Analysis of Musical Content in Digital Audio

Simon Dixon
Austrian Research Institute for Artificial Intelligence,
Schottengasse 3, Vienna 1010, Austria.
simon@oefai.at

1 Introduction

The history of audio analysis reveals an intensely difficult, laborious and error-prone task, where analysis tools have proved helpful, but final measurements have been based mostly on human judgement. Only since the 1980's did it begin to become feasible to process audio data automatically with standard desktop computers, and with this development, audio content analysis took an important place in fields such as computer music, audio compression and music information retrieval. That the field is reaching maturity is evident from the recent international standard for multimedia content description (MPEG7), one main part of which relates to audio (ISO, 2001).

Audio content analysis finds applications in automatic indexing, classification and content-based retrieval of audio data, such as in multimedia databases and libraries. It is also necessary for tasks such as the automatic transcription of music and for the study of expressive interpretation of music. A further application is the automatic synchronisation of devices such as lights, electronic musical instruments, recording equipment, computer animation and video with musical data. Such synchronisation might be necessary for multimedia or interactive performances, or studio post-production work.

In this chapter, we restrict ourselves to a brief review of audio analysis as it relates to music, followed by three case studies of recently developed systems which analyse specific aspects of music. The first system is BeatRoot (Dixon, 2001a,c), a beat tracking system that finds the temporal location of musical beats in an audio recording, analogous to the way that people tap their feet in time to music. The second system is JTranscriber, an interactive automatic transcription system based on (Dixon, 2000a,b), which recognises musical notes and converts them into MIDI format, displaying the audio data as a spectrogram with the MIDI data overlaid in piano roll notation, and allowing interactive monitoring and correction of the extracted MIDI data. The third system is the Performance Worm (Dixon et al., 2002), a real time system for visualisation of musical expression, which presents in real time a two dimensional animation of variations in tempo and loudness (Langner and Goebl, 2002).

2 Background

Sound analysis research has a long history, which is reviewed quite thoroughly by Roads (1996). 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. Although these methods are of great importance to the speech recognition community, there are few situations in which a musical signal is monophonic, so this type of pitch detection is less relevant in computer music research.

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 advanced time-frequency distributions are also used. Building upon these methods, the specific application of automatic music transcription has a long research history (Moorer, 1975; Piszczalski and Galler, 1977; Chafe et al., 1985; Mont-Reynaud, 1985; Schloss, 1985; Watson, 1985; Kashino et al., 1995; Martin, 1996; Marolt, 1997, 1998; Klapuri, 1998; Sterian, 1999; Klapuri et al., 2000; Dixon, 2000a,b). Certain features are common to many of these systems: producing a time-frequency representation of the signal, finding peaks in the frequency dimension, tracking these peaks over the time dimension to produce a set of partials, and combining the partials to produce a set of notes. The differences between systems are usually related to the assumptions made about the input signal (for example the number of simultaneous notes, types of instruments, fastest notes, or musical style), and the means of decision making (for example using heuristics, neural nets or probabilistic reasoning).

The problem of extracting rhythmic content from a musical performance, and in particular finding the rate and temporal location of musical beats, has also attracted considerable interest in recent times (Schloss, 1985; Longuet-Higgins, 1987; Desain and Honing, 1989; Desain, 1993; Allen and Dannenberg, 1990; Rosenthal, 1992; Large and Kolen, 1994; Goto and Muraoka, 1995, 1999; Scheirer, 1998; Cemgil et al., 2000; Eck, 2000; Dixon, 2001a). 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 advances, the field of performance research is yet to experience the benefit of computer 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 (Scheirer, 1995; Dixon, 2000a). 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. In the following sections, 3 systems are presented which take the approach of encoding as much as possible of this intelligence in the software and then presenting the results in a format that is easy to read and edit via a graphical user interface, so that the systems can be used in practical settings. This approach has proved to be very successful in performance research (Goebl and Dixon, 2001; Dixon et al., 2002; Widmer, 2002).

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 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 (for a full description, see Dixon, 2001a,c).

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*). At any time, there may exist multiple hypotheses regarding each of these processes; these are modelled by a multiple agent architecture in which agents representing each hypothesis compete and cooperate in order to find the best 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.

3.1 Onset Detection

Rhythmic information in music is carried primarily by the timing of the beginnings (onsets) of notes. 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

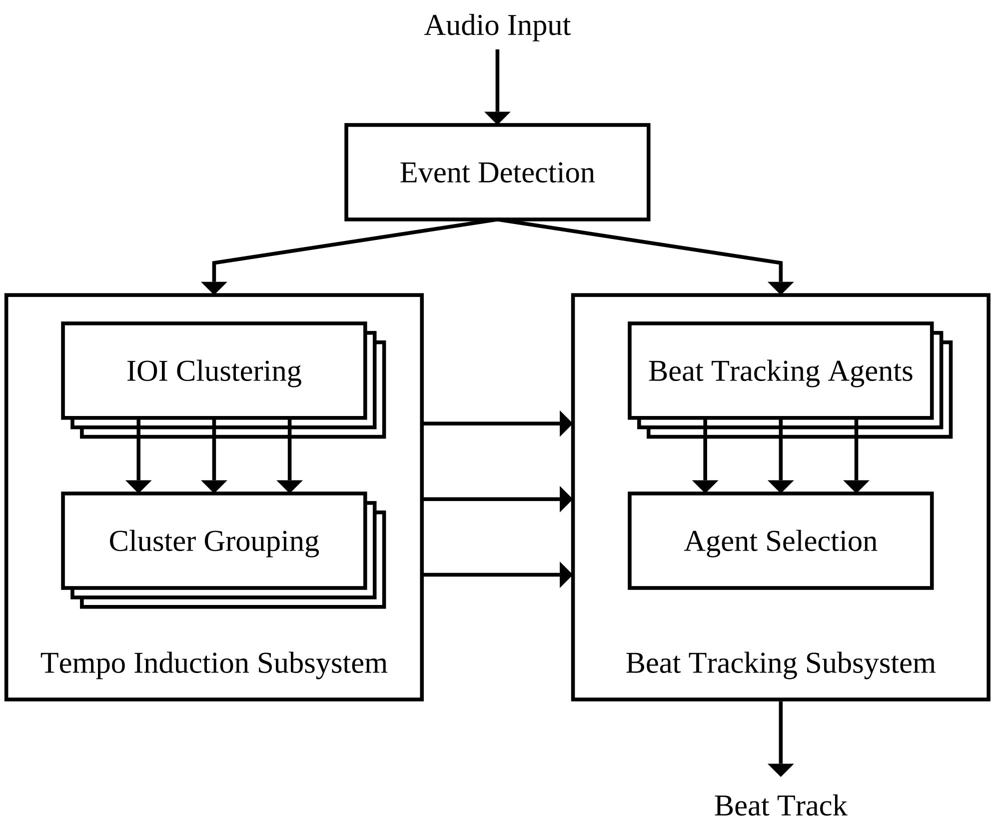

Figure 1: System architecture of BeatRoot

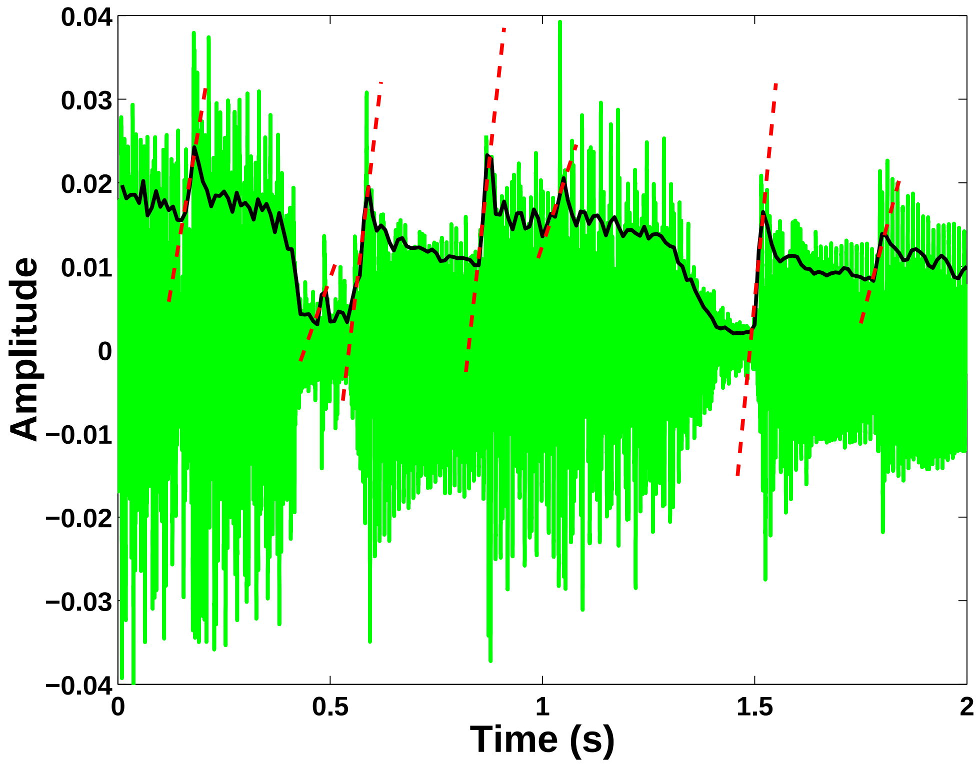

Figure 2: "Surfboard" method of onset detection, showing the audio signal in light grey, the smoothed signal (amplitude envelope) in black, and the detected onsets as dashed dark grey lines

algorithm. This algorithm is based on the "surfboard" method of Schloss (1985), 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. Occasional false onsets are detected, such as those caused by amplitude modulation in the signal. However, this is no great problem for the tempo induction and beat tracking algorithms, which are designed to be robust to noise. It turns out that 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.

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

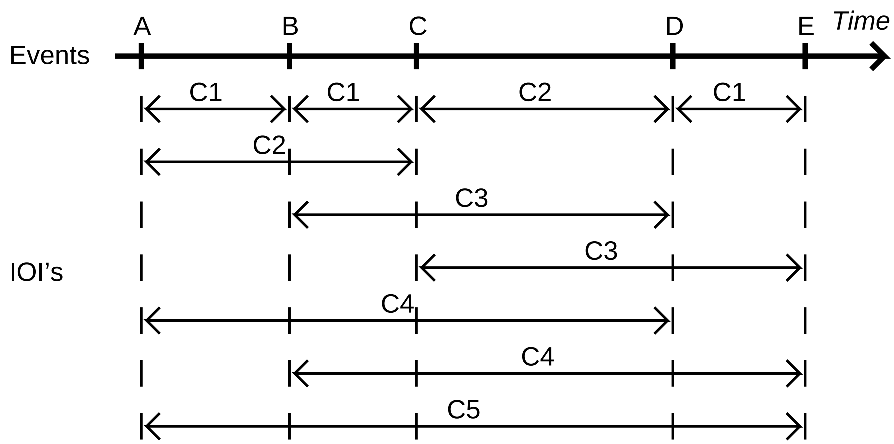

Figure 3: Clustering of inter-onset intervals: each interval between any pair of events is assigned to a cluster (C1, C2, C3, C4 or C5)

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 prediction of the next beats.

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. 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. Then any missing beats are interpolated, and the agent provides an evaluation function which rates how well the predicted and actual beat times correspond. The rating is based on how evenly the beat times

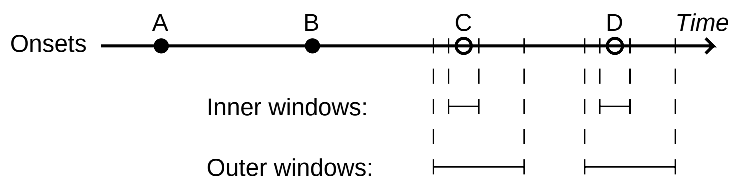

Figure 4: Tolerance windows of a beat tracking agent after events A and B have been determined to correspond to beats

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.

Various special situations can occur: an agent can fork into two agents if it detects that there are two possible beat sequences; two agents can merge if they agree on the rate and phase of the beat; and an agent can be terminated if it finds no events corresponding to its beat predictions (it has lost track of the beat). 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 5).

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

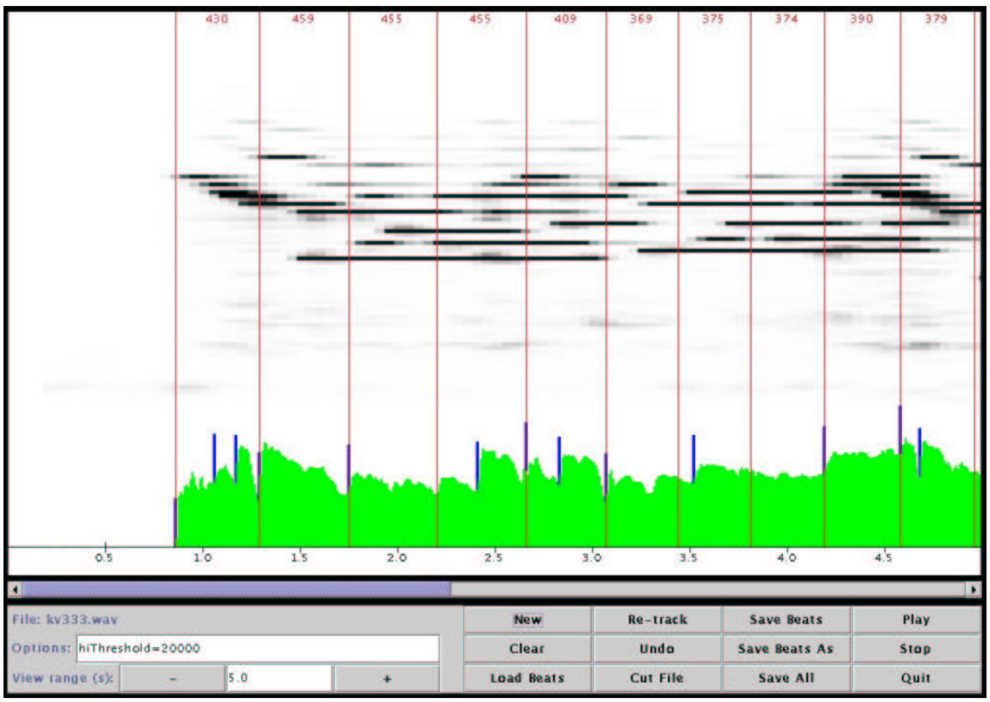

Figure 5: Screen shot of BeatRoot processing the first 5 seconds of a Mozart piano sonata, showing the inter-beat intervals in ms (top), calculated beat times (long vertical lines), spectrogram (centre), waveform (below) marked with detected onsets (short vertical lines) and the control panel (bottom)

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 creates a difficulty for making an objective evaluation of the system. The automatic beat tracking algorithm has been tested on several sets of data: a set of 13 complete piano sonatas, a large collection of solo piano performances of two Beatles songs and a small set of pop songs. In each case, the system found an average of over 90% of the beats (Dixon, 2001a), and compared favourably to another state of the art tempo tracker (Dixon, 2001b). Tempo induction results were almost always correct, so the errors were usually related to the phase of the beat, such as choosing as beats onsets half way between the correct beat times. 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 systems), and the synchronisation of musical performance with computers or other devices. Presently, BeatRoot is being used in a large scale study of interpretation in piano performance (Widmer, 2002), to extract symbolic data from audio CDs for automatic analysis.

4 JTranscriber

The goal of an automatic music transcription system is to create, from an audio recording, some form of symbolic notation (usually common music notation) representing the piece that was played. For classical music, this should be the same as the score from which the performer played the piece. There are several reasons why this goal can never be fully reached, not the least of which is that there is no one-to-one correspondence between scores and performances. That is, a score can be performed in different ways, and a single performance can be notated in various ways. Further, due to masking, not everything that occurs in a performance will be perceivable or measurable. Recent attempts at transcription report note detection rates around 90% for piano music (Marolt, 2001; Klapuri, 1998; Dixon, 2000a), which is sufficient to be somewhat useful to musicians.

A full transcription system is normally conceptualised in two stages: the signal processing stage, in which the pitch and timing of all notes is detected, producing a symbolic representation (often in MIDI format), and the notation stage, in which the symbolic data is interpreted in musical terms and presented as a score. This second stage involves tasks such as finding the key signature and time signature, following tempo changes, quantising the onset and offset times of the notes, choosing suitable enharmonic spellings for notes, assigning notes to voices in polyphonic passages, and finally laying out the musical symbols on the page. In this section, we focus only on the first stage of the problem, detecting the pitch and timing of all notes, or in more concrete terms converting audio data to MIDI.

4.1 System Architecture

The data is processed according to Figure 6: the audio data is averaged to a single channel and downsampled to increase processing speed. A short time Fourier transform (STFT) is used to create a time-frequency image of the signal, with the user selecting the type, size and spacing of the windows. Using a technique developed for the phase vocoder (Flanagan and Golden, 1966) and later generalised as *time-frequency reassignment* (Kodera et al., 1978), a more accurate estimate of the sinusoidal energy in each frequency bin can be calculated from the rate of change of phase in each bin. This is performed by computing a second Fourier transform with the same data windowed by a slightly different window function (the phase vocoder uses the same window shape shifted by 1 sample). When the nominal bin frequency corresponds to the frequency calculated as the rate of change of phase, this indicates a sinusoidal component (see Figure 7). This method helps to solve the problem that the main lobe of low frequency sinusoids is wider than a semitone in frequency, making it difficult to resolve the sinusoids accurately (see Figure 8).

The next step is to calculate the peaks in the magnitude spectrum, and to combine the frequency estimates to give a set of time-frequency atoms, which represent packets of energy localised in time and frequency. These are then combined with the atoms from neighbouring frames (time slices), to create a set of freqeuncy tracks, representing the partials of musical notes. Any atom which has no neighbours is deleted, under the assumption that it is an artifact or part of the transient at the beginning of a note. The final step is to combine the frequency tracks by finding the most likely set of fundamental frequencies that would give rise to the observed tracks. Each track is assigned to a note, and the pitch, onset time, duration and amplitude of the note are estimated from its constituent partials.

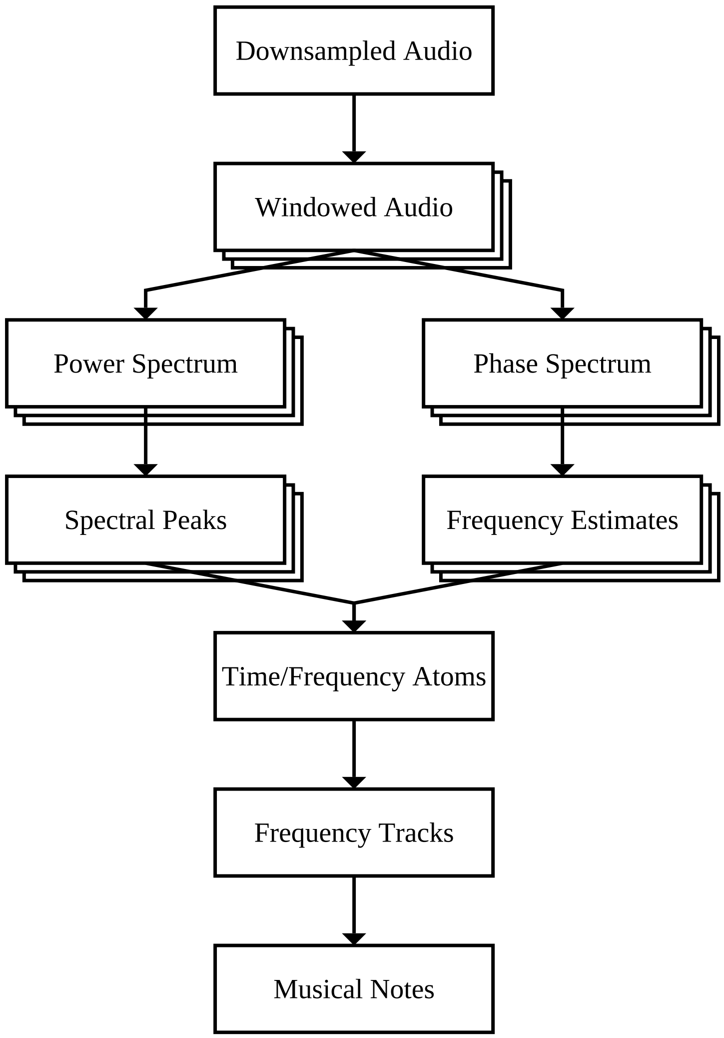

Figure 6: System architecture of JTranscriber

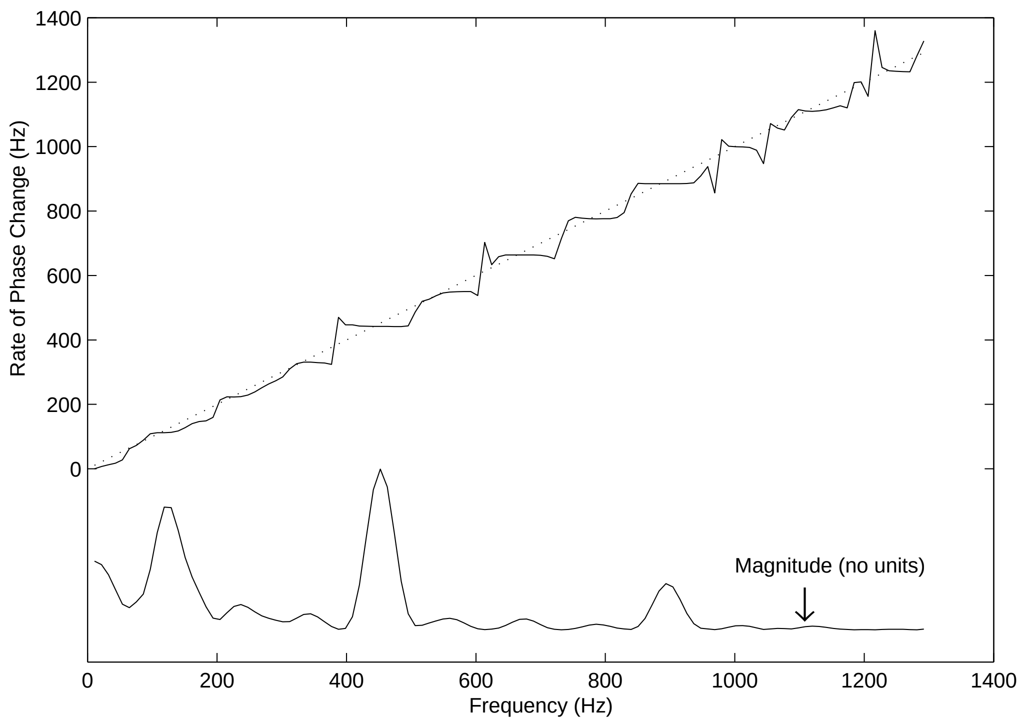

Figure 7: Rate of change of phase (vertical axis) against FFT frequency bin (horizontal axis), with the magnitude spectrum plotted below to show the correlation between magnitude peaks and areas of fixed phase change across frequency bins.

4.2 Implementation

An example of the output is displayed in Figure 8, showing a spectrogram representation of the signal using a logarithmic frequency scale, labelled with the corresponding musical note names, and the transcribed notes superimposed over the spectrogram in piano roll notation. (The piano roll notation is colour and partially transparent, whereas the spectrogram is black and white, which makes the data easily distinguishable on the screen. In the grey-scale diagram the coloured notes are difficult to see; here they are surrounded by a solid frame to help identify them.) An interactive editing system allows the user to correct any errors made by the automatic transcription system, and also to assign notes to different voices (different colours) and insert high level musical structure information. It is also possible to listen to the original and reconstructed signals (separately or simultaneously) for comparison.

An earlier version of the transcription system was written in C++, however the current version is being implemented entirely in Java, using the JavaSound API. Although the Java version is slower, this is not a major problem, since the system runs at better than real time speed (i.e. a 3 minute song takes less than 3 minutes to process on a 2GHz Linux PC). The advantages of using Java are shorter development time, as it is a better language, and portability, since the libraries used are platform independent.

4.3 Testing

The system was tested on a large database of solo piano music consisting of professional performances of 13 Mozart piano sonatas, or around 100000 notes (Dixon, 2000a). These pieces were performed on a computer monitored grand piano (Bösendorfer SE290), and were converted to MIDI format. At the time of the experiment, audio recordings of the original performances were not available, so a high quality synthesizer was used to create audio files

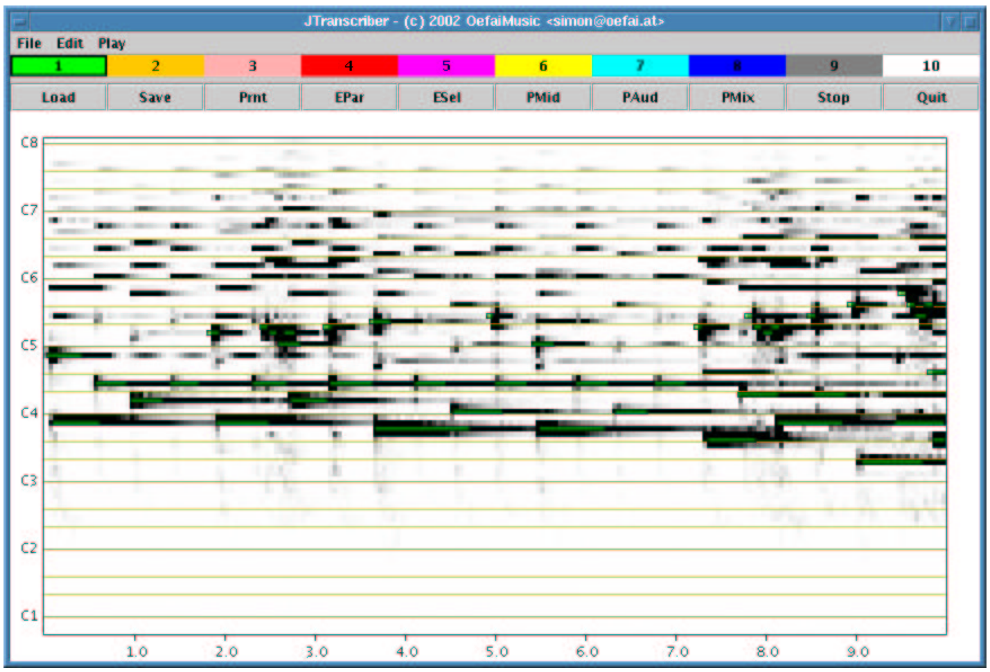

Figure 8: Transcription of the opening 10s of the 2nd movement of Mozart's Piano Sonata K332. The transcribed notes are superimposed over the spectrogram of the audio signal (see text). It is not possible to distinguish fundamental frequencies from harmonics of notes merely by viewing the spectrogram.

using various instrument sounds, and the transcription system's accuracy was measured automatically by comparing its output to the original MIDI files. A simple formula combining the number of missed notes, falsely recognised notes and played notes gave a percentage score on each instrument sound, which ranged from 69% to 82% for various different piano sounds. These figures represent that approximately 10-15% of the notes were missed, and a similar number of the reported notes were false. (Some authors use a different metric, which would award the system 85-90% correct.)

The most typical errors made by the system are thresholding errors (discarding played notes because they are below the threshold set by the user, or including spurious notes which are above the given threshold) and octave errors (or more generally, where a harmonic of one tone is taken to be the fundamental of another, and vice versa). No detailed error analysis has been performed yet, nor has any fine tuning of the system been performed to improve on these results.

5 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 (Dixon et al., 2002) 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 (Widmer, 2002).

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 (Langner and Goebl, 2002). 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 (Zwicker and Fastl, 1999), 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.

5.1 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 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 9, which is a screen shot of a window which can be viewed in real time as the program is running.

5.2 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, set-

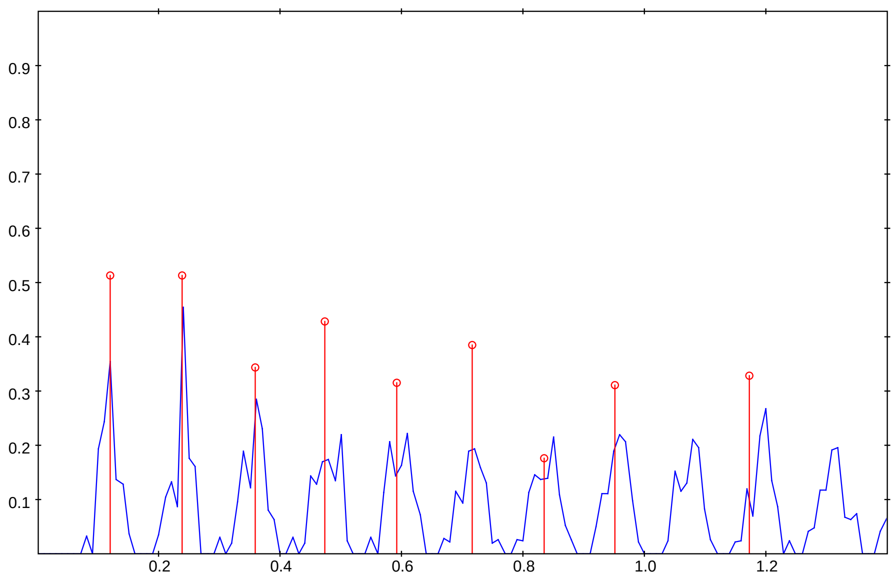

Figure 9: Screen shot of a weighted IOI histogram and the adjusted cluster centres (shown as vertical bars with height representing cluster weight) for part of the song Blu-bop by Béla Fleck and the Flecktones. The horizontal axis is time in seconds, and the vertical axis is weight.

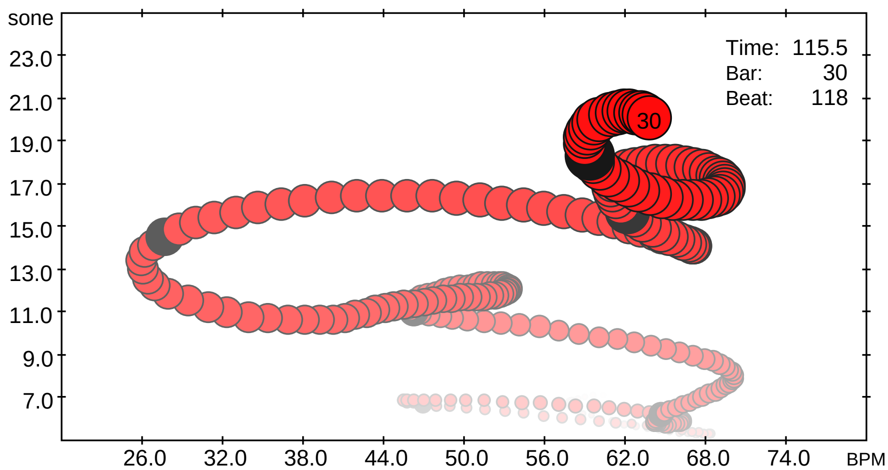

Figure 10: Screen shot of the Performance Worm showing a the trajectory to bar 30 of Rachmaninov's Prelude op.23 no.6 played by Vladimir Ashkenazy. The horizontal axis shows tempo in beats per minute, and the vertical axis shows loudness in sones.

ting 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 10.

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 interpretive 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.

6 Future Work

A truism of signal analysis is that there is a tradeoff between generality and accuracy. That is, the accuracy can be improved by restricting the class of signals to be analysed. It is both the strength and the weakness of the systems presented in this chapter that they are based on very general assumptions, for example, that music has a somewhat regular beat, and that notes are quasi-periodic (they have sinusoidal components at approximately integer multiples of some fundamental frequency). In fact if these assumptions do not hold, it is even difficult to say what a beat tracking or transcription 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. This has in fact been the approach of many earlier systems, which started from restrictive assumptions and left open the possibility of working towards a more general system. The problem with this approach is that it is rarely clear whether simple methods can be scaled up to solve more complex problems. On the other hand, fine tuning a general system by modules specialised for particular instruments or styles of music seems to hold a lot more promise.

Since the current systems are being used primarily for performance research, it is reasonable to consider the incorporation of high-level knowledge of the instruments or the musical scores into the systems. 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 (Raphael, 2001), and it is likely that one of these approaches will be adequate for our purposes.

A transcription system would also benefit from models of the specific instruments used or the number of simultaneous notes or possible harmonies. There are many situations in which this is not desirable; as an alternative we proposed in (Dixon, 1996) a dynamic modelling approach, where the system fine-tunes itself according to the instruments which are playing at any time.

7 Conclusion

Although it is a young field, analysis of musical content in digital audio is developing quickly, building on the standard techniques already developed in areas such as signal processing and artificial intelligence. A brief review of musical content extraction from audio was presented, illustrated by three case studies of state of the art systems. These systems are essentially based on a single design philosophy: rather than prematurely restricting the scope of the system in order to produce a fully automated solution, the systems make a fair attempt to process real world data, and then give the user a helpful interface for examining and modifying the results and steering the system. In this way, we are building research tools which are useful to a community that is wider than just other practitioners of musical content analysis.

Acknowledgements

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. Special thanks to the Bösendorfer Company, Vienna, for some of the performance data used in this work.

References

Allen, P. and Dannenberg, R. (1990). Tracking musical beats in real time. In *Proceedings of the International Computer Music Conference*, pages 140–143, San Francisco CA. International Computer Music Association.

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

Chafe, C., Jaffe, D., Kashima, K., Mont-Reynaud, B., and Smith, J. (1985). Techniques for note identification in polyphonic music. In *Proceedings of the International Computer Music Conference*, San Francisco CA. International Computer Music Association.

Desain, P. (1993). A connectionist and a traditional AI quantizer: Symbolic versus sub-symbolic models of rhythm perception. *Contemporary Music Review*, 9:239–254.

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

Dixon, S. (1996). A dynamic modelling approach to music recognition. In *Proceedings of the International Computer Music Conference*, pages 83–86, San Francisco CA. International Computer Music Association.

Dixon, S. (2000a). Extraction of musical performance parameters from audio data. In *Proceedings of the First IEEE Pacific-Rim Conference on Multimedia*, pages 42–45.

Dixon, S. (2000b). On the computer recognition of solo piano music. *Mikropolyphonie*, 6.

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

Dixon, S. (2001b). An empirical comparison of tempo trackers. In *Proceedings of the 8th Brazilian Symposium on Computer Music*.

Dixon, S. (2001c). An interactive beat tracking and visualisation system. In *Proceedings of the International Computer Music Conference*, pages 215–218, San Francisco CA. International Computer Music Association.

Dixon, S., Goebl, W., and Widmer, G. (2002). Real time tracking and visualisation of musical expression. In *Music and Artificial Intelligence: Second International Conference, ICMAI2002*, pages 58–68, Edinburgh, Scotland. Springer.

Eck, D. (2000). *Meter Through Synchrony: Processing Rhythmical Patterns with Relaxation Oscillators*. PhD thesis, Indiana University, Department of Computer Science.

Flanagan, J. and Golden, R. (1966). Phase vocoder. *Bell System Technical Journal*, 45:1493–1509.

Goebl, W. and Dixon, S. (2001). 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*, pages 65–76, University of Jyväskylä, Finland.

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

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

ISO (2001). *Information Technology — Multimedia Content Description Interface — Part 4: Audio*. International Standards Organisation. 15938-4:2001.

Kashino, K., Nakadai, K., Kinoshita, T., and Tanaka, H. (1995). Organization of hierarchical perceptual sounds: Music scene analysis with autonomous processing modules and a quantitative information integration mechanism. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Klapuri, A. (1998). Automatic transcription of music. Master's thesis, Tampere University of Technology, Department of Information Technology.

Klapuri, A., Virtanen, T., and Holm, J.-M. (2000). Robust multipitch estimation for the analysis and manipulation of polyphonic musical signals. In *Proceedings of the COST-G6 Conference on Digital Audio Effects*, Verona, Italy.

Kodera, K., Gendrin, R., and de Villedary, C. (1978). Analysis of time-varying signals with small BT values. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):64–76.

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

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

Longuet-Higgins, H. (1987). *Mental Processes*. MIT Press, Cambridge MA.

Marolt, M. (1997). A music transcription system based on multiple-agents architecture. In *Proceedings of Multimedia and Hypermedia Systems Conference MIPRO'97*, Opatija, Croatia.

Marolt, M. (1998). Feedforward neural networks for piano music transcription. In *Proceedings of the XIIth Colloquium on Musical Informatics*, pages 240–243.

Marolt, M. (2001). SONIC: Transcription of polyphonic piano music with neural networks. In *Proceedings of the Workshop on Current Directions in Computer Music Research*, pages 217–224, Barcelona, Spain. Audiovisual Institute, Pompeu Fabra University.

Martin, K. (1996). A blackboard system for automatic transcription of simple polyphonic music. Technical Report 385, Massachussets Institute of Technology Media Laboratory, Perceptual Computing Section.

Mont-Reynaud, B. (1985). Problem-solving strategies in a music transcription system. In *Proceedings of the International Joint Conference on Artificial Intelligence*. Morgan Kaufmann.

Moorer, J. (1975). *On the Segmentation and Analysis of Continuous Musical Sound by Digital Computer*. PhD thesis, Stanford University, CCRMA.

Piszczalski, M. and Galler, B. (1977). Automatic music transcription. *Computer Music Journal*, 1(4):24–31.

Raphael, C. (2001). Synthesizing musical accompaniments with Bayesian belief networks. *Journal of New Music Research*, 30(1):59–67.

Roads, C. (1996). *The Computer Music Tutorial*. MIT Press, Cambridge MA.

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

Scheirer, E. (1995). Extracting expressive performance information from recorded music. Master's thesis, Massachusetts Institute of Technology, Media Laboratory.

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

Schloss, W. (1985). *On the Automatic Transcription of Percussive Music: From Acoustic Signal to High Level Analysis*. PhD thesis, Stanford University, CCRMA.

Sterian, A. (1999). *Model-Based Segmentation of Time-Frequency Images for Musical Transcription*. PhD thesis, University of Michigan, Department of Electrical Engineering.

Watson, C. (1985). *The Computer Analysis of Polyphonic Music*. PhD thesis, University of Sydney, Basser Department of Computer Science.

Widmer, G. (2002). In search of the Horowitz factor: Interim report on a musical discovery project. In *Proceedings of the 5th International Conference on Discovery Science*, Berlin. Springer.

Zwicker, E. and Fastl, H. (1999). *Psychoacoustics: Facts and Models*. Springer, Berlin. Second Edition.