# Musicians and Machines: Bridging the Semantic Gap In Live Performance

Stark, Adam

# Musicians and Machines: Bridging the Semantic Gap In Live Performance

Thesis submitted in partial fulfilment
of the requirements of the University of London
for the Degree of Doctor of Philosophy

**Adam M. Stark**

Submitted: March 2011
Revised: August 2011

Centre for Digital Music,
Department of Electronic Engineering and Computer Science,
Queen Mary, University of London

I certify that this thesis, and the research to which it refers, are the product of my own work, and that any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline. I acknowledge the helpful guidance and support of my supervisor, Prof. Mark Plumbley.

# Abstract

This thesis explores the automatic extraction of musical information from live performances – with the intention of using that information to create novel, responsive and adaptive performance tools for musicians.

We focus specifically on two forms of musical analysis – harmonic analysis and beat tracking. We present two harmonic analysis algorithms – specifically we present a novel chroma vector analysis technique which we later use as the input for a chord recognition algorithm. We also present a real-time beat tracker, based upon an extension of state of the art non-causal models, that is computationally efficient and capable of strong performance compared to other models. Furthermore, through a modular study of several beat tracking algorithms we attempt to establish methods to improve beat tracking and apply these lessons to our model.

Building upon this work, we show that these analyses can be combined to create a beat-synchronous musical representation, with harmonic information segmented at the level of the beat. We present a number of ways of calculating these representations and discuss their relative merits.

We proceed by introducing a technique, which we call *Performance Following*, for recognising repeated patterns in live musical performances. Through examining the real-time beat-synchronous musical representation, this technique makes predictions of future harmonic content in musical performances with no prior knowledge in the form of a score.

Finally, we present a number of potential applications for live performances that incorporate the real-time musical analysis techniques outlined previously. The applications presented include audio effects informed by beat tracking, a technique for synchronising video to a live performance, the use of harmonic information to control visual displays and an automatic accompaniment system based upon our performance following technique.

*For Mum, Dad, Simon and Melina*

# Acknowledgements

Writing a PhD thesis has been one of the most inspiring experiences of my life. That I have been able to dedicate these years to the exploration of my childhood love – music – is something for which I will always feel immense gratitude.

First and foremost I thank my supervisor Prof. Mark Plumbley, whose encouragement, tireless reading of papers and chapters, constructive feedback and optimism towards my work have not only made the research interesting and productive, but pushed back the boundaries of what I thought I could achieve. I would also like to thank Dr. Simon Dixon and Dr. Nick Bryan Kinns for their helpful guidance. I acknowledge the contribution of Dr. Matthew Davies to sections 3.1 and 3.4 and Chapter 4, which were based upon collaborative work.

The Centre for Digital Music has been a wonderful place to write a PhD. In particular I wish to thank Matthew Davies and Andrew Robertson for their advice, knowledge, feedback and good humour. I also received considerable help from Rebecca Stewart, Tim Murray Browne, Dan Stowell, Steve Welburn, Samer Abdallah, Kurt Jacobson, Matthias Mauch and Katy Noland. Thanks also to everyone who I have met along the way – Alice Clifford, Amélie Anglade, Andrew Nesbit, Anssi Klapuri, Aris Gretsistas, Asterios Zacharakis, Chris Cannam, Chris Harte, Dan Tidhar, Daniele Barchiesi, Dawn Black, Di Mainstone, Dimitrios Giannoulis, Enrique Perez Gonzalez, Gyorgy Fazekas, Hassan Khalil, Ioannis Patras, Ivan Damnjanovic, Janine Lajudie, Jean-Baptiste Thiebaut, Josh Reiss, Ken O'Hanlon, Maria Jafari, Martin Morrell, Mathieu Barthet, Mehdi Zadeh, Michael Terrell, Peter Foster, Robert Macrae, Robin Fencott, Sarah Nicolls, Sefki Kolozali, Wen Xue and Yves Raimond.

I would like to thank Anssi Klapuri, Matthew Davies and Stephen Hainsworth for making available their annotated databases for my beat

tracking evaluation. I would also like to thank the following people for contributing compositions to the evaluation of my performance follower in Chapter 5: Andrew Robertson, Claire Robbin, Jamie Dale, Peter Greliak, Philip Bazany, Simon Stark, Steve Willey and Tom Barton. I would also like to thank the EPSRC for funding this work.

I would not have been able to write a single word without the constant support of so many good people. I wish to thank Ed Nesbit, for reading through my drafts; Jennifer Sandella, Katrina Krämer and Olivia Crowther for creating warm and exciting places for me to live; Hannah Morgan, Siew Cottis, Joe Bartlett, Terry Murphy, Earl Edgar, James Whyard, James Pritchard, Sam Coyle, Lauren Moysey and Mark Parsons for the music, and standing around in rehearsal rooms while I edited badly written code; Anna Parker, Alex Wilson, Jen Southern, Lucy McDonald, Josie O'Donoghue, Elena Trivelli, Laura Thrussell, Rebecca Wood, Amy Jones, Tom Barton, Ingrid Geser, Paula Smolarksa, Tali Febland, Jessica Griggs, Jamie Dale, Charlotte Twining, Camilla Sutherland, Xavier Marco Del Pont, Tessa Whitehouse, Sophie Robinson, Georgie Lord, Peter Greliak, Jake Tully, James Ash, Lydia White, John Herbert and Mike Clease for their endless calm and insight; Natalie Ballinger, for keeping me afloat; Karie Meltzer, for putting me up in Austin and her inspiring ambition; Claire Robbin, for whom the music never stops; Philippa Palmer, whose wit and energy I can only dream of; Lizzy Williamson, for her gentle wisdom and words words words; Omar Rahwangi, for plans and music on the long road home; Steve Willey, for coffee and revolution; Matthew Rose, for defining loyalty; Philip Bazany, for twenty long years; and Rosheen Brennan, who gave me the strength to begin this journey, and made me realise what it really means to be 'counting the beats'.

Everything I have done in the last few years has been underpinned by the loving support of my family. I wish to thank my grandparents, Norman and Veronica Stark, who are a constant inspiration to me. My brother, Simon, and my sister, Melina, I am so grateful for. Finally, for teaching me to love music and books, and for giving me every opportunity I could have hoped for, I wish to thank my parents – my father, Tony, and my mother, Chrissy.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

This thesis investigates the automatic analysis of musical audio in live performances and the use of such analysis to inform decision-making machines controlling multimedia applications. The implication is of machines aware of the nature of their immediate musical surroundings, able to contextualise them within the temporal development of the performance and capable of producing a coherent response. Our focus will be upon the degree to which machines can be made to recognise changes in the salient characteristics of musical performances, such as rhythmic and harmonic developments.

In the same way in which a human musician may 'listen' and 'respond', our approach will be based upon an analysis of musical surroundings, the result of which will inform one of a number of live performance applications. We first develop a number of real-time signal processing algorithms to extract musical information from live performances. We then combine these algorithms to extract information about temporal developments in musical performances. Finally, we use these analysis techniques to inform a number of applications for use in live performances.

We focus upon live performance because it presents a number of challenges different to the case of processing entire audio files. Firstly, live performances are fast moving and variable and there is no time to provide meta-data about changes in the performance. Furthermore, there is a lack of future information in a live performance. The result is that any techniques to track a live performance must either be instantaneous (or have acceptable latency) in their extraction of information or be predictive of future information in some way. Finally, the time constraints provided by

the real-time nature of a performance require any analysis to be computationally efficient. This differs from the case of processing an audio file where future information is available and we have no time constraints for processing the sound.

## 1.1 Music and Machines

By giving some musical 'understanding' to machines in a live performance, we can have them control musically sensitive applications – responding automatically to changes in tempo, or harmonic developments in a piece of music.

However, this implies a degree of autonomy on the part of the machine – it will make decisions in response to musical changes with no human interference. While others have spoken of the discomfort that some may have with an autonomous machine partaking in a live performance [Collins, 2007], this is certainly not a new idea.

There is already a rich tradition of autonomous decision making in music. While experiments with rule-based composition date back for hundreds of years [Nierhaus, 2009, Chapter 2], the first experiments with automated composition using computers began in the 1950s [Ames, 1987]. These saw digital computers used to generate pieces of music according to sets of rules or using statistical processes. There continues to be much interest in the algorithmic composition of music [Miranda and Biles, 2007; Nierhaus, 2009]. Furthermore, the development of more powerful technology has allowed the development of 'interactive' music systems [Rowe, 1993; Collins, 2006b] that collaborate with human performers in real-time, generating or transforming musical material in response to changes in music.

There also exist cross-disciplinary examples of autonomous decision-making in creative fields [Boden, 2004]. *The Painting Fool* [Colton, 2008, 2009] is a computer program capable of painting, interpreting through keywords the artistic style of the picture including the level of abstraction, the colours to be used, the physical materials to be simulated and the painting style itself. The system won the British Computer Society Machine

Intelligence Competition in 2007. Systems for the automatic generation of poetry also exist [Gervás, 2002].

Technology in general has had a huge influence on the development of musical styles and practices during the 20th century [Braun, 2002]. Developments in analog and digital recording technology have changed the relationship between people and music – allowing music to be taken home to be played repeatedly, leading to new musical practices and concepts such as the DJ [Prendergast, 2000]. The studio itself has become a musical instrument, with the creative use of musical production now commonly used in popular (as well as classical) music, including The Beatles (working with George Martin), Pink Floyd, Nine Inch Nails and much modern electronica and hip hop [Moorefield, 2005]. Live performances in music now often involve a range of analog and digital technology, such as audio effects units, synthesisers and software sequencers [Roads, 1996, Chapters 14 and 15].

In this thesis we investigate how the automatic analysis of musical audio can facilitate new applications for live musical performances. We now consider the practical use of such technology and its potential to improve the relationship between musicians and machines in live performances.

### 1.1.1 Empowering Musicians through Machine Autonomy

In recent years, it has become more common to see laptops and other computing technology on stage with musicians [Collins et al., 2003; Stuart, 2003; Zadel and Scavone, 2006]. We focus in this thesis on the use of a machine within a live performance involving one or more human musicians.

There are some clear benefits to using computer technology in a performance. For example, it gives us the ability to use powerful software to manipulate sound samples and create textures, in real-time, that would be difficult to create with instrumentation alone. We can also play back recorded material with ease during performances.

These advances have created new problems, however. The first major problem is the lack of performativity (the visual stimulus of the physical movements of performers) associated with using a personal computer

during a live performance [Stuart, 2003; Zadel and Scavone, 2006]. The second is that there is a 'semantic gap' between computers and humans – computers do not have a musical 'understanding' of the sounds made by their human counterparts. To perform with computer technology on stage, human performers must adhere to the often inflexible mechanisms of the computer, which can restrict human performance and expression through the need to manually enter performance information or to synchronise with computer timing. For example, in order to synchronise rhythmically with a computer, musicians must play to a computer defined sequence of pulses, or 'click track'. This can lead to an expressionless and mechanical performance.

By giving some musical 'understanding' to machines in performance, we can actually allow greater freedom for musicians by placing the musicians in control through musical expression, having the machine follow the musical developments of human performers. An example of this is work to allow a recorded accompaniment to synchronise with a human drummer playing an expressive performance [Robertson and Plumbley, 2007]. This is achieved by tracking the tempo variations in the performance and adapting the rate of playback of the accompaniment accordingly.

Furthermore, by allowing the machine to automatically follow the human performance, human maintenance of the computer during performance is not necessary, improving the performer's relationship with the audience. As Young [2009] has argued, "[w]ould any musical *effect* necessarily be lost if those apparently ineffectual laptop-focussed actions are substituted by the machinations of a 'creative' algorithm?".

By accepting that computers are already a commonplace part of musical performance, we can see that allowing them some musical understanding and autonomy can actually give greater control to musicians, and allow them to perform as if playing with human musicians while still making use of the sonic potential of computer technology in performance.

## 1.2 Objectives and Motivation

The motivating force behind this thesis is the belief that if we can automatically decipher human musical concepts – such as beat, harmony, rhythm and melody – from real-time musical signals, then we can create an infrastructure upon which novel, responsive and seemingly 'intelligent' music and multimedia applications can be created.

The broad aim of this work is to develop a number of real-time analysis algorithms to create a live performance musical 'information framework' upon which applications can be developed.

#### Scope of the Work

Given the diversity of music across time and between different cultures, producing technology able to respond equally to all forms of music would be an immense challenge. As a result it makes sense to restrict the scope of our research.

The music under consideration in this thesis is Western music and within that, broadly music from the Rock and Pop genres [Frith et al., 2001; Everett, 2009]. We choose Western music because the use of tonality, rhythm and other characteristics often conform to well understood models of music theory – and it is also the music of choice of a great deal of other research in the field of musical audio analysis. We choose Rock and Pop music because while musicians from all disciplines are interested in computer music, we believe that it is musicians performing in those genres that would be most interested in applications for connecting musicians and technology.

Finally, we will focus specifically on two aspects of music in this thesis – beat and harmonic information. Within the scope of the research, we have the following specific objectives:

1. To develop musical audio analysis algorithms to provide information about the beat and harmonic information in real-time. These algorithms must be both robust, in terms of being reliable, and also computationally efficient.

2. We will explore the combination of these algorithms to produce harmonic information segmented by the beat in real-time.

3. We will develop a technique for following temporal developments in musical performances, based upon the beat and harmonic analysis techniques.

4. We will present a number of live performance applications based upon the information from the musical audio analysis algorithms.

## 1.3 Outline of Thesis

The work in this thesis is focused upon developing techniques for analysing music in real-time before using the results of that analysis to inform live performance applications. The work is split into the following chapters:

#### Chapter Two

In Chapter 2 we present a review of the relevant theoretical background and related research necessary to contextualise the work in subsequent chapters. We begin with a discussion of musical interaction and live performance systems, examining paradigms for their classification and outlining a number of live performance systems presented by others.

As many of these live performance systems are built using some form of musical audio analysis, we proceed to discuss the two forms of musical audio analysis relevant to the work in this thesis. First, we discuss beat tracking – examining why it is a difficult problem and discussing the details of a number of different prior approaches. We then examine harmonic analysis – in particular chroma analysis techniques and chord recognition – outlining the various techniques presented previously.

Finally, we present an overview of the existing technology infrastructure that is useful in implementing real-time performance applications.

**Chapter Three**

In Chapter 3 we present several techniques for real-time analysis of musical audio signals. We present a real-time beat tracking model based upon an extension of a non-causal dynamic programming based approach. We show that our approach is comparable with other state of the art causal models and close in performance to state of the art non-causal models. Finally, we show that our approach is computationally efficient compared to other state of the art models.

We then present a real-time chroma analysis technique for live performances based upon mapping the energy in spectral peaks to pitch classes of the chromagram. This chroma analysis technique is then used as part of a real-time chord recognition technique, by classifying the real-time chromagram output as one of a number of chord labels through a template matching approach that minimises the energy in the chromagram when expected note positions are masked out.

Finally, we present a number of methods for combining our beat tracking and harmonic analysis techniques to produce a beat-synchronous harmonic representation in real-time. We then show how these methods can be used to compute a beat-synchronous spectrogram, chromagram and chord sequence in real-time.

**Chapter Four**

In Chapter 4 we investigate methods for improving beat tracking through a modular evaluation of five state of the art beat tracking algorithms. Our motivation is to apply any lessons learned to our real-time beat tracker in order to improve its performance.

We split five beat tracking models into their input feature and tracking model. The resulting 20 combinations of four input features and five tracking models are then evaluated on two databases and the results discussed. We then proceed to compare the best single feature to an 'oracle' feature, where the best feature for each file is used. The considerable performance gains imply that the use of a variety of signal dependent features, rather than a single feature for all files, would increase performance. We find

evidence that the same may be true of the parameters of tracking models.

Finally, we review the findings of the chapter before proceeding to apply some of the lessons learned to our real-time beat tracker.

**Chapter Five**

In Chapter 5 we present a technique for predicting harmonic content in musical performances with no prior knowledge in the form of a score. This 'performance following' technique makes use of the real-time beat synchronous harmonic analysis technique presented in Chapter 3.

We show that by comparing, in real-time, the most recent few seconds to the most recent few minutes – contextualising recent developments in the performance within the larger piece – we are able to predict the future harmonic content of performances in the presence of the repetition of musical patterns.

We present an objective evaluation of our technique compared to other models, including an N-gram model, Factor Oracle and random predictor, and show that our model is able to predict a large proportion of repeated content in 32 pieces for acoustic guitar – and more than any other model.

**Chapter Six**

Having presented – in Chapters 3, 4 and 5 – a number of techniques for the analysis of musical signals, in Chapter 6 we present a number of real-time applications that make use of this analysis.

We present a number of beat-synchronous audio effects – where audio effect parameters are informed by the tempo and beat of a real-time beat tracker. Also making use of our real-time beat tracker, we present a technique for automatically synchronising a video to a live performance, automatic beat-synchronous lighting patterns and the automatic synchronisation of audio samples to the beat and tempo of a live performance.

Making use of our real-time chroma analysis technique we present an application whereby harmonic information is mapped to the colour tone of a video. Finally, we demonstrate that our performance following technique can inform a simple music accompaniment system by using information

in the chromagram to deduce a root note through our chord recognition technique, which is then output as a bassline accompaniment.

**Chapter Seven**

In Chapter 7 we conclude with an overview of the work in this thesis, a review of the key contributions, a summary of third party use of this research and the directions for future work.

## 1.4 Publications

The work in this thesis has been published in a number of journal and conference papers:

**Journal Paper**

- Adam M. Stark and Mark D. Plumbley, "Performance Following: Real-Time Prediction of Musical Sequences Without A Score," *IEEE Transactions on Audio, Speech and Language Processing*, Accepted for Publication, 2011, [Stark and Plumbley, 2011]

**Conference Papers**

- Adam M. Stark and Mark D. Plumbley, "Performance Following: Tracking a Performance Without a Score", In *Proceedings of The International Conference on Acoustics Speech and Signal Processing (ICASSP 2010)*, Dallas, TX, USA, pp 2482-2485, March 2010. [Stark and Plumbley, 2010]

- Adam M. Stark, Matthew E. P. Davies and Mark D. Plumbley, "Real-Time Beat-Synchronous Analysis of Musical Audio", In *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09)*, Como, Italy, 1-4 September, 2009. [Stark et al., 2009]

- Adam M. Stark and Mark D. Plumbley, "Real-Time Chord Recognition For Live Performance", In *Proceedings of the 2009 International*

*Computer Music Conference (ICMC 2009)*, Montreal, Canada, 16-21 August 2009. [Stark and Plumbley, 2009]

- Adam M. Stark, Matthew E. P. Davies and Mark D. Plumbley, "Rhythmic Analysis For Real-Time Audio Effects", In *Proceedings of the 2008 International Computer Music Conference (ICMC 2008)*, Belfast, UK, 2008. [Stark et al., 2008]

- Adam M. Stark, Matthew E. P. Davies and Mark D. Plumbley, "Real-Time Beat-Synchronous Audio Effects", In *Proceedings of New Interfaces for Musical Expression (NIME 2007)*, New York, NY, USA, pp 344-345, June 6-10, 2007. [Stark et al., 2007b]

- Adam M. Stark, Matthew E. P. Davies and Mark D. Plumbley, "Audio Effects for Real-Time Performance Using Beat Tracking", In *Proceedings of the the 122nd Audio Engineering Society (AES) Convention*, Vienna, Austria, paper 7156, May 58 2007 [Stark et al., 2007a]

## 1.5 Thesis Contributions

The main novel contributions of this thesis are:

- Chapter 3: An efficient and robust real-time beat tracking algorithm developed by extending a previously non-causal technique based upon dynamic programming.

- Chapter 3: An efficient and robust real-time chord recognition technique based upon: 1) a chromagram calculation technique based upon mapping only the energy in spectral peaks to pitch classes rather than all spectral bins within a certain range; and 2) a chord classification technique based upon masking out expected note positions in the chromagram and minimising the residual energy.

- Chapter 4: A modular study of five state of the art beat tracking models, splitting them into their input feature and tracking model

and examining the effect of all possible combinations of input features and tracking models.

- Chapter 5: A technique – which we call 'Performance Following' – for automatically predicting harmonic content in polyphonic music performances based upon the identification of repeated musical patterns.
- Chapter 6: A number of live performance applications including beat-synchronous audio effects, an algorithm for the synchronisation of video to a live performance and the use of harmonic analysis to control video colour tone in real-time.

We turn now to a review of previous work in systems for live musical performances and musical audio analysis techniques.

# Chapter 2

# Background

In this thesis we aim to develop ways in which technology can be easily integrated into live musical performances. The intention is to develop live musical performance systems that are aware of their musical surroundings through information provided by real-time musical audio analysis.

In this chapter we will discuss related research in this area. In particular we will discuss a number of different techniques for musical audio analysis – specifically beat tracking and harmonic analysis. We will also discuss existing technology infrastructures that can be used to implement live performance tools. First of all, however, we turn to a study of related research into live musical performance systems.

## 2.1 Musical Interaction

In much previous literature on the use of computer systems in live performances, a useful paradigm has been the concept of *interaction* and the description of these computer systems as *interactive music systems*.

Rowe [1993] defines interactive music systems as "those whose behavior changes in response to musical input". He goes on to say that this responsiveness allows such systems to participate in live performances of both notated and improvised music. This implies that two broad processes take place in an interactive system. The first is some analysis of the input, which may be in a symbolic form, such as MIDI, or a raw audio signal. The second process is the creation of some response based upon the result of that analysis.

However, others have considered other criteria key to an interactive music system. Much has been written on the subject – a review is presented in Jordà [2005] – and we will not attempt to cover it all here. Rather, we focus on two commonly visited notions which we refer to as *complexity* and *mutualism*, discussing each in turn.

### Complexity

The first criteria, which we call *complexity*, is that the response should seem sufficiently autonomous and unpredictable – perhaps using the generation of pseudo-random numbers or a knowledge-based system in the process of the response.

Chadabe [1984] introduced the concept of "interactive composing", describing it as a two stage process. He says that it involves "(1) creating an interactive composing system and (2) simultaneously composing and performing by interacting with that system as it functions". Furthermore, he argues that the interactive composing system is a form of "intelligent instrument" that "responds to a performer in a complex, not entirely predictable way". Dobrian [2004] has drawn a distinction between systems that are *interactive* and systems that are simply *reactive*. He states that systems that respond "instantly to the sound or gestures of a live performer based on a programmed algorithm" are simply "reacting to its input in a pre-determined way". He continues, adding that "[t]he computer can only be purported to be acting autonomously if it is programmed to make some decisions of its own that are not fully predicted by the algorithm".

Spiegel [1992] has also indicated that the "predictability and repeatability (versus randomness from the user's viewpoint) of musical result from a specific repeatable human interaction" is an important dimension of interactive music systems.

### Mutualism

The second criteria we shall refer to as *mutualism* – the requirement that both the human performer and computer system are able to 'hear' one another and influence one another's performance in some way.

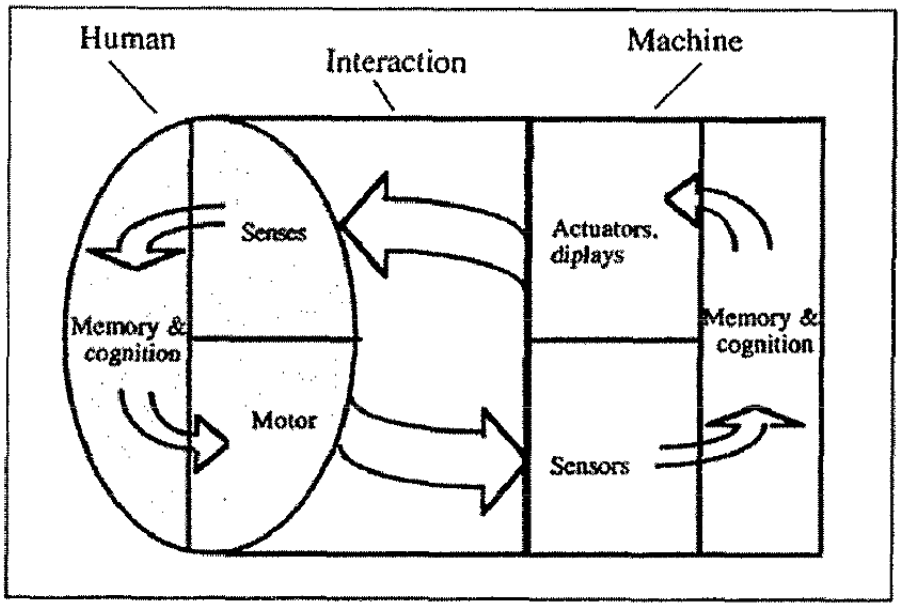



Figure 2.1: Human-Machine Interaction (from Bongers [1999])

Chadabe [1984] has described an interactive relationship between a performer and a computer system in a live performance as one where the "computer responds to the performer and the performer reacts to the computer, and the music takes its form through that mutually influential, interactive relationship". Bongers [1999] has also identified mutual influence as important, arguing that "[m]any interactive systems in new media arts are in fact reactive systems. Ideally, interaction between a human and a system should be mutually influential". His depiction of such a "human-machine interaction loop" has been reproduced in Figure 2.1. As can be seen, both parties – the human and the machine – are able to 'sense' one another's outputs in some way and respond, based upon some thought or processing.

Jordà [2005] has suggested that the quantity and nature of the interaction between two subjects could be represented in a two-dimensional space. The $x$-axis represents the degree of influence of the human performer on the computer, while the $y$-axis represents the influence of the computer on the human performer. It is indicated that the area of interest for interactive systems are those that have both high $x$ and $y$ values –

situations with mutual influence.

Others have also stressed the importance of mutual influence, including Dobrian [2004] who has claimed that "the prefix inter- in the word interactivity implies mutual influence between agents".

There is clearly no standard definition of an 'interactive music system', and we shall not attempt to offer one here. The work in this thesis is aimed predominantly at exploring the musical audio analysis techniques which may inform an interactive music system. We explore some example applications in Chapter 6, some of which may be considered to be interactive to an extent. We also find equally interesting those systems which may be considered "reactive" by others such Bongers [1999] or Dobrian [2004]. For example, the automatic control via audio feature extraction of video – which may be viewed by an audience rather than the human performer and thus provides no mutual influence – is still an exciting application.

We now proceed with a paradigm for classifying these interactive music systems (introduced by Rowe [1993]) before considering the state of the art in the field.

### 2.1.1 Classification of Interactive Music Systems

Rowe [1993] considers three dimensions as the basis for differentiating different forms of interactive music system, which we now discuss in turn.

#### Score-driven vs. Performance-driven

Score-driven systems attempt to match a musical input to a stored score or stored music fragments. Performance-driven systems, on the other hand, do not anticipate that the musical input will correspond to a specified score or stored representation. Rather the input is unexpected and is processed it as it arrives.

#### Musical Response: Transformative, Generative or Sequenced?

Transformative systems take some existing musical material and apply transformations to it to produce variants. Generative systems generate

their own output by using sets of rules to produce musical output using only fragmentary stored information, such as scales or phrases. Sequenced systems use pre-recorded musical accompaniments whose parameters, such as tempo, dynamics or rhythm, are varied in real-time in response to the input signal.

#### Instrument Paradigm vs. Player Paradigm

A system belonging to the instrument paradigm will seem like an extension of a musical instrument while a system belonging to the player paradigm will produce an output that seems like a separate musical entity, or an accompaniment.

It should be noted that Rowe's is not the only model for classifying and describing interactive music systems and that others have been presented, such as that of Spiegel [1992].

### 2.1.2 Prior Work

The field of interactive music systems is wide, varied and the subject of numerous books and PhD theses. Here we present a selection of work to illustrate the nature of recent and past research in the area. For more comprehensive reviews of work on interactive music systems, the reader may wish to consider Rowe [1993], Roads [1996, Chapter 15], Rowe [2001], Jordà [2005] and Collins [2007]. We now present a number of interactive music systems and some areas of interest within the field.

#### Cypher

Rowe [1992] has presented *Cypher*, an interactive music system that is composed of a 'listener' and a 'player'. Taking as input a MIDI stream, the listener is a multi-agent system that extracts musical features from the input. Incoming MIDI events are classified within a feature space where the dimensions include loudness, duration, harmony, register and density. The user can then configure the player to react to these features in certain ways, using a graphical interface. The player can either: transform

material arriving from some MIDI source; sequence stored stored material for playback; or generate new material. An internal critic examines the output and ensures a consistency of style based upon pre-programmed aesthetic preferences.

#### Voyager

Jazz trombonist George Lewis [2000] developed the interactive system *Voyager* between 1986 and 1988. The system is conceived in the 'player' paradigm (in the sense of the taxonomy of Rowe [1993]), although Lewis describes the system as having 64 asynchronous players, generating single voice MIDI outputs. These players are then organised into ensembles, with associated behaviours. A function is called internally every 5 to 7 seconds to organise the players into new ensembles with new behaviours, specifying characteristics such as timbre, choice of melody algorithm, pitch set and tempo. This means that the system is capable of generating music in the absence of human input.

To interact with human performers, the system takes input from either a MIDI keyboard or uses pitch-to-MIDI conversion to track Lewis's trombone. The input MIDI stream is analysed for information such as pitch averages, velocity and note spacing and this is used to decide in greater detail how each ensemble responds to the input – defining output characteristics such as tempo, note spacing, octave range and volume.

Lewis stresses the degree of equality between human and computer sound generations in Voyager, describing it as "a nonhierarchical, improvisational, subject-subject model of discourse, rather than a stimulus/response setup".

#### William Hsu and John Butcher

Hsu [2005] has presented an interactive music system that improvises with British saxophonist John Butcher, emphasising the analysis of musical timbre. The system extracts a number of timbral features from the input sound – Butcher's saxophone – including noisiness, the presence of inharmonic partials, the sharpness of attacks and roughness. A number

of non-timbral features are also extracted including the pitch range and density of note onsets.

The musical response of the system is generated by a number of agent improvisers. Each agent controls a module that may be a transformation of the sound (such as an audio effect) or a synthesis module. Each module receives information about the characteristics of the sound and then uses a set of internal rules to determine a musical response (or whether to play at all). Each module can perform independently or in conjunction with some of the other modules.

#### Eighth Nerve

Ciufo [2003] has presented the interactive system *Eighth Nerve* for prepared electric guitar and computer. The electric guitar is fitted with a number of sensors mounted on the guitar, including toggle switches and a pressure sensor on the neck. Audio analysis is also performed on the signal from the guitar, including onset detection, amplitude monitoring and the extraction of timbral features. The features from the audio analysis and outputs from the mounted sensors are then combined to control signal processing manipulations of the guitar signal and of recorded audio buffers in real-time.

#### The Continuator

The *Continuator* by Pachet [2002] is a system capable of imitating musical style. A Markov model is used to represent sequences and sub-sequences of notes either from a performance or from data stored in a MIDI file. The Markov model is then used to generate new material in real-time by traversing a prefix tree according to transition probabilities. The Markov model is augmented to account for elements such as rhythm and beat and is adapted to be able to handle imprecision.

#### Score Following

Much research in the area of musical interaction focuses upon *score following* [Orio et al., 2003]. These systems attempt to match a human

performance to a stored version of that performance in the form of a musical score. Based upon this information an accompaniment is provided to the human musician that increases or decreases its tempo in order to stay synchronised with the human performer.

The first work on score following was presented, separately, by Vercoe [1984] and Dannenberg [1984]. Vercoe [1984] presented a 'synthetic performer', which attempts to match notes played by a flautist to a score, with the intention of automatically playing an accompaniment. The approach was comprised of three main stages. The first was a 'listen' stage, where optical sensors were installed on the keys of the flute and used in combination with audio pitch analysis to determine the pitch and attack times of notes. The second stage, 'perform', involves using the information about the pitch and attack times of notes to identify the position of the flautist on the score. The flautist score position is then compared to the score position of the 'synthetic performer' and a 'catch-up' action is then determined to align the two. The third and final stage is 'learn', where it is suggested that the performance of the system could be improved if it could learn the musical response of performers to certain scores.

Dannenberg [1984] considers the problem of matching a solo performance to a score. He considers the solo performance as a real-time stream of musical events, and the score as a stored, ordered list of expected events. The problem of matching the performer to the score is then approached as one of finding the best match between the two streams. To achieve this a dynamic programming sequence alignment technique is used to find the position in the score of each arriving musical event from the performance. The system is implemented using pitch detection (assisted by hardware for reasons of limited processor speed) of a trumpet sound. Baird et al. [1993] presented a similar system based upon MIDI input that uses a short time window to align performer and score and allows for missed notes, extra notes and notes held for longer than expected.

Raphael [1999] presented a probabilistic approach to the problem of partitioning an acoustic signal of a monophonic instrument into notes and rests, given a score. Using hidden Markov models [Rabiner, 1989], he models the notes in the score as a sequence of hidden states that will

produce a series of observations – in practice, audio frames. The score following problem is then approached as one of recovering the sequence of hidden states (i.e. notes in the score) that produced the observations (audio frames). This technique was later used to track a live performer in the automatic accompaniment system 'Music Plus One' Raphael [2001], where a probabilistic model is used to match an accompaniment to the performance of a soloist.

Cont [2010] has presented a probabilistic model for score following based upon two coupled agents for audio and tempo. With states representing events in the symbolic score, the agents work together to decode the best sequence of states given the observation of the input audio. The result is a real-time determination of the both the score position and current tempo.

#### Generative Systems

The possibility of computers generating music independently of humans has been of interest to many in recent years. Generative music involves the automatic generation of music, for example by algorithms, sets of rules or statistical techniques. We shall present some illustrative examples here, but for deeper studies see Miranda and Biles [2007] or Nierhaus [2009].

A notable example is the work done in Experiments in Musical Intelligence by Cope [1996] where new pieces in a certain style are composed automatically from analysis of two or more pieces in that style. This is achieved by identifying patterns common in pieces from that style and using an augmented transition network to re-arrange these patterns to create a new piece. However, these pieces are generated offline and are not used in a live performance system.

Conklin [2003] argues that increased attention should be given to the use of analytical statistical models for the purpose of generating music. He claims that the goal of analytic statistical models is to assign high probabilities to new pieces in a given style. This process can then simply be used to evaluate a number of candidate musical generations, with those of low probability eliminated. Therefore the problem of music generation

can be reduced to sampling from a statistical model.

Blackwell and Young [2004] introduced the 'swarm granulator' for generating parameter settings for granular synthesis. Inspired by decentralised organisational processes in nature such as those exhibited by swarms of bees or shoals of fish, the authors present a particle swarm model where each particle moves around a 6-dimensional space. The particles have no centralised control, and each particle is only aware of a number of other nearby particles, rather than the whole swarm. 'Attractors', locations in the space, are created by some external interpreting system – in some cases a human musician. This is achieved by extracting parameters from an incoming audio stream. Organisation is then achieved through behavioural rules for each particle including moving towards 'attractors' at locations in the space, matching the speed of other particles and avoiding collisions. The position of the particles in the space determine the sound characteristics of individual sound 'grains' of a granular synthesis technique – specifically pitch, amplitude, duration, the time between successive grains and grain attack and decay times.

Biles [1994] has presented *GenJam* – a system for improvising jazz solos based upon a genetic algorithm. Performing over a standard rhythm section, the system outputs solo material which is then rated as either 'good' or 'bad' by its human counterparts. Based upon this feedback, certain phrases attain higher fitness values than others. The system can then use these measures to choose phrases from which to produce 'offspring', gradually learning to play better solos (from the perspective of its human counterparts).

The majority of the live performance systems we have described here process some form of musical input – either symbolic or an audio signal. We are concerned in this thesis with the development of live performance technology that is capable of processing real audio signals. Raw audio signals only contain information about the physical characteristics of sound. More 'musical' notions such as pitch, beat and rhythm are not explicitly

expressed but rather interpreted by human listeners. As a result, raw audio signals are insufficient in themselves for providing information to live performance tools – rather the 'semantic' information must be extracted from them through some form of analysis.

We now turn to recent work in two areas of musical signal analysis – beat tracking and harmonic analysis.

## 2.2 Beat Tracking

Beat tracking is the identification of a regular pulse in a piece of music, similar to the informal task of tapping ones foot 'in time' to a piece. To a human, even a non-musician, this is often a trivial task that can be achieved with little conscious effort. However, as we shall describe here, the automation of this process in computer systems has proved difficult to solve comprehensively.

The musical input to a beat tracker will be characterised by musical events. These events will often coincide with beat locations, however many will also occur at non-beat locations and many beat locations will be marked by no event. The regular beat is therefore *implicit*, rather than explicit. As Goto [2001] has noted, "[a]lthough in the brains of performers music is temporally organized according to its hierarchical beat structure, this structure is not explicitly expressed in music". Continuing he claims that "[t]he principal reason that beat tracking is intrinsically difficult is that it is the problem of inferring an original beat structure that is not expressed explicitly".

In this section we outline a number of beat tracking approaches, from early models to the current state of the art.

### Early Approaches

Early beat tracking models attempted to find a regular beat from symbolic onset times. Allen and Dannenberg [1990] introduced a model that selects between multiple possible interpretations of each note onset – each interpretation corresponding to a state with a different metric position,

beat period and beat phase. An evaluation function evaluates each state according to musical heuristics. The most credible state is chosen and used to make beat predictions.

Rosenthal [1992] has presented a technique that, given symbolic onset times from MIDI data, finds sets of regularly spaced onsets corresponding to 'rhythmic levels'. These levels are studied to see if they can be classed into families before various criteria based upon human musical listening traits are used to rank the various hypotheses.

Large [1995] developed a method whereby the period and phase of an oscillator are modified in order to synchronise it with a sequence of incoming pulses, corresponding to onset times.

The detection of beats directly from audio signals can be achieved by using onset detection functions (reviewed in Bello et al. [2005]) – a mid-level representation large at the onset of musical notes and small elsewhere. We proceed by reviewing several approaches to audio beat tracking.

#### Scheirer

Scheirer [1998] suggests that human perception of rhythmic information is based upon an analysis of multiple frequency bands. Based upon this he splits the input signal into 6 octave spaced frequency bands. For each band, the amplitude envelope is calculated and the derivative taken. The resulting signal is then half wave rectified.

Each envelope derivative is then passed into a comb filterbank where each comb filter has a different delay. Comb filters with delays matching the periodicities of each derivative signal will respond more strongly than others. The outputs of the filterbank for each envelope derivative are then summed. The result is then examined for the comb filter resonator that produced the maximum energy. Once identified, the delay of this filter is used as the tempo estimate of the signal.

Then, the approach makes use of delay vectors used to implement the comb filters. These delay vectors contain delayed samples and their output can be interpreted as the predicted output of a given resonator. The delay vectors from the frequency channels for the resonators corresponding to

the tempo are summed. Finally, the beat phase is extracted through reference to the peak of the resulting signal.

#### Dixon

Dixon [2001] has introduced *BeatRoot*, a system that can operate upon audio or symbolic MIDI data. The system is composed of a tempo induction stage and a beat tracking stage. In the tempo induction stage onset times are extracted from the input. In the case of MIDI, the onset times of individual notes are known but as the signal may be polyphonic, onsets that are close together are classed as the same event. A salience value is also given to each onset based upon the MIDI data on the duration, density, dynamics and pitch of each note.

In the case of audio, an amplitude envelope is calculated and a peak picking algorithm used to identify local maxima in the slope of the envelope. Peaks within 50ms of greater peaks are rejected to retain only the most salient onsets – with a salience value calculated for each onset as a function of the amplitude envelope value.

Once onsets have been calculated, the times between onsets or 'inter-onset intervals' (IOI), are extracted. The inter-onset intervals are then assigned to clusters of IOIs of a similar length. The resulting clusters are ranked according to the number of elements they contain – with the top clusters representing a set of hypotheses of the basic tempo of the music.

The beat tracking stage compares several beat tracking 'agents', initialised with different tempo hypotheses and choses the one that best fits the onset data from the input. This decision is based upon the regularity of beat times, how often beat times match rhythmic events and the salience of matched rhythmic events.

The model was later updated [Dixon, 2007], with improvements including replacing the original amplitude envelope input feature with a spectral flux onset detection function [Dixon, 2006b] to allow better representation of non-percussive onsets and a re-implementation in the Java programming language.

### Goto

Goto [2001] presents a system that is able to recognise beats at multiple metrical levels. The system, assuming a steady tempo and a time signature of 4/4, examines onset times, chord changes and drum patterns in the input signal.

Onsets are extracted through a peak-picking process based upon the rate of power increase across 7 sub-bands. The auto-correlation of the onset sequence is used to calculate the tempo before cross-correlation is used to predict quarter-note level beat times.

Two kinds of musical knowledge are then used to extract beat information at higher metrical levels. Firstly, the quarter-note beat times are used to derive a measure of chord change probability by comparing the dominant frequencies present in each inter-beat interval. This chord change probability is then combined with several musical assumptions – for example, that chord changes are more likely at the measure level than at lower metrical levels – to estimate half-note and measure level beat times.

Secondly, bass and snare drum onsets are extracted through analysis of the onset sequence, examining the width of noise spread across the frequency axis. These drum onsets are then compared to a number of pre-stored drum patterns, with matching patterns used to infer half-note beat times through the musical assumption that the start of a pattern indicates a half-note beat time.

To deal with ambiguity, beat tracking is performed by multiple agents with competing hypotheses of the beat structure. These are selected between based upon which agent is most reliable according to an evaluation function.

### Dannenberg

Dannenberg [2005] has presented a beat tracker that makes use of information about musical structure to inform beat placement. Onset times are extracted from the audio signal through a thresholding process on an onset detection function that emphasises changes in the high frequency content of signals. Initial estimates of the beats are then calculated by

evaluating beat pattern templates at each onset location and using some smoothing to avoid sharp changes in tempo.

To improve these beat estimates, musical structure is incorporated into the model with the hypothesis that if two sections of music are similar (in terms of their harmonic development), we would expect them to have a similar beat structure. The technique compares the "structural consistency" of harmonically similar regions – determined by a self-similarity matrix calculated from harmonic information in the form of chroma vectors.

By using structural information to inform beat estimates, substantial improvements in performance are demonstrated over the 'basic' beat tracker (without structural information) – although the database used for evaluation was small in comparison to other studies and the evaluation subjective, rather than objective.

**Klapuri, Eronen and Astola**

Klapuri et al. [2006] present a model that builds upon the comb filter approach of Scheirer [1998]. While the Scheirer model analyses the tempo at each of six wide sub-bands, this leaves it unable to detect harmonic changes as the bands are not narrow enough to identify specific frequency components. Klapuri et al. also observe that using a number of bands sufficient to identify harmonic change would render each of those bands too narrow to be analysed for rhythmic periodicity.

As a solution, 36 sub-bands are calculated, the derivatives of power envelopes are taken and then each nine adjacent sub-bands are summed to produce a four channel onset detection function. Each of the four channels contains information about harmonic change due to the initial approach with the narrower 36 bands, but they now represent four broader frequency ranges and are suitable for the tracking of periodic rhythmic components.

The four channels are passed through a bank of comb filter resonators and the outputs are summed to produce an observation function. This observation function is then passed to a probabilistic model for jointly estimating the period at three metrical levels – the *tatum*, the fastest

metrical level [Bilmes, 1993], the *tactus*, the 'beat' or 'foot-tapping' rate and the measure level.

Finally, these period observations are used, in combination with the outputs of the resonators, to estimate the phase of the beats in the signal using two parallel hidden Markov models for the tactus and measure phases respectively. The tatum phase is estimated from the tactus phase.

#### Laroche

Laroche [2003] has presented an approach based upon dynamic programming. First an 'energy flux' onset detection function is calculated by summing the first order differences between corresponding bins in adjacent audio frames – and half-wave rectifying the resulting signal in order to keep only positive differences.

For each analysis frame, the cross-correllation is calculated between the energy flux signal and a number of 'expected' energy flux signals (created by spacing discrete pulses by a given beat period hypothesis and started from a given hypothesis of the downbeat location). The result is a matrix of correlation values over various tempi and downbeat locations. The number of candidates is then reduced: After normalising the matrix a tempo likelihood function is created by taking the best downbeat candidate for each tempo candidate. The local maxima of this function are searched for the 10-15 best candidate tempi. Then, for each of these, the 10-15 best candidate downbeat locations are selected.

The result of this processing is a number of tempo and downbeat candidates at each analysis frame. A dynamic programming technique is then used to select the best sequence of candidates, using a transition score that enforces the constraints that the tempo changes are smooth and the downbeat locations are consistent with the tempo.

#### Ellis

Ellis [2007] has also presented an algorithm for beat tracking based upon

dynamic programming. An onset strength envelope is extracted by resampling the input audio to 8kHz and extracting short-time Fourier transform frames. These are converted to an "approximate auditory" representation with 40 bands on the Mel scale – a perceptual frequency scale. The first-order difference in each band is taken and the positive differences summed across all bands and the final function smoothed.

From this function a global tempo estimate for the signal is calculated from the weighted output of an auto-correlation function. This global tempo estimate is then used with the onset strength envelope to create a transition cost function which in turn is used in the calculation of a recursive function with peaks at likely beat locations. Finally, a "backtrace" step is performed to identify the optimal beat sequence.

#### Davies and Plumbley

Davies and Plumbley [2007] present a system that calculates an onset detection function based upon the complex spectral difference between adjacent Fourier transform frames. The auto-correlation of the onset detection function is then calculated and passed through a weighted comb filterbank. The output of this comb filterbank has peaks at lags that match the periodicities in the auto-correlation function and is used to estimate the beat period (or time between beats). Analysis of the most recent single beat period of the onset detection function is then performed to extract the beat phase. Based upon this information beat times can either be estimated offline or predicted into the future in real-time.

Two different states are used for beat tracking that have different tempo weightings that bias the choice of tempo from the output of the comb filterbank. The general state has a wide weighting that favours common tempi. Should a reasonably consistent tempo be observed then a context-dependent weighting is created using a narrow Gaussian that favours tempi close to the current observation. Should the general state observe a consistent tempo at one different to that of the context-dependent state, then a new context-dependent state is created based upon this tempo.

### Collins

Collins [2005] has presented a beat tracker specifically focused upon the real-time tracking of an acoustic drum kit. The approach is based upon a causal implementation of the approach of Laroche [2003]. However, stressing the difficulty of consistently determining the phase position of beats, higher level information is incorporated into the dynamic programming stage. Specifically, information about kick drum and snare drum events – inspired by Goto [2001] – is compared to a number of drum patterns, resulting in a pattern score. Also, the hypothesis that the beat will often coincide with the presence of low frequency energy is used to calculate a bass cost. To prevent spurious tempo changes, a tempo/phase candidate must be selected twice consecutively before being accepted.

The model was implemented in SuperCollider and used in a system for matching algorithmically generated drums to a human drummer.

### Robertson and Plumbley

Robertson and Plumbley [2007] present a system designed specifically for a live performance situation and for the analysis of drums. The system assumes an approximately steady tempo, that the approximate tempo is known beforehand and that the bass drum signal is rhythmical and reasonably uncomplicated. Onsets are extracted through the analysis of the bass drum and compared to the expected beat times using a Gaussian weighting. Should an onset be accurate enough and exceed a threshold then the system tempo is updated according to the difference between the expected and actual onset time. The system is capable of automatically adjusting the width of the gaussian based upon the accuracy of the drummer. The system is able to speed up and slow down to remain in time with a live performance in a way that allows the implementation of an auto-accompaniment system for live performance

### Degara el al.

Degara et al. [2011] present a probabilistic technique that explicitly models both beat and non-beat states – and provides an accompanying reliability

measure to automatically estimate the reliability of tracked beats.

The first step uses a complex spectral difference onset detection function as to calculate a beat period salience signal – the output of the comb filter bank in the tempo estimation phase of the Davies and Plumbley [2007] approach. This signal is calculated at time instants throughout each audio file and is used to calculate a beat period path – expected to be a slowly varying process.

A hidden Markov model is then used to estimate beat times – with states representing the amount of elapsed time since the last beat in frames. For any given state there are only two possible transitions – to the next state or to the first state (the 'beat state'). The most likely state sequence is decoded using the Viterbi algorithm and the beat times taken to be when the beat state was visited in the state sequence.

The authors note that if the beat period salience signal is poor, then beat tracking quality can also be poor. This observation is used to create a reliability measure for beat tracking. By using a k-Nearest Neighbour algorithm to learn the relationship between three features of the beat period salience signal and beat tracking performance, a beat period salience signal from an arbitrary input signal can be assessed for likely beat tracking quality.

#### Real Time vs. Offline Beat Tracking

Of the approaches discussed here, a number are purely 'offline' in that they are non-causal solutions whereby future information is used to estimate current beat times. These include Dixon [2007], Ellis [2007] and Degara et al. [2011]. For estimating beats in a live performance, we need a model that operates causally and in real-time. Of the models discussed here, several are either causal or present causal versions of their algorithms including Scheirer [1998], Goto [2001], Collins [2005], Klapuri et al. [2006], Robertson and Plumbley [2007] and Davies and Plumbley [2007].

For more information on beat tracking, reviews are presented in Gouyon and Dixon [2005], Gouyon [2005], Hainsworth [2006] and McKinney et al. [2007].

## 2.3 Harmonic Analysis

Harmonic developments in polyphonic music can provide much information about a live performance. In the immediate sense, information about the harmonic content of audio signals allows musicians to play coherent, harmonically relevant accompaniments with one another. Furthermore, changes in the harmonic content over time can indicate structural properties of a piece of music, including bar boundaries and the dominant themes of a piece.

This harmonic and structural information can be automatically extracted by analysis algorithms in real-time and may be used to develop effective live performance tools and increase the ability of computer systems to participate in live performances with musicians.

In this section we present a review of previous research on the extraction of harmonic information from musical signals. We first detail a number of techniques for extracting harmonic information in the form of *chroma features* from audio signals. We then proceed to discuss the use of these features in *automatic chord recognition*.

### 2.3.1 Chroma Features

In the majority of previous approaches to automatic chord recognition [Harte and Sandler, 2005; Bello and Pickens, 2005; Mauch and Dixon, 2008], the first step has been to convert audio frames into a $12 \times 1$ vector representation with values representing the energy present in each of the 12 semitone pitch classes found in western music. This feature is called a *chroma vector*, and is also commonly referred to as a *pitch class profile (PCP)* or *chromagram* [Fujishima, 1999]. An example chroma vector can be seen in Figure 2.2.

#### Constant-Q Based Chroma Features

Several different techniques have been used to calculate chroma vectors. Some systems [Bello and Pickens, 2005; Harte and Sandler, 2005; Lee and Slaney, 2008] make use of the constant-Q transform [Brown, 1991].

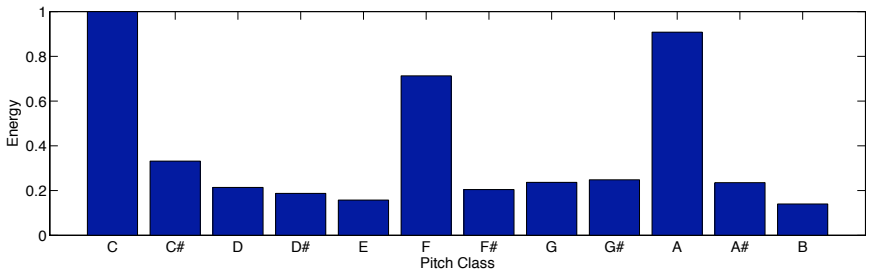



Figure 2.2: An example of a chroma vector or chromagram.

The constant-Q transform is a frequency domain representation with logarithmically spaced frequencies that follow those of the equal tempered scale. The constant-Q transform is generally calculated from some base frequency and for a number of octaves. A chromagram can be simply calculated by summing each pitch class over all octaves.

A variations on this technique has been presented by Harte and Sandler [2005] who have accounted for the problem of instruments deviating in tuning from standard concert pitch. To achieve this, a 36 bin chromagram is calculated to which a tuning algorithm is applied to create a 12 bin chromagram that accounts for tuning variations.

#### Chroma Features Calculated From the DFT

Other techniques calculate the chromagram directly from the Discrete Fourier Transform (DFT) of the input signal. Fujishima [1999] calculates each bin of the chroma vector by summing all spectral bins in the DFT that belong to that pitch class. Sheh and Ellis [2003] employ a similar technique but use a quarter-tone resolution for the chromagram to allow for differences in tuning.

A particular problem identified by several authors has been the presence of energy from the harmonics of pitched notes – tones at integer multiples of the fundamental frequency – in the chromagram. For example, the 3rd harmonic will add energy to the chroma bin of the note an interval of a fifth above the fundamental – even if that note was not sounded in the signal.

Potential improvements have been suggested by Lee [2006], who has

proposed an 'enhanced pitch class profile'. By multiplying the energy at each frequency in the magnitude spectrum by its harmonics, the Harmonic Product Spectrum (HPS) of the DFT is calculated. In this representation the fundamental frequencies are emphasised over harmonics of those fundamentals. The HPS is then used to calculate the chromagram rather than the DFT itself. Cremer and Derboven [2004] present a technique that uses a frequency warped FFT followed by the erasing of overtones of fundamental frequencies and the separation of tonal components from transients.

Peeters [2006] proposes the use of a 'harmonic peak subtraction function' to reduce the influence of higher harmonics of each pitched note when calculating the chromagram. This process is applied to the magnitude spectrum of the signal and the result mapped to a chroma representation. In other work, Gómez [2006] has suggested a Harmonic Pitch Class Profile where only spectral peaks are mapped to chroma bins and the contribution of harmonics to the fundamental frequency that produced them is accounted for.

#### Other Approaches

Müller et al. [2009] has suggested a technique for increasing the robustness of chroma features to timbral changes. The approach is based upon the observation that lower mel-frequency cepstral coefficients (MFCCs) are closely related to timbre. The energy in 120 frequency bands corresponding to MIDI pitches is calculated from the input signal. The Discrete Cosine Transform (DCT) is then taken of these 120 pitch values. The result is similar in nature to the MFCCs. By discarding the lower coefficients, performing an inverse DCT and mapping the resulting pitch values to a chroma scale, the authors create a chroma feature with reduced sensitivity to changes in timbre.

### 2.3.2 Chord Classification

Once the chromagram has been calculated, several techniques can be used to classify the chromagram and thus give it a chord label. These approaches can be largely grouped into template matching techniques and statistical approaches.

#### Statistical Approaches

Several attempts have been made to classify chords using statistical techniques, in particular hidden Markov models (HMMs). Sheh and Ellis [2003] use Expectation-Maximization (EM) trained HMMs to recognise chords and chord boundaries with a single Gaussian used to model the observations. The recognition performance is poor, possibly due to the small size of the training set, but when the sequence of chords is provided to the system, it can recognise chord change boundaries with relative accuracy.

Bello and Pickens [2005] present a HMM-based technique that incorporates musical knowledge into the initialisation of the model. The state transition matrix is informed by the circle of fifths, making transitions to chords related in this way more likely than others. Also, the mean and covariance matrix of the multi-variate Gaussian used to model each observation is informed by music theoretic notions of the relationship between musical pitches. The HMM operates upon beat-synchronous segments – created using the Davies and Plumbley [2007] beat tracker – based upon the hypothesis that chord changes are more likely to occur at beat locations. Increased performance is demonstrated for the beat-synchronous approach over a frame-by-frame approach where no beat information is used.

Lee and Slaney [2008] make use of the fact that knowledge of the musical key – the harmonic centre of the music – implies that certain chords are more likely than others. They implement 24 different HMM-based chord detectors for the 12 major and 12 minor keys. Each HMM has a unique set of state transitions and uses a single multivariate Gaussian to model each observation. This approach is used to simultaneously estimate

chord and key information. The performance of the model using two different types of features is examined. The first feature is a 12-dimensional chroma vector similar to those used by other models. The second is a 6-dimensional tonal centroid vector (used by Harte et al. [2006] for the purpose of detecting harmonic changes in audio signals). The tonal centroid vectors result in better performance than the chroma features for this approach.

Papadopoulos and Peeters [2007] compared and contrasted several different techniques for defining observation probabilities and state transition matrices when using HMMs for chord recognition. The best results were for modelling the observation probabilities through correlations with a set of chord templates and using a state transition matrix based upon the proximity of musical keys derived through perceptual tests.

Many of the above models use a single multivariate Gaussian to model the observation distribution. Others, such as Mauch and Dixon [2008], have used Gaussian mixtures to model a chord as being made up of a number of sonorities. Bass and treble chromagrams are used as observations for a HMM-based approach. Gaussian mixtures have also been used for modelling observations by Khadkevich and Omologo [2009]. However, it has been shown by Cho et al. [2010] that the benefits brought by more complex models are limited and largely offset by appropriate parameterisation and pre- and post-filtering strategies.

Elsewhere, artificial neural networks have been used to classify chords directly from a chromagram [Zhang and Gerhard, 2008]. In this particular case, the specific voicing constraints of a guitar are taken into account when considering the likelihood of a given chord.

#### Template Matching Approaches

Several pattern matching techniques have been used to classify chroma features as one of a number of chords. These generally compare how similar the chroma vector is to a set of chord profiles, usually in the form of a bit mask. The bit mask representation is a 12-dimensional vector containing a 1 where notes are present and 0 elsewhere. A C Major triad

would thus be represented as [1,0,0,0,1,0,0,1,0,0,0,0]. The nature of this representations allows direct comparison with chroma vectors.

Fujishima [1999] compares two chord classification techniques. The first is a 'nearest neighbour' method where the selected chord is the one that minimises the distance, $\delta_i$, between the $i$th bit mask $T_i$ and the chromagram, $C$:

$$\delta_i = \sum_{p=0}^{P-1} (T_i(p) - C(p))^2 \tag{2.1}$$

where $P = 12$, the number of notes in an octave.

The second technique is a 'weighted sum' method where the selected chord maximises $\Delta_i$ – the dot product between the chromagram, $C$, and a weighted version of the $i$th bit mask, $W_i$:

$$\Delta_i = \sum_{p=0}^{P-1} (W_i(p) \cdot C(p))^2 \tag{2.2}$$

where $P = 12$, the number of notes in an octave. The weighting of the bit mask allows chords with different numbers of notes to be compared effectively (given that chords with more notes – and hence more '1s' in the bit mask – will naturally produce larger sums than chords with fewer notes). The weighting also allows more common chords to be favoured over those that are less common.

Harte and Sandler [2005] use a similar classification technique to the weighted sum presented by Fujishima [1999], however the chords that the algorithm attempts to classify are all three note chords, so no weighting is used. A technique similar to the weighted sum has also been used by Cremer and Derboven [2004] for chord recognition.

Oudre et al. [2009] experiment with chord models that deviate from a pure bit mask, adding energy to account for either 1, 4 or 6 harmonics. Also, a number of different measures of fit are considered – the Euclidean distance, Itakura-Saito divergence and the Kullback-Leibler divergence. The authors found that the single harmonic chord model and Kullback-Leibler divergence were the best choice for an evaluation on 13 Beatles albums.

#### Non-Chromagram Approaches

One exception to the chromagram approach is that of Nawab et al. [2001]. Using the result of a constant-Q transform, the authors consider the problem as one of identifying all resolvable and unresolvable fundamentals. Resolvable fundamentals are those that are not obscured by the harmonics of other fundamentals occurring lower in frequency. Should a harmonic occur too close in frequency to another fundamental, it is considered unresolvable. The approach presented considers the lowest fundamental to be resolvable, stores it and then removes its harmonics. The process is repeated until all resolvable fundamentals have been found. Should more than one chord be possible given the resolvable fundamentals, then various test cases are used to look for evidence of unresolvable fundamentals.

Of the chord detection techniques presented here, MIREX 2010 Chord Description weighted overlap ratio scores were: Cho et al. [2010] (0.78), Khadkevich and Omologo [2009] (0.78), Oudre et al. [2009] (0.74) and Papadopoulos and Peeters [2007] (0.68). The highest scoring model was presented by Mauch and Dixon [2010] (0.79). The results are available online[1].

## 2.4 Existing Technology Infrastructure

There already exists a wide range of commercial and non-commercial software environments that allow the design of systems for live musical performance. We will discuss the ones relevant to this thesis here.

#### Max/MSP

*Max/MSP*[2] is a real-time graphical programming language for music and multimedia. With a range of functionality the user can create a program by moving connectable 'objects' around a 'patch'. The environment allows the processing of MIDI, audio, video, images and sensor information.

[1] MIREX 2010: Audio Chord Description: `http://nema.lis.illinois.edu/nema_out/mirex2010/results/ace/summary.html` (Accessed 18/08/2011)

[2] Max/MSP: `http://cycling74.com/products/maxmspjitter/` (Accessed 27/02/2011)

A key benefit of Max/MSP from our perspective is the ability to design 'externals' (or third party objects) using programming languages such as C++. We can design an external to take any form of input (audio, MIDI or other data) and send any form of output. As a result, and combined with its real-time functionality, this makes it highly useful as a tool for designing musical audio analysis tools as modular units within real-time musical applications.

It is important to note that there are several environments similar in capability to Max/MSP including *SuperCollider*[3], *Pure Data*[4], *CSound*[5] and *Chuck*[6]. We choose Max/MSP for reasons of familiarity with the language and its external programming API.

#### Ableton Live

Software sequencer *Ableton Live*[7] differs to many others in that it has an interface for live performance, allowing easy sound manipulation and triggering of samples. In particular it has built in capabilities for time-stretching audio samples. Due to easy integration with Max/MSP through MIDI communication it is (for our purposes) a useful tool for audio sample playback. Furthermore, Max/MSP has now been fully integrated within Ableton Live with the *Max for Live*[8] extension to Ableton Live.

#### openFrameworks

Software library *openFrameworks*[9] allows easy control and manipulation of video and graphics through the C++ programming language. Through MIDI and OSC it can communicate with Max/MSP, making it very useful for visual applications informed by music.

[3]SuperCollider: `http://supercollider.sourceforge.net/` (Accessed 27/02/2011)
[4]Pure Data: `http://puredata.info/` (Accessed 27/02/2011)
[5]CSound: `http://csound.sourceforge.net/` (Accessed 27/02/2011)
[6]Chuck: `http://chuck.cs.princeton.edu/` (Accessed 27/02/2011)
[7]Ableton Live: `http://www.ableton.com/live-8/` (Accessed 27/02/2011)
[8]Max For Live: `http://www.ableton.com/maxforlive` (Accessed 16/07/2011)
[9]openFrameworks: `http://www.openframeworks.cc/` (Accessed 27/02/2011)

## 2.5 Summary

In this chapter we have outlined research relevant to the work in this thesis. In particular we have studied technologies for use in interactive performances involving human musicians and machines and outlined a means for classifying such systems. We have also discussed two forms of musical audio analysis – the extraction of the dominant metrical pulse from musical signals, or beat tracking, and harmonic analysis, in particular chroma analysis and chord recognition.

# Chapter 3

# Real-Time Musical Audio Analysis

While sound is essentially a physical phenomenon, it is the ways in which it manifests itself in the mind of the listener that allows us to describe some sounds as 'musical' [Lerdahl and Jackendoff, 1983]. If the listener is a musician, then such interpretation of sound is fundamental to participating in a musical performance.

In order to be able to take part in a musical performance, one must first be able to listen to – and comprehend in some way – the developments of the piece in question. There must be some underlying understanding of musical concepts such as tempo, beat and harmony, amongst others. For computer systems, the ability to make the conversion from raw sound waves to 'musical information' is vital if these systems are to be able to adapt their behaviour in a way that is related to musical developments in a performance. Therefore, we first turn our attention to the topic of extracting high level *musical features* from audio. Our focus is on developing techniques for analysis that are real-time, robust and computationally efficient.

In this chapter we focus upon two characteristics of music in particular: *beat* and *harmony*. We present an efficient real-time beat tracker capable of discovering and following the beat in musical performances. We then describe an efficient algorithm for calculating a harmonic representation known as a *chromagram* in order to represent harmonic developments in musical performances. Building upon this, we use the chromagram representation as input to a *chord recognition* algorithm to append high level musical labels to 'chunks' of harmonic information.

Finally, we integrate these elements, presenting a technique for calculating, in real-time, beat-synchronous sequences of harmonic information or labels. These sequences represent information about the temporal and harmonic structure of a performance and give us a framework upon which we can build musical applications.[1]

## 3.1 A Real-Time Beat Tracker

Beat tracking is the automatic determination of the dominant pulse in music. The difficulty of the task is inferring a regular beat from a signal in which the beat is implicit – the musical events in the music may often occur on the beat, but they will also occur at other times and some beats will be marked by no event.

In developing an automatic beat tracker for live performance, we present ourselves with two further challenges. Firstly, we wish to detect beats in real-time, and consequently we have no access to future information. Therefore, we must develop a *causal* approach where past and present information alone are used to inform predictions of future beat locations. Secondly, in order to actually run in real-time the algorithm must be *computationally efficient*.

In this section we present `btrack~` – a real-time beat tracking model. The model draws on two existing systems – the tempo induction of the Davies and Plumbley [2007] method and the dynamic programming approach of Ellis [2007].

### 3.1.1 Input Feature

The first stage of our algorithm is to compute an input feature from the incoming audio signal. We choose as an input feature the *complex spectral difference* onset detection function [Bello et al., 2004] as it has been shown previously to be the strongest in a comparison of 172 different input features [Gouyon et al., 2007]. This feature is large when signals are unstable and small for steadier state signals. The onset of musical notes

[1] The work in sections 3.1 and 3.4 was completed collaboratively with Matthew E. P. Davies

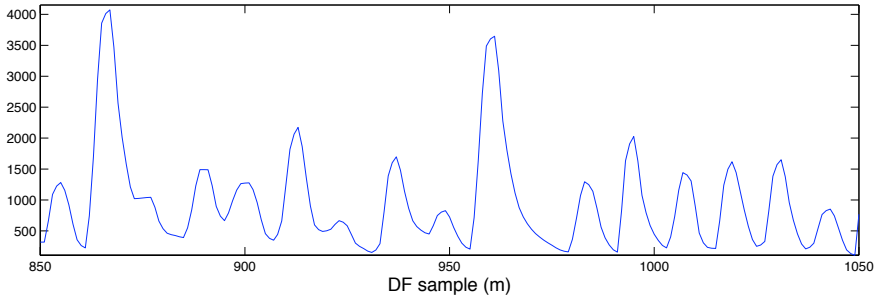



Figure 3.1: An example of a complex spectral difference onset detection function

are often marked by attack transients, which are short and unpredictable segments of sound [Bello et al., 2004]. As a result, this function produces peaks at likely note onset locations. The following is a description of the calculation of this function.

We calculate the input feature with a temporal resolution of 11.6ms, following both Davies and Plumbley [2007] and Gouyon et al. [2007]. Taking an input signal with a sampling frequency of 44.1kHz, we achieve this by processing audio frames of 1024 samples in length with a 512 sample hop size.

For the $m$th audio frame, we calculate a spectral frame $X_k(m)$, where $k$ is the bin number, by first multiplying the frame by a hanning window and then performing a Fourier transform. We then calculate the $m$th complex spectral difference onset detection function sample $\Gamma(m)$ as the Euclidean distance between the observed spectral frame $X_k(m)$ and a predicted spectral frame $\hat{X}_k(m)$:

$$\Gamma(m) = \sum_{k=1}^{K} \sqrt{\left\{ \left[ \Re(X_k(m)) - \Re(\hat{X}_k(m)) \right]^2 + \left[ \Im(X_k(m)) - \Im(\hat{X}_k(m)) \right]^2 \right\}} \tag{3.1}$$

where $K = 1024$, the number of samples in the audio frame. The predicted spectral frame $\hat{X}_k(m)$ is based upon the energy and expected phase values derived from the previous spectral frame $X_k(m-1)$. See Bello et al. [2004] for a full derivation. An example complex spectral difference onset detection function can be seen in Figure 3.1.

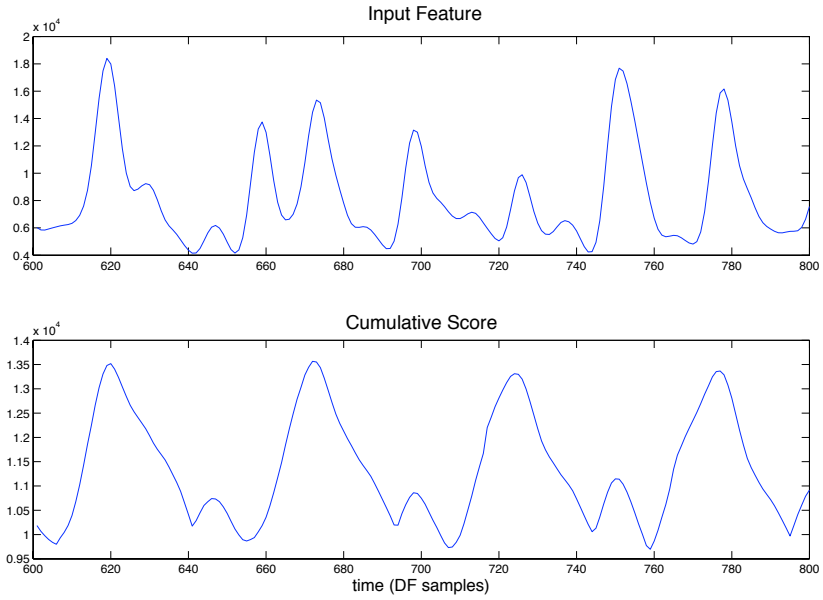



Figure 3.2: An example of an input feature and the corresponding cumulative score

### 3.1.2 Phase Determination: Recursive Feature

The aim of our beat tracking technique is to causally decode the sequence of beats from the input feature. The basis for this technique is a recursive feature, $C^*$, called the *cumulative score* [Ellis, 2007], which is calculated from the input feature, $\Gamma(m)$. At a given point $C^*(m)$, the cumulative score represents the best possible score of all possible beat sequences ending at the point $m$. It is a regular function with peaks at likely beat locations. An example of a cumulative score calculated from an input feature can be seen in Figure 3.2.

The cumulative score is calculated as a weighted sum of: 1) the current input feature sample $\Gamma(m)$ as defined in section 3.1.1; and 2) the beat likelihood at the current input feature sample, $\Phi(m)$, which we turn to now.

Specifically, to determine the beat likelihood at the current input feature sample, we examine the past values of the cumulative score over the interval $[m - 2\tau_b, m - \tau_b/2]$ where $m$ is the current input feature sample and $\tau_b$ is the *beat period* – the time between beats in detection function

samples at the time of beat $b$. We firstly define a log-Gaussian transition weighting that favours the time exactly $\tau_b$ samples in the past:

$$W_1(v) = \exp\left(\frac{-(\eta \log(-v/\tau_b))^2}{2}\right) \tag{3.2}$$

where $v = -2\tau_b, ..., -\tau_b/2$, $\eta$ is the 'tightness' of the transition weighting and $\tau_b$ is the beat period or time between beats. We shall return to the calculation of $\tau_b$ in section 3.1.4. We then calculate the beat likelihood at the current sample, $\Phi(m)$, by choosing the maximum value of the cumulative score over the period $[m - 2\tau_b, m - \tau_b/2]$ after applying the transition weighting $W_1$:

$$\Phi(m) = \max_v(W_1(v)C^*(m + v)) \tag{3.3}$$

where $m$ is the current detection function sample and $v = -2\tau_b, ..., -\tau_b/2$. This process is depicted in Figure 3.3.

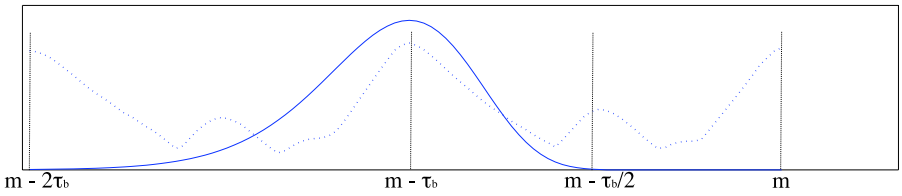



Figure 3.3: Using a log-Gaussian transition weighting (solid line) over the past of the cumulative score (dotted line) to determine the beat likelihood at the current detection function sample, $m$. $\tau_b$ is the beat period in detection function samples.

Finally, we calculate the current sample of the cumulative score, $C^*(m)$, using a weighted sum of the current detection function sample $\Gamma(m)$ and the beat likelihood at the current detection function sample, $\Phi(m)$:

$$C^*(m) = (1 - \alpha)\Gamma(m) + \alpha\Phi(m) \tag{3.4}$$

where $\alpha$ determines the weighting of the two components.

We choose the parameter settings $\alpha = 0.9$ and $\eta = 5$. We will present a detailed explanation of these choices in section 3.1.7.

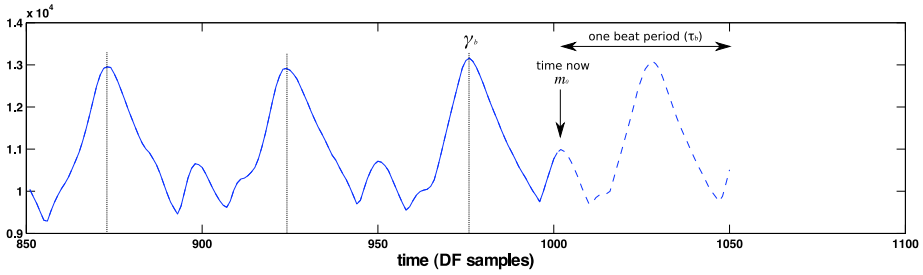



Figure 3.4: The generation of the cumulative score (solid line) into the future (dotted line) from the point $m_0$ for one beat period $\tau_b$ into the future. $\gamma_b$ is the most recently predicted beat.

The cumulative score $C^*$ is now updated with every new detection function sample $\Gamma(m)$, allowing calculation in real-time.

### 3.1.3 Phase Determination: Predicting Future Beats

The approach of Ellis [2007] is non-causal because, as described in Chapter 2, it first calculates the cumulative score for the entire signal – storing the location of the best previous beat for each sample $m$ – before finally performing a *backtrace* from the final beat identified as the maximum value in the final beat period of the cumulative score. In order to implement a beat tracker in real-time, we must make predictions of future beat locations without observing the entire signal.

In order to achieve this, our approach takes advantage of the fact that the cumulative score $C^*$ is a recursive feature. This causes it to maintain some periodic momentum even in the presence of silence. We make use of this property by continuing to generate the cumulative score, $C^*$, into the future. We do this at a fixed point, $m_0$, after the most recent beat, $\gamma_b$, has elapsed, $m_0 = \gamma_b + \tau/2$ and we generate this 'future cumulative score' – the dashed line in Figure 3.4 – for one beat period into the future.

To this one-beat window of the future cumulative score, we apply a Gaussian weighting, $W_2$, with $\mu = \frac{\tau_b}{2}$ and $\sigma = \frac{\tau_b}{2}$ so that it is centred on the most likely beat location (i.e. $m_0 + \tau_b/2$), generating a beat expectation function, $\Psi$:

$$\Psi(v) = \Phi(m_0 + v)W_2(v) \tag{3.5}$$

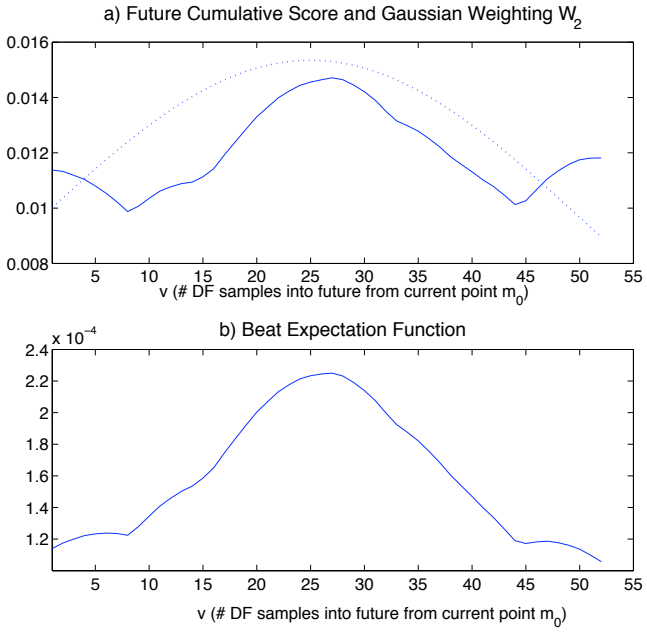



Figure 3.5: a) The future of the cumulative score (solid line) and the Gaussian weighting $W_2(v)$ (dotted line). b) The beat expectation function $\Psi(v)$ – the index of the maximum value of this function determines the location of our beat prediction.

where $v = 1, ..., \tau_b$ specifies the future one-beat window, $\Phi$ is equivalent to setting $\alpha = 1$ in equation 3.4 and $W_2$ is the Gaussian weighting defined by:

$$W_2(v) = \exp\left(\frac{-(v - \tau_b/2)^2}{2(\tau_b/2)^2}\right). \tag{3.6}$$

The future cumulative score and Gaussian weighting $W_2$ can be seen in Figure 3.5 a). The resulting beat expectation function $\Psi$ can be seen in Figure 3.5 b).

Finally, we predict the next beat, $\gamma_{b+1}$, to be:

$$\gamma_{b+1} = m_0 + \arg\max_v \Psi(v) \tag{3.7}$$

where $v = 1, ..., \tau_b$.

### 3.1.4 Tempo Induction

In the approach presented by Ellis [2007] a global tempo estimate is used for each audio signal. We are not able to observe the entire signal as we are presenting a real-time algorithm. Furthermore, we wish to be able to track beats in signals that vary in tempo. As a result, we incorporate the tempo estimation stage of the Davies and Plumbley [2007] model to estimate the value of $\tau_b$, the beat period estimate at the $b$th beat $\gamma_b$.

At the $b$th beat, $\gamma_b$, we take the previous six seconds worth of samples from the onset detection function $\Gamma(m)$. This is a 'snapshot' of the recent past of the audio signal long enough to determine the tempo from and short enough to be relevant to the time at which the tempo is being estimated.

The peaks in this function are preserved using an adaptive moving mean threshold – a local average is subtracted from each sample and negative values set to zero, resulting in a modified detection function $\tilde{\Gamma}(m)$.

The auto-correlation of this function is then taken and passed through a shift-invariant comb filterbank weighted by a tempo preference curve. The output of this comb filterbank, $R(l)$, has peaks at likely candidates for the beat period. Initially, we select the beat period as the lag with the maximum value in $R(l)$ (see top of Figure 3.6). For a more detailed description of the calculation of $R(l)$, see Davies and Plumbley [2007].

It has been previously observed that humans may tap the tempo at different metrical levels [McKinney et al., 2007]. Furthermore, previous beat trackers have suffered from the problem of switching metrical levels [Davies and Plumbley, 2007]. As a result, we restrict the range of possible tempi to a single tempo octave from $t_{\min} = 80$ beats per minute (bpm) to $t_{\max} = 160$ bpm. We map the output of the comb filterbank $R(l)$ from the lag domain to the tempo domain between $t_{\min}$ and $t_{\max}$ to give $R_b(t)$ by:

$$R_b(t - t_{\min}) = R(|60/(f_r \times t)|) \tag{3.8}$$

where $t = t_{\min}, ..., t_{\max}$ and $f_r$ is the temporal resolution of the onset detection function in seconds, in this case $f_r = 0.01161$ seconds. Plots of both $R(l)$ and $R_b(t)$ can be seen in Figure 3.6.

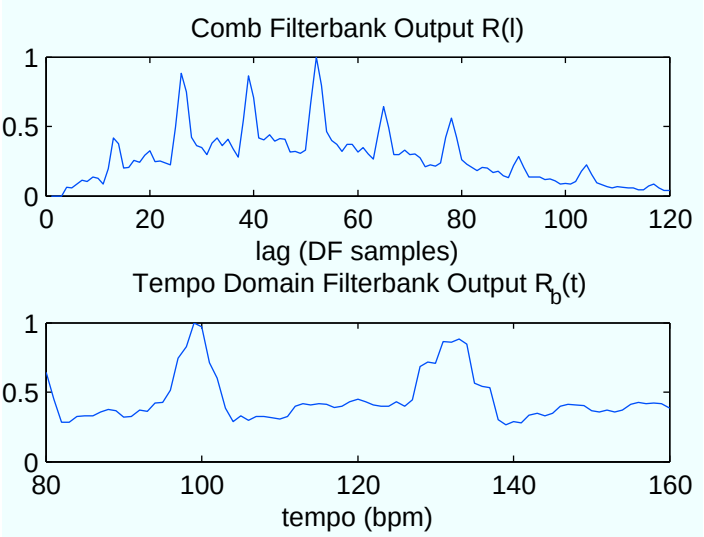



Figure 3.6: Top: The output of the comb filterbank $R(l)$. Bottom: $R_b(l)$ – the result of mapping $R(l)$ into the tempo domain.

While sharp tempo changes in music do occur, tempo in music is largely a slowly changing process. As a result we place transition costs on possible tempo changes to make sharp changes in tempo less likely. We achieve this by making the decision on the current tempo $t_b$ dependent on the previous tempo estimate $t_{b-1}$.

We wish to favour changes in tempo – from the current tempo $t_i$ to a new tempo $t_j$ – that are small so that the new tempo is close to the current tempo. In order to achieve this, we use a transition matrix $A(t_i, t_j)$ where each column contains a Gaussian of fixed standard deviation $\sigma$:

$$A(t_i, t_j) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(t_i - t_j)^2}{2\sigma^2}\right) \tag{3.9}$$

where $t_i, t_j = 1, ..., (t_{\max} - t_{\min})$ and $\sigma = (t_{\max} - t_{\min})/8$ so that each Gaussian is wide enough to capture small changes in tempo but narrow enough to favour the hypothesis that tempo is a slowly changing process. The resulting transition matrix can be seen in Figure 3.7.

At each step, we store the previous tempo likelihood $\Delta_{b-1}$. An example can be seen in Figure 3.8. The stored tempo likelihood $\Delta_{b-1}$ is multiplied by the transition matrix $A$ and the maximum value from each column is taken to create a tempo probability distribution $\theta_b$:

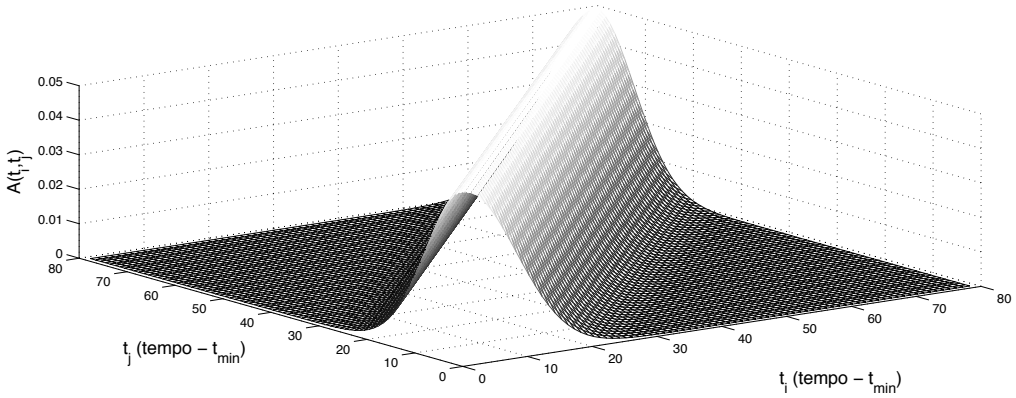



Figure 3.7: The transition matrix $A(t_i, t_j)$.

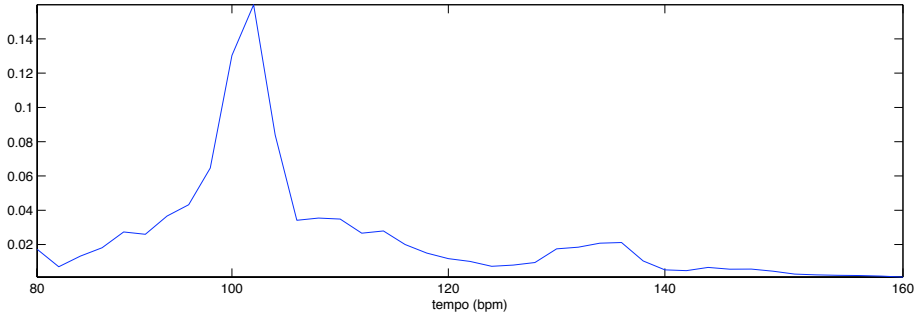


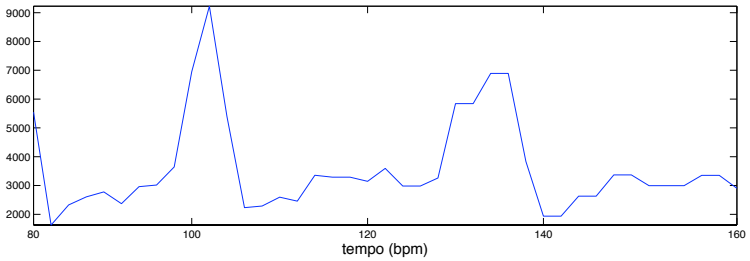

Figure 3.8: The stored tempo hypothesis probabilities from the previous iteration $\Delta_{b-1}$.

$$\theta_b(t_j) = \max_{t_i} \left( A(t_i, t_j) \Delta_{b-1}(t_i) \right) \tag{3.10}$$

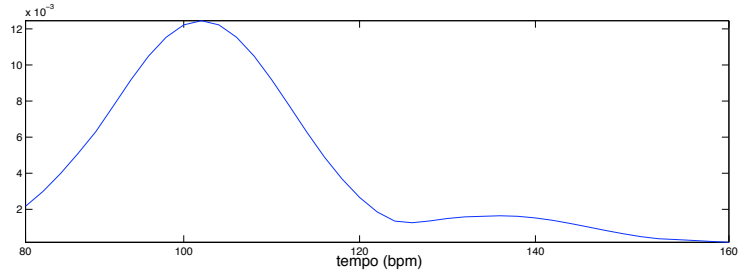
for $t_j = 1, ..., (t_{max} - t_{min})$. An example of the resulting tempo probability distribution $\theta_b$ – created using the transition matrix shown in Figure 3.7 and the example $\Delta_{b-1}$ displayed in Figure 3.8 – can be seen in Figure 3.9(b).

We then calculate the tempo likelihood for the current iteration, $\Delta_b$, (see Figure 3.9(c)) by multiplying the current tempo domain comb filterbank output $R_b$ (Figure 3.9(a)) by the tempo probability distribution $\theta_b$ (Figure 3.9(b)):

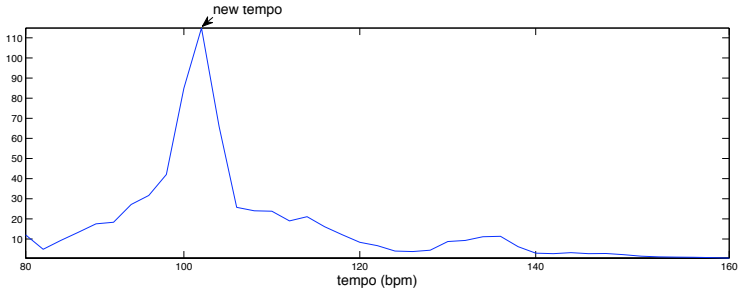
$$\Delta_b(t_j) = R_b(t_j)\theta_b(t_j). \tag{3.11}$$




(a) $R_b$




(b) $\theta_b$




(c) $\Delta_b$

Figure 3.9: a) The output of the comb filterbank $R_b$ showing tempo candidates; b) A tempo probability distribution $\theta_b$; and c) The tempo likelihood for the current iteration, $\Delta_b$, calculated by multiplying $R_b$ and $\theta_b$.

for $t_j = 1, ..., (t_{max} - t_{min})$. Finally, to prevent $\Delta_b$ approaching zero, we normalise it to sum to unity. We then find the current tempo $t_b$ as the index of the maximum value of $\Delta_b$:

$$t_b = t_{\min} + \arg\max_{t_j}(\Delta_b(t_j)). \tag{3.12}$$

Finally, we convert the tempo (in bpm) $t_b$ to the beat period $\tau_b$ in detection function samples by:

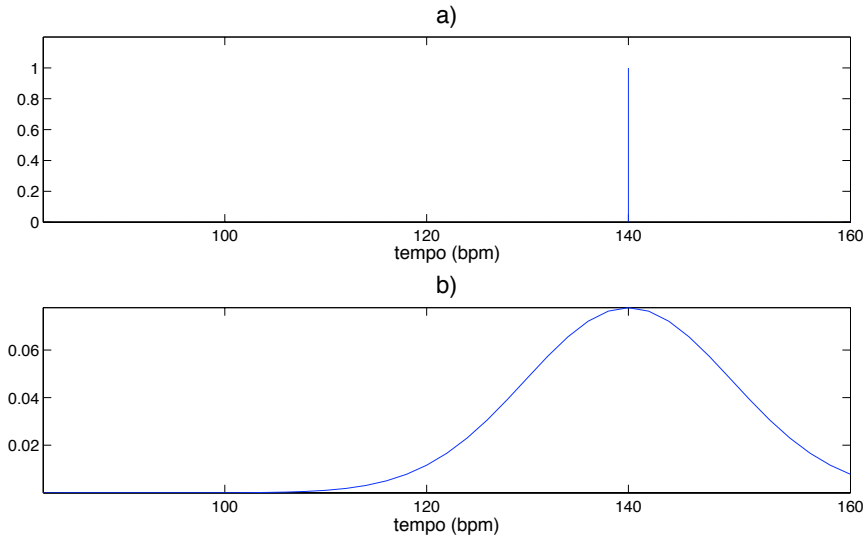



Figure 3.10: a) The creation, for the purposes of a tempo initialisation, of an artificial tempo likelihood for the previous beat $\Delta_{b-1}$ by setting all values to zero except for a '1' at the desired tempo of 140bpm. b) The resulting tempo probability distribution after operations with the matrix. As can be seen, it is biased around the new tempo.

$$\tau_b = \left| \frac{60}{f_r \times t_b} \right| \tag{3.13}$$

where $f_r$ is the resolution of the detection function, in our case 512/44100 = 0.01161 seconds.

### 3.1.5 Using Prior Knowledge

#### 'Count In' Feature

We are attempting to design a real-time beat tracker for live performance. In such a performance situation a useful piece of functionality would be the ability to have some form of 'count in' – an initialisation of the first beat and the tempo. This would allow musicians to indicate to the beat tracker the initial state of any performance, avoiding the process of the beat tracker taking several seconds to find the correct beat. We implement such functionality as follows, updating parts of the model in three steps:

1. We described in section 3.1.4 how the previous tempo likelihood is

used in the calculation of the new tempo. Therefore, to set a tempo artificially for a count in, we change the state of the previous tempo likelihood $\Delta_{b-1}$ so that when the tempo is next calculated it is biased to the new tempo. We set every value in $\Delta_{b-1}$ to zero except for a '1' in the bin indicating the desired tempo. Figure 3.10 shows $\Delta_{b-1}$ from such a tempo initialisation and the result of operations with the matrix, resulting in a tempo probability distribution $\theta_b$ biased around the new tempo.

2. As the initialisation is happening at the very first beat, we indicate that the current detection function sample $m$ is a beat. This means that the next beat prediction will take place at $m + \tau_b/2$.

3. Finally, the cumulative score should contain peaks at likely beat locations and so we must artificially create such a circumstance to reflect the new tempo and beat phase. We therefore fill the cumulative score $C^*$ with delta functions spaced at intervals of the desired beat period. If the current sample is $m$, we will have delta functions at $[..., m-2\tau, m-\tau, m]$ in such a way that they indicate that a beat should occur at the present detection function sample $m$.

**Fixed Tempo Version**

It is possible that we will know in advance that the music we wish to beat track in real-time will occupy a narrow tempo range, for example 120–130bpm. If this is the case, then it may make sense to instruct the beat tracker not to search all other tempi as possible candidates.

In practice, we achieve this in a similar way to step 1 in the tempo update for the 'count in' feature in section 3.1.5. We update $\Delta_{b-1}$ with zeros except for a '1' indicating the desired tempo. The difference to the 'count in' feature is that we do this for every tempo update, rather than as a one off, thus permanently restricting the possible tempo estimates to those around some given tempo.

### 3.1.6 Evaluation

We evaluated our beat tracking model on an existing annotated database [Hainsworth, 2004] comprised of 222 audio files each approximately 60 seconds in length. Each file is accompanied by a sequence of beat annotations, recorded as beat times in seconds and created by a human listener tapping in time to the piece. The genre breakdown of the database is as follows: Rock and Pop (68), Dance (40), Jazz (40), Classical (30), Folk (22) and Choral (22). We evaluated three versions of our model – one with no initialisation (RT), one initialised with a 'count in' as described in section 3.1.5 ($\mathrm{RT}_{\mathrm{count}}$) and one initialised with both a 'count in' and restricted to a narrow tempo range around the main tempo of the song according to section 3.1.5 ($\mathrm{RT}_{\mathrm{count,fix}}$).

We conducted the evaluation by comparing, for a given audio file, a sequence of beat predictions in seconds $\gamma_1, ..., \gamma_B$ from the beat tracker to a sequence of beat annotations at the metrical level $\lambda$ given by $a_1^\lambda, ..., a_J^\lambda$, also in seconds. We assessed the performance of our technique using a number of evaluation measures.

#### Evaluation Measures: $\mathbf{AML}_c$

The $\mathrm{AML}_c$ (Allowed Metrical Levels, with Continuity required) evaluation measure – developed by Hainsworth [2004] and Klapuri et al. [2006] – requires that there be some continuity in the beats output by the beat tracker. Firstly, each beat $\gamma_b$ is considered as correct based upon three conditions. Given the nearest annotation $a_j^\lambda$ (at the metrical level $\lambda$), the inter-beat interval $I_b^\gamma = \gamma_b - \gamma_{b-1}$ and the inter-annotation interval $I_j^{a^\lambda} = a_j^\lambda - a_{j-1}^\lambda$:

1. The distance from the beat $\gamma_b$ to the annotation $a_j^\lambda$ must be within $\pm\rho\%$ of the inter-annotation interval $I_j^{a^\lambda}$.

2. The distance from the previous beat $\gamma_{b-1}$ to the previous annotation $a_{j-1}^\lambda$ must be within $\pm\rho\%$ of the previous inter-annotation interval $I_{j-1}^{a^\lambda}$.

3. The inter-beat interval $I_b^{\gamma}$ must be within $\pm\rho\%$ of the inter-annotation interval $I_j^{a^{\lambda}}$.

where $\rho = 17.5\%$. The $\text{AML}_c$ measure calculates, for each metrical level, the number of correct beats in each continuously correct segment $\Upsilon_z^{\lambda}$ where $Z$ is the total number of segments found. The accuracy score for the metrical level $\lambda$ is then calculated as the ratio of the longest continuously correct segment to the number of annotated beats $J^{\lambda}$ in the annotations:

$$\text{AML}_c^{\lambda} = \frac{\max(\Upsilon_z^{\lambda})}{J^{\lambda}} \times 100\% \tag{3.14}$$

The above score is calculated for several versions of the annotations at different metrical levels - the original annotations, the annotations at double and half the tempo and the annotations on the off-beat (the $\pi$-phase error described by Goto and Muraoka [1997]). Then the final score is taken as the highest score of comparisons with all metrical levels:

$$\text{AML}_c = \max_{\lambda} \text{AML}_c^{\lambda} \tag{3.15}$$

**Evaluation Measures: $\text{AML}_t$**

The $\text{AML}_t$ (Allowed Metrical Levels, total number of correct beats) evaluation measure is a measure of the total number of correct beats, regardless of continuity. Also developed by Hainsworth [2004] and Klapuri et al. [2006], $\text{AML}_t$ uses the same three conditions for accepting beats as correct as $\text{AML}_c$. However, the score for each metrical level $\lambda$ is determined by taking the ratio of the total number of correct beats to the number of annotations, $J^{\lambda}$:

$$\text{AML}_t^{\lambda} = \frac{\sum_{z=1}^{Z}(\Upsilon_z^{\lambda})}{J^{\lambda}} \times 100\% \tag{3.16}$$

We then take the final score to be the maximum score across all metrical levels:

$$\text{AML}_t = \max_{\lambda} \text{AML}_t^{\lambda} \tag{3.17}$$

**Evaluation Measures: LML**

The evaluation measures $\text{AML}_c$ and $\text{AML}_t$ determine a beat to be correct if it falls anywhere within a specified range of an annotation so that regardless of the distance of the beat from the annotation it is considered equally correct. This unfortunately takes no account of the accuracy of beats in terms of localisation. Cemgil et al. [2001] have presented an evaluation measure that penalises beat location with a Gaussian error function – however it does not allow comparison at multiple metrical levels meaning that relevant beats at a different metrical level may be unfairly punished.

We present here our own evaluation measure, LML (Localisation with Allowed Metrical Levels), an extension of the Cemgil et al. [2001] measure, taking into account both localisation of beats and annotations at multiple metrical levels. When comparing each beat, $\gamma_b$, to each annotation, $a_j^\lambda$ at metrical level $\lambda$, localisation is penalised using a Gaussian error function, $W$:

$$W(x) = \exp(-x^2/2\sigma_e^2) \tag{3.18}$$

where $x = \gamma_b - a_j^\lambda$ and we use $\sigma_e = 40$ms (note that $\sigma_e$ is fixed regardless of tempo) following Cemgil et al. [2001]. The accuracy score is then given by:

$$\text{LML}^\lambda = \frac{\sum_{j=1}^{J^\lambda} \max_b(W(\gamma_b - a_j^\lambda))}{(B + J^\lambda)/2}. \tag{3.19}$$

As with $\text{AML}_c$ and $\text{AML}_t$, we then choose the maximum score over all metrical levels as the final result:

$$\text{LML} = \max_\lambda \text{LML}^\lambda \tag{3.20}$$

**Evaluation Measures: Information Gain**

The use of tolerance windows to accept beats as either correct or otherwise involves a somewhat arbitrary choice of the size of that tolerance window. Responding to this, Davies et al. [2011] have introduced an 'information gain' measure that does not rely on tolerance windows.

For each annotation $a_j$, all beats $\gamma_b$ that fall inside a one beat window centered on $a_j$ are considered. The timing error of each of these beats to the annotation is then normalised relative to the one beat annotation window (from half way between annotations $a_{j-1}$ and $a_j$ to half way between $a_j$ and $a_{j+1}$). This is performed for all beats, resulting in the error of the sequence of beats, given the annotations. In order to deal with the case of the under-detection of beats (where some annotations would have no beat errors to consider) the error of the sequence of annotations according to the beats is also calculated. Based upon this information, two beat error distributions are calculated over the range [-0.5,0.5] of a normalised beat period – one for the timing error of beats to annotations and one for annotations to beats.

If we were to place beats at randomly assigned times in the signal, we would expect little coherence in terms of a regular spacing. As a result, we would expect the beat error distribution to be approximately uniform – with errors evenly distributed. The information gain measure is calculated by measuring the Kullback-Leibler (KL) divergence between the observed beat error distribution and a uniform distribution. The resulting measurement indicates how related the two distributions are – lower values indicating that the observations are closer to the uniform distribution and unrelated to the signal, higher values indicate that there is some degree of coherence to the beat error distribution. In more simple terms we have a measure of how much better than random the sequence of beats are. The lowest KL divergence of the two distributions is taken as the final information gain score. For a full derivation, see [Davies et al., 2011].

**Comparison to Other Beat Tracking Models**

We also compared our beat tracking technique to several others described in Chapter 2. These were the approach of Scheirer [1998] (SC), both the causal ($KL_c$) and non-causal ($KL_{nc}$) approaches of Klapuri et al. [2006], the *BeatRoot* algorithm (BR) presented by Dixon [2006a] and the approach of Ellis [2007] (EL).

| **Beat Tracker** | | **AML$_c$** (%) | **AML$_t$** (%) | **LML** (%) | **Info. Gain** (bits) |
|---|---|---|---|---|---|
| *Non-Causal* | DP | 70.5 | 79.1 | 66.6 | 1.91 |
| | KL$_{\text{nc}}$ | 69.7 | 79.3 | 67.2 | 1.96 |
| | BR | 52.3 | 73.1 | 63.0 | 1.56 |
| | EL | 50.4 | 71.6 | 61.3 | 1.48 |
| *Causal* | KL$_{\text{c}}$ | 65.7 | 76.5 | 65.3 | 1.88 |
| | SC | 28.9 | 53.3 | 55.9 | 1.39 |
| *Proposed* | RT | **66.0** | **74.9** | **63.7** | **1.73** |
| | RT$_{\text{count}}$ | 68.0 | 77.2 | 65.5 | 1.83 |
| | RT$_{\text{count,fix}}$ | 69.2 | 79.9 | 65.8 | 1.87 |
| | Human | 57.3 | 87.0 | 63.2 | 1.69 |

Table 3.1: The results of our evaluation on 4 different evaluation measures.

We also compared our algorithm to an offline equivalent of our algorithm developed by Davies and Plumbley (DP) and available to download as a Sonic Visualiser plug-in[2]. Like our model, the DP model combines the tempo estimation phase of the earlier two-state Davies and Plumbley [2007] model and the dynamic programming approach of Ellis [2007]. However, the difference between the DP model and our approach, RT, is that the DP model determines the beat times according to the backtrace approach of Ellis [2007] rather than our predictive approach, causing the DP model to be a non-causal algorithm.

Finally, we included the results of a human annotator (in this case Dr. Matthew Davies) tapping along to each piece of music in real-time.

### 3.1.7 Results

The results of the evaluation on the 222 files in the database can be seen in Table 3.1. As can be seen, our technique with no initialisation (RT) outperforms the Scheirer model (SC) and is comparable to the causal Klapuri et al. [2006] (KL$_{\text{c}}$) technique. It also outperforms two of the non-causal models on all evaluation measures – Dixon's *BeatRoot* (BR) model

[2]http://isophonics.net/QMVampPlugins

| Beat Tracker | Implementation Language | Time Taken For Evaluation |
|---|---|---|
| $KL_c$ | Matlab & C++ | 2224 seconds |
| SC | C++ | 1484 seconds |
| RT | **C++** | **429 seconds** |

Table 3.2: The time taken for each of the three causal beat tracking models to complete the evaluation. As can be seen, our model (RT) was the quickest.

and the Ellis (EL) model.

The initialisation of our model with a 'count in' ($RT_{count}$) results in a small increase in performance followed by another increase when the fixed tempo constraint is included too ($RT_{count,fix}$).

#### Efficiency

We are building a real-time system and so computation time is limited. This means that the computational efficiency of the algorithms is of importance when considering the quality of a real-time algorithm. As a result, we recorded the computation time of the causal models ($KL_c$, SC and RT) by examining how long they took to process the 222 audio files in the database used for the evaluation (around 200 minutes worth of audio)[3]. Our model (RT) took the least time (429 seconds) compared to the causal Klapuri ($KL_c$) model (2224 seconds) and the Scheirer (SC) model (1484 seconds). Our model and the Scheirer model were implemented in C++ while the Klapuri model was a mixture of Matlab and C++. These results can be seen in Table 3.2.

#### Parameterisation

We chose the values of two of the parameters of our model – $\eta$, the 'tightness' of weighting $W_1$, and $\alpha$, the mixing component used in the calculation of the cumulative score $C^*$ – by evaluating various settings on a database of 60 audio files (a different database to that used in the evaluation and the second database used in Chapter 4). We examined values of $\eta$ between

[3]We tested the algorithms on a 2 GHz Intel Mac running OS X 10.5 with 1 GB of RAM

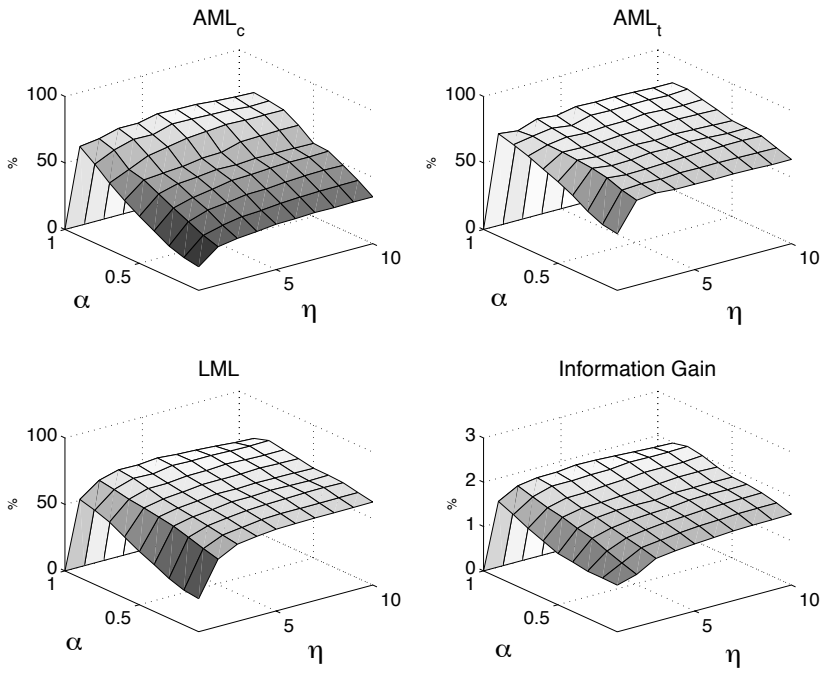



Figure 3.11: Beat tracking accuracy surfaces for different parameter settings of $\alpha$ and $\eta$. All evaluation measures indicate that the ideal parameter settings are $\alpha = 0.9$ and $\eta = 5$. The parameter $\alpha$ appears to have a much greater effect on performance than $\eta$.

1.0 and 10.0 in steps of 1.0 and values of $\alpha$ between 0.1 and 1.0 in steps of 0.1. We examined the results of all 100 combinations of these settings. We chose our parameter settings according to the LML measure as it takes into account localisation as well as just the number of correct beats and the localisation of beats is likely to be affected by these parameter settings.

We have plotted the beat tracking accuracy surfaces in Figure 3.11. The data shows that the effect of varying $\alpha$ is greater than varying $\eta$. We found that for this database the optimal settings for the parameters were $\alpha = 0.9$ and $\eta = 5$.

Our real-time beat tracker was implemented in C++ as a Max/MSP external called '`btrack~`'.

## 3.2 Chroma Analysis

We have seen the presentation of a real-time beat tracking model in section 3.1. However, in order to inform real-time applications we also wish to extract information about the harmonic developments in a performance. Therefore, in order to represent harmonic information in audio signals, in this section we outline a real-time chroma analysis technique.

A *chroma vector* is a $12 \times 1$ vector with values representing the energy present in each of the 12 semitone pitch classes found in western music. Others have presented techniques to calculate similar features in the past. Some [Bello and Pickens, 2005; Harte and Sandler, 2005; Lee and Slaney, 2008] have made use of the constant-Q transform [Brown, 1991]. Others [Fujishima, 1999; Sheh and Ellis, 2003] have calculated the chroma vector directly from the discrete Fourier transform (DFT) of the input signal by mapping the energy in spectral bins to one of a number of pitch classes.

Many of the techniques that calculate chroma vectors directly from the DFT tend to map the energy from all frequency bins in a range around a given pitch class frequency to a pitch class in the chroma vector. A disadvantage of this is that while pitched notes produce harmonics at a single frequency, energy from a number of bins – potentially containing unwanted noise – is included in the chroma vector. As an alternative to these approaches, we present here a technique based upon identifying only the energy within certain harmonics within a given range.

When considering the range of frequencies to examine, we must take into account that lower frequencies require greater frequency resolution to differentiate between semi-tone pitch classes. Furthermore, larger frequency resolutions mean larger frame sizes and in turn larger (and more time consuming) spectral transforms. As a result there is a trade-off between the lowest examinable frequency and the complexity of the model. We choose to examine two octaves of the spectrum, from $f_{C3} = 130.81$Hz to $f_{C5} = 523.25$Hz.

We can reduce the complexity of the transform needed by reducing the resolution of the signal. Therefore, we first downsample the audio to 11025Hz, taking audio frames of 8192 audio samples (0.74s) in length.

This allows us to comfortably achieve quarter-tone frequency resolution at 130.81Hz. We use a hop size of 1024 samples to achieve around 10 estimates per second.

Each audio frame is multiplied by a Hamming window to reduce the 'edge' effects resulting from finite frame sizes [Harris, 1978] before the magnitude spectrum, $X(k)$, is calculated using the discrete Fourier transform (DFT). We then take the square root of the magnitude spectrum in order to reduce the amplitude difference between harmonic peaks.

Next, the fundamental frequencies of the notes in the two octaves we consider, starting from $f_{C3} = 130.81$Hz are calculated by:

$$f(n) = f_{C3} \cdot 2^{(n/12)} \tag{3.21}$$

for $n = 0, ..., P-1$ where $P = 24$, the number of notes in the two octaves we are examining.

For each fundamental frequency $f(n)$ we consider two harmonics – the fundamental frequency itself and the second harmonic which is an octave above the fundamental at a frequency of around $2 \times f(n)$. In real instruments, there tends to be some degree of inharmonicity and so the frequency of harmonics will deviate slightly from exact integer multiples of the fundamental.

Some approaches deal with this uncertainty by summing the energy in a number of frequency bins around where the harmonic is expected to be. A key part of our approach is that rather than summing all the energy in a spectral range (a 'bin mapping' technique) which can add unwanted noise to our chroma vector, we instead find the maximum value in each range. Should a harmonic tone be present in this range, the maximum value will likely identify the harmonic. If one is not present, we expect the maximum value to correspond to a relatively low noise value. Figure 3.12 compares our approach to that of a 'bin mapping' approach.

We first calculate energy values for each note in the two octaves we are considering:

$$S(n) = \sum_{h=1}^{H} \left( \max_{k_0 \leq k \leq k_1} X(k) \right) \left( \frac{1}{h} \right) \tag{3.22}$$

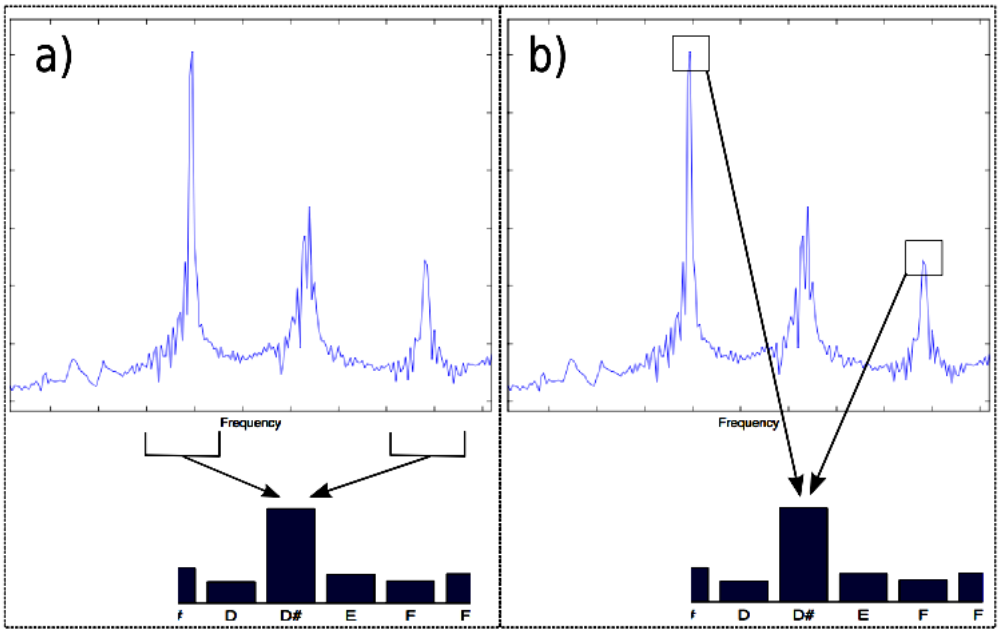



Figure 3.12: a) The 'bin mapping' technique: energy in spectral bins is mapped to a certain pitch class. b) Our technique: the maximum value in a given range is used as the amplitude value for the harmonic contributing to that pitch class.

where $n = 0, ..., P-1$ where $P = 24$, $h$ is the number of the harmonic, $H \geq 1$ and is the number harmonics under consideration for each note (we investigate values of $H = 2$ and 3), $k_0 = k'^{(n,h)} - (r \cdot h)$, $k_1 = k'^{(n,h)} + (r \cdot h)$, where the value $r$, for which we choose 2, is the number of bins to search either side of a frequency for a maximum value and

$$k'^{(n,h)} = \text{round}\left(\frac{f(n) \cdot h}{(f_s/L)}\right) \tag{3.23}$$

where $f_s$ is the sampling frequency and $L$ is the frame size, in our case 8192 samples. As $r$ is multiplied by the number of the harmonic $h$, a wider search is performed for higher harmonics than lower ones. Finally, the 12 values of the chroma vector, $C$, are calculated by summing bins of $S$ from the same pitch class in each octave:

$$C(n) = \sum_{d=0}^{D-1} S(n + (12 \cdot d)) \tag{3.24}$$

where $d$ is the number of the octave in question and $D = 2$, the number of octaves under consideration. An example of a chroma vector can be seen

in Figure 3.13. This chroma analysis technique was implemented in C++ as a Max/MSP external – '`chroma~`' – for real-time use.

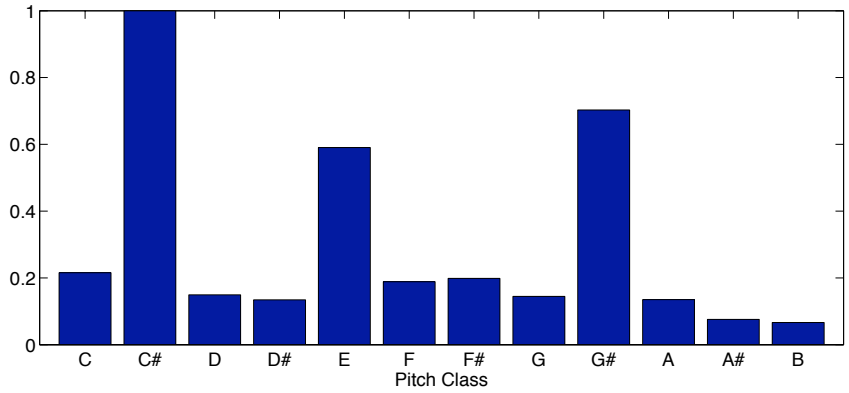



Figure 3.13: An example of a chroma vector from a C♯ minor chord. Strong energy can be seen for the C♯, E and G♯ pitch classes and a residual noise floor in the rest.

## 3.3 Chord Recognition

A chord is the simultaneous sounding of two or more musical notes. The interval relationships between these notes determine the type of chord. The process of chord recognition is one of assigning a label to a section of audio. In this section we build upon our chroma vector analysis technique presented in section 3.2 by presenting a technique to classify a given chroma vector into one of a number of chord types.

Cho et al. [2010] have identified that offline (i.e. non real-time) chord recognition techniques can use some pre-filtering of the chroma vectors over time to dampen the effect of transients and noise. Furthermore, post-filtering of annotated chord labels can also be used to deal with spurious chord labels that only last for a short time. Such processing is not possible in a real-time context as we must attach a chord label to an audio frame with no information on future chroma vectors or chord labels. As a result, the approach presented here focuses on developing a chord recognition technique that can accurately identify chords from single frames without using pre and post-filtering techniques.

A shortfall of some previous approaches to chord recognition is the limited number of chords classified. Some systems [Bello and Pickens, 2005; Papadopoulos and Peeters, 2007] only consider major and minor triads, leading to problems when certain chords are encountered. For example, a C Major 7 chord (the notes C, E, G and B) contains both a C Major triad (C, E and G) and an E minor triad (E, G and B) and so a model only modelling major and minor triads would be too simplistic in this case.

We present a technique capable of classifying chroma vectors as one of 108 chords – specifically the 12 variations of major, minor, diminished, augmented, suspended 2nd, suspended 4th, major 7th, minor 7th and dominant 7th chords. These are more chords than have been considered by the majority of approaches in the literature. We believe that this is necessary if chord recognition on real-world signals is to be achieved.

### 3.3.1 Chroma Vector Classification

Following much of the literature, we approach chord recognition as a problem of classifying chroma vectors into a number of classes. Many existing approaches use template matching techniques to compare how similar a chroma vector is to a set of chord profiles. These chord profiles are usually in the form of 'bit masks'. A bit mask is a 12 x 1 vector containing a 1 where a note is present and a 0 elsewhere, with the bits representing the presence of notes from C to B in semi-tone steps. A C Major chord would be represented as [1,0,0,0,1,0,0,1,0,0,0,0].

Most existing template matching approaches either i) choose the bit mask with the minimum Euclidean distance to the chroma vector [Fujishima, 1999] or ii) find the bit mask that maximises the dot product with the chroma vector [Fujishima, 1999; Cremer and Derboven, 2004; Harte and Sandler, 2005]. For the latter case a weighting can be used to distinguish between chords containing different numbers of notes.

When using the above approaches with bit masks some problems can be encountered. A bit mask representation implies that the energy in a chroma vector bin for a note that is present will be both close to 1 and

similar in value to that of other sounded notes. In reality, the energy in bins for sounded notes is variable - as can be seen in the example chroma vector in Figure 3.13. Furthermore, the relative amplitude of notes can vary, making the pre-calculation of more 'realistic' chord profiles problematic.

When using bit masks, an approach based upon the Euclidean distance will yield best results for a chroma vector that has values of exactly 1 for all notes present and 0 otherwise – but our experience has shown that this is unlikely to be the case. Using a dot product we have a problem when classifying between chords that contain different numbers of notes as chords containing more notes will inherently produce larger values. Others using such techniques [Fujishima, 1999] have employed a weighting to differentiate between chords that contain three notes and those that contain four or more. As the energy of notes represented in the chroma is variable, the setting of such a weighting is difficult to decide on. Based upon this we wish to develop a technique that avoids the problem of the variable amplitude of notes in chords.

### 3.3.2 Minimising Residual Energy

In order to solve this problem, we classify chords by masking out the notes that are hypothesised to be in the chord by each bit mask. We then choose the template that minimises the residual energy. We achieve this by finding the minimal dot product between the chroma vector and a 'complimentary' bit mask:

$$\delta_i = \frac{\sqrt{\sum_{n=0}^{P-1} \bar{T}_i(n)(C(n))^2}}{P - N_i} \tag{3.25}$$

where $C$ is the chroma vector, $\bar{T}_i(n) = 1 - T_i(n)$ where $T_i$ is the $i$th bit mask, $N_i$ is the number of notes in the $i$th bit mask and $P = 12$, the number of notes in an octave. We divide by $(P - N_i)$ in order to prevent chords with fewer notes having a natural advantage over others. This is in effect a 'weighting' between chords – however we are setting a weighting based upon the energy of the noise floor which we hypothesise will be more consistent than weighting between the energy of sounded

notes. Our reasoning for this is that noise, by its very nature, should not appear disproportionately within certain pitch classes and so pitch classes where there is not a sounded note should have similar levels of energy. We choose the chord that minimises $\delta_i$.

### 3.3.3 Chromagram-Unresolvable Chords

Certain chords, when represented using a chromagram, are indistinguishable from other chords as they contain exactly the same notes. From the set of chords that we are attempting to classify, this happens between some suspended 2nd and suspended 4th chords and between augmented chords. When represented via a chroma vector, our algorithm has no way of distinguishing between these chords and so we accept as correct the equivalent chord labels. For example, we accept as equivalent C augmented 5th, E augmented 5th and G♯ augmented 5th chords as they all contain the same three notes (C, E and G♯). An example within suspended chords is that Csus2 contains the same notes as Gsus4 (C, D and G).

Each augmented chord has 2 possible equivalents (resulting in 4 augmented chords rather than 12) and each suspended 2nd chord has an equivalent suspended 4th chord and vice versa (12 sus2/sus4 chords rather than 24). This means that 20 of the chords we are classifying are equivalent to others. This reduces the number of unique chords we are considering slightly from 108 to 88.

### 3.3.4 Evaluation

We evaluated our chord recognition technique through assessing its ability to classify individual audio frames from recordings of guitars. We created a test set of real world examples of 180 chords played on two different guitars and extracted 4 audio frames randomly from the recording of each chord. Each frame was 8192 samples in size and the audio was at a sampling frequency of 11025Hz. The result was 1440 audio frames with accompanying labels totalling over 17 minutes of audio. For each chord type (major, minor, etc.) there was at least a whole octave in the data set, with some types having 2 octaves. This evaluation assesses the ability

**for H = 2**

| Chord Type | R(%) | RQ(%) | RQI(%) |
|---|---|---|---|
| Major/Minor | 100 | 97.1 | 86.2 |
| Diminished | 100 | 100 | 100 |
| Augmented | 73.6 | 73.6 | 73.6 |
| Sus2 / Sus4 | 100 | 99.2 | 97.4 |
| Major 7 | 99.3 | 99.3 | 99.3 |
| Minor 7 | 94.4 | 94.4 | 94.4 |
| Dominant 7 | 100 | 100 | 100 |
| **Total (Over Examples)** | **96.7** | **95.8** | **92.4** |

**for H = 3**

| Chord Type | R(%) | RQ(%) | RQI(%) |
|---|---|---|---|
| Major/Minor | 78.6 | 77.1 | 76.6 |
| Diminished | 95.8 | 95.8 | 95.8 |
| Augmented | 88.9 | 88.9 | 88.9 |
| Sus2 / Sus4 | 100 | 99.7 | 98.7 |
| Major 7 | 99.3 | 99.3 | 99.3 |
| Minor 7 | 98.6 | 98.6 | 98.6 |
| Dominant 7 | 100 | 95.8 | 95.8 |
| **Total (Over Examples)** | **92.7** | **91.8** | **91.4** |

Table 3.3: The results of evaluating our chord recognition technique on the database. We present results for two different values of H, the number of harmonics in the chromagram calculation technique. Results are given for correct root note (R), correct root note and chord quality (RQ) and correct root note, quality and other intervals (RQI).

of our technique to identify chords in single polyphonic instruments and so our evaluation and results will differ from more generic evaluations on large databases of commercial recordings such as those who have evaluated on the collected Beatles recordings [Mauch and Dixon, 2008; Oudre et al., 2009].

We evaluated our model using values of H – the number of harmonics to consider for each note – of 2 and 3. The label of each audio frame consists of the *root note* of the chord, the *quality* (major, minor, diminished, etc.)

and any other *intervals* present in the chord (e.g. minor 7th). We recorded the performance of our technique at all three levels.

The results of the evaluation can be seen in Table 3.3. As can be seen from the results, our technique is able to classify most chord types with high accuracy. While some chords types, such as augmented chords, show better performance with $H = 3$, the technique performs most reliably across chord types when $H = 2$. In particular, for $H = 3$ many more major and minor chords are misclassified than for $H = 2$. Performance is also higher overall for 2 harmonics and so we take it to be the best choice for our model, given the dataset.

In the table of results for which $H = 2$, the score for major and minor chords is lower when considering all intervals compared to the other two levels as many major and minor chords were incorrectly classified as major 7th and minor 7th chords. The reason for this is that the third harmonic of the major or minor 3rd note in the chord causes energy to be brought into the chroma vector in the pitch class that is necessary to make a major or minor chord a 7th chord. For similar reasons the incorrectly classified augmented chords were all misclassified as major chords.

#### Comparison To Other Techniques

We also compared our technique to the work of others. We compared against two other chroma calculation techniques – the technique used by Bello and Pickens [2005] (an adaptation of the constant-Q based technique presented by Harte and Sandler [2005]) which we refer to as CQ, and another technique largely related to the techniques used in [Fujishima, 1999; Sheh and Ellis, 2003] where spectral bins are mapped to chroma bins. We refer to this latter technique, based upon 'bin mapping', as BM.

We also compared our chord classification technique to two others. The first was a nearest neighbour classifier (NN) and a weighted sum (WS) technique. We use the shorthand SA (from Stark chromA) for our chroma calculation technique and SD (from Stark chorD) for our chord classification technique.

The results of this comparison can be seen in Table 3.4. Results are

| Chord Classifier | Chroma | R(%) | RQ(%) | RQI(%) |
|---|---|---|---|---|
| WS | BM | 79.5 | 74.5 | 68.4 |
| | CQ | 73.9 | 68.8 | 64.7 |
| | SA | 87.0 | 83.8 | 76.5 |
| NN | BM | 92.5 | 88.8 | 85.4 |
| | CQ | 81.2 | 77.6 | 73.3 |
| | SA | 96.8 | 95.8 | 94.6 |
| **SD** | BM | 92.4 | 89.7 | 80.1 |
| | CQ | 84.9 | 80.1 | 67.2 |
| | **SA** | **96.7** | **95.8** | **92.4** |

Table 3.4: A comparison of the different combinations of three chroma calculation techniques and three chord classification techniques. The chroma analysis techniques are a 'bin mapping' technique (BM), a constant-Q based approach (CQ) and our approach (SA). The chord classification techniques are a weighted sum approach (WS), a nearest neighbour algorithm (NN) and our approach (SD). We have shown the results for three levels - the correct root note (R), the correct root note and chord quality (RQ) and the correct root note, chord quality and other intervals (RQI).

given for just the correct root note, the correct root note and chord quality and for the correct root note, chord quality and other intervals. The best combination at all three levels is of our chroma analysis technique (SA) combined with the nearest neighbour classifier (NN). Our chord classification technique (SD) combined with our chroma analysis technique (SA) is slightly behind this best score but comparable to it at all three levels. Our chroma analysis (SA) technique yields the strongest result for each classifier.

#### Efficiency

We also analysed the efficiency of both the three chroma analysis techniques and the three chord classification techniques. We analysed how long it took to process all the examples in the database used for the evaluation. All techniques were implemented in Matlab. The results can be seen in Table 3.5. As can be seen, our chroma analysis technique is faster than the other two approaches, calculating chroma vectors for all 1440

| Chroma Technique | Time Taken For Evaluation |
|---|---|
| BM | 12.9 seconds |
| CQ | 15.1 seconds |
| **SA** | **6.5 seconds** |

| Chord Classifier | Time Taken For Evaluation |
|---|---|
| WS | 14.2 seconds |
| NN | 8.3 seconds |
| **SD** | **5.1 seconds** |

Table 3.5: The time taken for the chroma analysis techniques (top) and the chord classifiers (bottom) to complete the evaluation of all 1440 examples. The chroma analysis techniques are our approach (SA), a constant-Q based approach (CQ) and a 'bin mapping' technique (BM). The chord classification techniques are our approach (SD), a nearest neighbour algorithm (NN) and a weighted sum approach (WS). All techniques were implemented in Matlab and tested on a 2 GHz Intel MacBook running OS X 10.5 with 1 GB of RAM.

examples in 6.5 seconds.

Our chord classification technique is also faster than the other two techniques, classifying chroma vectors for all 1440 examples in 5 seconds. This contrasts with the 8.3 seconds taken by the nearest neighbour (NN) technique for which we achieve comparable scores.

Our real-time chord recognition technique was implemented in C++ as a Max/MSP external called '`chorddetect~`'.

## 3.4 Beat-Synchronous Analysis

For music with a strong beat, it is desirable to create a representation whereby harmonic sequences can be represented in the same way regardless of changes in tempo. We also wish to calculate this representation in real-time. As a result, we present here three techniques for real-time beat-synchronous analysis by combining the real-time beat tracking system presented in section 3.1 with our harmonic analysis techniques – the

chroma analysis technique presented in section 3.2 and our chord recognition technique from section 3.3. In addition to these we also present a beat-synchronous spectrogram – where a single magnitude spectrum is calculated for each inter-beat interval.

### 3.4.1 A Model for Beat-Synchronous Analysis

In creating a beat-synchronous representation, we wish to calculate one feature representation (a symbol or vector) per inter-beat interval, the time between beats. We define $L$ to be the length of each inter-beat interval in audio samples. This is calculated using the time between beats in seconds, and the sampling frequency $f_s$, with $L = (\gamma_b - \gamma_{b-1}) \cdot f_s$.

Each inter-beat interval will contain a number of audio frames, $Q = \lfloor \frac{L}{N} \rfloor$, where $N$ is the length of each audio frame in audio samples. In order to obtain a beat-synchronous representation, we wish to process these audio frames so that we result in a single harmonic representation per inter-beat interval of length $L$.

To achieve this, time domain audio must be converted into the frequency domain via a spectral transform – and this resulting spectral transform used to calculate some form of harmonic representation. Depending upon whether the spectral transforms and the calculation of harmonic representations are carried out together on each audio frame, together on the larger inter-beat interval of length $L$ or separately, we can identify three methods for calculating beat-synchronous sequences in real-time. We now discuss each, and their merits, in turn.

#### Method 1

The first method accumulates all the audio from the $Q$ audio frames within an inter-beat interval into a larger buffer. A spectral transform (e.g. the FFT) is then performed on this larger buffer, followed by a transformation to some harmonic representation (such as a chroma vector). This process can be seen in the top row of Figure 3.14.

A first problem with this method is that the amount of audio that it is necessary to accumulate – and therefore the length of signal upon

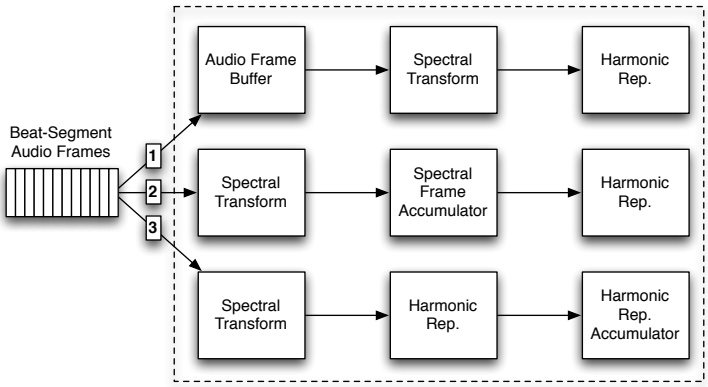



Figure 3.14: Three different methods for beat-synchronous harmonic analysis. 'Rep.' stands for 'Representation'.

which we will perform the spectral transform – varies from beat to beat according to the tempo.

Furthermore, by performing a spectral transform on a larger buffer of accumulated audio frames, method 1 is also more computationally demanding than performing a spectral transform on each individual frame. We can demonstrate this as follows.

Assuming the length of each audio frame $N$ is a power of 2, computing the Fast Fourier Transform (FFT) of a single longer segment of length $N$ requires more calculations than calculating $Q$ FFTs of length $N/Q$. Given that the complexity of the FFT is $O(N \log(N))$, this is demonstrated by:

$$O(N \log(N)) > O(Q \cdot \frac{N}{Q} \log(\frac{N}{Q})) \tag{3.26}$$

for $Q = 2^d$ and $1 \leq d < r$ for $N = 2^r$. This reduces to:

$$O(N \log(N)) > O(N \log(\frac{N}{Q})). \tag{3.27}$$

In addition to this, all processing for method 1 is carried out in a single step, rather than being distributed across time. A possible benefit of method 1, however, is that the larger spectral transform would allow greater frequency resolution for analysis purposes.

#### Method 2

The second method for calculating beat-synchonous analysis performs a spectral transform on each frame of length $N$ before accumulating the results of these spectral transforms to represent the entire inter-beat interval. Finally, a transformation to some harmonic representation is performed on the accumulated spectral frame. This approach (displayed in the middle row of Figure 3.14) distributes the computation of spectral transforms across smaller frames and, as was shown during the explanation of method 1, is more efficient than computing a single spectral transform on a larger buffer of multiple audio frames.

Another benefit is that by only computing a single harmonic representation, processor usage is minimised. However, it is possible that our harmonic analysis algorithm may benefit from the accumulation of multiple harmonic representations.

#### Method 3

The third method calculates a spectral transform and then a harmonic representation for each audio frame of length $N$. The harmonic representations are then accumulated to arrive at the single beat-synchronous harmonic feature vector. The difference between methods 2 and 3 is that method 2 uses temporal smoothing of the results of the spectral transforms while method 3 uses temporal smoothing of the harmonic representation.

It is also possible, if the harmonic analysis merely involves a summation of spectral bins, that methods 2 and 3 could produce identical results. However, we re-iterate that as method 2 only computes a single harmonic representation from an accumulated spectral frame, it is more efficient than method 3.

### 3.4.2 Frame Overlap

In order to calculate harmonic representations with sufficient frequency resolution for analysis purposes, it may be necessary to use a relatively large audio frame size – perhaps over 0.25 seconds long – which can result in having very few frames per beat (1 or 2). As a solution we can use a

small hop size (e.g. around 0.1 seconds) and have a number of overlapping frames. This does, however, present us with a problem as at beat locations we may start to include audio from the previous inter-beat interval – blurring the concept of the beat-synchronous approach. As a result, we suggest clearing the audio buffer at each beat after the analysis by replacing it with zeros so that no audio from the previous inter-beat interval is considered in the current inter-beat interval.

We now present techniques for calculating a beat-synchronous spectrogram, chromagram and chord sequence.

### 3.4.3 Beat-Synchronous Spectrogram

By producing a single magnitude spectrum for each inter-beat interval, we can calculate a beat-synchronous spectrogram. To achieve this, we calculate each spectral frame $f$ using the Fourier transform:

$$X_f(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \tag{3.28}$$

for $0 \leq k < N$, where $x(n)$ are the samples of the audio frame and $N$ is the frame size. Then we calculate the Fourier transform for the beat segment, $b$, by:

$$X_b(k) = \sum_{f=0}^{F-1} |X_f(k)| \tag{3.29}$$

for $0 \leq k < N$, where $F$ is the number of frames within the inter-beat interval $[\gamma_{b-1}, \gamma_b]$. For method 1, $F = 1$ and for methods 2 and 3 $F > 1$.

### 3.4.4 Beat-Synchronous Chromagram

We calculate a beat-synchronous chromagram, $\Phi_b(i)$, using the technique presented in section 3.2, as follows:

$$\Phi_b(i) = \sum_{h=0}^{H-1} \Phi_h(i) \tag{3.30}$$

where $i$ is the chroma bin index, $i = 0, 1, ..., I-1$ where $I = 12$ and $\Phi_h$ is the $h$th chromagram calculated from H spectral frames. For methods 1

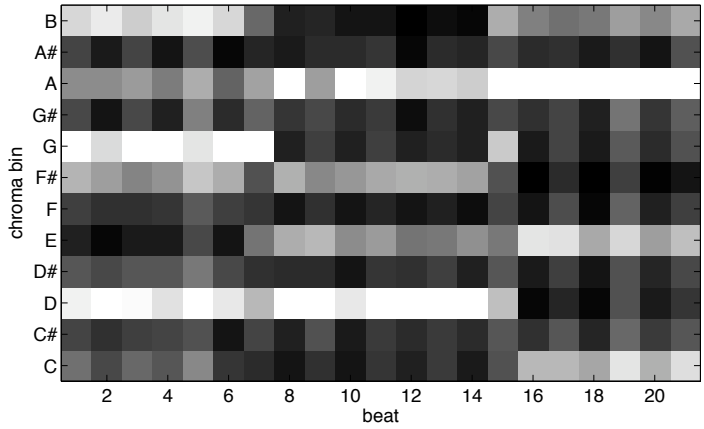



Figure 3.15: An beat-synchronous chroma vector sequence

and 2, $H = 1$, while $H > 1$ for method 3. An example beat-synchronous chroma vector sequence can be seen in Figure 3.15

### 3.4.5 Beat-Synchronous Chord Analysis

We implement a beat-synchronous chordal analysis by classifying the beat-synchronous chromagram presented in section 3.4.4 using the chord recognition technique presented in section 3.3.

## 3.5 Summary

In this chapter we have presented methods for extracting information from a musical signal in real-time. These information signals provide a fundamental layer upon which interactive applications can be built.

Specifically we have presented a real-time beat tracker that is computationally efficient compared to other approaches while showing robust performance. We have presented chroma analysis and chord recognition techniques that show comparable performance to other state of the art approaches while taking a fraction of the processing time. Finally, we have shown that we can combine beat tracking and harmonic analysis to provide real-time sequences of harmonic vectors segmented by the beat.

All the analysis techniques detailed in this chapter were implemented

in C++ as Max/MSP externals which are publicly available[4].

[4] http://www.eecs.qmul.ac.uk/~adams/software.html

# Chapter 4

# Investigating Methods For Improving Beat Tracking

We have presented, in Chapter 3, several algorithms for extracting harmonic and temporal information, in real-time, from musical performances. While none of our analysis algorithms are perfect, informal use of our algorithms in real-time indicates that errors in real-time beat tracking produce more noticeable and unpalatable errors. Furthermore, in the case of beat-synchronous analysis, the accurate segmentation of harmonic information by the beat relies upon accurate beat tracking. In this sense, beat tracking errors have the potential to propagate to other analysis modules.

As a result we turn in this chapter to a deeper study of beat tracking systems in order to investigate methods to improve beat tracking that can be applied to our real-time model.

Specifically, we describe the results of a modular evaluation of five state of the art beat tracking systems – comparing and contrasting their different input features and tracking models. While others have conducted evaluations of beat tracking models [McKinney et al., 2007], we consider their constituent parts separately, examining the effect of varying them. Based upon this, we draw some conclusions about aspects of the beat tracking systems such as their input features and parameterisation, returning to our real-time beat tracker to examine ways to improve performance based upon these conclusions. It should be noted that we are not presenting an implemented improvement to our model – rather we present a detailed

investigation into methods for doing so in future[1].

## 4.1 Modular Components

In this chapter we study five beat tracking models which between them use four different input features. Here we provide a brief summary of the nature of each modular component. For more detailed explanations, see the relevant publication (referenced in each description).

### 4.1.1 Input Features

The four input features used in the modular evaluation are all calculated from audio signals sampled at 44.1kHz. For completeness we include information on the frame sizes, hop sizes and resulting input feature resolutions used as detailed in the original publications.

#### Bandwise Accent Signals (BAS)

Introduced by Klapuri et al. [2006], Bandwise Accent Signals are calculated from 1024 sample frames with a 512 sample hop size. The Fourier transform of these frames is taken and used to calculate power envelopes at 36 sub-bands on a critical-band scale. Each sub-band is upsampled by a factor of two, smoothed and half-wave rectified. A weighted average of each band and its first order differential is taken. Finally, the 36 bands are summed to create a four channel input feature with a resolution of 5.8ms.

#### Spectral Flux (SFX)

Dixon [2006b] has introduced the Spectral Flux input feature, calculated from audio frames of 2048 samples with a 512 sample hop size – corresponding to a resolution of 11.6ms. The Fourier transform of each frame is taken, which is in turn used to calculated the magnitude spectrum. Each input feature sample is then calculated as the sum of the positive differences in magnitude for each frequency bin of the magnitude spectrum

[1]This chapter is based upon work completed collaboratively with Matthew E. P. Davies

compared to the corresponding bin in the previous frame[2].

#### Mel Auditory Feature (MAF)

The Mel Auditory Feature, introduced by Ellis [2007], is calculated by first re-sampling the input audio to 8kHz. The Fourier transform is taken from 256 sample frames with a 32 sample hop size, corresponding to a resolution of 4ms. The resulting frequency domain representations are converted to an approximate "auditory" representation using a Mel-frequency spaced 40 band grouping. The first order difference is taken and then the result is half wave rectified. Finally, the result is summed across frequency bands and then smoothed to create the final input feature.

#### Complex Spectral Difference (CSD)

The Complex Spectral Difference [Bello et al., 2004] is calculated from 1024 sample frames with a 512 sample hop size. This results in a resolution of 11.6ms. The input feature is large if there is a large change in magnitude or a deviation from expected phase values. Given that signals tend to be less stable during the onset of musical notes, we expect to see larger values at those locations.

### 4.1.2 Tracking Models

We now summarise the five tracking models used in the modular evaluation.

#### Klapuri et. al. (KL)

The Klapuri et al. [2006] tracking model takes as an input feature the Bandwise Accent Signals (BAS). This four-channel feature is passed through a bank of comb filter resonators. A probabalistic framework based upon a hidden Markov model is then used to track beats at three metrical levels. Specifically these are the *tatum*, the fastest metrical level, the

[2]We re-implemented the Spectral Flux input feature according to Dixon [2006b], using the stated 512 sample hop size rather than 441 in the paper as this was the resolution accepted by the software implementation of the corresponding tracking model.

*tactus*, the level at which humans are most likely to tap their foot and the *measure* level, indicating the first beat of each bar.

#### BeatRoot (BR)

Dixon [2006a] presented *BeatRoot*, which takes the Spectral Flux (SFX) feature as an input feature. Onset times are extracted through peak picking on the input feature before inter-onset interval (IOI) times of similar lengths are clustered together. Based upon the number of IOIs in each cluster and the integer relationships between the lengths of the IOIs in the clusters, a number of tempo hypotheses are calculated. Then, multiple beat tracking agents initialised with different tempo hypotheses and phases are passed through an evaluation function which assigns a rating for each agent based upon how evenly its predicted beat times are spread, how many of them coincide with onsets and the salience of those onsets. The beat times of the agent with the highest score are chosen as the output of the algorithm.

#### Ellis (EL)

The Ellis [2007] tracking model attempts to find beat locations from the Mel Auditory Feature (MAF). Using a method based upon dynamic programming, a recursive function is calculated which for each sample point indicates the best possible score for all beat sequences ending at that point. After this function has been calculated, beat times are recovered by performing a 'backtrace' step.

#### Davies and Plumbley (DP)

The Davies and Plumbley model is an adaptation of the Ellis [2007]. Based on the Complex Spectral Difference (CSD) input feature, it retains the recursive feature and backtrace of the Ellis model but replaces the tempo estimation phase with one similar to that of a previous Davies and Plumbley approach [Davies and Plumbley, 2007]. The algorithm is available as

a Sonic Visualiser plug-in[3].

**BTrack (RT)**

The *BTrack* model is our real-time approach, presented in Chapter 3, with no count in or tempo initialisation.

### 4.1.3 Summary of Modular Components

In Table 4.1 we present a summary of the four input features, detailing their resolution, number of channels and the tracking models that were designed to use them.

| Input Feature | Resolution | # Channels | Tracking Model(s) |
|---|---|---|---|
| BAS | 5.8ms | 4 | KL |
| SFX | 11.6ms | 1 | BR |
| MAF | 4ms | 1 | EL |
| CSD | 11.6ms | 1 | DP & RT |

Table 4.1: A summary of the four input features, their resolution, number of channels and the tracking models that were designed to use them.

## 4.2 A Modular Evaluation: Method

In order to investigate ways to improve beat tracking, we now present an evaluation that considers their input features and tracking models as separate modular components. By varying the input features for a given tracking model (and conversely, the tracking model for a given input feature) our objective is to identify ways to improve beat tracking, and to apply these to our real-time beat tracker.

Splitting five beat tracking models into their input feature and tracking model, we conduct a modular evaluation of the resulting four input features and five tracking models.

Reference to Table 4.1 shows that there is variation in the temporal resolution and number of channels of the input features. The resolution

[3]http://isophonics.net/QMVampPlugins

and number of channels expected by each tracking model also varies. This makes using an arbitrary input feature with an arbitrary tracking model non-trivial. To overcome these problems, and thereby allow the evaluation of all 20 combinations of input features and tracking models, we used the following four steps.

Given an input feature and a tracking model: 1) The input feature is extracted from the audio file. 2) If the resolution of the input feature does not match the resolution expected by the tracking model, then the input feature is re-sampled accordingly. 3). If the number of channels of the input feature does not match that expected by the tracking model (the BAS input feature has four channels while the others have just one), then the number of channels is adjusted as follows. If there is only one channel and the tracking model expects $K$ channels, then the single channel is replicated to produce $K$ channels. If the input feature contains multiple channels and the tracking model only expects a single channel, then the $K$ channels are summed together. 4) Finally, the tracking model is used to find beats from the input feature.

### 4.2.1 Databases

We evaluated each input feature-tracking model pair on two databases. Rather than combine them into a single larger database, the use of two databases allowed us to identify trends repeated on both databases, despite their different compositions, and so draw more general conclusions about the relative performance of different models. Furthermore, the databases themselves will be identifiable to those familiar with the literature as they have been used in several other publications [Hainsworth, 2004; Klapuri et al., 2006; Davies and Plumbley, 2007; Degara et al., 2011]. It should be noted that the separate databases were used for comparison, rather than N-fold cross validation.

**Database 1: The Hainsworth Database**

The *Hainsworth* database [Hainsworth, 2004] consists of 222 excepts of approximately 60 seconds from commercial audio files. The genre breakdown of this database is as follows: Rock and Pop (68), Dance (40), Jazz (40), Classical (30), Folk (22) and Choral (22). A full list of the tracks in the database, with accompanying genre labels, is available in the appendix of Stephen Hainsworth's PhD thesis [Hainsworth, 2004].

**Database 2: The Klapuri Database**

The *Klapuri* database, used to to evaluate beat tracking techniques in Klapuri et al. [2006], consists of 474 approximately 60 second excerpts of commercial audio files as with Database 1. The genre breakdown of this database is: Electronic/Dance (66), Rock/Pop (124), Jazz/Blues (94), World/Folk (15), Classical (84), Hip Hop/Rap (37) and Soul/RnB/Funk (54). A full list of the tracks in the database, with accompanying genre labels, is available online[4].

Both databases contain commercial audio files and as a result are not publicly available.

### 4.2.2 Evaluation Measure

We choose as the evaluation measure the LML measure described in Chapter 3 (see page 71 for details) as it takes into account both localisation and evaluates beat sequences at multiple metrical levels.

## 4.3 A Modular Evaluation: Results

We ran all 20 combinations of input features and tracking models on both databases and evaluated the resulting beat times using the LML evaluation measure. The results can be seen in Table 4.2.

[4] `http://www.cs.tut.fi/~klap/iiro/meter/database.html` (Accessed 14/08/2011)

**Database 1**

| **Tracking** | **Input Feature** | | | |
|---|---|---|---|---|
| **Model** | BAS | SFX | MAF | CSD |
| KL | **67.2** | 53.7 | 61.7 | 57.1 |
| BR | 62.8 | 63.0 | 52.7 | 63.7 |
| EL | 64.2 | 57.6 | 61.3 | 60.8 |
| DP | 63.4 | 64.7 | 55.1 | 66.6 |
| RT | 56.7 | 64.2 | 47.7 | 63.7 |

**Database 2**

| **Tracking** | **Input Feature** | | | |
|---|---|---|---|---|
| **Model** | BAS | SFX | MAF | CSD |
| KL | **70.3** | 53.0 | 63.3 | 56.4 |
| BR | 67.1 | 65.2 | 54.1 | 66.5 |
| EL | 65.8 | 56.8 | 63.7 | 60.1 |
| DP | 67.2 | 63.7 | 58.2 | 67.2 |
| RT | 60.5 | 65.9 | 50.8 | 66.9 |

Table 4.2: Results of the modular evaluation of each input feature on each tracking model on the two databases (%). The original combinations of input features and tracking models are underlined. The best score on each database is in bold text.

The distribution of results appears to be similar for both databases. The best performing combination on both databases was for the Bandwise Accent Signals input feature and the Klapuri tracking model {BAS, KL} scoring 67.2% on Database 1 and 70.3% on Database 2.

Other input feature and tracking model combinations also perform well. These include the {CSD, DP} combination, the {CSD, BR} and {SFX, BR} combinations (consistent with [Gouyon et al., 2007]) and {BAS, EL}. Our approach, the {CSD, RT} combination, scores comparably well given that it is the only causal model considered.

As one may expect, tracking models generally achieve some of their higher scores using the input feature that they were designed to use (scores for these combinations are underlined in Table 4.2). However this is not

**Database 1**

| **Input Feature** | BAS | CSD | SFX | MAF |
|---|---|---|---|---|
| Best Score | 67.2 | 66.6 | 64.7 | 61.7 |
| Mean Score ($\mu$) | 62.9 | 62.4 | 60.7 | 55.7 |
| Std. Dev. ($\sigma$) | 3.8 | 3.6 | 4.8 | 5.9 |

**Database 2**

| **Input Feature** | BAS | CSD | SFX | MAF |
|---|---|---|---|---|
| Best Score | 70.3 | 67.2 | 65.9 | 63.7 |
| Mean Score ($\mu$) | 66.1 | 63.4 | 60.9 | 58.0 |
| Std. Dev. ($\sigma$) | 3.6 | 4.9 | 5.7 | 5.7 |

Table 4.3: The best scores, mean scores and standard deviation of each input feature across all tracking models (%)

always the case – for example the Ellis (EL) tracking model performs best using the Bandwise Accent Signals (BAS), rather than its own feature. Some other tracking models score comparably with other features to how they perform with their own feature such as {SFX, RT} and {BAS, DP}.

### 4.3.1 Input Feature Results

In order to draw some more general conclusions about input features we consider their performance across all tracking models. We present in Table 4.3 for each input feature: the best score using any given tracking model (labelled 'best score'), the mean score across all five tracking models and the standard deviation of those scores.

We can see from the table that the Bandwise Accent Signals (BAS) feature has both a higher mean score and a higher best score than other features, on both databases. The higher mean score is accompanied by a comparably low standard deviation across tracking models indicating that the tracking models that use BAS as an input feature generally perform well. We conclude from this that BAS is the strongest of the four input features.

We find also that the Mel Auditory Feature (MAF) has the lowest

best score and the lowest mean score on both databases. From this we can conclude that it is the weakest of the four features.

Of the remaining two input features, the Complex Spectral Difference (CSD) outperforms the Spectral Flux (SFX) on both best scores and mean scores for both databases.

Overall, it appears that some input features are more informative than others with respect to beat tracking. We can see that the BAS feature is the strongest, implying that it better represents the location of onsets in music signals than others such as MAF.

#### Effect of the Input Feature On The Results

The results indicate that some input features such as the Bandwise Accent Signals (BAS) result in higher beat tracking performance more generally, and others such as the Mel Auditory Feature result in lower performance. We now investigate the extent of the effect of the general 'quality' of an input feature – as given by these results – on beat tracking performance.

We take, for each input feature, the mean score across tracking models to be a measure of the 'quality' of that feature. We then calculate the correlation between the score achieved for each input feature-tracking model pair on database 1 and the mean score for the input feature used, calculated from the results of database 2. We also calculate the reverse situation with databases 1 and 2. The results can be seen in Figure 4.1.

In both examples there is a moderate correlation (correlation coefficients of 0.52 and 0.51) between the quality of the input feature (extracted from one database) and the result of using some tracking model with that feature (on the other database). From this we can conclude that – to an extent – a good input feature is indicative of good beat tracking performance and so the quality of the input feature is a significant part of the beat tracking process.

The correlation is moderate rather than strong as there is another variable factor, the tracking model.

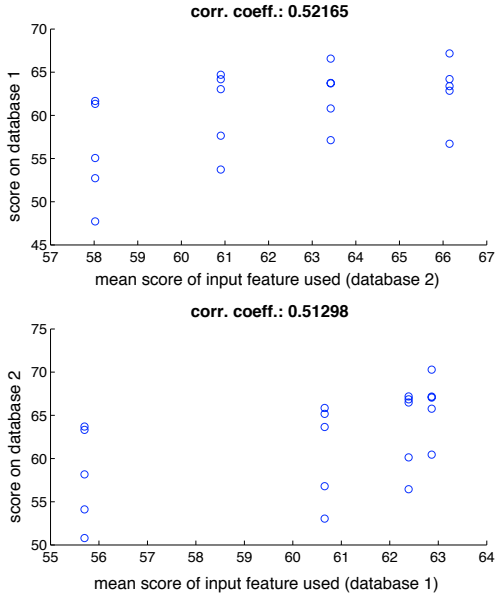



Figure 4.1: The correlation between the quality of an input feature, extracted as the mean score from one database, and the performance of various tracking models using that feature on the other database.

### 4.3.2 Tracking Model Results

We present in Table 4.4, for each tracking model, the best single score with any given input feature (labelled 'best score') and the mean score and standard deviation across all four input features.

These results tell a different story to those of the input features. The Klapuri et al. (KL) tracking model achieves the highest score for a single input feature on both databases – however it has one of the lowest mean scores across all input features on both databases and a comparatively high standard deviation. Furthermore, Table 4.2 shows that it achieved the highest single scores using its own feature (BAS). This indicates that the Klapuri et. al tracking model has a high degree of *feature dependence*.

Conversely, the Davies and Plumbley (DP) model has a strong best single score on both databases and a higher mean score than other tracking models. This indicates that the DP model has a degree of *feature independence*.

The other three tracking models show varying degrees of dependence

**Database 1**

| **Tracking Model** | KL | BR | EL | DP | RT |
|---|---|---|---|---|---|
| Best Score | 67.2 | 63.7 | 64.2 | 66.6 | 64.2 |
| Mean Score ($\mu$) | 59.9 | 60.6 | 61.0 | 62.4 | 58.1 |
| Std. Dev. ($\sigma$) | 5.8 | 5.3 | 2.7 | 5.1 | 7.7 |

**Database 2**

| **Tracking Model** | KL | BR | EL | DP | RT |
|---|---|---|---|---|---|
| Best Score | 70.3 | 67.1 | 65.8 | 67.2 | 66.9 |
| Mean Score ($\mu$) | 60.8 | 63.2 | 61.6 | 64.0 | 61.0 |
| Std. Dev. ($\sigma$) | 7.6 | 6.1 | 4.0 | 4.3 | 7.4 |

Table 4.4: The best scores, mean scores and standard deviation of each tracking model across all input features (%)

on their input features in a similar way to the KL and DP models.

A consequence of these results is that the mean score across input features is not a meaningful measure of the 'quality' of a tracking model overall. Therefore we do not attempt to analyse the correlation between the 'quality' of a tracking model - determined by the mean score on one database - with the results of the tracking models on the other database.

We can conclude from our results that, for some tracking models, the choice of input feature is very important to the level of performance and that tracking models in general have varying degrees of dependence on their original input features.

### 4.3.3 Genre Specific Breakdown

In order to explore the response of input features and tracking models to different types of signals, we proceed with a genre-specific analysis of both components.

#### Input Features

Figures 4.2(a) and 4.2(b) show the scores for each input feature on different genres for each database. The genre breakdown of the two databases is

not identical but we can draw some conclusions by comparing the two.

On database 1, the "difficult" genres (i.e. performance is generally low) appear to be Choral, Classical, Folk and Jazz. Better performance is achieved on Dance and Rock. For database 2, Classical, Jazz/Blues and World/Folk appeared to be more challenging with better performance on the Electronic/Dance, Rock/Pop, Hip Hop/Rap and Soul/RnB/Funk genres.

In terms of the relative performance of different input features, the Classical genre on both databases shows a clear pattern. The Bandwise Accent Signals (BAS) and Mel Auditory Feature (MAF) clearly outperform the other two features, on both mean scores and best scores.

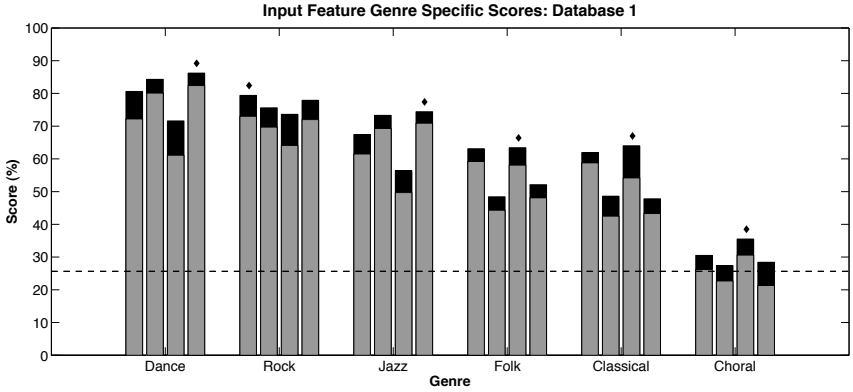
The Rock genre from database 1 and the Rock/Pop genre from database 2 show similar distributions. All input features score reasonably well with the Bandwise Accent Signals (BAS) and Complex Spectral Difference (CSD) showing the best performance for both mean and best scores. Similar relative scores between input features are found for the Hip Hop/Rap and Soul/RnB/Funk genres on database 2, indicating similar instrumentation and tempo dynamics to the Rock/Pop genre.

For the Jazz (database 1) and Jazz/Blues (database 2) genres we can see that the Spectral Flux (SFX) and Complex Spectral Difference (CSD) features outperform the Mel Auditory Feature (MAF) for both databases. However, we can see that on database 1 the Bandwise Accent Signals (BAS) feature is weaker than SFX and CSD while for database 2 BAS is the strongest feature overall. This could be due to the fact that the Jazz/Blues genre contains files from the blues genre while the equivalent category in database 1 does not.

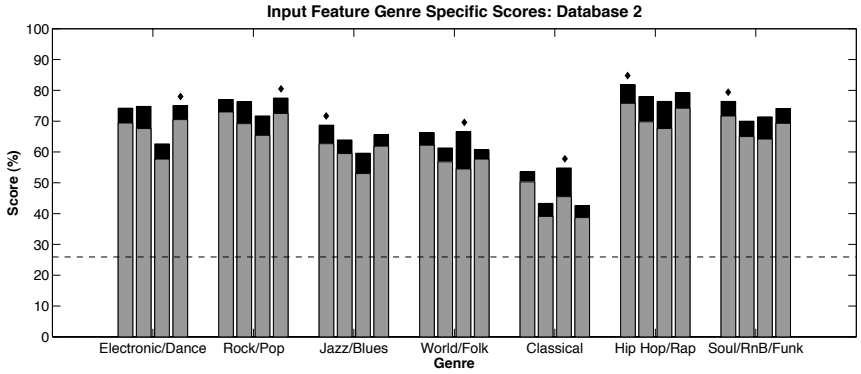
For the Dance (database 1) and Electronic/Dance (database 2) we see a similar picture to the Jazz and Jazz/Blues genres - there is a similar distribution although the scores overall are generally better for the Dance and Electronic/Dance genres.

The Folk genre on database 1 shows a similar distribution to the Classical genre. However, we do not see a similar pattern for the World/Folk genre for database 2.

Finally, on the Choral genre from database 1, all input features show




(a) Genre Specific Scores for Input Features on Database 1




(b) Genre Specific Scores for Input Features on Database 2

Figure 4.2: Genre specific breakdown for input features on database 1 (4.2(a)) and database 2 (4.2(b)). The grey bars show the mean score across trackers, the black bars show the best score for any single tracker. The four bars for each genre represent different input features, from left to right: Bandwise Accent Signals (BAS), Spectral Flux (SFX), Mel Auditory Feature (MAF) and Complex Spectral Difference (CSD). The diamond indicates the best input feature for the given genre and the dotted line is the baseline 'random' score for a beat tracker that simply outputs beats at 120bpm regardless of the signal.

poor performance. The dotted line in Figure 4.2(a) shows the score for a tracking model that outputs beats at 120bpm regardless of the characteristics of the input signal. For all input features, performance is only marginally above this line for the best scores and below it in some cases for the mean scores.

The wider point to be taken from this analysis is that we cannot assume that one input feature will be more appropriate for all genres of music.

We can see clearly that an input feature that is strong on one genre (e.g. the Mel Auditory Feature (MAF) on Classical music) can be weak on others (e.g. Dance or Electronic/Dance). The increased use of percussion in some genres (such as Rock and Pop music) or the slower, more tonal changes present in others (such as Choral music) mean that signals from each genre are best represented by different types of input features. We conclude that this is evidence against the use of a universal input feature for all signal types.

**Tracking Models**

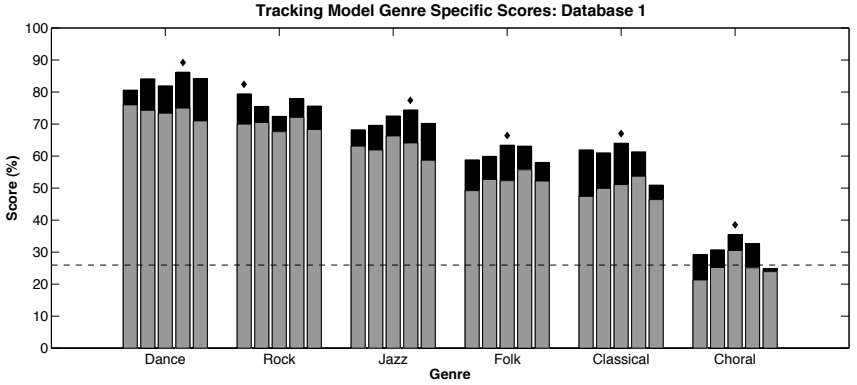
In Figures 4.3(a) and 4.3(b) we display, for each tracking model, the scores for the different genres of the two databases.

By inspection we can see that, when comparing the results to the input feature results shown in Figures 4.2(a) and 4.2(b), in general there is much less difference in performance between tracking models for a given genre than there is when using different input features. Specifically, for a given genre the average standard deviation in the best scores for different input features is 6.2% for database 1 and 3.9% for database 2. This compares with the tracking models where we have average standard deviations of 3.1% for database 1 and 2.9% for database 2. It seems that there is much less variation in performance for a given genre among tracking models than when using different input features.

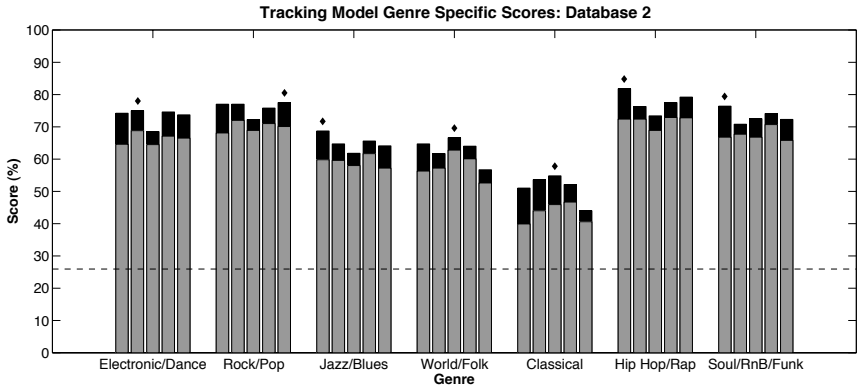
A consequence of this result is that it is more difficult to make generalisations about the relative performance of tracking models on different genres. We can, however, identify some clear trends.

We can see that our real-time tracking model (RT – the right most bar in each bar graph) performs comparably to the non-causal models on genres characterised by a strong beat and lower tempo variation, such as Dance and Rock and Pop music. However, when we consider genres where tempo variation is more common and percussive instruments less frequently used – such as Classical, Choral or World/Folk music – then the real-time model clearly performs worse than the non-causal models.

The implication of this result is that the advantages of a non-causal




(a) Tracking Model Specific Scores for Input Features on Database 1




(b) Tracking Model Specific Scores for Input Features on Database 2

Figure 4.3: Genre specific breakdown for tracking models on database 1 (4.3(a)) and database 2 (4.3(b)). The grey bars show the mean score across input features, the black bars show the best score on any single input feature. The five bars for each genre represent different tracking models, from left to right: Klapuri et. al (KL), BeatRoot (BR), Ellis (EL), Davies and Plumbley (DP) and our real-time model (RT). The diamond indicates the best tracking model for the given genre and the dotted line shows the baseline 'random' score for a beat tracker that simply outputs beats at 120bpm regardless of the signal.

approach mainly exercise themselves in the presence of tempo variation and minimal percussion and that for music with a strong beat there is little difference between causal and non-causal approaches.

**Database 1**

| **Tracking Model** | KL | BR | EL | DP | RT | $\mu$ |
|---|---|---|---|---|---|---|
| Best Input Feature | 67.2 | 63.7 | 64.2 | 66.6 | 64.2 | **65.2** |
| Oracle Input Feature | 73.7 | 73.0 | 75.0 | 75.6 | 71.4 | **73.7** |
| Performance Difference | +6.5 | +7.3 | +10.8 | +9.0 | +7.2 | **+8.6** |

**Database 2**

| **Tracking Model** | KL | BR | EL | DP | RT | $\mu$ |
|---|---|---|---|---|---|---|
| Best Input Feature | 70.3 | 67.1 | 65.8 | 67.2 | 66.9 | **67.4** |
| Oracle Input Feature | 74.4 | 75.4 | 75.2 | 75.8 | 72.9 | **74.7** |
| Performance Difference | +4.1 | +8.4 | +9.4 | +8.6 | +6.0 | **+7.3** |

Table 4.5: Results, for each tracking model, comparing the best performance using a single input feature for the whole database, and an oracle input feature (%)

## 4.4 The Effect of Improved or Varied Input Features

In this section we wish to establish two things. Firstly, to what extent can beat tracking performance be improved by using existing tracking models, improving the input feature alone? Secondly, we investigate the effect of diversifying the input features used – if we better choose our input features for a given signal, how much can we improve performance?

In order to answer these questions, we investigate, for each tracking model, the difference in performance between with the best single input feature and an *oracle input feature*. This oracle input feature is a selection between all four input features such that, for each file, the input feature that results in the best performance according to our evaluation measure is used. The results can be seen in Table 4.5.

On both databases we can see that there is an average improvement in performance of over 7% when using the oracle input feature rather than any single input feature. From this we can conclude two things. Firstly, the results show that existing tracking models could show considerably better performance if given a more appropriate or informative input feature. Secondly, that the use of different input features for different types

of signal has the potential to improve performance. We now discuss these in detail.

**Improving Beat Tracking By Improving the Input Feature**

As tracking models appear to be able to show improved performance when given a more appropriate input feature, research into an improved input feature seems fertile ground for improving beat tracking. Based upon this result we decided to investigate how well each tracking model was able to perform given an artificial input feature that is unequivocal about the location of the beats in the signal. This hypothetically 'optimal' input feature, which we shall call the *ideal delta feature*, is simply a series of delta functions at the locations of beats as specified in the beat annotations. The results can be seen in Table 4.6.

**Database 1**

| **Tracking Model** | KL | BR | EL | DP | RT |
|---|---|---|---|---|---|
| Best Score | 67.2 | 63.7 | 64.2 | 66.6 | 64.2 |
| Ideal Delta Score | 82.1 | 95.9 | 94.8 | 93.3 | 77.3 |

**Database 2**

| **Tracking Model** | KL | BR | EL | DP | RT |
|---|---|---|---|---|---|
| Best Score | 70.3 | 67.1 | 65.8 | 67.2 | 66.9 |
| Ideal Delta Score | 81.2 | 96.8 | 97.0 | 92.9 | 75.3 |

Table 4.6: Results, for each tracking model, comparing the best performance on each database, and the performance using a 'perfect' feature derived from the annotations (%)

The results show that several trackers are able to score in excess of 90% on both databases, with significant improvements from the others. While the feature given to the trackers is entirely hypothetical and unlikely to be derivable from an audio file, these results re-inforce our earlier analysis that significant improvement in performance is possible using existing tracking models and improving the input feature.

Furthermore, it appears that the tracking models are capable of tracking the sequences of beat annotations. This implies that they are not

under-performing in terms of an untrackable sequence of beats – caused by characteristic tempo variation (e.g. rubato) – rather it appears that the beats cannot be distinguished well enough from existing input features.

We stress that this 'ideal delta feature' is hypothetical and that the likelihood of extracting such a feature from a real audio signal is small. Therefore, the different performances in Table 4.6 do not indicate that certain tracking models perform better than others – for example a 'naive' peak picking algorithm could score 100% on this feature, but it would be a poor beat tracker given a more complicated feature. Furthermore, some trackers may find the nature of the input feature – a delta train – very different to the kind of signals that they were designed for. This may account for differences in performance. We did not attempt to "design" tracking model-specific idealised features. The fundamental point to be taken from the results of an evaluation with such a feature is that existing tracking models can show large performance gains when given more information about beat locations in the signal.

**Improving Beat Tracking By Using A Variety Of Input Features**

The results from Table 4.5 show that if we choose the most appropriate input feature for each file (an 'oracle' input feature), then we can gain a considerable improvement in performance. It appears that different input features provide different information about audio signals. Furthermore, some features are more similar than others. We demonstrate this in Figure 4.4.

Figure 4.4 a) shows the correlation between the results of tracking using the Davies and Plumbley (DP) tracking model with both the Complex Spectral Difference (CSD) input feature and the Bandwise Accent Signals (BAS) input feature on database 2. Both these combinations achieve a score of 67.2% (see Table 4.2).

We can see clearly from this example that the most suitable of the CSD and BAS input features depends upon the audio file under consideration. Indeed, if we were able to choose the best feature for each file we would be able to improve the results by 6.7%. Rather than being separate competing

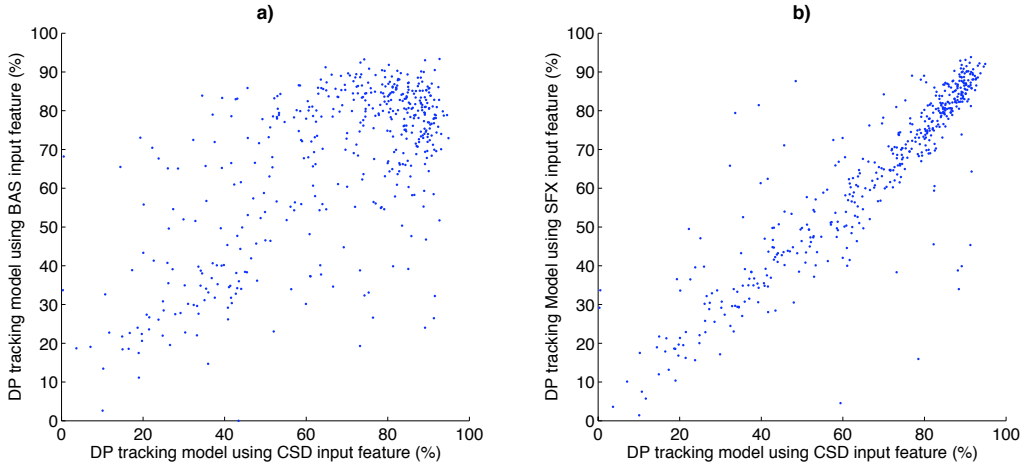



Figure 4.4: a) The correlation between the results of using the Davies and Plumbley (DP) tracking model with two input features - Complex Spectral Difference (CSD) and Bandwise Accent Signals (BAS) on the 474 audio files in Database 2. b) The correlation between the results of using the Davies and Plumbley (DP) tracking model with Complex Spectral Difference (CSD) and Spectral Flux (SFX).

techniques, the features contain *complementary* information that cannot currently be provided by a single input feature. The ability to select between these features automatically would allow us to improve average performance.

Figure 4.4 b) shows the correlation between the results of tracking using the DP tracking model with the CSD input feature and the Spectral Flux (SFX) input feature. We can see in this case that there is much closer correlation between the results of using the two different input features. This implies that they convey broadly similar information and so there is less potential for improvements in performance by choosing between these two features.

### Combined Input Features

Our analysis of state of the art beat tracking techniques has shown that even if a tracking model achieves similar scores with two different input features this does not mean that the input features are representing the same information.

**Database 1**

| Tracking Model | Original Feature Score | Input Features | Comb. Score | Score Difference |
|---|---|---|---|---|
| KL | 67.2 | BAS + MAF + CSD | 67.0 | -0.2 |
| BR | 63.0 | BAS + SFX | 67.3 | +4.3 |
| EL | 61.3 | MAF + CSD | 64.8 | +3.5 |
| DP | 66.6 | BAS + SFX + CSD | 69.8 | +3.2 |
| RT | 63.7 | BAS + SFX + CSD | 65.2 | +1.5 |

**Database 2**

| Tracking Model | Original Feature Score | Input Features | Comb. Score | Score Difference |
|---|---|---|---|---|
| KL | 70.3 | BAS + MAF + CSD | 69.7 | -0.6 |
| BR | 65.2 | BAS + SFX | 68.3 | +3.1 |
| EL | 63.7 | MAF + CSD | 64.9 | +1.2 |
| DP | 67.2 | BAS + SFX + CSD | 70.7 | +3.5 |
| RT | 66.9 | BAS + SFX + CSD | 68.5 | +1.6 |

Table 4.7: The results for each tracking model of using various combinations of input features compared with the score for its original input feature (%). Comb. Score refers to the Combined Score.

In order to exploit this finding, we explored all possible combinations of 2, 3 and 4 input features by normalising them by their standard deviations and summing them together. For example, for two arbitrary input features $\Gamma_1$ and $\Gamma_2$ we calculate the $m$th detection function sample of a combined input feature $\Gamma_c$ by:

$$\Gamma_c(m) = \frac{\Gamma_1(m)}{\sigma_1} + \frac{\Gamma_2(m)}{\sigma_2} \tag{4.1}$$

where $\sigma_1$ and $\sigma_2$ are the respective standard deviations of the input features $\Gamma_1$ and $\Gamma_2$.

We chose the combinations that achieved a balance of both the best scores on database 1 and conceptual simplicity - for example if a combination of 2 features scored slightly less but comparably to a 4 feature combination, we chose the 2 feature combination. The combinations were

then tested on database 2 independently.

The results of this experiment can be seen in Table 4.7. As we can see, the use of a combination of features rather than a single feature is able to improve performance on almost all tracking models, on both databases, with the exception of the Klapuri et al. (KL) model. We hypothesise that the lack of an improvement for the Klapuri et al. (KL) model may be due to the highly feature-dependent nature of that tracking model that we observed earlier. We found that different tracking models responded better to certain combinations depending on the single features they favoured initially. However, we were able to show experimental improvements for almost all tracking models by using a combination of several input features rather than one on its own.

We can conclude from the results that there is clear scope for improving beat tracking by either a) combining complimentary information from multiple features into a single feature or b) selecting between different types of input features for different types of signal.

## 4.5 The Effect Of Flexible Parameterisation

In this section we attempt to establish the potential for improving beat tracking by using existing input features and only improving the tracking model. We approach this by examining the difference in performance, for a given input feature, between the best performing single tracking model and an *oracle tracking model*. The oracle tracking model selects the best performing tracking model per file, for a given input feature. The results are displayed in Table 4.8.

The results show that if we were able to choose the best tracking model for each file, we could achieve an average improvement of at least 5.6%. This is evidence of the benefits of a 'group of experts' system that somehow chooses between a number of tracking models depending on the signal in question. While the use of a 'voting mechanism' to select between different models has been suggested for tempo estimation [Gouyon et al., 2006], selecting between a number of different tracking models is not a particularly

**Database 1**

| **Input Feature** | BAS | SFX | MAF | CSD | $\mu$ |
|---|---|---|---|---|---|
| Best Score | 67.2 | 64.7 | 61.7 | 66.6 | **65.0** |
| Oracle Tracker Score | 73.6 | 69.5 | 67.7 | 71.5 | **70.6** |
| Performance Difference | +6.4 | +4.8 | +6.0 | +5.0 | **+5.6** |

**Database 2**

| **Input Feature** | BAS | SFX | MAF | CSD | $\mu$ |
|---|---|---|---|---|---|
| Best Score | 70.3 | 65.9 | 63.7 | 67.2 | **66.8** |
| Oracle Tracker Score | 77.0 | 72.1 | 71.5 | 74.2 | **73.7** |
| Performance Difference | +6.7 | +6.2 | +7.7 | +7.0 | **+6.9** |

Table 4.8: Results, for each input feature, comparing the best performance obtained using a single tracking model for the whole database, and an oracle tracking model (%)

elegant solution due to the amount of memory, processor power and storage necessary to maintain several algorithms simultaneously (particularly for our real-time case).

Our preference then, in terms of the simplicity of approach to beat tracking, is to use a single tracking model, but still attempt to take advantage of the benefits of signal dependent processing. Therefore, we decided to investigate the benefits of signal dependent parameters for tracking models. We investigated parameter settings in two of the five tracking models, the BeatRoot (BR) model and our real-time model (RT). We investigated three parameters of the BR model and two parameters of the RT model.

For each tracking model we compared the results of the original parameter settings, the best parameter settings over the database and the oracle parameter settings – where the best parameter settings for each file were used.

### 4.5.1 Parameterisation of Tracking Models: BR Model

For the BeatRoot (BR) model we investigated the following three parameters (see Dixon [2006a] for more details):

- $BR_{\alpha}$: The parameter determining the rate of decay of a thresholding function for which the input feature must be greater than to be considered an onset. The original value of this parameter for the BR model was 0.84. We investigated values in the range [0.72, 0.96] in increments of 0.04.

- $BR_{\delta}$: The threshold above the local mean that a peak must reach to be considered an onset. The original value of this parameter for the BR model was 0.35. We investigated values in the range [0.2, 0.5] in increments of 0.05.

- $BR_{\theta}$: The upper limit of the time difference in milliseconds by which an inter-onset interval can be considered part of an existing cluster of inter-onset intervals. The original value of this parameter for the BR model was 25ms. We investigated values in the range [10ms, 40ms] in increments of 5ms.

### 4.5.2 Parameterisation of Tracking Models: RT Model

For our real-time model (RT) we investigated two parameters:

- $RT_{\alpha}$: The mixing coefficient[5] between the input feature and the recursive score from Equation 3.4. We investigated values in the range [0.5, 0.95] in increments of 0.05. The original setting for the RT model was 0.9.

- $RT_{\eta}$: The 'tightness' of a log-Guassian transition weighting used in the search for the most likely previous beat in the recursive score, from Equation 3.2. We investigated values in the range [3, 7] in unit increments. The original setting for the RT model was 5.

### 4.5.3 Parameterisation of Tracking Models: Results

We ran all possible combinations of the parameters of each tracking model within the stated ranges on database 1. We held back database 2 to check

[5]Note that $BR_{\alpha}$ and $RT_{\alpha}$ have entirely different meanings, we simply wanted to maintain a notation consistent with the original description of each algorithm

**Database 1**

| Tracking Model | Original Parameters | Best Parameters | Oracle Parameters |
|---|---|---|---|
| BR | 63.7 | 65.6 | 69.2 |
| RT | 63.7 | 65.2 | 70.4 |

Table 4.9: Results of the best possible parameter settings and an oracle parameter setting compared to the original parameter settings of two tracking models, the BeatRoot (BR) model and our real-time (RT) model. (%)

the results of parameter settings on this database independently.

In Table 4.9 we present the results, for each tracking model, of the original parameter settings, the best single set of parameter settings over the whole database and the oracle parameter setting – the best parameters chosen for each file.

For the BeatRoot (BR) model, the best parameter settings on database 1 were $\mathrm{BR}_\alpha = 0.96$, $\mathrm{BR}_\delta = 0.2$ and $\mathrm{BR}_\theta = 20ms$. This led to a 1.4% improvement on Database 2 from 66.5% to 67.9%.

For our real-time model (RT), the best parameter settings on database 1 were $\mathrm{RT}_\alpha = 0.75$ and $\mathrm{RT}_\eta = 7$. This led to a 0.4% improvement on Database 2 from 66.9% to 67.3%.

However, the results show that using the oracle parameter – where the best parameter settings per file are chosen – leads to much higher scores than optimising a single set of parameters of the database.

The oracle parameter setting for the BR model yield a score of 69.2% – a performance increase of 3.6% above the best single set of parameter settings. Similarly, the oracle parameter setting for the RT model resulted in a score of 70.4% – an improvement of 5.2% over the best single set of parameters.

We believe that these results are clear evidence of the benefits of implementing some form of automated flexible parameterisation in tracking models so that the appropriate parameter settings are used in a given signal.

## 4.6 Review Of Key Findings

Our motivation for conducting a modular evaluation of beat tracking models was to attempt to discover potential ways to improve our real-time beat tracker. Before returning to our real-time beat tracking model, we proceed by summarising the key findings of the chapter.

On the relationship between input features and tracking models and the beat tracking scores they generate:

- We found in section 4.3.1 that for a given input feature a strong mean score across beat trackers implies a strong best score for that feature. We also found that a strong mean score on one database is a moderate determinant of the score of any of the tracking models investigated using that feature on another database. This implies that a good input feature is indicative of good beat tracking more generally.

- We found in section 4.3.2 that tracking models show varying degrees of *feature independence*. Some tracking models, such as the Davies and Plumbley (DP) model, achieve strong scores with a number of features while others, e.g. the Klapuri et al. (KL) model, clearly perform better using a certain input feature - offten the one they were designed to use. As a result, a good tracking model, though a key component, does not directly indicate good beat tracking performance – an appropriate input feature must be used.

On the relationship between input features, tracking models and genre:

- We found in section 4.3.3 that different input features are better at representing different types of signals.

- We found in section 4.3.3 that there is less difference in performance when using different tracking models for a given genre than there is when using different input features.

- We found that there was only a clear difference in performance between non-causal models and our real-time model for genres with

more pronounced tempo dynamics and a lack of a strong beat, such as Classical and Choral music.

On improving beat tracking by improving input features:

- We found in section 4.4 that we could achieve a hypothetical average improvement of over 7.3% in beat tracking performance using existing tracking models just by using an improved input feature.
- Also in section 4.4 we found that when given a hypothetically 'perfect' input feature, all tracking models show drastic improvements and three out of the five tracking models examined scored over 90% on both databases. This implies that there is strong potential for beat tracking improvement by designing a better input feature.
- We showed in section 4.4 that the input features considered in this paper contain some complementary information.

On improving beat tracking by improving tracking models:

- We found in section 4.5 that we could hypothetically achieve an average improvement of over 5.6% in beat tracking performance by selecting the most appropriate tracking model for a given audio file – the 'oracle' tracking model. We believe that this is evidence that a 'group of experts' model that selects automatically between a number of tracking models could improve performance.
- We found in section 4.5.3 that the use of different parameter settings for different signals shows considerable improvement over using the optimal single set of parameters. From this we can conclude that some form of automatic flexible parameterisation could show an improvement over the use of a single set of parameters.

## 4.7 Real-Time Beat Tracking: Revisited

In this chapter we have identified a number of potential ways to improve beat tracking including using varied input features and parameters or

combining the information in two or more input features. We now return to our real-time beat tracker, introduced in Chapter 3, to attempt to use these discoveries to improve our model.

### 4.7.1 Genre Specific Improvements

In this section we attempt to improve our model using genre specific input features and genre specific parameterisation. For offline beat trackers processing large databases of audio files, genre specific processing requires that genre labels are known for all tracks and so this approach has limited use. However, when we are using a real-time beat tracker in a live performance context, the genre of the piece being performed will almost certainly be known in advance, and so it seems entirely sensible that – given that we have seen that different genres and signals are best approached with different input features and parameters – the most appropriate input feature and parameters should be used for the case in hand.

#### Genre Specific Input Features

Table 4.10 shows a comparison, for each genre, between the performance of our real-time beat tracker using the original feature (Complex Spectral Difference or CSD) and with a set of genre specific input features. The results show that overall we can get a gain of 2.4% in performance – with some sizeable gains in specific genres (Dance: +2.1%, Folk: +8.1% and Classical: +6.0%).

#### Genre Specific Parameters

We investigated the best set of parameters for each genre and recorded the difference in performance between our model with the original parameters and with the set of genre specific parameters. The original input feature was used in all cases. The results can be seen in Table 4.11.

We can see an overall improvement of 2.6% with marginal improvements in all genres. The genres that appear to benefit from this approach the most are the Classical (+5.5%) and Rock (+3.3%) genres.

**Database 1: Genre Specific Input Features**

| Genre | Original Feature | Genre Specific Feature | Score Difference |
|---|---|---|---|
| Dance | 82.1 | 84.2 (SFX) | +2.1 |
| Rock | 74.9 | 75.6 (SFX) | +0.7 |
| Jazz | 70.2 | 70.2 (CSD) | 0.0 |
| Folk | 49.9 | 58 (BAS) | +8.1 |
| Classical | 44.9 | 50.9 (BAS) | +6.0 |
| Choral | 23.5 | 24.9 (MAF) | +1.4 |
| **Whole Database** | **63.7** | **66.1** | **+2.4** |

Table 4.10: A comparison of the performance of our real-time tracking model (RT) using its original Complex Spectral Difference (CSD) input feature and using a set of genre specific features (%)

**Database 1: Genre Specific Parameters**

| Genre | Original Parameters | Genre Specific Parameters | Score Difference |
|---|---|---|---|
| Dance | 82.1 | 83.0 | +0.9 |
| Rock | 74.9 | 78.2 | +3.3 |
| Jazz | 70.2 | 71.6 | +1.4 |
| Folk | 49.9 | 52.2 | +2.3 |
| Classical | 44.9 | 50.5 | +5.6 |
| Choral | 23.5 | 25.1 | +1.6 |
| **Whole Database** | **63.7** | **66.3** | **+2.6** |

Table 4.11: A comparison of the performance of our real-time tracking model (RT) using its original parameters and using a set of genre specific parameters (%)

#### Genre Specific Input Features and Parameters

We tested our beat tracker to find the best combination of input feature and tracking model parameters for each genre. The results can be seen in Table 4.12.

We can see that this genre specific arrangement of the beat tracker gains us an improvement in performance of 4% overall. We also see fairly

**Database 1: Genre Specific Parameters and Input Features**

| Genre | Original Setting | Genre Specific Setting | Score Difference |
|---|---|---|---|
| Dance | 82.1 | 84.3 | +2.2 |
| Rock | 74.9 | 78.2 | +3.3 |
| Jazz | 70.2 | 71.6 | +1.4 |
| Folk | 49.9 | 58.4 | +8.5 |
| Classical | 44.9 | 53.7 | +8.8 |
| Choral | 23.5 | 26.5 | +3.0 |
| **Whole Database** | **63.7** | **67.7** | **+4.0** |

Table 4.12: A comparison of the performance of our real-time tracking model (RT) using its original parameters and input feature and using a set of genre specific input features and parameters (%)

dramatic increases in performance for the Folk and Classical genres, with moderate increases for other genres.

### 4.7.2 Future Work: Automatic Parameter Selection

We have seen in this section that genre specific programming of beat trackers can increase performance. However, setting input features and parameters by genre only gives us a moderate increase compared to the 'oracle' input features and parameters studied in this chapter. A better approach would be some way to automate parameter settings based upon signal characteristics. Indeed, we saw in section 4.5.3 that if we can choose the optimal parameter settings for each file we can improve performance from 63.7% to 70.4%. Based upon this result we believe that investigation into some form of automatic parameter selection is a clear contender for future work.

## 4.8 Summary

In this Chapter we have presented the results of a modular evaluation of five beat tracking systems, considering their input features and tracking models separately. We have examined the results of combining different

input features and tracking models. From these analyses we draw several conclusions about the potential for improving beat tracking systems.

Firstly, we can conclude that the approach of using a single input feature for all signal types does not adequately take account of the differences between signals from different genres.. Our research shows that by choosing a more appropriate input feature for a given signal we can achieve considerable improvements in performance. We suggest two possible improvements, either: a) an automatic selection between input features based upon signal characteristics; or b) some form of intelligent combination of input features – or the designing of a new input feature that retains the characteristics of several existing input features.

Similarly, the use of varied parameter settings for different signals – rather than a single global parameter setting – can lead to considerable improvements in performance. A clear challenge for future work will be developing some way to automatically choose appropriate parameters for a given signal.

In future we aim to develop a beat tracking system that exploits these two findings to create a beat tracker that is automatically responsive to the type of signal in which it is trying to discover beat locations.

# Chapter 5

# Performance Following

We have presented in previous chapters a number of techniques for extracting information from musical performances – including current harmonic content and indications of beat locations in real-time. However, there is more to music than the current state of the performance. In particular, we turn in this chapter to its contextualisation within the wider piece – the development of a performance over time.

There has been much research into the field of *score following*. Score following is the automatic matching of musical notes in a performance to those in a score. Through this process we can discover the current position of a performance within a score which allows us to know the future of the performance. As a result, automatic accompaniments can be played.

In some cases, however, no score exists. This is often true for many forms of music, such as rock and pop music. Music may also be improvised, meaning that the production of a score is not possible.

In this chapter we present a technique for predicting the future of performances for which no score exists. We refer to this problem as *performance following*. Our solution takes advantage of the fact that much music contains repetitions of musical phrases [Ockelford, 2005, Ch. 1]. By developing a technique that is able to recognise repeated musical patterns, we can examine the course of previous occurrences of these patterns to make predictions about the future.

Our approach attempts to contextualise recent musical developments (the last few seconds) within the longer term developments of the performance (the last few minutes). Through this process we can identify repeated patterns and make predictions about the future of the performance.

With knowledge of the future harmonic content of the performance we can generate some automatic musical accompaniment, such as a bassline or a melody, with no prior knowledge in the form of a score.

## 5.1 Background

Many forms of music are characterised by the repetition of musical patterns. In prior research, these musical patterns have been represented as sequences of notes [Mongeau and Sankoff, 1990], chords [Pardo and Birmingham, 2001] or other musical features such as chroma features [Dannenberg and Hu, 2003].

Given that these sequences are partially repetitive, the problem of modelling these sequences and making predictions based upon repetition has been an area of interest in recent years.

### 5.1.1 Musical Prediction and Sequence Modelling

Musical prediction has been approached from a number of directions. Some have taken an information theoretic [Pearce et al., 2010] perspective, attempting to model the perceptions of human listeners [Dubnov, 2008; Abdallah and Plumbley, 2009]. Others have made no attempt to model human cognitive processes, employing predictive techniques purely for some practical application, such as predicting chords in Jazz music [Thom, 1995]. In the present work we are also concerned only with solving a practical problem and therefore make no attempt to model human cognitive processes.

Some early work on musical prediction attempted to learn production rules for music from examples. Kohonen [1989] developed a context sensitive grammar which attempts to predict the next element of a sequence based upon a context of a certain length. Should the context provided present two or more equally plausible predictions then the length of the context is dynamically expanded until such conflicts are resolved. In other work, Thom [1995] has presented a technique for predicting chords in jazz based upon N-gram models, showing an ability to predict chords 53% of

the time.

In more recent work, Pachet [2002] introduced the *Continuator*, a live performance system that learns the stylistic characteristics of a performance in real-time or from a stored MIDI file. The input to the system is a symbolic music representation which is processed by a model based on an extension of a Markov model, storing sequences and sub-sequences of notes. Then, new music can be generated in the same style by traversing a prefix tree according to transition probabilities.

Conklin [2003] has argued that there is a strong relationship between analytic and generative models of music and that music generation can be achieved by sampling from analytic statistical models. He highlights the limitations of certain models, including N-gram and Markov-based techniques, that have very specific contexts. Specifically he argues that training corpora are limited in size, so only a few possible sequences will be encountered and therefore there is a sparse data problem. He suggests that transposition of sequences to a common key and smoothing of short and long contexts can be used to mitigate these problems.

Assayag and Dubnov [2004] have suggested using *Factor Orcales* [Allauzen et al., 1999] – automata in the form of a linear chain of states capable of representing all repeated sub-strings in a string – for the analysis and generation of musical sequences. A strength of the Factor Oracle structure is that it is able to represent the presence and location of repeated sub-sequences of variable length. Once a state automata is constructed, it is able to model sequences of length $N$ with a linear number of states $(N + 1)$ and transitions (at most $2N - 1$). Factor Oracles are the basis for the automatic accompaniment system OMAX [Assayag et al., 2006] for polyphonic MIDI and its equivalent for monophonic audio, OFON.

A difficult aspect of Factor Oracles is that they require a single discrete state for each symbol. For MIDI data and monophonic audio, this is possible, assuming in the case of monophonic audio that we have a reliable pitch detector. However, when dealing with polyphonic audio, it is more complicated to convert musical features, such as spectral vectors, into single discrete symbols.

Two suggestions for implementing Factor Oracles using spectral vectors, representing polyphonic audio, have been made. Dubnov et al. [2007] suggest assigning state transitions based upon thresholding a Euclidean distance function between spectral vectors. Bloch et al. [2008] suggest using a "relatively severe quantisation" to reduce spectral vectors into a small number of discrete classes which can then be processed by a Factor Oracle model.

In order to implement either of these two approaches for using Factor Oracles with polyphonic audio, we must use some form of thresholding of a distance function – in the first case to decide on whether to assign transitions and in the second case to decide which class a new spectral vector belongs to. However, due to artefacts, variations in instrumentation, the intensity of performance or the octave at which a particular pattern is played, spectral vectors from musical audio that humans may consider harmonically similar may be different in their constitution. As a result, small values for distance functions are not guaranteed for 'similar-sounding' vectors and so any thresholding of distance functions will be imperfect in some way. Certainly the problem of classifying harmonic vectors into discrete classes is related to the problem of chord recognition, which at present is still an unsolved problem [Cho et al., 2010].

For our application, we wish to be able to recognise repeated musical patterns from polyphonic audio. Therefore, we need a technique capable of 1) comparing sequences of spectral vectors extracted from polyphonic audio; and 2) allowing the comparison of sequences that are non-exact, possibly containing inserted, deleted or substituted elements. We now turn to a widely used family of techniques for the alignment of sequences based upon similar sub-sequences.

### 5.1.2 Sequence Alignment in Music

While string matching algorithms have been a topic of interest in several fields [Gusfield, 1997], many existing techniques for comparing musical sequences are based upon algorithms for comparing sequences of biological data. For the global comparison of two non-exact sequences of amino

acids, Needleman and Wunsch [1970] introduced a technique that first calculates a score matrix based upon the similarity of these two sequences. In the simplest case, a value of 1 is assigned for matching elements and 0 for mismatches. Then, a dynamic programming algorithm is used to calculate all possible pathways through the score matrix. Finally, a traceback step is performed to calculate the best alignment of the two sequences. As all possible pathways through the score matrix are considered, the algorithm allows for sequences to be aligned that are non-identical – possibly containing inserted, deleted or substituted elements.

Building on this work, Smith and Waterman [1981] introduced a technique for computing local alignments between sequences – that is the highest scoring sub-sequence match of two longer sequences. The BLAST algorithm, a faster approximation of local alignment, was presented by Altschul et al. [1990] – although this did not allow for 'gaps' originating from inserted and deleted elements. A gapped version followed later [Altschul et al., 1997].

Sequence alignment techniques such as these have been applied to music in many cases. Mongeau and Sankoff [1990] used sequence alignment techniques to calculate a value of similarity between two musical scores. Monophonic scores were represented as sequences of pitch-duration pairs. These sequences are then aligned taking into account insertion, deletion and substitution as in Needleman and Wunsch [1970]. However, two additional concepts are considered – the replacement of one note by several and several by one.

#### Sequence Alignment In Music Information Retrieval

In the field of music information retrieval, Hu et al. [2003] present a technique that, given an audio file, attempts to find the corresponding MIDI file or vice versa. Rather than attempt any transcription, this is achieved through converting the MIDI file to audio and then computing sequences of chroma vectors from both audio files. These sequences are then aligned by computing a score matrix from the Euclidean distance between all chroma

vectors and then computing the best alignment using dynamic programming techniques and a trace back similar to those described above. The similarity between the audio file and the MIDI file is determined by the average distance along the path. Ferraro and Hanna [2007] investigate various optimisations of alignment algorithms for music through the consideration of different pitch representations, different substitution costs for notes and consideration of note duration. They also find that local alignments show increased performance over global alignments in MIR applications. Robine et al. [2007] suggest improvements to alignment algorithms through the incorporation of some music theory. They consider tonal information, the presence of 'passing notes' – those that do not belong to any chords formed by other notes sounding at the time – and give emphasis to certain beats in the bar by considering 'strong' and 'weak' beats. A review of techniques for melodic similarity used in MIR, as well as general MIR topics, has been presented by Casey et al. [2008].

#### Sequence Alignment For Analysing Music Structure

Some have used sequence alignment techniques for analysing the structure of musical pieces. Dannenberg and Hu [2002] use dynamic programming based sequence alignment techniques to discover repeated segments in a piece of music and therefore to create structural descriptions of a piece of music. They present three approaches based upon monophonic pitch, chroma and chord progressions derived from a polyphonic transcription. Pairs of similar segments are computed and then clustered to identify the different segments present in the music. Kilian and Hoos [2004] adapt the widely used BLAST algorithm [Altschul et al., 1990] for use with musical data and apply it to the problems of analysing the structure of an input file or for retrieving occurrences of a given search pattern. Meredith et al. [2003] have presented a number of algorithms able to identify repeated material in polyphonic music. In other work, a statistical model for the segmentation of melodies has been presented by Pearce et al. [2008].

**Musical Sequence Alignment In Live Performances**

While the systems above largely process audio files in an offline context, we wish to use it in a real-time live performance situation. Several score following systems have made use of sequence alignment techniques in real-time.

Dannenberg [1984] has presented a technique for comparing a monophonic performance to a score in real-time. Pitch estimation is used to turn the monophonic audio into a series of pitches, which are then compared to a score using a dynamic programming sequence alignment technique. This approach was later adapted to handle polyphonic keyboard performances [Bloch and Dannenberg, 1985].

Some genres of music – in particular folk and popular music – use scores that provide only limited information such as the main melody or chord sequences. These are referred to as partially specified scores. Pardo and Birmingham [2001] outline an approach for the comparison of polyphonic MIDI performances to a partially specified score in the form of a sequence of chords. A chord sequence is extracted from the MIDI performance and this is compared to the partially specified score using a global sequence alignment.

Dannenberg and Hu [2003] suggested that their technique for comparing polyphonic audio to a symbolic MIDI file could be used as part of a real-time system for polyphonic performances. By synthesising audio from the MIDI file, a comparison of two audio files was made by extracting sequences of chroma vectors from the audio and then computing an alignment between the two.

A technique called *MATCH* has been presented by Dixon and Widmer [2005] for aligning polyphonic audio recordings of different performances of the same piece of music. This is intended to be a useful tool for musicologists who can use it to switch between time aligned performances in real-time. With audio files represented as sequences of spectral vectors – a linear representation at low frequencies and logarithmic at high frequencies – spectral difference vectors for each audio frame compared

to its predecessor are calculated. The Euclidean distance between vectors is then used to compute a cost matrix before an alignment is found through dynamic programming based sequence alignment. This technique was later applied to the problem of tracking live performances in real-time [Dixon, 2005].

It should be mentioned that the problem of score following has been approached without using sequence alignment techniques. For example, Raphael [2001] implemented a score following system based upon hidden Markov models [Rabiner, 1989] while Cont [2010] presented *Antescofo*, a system also based upon a probabilistic framework but using an 'anticipatory' system whereby a predictive model of future information is used to inform current choices.

## 5.2 Approach

The live performance systems described above all compare a musical performance to some form of score. However, we are approaching the problem of predicting future harmonic content in performances when no score is available.

Our approach is based upon the hypothesis that much music contains repetition of musical patterns. Our aim is to 'recognise' these repeated patterns by comparing the present developments of a performance to the past. If the current musical pattern has been repeated in the past, and if we can locate those previous occurrences, then we can use them to make predictions about the future of the performance.

Score following techniques typically compare a live performance to a score in order to discover the location of the performance within the score. As we have no score to match against, we instead contextualise the recent past within the longer term history of the performance. Specifically we compare the most recent few seconds of the performance to the last few minutes. The past of the performance itself is used as if it was the 'score'.

There are some problems to overcome with this approach. Firstly, tempo variations in the performance may mean that the same musical pattern may be of different lengths depending on when it is performed. A

second problem is that comparing high quality audio signals in their raw form is computationally expensive.

We therefore approach this problem by representing musical patterns using real-time beat-synchronous sequences, with a single vector for each inter-beat interval. We then implement a sequence alignment technique based upon dynamic programming to make predictions of future harmonic content based upon the past of the performance.

#### Overview of System

We use as an input to our system a polyphonic audio signal from a single instrument, such as a guitar or piano. By choosing a single instrument we can avoid the problems caused by percussive instruments that may be found in a mixture of a whole ensemble. In real-time, using chroma analysis and either a beat tracker or fixed tempo click track, we convert this signal into a beat-synchronous sequence of chroma vectors. We then use this beat-synchronous sequence as an input to our performance following technique which outputs a chroma vector as a prediction of the harmonic content of the next inter-beat interval. A graphical overview is displayed in Figure 5.1.

## 5.3 Beat-Synchronous Sequences

Our approach involves the comparison of current musical patterns to those in the past – in practice comparing the harmonic content of current audio to that of previous audio. This approach presents us with a number of problems. Firstly, we must be able to compare musical patterns that occur at different tempi and so will be of different lengths. Secondly, comparisons of large amounts of audio – spanning several minutes – is computationally expensive. Finally, we have the problem that some audio frames will contain audio from before and after a harmonic change, making the determination of the harmonic content of such frames non-trivial. In order to solve these problems we use the beat-synchronous sequences described in Chapter 3.

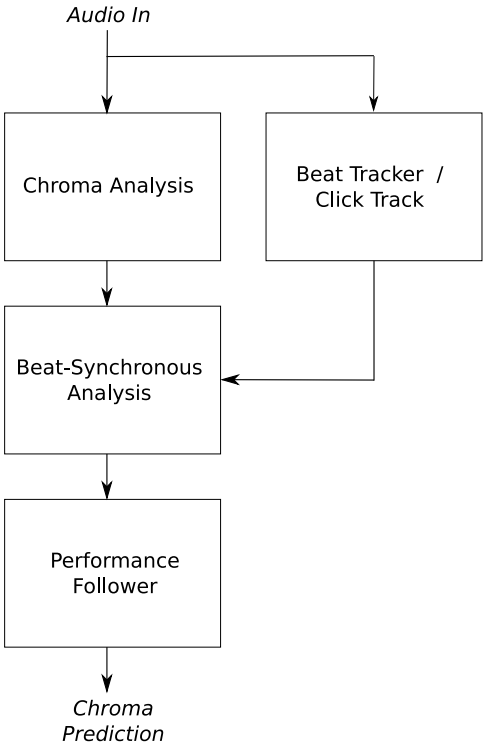



Figure 5.1: a) An overview of the performance following technique.

For our application, there are several advantages to representing music beat-synchronously. Firstly, we are able to represent the same musical theme with the same number of features regardless of tempo, allowing us to more easily compare musical sequences.

A second benefit is that a beat-synchronous representation reduces the amount of information needed to represent a musical sequence – and therefore the computational cost of any subsequent sequence comparison. Given 60 seconds of audio at 44.1kHz, if we use a frame size of 1024 samples then we will need $(\frac{44100}{1024} \times 60) > 2,500$ feature vectors to represent the signal. However, if the tempo is 120 beats per minute, then a beat-synchronous representation can represent the same signal with just 120 feature vectors.

Finally, beat-synchronous techniques have been shown to increase performance in harmonic analysis in other research [Bello and Pickens, 2005]. This is because harmonic changes often occur at beat locations, and so the use of beat-synchronous representations allows us to avoid some of the

problems of audio frames that contain harmonic information from both before and after a harmonic change.

In practice, for our application we make use of beat synchronous sequences of chroma vectors. We represent the $k$th inter-beat interval – the time between beats $[\gamma_{r-1}, \gamma_r]$, where $r = k + 1$ – with a beat-synchronous vector $\Delta_k$, calculated according to method 2 in section 3.4.

### 5.3.1 Real-Time Beat Tracking vs. Fixed Tempo Click Tracks

In order to extract beat-synchronous sequences in real-time, we need some measure of the beat. We have suggested in Chapter 3 that a real-time beat tracker can be used for this purpose. While we have detailed a reasonably robust and efficient real-time beat tracker in section 3.1, the beat tracking evaluation in Chapter 4 has shown that 100% guarantees of performance accuracy are still not possible with current beat tracking models.

A second option is to use a fixed tempo click track. Such a computationally 'correct' source of beats has the benefit of being reliable but the disadvantage of being inflexible – performers must adhere to the tempo of the click track.

For our work here we suggest that either option can be used based upon their relative merits depending on the musical style.

## 5.4 Performance Following: Sequence Prediction

In order to perform sequence prediction, we maintain two sequences of beat-synchronous chroma vectors. The first, $A = a_1, a_2, ..., a_N$ is a sequence containing the $N$ most recent beat-synchronous vectors, which we we refer to as the *long term memory*. The second sequence, $B = b_1, b_2, ..., b_M$, is a shorter sequence containing the $M < N$ most recent beat-synchronous vectors, which we will call the *short-term memory*. Specifically, we define:

$$A = \Delta_{k-N+1}, ..., \Delta_k \tag{5.1}$$

$$B = \Delta_{k-M+1}, ..., \Delta_k \tag{5.2}$$

where $\Delta_k$ is the $k$th, and most recent, beat-synchronous chroma vector.

### 5.4.1 Choice of Sequence Lengths, $N$ and $M$

We wish to contextualise the shorter sequence, $B$, within the longer term developments of sequence $A$. We therefore choose the values of $N$ and $M$ accordingly. For the length, $N$, of the longer sequence $A$, we must choose a value large enough to store recent developments in the music, while being small enough to allow computation to occur in real-time. We suggest that this sequence should represent a length of time of around 2 or 3 minutes as this will likely include recent musical developments – though for different musical styles it can be adapted. Given a hypothetical tempo of 120bpm, a length of 300 for $N$ allows the storage of the most recent 2 minutes and 30 seconds, which seems to easily allow computation in real-time (using $N = 300$ and $M = 50$, our model can process a 4 minute piece at 120bpm – 480 vectors – in just under 6 seconds).

The value $M$ determines the length of the longest sub-sequence match between the short-term and long-term memory. We will discuss the choice of the value of $M$ in detail in section 5.6.1, but for now we suggest a value in the range 5 to 50.

### 5.4.2 Sequence Alignment

In order to discover, in sequence $A$, occurrences of similar musical patterns within sequence $B$, we use a sequence alignment technique, computing an alignment matrix between the two. Our technique is an adaptation of the Smith-Waterman algorithm [Smith and Waterman, 1981].

The first step of our technique is to compute a self-similarity score matrix $s(i, j)$. This is achieved by calculating the inner product of $a_i$, the $i$th beat-synchronous vector of sequence $A$, and $b_j$, the $j$th beat-synchronous vector of sequence $B$:

$$s(i, j) = \sum_{v=1}^{V} a_i(v) \times b_j(v) \tag{5.3}$$

where $v$ is the index of the bin number for each feature vector and $V = 12$, the length of each chroma vector.

We then calculate an alignment matrix $H$ using dynamic programming. The value $H_{i,j}$ indicates the best score for the alignment of two

sub-sequences ending in $a_i$ and $b_j$. We initialise the values of $H_{i,0}$, $H_{0,j}$ and $H_{0,0}$ to zero. We then proceed with the calculation of the rest of the matrix:

$$H_{i,j} = \max \begin{cases} H_{i-1,j-1} + s(i,j) \\ H_{i-1,j} - W \\ H_{i,j-1} - W \\ 0 \end{cases} \tag{5.4}$$

where $1 \leq i \leq N$, $1 \leq j \leq M$, and $W$ is the gap penalty that penalises alignments that contain inserted or deleted elements. We choose $W = \frac{4}{3}$ in a similar way to Smith and Waterman [1981]. In Figure 5.2 we show an example of an alignment matrix resulting from the comparison of a longer sequence of 300 beat-synchronous vectors and a shorter sequence of 50 beat-synchronous vectors.

From the alignment matrix $H$, we can make a prediction of the next likely element in the beat-synchronous sequence. We do this by finding points where the last elements of sequence $B$ align strongly with sequence $A$. For the example in Figure 5.2 there are three potential alignments, identified by strong diagonals and highlighted by arrows. Finding these points of strong alignment is then achieved by analysing the final column of the matrix $H$ and choosing the value

$$y = \arg \max_{1 \leq i < N} H_{i,M} \tag{5.5}$$

In practice, harmonic content may be slow in changing and so strong alignments may appear for sequences just one or two beats in the past, which are unlikely to be correct. Therefore, we wish to indicate that these alignments are unlikely and to favour alignments further in the past. As a result, we amend Equation 5.5 by adding the parameter $\beta$ which allows us to not search the most recent $\beta$ elements in the long term memory for a strong alignment:

$$y = \arg \max_{1 \leq i < (N-\beta)} H_{i,M} \tag{5.6}$$

If we choose smaller values of $\beta$ we allow shorter repeats to be identified, while larger values allow quicker identification of correct alignments,

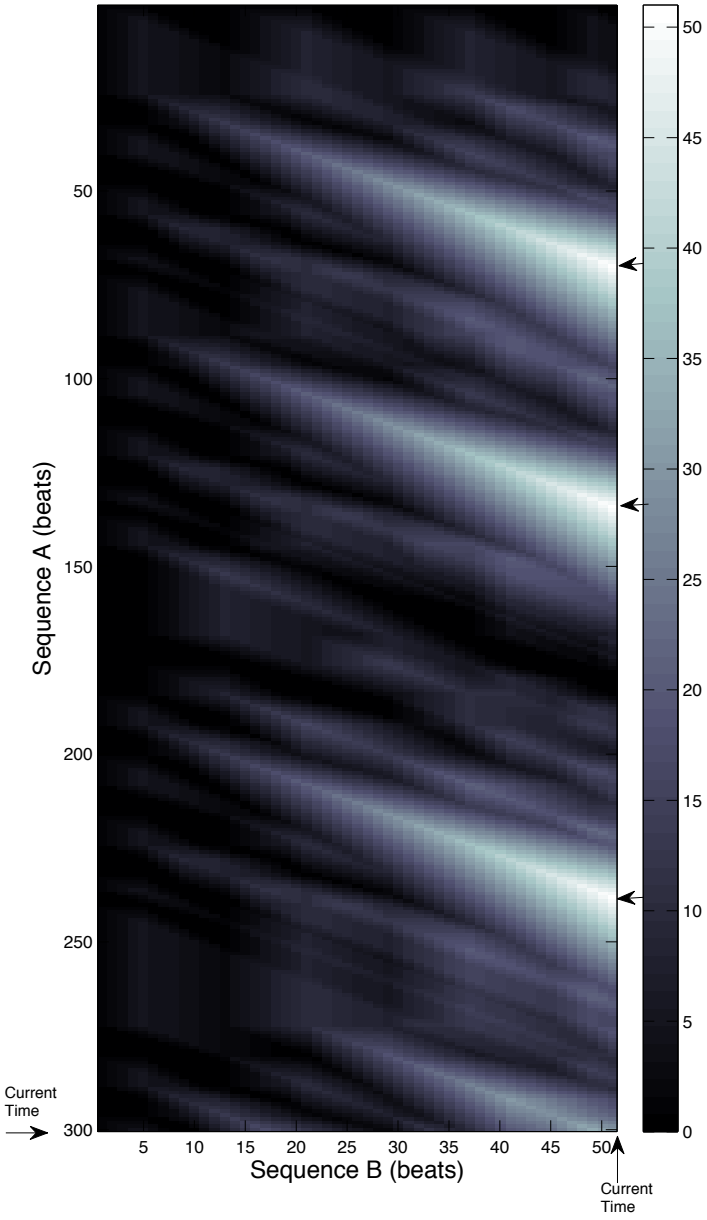



Figure 5.2: The matrix resulting from the comparison of the most recent 300 beats of a performance with the most recent 50. As can be seen in the final column of the matrix, there are three potential alignments (identified by arrows) indicating that the fragment contains a repeated part of the performance.

assuming that the period of repetition is longer than $\beta$. Through experimentation we choose $\beta = 10$.

Finally, we choose the previous feature vector – specifically the one following the strongest alignment indicated by $y$ – as a prediction of the content of the next element, $\hat{b}_{M+1}$, of the beat-synchronous sequence $B$:

$$\hat{b}_{M+1} = a_{y+1} \tag{5.7}$$

## 5.5 Evaluation

### 5.5.1 Musical Sequence Alignment Database

In order to evaluate our system, we created a database by asking a number of musicians to compose acoustic guitar pieces. The musicians were given no specific instructions on the nature of the content for the pieces and no mention was made of the repetition of musical themes. The musicians did, however, have a natural stylistic leaning towards the rock, pop and folk genres. The result is a database of 32 pieces in those styles, totalling over 104 minutes of audio. All files in the database are mono, 16-bit audio at 44.1kHz. The pieces in the database have an average length of 3 minutes 16 seconds with a standard deviation of 36 seconds.

For each audio file, the beats were labelled using a beat tracker with any erroneously tracked beats corrected by a human annotator so that all beat times were correct.

In assessing the performance of our technique, we initially considered using the Euclidean distance between predicted and observed feature vectors. However, our initial tests showed that distance measures such as these did not necessarily show a small distance between vectors that were harmonically similar to the human ear. This may have been the result of variations in the intensity of the performance, or the octave at which a given harmonic sequence was performed.

As a result, we created a ground truth in order to perform an objective evaluation of our technique. For each audio file, we grouped repeated sections together. For example, for a given piece, the content of inter-beat intervals (IBIs) 1 to 32 may be repeated during IBIs 33 to 64 and 97 to

128. Such a scenario is depicted in Figure 5.3.

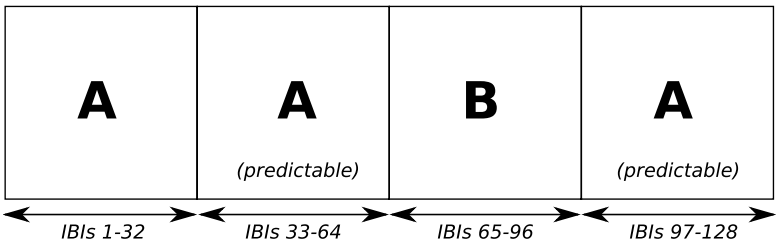



Figure 5.3: The content of inter-beat intervals (IBIs) 1-32 is repeated from 33-64 and 97-128. This is shown as the three occurrences of section A. In this example, the content of inter-beat intervals 65-96, labelled B, is not repeated elsewhere and so is not 'predictable'. Overall in this example, 50% of content is 'predictable'.

In order to record such information as an annotation, we use the syntax

[[1,32],[33,64],[97,128]].

Based upon such an annotation, for IBI 33 we allow a prediction of IBI 1, for IBI 34 we accept a prediction of IBI 2 and so on. There may be multiple repeats of a musical pattern and so for later repetitions multiple IBIs may be accepted as correct. For example, for IBI 97 we accept predictions of IBI 1 or IBI 33. Our final annotation is a list of acceptable predictions (APs) for each IBI:

| **IBI** | **APs** |
|---|---|
| 1: | none |
| 2: | none |
| ⋮ | ⋮ |
| 32: | none |
| 33: | 1 |
| 34: | 2 |
| ⋮ | ⋮ |
| 97: | 1,33 |
| 98: | 2,34 |
| ⋮ | ⋮ |

To evaluate a system on the database, for each prediction made, we assess it as either correct or incorrect based upon whether the prediction

made is in the list of acceptable predictions for each inter-beat interval. For each file we obtain the percentage of correct predictions and then take the mean to arrive at a final score for the system in question.

It is clear from our description of the annotations that some IBIs will have no acceptable predictions and therefore cannot be considered 'predictable'. We do not assess the performance of our technique on these IBIs as the results for each audio file would vary greatly depending upon the amount of repeated content.

While the stylistic content of the database is similar to much popular music, it is interesting to note that 94.1% of the content is made up of sections that are repeated elsewhere in the piece and of 76.2% is 'predictable' in the terms we have described. We have made the database, annotations and all source code available online under a Creative Commons license so that our results may be independently reproduced[1].

### 5.5.2 Methodology

A difficulty in performing such an evaluation is that if we use a beat tracker and some of the beats are incorrectly tracked, then it becomes impossible to tell if poor performance is due to poor localisation of harmonic content by the beat tracking or poor performance of our predictive performance following technique. Therefore, for the evaluation, we decided to evaluate our technique using the beat annotations from the database, rather than allow for the errors that may occur using a purely automatic beat tracker.

It is accepted that if we were to use a beat tracker in a live performance context (rather than the evaluation) then any beat tracking errors would result in a detrimental effect on performance. However, our experiments show that these situations do not occur with a regularity that would make the system unusable. It is also perfectly possible to largely avoid this problem by using constant tempo click tracks.

We made predictions for each audio file as follows. For each audio file we calculated a sequence of beat-synchronous chroma features. We then

[1] `http://www.eecs.qmul.ac.uk/~adams/pf/`

used our performance following technique to attempt to predict acceptable harmonic content from the past of the performance. At each step we recorded the index number of the inter-beat interval of the predicted chroma vector. We then calculated a score as a percentage of correct predictions given the ground truth annotations from the database.

### 5.5.3 Comparison To Other Techniques

In order to assess the performance of our technique relative to others, we implemented three other techniques – a random predictor, an N-gram based model and a Factor Oracle based technique.

#### Random Predictor

As a baseline, we implemented a random predictor that simply chose a random inter-beat interval from the past of the performance as a prediction of future content.

#### Factor Oracle Predictor

We implemented a Factor Oracle (FO) according to Assayag and Dubnov [2004], making use of K-means clustering to group individual beat synchronous feature vectors into clusters so that a model with discrete states such as a FO could process the sequence of data. We followed suffix links in the model which identify previously repeated factors and used these to choose past inter-beat intervals as predictions of future content.

We experimented with using different numbers of clusters, trying values between 1 and 20. We recorded the result for the best single number of clusters (4 clusters) and also the result for a version where the best number of clusters is chosen for each file. Given that some random initialisation is involved with the K-means clustering, we ran the results 20 times and took the mean for each file.

#### N-gram Predictor

We also implemented a predictor based on an N-gram model [Jurafsky and Martin, 2000, Ch. 6], again making use of K-means clustering to

turn the data into a sequence of discrete symbols. We processed the sequences causally, recording the best transition given a certain sequence. The prediction recorded was the inter-beat interval number from the past of the performance for the transition the last time the given sequence occurred.

Specifically we chose lengths $L$ of 3, 5, 10 and 15 for the N-gram model. We used the optimal number of clusters for each value of $L$ and, as with the Factor Oracle approach, we averaged scores over 20 runs to reduce the effect of any random initialisation during clustering.

## 5.6 Results and Discussion

The results of our evaluation can be seen in Table 5.6. The highest scoring model is our performance following technique (PF) with a short term memory length of $M = 20$, which has a mean score of 75.3% correctly predicted IBIs. Given the standard deviation of 13.9% and the sample size, this gives a 95% confidence interval of $[75.3\% \pm 4.81]$ calculated according to Flexer [2006]. This is also a prediction of a majority of content in the database overall – 57.4% – given that the mean scores are only on the 'predictable' 76.2% of the database.

Our technique also performs better than both the N-gram and Factor Oracle techniques. We believe that the need to cluster vectors into discrete classes before processing is a problem for these latter techniques and that a strength of our model is that it can locate repetitions of sequences of polyphonic harmonic data without making the data discrete.

There are, however, several areas for potential improvement, to which we turn now.

### 5.6.1 Short-Term Memory Length (M)

Our evaluation showed that the best single value for $M$ was 20. However, different pieces were suited to different values of $M$. Figure 5.4 shows the number of examples that achieved their maximum score for a given value of $M$. We can clearly see that the majority of examples could have scored

better with values of $M$ other than 20. Ten achieved their best score for $M = 15$ but another 19 examples (59%) scored best with values of $M$ that were neither 15 or 20.

The implication of these results is that our model could be improved by implementing $M$ as an adaptive parameter. If we could find some way of automatically adjusting $M$ to suit the characteristics of the piece in question then our evaluation suggests we could achieve an improvement of over 3%. By choosing the best value of $M$ for each file for our evaluation – an 'oracle' approach – we reach a 95% confidence interval of [78.7%±4.08].

| **Model** | **Parameter** | **Score / %** | $\sigma$ / % |
|---|---|---|---|
| PF | $M = 5$ | 50.6 | 17.4 |
| PF | $M = 10$ | 69.6 | 17.6 |
| PF | $M = 15$ | 75.1 | 14.3 |
| PF | $M = 20$ | 75.3 | 13.9 |
| PF | $M = 25$ | 74.3 | 13.9 |
| PF | $M = 30$ | 73.2 | 14.2 |
| PF | $M = 35$ | 72.6 | 14.4 |
| PF | $M = 40$ | 71.5 | 15.0 |
| PF | $M = 45$ | 70.4 | 15.4 |
| PF | $M = 50$ | 69.3 | 16.0 |
| FO | $D = 4$ | 63.0 | 19.1 |
| FO | Best $D$ Per File | 68.5 | 16.6 |
| N-Gram | $L = 3, D = 13$ | 34.8 | 15.4 |
| N-Gram | $L = 5, D = 6$ | 38.0 | 16.9 |
| N-Gram | $L = 10, D = 3$ | 42.1 | 17.7 |
| N-Gram | $L = 15, D = 3$ | 38.0 | 20.8 |
| Random | - | 2.2 | 1.4 |

Table 5.1: The results of evaluating our performance following technique (*PF*), a Factor Oracle (*FO*), an N-gram model (*N-Gram*) and a random predictor (*Random*) on a database of 32 annotated audio files. Results are given as mean percentage scores (%) and the standard deviation of those scores ($\sigma$). The parameter $M$ is the short-term memory length of the performance following technique, the parameter $D$ is the number of clusters used in the K-means clustering for the N-gram and Factor Oracle models, and the parameter $L$ is the length used in the N-gram model.

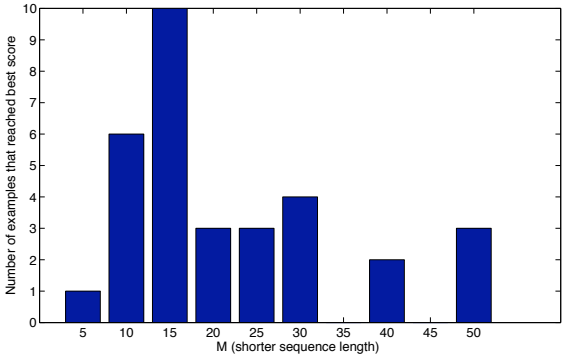



Figure 5.4: The number of examples achieving their maximum score for a given value of $M$, the short-term memory length.

While our technique can pick up on repetitions of varying lengths up to a size of $M$ (due to an ability to match sub-sequences), we may also be able to improve performance by combining several performance following techniques with varying values of $M$.

Informal real-time use of our model indicates that smaller values of $M$ cause the system to be more responsive to fast changes in the music while larger values provide more context and allow predictions based upon more analysis of sequence development over the past of the performance.

For choosing a single value of $M$, it would be valuable to consider data from psychological studies into short term memory in humans [Anderson, 2000] and music theoretic concepts of the time scales of music – in particular the grouping of sounds into phrase structures at the 'meso-level' [Roads, 2004].

### 5.6.2 The Selection of Features

A limitation of our approach is that we only use harmonic features. There are many other aspects of music, such as rhythmic and timbral information, that characterise musical patterns and we may be able to increase the performance of our system by representing these aspects in musical sequences.

Beat-synchronous features are beneficial to our system in a number of ways. In addition to those discussed in section 5.3, harmonic information is

'smoothed' through being made beat synchronous (i.e. there is a reduction in the resolution), so variations in the harmonic themes can be tolerated. For example, in the case of an acoustic guitar, a strummed sequence can be successfully compared with an arpeggiated version of that chord sequence.

There are, however, limitations to using beat-synchonous features. We may have harmonic changes at a rate that is faster than the metrical level used by the beat-synchronous process. In this situation the temporal resolution of the beat-synchronous sequence would be too low to correctly represent the content of the performance. An obvious solution would be to use a faster metrical level. However we may then reach a point where the amount of audio in inter-beat intervals is too short to allow sufficient frequency resolution to accurately distinguish frequencies in the signal. The use of overlapping frames would only 'blur' representations across beat boundaries.

### 5.6.3 Evaluation of the Effect of Beat Tracking Errors

It would be informative to perform an evaluation that examined the effect of beat tracking errors on our approach. Given that our current evaluation is based upon annotated beat locations, and considering that a beat tracker would place beats in different places, it is unclear how to make any meaningful comparison between our ground truth evaluation and an evaluation for the beat tracker. Indeed, given our discussions in section 5.5.1 of the problems using distance measures such as the Euclidean distance in our evaluation, it is not clear how to perform any objective evaluation when the beat locations are not correct.

Such an evaluation would also be useful in allowing us to assess the useful values of the gap penalty $W$ from Equation 5.4. This penalty is rarely used in the current evaluation as it is designed to deal with beat tracker errors - e.g. the situation where there are 5 beats where there should be 4, for example. Given that we are using the ground truth beat locations we do not encounter such errors in the current evaluation. We will investigate ways of performing an evaluation using a real-time beat tracker in future – perhaps through a subjective user-oriented study of the

system.

### 5.6.4 The Annotation Process

It would be useful to develop a formal definition of a 'repeated section'. In the current study, repeated sections were identified through the subjective perception of a musician but there are several different interpretations of what can be considered a 'repeated section'. However, given the variability and complexity of music, it is difficult to make such a formal definition.

For example, if the harmonic content in one inter-beat interval (IBI) was similar to that of the previous IBI then we could technically call this a repeat. However, many people would disagree with such a classification due to the brevity of the section. If we are to decide that a certain number of IBIs have to be repeated before it is considered a repeated section then we can only place some arbitrary threshold. Furthermore, this threshold would be undermined by different tempi, time signatures and metrical levels. We could also say that there must be a certain number of harmonic changes – but if the music is more complicated than a simple chord sequence then changes will be harder to identify.

### 5.6.5 Scope of the Work

Our technique is focused upon music that is based, at least partially, on the repetition of musical patterns, such as music from the rock, pop and folk genres. Therefore, we make no claim that this approach is extendable to all musical styles.

We also believe that the use of a single instrument, such as a guitar, as the input signal is a sensible approach. In a live performance, we are much more likely to obtain a good quality audio signal directly from a single accompanying instrument than to attempt to process a signal from a complicated mixture of instruments and then to deal with the multi-instrumental nature of that signal. Our use of a single instrument has precedents in other research – for example, during research on the beat tracking of drums by Robertson and Plumbley [2007].

## 5.7 Directions for Future Work

In addition to the areas identified for future study when assessing the results – namely creating an adaptive short-term memory length $M$, adding musical features that include more than just harmonic information and a review of the procedures used in the evaluation – there are several other areas that would be worthy areas for future work.

Currently the technique is focused upon making predictions of harmonic content in the presence of repetition. However, to be truly useful, we believe our technique should be integrated into a larger system capable of making predictions of harmonic content in the absence of repetition. The use of data from multiple performances and rehearsals may be useful in this context. It would also be useful to incorporate meta-data inputs to our system. These could be used to indicate sharp, unpredictable changes in tempo or harmonic shifts such as key changes.

The representations used for harmonic content could be amended so that transposed versions of musical patterns could also be identified and harmonic predictions made.

Finally, given that we have full sequences of harmonic data, we can use our Performance Following technique as the input to an automatic accompaniment system. We will return to this idea in Chapter 6.

## 5.8 Summary

In this chapter we have presented a technique for predicting harmonic content in polyphonic audio signals based upon the repetition of musical patterns, with no access to a musical score. Furthermore, our technique is real-time, allowing its use in live performances.

With beat-synchronous sequences of harmonic features as an input, we have used a sequence alignment technique based upon dynamic programming to identify repetitions in the past of recent musical developments within the wider context of the performance. Based upon the events proceeding these repetitions, we have made predictions of future harmonic content.

We have conducted an objective evaluation of our technique on a database of acoustic guitar pieces, comparing it to several other techniques. Our approach outperforms the others in the evaluation, and we have demonstrated that we are able to predict a large majority of repeated content, and a majority of content overall, with no prior information in the form of a score.

## Reproducible Research

We have made available, under a Creative Commons license, both the data set and Matlab source code so that our results can be independently reproduced. We have also made available a real-time implementation of our technique for the Max/MSP environment[2].

[2] http://www.eecs.qmul.ac.uk/~adams/pf/

# Chapter 6

# Musical Audio Analysis In Practice: Applications for Live Performance

In previous chapters we have described a number of techniques for extracting information from live performances in real-time. The information from such musical audio analysis can be used by some form of live performance application to produce a response – audible, visual, tactile or otherwise. We now describe a number of different applications for live performance based upon the analysis techniques presented in Chapters 3, 4 and 5.

## 6.1 Beat Tracking and Audio Effects

Audio effects are now commonplace tools within live musical performances and recording studios. Each audio effect will have a number of parameters, controllable by musicians, such as the length of a delay time in milliseconds, or the rate of an oscillator in Hertz (Hz). In some situations it can be desirable to link the parameter values to the tempo or beat of a performance.

However, effects that are synchronised to the tempo and beat can be difficult to create in live performance situations. For example, to create a delay effect (similar to an 'echo') where the signal is delayed for the length of one beat, we must know the exact tempo of the performance when setting the delay time. Calculating tempo information is not easy (or desirable) for musicians to do in real-time performance situations. Furthermore, the tempo of the performance may later change, rendering previous audio effect settings obsolete.

One solution to such a problem is for musicians to synchronise their performance to an artificial 'click track' – a computer controlled series of pulses at a specified tempo. This click track will not vary in tempo and so audio effects can be created that are related in some way to the beat or tempo of the performance. Synchronisation is maintained as long as the musician does not deviate from the click track. Unfortunately, this 'flat tempo' approach can lead to performances that are mechanical or inexpressive.

Another solution is to use one of several commercially available 'tap-tempo' pedals, such as the Line 6 Echo Park[1]. These pedals allow the user to 'tap' their foot twice in time to the beat, with the time between 'taps' used to inform the effect of the tempo. Such an interface does allow the creation of effects synchronised to the beat or tempo in expressive time-varying performance. However, as the tempo in the performance varies, the musician will have to update the pedal with the new tempo. This is undesirable as it creates a need for the musician to concentrate on the maintenance of technology rather than the performance itself.

In this section we present a solution to this problem by using beat and tempo information from a real-time beat tracker to automatically update audio effect parameters. The musician is able to set parameters in terms relative to the tempo or beat, for example applying a delay time of a single beat, rather than a number of milliseconds. Beat tracking information allows us to automatically update parameter settings to stay synchronised with a live performance.

Beat-trackers provide us with two pieces of information – the performance tempo and the location, or phase, of the beats in the performance. We describe both *tempo-synchronous audio effects*, that make use of only tempo information, and *beat-synchronous audio effects* that make use of both tempo and beat phase information.

[1] `http://line6.com/tonecore/echoPark.html` (accessed 25th February 2011)

### 6.1.1 Tempo-Synchronous Audio Effects

Tempo-synchronous audio effects automatically adapt effect parameters to synchronise with the performance tempo – as provided by a real-time beat tracking system. We present here a tempo-synchronous delay effect.

#### Tempo-Synchronous Delay

A delay effect creates the impression of an 'echo' by delaying sounds for a certain amount of time and mixing them with the original signal. A conventional delay effect uses a user-defined *delay time* parameter, specified as a number of audio samples (though the user will normally control this in milliseconds). The output signal, $y[n]$, is calculated by [Zolzer, 2002]:

$$y[n] = x[n] + \alpha \cdot x[n - Q] \tag{6.1}$$

where $x[n]$ is the input signal, $Q$ is the number of audio samples to delay the signal by and $\alpha$ is the gain factor that controls the amplitude of the delayed signal.

In order to make this effect tempo-synchronous, we replace the parameter $Q$ with a value related to the beat period, $\tau$, provided by the beat tracker:

$$y[n] = x[n] + \alpha \cdot x[n - (\lambda \cdot \tau)]. \tag{6.2}$$

The user is now able to control the delay time in multiples of the beat using the $\lambda$ parameter, rather than by setting a delay time as a number of audio samples $Q$.

At each beat we will receive a new estimate of the beat-period from the beat tracker. The likely scenario is that there will be variations in the tempo and this will cause sharp changes in the length of the delay time. A potential side effect of this is that there can be phase mismatches between the delayed audio signal before and after the tempo change. This can lead to unpleasant audible artefacts. In order to avoid this problem, we crossfade the pre- and post-tempo change delayed signals for the length of a single audio frame (in the case of our real-time beat tracker presented

in Chapter 3 this is 512 audio samples). The result is a smooth change from one delayed signal to another, with no artefacts.

### 6.1.2 Beat-Synchronous Audio Effects

Beat-synchronous audio effects differ from tempo-synchronous audio effects in that they use both the tempo and beat location information to inform performance. Some audio effects have an inherent frequency at which they perform their processing – for example the modulation of the amplitude of an audio signal in a tremolo effect will happen at a rate specified in Hz. They are often controlled by an oscillating signal or *oscillator*.

If we are to synchronise effects such as these to the temporal characteristics of live performances, then in addition to relating the frequency of an oscillator (in Hz) to the tempo of the performance, we may want to cause the 'phase' of the oscillator to be synchronised with the location of beats. For example, we may want the maxima of the oscillator to occur at beat locations.

In this section we present a number of *beat-synchronous audio effects* that make use of both tempo and beat phase information from a real-time beat tracker. Specifically we present a technique for creating a beat-synchronous oscillator from a phasor signal and then detail its use in a number of oscillator based effects – a tremolo, a flanger and an auto-wah.

#### A Beat-Synchronous Phasor

The key component of beat-synchronous audio effects is a beat-synchronous phasor. The beat-synchronous phasor, $\theta$, takes values in the range $[0, 2\pi]$ representing the phase position of some arbitrary oscillator with the constraint that the cycle should begin at zero at one beat and end at $2\pi$ after $R$ beats, where $R$ is the length of the phasor cycle in beats. Once the phasor reaches $2\pi$, it will automatically reset to zero.

Tempo changes in the performance will cause the regularity of beat times from the beat tracker to change. Therefore the frequency of the beat-synchronous phasor must be adjusted to maintain synchronisation with the beats. This involves setting the frequency of the phasor in such

a way that it will end its next cycle after $R$ beats.

At each beat, occurring at detection function sample $m$, the value of the phasor is $\theta(m)$. The target value of the phasor at the next beat, $\theta_{tar}$, is dependent upon the number of beats in a phasor cycle, $R$, and the index number of the next beat (between 1 and $R$) of the next beat in the cycle of the $R$ beats:

$$\theta_{tar} = (\frac{r+1}{R}) \cdot 2\pi \tag{6.3}$$

where $r = 0, ..., R-1$, the number of the current beat in the cycle of $R$ beats. For example, if we have a phasor that will repeat over 4 beats, and the next beat will be the 3rd beat, then the target value of the phasor at the next beat will be $\theta_{tar} = \frac{3}{4} \cdot 2\pi = \frac{3\pi}{2}$.

In order for the phasor to maintain synchronisation with the beat tracker, we must update the value of the phasor at each time step with an incremental value that will cause it to reach its target value, $\theta_{tar}$, at the next beat . At the $m$th audio frame, the value of the beat-synchronous phasor is calculated by:

$$\theta(m) = \theta(m-1) + \zeta \tag{6.4}$$

where $\theta(m-1)$ is the value of the beat-synchronous phasor at the previous time-step and $\zeta$ is an incremental value. This incremental value is calculated at each beat and is dependent on the current value of the phasor, the target value of the phasor at the next beat, $\theta_{tar}$, and the beat period, $\tau$, in detection function samples.

The incremental value is calculated differently depending upon whether the beat at which it is updated is one where we expect the phasor cycle to end and reset to zero – a 'reset beat' – or one where we do not expect the phasor to reset to zero – a 'non-reset beat'.

At non-reset beats, we simply calculate the incremental value, $\zeta$, by:

$$\zeta = \frac{\theta_{tar} - \theta(m)}{\tau}. \tag{6.5}$$

If we expect the beat-synchronous phasor to reset at the beat in question, then we must analyse the state of the phasor before deciding on the

incremental value. There are two possible scenarios.

Firstly, it is possible that the beat in question was earlier than expected and the phasor has not yet reset to zero – in which case we must adjust the incremental value so that the phasor increases at a rate that allows it to reach the end of its cycle and then reach its target value by the next beat:

$$\zeta = \frac{(\theta_{tar} - \theta(m)) + 2\pi}{\tau}. \tag{6.6}$$

If, however, the beat is later than expected and the phasor has already reset, then we calculate the value of the incremental value as in Equation 6.5.

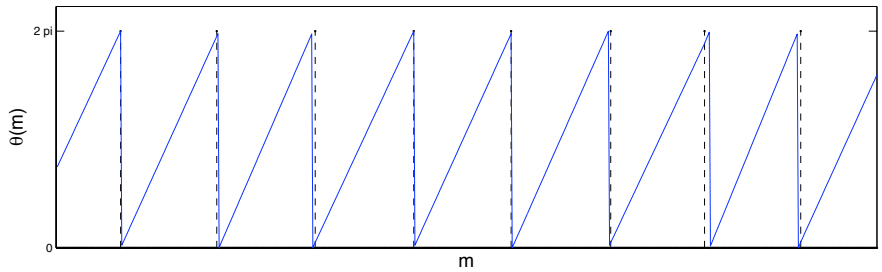
The beat-synchronous phasor uses a linear form of synchronisation. However there may be more complicated, non-linear equations that would provide better synchronisation. These will be investigated in future. The algorithm for the beat-synchronous phasor can be seen as pseudo-code in Algorithm 1. Example plots of the beat-synchronous phasor can be seen in Figures 6.1(a) and 6.1(b).

#### A Beat-Synchronous Low Frequency Oscillator

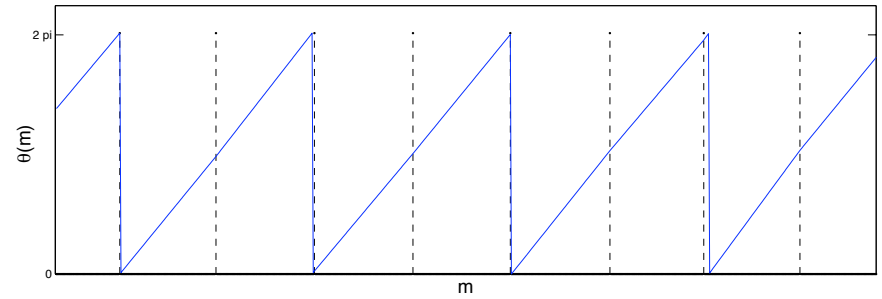
In order to create an oscillator for beat-synchronous audio effects, we use the beat-synchronous phasor, $\theta$, to create a beat-synchronous low frequency oscillator (LFO), $\Lambda$. This LFO will have a frequency related automatically to the beat period of the performance and be phase aligned with the beats. We allow the user to specify the frequency of the LFO as a number of *cycles per beat*, $\Omega$. This frequency is determined by the length of the beat-synchronous phasor cycle in beats $R$ and a frequency $\omega$:

$$\Omega = \frac{\omega}{R} \tag{6.7}$$

If we wish to have an oscillator at a rate of $\Omega \geq 1$ (one or more cycles per beat) then we set $\omega \geq 1$ and $R = 1$. However, if we want cycles that last for multiple beats so that there is less than one cycle per beat, i.e. $\Omega < 1$ then we set $\omega = 1$ and $R > 1$.




(a) A Beat-Synchronous Phasor with Cycle Lengths of $R = 1$




(b) A Beat-Synchronous Phasor with Cycle Lengths of $R = 2$

Figure 6.1: Beat-synchronous phasors. The phasor signal is the solid line and the beats are indicated by vertical dotted lines

We then calculate a beat-synchronous low frequency oscillator by taking some function of the beat-synchronous phasor, $\theta$. For example, a raised cosine beat-synchronous LFO:

$$\Lambda_{cos}(m) = \frac{\cos(\omega \cdot \theta(m)) + 1}{2} \tag{6.8}$$

Note that we add one and divide by two in order to set the range of the beat-synchronous LFO to [0,1]. Figures 6.2(a), 6.2(b) and 6.2(c) show three cosine beat-synchronous low frequency oscillators at different frequencies.

### A Beat-Synchronous Tremolo

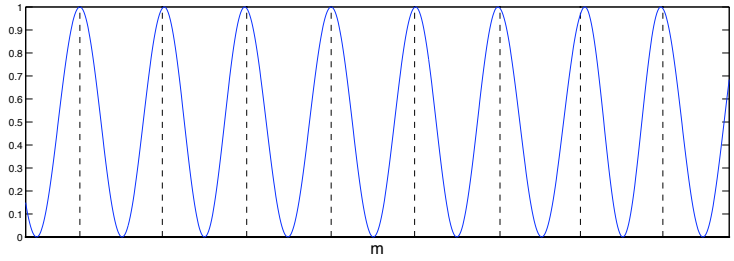
A tremolo effect is the modulation of the amplitude of an audio signal by a low frequency oscillator (LFO) [Zolzer, 2002]. We can create a beat-synchronous tremolo using a cosine beat-synchronous LFO:

**Algorithm 1** A Beat-Synchronous Phasor

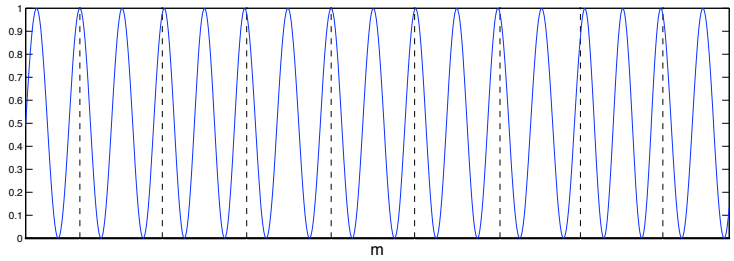
{Initialise variables...}
Given sampling frequency $f_s$, hop size $H$, initial tempo bpm = 120.
$r = 0$
$R$ = length of phasor in beats
$\tau = \text{round}((60/\text{bpm}) \cdot (f_s/H))$
$\zeta = \frac{2\pi}{R}/\tau$
$\theta(0) = 0$
resetflag = 0
{Main Algorithm}
**for** each detection function sample, $m$, starting from $m = 1$ **do**
  {Update phasor}
  $\theta(m) = \theta(m-1) + \zeta$
  **if** $\theta(m) > 2\pi$ **then**
    $\theta(m) = \theta(m) - 2\pi$
  **end if**
  {If there is a beat}
  **if** beat event $\gamma_b$ at sample $m$ **then**
    $\tau = \gamma_b - \gamma_{b-1}$ {get new beat period $\tau$}
    $r = r + 1$ {set current beat number}
    **if** $r == R$ **then**
      $r = 0$
    **end if**
    $\theta_{tar} = (\frac{r+1}{R}) \cdot 2\pi$ {get target value for $\theta$ at next beat}
    {if this is a beat where we expect the phasor to reset}
    **if** $r == 0$ **then**
      **if** resetflag $< 0$ **then**
        {beat is late, phasor has already reset to zero}
        $\zeta = \frac{\theta_{tar} - \theta(m)}{\tau}$
        resetflag = 0
      **else**
        {beat is early/on time, phasor has not yet reset to zero}
        $\zeta = \frac{(\theta_{tar} - \theta(m)) + 2\pi}{\tau}$
        resetflag = 1
      **end if**
    **else**
      $\zeta = \frac{\theta_{tar} - \theta(m)}{\tau}$
    **end if**
  **end if**
**end for**

$$y[n] = x[n] \cdot \Lambda_{cos}(m) \tag{6.9}$$

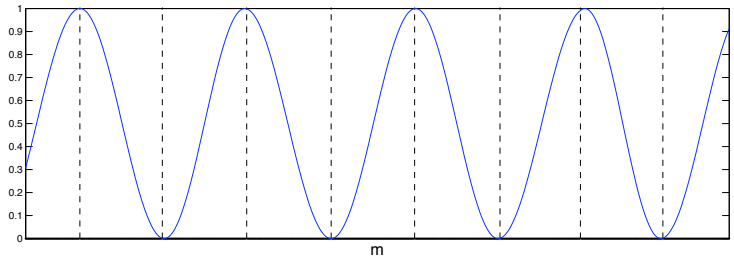
where $n$ is the index of each audio sample, $x[n]$ and $y[n]$ are the input and output signals and $\Lambda_{cos}(m)$ is the value of a cosine beat-synchronous LFO at detection function sample $m$. Our experience of using the effect




(a) A Beat-Synchronous LFO at $\Omega = 1$ cycle per beat, with $\Omega = \frac{\omega}{R}$ with $\omega = 1$ and $R = 1$.




(b) A Beat-Synchronous LFO at $\Omega = 2$ cycles per beat, with $\Omega = \frac{\omega}{R}$ with $\omega = 2$ and $R = 1$.




(c) A Beat-Synchronous LFO at $\Omega = 0.5$ cycles per beat, with $\Omega = \frac{\omega}{R}$ with $\omega = 1$ and $R = 2$

Figure 6.2: Beat-synchronous Low Frequency Oscillators using a cosine function. The solid line is the oscillator signal while the vertical dotted lines indicate the beats.

indicates that we would typically expect the frequency to be one or more cycles per beat, i.e. $\Omega = \frac{\omega}{R} \geq 1$ where $R = 1$ and $\omega \geq 1$.

### A Beat-Synchronous Flanger

A flanger is an audio effect whereby two identical audio signals are mixed together – with one signal delayed by a small and varying amount [Zolzer,

2002]. The effect results in a 'whooshing' sound passing through the signal. We can create a beat-synchronous flanger by using a cosine beat-synchronous LFO to modulate the length of a variable delay line of length $D$:

$$D(m) = \Lambda_{cos}(m) \cdot D_{max} \tag{6.10}$$

where $D_{max}$ is the maximum length of the delay line (usually around 2ms). The effect is then implemented as follows:

$$y[n] = x[n] + \alpha \cdot x[n - D(m)] \tag{6.11}$$

where we choose $\alpha = 0.7$. We would typically expect there to be one or more beats per cycle for this effect, i.e. $\Omega = \frac{\omega}{R} \leq 1$ where $R \geq 1$ and $\omega = 1$.

**A Beat-Synchronous Auto-Wah**

A 'wah' effect produces a colouring of the sound in such a way that it sounds like the human voice uttering the sound "wah-wah". This is achieved by moving a bandpass filter up and down in the frequency range where human vocal formants vary to change the sound of vowels [Zolzer, 2002]. In many cases, the cutoff frequency of the bandpass filter is controlled by a pedal. An 'auto-wah' effect controls the cutoff frequency of the bandpass filter using an oscillator, rather than a foot pedal. We can create a beat-synchronous auto-wah effect by using a beat-synchronous LFO to modulate the cutoff frequency $f_c$:

$$f_c(m) = f_{base} + (\Lambda_{cos}(m) \cdot f_{range}) \tag{6.12}$$

where $f_{base}$ and $f_{range}$ are respectively the base cutoff frequency and sweep range, in Hz, of the bandpass filter. We would expect typical usage of this effect to be with a beat-synchronous LFO operating at one or more cycles per beat, in a similar way to the beat-synchronous tremolo. Through experimentation we choose the values $f_{base} = 100$Hz and $f_{range} = 1000$Hz.

### 6.1.3 Implementation

The beat-synchronous audio effects are implemented in C++ and wrapped as externals for Max/MSP for use with real-time audio signals[2].

## 6.2 Synchronising Video to a Live Performance

Many have experimented with the projection of video to accompany live musical performances. For example there is the real-time sequencing and mixing of pre-stored video footage, synchronised to music, often referred to as *VJing* [Spinrad, 2005]. Information from audio processing such as audio splicing and onset detection has been used to directly manipulate video playback [Collins and Olofsson, 2006]. In other work Lew [2004] developed a musical instrument as an interface for controlling the live editing of video footage or *Live Cinema*. A detailed discussion of live audiovisuals has been presented by Alexander and Collins [2007].

As indicated by the rise of music video channels such as *MTV* since the 1980s, it is now commonplace for bands and musicians to produce videos to accompany some of their pieces of music. Indeed, many independent artists now produce videos for their music – or collaborate with filmmakers to do so. The projection of these videos during live performances is also becoming more commonplace.

However, there remains a problem. The video is likely to contain time sensitive editing related to events in the music. If a band projects their music video during a live performance – given that the live performance tempo will differ from that of the recording the video was edited to – how can the band ensure that the events in the video and the events in the music coincide?

A simple solution is for the band to play to a click track – played into the headphones of a key performer, such as the drummer. By playing at a computer defined tempo the band will be able to play to the exact tempo of the original performance, allowing the video to stay in time. Unfortunately, this option heavily detracts from the expressive nature of

[2] `http://www.eecs.qmul.ac.uk/~adams/software.html`

a live performance, forcing the musicians to stay in time with an inflexible computer clock.

To allow musicians to be able to perform in time with their music videos while also performing expressively, here we outline a solution that uses information about the beats in the performance, provided in real-time by a beat tracking system, to adjust the speed of the video to stay in time with a live performance.

In particular we are attempting to match the beats in a live performance – which will likely contain tempo variations at points – with beat associated video frames – which in turn are determined by a different performance which will contain different tempo variations from the live performance.

We now present a solution that, at each beat, examines the current frame of the video, the current tempo and the desired frame of the video and the next beat (according to a set of annotated beats) and sets the frame rate of the video accordingly so that synchronisation between the video and live performance is achieved.

We consider a video, at $F$ frames per second, with associated annotated beat times as video frame numbers, $A_y$, for the recorded audio to which the video was originally edited to. We begin to play the video from the first beat, $\gamma_0$, occurring at zero seconds, at a frame rate, $f_{rate}$, equal to the initial frames per second, i.e. $f_{rate} = F$.

Then, at each beat indicated by the beat tracker, we record the time in seconds of the beat, $\gamma_b$, and also the video frame in the video at which it has occurred, $V_b$. We then record the index of the nearest beat in the annotations, $g$:

$$g = \arg\min_y |V_b - A_y|. \tag{6.13}$$

We are then able to calculate the new frame rate of the video by dividing the number of video frames to the next beat by the inter-beat interval of the performance in seconds:

$$f_{rate} = \frac{A_{g+1} - V_b}{\gamma_b - \gamma_{b-1}}. \tag{6.14}$$

Figure 6.3: A performance using the automatic synchronisation of video (displayed on screens behind the performers) to a live performance. The author is on the right. Photograph: Chris Matthews.

When building such a system, we are making the assumption that the structure of the live performance is identical to that of the recorded audio for which the video was made.

In practice, adjusting the frame rate of the video at every beat (every 0.5 seconds at an example tempo of 120bpm) is unnecessary. Rather, our experience shows that the video frame rate needs to be adjusted only every few seconds to maintain synchronisation. Furthermore, as we identify the nearest beat in the annotations given each beat from the performance (in Equation 6.13), at faster tempi it only takes small beat timing errors (around 0.2 seconds at 160bpm) to cause the wrong beat to be identified. This makes the system vulnerable to small beat tracking errors.

For these reasons, we suggest updating the video at the bar level. This involves both annotating the video at the bar level and inputting only bar level beats to the system. If the piece is in a $\frac{6}{4}$ time signature, with 6 beats in a bar, for example, then we would input the first of every 6 beats. The time signature can be specified prior to the performance.

Figure 6.3 shows an image from a performance where this algorithm was used to synchronise video playback to a performance – although in

the case in question the performance beat times were entered by a human using a computer keyboard, rather than a beat tracking algorithm. This was due to the variability of beat tracking performance on different pieces of music and the practicality of providing audio to a projecting laptop in a small venue. However, we have successfully used automatic beat tracking in rehearsal to synchronise video to live performers.

#### Implementation

The video synchronisation technique is implemented in C++ using openFrameworks to control the rate of the video in response to beat events sent by our real-time beat tracker from Max/MSP.

## 6.3 Visual Displays Informed By Harmonic Analysis

Through our real-time chroma analysis algorithm (presented in section 3.2) we have a signal giving us information about the harmonic content of a performance. In this section we describe a technique for linking this harmonic information automatically to visual displays to create a responsive multimedia performance.

We can use the information from our chroma analysis algorithm to control the tonal balance of the red, green and blue in a video which can lead to interesting synchronisation between visual accompaniments and live performances. A mapping from a harmonic profile to visual effects is creative in nature and so here we demonstrate what should be considered a single example mapping.

The parameters passed to the video are the amount of red ($v_r$), green ($v_g$) and blue ($v_b$) expressed as values between 0 and 1. Given the 12 values in our chroma vector, $C$, produced by our chroma analysis technique in Chapter 3, we calculate the values of the three colour components in the video as follows:

$$v_r = \frac{\text{round}(C(0) + C(7))}{2} \tag{6.15}$$

$$v_g = \frac{\text{round}(C(2) + C(9))}{2} \tag{6.16}$$

(a) A video image with a dominant red colour tone due to harmonic content containing a large amount of the C and G pitch classes.

(b) A video image with a dominant green colour tone due to harmonic content containing a large amount of the D and A pitch classes.

(c) A video image with a dominant yellow colour tone due to harmonic content containing a large amount of the G and D pitch classes.

(d) A video image with a dominant blue colour tone due to harmonic content containing a large amount of the E and B pitch classes.

Figure 6.4: A video of a zoetrope affected by different colour balances determined by our mapping from the real-time chroma analysis algorithm to the proportions of red, green and blue in the video image.

$$v_b = \frac{\text{round}(C(4) + C(11))}{2} \tag{6.17}$$

This mapping means that the amount of red in the video is determined by

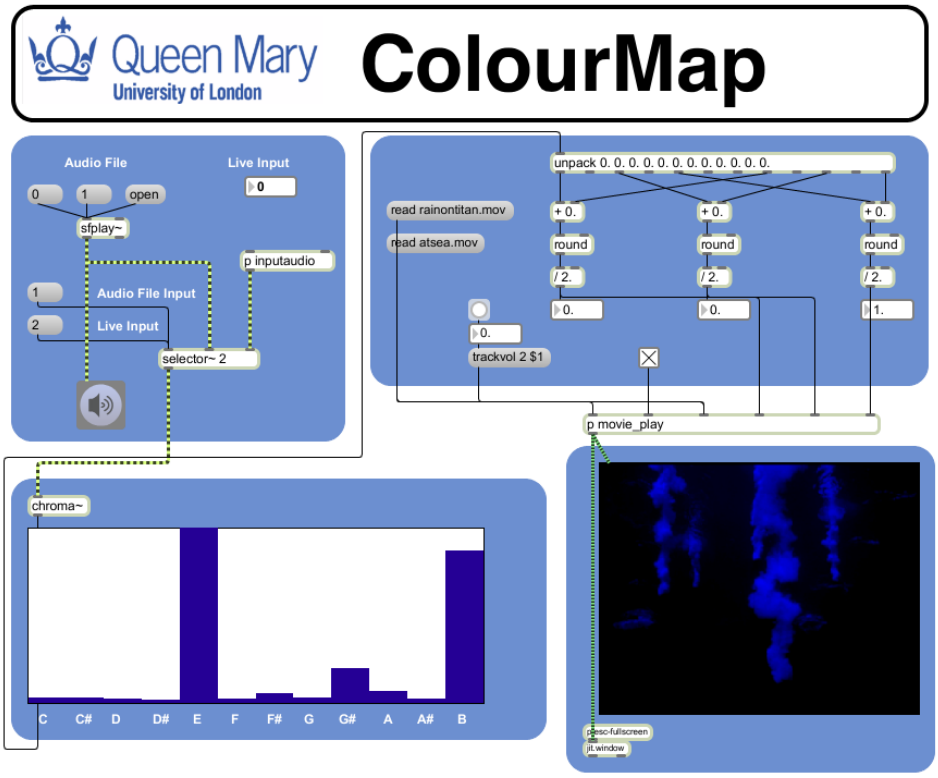



Figure 6.5: An implementation, in Max/MSP, of video display colour tone being affected by harmonic analysis.

the amount of C and G in the signal, the amount of green is determined by the amount of D and A in the signal and the amount of blue is determined by the amount of E and B in the signal. As these pairs of notes are separated by an interval of a 5th, many chords will contain both notes. This creates harmonic 'centres' linked to certain colour balances.

Figures 6.4(a) to 6.4(d) show four screenshots of a video of a zoetrope (an early spinning cylindrical device for animating images – the user looks through slits to see the animation) affected by four different harmonic profiles from a chroma vector.

### Implementation

The harmonic control of video colour tone is implemented in Max/MSP using our chroma analysis external, `chroma~`. The patch can be seen in Figure 6.5.

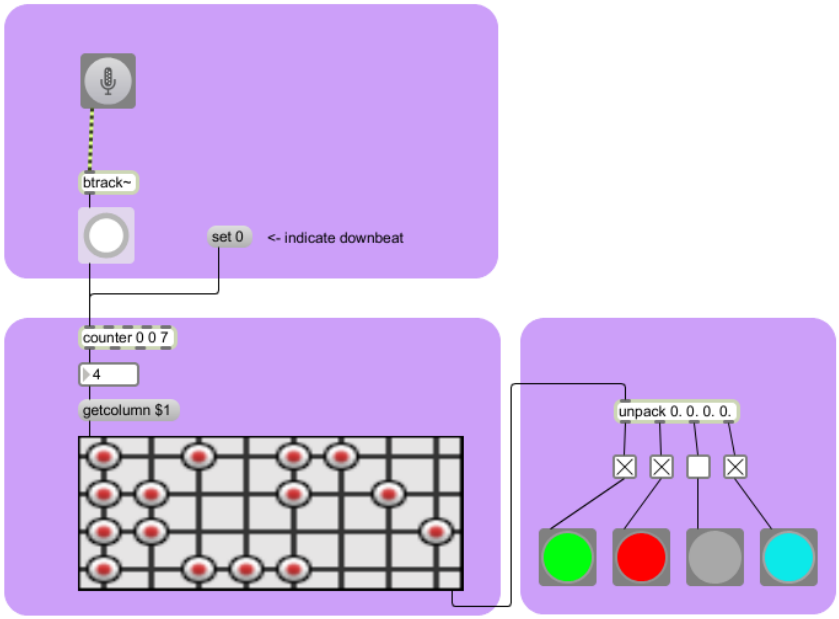



Figure 6.6: An implementation, in Max/MSP, of a beat-light matrix controlled by our real-time beat tracker, btrack.

## 6.4 Beat-Synchronous Lighting Patterns

It can be very visually effective for lighting changes to be synchronised with the beats of a performance. For controlling such lighting we present an interface that allows easy specification of patterns which are then automatically synchronised to the beat of the performance.

The interface we present is called a *beat-light matrix*, $M$. It is a $B \times L$ matrix where $B$ is the number of beats in the pattern (usually the number of beats in the bar) and $L$ is the number of lights. Each entry $M(i, l)$ is binary, indicating whether light $l$ should be on or off at beat $i$.

Our real-time beat tracker is then used to cycle through the $B$ columns of the matrix and each $L$-bit binary column is used to turn the $L$ lights on or off.

Figure 6.6 shows an example implemented in Max/MSP with an 8 beat cycle and 4 lights. In the top left hand panel we can see the audio input fed to our beat tracker *btrack*. The beat events from this beat tracker are then fed into the bottom left hand panel where they control a counter

running from 0 to 7. These values select the columns from the beat-light matrix which has a lighting pattern pre-programmed into it. Each 4-bit binary column is then sent to the third panel in the bottom right which determines whether the 4 lights are 'on' or 'off'. The image has been taken when the 5th column of the matrix (the counter says 4, as it starts at 0) and so three lights are on (green, red and blue) as indicated by the column of the matrix.

#### Implementation

The beat-synchronous patterned lighting is implemented in Max/MSP using our real-time beat tracking external, `btrack~`. Currently only simulated lights are controlled in the software – however, communications technologies for controlling lighting such as DMX could be used to control real lighting with minimal adaptations to the current implementation.

## 6.5 Automatic Accompaniment Using Performance Following

In Chapter 5 we presented our 'performance following' technique for predicting harmonic content in musical performances containing repetition. The prediction, at each beat, was of a $12 \times 1$ chroma vector indicating the predicted harmonic content of the beat in question.

Making use of this information, we implement an automatic bassline accompaniment by using the chord classification technique presented in section 3.3 to discover the root note of the harmonic vector. We then use the pitch class of the root note to generate a MIDI note which is sent to a synthesiser in Ableton Live.

The resulting combination of our performance following technique and the use of the root note information from our chord detector is that we have a simple automatic accompaniment system capable of identifying repeated musical patterns and playing a bassline, in real-time, with no prior knowledge.

This is currently a very literal bassline as our chord detection algorithm

does not detect inversions and no other tones are substituted in for the root note. In future work we will investigate using the full polyphonic nature of the predicted chroma vectors to inform a real-time generative music system capable of more interesting accompaniments than a simple bassline. This would give the system the degree of complexity (as defined in Chapter 2) necessary to produce a fully interactive music system.

## 6.6 Beat-Synchronous Sample Playback

In a live performance, due to variations in tempo, the use of recorded samples that have some form of 'beat' involves ensuring that the tempo and beat locations of the sample match those of the performance. In this section we present a method for automating this process.

The information provided by our real-time beat tracker allows us to inform existing technology of the nature of the performance in which it is involved. In this particular case we use information from our real-time beat tracker to inform live performance software *Ableton Live* of the performance tempo.

Ableton Live is equipped with an interface for identifying the beat locations of recorded sound files so that they can be effectively time stretched, in real-time, to be performed at any tempo. By providing information about the beat and tempo to Ableton Live we can automatically cause any sound sample to be played in time with a live performance.

Feeding a live audio signal into our beat tracker in Max/MSP, we then send each beat to Ableton Live as a MIDI message. Ableton Live receives each beat as a 'tap' for its tap-tempo interface which automatically adjusts the system metronome. In Ableton Live we can then trigger a sound file at any beat and it will play in time with the live performance, with no other instruction from a human. This process is depicted in Figure 6.7.

## 6.7 Summary

In this chapter we have demonstrated a number of ways in which real-time musical audio analysis algorithms – in particular those presented in

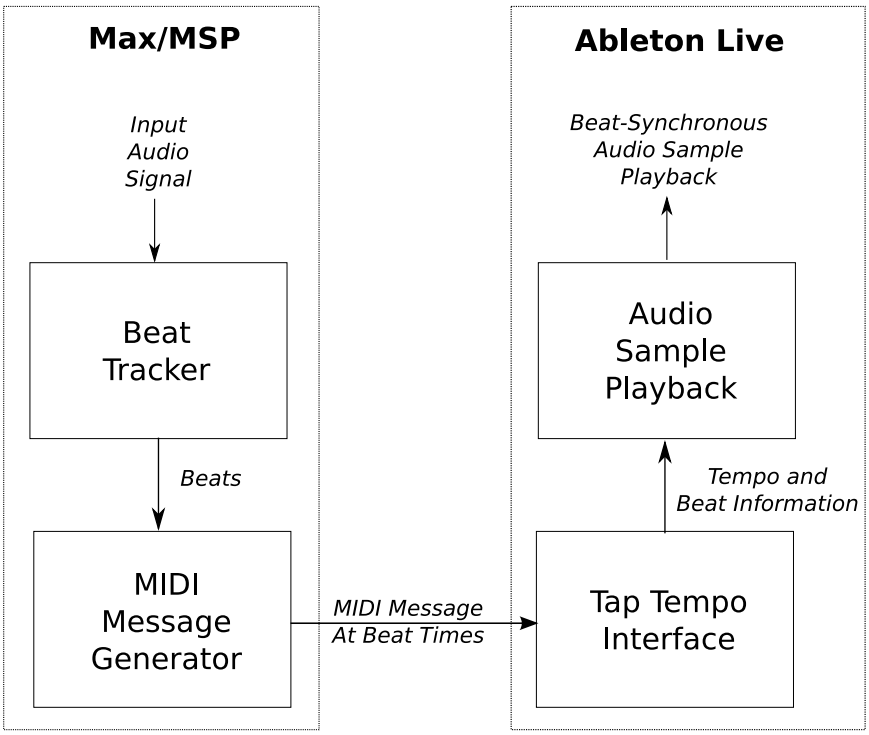



Figure 6.7: The use of a real-time beat tracker, in Max/MSP, to send beat information to Ableton Live, allowing beat-synchronous sample playback.

preceding chapters – can be used to create novel performance tools for musicians.

We have demonstrated that real-time beat tracking can be used to control the parameters of audio effects, to synchronise a video to the performance of a live band, to synchronise the tempo of audio samples to live performances and to create lighting patterns that automatically change in time to the beat. We have shown also that harmonic analysis can be used to control the colour tone of video displays. Finally we have shown that the predictions from our performance following technique can be used to generate a simple bass line accompaniment in real-time, with no prior knowledge in the form of a score.

# Chapter 7

# Conclusion

In this thesis, we have explored the use of musical audio analysis to inform applications in live musical performances. We have developed a number of musical audio analysis algorithms and explored ways of following temporal developments in musical performances – allowing live performance technologies to have a representation of their musical surroundings.

Specifically we have developed real-time algorithms for extracting beat and harmonic information, combined these to create a beat-synchronous harmonic representation, using the result in a 'performance following' technique for identifying repeated patterns in musical performances. We have then demonstrated the use of these analysis techniques in applications for live performance.

In Chapter 1 we outlined four specific objectives for this thesis. The first was the development of real-time musical audio analysis algorithms which we met with the beat tracking, chroma analysis and chord recognition techniques presented in Chapter 3. The second objective was the exploration of the combination of these musical audio analysis algorithms, which we explored in section 3.4, outlining techniques for calculating beat-synchronous harmonic representations. In Chapter 5, the use of these representations as an input for our performance following technique for predicting the harmonic content of repeated musical patterns met our third objective. Finally, meeting our fourth objective, we have outlined in Chapter 6 a number of live performance applications based upon these techniques for musical audio analysis that automate the control of audio effects, audio samples, video and lighting in real-time performance situations.

## 7.1 Thesis Contributions

We now summarise the main contributions of the work in this thesis.

#### A Real-Time Beat Tracker

In section 3.1 we presented a computationally efficient and robust beat tracker for real-time audio signals. Extending an existing offline model [Ellis, 2007] based upon dynamic programming, we have replaced its non-causal tempo estimation phase and method for determining beat locations with causal techniques.

We have shown that the resulting causal beat tracker outperforms the Scheirer [1998] model and is comparable in performance to the causal model proposed by Klapuri et al. [2006]. Our model, however, is considerably more computationally efficient than these approaches. Our real-time approach also outperforms two of the four non-causal models proposed – the Dixon [2007] and Ellis [2007] models – despite having no access to future information.

This beat tracker was implemented as an external for Max/MSP called '`btrack~`', enabling its use in live performance contexts.

#### A Real-Time Chord Recognition Technique

We presented, in sections 3.2 and 3.3, real-time chroma analysis and chord recognition techniques specifically focused on use in a live performance context. By mapping energy from spectral peaks to pitch classes, rather than larger spectral ranges, we presented a chroma analysis calculation technique suitable for real-time performance. This chroma analysis technique was the input to a chord classification technique that compares the chroma vector to a number of templates, masking out expected note positions and choosing the template that minimises residual energy.

We have shown that our approach performs comparably to other chroma analysis and chord recognition techniques when classifying audio frames of acoustic guitar signals as chords. However, our model is considerably more computationally efficient than the other techniques it

was compared to.

We have presented real-time implementations of these techniques as Max/MSP externals called '`chroma~`' and '`chorddetect~`' for practical use in musical performances.

**A Modular Beat Tracking Evaluation**

In Chapter 4 we conducted a modular study of five state of the art beat trackers. Splitting them into their input feature and tracking model we evaluated the resulting 20 combinations on two separate databases. We have examined the results with specific focus on the input features and tracking models outside of the context of their original combinations. We also studied the variations in their performance when applied to different genres, observing that different input features resulted in the best performance for different genres of music.

We compared, for each tracking model, the best score with a single input feature to the score with an 'oracle' input feature – where the best input feature is used for each audio file. The substantial increases in performance indicate that some automatic method for choosing the most appropriate input feature for a given audio signal would improve performance. We conducted similar studies using an 'oracle' tracking model and 'oracle' parameter settings for a given tracking model. Again, these studies implied that selection of appropriate models or parameters for different signals would improve performance.

Finally, we studied the result of applying some of these lessons to our real-time tracking model by choosing genre specific input features and parameter settings.

**A Performance Following Technique**

We presented a technique in Chapter 5 for predicting harmonic information in musical performances with no prior information in the form of a score. Taking as an input the real-time beat-synchronous harmonic representation outlined in Chapter 3, we compare the most recent few seconds

of a performance to the most recent few minutes. This process of contextualisation allows us to identify repetitions of musical patterns, from which we can make predictions of future harmonic content.

We have compared our technique to others in an objective evaluation. We were able to show that our technique was able to predict a majority of repeated content, and more than any of the techniques to which it was compared. Finally, based upon this technique we have presented a simple bassline following application.

**Live Performance Applications**

Making use of the real-time musical audio analysis techniques presented in earlier chapters we demonstrated several novel live performance applications in Chapter 6. By making use of information from our real-time beat tracking technique we have shown that we can automatically link audio effect parameters to the beat and tempo of live performances. We have demonstrated a tempo-synchronous delay and a beat-synchronous tremolo, auto-wah and flanger.

We have shown that real-time beat tracking information can be used to synchronise a video to a live musical performance by adjusting the frame rate of the video to match beats in the performance to annotated beats relating to the video. We have also shown that beat tracking can be used to synchronise both audio samples and lighting patterns to the beat and tempo of live performances.

Using our harmonic analysis technique we have shown that we can relate the colour tone of videos in live musical performances to the harmonic changes of musical instruments.

In addition to the contributions above, we believe that a key test of the strength of this research is the wider impact it has on others – in particular its practical use in performances and integration into applications. We now turn to the (known) use of this work by others, at the time of writing.

## 7.2 Wider Use of This Research

It is cause for confidence in the musical audio analysis tools presented in this thesis that they have been used by third parties for both real-time musical applications and in commercial software. Here we outline this third party use.

#### Project LSD

The Max/MSP implementations of the real-time beat tracking (`btrack~`) and chord recognition (`chorddetect~`) techniques presented in Chapter 3 of this thesis have been used as part of *Project LSD* – an "interactive music visualizer". The project was created as part of a bachelor group project at the Department of Media Technology at the University of Aalborg, Denmark [Hansen et al., 2010]. Martin Weiss Hansen, one of the four instigators, explains the project:

> A prototype of the interactive music visualizer, LSD (Listen, See & Dance), was developed. Beat tracking, chord detection and MFCC extraction are used as parameters to the creation of the music visualization. The users can interact with the system by moving their bodies, waving glowsticks and shouting or cheering.
>
> The prototype was tested at a party arranged by Art & Technology at Platform4, Aalborg. LSD was set up to accompany three DJs playing at the party. The experiment showed that the attendees were engaged when they interacted with the music visualization.

#### Exploration of Real-Time Rhythmic Automatic Accompaniment

The Max/MSP implementation of the real-time beat tracker (`btrack~`) has been used in a master's dissertation by Tim Canfer [2010] focused on an exploration of real-time rhythmic automatic accompaniment (which focused also on the drum tracker *B-Keeper*, presented by Robertson and Plumbley [2007]). In the dissertation, Canfer concludes:

> The real surprise of this project was the performance of the most recent beat trackers, which has been extremely encouraging for the prospects of a fully working system of real-time rhythmic automatic accompaniment as suggested in section 5.2. Prior to exploring B-Keeper and `btrack~`, there was a very real possibility that even the most up-to-date technology was just not sophisticated enough to be able to supply an accurate enough beat tracker for the system. In fact, the general impression that I have received from musicians, academics and laypeople alike was that I would quickly come down out of the science fiction fantasy idea when I started to look at the practical limitations of technology. While still not wanting to be unrealistic about the prospects of this real-time rhythmic automatic accompaniment system, it does look like there may be a real possibility of this being able to work on stage. [Canfer, 2010]

**Capo 2**

In the context of single audio files, our chord classification technique, presented in section 3.3, has been used as part of a commercial software package called *Capo 2* by a Canada-based company called *SuperMegaUltraGroovy*[1]. This software aids musicians by helping them to learn to play music in their collections by annotating them with information such as chord labels. In a personal communication, Christopher Liscio from SuperMegaUltraGroovy explains the use of our chord recognition technique in Capo 2:

> Capo uses this chord recognition technique as a basis for its Chord Marker feature. As the user listens to a song, they tap the 'k' key to drop a marker that contains a detected chord for that time period. If the detection is incorrect, the user may easily replace the chord by typing in simple strings such as 'bbmaj7' or 'eb9'.

[1] `http://capoapp.com/`

> When a song is loaded, the chromagram representation is calculated alongside the spectrogram display (which is based on a modified version of the Constant Q Transform). When the user drops a chord marker, the stored chromagram representation is used to determine the chord at the current time. This is achieved by averaging successive chromagram frames starting at the current time, and within a 300ms time window.

We believe that these accounts of the use of our musical audio analysis tools in practice demonstrate that their level of performance is such that they are practically useful as tools for musicians.

## 7.3 Future Work

We now outline some areas for potential future research that have arisen from the work in this thesis. We see the main areas as broadly the improvement of the musical audio analysis algorithms presented, the development of musical audio analysis algorithms to represent musical features other than beat and harmonic information, the development of more complex live performance applications and the communication of the potential of these techniques to the wider music community.

### 7.3.1 Improving Musical Audio Analysis Techniques

We have seen the development of a number of applications based upon real-time musical audio analysis. Due to the fact that they rely on the information from the analysis, these applications are only as good as the analysis techniques that underpin them. We have shown that the analysis techniques we have presented are as robust as state of the art techniques, while being comparatively computationally efficient. We have also shown in this chapter that others have been able to successfully use these techniques for practical applications. However, there is still room for improvement if they are to be truly reliable and robust in all situations.

### Beat Tracking

Due to the sensitivity of humans to timing differences between musical events [Hirsh, 1959] – and therefore the sensitivity to beat tracking errors – a clear area for future study is the improvement of real-time beat tracking.

Through our modular study of beat tracking algorithms, presented in Chapter 4, we found evidence that a 'one size fits all' approach to beat tracking – where a single input feature and set of parameters are used for all signal types – is limited in its approach (in agreement with earlier work by Collins [2006a]). Future work would involve discovering automatic ways to select appropriate input features and parameters, based upon some signal characteristics.

We also found in Chapter 4 that our real-time beat tracking model performed comparably to non-causal approaches for genres of music often characterised by a strong beat, such as Rock, Pop and Dance music. However, the difference between our real-time approach and the non-causal techniques was larger for genres such as Classical and Folk music that more often contain stylistic tempo variations. As a result, future work should address the performance of real-time models on these musical styles.

### Harmonic Analysis

While beat tracking errors can be noticed with very low tolerance, the errors in our harmonic analysis techniques, in particular the chroma analysis technique, are not quite as critical. Where there are small errors, we can measure the Euclidean distance between the correct and actual chroma vectors, and it is unlikely that the distance will be large if the signal is clean. We are faced with two small problems, however.

Firstly, while small errors are unlikely to produce a wildly different chroma vector to the correct one, they can lead to misclassifications for subsequent analysis such as chord recognition. As a result, future work may involve techniques to improve either the chroma analysis technique or to build tolerance for energy produced by harmonics into the chord classification technique.

The second problem we face is the calculation of chroma vectors from

larger ensembles. It has been noted that the presence of percussive instruments can introduce noise to the chromagram that may adversely affect any applications that make use of it [Harte and Sandler, 2005; Lee, 2006; Cho et al., 2010]. It would be beneficial to examine how pre- and post-filtering strategies used to mitigate errors in offline chord analysis [Cho et al., 2010] could be implemented in real-time.

Dealing with multiple instruments is another complication as some may be playing tones that are not part of the main chord. Mauch and Dixon [2008] have identified the need to model "non-chord notes" as well as those expected to be in a given chord, which would aid chord recognition in such situations. It could be claimed that by using a direct audio signal from a single instrument we can avoid this problem – however it would increase the practical use of chroma vector informed applications if we could implement a technique that could accurately represent harmonic information from larger ensembles and so this will be a topic of future work.

#### Performance Following

Our performance following technique currently predicts harmonic information in the presence of repetition. For it to be truly useful as an information source for automatic accompaniment systems, it must be built into a larger system capable also of making predictions of harmonic content in the absence of repetition.

It would also be useful to allow the prediction of harmonic content at a resolution finer than the level of the beat as many musical events will be missed by the use of 'one harmonic vector per beat' approaches. A beat-synchronous approach could be developed that segmented the signal at faster metrical levels.

### 7.3.2 Implementing Other Musical Feature Extraction Techniques

We have studied two forms of musical audio analysis – beat tracking and harmonic analysis. There is, of course, much more to music than these

two aspects, and so a clear topic for future work is the implementation of other musical audio analysis techniques – and the exploration of their potential use in applications.

Specifically, we suggest developing robust and efficient techniques for the real-time analysis of monophonic pitch, multi-pitch extraction, melody extraction and musical key analysis. Extending beat tracking, we believe that information about the downbeat and other metrical levels would be invaluable. Beyond this, rhythmic information could be examined. Finally, the incorporation of timbral information into applications would be of great interest.

### 7.3.3 Developing More Complex Live Performance Applications

The majority of the applications presented in this thesis are 'reactive', rather than 'interactive'. They all change their behaviour in response to a musical input, and some (such as the beat synchronous audio effects) have the property of mutual influence in that the change in behaviour is an audible effect to the human performer. However, it could be argued that applications such as the automatic bassline generator in Chapter 6 do not possess a high enough degree of complexity to be fully interactive systems. In future we will investigate more highly interactive music systems based upon the musical audio analysis techniques developed in this thesis.

### 7.3.4 Expanding Awareness of the Technology in the Wider Music Community

From my personal experience, it has been clear that while there are clearly many exciting potential creative situations that can be explored through the use of musical audio analysis in live performance, outside of the academic and computer music circles these things are not seen as possibilities. While some forms of beat tracking, for example, are available as part of commercial software such as *Traktor*[2], the intended use of the tool is with

[2]Traktor Pro 2: `http://www.native-instruments.com/en/products/dj/traktor-pro/` (Accessed 11/08/2011)

music at a fixed tempo and the software is primarily aimed at DJs, rather than musicians involved in live performances with varying tempi.

The limited use of more sophisticated musical audio analysis algorithms in live performance is because either musicians do not know that this kind of technology exists, or if they do, it is so wrapped up in technical terms and expert jargon that they have no way to access it. As a result, I will be embarking upon a six month project to raise awareness and access to these ideas in the wider music community. This project is discussed in the next section.

## 7.4 Musical Interaction In The Community

Following on from the work in this thesis, I will be conducting a musical outreach project entitled 'Musical Interaction In The Community'. The project will be comprised of four stages.

The first will see the development of a musical audio analysis and applications library for live performance. This will involve the implementation of real-time musical audio analysis techniques – mainly refinements of those already completed for this thesis – for Max/MSP and SuperCollider. This will be easily achievable as all code is in the form of C++ classes which can be incorporated into different software environments with only a few lines of code. Then a number of musical applications will be designed as patches for Max for Live, Max/MSP and SuperCollider – incorporating openFrameworks for visual applications involving videos.

The second stage will involve the development of an attractive, clear and user friendly website. As well as hosting the externals, objects, patches and open source code, this website will also host tutorials – in video and text form – explaining in non-expert terms what each component or application does, how to make use of it and how to set it up in a live performance situation. The website will also contain clear, jargonless documentation on each component, downloadable in pdf format. Finally, for maximum potential communication, the website will be fully integrated with social media such as Facebook and Twitter, with a blog explaining the latest developments.

The third stage will be an 'outreach' phase, with myself and others from Queen Mary working 'in situ' with bands, musicians and installation artists to help them use this research in their musical projects. This will have the benefit for them of face-to-face explanation of the technology – and, for us, of feedback on technological implementations and the potential for new ideas for applications. In order to increase participation from non-London-based artists, during the project we will make several trips to elsewhere in the UK.

Further events will involve a number of workshops at Queen Mary, talks at music performance schools, outreach events in secondary schools and articles written for non-expert publications and magazines. Finally, a series of concerts will be organised to showcase the work of musicians who have been involved in the project.

The final stage will involve writing publications in the form of journal and conference papers on the experience of communicating the research to non-expert musicians and on any new applications developed during discussions with musicians.

## 7.5 Perspectives

In this thesis we have explored a potential shift in the way humans interact with technology in live performances. We have explored the idea that machines can make independent analyses of their musical surroundings and use that information to inform choices in musical and multimedia applications.

The eventual success of this approach will be determined, like many ideas, by its longevity and breadth of uptake amongst musicians and performers. We believe that the external use of the analysis techniques so far has been promising with positive responses on their reliability and level of performance.

We have seen in this thesis that the use of multiple real-time musical audio analysis techniques in performance situations is possible, practical and has the potential to inform a number of new creative applications for live performance. If there is the will amongst musicians we believe that

analysis techniques and applications such as these could become commonplace in musical performances in the future.

# Bibliography


Abdallah, S. and Plumbley, M. D. (2009). Information dynamics: Patterns of expectation and surprise in the perception of music. *Connection Science*, 21(2):89–117.

Alexander, A. and Collins, N. (2007). Live audiovisuals. In Collins, N. and d'Escriván, J., editors, *The Cambridge companion to electronic music*. Cambridge University Press.

Allauzen, C., Crochemore, M., and Raffinot, M. (1999). Factor oracle: A new structure for pattern matching. In Pavelka, J., Tel, G., and Bartosek, M., editors, *Proceedings of SOFSEM'99, Theory and Practice of Informatics*, Lecture Notes in Computer Science, pages 291–306.

Allen, P. E. and Dannenberg, R. B. (1990). Tracking musical beats in real-time. In *Proceedings of International Computer Music Conference*, pages 140–143.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402.

Ames, C. (1987). Automated composition in reterospect: 1956-1986. *Leonardo*, 20(2):169–185.

Anderson, J. R. (2000). *Learning and Memory: An Integrated Approach, 2nd Edition*. Wiley.

Assayag, G., Bloch, G., and Chemillier, M. (2006). OMAX-OFON. In *Sound and Music Computing Conference*.

Assayag, G. and Dubnov, S. (2004). Using factor oracles for machine improvisation. *Soft Computing*, 8(9):604–610.

Baird, B., Blevins, D., and Zahler, N. (1993). Artificial intelligence and music: Implementing an interactive computer performer. *Computer Music Journal*, 17(2):73–79.

Bello, J. P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., and Sandler, M. B. (2005). A tutorial on onset detection in music signals. *IEEE Transactions on Audio, Speech and Language Processing*, 13(5):1035–1047.

Bello, J. P., Duxbury, C., Davies, M., and Sandler, M. B. (2004). On the use of phase and energy for musical onset detection in the complex domain. *IEEE Signal Processing Letters*, 11(6):553–556.

Bello, J. P. and Pickens, J. (2005). A robust mid-level representation for harmonic content in music signals. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, pages 304–311, London, UK.

Biles, J. A. (1994). Genjam: A genetic algorithm for generating jazz solos. In *Proceedings of International Computer Music Conference*, pages 131–137.

Bilmes, J. A. (1993). Timing is of the essence: Perceptual and computational techniques for representing, learning and reproducing expressive timing in percussive rhythm. Master's thesis, Massachusetts Institute of Technology.

Blackwell, T. and Young, M. (2004). Swarm granulator. In *EvoWorkshops*. Springer Verlag.

Bloch, G., Dubnov, S., and Assayag, G. (2008). Introducing video features and spectral descriptors into the omax improvisation system. In *Proceedings of International Computer Music Conference*.

Bloch, J. J. and Dannenberg, R. B. (1985). Real-time computer accompaniment of keyboard performances. In *Proceedings of International Computer Music Conference*, pages 279–289.

Boden, M. A. (2004). *The creative mind: myths and mechanisms*. Routledge.

Bongers, B. (1999). Exploring novel ways of interaction in musical performance. In *Proceedings of the 3rd Conference on Creativity and Cognition*.

Braun, H.-J., editor (2002). *Music and technology in the twentieth century*. John Hopkins University Press.

Brown, J. C. (1991). Calculation of a constant-q spectral transform. *Journal of the Acoustical Society of America*, 89:425–434.

Canfer, T. (2010). An exploration of real-time rhythmic automatic accompaniment. Master's thesis, School of Music, University of Leeds.

Casey, M. A., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., and Slaney, M. (2008). Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696.

Cemgil, A. T., Kappen, B., Desain, P., and Honing, H. (2001). On tempo tracking: Tempogram representation and kalman filtering. *Journal of New Music Research*, 28(4):259–273.

Chadabe, J. (1984). Interactive composing: An overview. *Computer Music Journal*, 8(1).

Cho, T., Weiss, R. J., and Bello, J. P. (2010). Exploring common variations in state of the art chord recognition systems. In *Proceedings of Sound and Music Computing Conference*.

Ciufo, T. (2003). Design concepts and control strategies for interactive improvisational music systems. In *Proceedings of MAXIS International Festival/Symposium of Sound and Experimental Music*, Leeds, UK.

Collins, N. (2005). DrumTrack: Beat induction from an acoustic drum kit with synchronised scheduling. In *Proceedings of International Computer Music Conference*.

Collins, N. (2006a). Towards a style-specific basis for computational beat tracking. In *Proceedings of International Conference on Music Perception and Cognition*.

Collins, N. (2006b). *Towards Autonomous Agents for Live Computer Music: Realtime Machine Listening and Interactive Music Systems*. PhD thesis, University of Cambridge.

Collins, N. (2007). Musical robots and listening machines. In Collins, N. and d'Escriván, J., editors, *The Cambridge companion to electronic music*. Cambridge University Press.

Collins, N., McLean, A., Rohrhuber, J., and Ward, A. (2003). Live coding in laptop performance. *Organised Sound*, 8(3):321–330.

Collins, N. and Olofsson, F. (2006). klipp av: Live algorithmic splicing and audiovisual event capture. *Computer Music Journal*, 30(2):8–18.

Colton, S. (2008). Creativity versus the perception of creativity in computational systems. In *Proceedings of the AAAI Spring Symposium on Creative Systems*.

Colton, S. (2009). The painting fool - http://www.thepaintingfool.com/.

Conklin, D. (2003). Music generation from statistical models. In *AISB 2003 Symposium on Artificial Intelli- gence and Creativity in the Arts and Sciences*, pages 30–35, Aberystwyth, Wales.

Cont, A. (2010). A coupled duration-focused architecture for realtime music to score alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6).

Cope, D. (1996). *Experiments in Musical Intelligence*. A-R Editions.

Cremer, M. and Derboven, C. (2004). A system for harmonic ananlysis of polyphonic music. In *Proceedings of the AES 25th International Conference*, pages 115–120, London, UK.

Dannenberg, R. B. (1984). An on-line algorithm for real-time accompaniment. In *International Computer Music Conference*, pages 193–198.

Dannenberg, R. B. (2005). Toward automated holistic beat tracking, music analysis, and understanding. In *Proceedings of International Conference on Music Information Retrieval*, pages 366–373.

Dannenberg, R. B. and Hu, N. (2002). Pattern discovery techniques for music audio. In *Proceedings of International Conference on Music Information Retrieval*, pages 63–70.

Dannenberg, R. B. and Hu, N. (2003). Polyphonic audio matching for score following and intelligent audio editors. In *Proceedings of International Computer Music Conference*, pages 27–33.

Davies, M. E., Degara, N., and Plumbley, M. D. (2011). Measuring the performance of beat tracking algorithms using a beat error histogram. *IEEE Signal Processing Letters*, 18(3):157–160.

Davies, M. E. P. and Plumbley, M. D. (2007). Context-dependent beat tracking of musical audio. *IEEE Transactions on Audio, Speech and Language Processing*, 15(3):1009–1020.

Degara, N., Rúa, E. A., Pena, A., Torres-Guijarro, S., Davies, M. E., and Plumbley, M. D. (2011). Reliability-informed beat tracking of musical audio signals. *IEEE Transactions on Audio, Speech and Language Processing (to appear)*.

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

Dixon, S. (2005). Live tracking of musical performances using on-line time warping. In *Proceedings of International Conference on Digital Audio Effects*, pages 92–97.

Dixon, S. (2006a). Beat tracking with beatroot. In *Proceedings of MIREX Annual Music Information Retrieval eXchange*.

Dixon, S. (2006b). Onset detection revisited. In *Proceedings of International Conference on Digital Audio Effects*, pages 133–137.

Dixon, S. (2007). Evaluation of the audio beat tracking system beatroot. *Journal of New Music Research*, 36(1):39–50.

Dixon, S. and Widmer, G. (2005). MATCH: A music alignment tool chest. In *Proceedings of International Conference on Music Information Retrieval*, pages 492–497.

Dobrian, C. (2004). Strategies for continuous pitch and amplitude tracking in realtime interactive improvisation software. In *Sound and Music Computing Conference*, Paris, France.

Dubnov, S. (2008). Unified view of prediction and repetition structure in audio signals with application to interest point detection. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):327–337.

Dubnov, S., Assayag, G., and Cont, A. (2007). Audio oracle: A new algorithm for fast learning of audio structures. In *Proceedings of International Computer Music Conference*.

Ellis, D. P. W. (2007). Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60.

Everett, W. (2009). *The foundations of rock: from "Blue suede shoes" to "Suite : Judy blue eyes"*. Oxford University Press.

Ferraro, P. and Hanna, P. (2007). Optimizations of local edition for evaluating similarity between monophonic musical sequences. In *Proceedings of the 8th International Conference on Information Retrieval (RIAO)*.

Flexer, A. (2006). Statistical evaluation of music information retrieval experiments. *Journal of New Music Research*, 35(2):113–120.

Frith, S., Straw, W., and Street, J. (2001). *The Cambridge companion to pop and rock*. Cambridge University Press.

Fujishima, T. (1999). Real-time chord recognition of musical sound: A system using common lisp music. In *Proceedings of International Computer Music Conference*, pages 464–467.

Gervás, P. (2002). Exploring quantitative evaluations of the creativity of automatic poets. In *Second workshop on creative systems, approaches to creativity in Artificial Intelligence and Cognitive Science (ECAI).*

Gómez, E. (2006). *Tonal Description of Music Audio Signals.* PhD thesis, Universitat Pompeu Fabra.

Goto, M. (2001). An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30:159–171.

Goto, M. and Muraoka, Y. (1997). Issues in evaluating beat tracking systems. In *Working Notes of the IJCAI-97 Workshop on Issues in AI and Music*, pages 9–16.

Gouyon, F. (2005). *A computational approach to rhythm description: Audio features for the computation of rhythm periodicity functions and their use in tempo induction and music content processing.* PhD thesis, Universitat Pompeu Fabra.

Gouyon, F. and Dixon, S. (2005). A review of automatic rhythm description systems. *Computer Music Journal*, 29(1):34–54.

Gouyon, F., Dixon, S., and Widmer, G. (2007). Evaluating low-level features for beat classification and tracking. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume IV, pages 1309–1312.

Gouyon, F., Klapuri, A., Dixon, S., Alonso, M., Tzanetakis, G., Uhle, C., and Cano, P. (2006). An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Speech and Audio Processing*, 14(5).

Gusfield, D. (1997). *Algorithms on strings, trees, and sequences: computer science and computational biology.* Cambridge University Press.

Hainsworth, S. (2004). *Techniques for the Automated Analysis of Musical Audio*. PhD thesis, University of Cambridge.

Hainsworth, S. (2006). Beat tracking and musical metre analysis. In Klapuri, A. and Davy, M., editors, *Signal processing methods for music transcription*. Springer.

Hansen, M. W., Doslo, I., Askholm, J., Nguyen, K. T., and Christensen, M. G. (2010). L.S.D - an interactive music visualizer. Bachelor Project By Group 674.

Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete fourier transform the use of windows for harmonic analysis with the discrete fourier transform. In *Proceedings of the IEEE*, volume 66, pages 51–83.

Harte, C. A. and Sandler, M. B. (2005). Automatic chord identification using a quantised chromagram. In *Proceedings of AES 118th Convention*, number 6412, Barcelona.

Harte, C. A., Sandler, M. B., and Gasser, M. (2006). Detecting harmonic change in musical audio. In *Proceedings of Audio and Music Computing for Multimedia Workshop*, pages 21–26.

Hirsh, I. J. (1959). Auditory perception of temporal order. *Journal of the Acoustical Society of America*, 31(6).

Hsu, W. (2005). Using timbre in a computer-based improvisation system. In *Proceedings of International Computer Music Conference*.

Hu, N., Dannenberg, R. B., and Tzanetakis, G. (2003). Polyphonic audio matching and alignment for music retrieval. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 185–188.

Jordà, S. (2005). *Digital Lutherie: Crafting musical computers for new musics' performance and improvisation*. PhD thesis, Universitat Pompeu Fabra.

Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall.

Khadkevich, M. and Omologo, M. (2009). Use of hidden markov models and factored language models for automatic chord recognition. In *Proceedings of International Conference on Music Information Retrieval*, pages 561–566.

Kilian, J. and Hoos, H. H. (2004). MusicBLAST - gapped sequence alignment for MIR. In *Proceedings of International Conference on Music Information Retrieval*, pages 38–41.

Klapuri, A., Eronen, A. J., and Astola, J. T. (2006). Analysis of the meter of acoustic musical signals. *IEEE Transactions on Audio, Speech and Language Processing*, 14(1):342–255.

Kohonen, T. (1989). A self-learning musical grammar, or 'associative memory of the second kind'. In *Proceedings of International Joint Conference on Neural Networks*, volume 1, pages 1–5.

Large, E. W. (1995). Beat tracking with a nonlinear oscillator. In *Working Working Notes of the IJCAI-95 Workshop on Issues in Artificial Intelligence and Music*, pages 24–31.

Laroche, J. (2003). Efficient tempo and beat tracking in audio recordings. *Journal of the Audio Engineering Society*, 51(4).

Lee, K. (2006). Automatic chord recognition using enhanced pitch class profile. In *Proceedings of International Computer Music Conference*, pages 306–313.

Lee, K. and Slaney, M. (2008). Acoustic chord transcription and key extraction from audio using key-dependent hmms trained on synthesized audio. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):291–301.

Lerdahl, F. and Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. MIT Press.

Lew, M. (2004). Live cinema: Designing an instrument for cinema editing as a live performance. In *Proceedings of New Interfaces of Musical Expression*.

Lewis, G. E. (2000). Too many notes: Computers, complexity and culture in voyager. *Leonardo Music Journal*, 10:33–39.

Mauch, M. and Dixon, S. (2008). A discrete mixture model for chord labelling. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, pages 45–50.

Mauch, M. and Dixon, S. (2010). MIREX 2010: Chord detection using a dynamic bayesian network. In *Proceedings of Music Information Retrieval Evaluation eXchange (MIREX)*.

McKinney, M. F., Moelants, D., Davies, M. E., and Klapuri, A. (2007). Evaluation of audio beat tracking and music tempo extraction algorithms. *Journal of New Music Research*, 36(1):1–16.

Meredith, D., Lemström, K., and Wiggins, G. A. (2003). Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. In *Cambridge Music Processing Colloquium*.

Miranda, E. R. and Biles, J. A., editors (2007). *Evolutionary computer music*. Springer.

Mongeau, M. and Sankoff, D. (1990). Comparison of musical sequences. *Computers and the Humanities*, 24(3):161–175.

Moorefield, V. (2005). *The producer as composer: shaping the sounds of popular music*. MIT Press.

Müller, M., Ewert, S., and Kreuzer, S. (2009). Making chroma features more robust to timbre changes. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 1877–1880.

Nawab, S. H., Ayyash, S. A., and Wotiz, R. (2001). Identification of musical chords using constant-q spectra. In *International Conference on Acoustics, Speech and Signal Processing*, volume 5, pages 3373–3376.

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two protiens. *Journal of Molecular Biology*, 48(3):443–453.

Nierhaus, G. (2009). *Algorithmic composition: paradigms of automated music generation*. Springer.

Ockelford, A. (2005). *Repetition in Music: Theoretical and Metatheoretical Perspectives*. Ashgate Publishing Limited.

Orio, N., Lemouton, S., and Schwarz, D. (2003). Score following: State of the art and new developments. In *New Interfaces for Musical Expression*.

Oudre, L., Grenier, Y., and Févotte, C. (2009). Template-based chord recognition: Influence of the chord types. In *Proceedings of International Conference on Music Information Retrieval (ISMIR)*, pages 153–158.

Pachet, F. (2002). The Continuator: Musical interaction with style. In *Proceedings of International Computer Music Conference*, pages 211–218.

Papadopoulos, H. and Peeters, G. (2007). Large-scale study of chord estimation algorithms based on chroma representation and hmm. In *Proceedings of 5th International Conference on Content-Based Multimedia Indexing*, pages 53–60.

Pardo, B. and Birmingham, W. P. (2001). Following a musical performance from a partially specified score. In *Proceedings of the 2001 Multimedia Technology and Applications Conference*, pages 202–207.

Pearce, M. T., Müllensiefen, D., and Wiggins, G. A. (2008). A comparison of statistical and rule-based models of melodic segmentation. In *Proceedings of the Ninth International Conference on Music Information Retrieval*, pages 89–94.

Pearce, M. T., Müllensiefen, D., and Wiggins, G. A. (2010). The role of expectation and probabilistic learning in auditory boundary perception: A model comparison. *Perception*, 39(10):1365–1389.

Peeters, G. (2006). Chroma-based estimation of musical key from audio-signal analysis. In *Proceedings of International Conference on Music Information Retrieval*.

Prendergast, M. (2000). *The Ambient Century: from Mahler to Moby - the evolution of sound in the electronic age*. Bloomsbury Publishing.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Raphael, C. (1999). Automatic segmentation of acoustic musical signals using hidden markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):360–370.

Raphael, C. (2001). Music Plus One: A system for flexible and expressive musical accompaniment. In *International Computer Music Conference*, Havana, Cuba.

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

Roads, C. (2004). *Microsound*. MIT Press.

Robertson, A. and Plumbley, M. D. (2007). B-Keeper: A beat-tracker for live performance. In *New Interfaces for Musical Expression*, pages 234–237.

Robine, M., Hanna, P., and Ferraro, P. (2007). Music similarity: Improvements of edit-based algorithms by considering music theory. In *Proceedings of the ACM SIGMM International Workshop on Multimedia Information Retrieval*, pages 135–141.

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

Rowe, R. (1992). Machine listening and composing with cypher. *Computer Music Journal*, 16(1):43–63.

Rowe, R. (1993). *Interactive Music Systems*. The MIT Press, Cambridge, MA, USA.

Rowe, R. (2001). *Machine Musicianship*. MIT Press.

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

Sheh, A. and Ellis, D. P. W. (2003). Chord segmentation and recognition using em-trained hidden markov models. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, pages 183–189.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197.

Spiegel, L. (1992). Performing with active instruments - an alternative to a standard taxonomy for electronic and computer instruments. *Computer Music Journal*, 16(3):5–6.

Spinrad, P. (2005). *The VJ Book*. Feral House.

Stark, A. M., Davies, M. E., and Plumbley, M. D. (2007a). Audio effects for real-time performance using beat tracking. In *Proceedings of 122nd Convention of Audio Engineering Society*.

Stark, A. M., Davies, M. E., and Plumbley, M. D. (2007b). Real-time beat-synchronous audio effects. In *New Interfaces for Musical Expression*.

Stark, A. M., Davies, M. E., and Plumbley, M. D. (2008). Rhythmic analysis for real-time audio effects. In *Proceedings of International Computer Music Conference*, Belfast.

Stark, A. M., Davies, M. E., and Plumbley, M. D. (2009). Real-time beat-synchronous analysis of musical audio. In *Proceedings of International Conference on Digital Audio Effects*, pages 299–304.

Stark, A. M. and Plumbley, M. D. (2009). Real-time chord recognition for live performance. In *Proceedings of International Computer Music Conference*.

Stark, A. M. and Plumbley, M. D. (2010). Performance following: Tracking a performance without a score. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 2482–2485.

Stark, A. M. and Plumbley, M. D. (2011). Performance following: Real-time prediction of musical sequences without a score. *IEEE Transactions on Audio, Speech and Language Processing*, (accepted for publication).

Stuart, C. (2003). The object of performance: Aural performativity in contemporary laptop music. In *Proceedings of 5th International Digital Arts and Culture Conference*.

Thom, B. (1995). Predicting chordal transitions in jazz: the good, the bad, and the ugly. In *Proceedings of IJCAI Workshop on AI and Music, International Joint Conference on Artificial Intelligence*.

Vercoe, B. (1984). The synthetic performer in the context of live performance. In *International Computer Music Conference*, pages 199–200, Paris, France.

Young, M. (2009). Creative computers, improvisation and intimacy. In Boden, M., D'Inverno, M., and McCormack, J., editors, *Dagstuhl Seminar Proceedings 09291, Computational Creativity: An Interdisciplinary Approach*, number 09291 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

Zadel, M. and Scavone, G. (2006). Laptop performance: Techniques, tools, and a new interface design. In *Proceedings of International Computer Music Conference*, pages 643–648.

Zhang, X. and Gerhard, D. (2008). Chord recognition using instrument voicing constraints. In *Proceedings of International Conference on Music Information Retrieval*, pages 33–38.

Zolzer, U. (2002). *DAFX: Digital Audio Effects*. John Wiley and Sons.