpretative style of the performer. Current investigations include the use of AI pattern matching algorithms to attempt to learn to recognise performers by the typical trajectories that their playing produces.

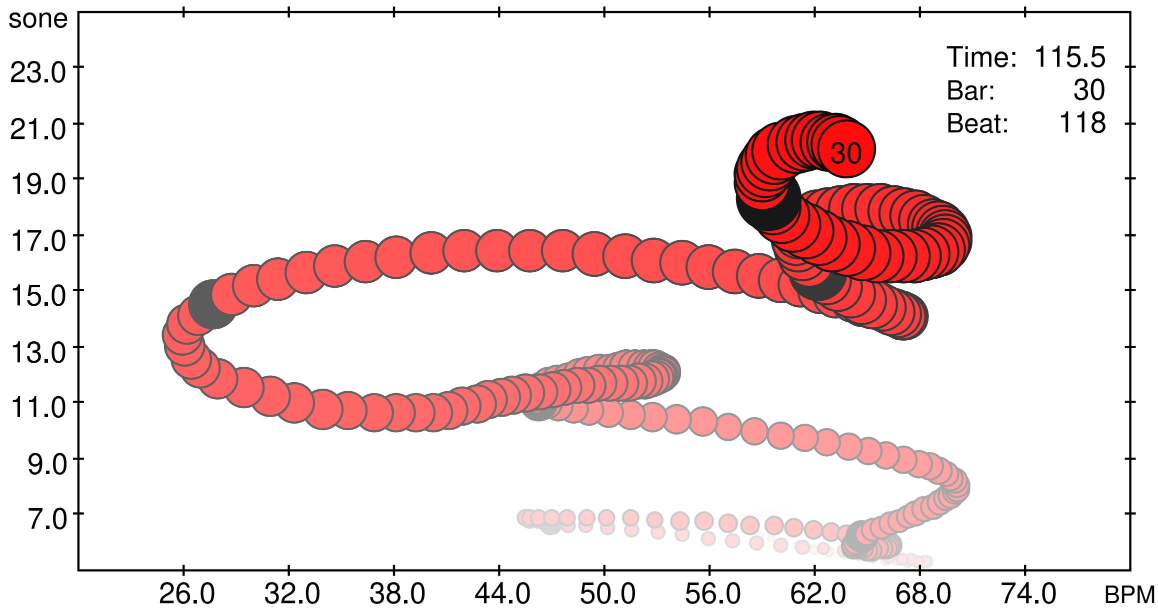

Figure 8. Screen shot of the Performance Worm showing a the trajectory to bar 30 of Rachmaninov's Prelude op.23 no.6 played by Vladimir Ashkenazy. The horizontal axis shows tempo in beats per minute, and the vertical axis shows loudness in sones.

5. CONCLUSION AND FUTURE WORK

In the tradeoff between generality and accuracy, this work leans towards the side of generality. It is clear that accuracy can be improved by restricting the class of signals to be analysed and tuning the systems to this restricted data type. So far, it is only assumed that the music has a somewhat regular beat, since if this is not the case, it is not even clear what a beat tracking system should do. Many other restrictions could be applied to the input data, for example, regarding instrumentation, pitch range or degree of polyphony, and the systems could be altered to take advantage of these restrictions and produce a more accurate analysis. However our work has shown that beat and tempo tracking can be performed bottom up from audio data for a number of different styles of music, without prior knowledge of the style.

Since the current systems are being used primarily for performance research in classical music, it is reasonable to consider the incorporation of high-level knowledge of the instruments or the musical scores into the systems, since these are known. By supplying a beat tracking or performance analysis system with the score of the music, most ambiguities are resolved, giving the possibility of a fully automatic and accurate analysis. Both dynamic programming and Bayesian approaches have proved successful in score following, for example for automatic accompaniment,[23] and it is likely that one of these approaches will be adequate for our purposes. Current work focusses on the score-based extraction of performance parameters from solo piano music. Finally, to put the work in perspective, we remind the reader that this work captures only a fraction of the factors which make up musical expression, and we do not expect to reach a complete understanding of the art of interpretation by the analysis of audio.

ACKNOWLEDGMENTS

This work was supported by the START programme (project Y99-INF) of the Austrian Federal Ministry of Education, Science and Culture (BMBWK). The Austrian Research Institute for Artificial Intelligence also acknowledges the basic financial support of the BMBWK.

REFERENCES

1. S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research* **30**(1), pp. 39–58, 2001.
2. S. Dixon, "An interactive beat tracking and visualisation system," in *Proceedings of the International Computer Music Conference*, pp. 215–218, International Computer Music Association, (San Francisco CA), 2001.
3. S. Dixon, W. Goebl, and G. Widmer, "Real time tracking and visualisation of musical expression," in *Music and Artificial Intelligence: Second International Conference, ICMAI2002*, pp. 58–68, Springer, (Edinburgh, Scotland), 2002.
4. W. Goebl and S. Dixon, "Analysis of tempo classes in performances of Mozart sonatas," in *Proceedings of VII International Symposium on Systematic and Comparative Musicology and III International Conference on Cognitive Musicology*, pp. 65–76, (University of Jyväskylä, Finland), 2001.
5. G. Widmer, "In search of the Horowitz factor: Interim report on a musical discovery project," in *Proceedings of the 5th International Conference on Discovery Science*, Springer, (Berlin), 2002.
6. 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.
7. C. Roads, *The Computer Music Tutorial*, MIT Press, Cambridge MA, 1996.
8. W. Schloss, *On the Automatic Transcription of Percussive Music: From Acoustic Signal to High Level Analysis*. PhD thesis, Stanford University, CCRMA, 1985.
9. P. Desain and H. Honing, "Quantization of musical time: A connectionist approach," *Computer Music Journal* **13**(3), pp. 56–66, 1989.
10. P. Allen and R. Dannenberg, "Tracking musical beats in real time," in *Proceedings of the International Computer Music Conference*, pp. 140–143, International Computer Music Association, (San Francisco CA), 1990.
11. D. Rosenthal, "Emulation of human rhythm perception," *Computer Music Journal* **16**(1), pp. 64–76, 1992.
12. E. Large and J. Kolen, "Resonance and the perception of musical meter," *Connection Science* **6**, pp. 177–208, 1994.
13. 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, International Computer Music Association, (San Francisco CA), 1995.
14. M. Goto and Y. Muraoka, "Real-time beat tracking for drumless audio signals," *Speech Communication* **27**(3–4), pp. 331–335, 1999.
15. E. Scheirer, "Tempo and beat analysis of acoustic musical signals," *Journal of the Acoustical Society of America* **103**(1), pp. 588–601, 1998.
16. A. 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, International Computer Music Association, (San Francisco CA), 2000.
17. D. Eck, *Meter Through Synchrony: Processing Rhythmical Patterns with Relaxation Oscillators*. PhD thesis, Indiana University, Department of Computer Science, 2000.
18. E. Scheirer, "Extracting expressive performance information from recorded music," Master's thesis, Massachusetts Institute of Technology, Media Laboratory, 1995.
19. S. Dixon, "Extraction of musical performance parameters from audio data," in *Proceedings of the First IEEE Pacific-Rim Conference on Multimedia*, pp. 42–45, 2000.
20. S. Dixon, "Learning to detect onsets of acoustic piano tones," in *Proceedings of the Workshop on Current Directions in Computer Music Research*, pp. 147–151, Audiovisual Institute, Pompeu Fabra University, (Barcelona, Spain), 2001.
21. S. Dixon, "An empirical comparison of tempo trackers," in *Proceedings of the 8th Brazilian Symposium on Computer Music*, 2001.
22. E. Zwicker and H. Fastl, *Psychoacoustics: Facts and Models*, Springer, Berlin, 1999. Second Edition.
23. C. Raphael, "Synthesizing musical accompaniments with Bayesian belief networks," *Journal of New Music Research* **30**(1), pp. 59–67, 2001.