Towards automatic extraction of harmony information from music signals

Harte, Christopher

Towards Automatic Extraction of Harmony Information from Music Signals

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

Christopher Harte

August 2010

Department of Electronic Engineering,
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, Professor Mark Sandler.

Abstract

In this thesis we address the subject of automatic extraction of harmony information from audio recordings. We focus on chord symbol recognition and methods for evaluating algorithms designed to perform that task.

We present a novel six-dimensional model for equal tempered pitch space based on concepts from neo-Riemannian music theory. This model is employed as the basis of a harmonic change detection function which we use to improve the performance of a chord recognition algorithm.

We develop a machine readable text syntax for chord symbols and present a hand labelled chord transcription collection of 180 Beatles songs annotated using this syntax. This collection has been made publicly available and is already widely used for evaluation purposes in the research community. We also introduce methods for comparing chord symbols which we subsequently use for analysing the statistics of the transcription collection. To ensure that researchers are able to use our transcriptions with confidence, we demonstrate a novel alignment algorithm based on simple audio fingerprints that allows local copies of the Beatles audio files to be accurately aligned to our transcriptions automatically.

Evaluation methods for chord symbol recall and segmentation measures are discussed in detail and we use our chord comparison techniques as the basis for a novel dictionary-based chord symbol recall calculation.

At the end of the thesis, we evaluate the performance of fifteen chord recognition algorithms (three of our own and twelve entrants to the 2009 MIREX chord detection evaluation) on the Beatles collection. Results are presented for several different evaluation measures using a range of evaluation parameters. The algorithms are compared with each other in terms of performance but we also pay special attention to analysing and discussing the benefits and drawbacks of the different evaluation methods that are used.

In memory of Stefan Demuth

Acknowledgements

I am very glad to finally have the opportunity of thanking all the people who have helped me to produce this thesis. It's taken a long time to finish it so the list has become very long...

I would like to start by thanking my supervisor Mark Sandler for his guidance and support throughout the last seven years. His continued confidence in me and my work has helped me immensely. I would also like to thank Mike Davies, Mark Plumbley and Juan Bello for their helpful advice and enthusiasm, particularly in my first few years as a Ph.D student.

My special thanks go to Matthew Davies for always being available to talk over ideas and particularly for his support while I was in the final stages of writing up. Without his help, I am quite sure I would never have completed this thesis.

I am greatly indebted to Laurie Cuthbert, Yue Chen and Andy Watson for reorganising my teaching commitments and extending work deadlines giving me the space to complete my write up during term time. I would also like to thank my slightly neglected year 3 and year 4 students for their understanding this term.

I am especially grateful to the following people for sitting through the listening tests that enabled me to verify my Beatles chord transcriptions. I hope that their ears will one day forgive me: Amir Dotan, Jonny Amer, Matthew Davies, Katy Noland, Ramona Behravan, Andrew Nesbit, Peyman Heydarian, Dawn Black, Chris Cannam, Kurt Jacobson, Yves Raimond, Mark Levy, Andrew Robertson, Chris Sutton, Enrique Peres Gonzales, George Fazekas, Maria Jafari, Beiming Wang, Emanuel Vincent and Becky Stewart.

I am grateful to my friends and research colleagues at UPF Barcelona and OFAI in Vienna for being so welcoming and fun to work with. Thank

you to Emilia Gómez, David Garcia, Fabien Gouyon, Elias Pampalk, Martin Gasser, Simon Dixon, Werner Goebel, David Stevens and Margit Demuth.

Thanks to Jan Weil, Craig Sapp and Johan Pauwels for help and comments on my transcriptions and chord tools.

Special thanks also to Mert Bay for providing the raw output data from the MIREX09 chord detection competition and answering my many questions about the MIREX evaluation process.

Thanks also to my friends and colleagues past and present at Queen Mary. In no particular order: Anssi, Samer, Chris D, Chris L, Dawn, Katy, Piem, Nico, Andrew, Thomas, Manu, Matthias, Amélie, Dan S, Chris S, Yves, Frank, Ling Ma, Karen, Ho, Andy M, Xue, Bindu, Zhang Dan, Gina, Wang Yi, Yashu, Paula and Vindya.

Many thanks to Mum, Dad, Simon, Jan and David for helping to proof read bits of the thesis for me.

Thanks to Simon Travaglia's BOFH for providing me with amusement when I needed a break from writing.

Finally I would like to thank Lingling, my friends, family and my ever-patient landlady Margaret for putting up with me while I was writing up.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis we address automatic extraction of harmony information, specifically chord symbols, from audio and the evaluation of algorithms designed to perform this task. In this chapter, we discuss our motivation for this work in section 1.1 and present an outline of the thesis structure in section 1.2. In section 1.3 we summarise the contributions made in this work and in section 1.4 we list our own papers related to the thesis.

1.1 Motivation

Harmony is one of the fundamental elements of tonal music. In recent years, automatic extraction of harmony information from audio has become a popular research area in the field of *music information retrieval* (MIR). Much work has been carried out on automatic methods for key recognition [SMW04, MXKS04, GÓ6, Pee06, CML07, ZR07, SIY+08, Nol09] and chord recognition from audio [SJ01, Fuj99, SE03, YKK+04, HS05, BP05, CPB05, BPKF07, PP08, LS08, RK08, VPM08, WDR09, OGF09c, KO09c, WEJ09, RUS+09, MND09b]. Mauch provides a detailed review of the current state of the art of this research area in his recently published doctoral thesis [Mau10].

In this thesis, we investigate automatic chord recognition and specifically focus on the evaluation of chord recognition algorithms. The progression of chords through time defines the harmonic structure of a piece of music so there are many potential uses for reliable audio chord recognition algorithms. Possible applications include automatic transcription

[WDR09], cover song detection [Bel07] and genre classification [ARD09, ABMD10, PSRI09].

In order to improve the chord recognition algorithms that we create, it is necessary to devise rigorous evaluation processes with which they may be tested. Availability of a large ground truth test set is also an important requirement if we wish to obtain reliable quantitative results from the evaluation process. When we originally started the work reported in this thesis, no such ground truth collection was available so we took it upon ourselves to create a large set of transcriptions. The corpus chosen for this transcription project was the 180 songs that make up the twelve original studio albums by the Beatles [Pol, Lew92].

Manual annotation of a large dataset is a time consuming task and human error is unavoidable in such an endeavour. To ensure that the transcription collection was as consistent and reliable as possible, we devised a machine-readable text syntax for chord labels. Using this syntax for the transcriptions enabled automatic parsing and error checking on the collection. Using this format also allowed us to generate synthesised audio of the chords in the transcriptions automatically that were subsequently used in human listening tests in order to verify the collection.

Details of the evaluation calculations used to produce results are often neglected in publications; the details of the recognition algorithms being evaluated are considered more important. Unfortunately, this leads to the situation where it is difficult to state, with confidence, that one particular algorithm performs better than another based on the results given in separate papers. We would argue that knowing *how* something is evaluated is equally important as *what* is being evaluated if the results are to be truly meaningful. For this reason, we have deliberately focused attention in this thesis on the development of general chord recognition evaluation measures. The evaluation methods we present have been designed to be appropriate for any chord recognition algorithm, given a particular set of evaluation parameters. Using these evaluation methods, researchers may confidently compare their results with others, like for like, by choosing the same set of evaluation parameters.

The *Music Information Retrieval Evaluation eXchange* (MIREX) [Dow08]

is an MIR community based initiative organised by the *International Music Information Retrieval Systems Evaluation Laboratory* (IMIRSEL) at the University of Illinois at Urbana-Champaign. The MIREX community holds a set of evaluation tasks each year, in connection with the *International Society for Music Information Retrieval* (ISMIR) conference, to compare the performance of the most recent MIR algorithms developed by community members. One research track for MIREX is the audio chord detection track. A primary motivation for the work on evaluation methods presented in this thesis has been to provide an improved system for evaluating the entries for future iterations of the MIREX chord detection task.

1.2 Thesis structure

In this section we outline the structure of the thesis, chapter by chapter.

Chapter 2: Theoretical underpinnings

We start chapter 2 by introducing important background information on fundamental aspects of music theory and psychoacoustics which are crucial to the understanding of harmony. We then present derivations and discussion of models for tonal space based on neo-Riemannian harmony theory. Using the theory behind Chew's spiral array [Che00] as a starting point, we then go on to propose a novel six-dimensional model for equal tempered pitch space. This model is used as the basis of our tonal centroid calculation which is employed in chord segmentation algorithms later in the thesis and has also been used by other researchers including Lee and Slaney [LS07, Lee08, LS08] in chord recognition.

Chapter 3: Chord extraction from audio

We introduce three chord recognition algorithms in chapter 3. All three are pure signal processing approaches using a tuned chromagram [HS05] generated from a constant-Q transform [Bro91] as the front end. No machine learning techniques are used in our recognition approach. The first

algorithm, outlined in section 3.1, is a simple system that recognises chords on a frame-by-frame basis. The second and third algorithms, discussed in sections 3.2, use the same front end as the first but pre-segment the chroma features before performing the chord recognition step. The segmentation process uses a harmonic-change detection function calculated from a six-dimensional tonal centroid, based on our pitch space model introduced in chapter 2. The second algorithm uses a simple peak-picking algorithm to find potential chord boundaries in the detection function. The third algorithm improves the segmentation performance by introducing hysteresis to the peak picking algorithm in order to discard small spurious peaks.

Chapter 4: Representing chords in plain text

In chapter 4, we discuss the issues associated with representing chord symbols in plain text, in a machine-readable but musically intuitive format. The challenges associated with this issue are discussed in detail in section 4.1. We then propose a specification for a chord symbol representation in section 4.2.1 and using this specification we go on to develop a logical model and a corresponding text syntax for chord labels in sections 4.2.2-4.3. To make the chord symbol syntax convenient for use in experiments, a toolkit for manipulating the labels has been written in Matlab. These tools are briefly discussed in section 4.4.

Chapter 5: Chord symbol comparison methods

In chapter 5, we discuss several methods for comparing chord symbols with each other. Our main motivation for investigating different chord comparison methods is to provide a formal basis for our frame-based recall evaluation methods discussed in chapter 8. In this case it is important that we can clearly define what constitutes a 'correct match' between a machine-estimated symbol generated by a chord recognition algorithm and a hand-annotated symbol in the ground truth test set. The comparison methods detailed in chapter 5 are also used to produce the statistics of the Beatles chord transcription collection discussed in chapter 6.

Chapter 6: A Reference Transcription Dataset

To enable rigorous testing of the chord recognition algorithms described in chapter 3 it is necessary to have a large hand-transcribed dataset for use as a ground truth in the evaluation process. In chapter 6 we describe the process of creating and verifying such an annotated dataset for the twelve Beatles studio albums. At the end of chapter 6, we present statistics of the Beatles transcription collection that were generated using the chord symbol comparison methods described in chapter 5.

Chapter 7: Local audio file alignment

The Beatles transcription collection was originally released in 2007. Since then, many researchers have used the collection in their work and some have found poor time alignment between the transcriptions and their local audio files to be a problem. Under copyright law, it is not possible for us to make our original copies of the Beatles audio files available for other researchers so time alignment will always be a potential issue when others use our transcriptions. In chapter 7 we propose a solution to this problem. We demonstrate a method that uses short audio fingerprints taken from the original annotated audio as guides for altering local audio files, thus allowing correct alignment. This technique provides a simple, legal way for researchers to acquire accurately aligned audio data for annotated data sets: an important factor in obtaining accurate experimental results.

Chapter 8: Chord recognition evaluation methods

In chapter 8, we examine the evaluation techniques for chord recognition currently in use in the community. We then propose a new approach to these techniques that will allow more general and fair comparisons to be drawn between algorithms.

We start by presenting a formal treatment of chord recall using parameterised ordered set matching functions introduced in chapter 5. We then introduce a new dictionary-based recall evaluation technique that provides a fairer way to judge algorithm performance within the constraints of the

algorithm's capability. A new chord sequence likeness measure is developed, based on the proportion of shared tones in pairs of chords, rather than a strict chord matching function with a binary output. We also propose an improved segmentation quality measure that complements chord symbol recall and provides an alternative perspective on the performance of a chord recognition system.

Chapter 9: Results

Using the evaluation techniques discussed in chapter 8, we present the results for the three chord recognition algorithms developed in chapter 3 compared with results for the other twelve entrants for the MIREX09 chord evaluation [mirb]. We compare evaluation results calculated using our chord symbol recall method with results calculated with the chord mapping techniques used in the MIREX08 and MIREX09 chord detection evaluations. The 180 songs from the Beatles chord transcription collection described in chapter 6 are used as the test set for all the evaluations.

Chapter 10: Conclusions and further work

In chapter 10 we make conclusions on the work presented in the thesis and suggest directions for future work based on our findings.

Document conventions and presentation

There are some two-word terms used frequently in this thesis which, in order to disambiguate from their component words, we have decided to concatenate into single words. Thus, we will refer to the type of a chord as a *chordtype* and the name of a pitch which is not attached to an associated octave register is referred to as a *pitchname*.

In this thesis we use teletype font for chord symbols and also for chord symbol variables in equations. To make it clear which is which, chord symbol strings are always written in quotes in-line, e.g. '`C:maj7`' and in teletype double quotes in equations e.g. $\overrightarrow{\mathbf{v}}_{\texttt{"C:maj7"}}$. Chord variables appear without quotes e.g. $\mathtt{R_X}$.

We use diagrams in many places because we consider them to be an immediate and intuitive way of conveying information to the reader. Unless explicitly referenced, all diagrams in this thesis have been created by the author using Inkscape[1] or generated in Matlab.

1.3 Contribution

The main contributions we have made in this thesis may be summarised thus:

Ch. 2 Introduction of a novel six-dimensional model for equal tempered pitch space.

Ch. 3 Introduction of a harmonic-change detection function using a tonal centroid based on the six dimensional model from chapter 2. We also demonstrate the use of the harmonic change detection function as a pre-segmentation technique for chord recognition.

Ch. 4 Development of a chord symbol model and syntax for chord transcription.

Ch. 5 Development of a system of novel chord symbol matching methods and a new chord sequence likeness measure.

Ch. 6 Creation of a large chord transcription collection for the Beatles corpus and use of chord matching methods from chapter 5 to analyse the statistics of the collection.

Ch. 7 Introduction of a novel algorithm that allows researchers to correctly align their local copy of audio files to our chord transcriptions.

Ch. 8 Development of improved evaluation methods for chord recognition based on the techniques from chapter 5.

The Beatles transcription collection and Matlab toolkit we have created have been used by many researchers in the community including [LB07,

[1] `www.inkscape.org`

Bel07, CB09, PP07, PP08, SVB09, RK08, RRD08, EGL09, MDH+07, AD08, RSN08, SVB09] since their initial release in 2007 and form a major part of the test set used for evaluation in the MIREX chord detection task [mirb].

1.4 Publications

This thesis contains and builds upon work previously published in the following papers:

[**HS05**] *Automatic chord identification using a quantised chromagram*, Christopher Harte and Mark Sandler. Proceedings of 118th Convention. Audio Engineering Society, 2005.

[**HSAG05**] *Symbolic representation of musical chords: A proposed syntax for text annotations*, Christopher Harte, Mark Sandler, Samer A. Abdallah, and Emilia Gómez. Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR 2005.

[**HSG06**] *Detecting harmonic change in audio*, Christopher Harte, Mark Sandler, and Martin Gasser. Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia, 2006.

Chapter 2

Theoretical underpinnings

This chapter provides background information on fundamental aspects of music theory which are crucial to the understanding of harmony. Important concepts are discussed from both music theory and psychoacoustics. This leads to the derivations and discussion of models for tonal space based on neo-riemannian harmony theory.

The final section of the chapter covers a six-dimensional model for pitch space that was developed by the author. This model is employed in audio chord segmentation algorithms later in the thesis and has also been used for chord recognition by other researchers including Lee and Slaney [LS07, Lee08, LS08].

2.1 Musical fundamentals

In this section we will define musical terms of reference used throughout the rest of the thesis and introduce fundamental concepts required for discussions in later chapters.

2.1.1 Musical terms of reference

Throughout this thesis, we will refer to various musical objects such as pitches, notes, chords, scales etc. It is therefore important to briefly state and explain what we mean, in the context of this work, by these terms.

Chord transcription

The final goal of this work is to produce computer algorithms that can recognise chords from recorded audio signals in order produce a chord transcription. To do this we must define precisely what we mean by the term 'chord transcription'. In music theory, a chord sequence is generally analysed as a succession of discrete, non-overlapping chord entities over time. In recorded audio however, it is effectively possible for chords to overlap because frequencies from the notes of one chord can still be present in the next if the sounds are subject to reverberation. In the context of this work, we will assume that chords cannot overlap so we define a chord transcription as a sequence of contiguous time segments, each containing a single chord label.

Notes, pitches and intervals

The *note* is the fundamental building block of all tonal music. In this work, we define a note as the combination of a *pitch* determining the fundamental frequency of the note and a duration that determines the length of time that the pitch is sounded for. A sequence of notes in time may form a melody and a collection of notes played simultaneously may sonify a chord.

In this thesis we will use standard scientific notation [You39] in which a pitch is defined by a *pitchname* and an octave number. A pitchname comprises a *natural* name plus zero or more sharps (♯) or flats (♭). Figure 2.1a shows the arrangement of pitches as keys on a piano keyboard along with associated pitch labels and frequencies over ten octaves. The ten octaves between pitch C0 at 17Hz and C10 at 16768Hz cover the majority of the range of audible frequencies for the human ear. The standard 88-key grand piano keyboard has keys between pitch A0 at 27.5Hz and C8 at 4186Hz. Common musical reference key 'Middle C' is pitch C4 (261.6Hz) and 'Concert A' is pitch A4 (440Hz). Figure 2.1b shows one octave, on the piano keyboard, with labels for all the pitchnames in both English and sol-fa naming systems. The smallest difference between two pitches on the piano keyboard is a *semitone*. The white keys on the keyboard are

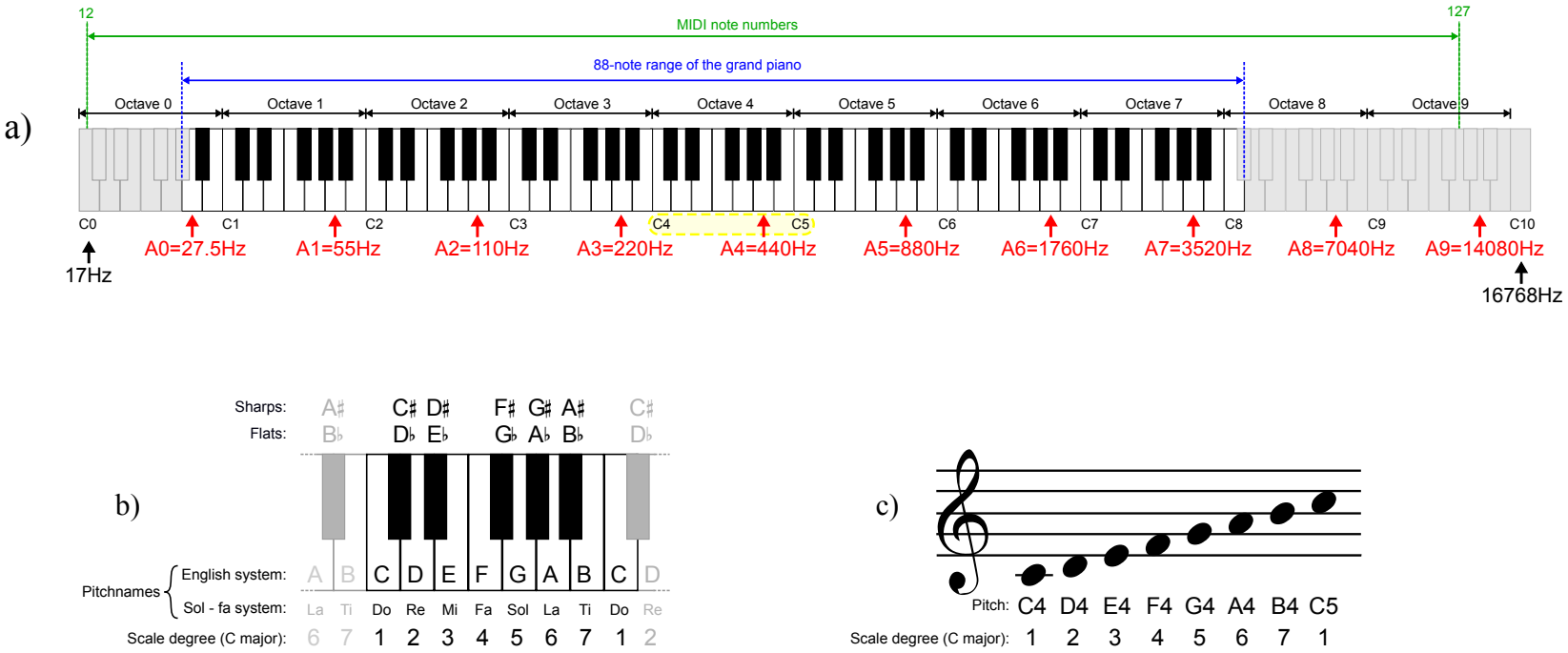


Figure 2.1: a) Pitches and associated frequencies shown in relation to the piano keyboard, pitch C4 being 'Middle C' and pitch A4 being 'Concert A'; b) Pitch names on the keyboard within one octave; c) Pitches in the octave C4 to C5 shown notated on a musical stave, the equivalent octave of the piano keyboard is highlighted in part a) of the figure.

Degree		**Name**	**Relation to Tonic**
1 First	I	Tonic	Unison
2 Major second	II	Supertonic	One tone above the tonic
(♭3 Minor third	♭III)	Mediant	Mid way between tonic and dominant
3 Major third	III	"	" " " " " "
4 Perfect fourth	IV	Subdominant	Fifth below the tonic
5 Perfect fifth	V	Dominant	Fifth above the tonic
(♭6 Minor sixth	♭VI)	Submediant	Mid way between the subdominant and the tonic
6 Major sixth	VI	"	" " " " " " " " "
(♭7 Lowered Seventh	♭VII)	Subtonic	One tone below the tonic
7 Seventh	VII	Leading Note	Leads into the tonic

Table 2.1: The degrees of the diatonic major scale plus the flattened degrees (shown in parentheses) present in the diatonic minor.

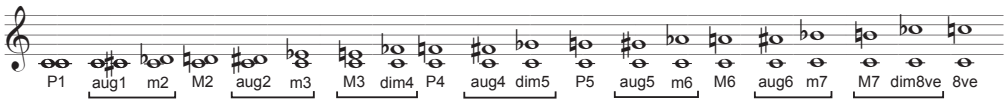


Figure 2.2: The two-note intervals from unison up to an octave relative to Middle C. Square braces under pairs of intervals denote enharmonic equivalents.

labelled by the natural names A-G. The black keys are labelled with the natural name of an adjacent white key and a sharp or flat to show that they are one semitone higher or lower than that natural respectively. Also shown, underneath the pitchnames in figure 2.1b and c, are the numeric *degrees* of the diatonic major scale based on C [Tay89].

The relative difference between two pitches is called an *interval*. Figure 2.2 shows the different two-note intervals between pitch C4 and pitch C5 the octave above. An interval can be expressed as a number of semitones or as an *enharmonic spelling* comprising a diatonic major *scale degree* plus zero or more sharps and flats. Table 2.1 shows a list of diatonic scale degrees (for both major and minor scales) plus the names that are commonly used to describe them in music theory. The first note of the diatonic major or minor scale is known as the *Tonic* and the fifth degree is known as the *Dominant*. The other degrees of the scale have names which reflect the note's relationship with the tonic or the dominant. For simplicity, the degrees of the scale are often referred to by the system of roman numerals. These names and relationships, along with the corresponding roman numerals, are given in table 2.1.

The perfect fourth and fifth intervals, denoted by a P, are made with the tones which are closest to the *prime* (the tonic reference note for the interval) in the harmonic series (covered in section 2.1.2). The Major (M) intervals are intervals formed by the tonic and the notes of the major scale other than the fourth and fifth. The Minor intervals (m) are one semitone less than their corresponding major interval. Increasing a perfect or a major interval by a semitone produces an *augmented* (aug) interval. Reducing a perfect interval by a semitone produces a *diminished* (dim) interval [SF]. The pairs of intervals which have square brackets beneath them, such as the aug4 and dim5, will sound the same if played on a

modern keyboard instrument. However, they have different notation (or *spellings*) on the musical stave because they may be different to each other depending on the tuning system which is in use [HA96]. These related pairs of intervals are known as *enharmonics* and their importance in harmony will be discussed further in the following sections.

2.1.2 Pitch perception and the harmonic series

When a string is plucked, the resulting pitch that we hear is not just the fundamental frequency, but a number of different frequency components. The string will vibrate at frequencies that are integer multiples of the fundamental frequency which are known as *harmonics* or harmonic *overtones* or *partials*. The first six harmonic modes of a vibrating string are shown in figure 2.3. The amplitudes of these harmonics relative to each other dictate the timbre of a single percieved pitch[Ols67, Ben90].

The different degrees of the diatonic major and minor scales are found in the harmonic series created by these overtones. The frequencies of degrees in the diatonic scales are related by simple integer ratios to the fundamental frequency of the tonic note of the scale (as shown in blue in figure 2.3).

We may model our perception of musical pitch using the $\log_2$ frequency scale [Coo99]. Figure 2.4 shows this by projecting a piano keyboard on to linear and logarithmic frequency scales with the harmonic series for the same pitch with fundamental frequency f on both. Harmonics on the linear frequency scale have a periodicity of $n.f$ where n is the index of the overtone. Therefore, as f decreases or increases, the harmonics of the pitch will get closer together or move further apart on the linear scale respectively. On the $\log_2$ scale however, the harmonics are periodic on $2^n.f$. The harmonic structure stays constant for pitches on the $\log_2$ frequency scale, invariant under changes in fundamental frequency.

2.1.3 Pitch height and chroma

The log scale preserves musical intervals under transformations which is why we hear a doubling in frequency as a jump up an octave regardless

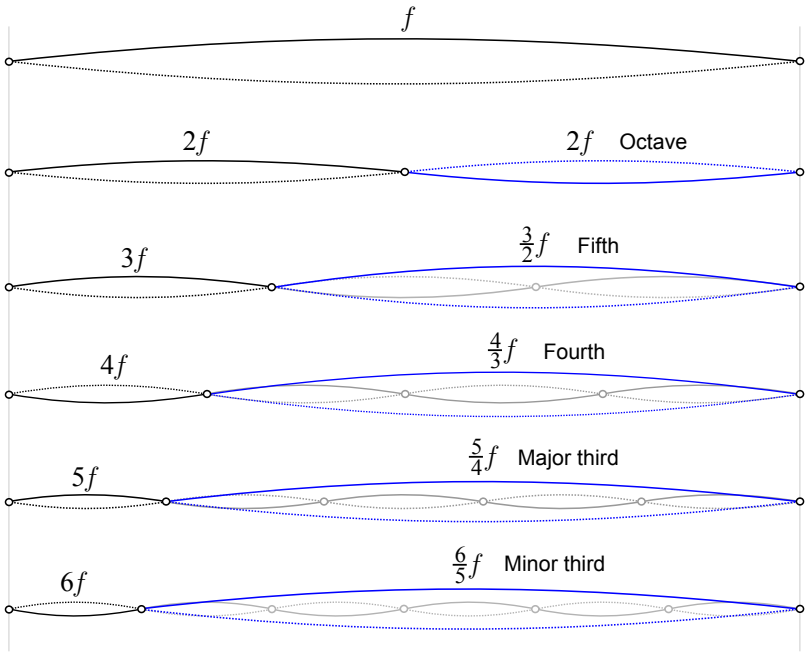


Figure 2.3: Harmonic modes of a vibrating string. Closely related intervals (shown in blue) within the same octave are formed from simple integer ratios of the fundamental.

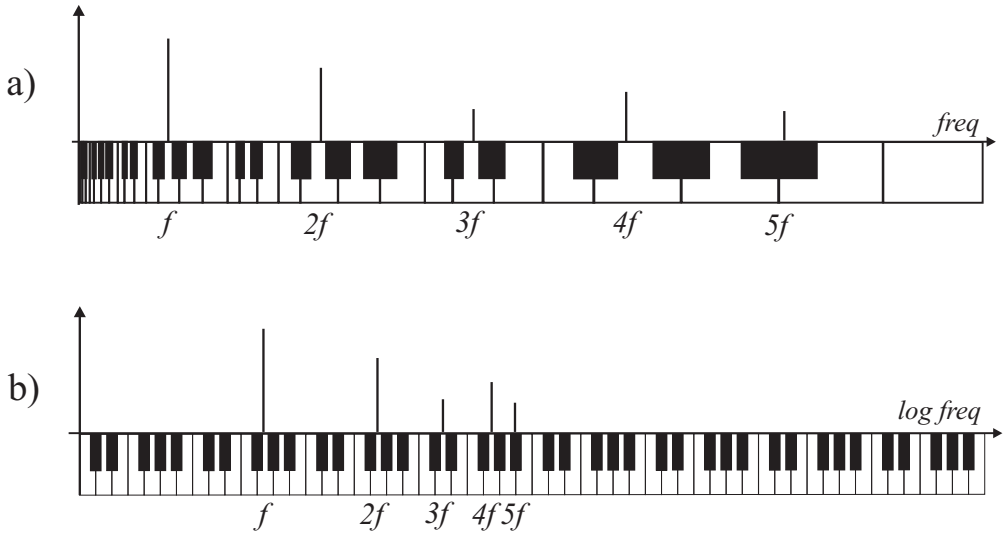


Figure 2.4: Perception of pitch: Five partials in the harmonic series of a musical tone shown on a) the linear frequency scale and b) the $\log_2$ frequency scale (amplitudes of the partials are arbitrary in the diagram).

of the starting frequency. However, the log scale does not capture the fact that certain intervals with particular properties such as octaves and

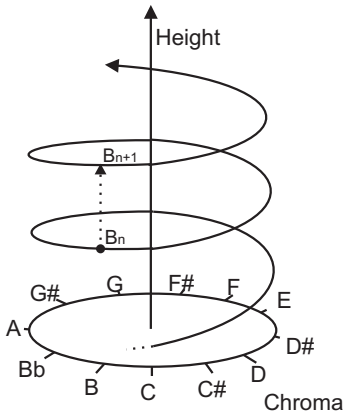


Figure 2.5: The pitch helix, as proposed by Moritz Drobisch

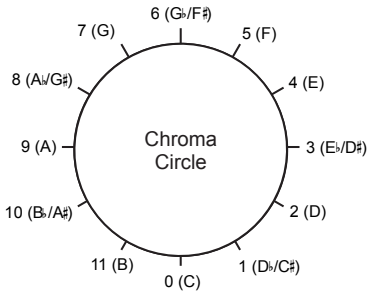


Figure 2.6: The Chroma Circle showing the 12 pitch classes around the circumference

perfect fifths are special (how they are special will be covered in more detail later in the chapter).

In 1855, the physicist Moritz Drobisch proposed tones be represented on a helix with octaves directly above each other in order to represent the importance of the octave interval in a model of pitch perception [Coo99]. A representation of this helix is shown in figure 2.5. Krumhansl and Shepard [KS79] found this same helical representation of pitch to be recoverable by applying multidimensional scaling to human responses in psychological experiments.

A note's position in the octave is referred to as its *Chroma* or *Pitch-class*. The *chroma circle* shown in figure 2.6, with the twelve pitch classes around the circumference, is the base of the pitch helix from figure 2.5.

Chroma is independent of pitch height. This can be demonstrated with the *Shepard Tones*, a group of twelve tones which are ambiguous in height

but unambiguous in pitch. These tones were developed by Shepard [She64] at Bell labs and produce an audio illusion of ever ascending pitches as the tones progress round the circle.

2.2 Harmony in western tonal music

Harmony is the movement between musical chords through time. The Oxford Dictionary of Music [Ken94] gives this definition for a chord:

> **Chord:** Any simultaneous combination of notes, but not usually fewer than three. The use of chords is the basic foundation of harmony.

In this thesis we will extend this definition to allow a chord to comprise zero or more notes. In doing so, we make it possible to consider single notes and non-tonal material (such as silence or purely percussive sounds) as 'chords'. Although this stretches the original definition somewhat, it enables us to design a chord labelling syntax in chapter 4 and build a system of chord comparison methods in chapter 5 which are subsequently used throughout the rest of the thesis.

2.2.1 Consonance and Dissonance

The degree to which a simultaneous combination of notes is perceived to be acceptable or pleasing in a given musical context is called *consonance* and its converse i.e. how unpleasant it is, is *dissonanace*.

In psychoacoustic experiments during the 1960s, Plomp and Levelt [PL65] linked the perceived consonance of two simultaneously sounding sine waves to the critical bandwidth in human hearing [HA96]. In the human ear, tones are defined to be within the same critical band if they are close enough in frequency such that their responses on the basilar membrane overlap. Tones which were equal in frequency were judged 'perfectly consonant' and tones with a frequency difference greater than one critical bandwidth were judged to be consonant. However, tones which differed in frequency by between 5% and 50% of a critical bandwidth were

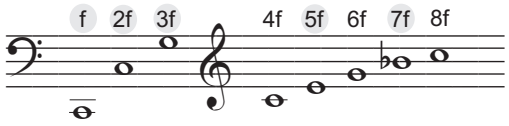


Figure 2.7: The first 8 harmonics of note C2 with the prime harmonics highlighted.

judged as dissonant with maximum dissonance occurring at a quarter of a critical bandwidth.

Musical instruments generally produce complex tones made up of many harmonically related frequency components (see section 2.1.2). For a combination of notes, all of the audible harmonics (up to the sixth or seventh) of each note contribute to the perceived consonance or dissonance of the chord and each pair of harmonic components must adhere to the rules determined above. Tones whose fundamental frequencies are related by small integer ratios, will have many overlapping harmonics and will therefore sound consonant together. For intervals which are not closely related, many pairs of harmonics may fall within the 5-50% of a critical bandwidth distance of each other and will therefore sound dissonant to our ears. Much harmonic theory is based on the ratios of the numbers 1 to 7 [Bal97] which links to the way the human auditory system perceives the first seven partials of a tone. Figure 2.7 shows the first eight harmonics of a tone (C2 - two octaves beneath middle C on the piano keyboard). It is the prime harmonics which are perceptually most important because the other harmonics are multiples (and hence harmonics) of these primes themselves. These prime harmonics give rise to what Balsach calls the *Convergent Chord* [Bal97] - for a C, considering the first 7 harmonics, the convergent chord is CGEB♭. This is a 'chord' which is present in a single note and it has certain properties which help to explain the existence of some of the rules of harmonic progression discussed later in this chapter.

In the convergent chord for the note C (CEGB♭) there is a perfect fifth interval C:G but there are also two intervals which are near fifths E:C

(an augmented fifth[1]) and E:B♭ (diminished fifth). These intervals tend to seek resolution towards a perfect fifth which causes a degree of internal tension in the note. This internal tension can be resolved by dropping to the note a fifth below (in the case of C this will be an F) which resolves the augmented fifth E:C to the perfect fifth F:C. It can also be resolved by dropping to the note a minor second below (from the C to a B) which resolves the diminished fifth E:B♭ to the perfect fifth E:B[2].

2.2.2 Chords

The two most common types of chord are the major and the minor triads. The major triad is made up of the tonic note plus the fifth and third degrees of the major scale; in the key of C the tonic major triad is CEG. This triad can be viewed as a major third interval (I-III) underneath a minor third (III-V). The minor triad is made up of the tonic note plus the fifth and the third of the minor scale which, with root note C is CE♭G. This triad is the complement of the major triad in that it is built from a minor third underneath a major third [Tay89].

Other common triad chords include augmented, diminished and suspended types. An augmented chord is built up of major thirds; the C augmented chord would thus contain the notes CEG♯. By contrast, the diminished chord is built up of minor thirds, hence the C diminished triad would be CE♭G♭. Suspended chords are not built of major and minor thirds but instead comprise a major second and a perfect fourth. The suspended second chord is a major second under a perfect fourth e.g. CDG; a suspended fourth is a perfect fourth underneath a major second e.g. CFG. In classical western harmony, suspended chords generally *resolve* the suspended note onto a major or minor third to form a more perceptually 'stable' triad chord.

The basic building blocks of harmony in western tonal music are the triads which are built upon the degrees of the scale that defines the key. Figure 2.8 shows the triads which are built on the C major scale. There

[1]For a definition of the two-note intervals, see figure 2.2.
[2]It is interesting to try this for yourself on a piano.

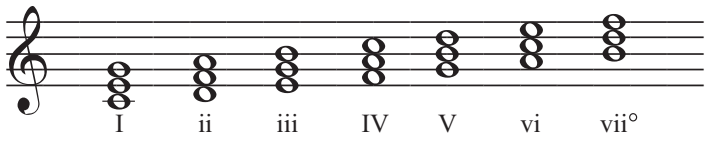


Figure 2.8: Triads built on the degrees of the C major scale

are three major triads, I, IV and V (C, F and G), three minor triads, ii, iii and vi (Dm, Em and Am) and a diminished triad vii° (B°). The diminished chord is given a lower case letter like a minor chord because its first interval is a minor third. Three-note chords other than the natural ones described above may occur in a major key but only with the addition of tones from outside the key which are known as *accidentals*. More triads can be formed in a minor key as there are several different minor scales [Tay89].

Name, root, type and family

In this thesis, we will use the compound term *chordname* to mean the full name given to a particular chord. For example, the chord formed by sounding the tones C, E and G simultaneously would generally be given the chordname 'C major'. A chordname is a concatenation of a *root* pitchname (in this case C) and a *chordtype* (major). The root pitchname of the chord is an absolute value. The chordtype defines which tones should be sounded relative to the root to complete the chord. For the 'major' chordtype, this will mean that the chord comprises the root plus tones a major third and a perfect fifth above the root. Chordtypes that share common characteristics may be grouped into a chord *family*. For example, we might say that major triad, major seventh and major sixth chords are all members of the 'major' chord family because they all share the three tones of the major triad as a foundation.

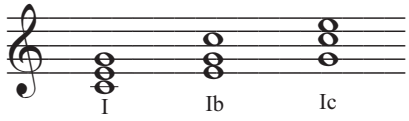


Figure 2.9: Inversions of a C Major Chord

Inversions and Extensions

The bass note played at the bottom of the chord does not necessarily have to be the root of the chord. Any one of the chord tones may be used as the bass note and which one is chosen has an important effect on the chord's sound and the way in which it functions in relation to other chords in the harmonic progression. Which degree of the chord becomes the bass note defines the chord's position or *Inversion*. Figure 2.9 shows the different inversions of the C major chord: the C major triad in root position (I), the first inversion (Ib) with the third as the bass note and the second inversion (Ic) with the fifth as the bass note. The inversion of a chord may often be written by giving the chord name and the bass note to use. For example the first inversion (Ib) of a C major chord would be written C/E and is referred to as 'C over E'.

The chords introduced so far have been triads which, by definition, are made up of three notes. Other, more complex combinations can be produced by adding more notes and *extending* the chords. The first common type of extended chord is produced by adding a 7^{th} interval to the chord. Adding a major 7^{th} to a major triad produces the Major 7^{th} (maj7) chord. Adding the minor 7^{th} to the major triad produces a 7^{th} (7) often referred to as a 'dominant 7^{th}' chord, so called because this type of extension is often used in dominant root chords. Adding the minor 7^{th} degree to a minor triad produces the minor 7th chord (min7).

Further extensions, beyond the confines of the original octave, may also be added. By adding another interval of a third on to a 7 chord a 9^{th} chord is created (the major 9th is the interval of an octave plus a major second, see figure 2.10(a)). The intervals 11 and 13 may also be used as extensions but the intervals 10, 12 and 14 have no function as extension

Figure 2.10: a) Extended intervals in relation to Middle C. b) These two chords share the same function as they contain the same notes but their level of consonance is different.

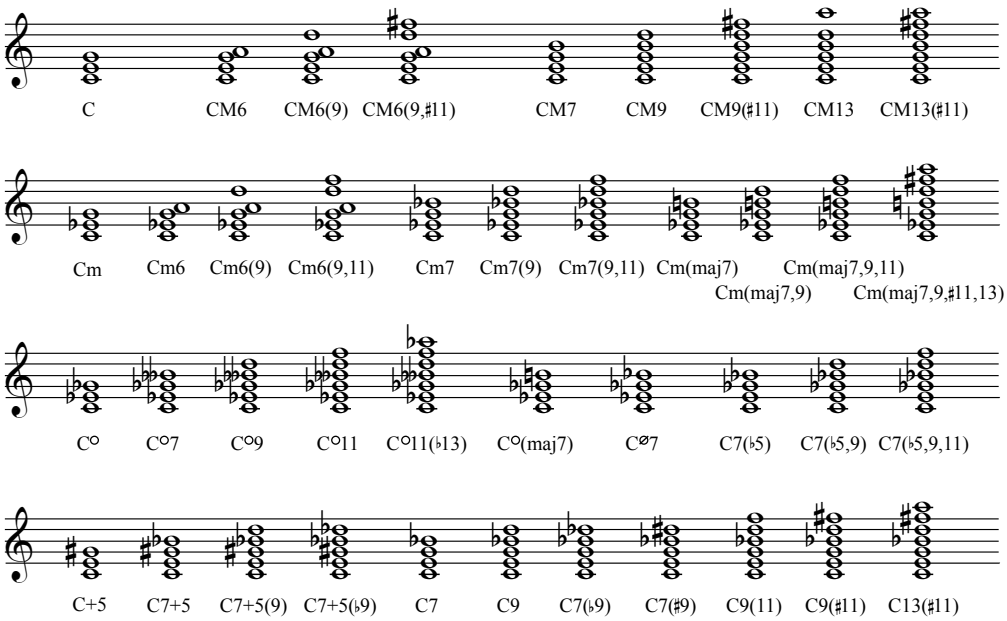


Figure 2.11: Standard Jazz chord types and their extensions (compiled from Coker [Cok64]). The circle symbol ∘ denotes a diminished chord, ∅ half diminished and +5 augmented.

notes because they are octave repetitions of the notes in the basic triad and 7^{th}; extensions greater than 13 are not used as they would also be octave repetitions of lower degrees of the chord. Figure 2.11 shows the common chord types used in Jazz and their extensions with major type chords on the first stave, minor types on the second stave, diminished type chords on the third stave and 7^{th} and augmented chords on the bottom stave; in the figure, extended major chords are denoted with capital 'M' and extended minor chords a lower case 'm'. The 9^{th} interval is an octave above the

2nd interval. It should be noted that this distance of at least an octave between the root and the extension is very important to the consonance of the chord. Figure 2.10(b) shows two C chords containing the notes CEGB♭DF. Although both chords are made up of the same pitches, the C9(11) chord sounds more consonant than the clustered chord (which could be considered to be a C7(2,4)). This is because when the pitches are sounded close together they have many dissonant pairs of harmonics. However, when separated by over an octave, they do not have so many close pairs of harmonics [Bal97].

2.2.3 Tuning Systems

The most important interval in western music, the octave, has a frequency ratio of 2:1. All the common western scales are based around an octave containing twelve semitones. Early tuning systems were based on the intervals found between the overtones in the harmonic series. Figure 2.7, earlier in the chapter, shows the first 8 harmonics of a tone. The important intervals can be found between these harmonics: The octave is found between the fundamental and the 2nd harmonic, the perfect fifth between the 2nd and 3rd and the perfect fourth between the 3rd and 4th. Most of the other intervals can be found between higher pairs of harmonics but these become difficult to isolate practically.

The Pythagorean scale is a tuning system built from perfect fifths. From a starting note, C for example, go up a perfect fifth (ratio 3:2) and you reach a G, go up another perfect fifth and you reach D and so on. By ascending by a perfect fifth twelve times you pass through each of the twelve different pitches in the chromatic scale (albeit in different octaves) and effectively return to a C again; we will call this new note $\hat{\mathrm{C}}$. This cycle of perfect fifth intervals can be shown as the spiral in figure 2.12. The frequency of $\hat{\mathrm{C}}$ is given by multiplying the frequency of C by the perfect fifth ratio to the power of twelve [HA96]:

$$\hat{\mathrm{C}} = \left(\frac{3}{2}\right)^{12} \cdot \mathrm{C} = 129.7463 \cdot \mathrm{C} \tag{2.1}$$

This is not, however, quite the same frequency as the note C', 7 octaves

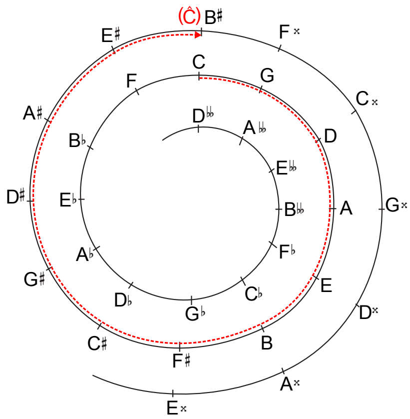


Figure 2.12: The sequence of pitchnames from double flats to double sharps shown on a spiral.

above C, given by multiplying the frequency of C by 2^7.

$$\text{C'} = 2^7 \cdot \text{C} = 128 \cdot \text{C} \tag{2.2}$$

Therefore, seven octaves is actually slightly flatter than twelve perfect fifths by an amount known as the *Pythagorean Comma.* Thus, although in theory we should have arrived at the same pitch, the note $\hat{\text{C}}$ should actually be considered as the enharmonic B♯, as shown on the spiral in figure 2.12, rather than a C. The frequencies of notes for the Pythagorean scale are thus calculated by ascending the required number of perfect fifths and dividing by the relevant power of two to bring the note back into the current octave. For example, the ratio used to produce the major third of the scale (E in the example of C major) will be given by ascending four perfect fifths (CGDAE) and descending two octaves i.e.

$$\frac{\text{E}}{\text{C}} = \left(\frac{3}{2}\right)^4 \cdot \left(\frac{1}{2}\right)^2 = \frac{81}{64}$$

Another tuning system, widely used in the past but not so popular today, is *Just intonation.* The Just diatonic scale is built by keeping the intervals that make up the major triads pure. That is to say that the tonic, subdominant and dominant triads each include interval ratios of a perfect fifth (3:2) and a major third (5:4) plus the dominant and subdominant keynotes are a perfect fifth and a perfect fourth above the tonic respectively. The major third of this scale is given by the ratio 5:4

so differs from that of Pythagorean scale for which it is calculated above as 81:64. Using Just intonation, a keyboard instrument must be tuned for playing in a particular key. The instrument will sound pleasing when used to perform pieces written in that key due to the consonance of the pure intervals. Pieces in closely related keys will also sound acceptable to the ear. However, for works written in keys distant from the intended tuning, the instrument may sound so out of tune as to render them unplayable.

It is time consuming and impractical to retune a keyboard instrument every time you want to play a piece in a different key. It is also costly to have twelve separate keyboards, so, for practical purposes, an alternative tuning solution is required. The solution is the *Equal Tempered* tuning where all the intervals between semitones are equal. This tuning system splits the octave equally using a ratio r_e between semitones which is equal to the twelfth root of two:

$$r_e = \sqrt[12]{2} \approx 1.0595 \tag{2.3}$$

This gives a tuning which makes modulation to all keys possible although no key will be perfectly in tune with the pure intervals of the Just scale. For practical considerations of consonance, the tuning works because the harmonics of the intervals are within the 5% critical bandwidth limit. However, with equal tempered tuning, chords will include beat frequencies between harmonics which would not be present with pure intervals. With equal tempered tuning, enharmonics can be considered as being equivalent to each other, thus the spiral from figure 2.12 closes to become the circle of fifths as shown in figure 2.13.

2.3 The Tonnetz

A number of music theorists have worked on representations of tonal space based on the relationships of pitches, intervals and triads. Riemann developed a planar representation of tonal space known as the *Tonnetz* [Rie15, WMR92]. Figure 2.14 shows his original diagram of the space (the symbols are in German musical notation). The Tonnetz is also referred to as the *harmonic network*. It is important to note that this model

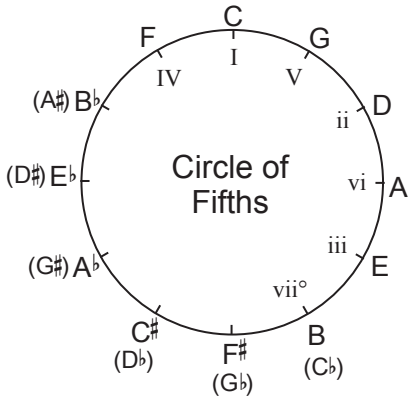


Figure 2.13: The circle of fifths; the roman numerals inside the circle denote chords built on degrees of the major scale in the key of C major.

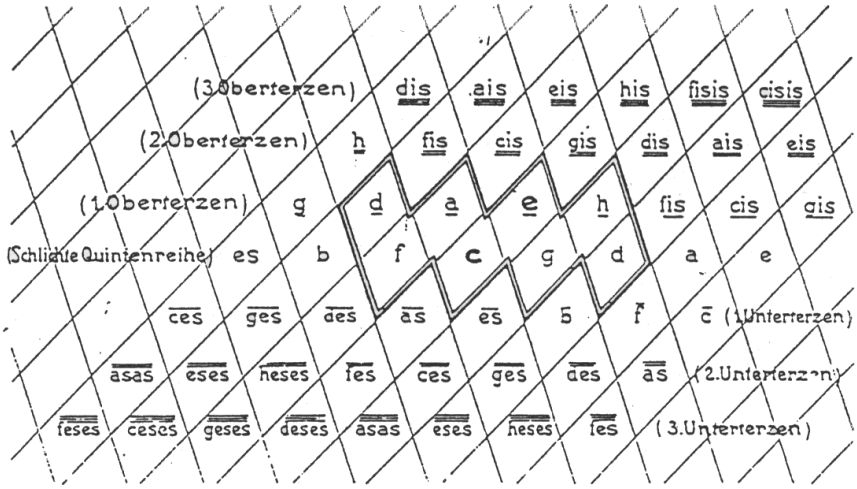


Figure 2.14: Riemann's Tonnetz [Rie15] reproduced from [Lon01]. N.B. In German musical notation the letter h denotes B.

is concerned with relationships between tones as opposed to the model described in the previous section which was based on relationships between keys. On the horizontal is the line of fifths, so moving to the right the interval is a fifth and moving to the left the interval is a fourth. Moving up and to the right is the interval of a major third and down to the right a minor third. Moving up to the left is a major sixth and down to the left is a minor sixth.

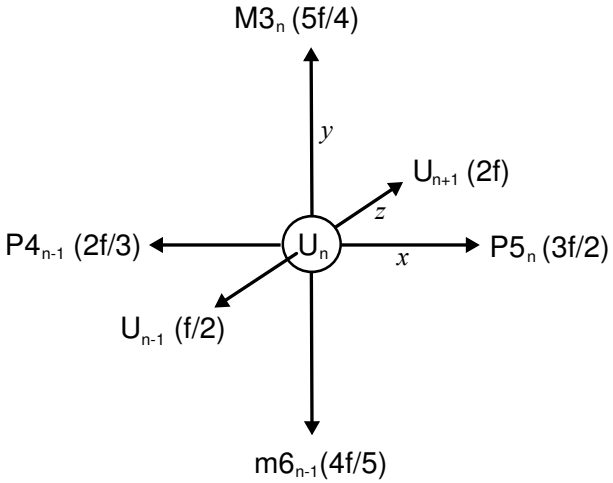


Figure 2.15: Axes of the three dimensional discrete space proposed by Longuet-Higgins for harmonic relations. Positive movement of one unit in the x axis transforms pitch U_n by a perfect fifth, in the y axis by a major third and in the z axis by an octave.

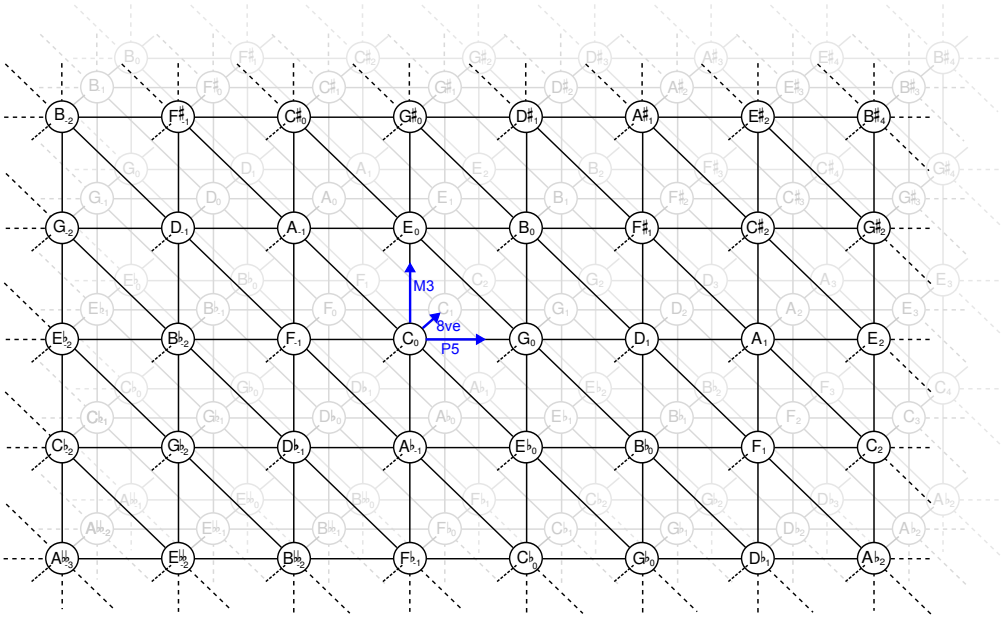


Figure 2.16: Diagram of part of Longuet-Higgins' pitch space.

2.3.1 Longuet-Higgins' pitch space

Longuet-Higgins arrived at a similar result through an alternative derivation [Ste02, LH62a, LH62b] by examining which integer frequency ratios

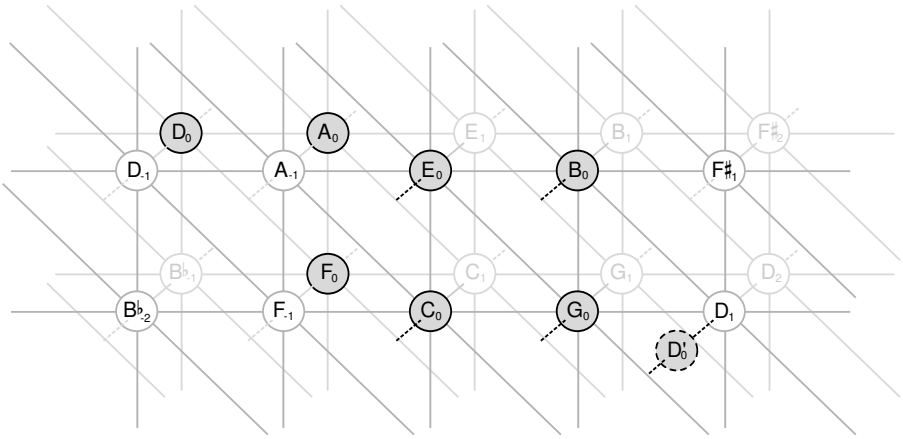


Figure 2.17: Pitches of a one octave major scale on C_0 in Longuet-Higgins' pitch space. D_0 and D'_0 are separated in pitch by a tuning comma.

created musical intervals. These ratios from 1 to 16 and their corresponding musical intervals in roman numerals plus their relation to the tonic note C are shown in table 2.2.

Longuet-Higgins showed that the harmonic intervals are those that have a ratio that can be expressed as $\frac{3}{2}^x \cdot \frac{5}{4}^y \cdot 2^z$ where x, y and z are integers representing perfect fifth, major third and octave translations respectively. Therefore, the harmonic relation between two notes may be represented as a vector in a three dimensional discrete space with C_0 at the centre, as shown in figure 2.16, with the z-axis going into the page.

The distances between pitches in Longuet-Higgins' pitch space is proportional to how closely related they are harmonically. Figure 2.17 shows the positions of pitches of a major scale starting on C_0 in the space. The tones of the major 7 chord (tonic, major third, perfect fifth and major seventh) in the same octave are close together, in the same plane on the z-axis. The second, fourth and sixth in the same octave are further away

Multiple	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Pitchname	C	C	G	C	E	G	-	C	D	E	-	G	-	-	B♭	C
Interval	I	I	V	I	III	V	-	I	II	III	-	V	-	-	♭VII	I

Table 2.2: Longuet Higgins' table of integer frequency multiples that create close harmonic intervals.

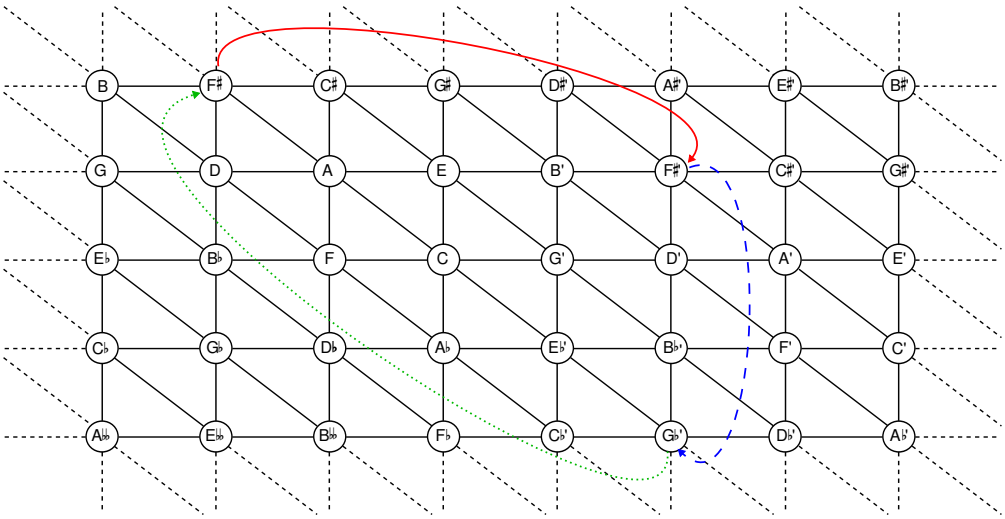


Figure 2.18: The plane of pitchnames formed when octave equivalence is assumed in Longuet-Higgins' pitch space model. The red solid line shows the repetition in the space when pitchname equivalence is assumed. The blue dashed line shows repetiton in the plane where enharmonic equivalence is assumed and the green dotted line shows the final repetition where both are assumed.

in the space to the left and +1 unit in the z axis. It is also possible to move to the right on the x axis and down in the z-axis to reach what appear to be the same pitches. It is interesting to note that the major ninth D_1 is closer to the tonic C_0 than either of the instances of the major second D_0. This supports points from the earlier discussion on consonance in section 2.2.2. The diagram shows two pitches labelled D_0 and D'_0 but these are not in fact the same frequency relation to the tonic; they are separated by a comma. The pitch labelled D_0 in the diagram is at coordinate {-2, +1, +1} relative to C_0 which translates to $(\frac{2}{3})^2 \times \frac{5}{4} \times 2 \times f = \frac{10}{9}f$ whereas the pitch labelled D'_0 in the diagram is at coordinate {+2, 0, -1} relative to C_0 which translates to $(\frac{3}{2})^2 \times \frac{1}{2} \times f = \frac{9}{8}f$.

Simplifying the model

Longuet-Higgins simplified the model by assuming octave equivalence, reducing the three dimensional space to a plane by discarding the z-axis. Figure 2.18 shows the plane that is formed which is equivalent to Riemann's tonnetz; because of the tuning commas, the plane is infinite.

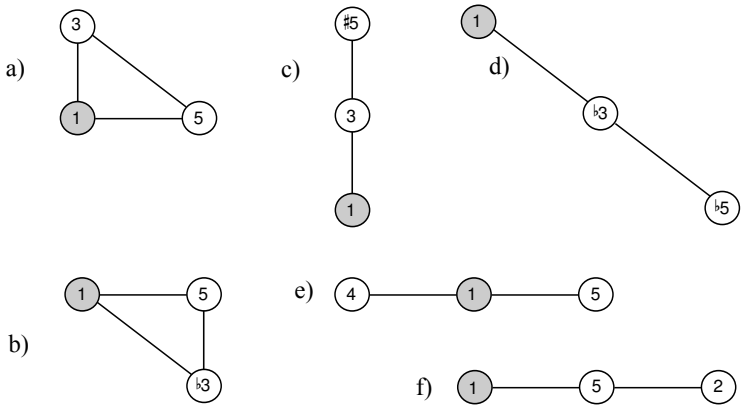


Figure 2.19: Triad (three-note) chord shapes on the tonnetz: a) major b) minor c) augmented d) diminished e) suspended fourth f) suspended second.

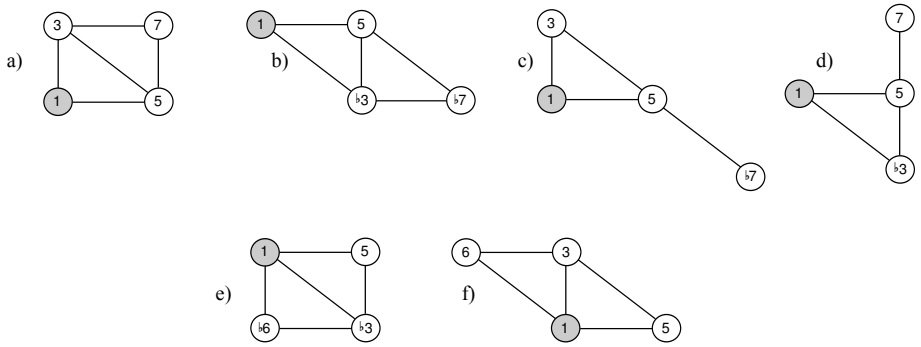


Figure 2.20: Tetrad (four-note) chord shapes on the tonnetz: a) major seventh b) minor seventh c) seventh d) minor (major seventh) e) minor sixth f) major sixth.

One interesting property of this planar harmonic network is that chords are constant patterns formed from adjacent pitchnames. For example, the major triad is an upwards pointing triangle whereas the minor triad is a downwards pointing triangle. This property is put to use later by Chew [Che01] in the Spiral array model. Figures 2.19 and 2.20 show the shapes of the common triad and tetrad chords on the plane. It should be noted that certain chord shapes are equivalent to each other depending on which element of the chord is assumed to be the root note. In the case of

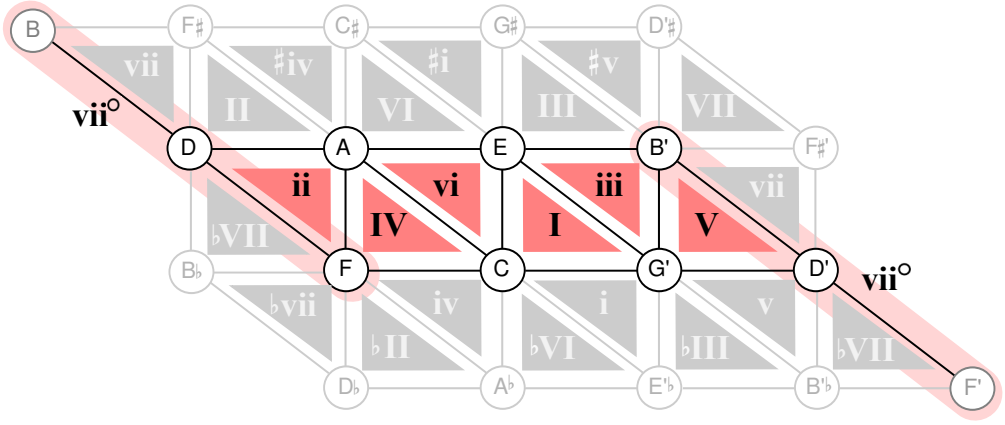


Figure 2.21: The diatonic chords of the major key arranged on the pitchname tonnetz (shown in red) for the key context of C major. Close chromatically related chords are also shown.

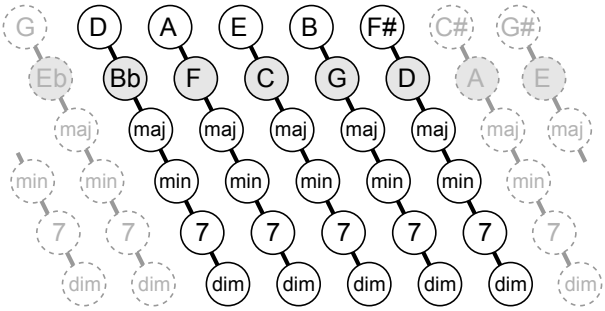


Figure 2.22: Bass buttons on an accordion are arranged in cycles of fifths horizonally with a line of major third basses above and related chords below.

the triad chords, the suspended second and suspended fourth are the same shape on the network. However, the root note of the suspended fourth is the middle element whereas the root note of the suspended second is the leftmost element. Thus, a C suspended fourth is equivalent to an F suspended second chord, both comprising pitchnames C, F and G. The same can be seen in the four-note tetrad chords in figure 2.20 where the major and minor sixth chords are the same shape as the minor and major seventh chords respectively.

Chord relationships and key regions on the tonnetz

Looking at the arrangement on the tonnetz of the naturally occuring chords from a major key, shows that the distance relationships between

chords in a progression are also closely linked to this network. Figure 2.21 shows the arrangement of chords in a major key. The harmonically related chords are close to the tonic chord in the plane. We can see that the dominant major chord (V) and the subdominant major (IV) are either side of the tonic major (I) on the x-axis. The tonic major chord is closest to its relative, parallel and mediant minors (vi, i and iii respectively). These minor chords all share two pitchnames with the tonic chord.

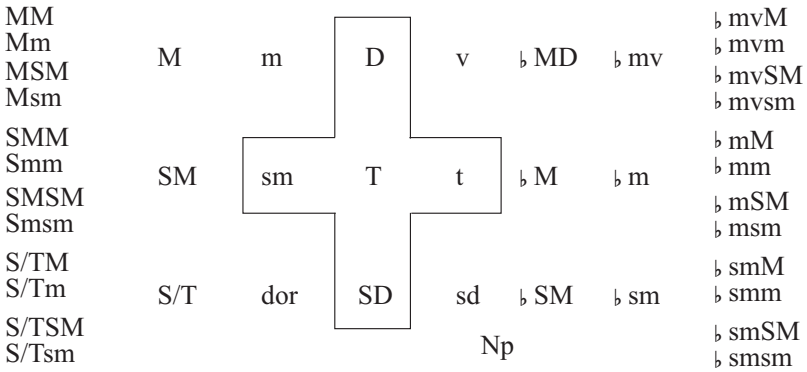
In many ways the arrangement of pitches on the tonnetz is more intuitive, in terms of harmony, than the arrangement based on pitch height and chroma that we see in notated music. Pitch height and chroma also form the basis for the layout of keys on many musical instruments including the piano. Some instruments, however, have developed using the line of fifths and in some cases even the thirds. For example, the accompaniment strings on a guitar zither of the kind made famous by Anton Karas[3] are tuned in cycles of fifths over two octaves. The bass buttons of accordions are also arranged in cycles of fifths. On larger models the basses also include thirds, as shown in figure 2.22.

Key regions on the tonnetz

The relationships between pitches, chords and keys on the tonnetz discussed thus far are also described by Lerdahl in his theory of tonal pitch space [Ler01]. Lerdahl considers three levels: the pitchclass level, chord level and regional level. The pitchclass level corresponds to the pitchname tonnetz, but assumes enharmonic equivalence which will be covered later in this section. The chordal level deals with the relationships between chords in the same region of the system and the regional level looks at the relationships between chords in different key *regions*. The relative arrangement of pitches and the chords they form, in any particular region, is invariant to absolute pitch of the tonic.

These relationships between keys and chords were also noted by Arnold Schoenberg [Sch54]. Schoenberg suggested that the idea of temporary modulation to other keys should be discarded in favour of the concept of

[3]Karas played the theme to the film 'The Third Man' on a guitar zither.



T	tonic	**Np**	Neapolitan
D	dominant	**dor**	Dorian
SD	subdominant	**S/T**	supertonic major
t	tonic minor	**♭M**	flat mediant major
sd	subdominant minor	**♭SM**	flat submediant major
v	five-minor	**♭MD**	flat mediant major's dominant
sm	submediant minor	**♭m**	flat mediant minor
m	mediant minor	**♭sm**	flat submediant minor
SM	submediant major	**♭mv**	flat mediant minor's five
M	mediant major		

Figure 2.23: Schoenberg's Chart of the Regions [Sch54] for a major key. N.B. All symbols in capitals refer to major keys; those in lower case to minor keys.

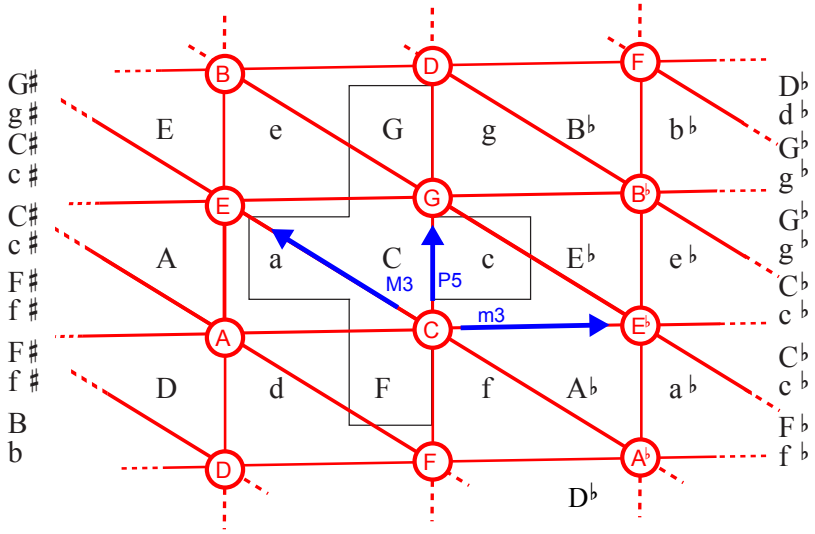


Figure 2.24: Shoenberg's regions for C major showing its relationship to the pitchnames on the Tonnetz (superimposed in red).

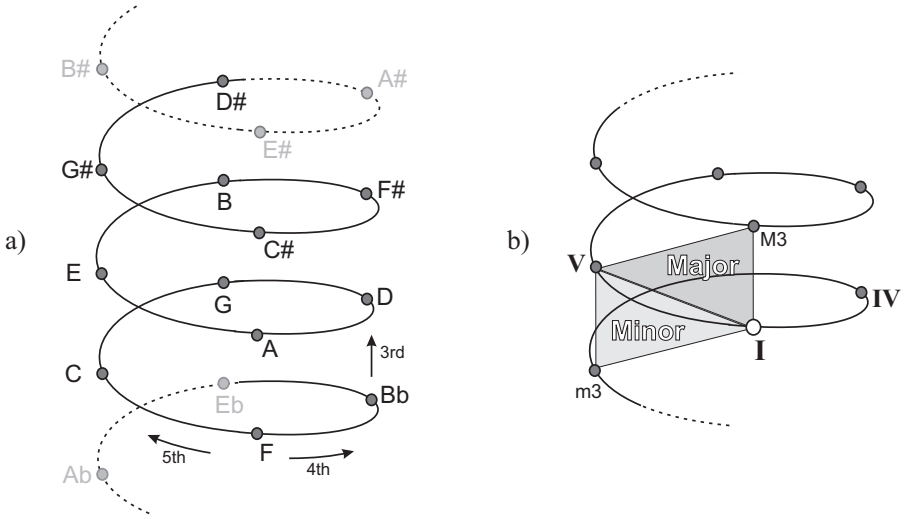


Figure 2.25: a) The spiral array as proposed by Chew b) major and minor triads become triangles inside the cylinder.

monotonality. When the harmony of the piece moves away from the tonic it is still considered as being in the same tonality, but in a different region of that tonality. With this in mind, Schoenberg developed the 'Chart of the regions' shown in Figure 2.23. This shows how the different regions relate to the central tonic. Regions which are closely related to the tonic, such as the dominant, subdominant and its parallel and relative minors[4], are close to it on the chart. Likewise, regions which are not closely related to the tonic are distant from it on the chart. Figure 2.24 shows the chart of regions for the key centre of C major and its relationship to the pitchnames on the Tonnetz.

2.3.2 Assuming pitchname equivalence: Chew's Spiral array

All points on the pitchname tonnetz are unique in terms of their enharmonic spelling and the number of commas they are from any given reference point, therefore the plane is infinite. However, if we assume pitchname equivalence, i.e. we choose to ignore the tuning commas, then the

[4]The parallel minor of a major key is the minor with the same tonic note e.g. the parallel for C major is C minor. The relative minor of a major key is the minor key which shares the same key signature and can be found by dropping three semitones from the major's key note e.g. For C major the relative minor is A minor.

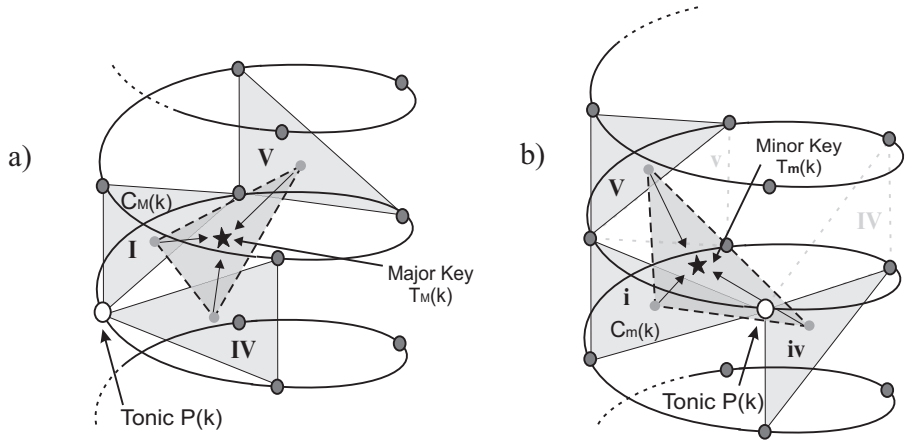


Figure 2.26: Geometric representations of the 'centre of effect' for a) a major key and b) a minor key on the spiral array

plane wraps on to the surface of a cylinder with the line of fifths forming a helix of enharmonic pitchnames. The resulting three-dimensional model is Chew's *Spiral Array* [Che01].

Figure 2.25(a) shows a graphical representation of the spiral array. The line of fifths wraps round itself forming a spiral that makes one full turn every four pitchnames. This means there is a major third interval separating any pitch on the spiral and the one directly above it.

In the spiral array model the surface of the cylinder that the spiral wraps around lies on the unit circle in the x, y plane. Chew defines $\mathrm{P}(k)$ as a point on the spiral at position $[x, y, z]^T$ representing a pitch of index k where pitch C is arbitrarily chosen to be $\mathrm{P}(0)$, fixed at location $[0, 1, 0]^T$ in the 3D space[5]. Each pitch on the spiral can be defined in terms of transformations from one of its neighbours using rotation and vertical translation.

$$\mathrm{P}(k+1) = \mathbf{R} \cdot \mathrm{P}(k) + \mathbf{h} \tag{2.4}$$

Where

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{h} = \begin{bmatrix} 0 \\ 0 \\ h \end{bmatrix}$$

[5]The position of the reference pitch in the 3D space is arbitrary. In her thesis, Chew points out that "It is the relation between the pitch representations that is of utmost importance, and where the spiral begins is of little consequence".

Where Chew's use of this model differs from the other tonal representations, discussed earlier in the chapter, is that in modelling chords and key centres in this three dimensional representation she allows musical entities to be defined inside the spiral. A chord is modelled as the composite result or *centre of effect* of its component pitches (see figure 2.25(b) showing the major and minor triads forming triangles on the spiral). This effect is a point in the space floating somewhere inbetween the chord pitches. Likewise, a key can be modelled as an effect of its defining chords (I, IV and V), as shown in figure 2.26. The effect is represented spatially by a convex combination of its components.

Mathematically, Chew represents a chord by a convex combination of its component pitches. The chord is represented by a weighted average of the positions of the component pitches. For example, the major triad is given as the root, $\mathrm{P}(k)$, the fifth, $\mathrm{P}(k+1)$ and the major third $\mathrm{P}(k+4)$ therefore the representation is:

$$\mathrm{C_M}(k) = v_1 \cdot \mathrm{P}(k) + v_2 \cdot \mathrm{P}(k+1) + v_3 \cdot \mathrm{P}(k+4) \tag{2.5}$$

Where Chew includes weights v_i that represent the importance of the pitch on the generated chord and

$$v_1 \geq v_2 \geq v_3 > 0 \qquad \text{and} \quad \sum_{i=1}^{3} v_i = 1.$$

In the same way, a key's effect is given as the weighted sums of its constituent chord effects. The centre of effect moves around in the space inside the spiral depending on the chords that are being played. Chew employs this movement in a key boundary finding algorithm [Che02]. When in one key area, the centre of effect will not move around very much. When there is a key change, the centre of effect will change its position to reside in the new key area. The boundary finding algorithm looks at the Euclidean distance between the centre of effect's positions in adjacent time frames. When the distance is large, the centre of effect must have moved a long way during that frame, suggesting a change in key.

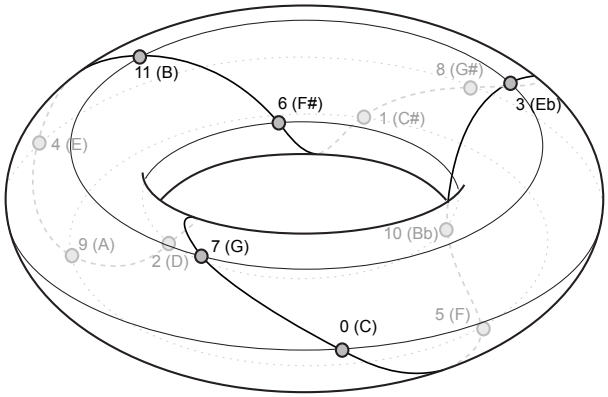


Figure 2.27: When enharmonic equivalence is assumed, the spiral array can be represented in four dimensions as a spiral on the surface of a hypertorus.

2.3.3 Toroidal models of tonal space

Chew's original work was based on symbolic music analysis from musical scores rather than audio so Just intonation may be assumed as the enharmonics are known. If enharmonic equivalence is assumed, the pitchnames of the spiral array map on to the twelve pitch classes causing the line of fifths helix to fold round on itself, forming a four dimensional torus. Figure 2.27 shows a graphical representation of this pitchclass torus; it should be noted that the diagram is only for visualisation and although the image presents the torus as a three dimensional object, the true distances in the space are such that the inner and outer diameters are in fact equal.

This toroidal model has been discussed by many other theorists including Cohn [Coh98] and Hyer [Hye95]. Hyer re-imagines Riemann's Tonnetz assuming enharmonic and octave equivalence to produce the representation shown in figure 2.28. The numbers shown at the nodes denote the number of semitone intervals between that node and the central node 0. The pitchclass level in Lerdahl's tonal pitch space theory [Ler01] is also equivalent to this toroidal model.

The ToMIR

Shoenberg's regions were presented in figure 2.23 showing the relation between different regions in the context of one key. From the circle of

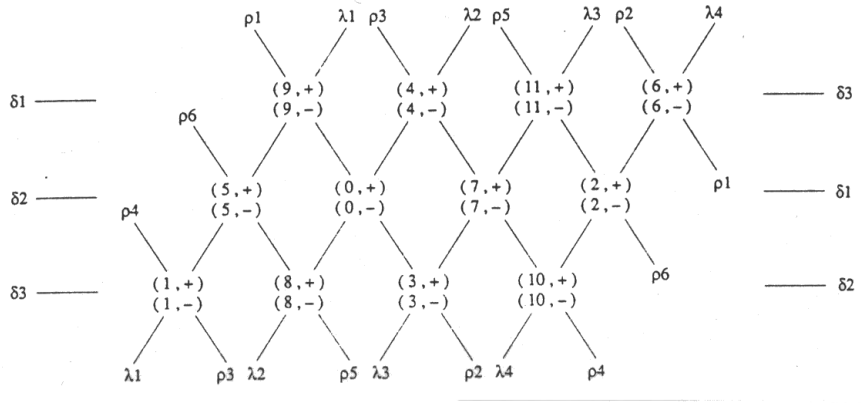


Figure 2.28: Diagram of equal tempered tonnetz relations reproduced from Hyer [Hye95].

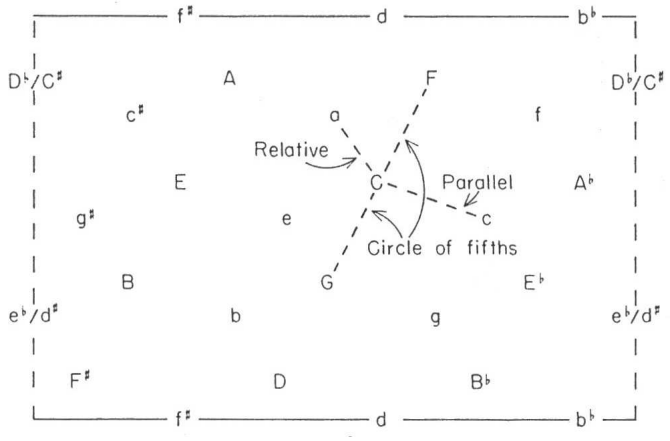


Figure 2.29: Inter-key distances - Krumhansl and Kessler's experiments in music perception and cognition map tonal space on to a torus (reproduced from London [Lon01])

fifths, introduced in section 2.2.3, it is possible to see that a cyclic pattern exists in the chart. This pattern is used by Blankertz, Purwins and Obermayer [BPO99, Pur05] to derive their *Toroidal Model of Inter-key Relations* or *ToMIR* which is equivalent to the key region level of the pitchclass toroid discussed in the previous section. The model is developed from the chart of the regions by taking the centre three columns and extending them along the line of fifths (discarding the Neapolitan Region). This 'strip of key' is shown in in figure 2.30a. It is possible to see that the top line of minor keys (the relative minors of the major keys

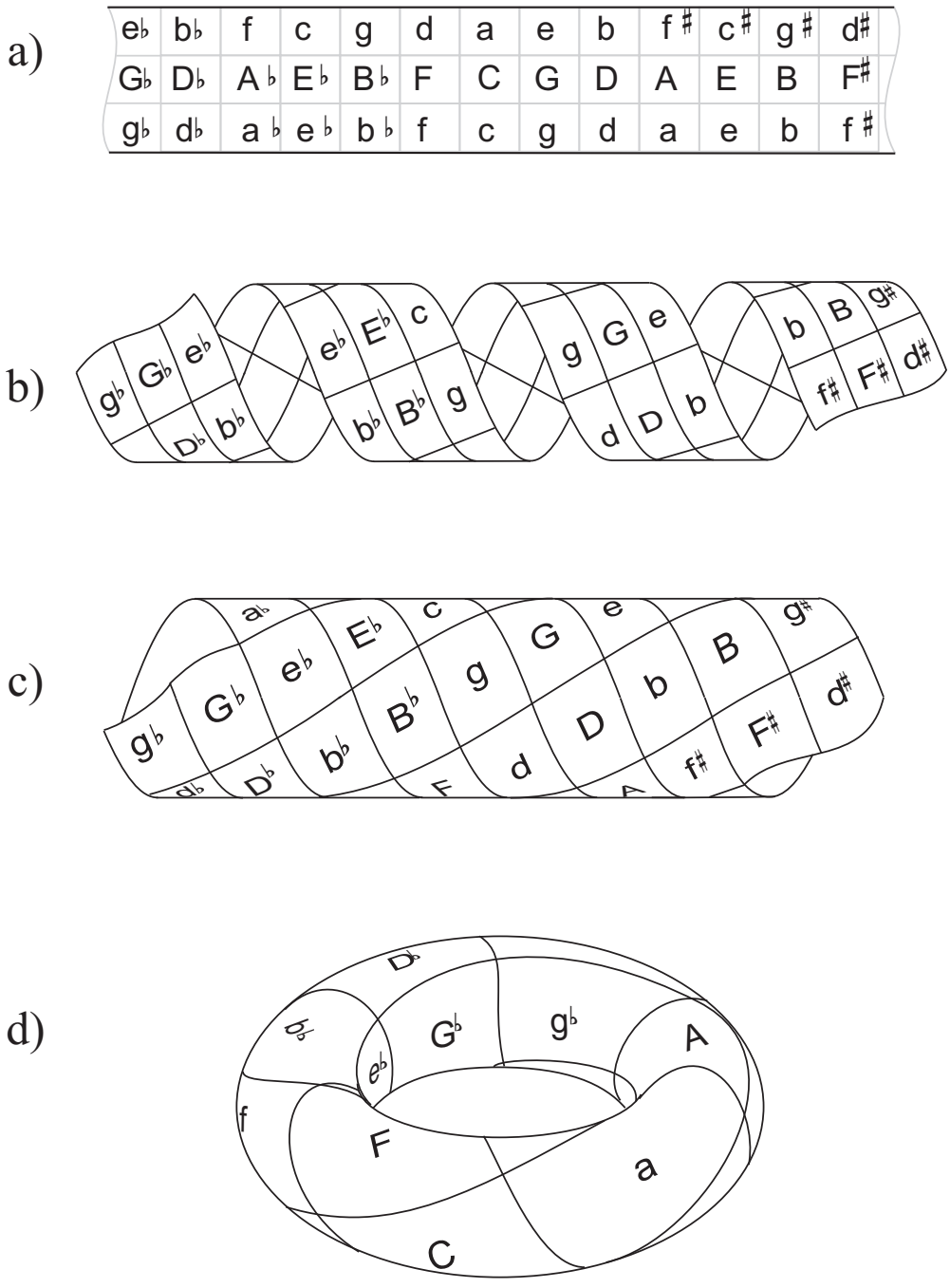


Figure 2.30: Derivation of the ToMIR from Harmonic Relations: a) A 'Strip of Key' derived from Schoenberg's Regions b) The strip is wrapped up on itself to form a spiral c) Doublings of minor keys are merged forming a double spiral on a tube d) The ends of the tube are joined together to create a Torus

in the middle line) can be lined up with the bottom line of minor keys (the parallel minors of the major keys) if the strip is twisted round on itself to form a spiral (as shown in figure 2.30b. As the minor keys in the top and bottom lines of the strip are equivalent to each other when lined up, the doublings may be merged forming a tube with two spirals on its surface, one for the major keys and one for the minors (figure 2.30c). If enharmonic equivalence is assumed, the tube may be folded round and the enharmonic keys joined together to form a torus; the two spirals of major and minor keys wrapping round its surface three times. This is shown in figure 2.30d.

Krumhansl and Kessler's psychoacoustic experiments with probe tone ratings [Kru90] empirically measured how well human subjects thought that certain notes and chords fit within a given tonal context. Tones and chords closely related to the tonal context scored highly and notes and chords which were distant did not score highly. These experiments produced the map of inter-key distances shown in figure 2.29 which maps directly on to ToMIR. Krumhansl also takes pains to point out that this torus is a three dimensional representation of a four dimensional space. The distance between two keys is the Euclidean distance in four dimensions, not the distance in two or three dimensions shown in the graphical representations.

2.4 Extension of the toroidal pitch space model

Chew's 'centre of effect' in the spiral array models each chord or key as a point in the space surrounded by the spiral of fifths. Each centre of effect point may be described by a single vector in that space. The distance between vectors for successive chords or key regions can be used to segment the music under analysis.

We wish to apply the same technique to the toroidal pitchclass model by describing chords as points in the four dimensional space. To visualise the space we can plot the four dimensions as two circles, one being the circle of fifths and the other the circle of minor thirds, as shown in figure 2.31. Four triad chords are shown in the figure, all with C as their

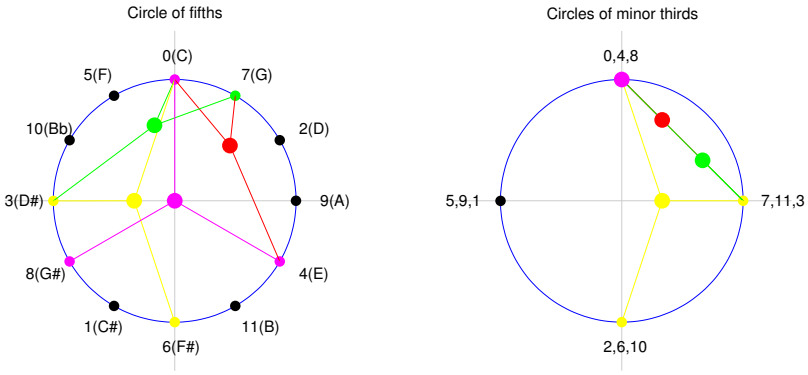


Figure 2.31: Chords shown as points in the four dimensional pitchclass space. C major triad (red), C minor triad (green), C augmented triad (magenta) and C diminished triad (yellow).

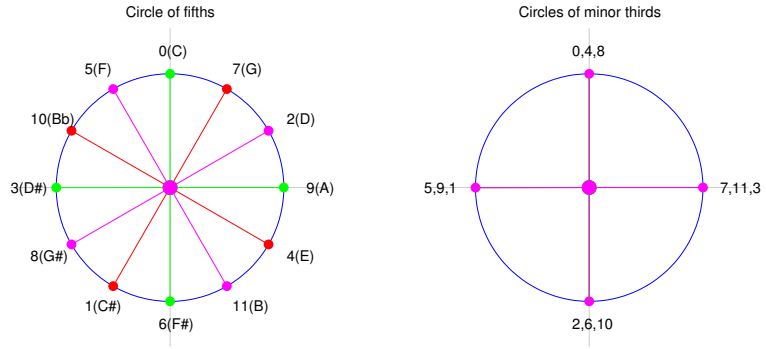


Figure 2.32: Three diminished seventh chords shown as points in the four dimensional pitch class space; green based on C, red based on C♯ and magenta based on D. All three occupy the same point at the centre of both circles despite none of them sharing any common tones.

root. The C major and minor triads are complements of each other in the space, the diminished triad is close to the centre in both circles and the augmented triad is at the centre of the circle of fifths but on the edge of the circle of minor thirds.

Unfortunately, with only these four dimensions, the model cannot discriminate between certain important chord types. For example, the three diminished seventh chords shown in figure 2.32 do not share any common tones, yet they all occupy the point at the centre of both circles. This is also the point occupied by the chord produced when all pitchclasses are present simultaneously. What we must remember is that this four dimensional space is really an alternative projection of the twelve dimensional

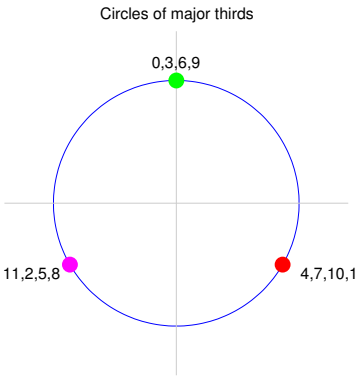


Figure 2.33: When projected as points in the circle of major thirds, the three diminished seventh chords from figure 2.32 are easily distinguished from each other.

space described by the twelve pitch classes. Projecting in four dimensions reduces the dimensionality but also throws away information. The diminished seventh chords are different from each other but our current projection hides this from us. To recapture this information we will introduce a new pair of dimensions which describe the circle of major thirds as shown in figure 2.33. By including these two new dimensions in our model, we may now distinguish between the different diminished seventh chords easily.

The six dimensional model gives us an interesting way to visualise the relationships between different chords. Figure 2.34 shows the diatonic chords for the key of C major plotted in the six dimensional space. The chords of the major key are close to each other in the circle of fifths, occupying several overlapping points. However, the positions of the chord centres in the other two circles are more spread out. Taking just the tonic C major chord and the supertonic D minor as an example (in figure 2.35) we can see that although they are both at the same point in the circle of fifths, they occupy complementary positions in the other two circles.

2.4.1 Distances in the six dimensional tonal model

The ratio of height to radius in Chew's spiral array is determined by ensuring that a set of inequalities, derived from the perceptual distances between musical intervals in equation 2.6, are satisfied [Che00]. Let d(i,i')

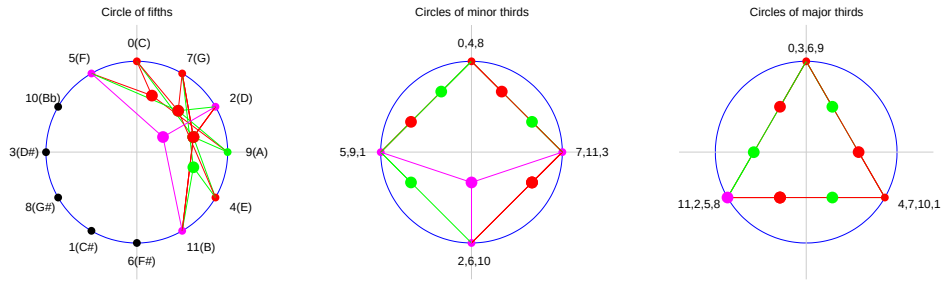


Figure 2.34: The diatonic chords in the key of C major. Major triads C, F and G are shown in red, minor triads D, E and A are shown in green and diminished triad B is shown in magenta.

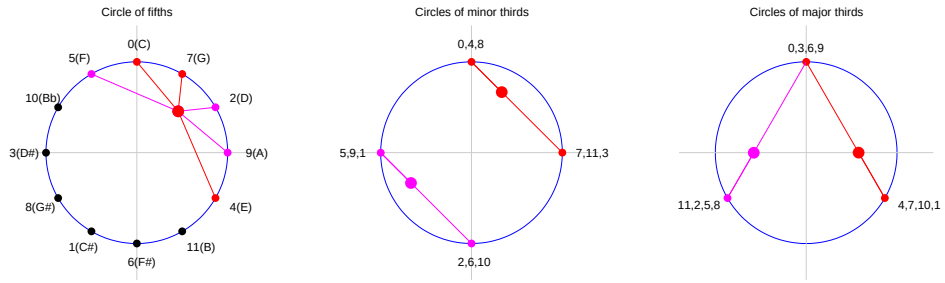


Figure 2.35: The C major triad (red) and D minor triad (magenta) in the six dimensional space. They occupy the same point in the circle of fifths but they occupy complementary positions in the other two circles.

be the tonal distance between two pitches separated by interval i, relative to the lower pitch, and i' is the complementary interval relative to the upper pitch. Therefore we see that the smallest distance is between the tonic and the perfect fourth and fifth intervals. This is followed by the major third then the minor third, the major second, the minor second and finally the diminished fifth.

$$d(\text{P5,P4}) < d(\text{M3,m6}) < d(\text{m3,M6}) < d(\text{M2,m7}) < d(\text{m2,M7}) < d(\text{d5,a4}) \tag{2.6}$$

Using the same method that Chew employs in [Che00], we have formulated these relationships in terms of distances within the six dimensional space and simplified them to the following inequalities (derivations for which can be found in appendix A). The circle of fifths has radius r_1

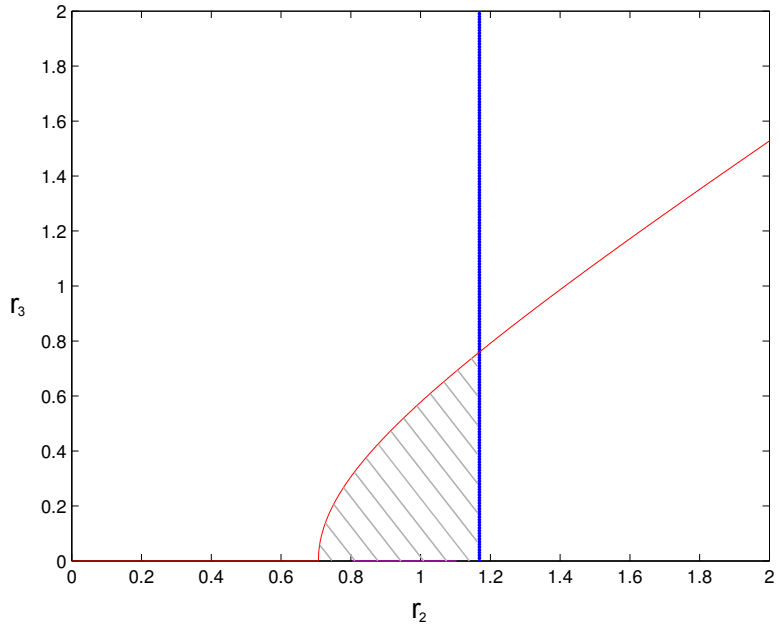


Figure 2.36: Plot of the inequalities that must be satisified for the interval distances in the six dimensional pitch space model when assuming circle of fifths radius $r_1 = 1$. The blue line corresponds to inequality 2.7 and red to inequality 2.8. The hatched area satifies all the constraints.

which we will set to 1, the circle of minor thirds has radius r_2 and the circle of major thirds radius r_3.

$$\frac{1}{\sqrt{2}} < r_2 < \sqrt{\frac{(1-\sqrt{3})}{2}} \tag{2.7}$$

$$r_3 < \sqrt{\frac{2{r_2}^2 - 1}{3}} \tag{2.8}$$

Figure 2.36 shows the inequalities plotted for different values of r_2 and r_3. The area where both are satisified determines the range of acceptable values we may use for r_2 and r_3 in our model when $r_1 = 1$. For the experiments in later chapters, where we use this model, we choose the values $r_2 = 1$ and $r_3 = 0.5$ in accordance with the inequalities.

Chapter 3

Chord recognition from audio

In this chapter we will introduce three chord recognition algorithms that we have developed. All three algorithms are purely signal processing approaches with no machine learning techniques involved. The first algorithm, outlined in section 3.1, is a simple system that recognises chords on a frame by frame basis using a tuned chromagram generated from a constant-Q transform. This algorithm was originally presented by the authors in the paper 'Automatic Chord Identification Using a Quantised Chromagram' [HS05] for the 2005 AES 118th convention.

The second and third algorithms, discussed in sections 3.2, use the same DSP front end as the first but employ a chord segmentation algorithm based on the harmonic change detection function first presented by the authors in the paper 'Detecting Harmonic Change In Musical Audio' [HSG06] for the 2006 ACM Multimedia conference.

Chapter contribution

Most of the work presented in this chapter is based on research carried out between 2005 and 2006. While it is acknowledged that the chord recognition systems we describe here are no longer the state of the art, we include them here to provide the reader with a more detailed explanation of the algorithms than is given in our papers covering the same work [HS05, HSG06]. The work described in those papers has been used in subsequent research by others including [LS07, Lee08, LS08] and cited by many more. Within this context, the main contributions in this chapter are:

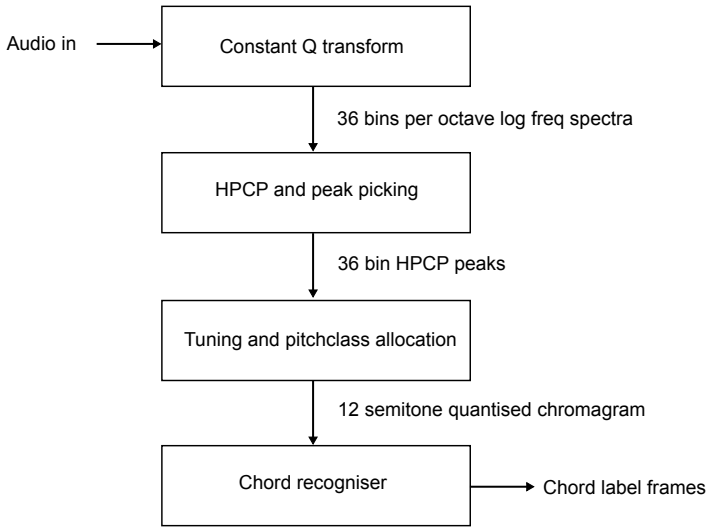


Figure 3.1: Block diagram of basic frame-based chord recognition system.

- Use of a tuning algorithm to generate a quantised chromagram.
- Generating the tonal centroid based on the 6D pitch class hypertorus.
- Development of the harmonic change detection function (HCDF) from the tonal centroid.
- Use of peak picked HCDF for chord recognition segmentation.
- Improved peak picking of HCDF for better chord segmentation.

3.1 Basic chord recognition system

In this section we will present the details of a simple frame-based chord recognition algorithm based on that described in our 2005 AES paper [HS05].

3.1.1 Audio front end

Our basic chord recognition system is a simple frame-based algorithm using a 12-bin semitone quantised chromagram derived from a constant-Q transform. A diagram of the main signal processing blocks in the system is shown in figure 3.1.

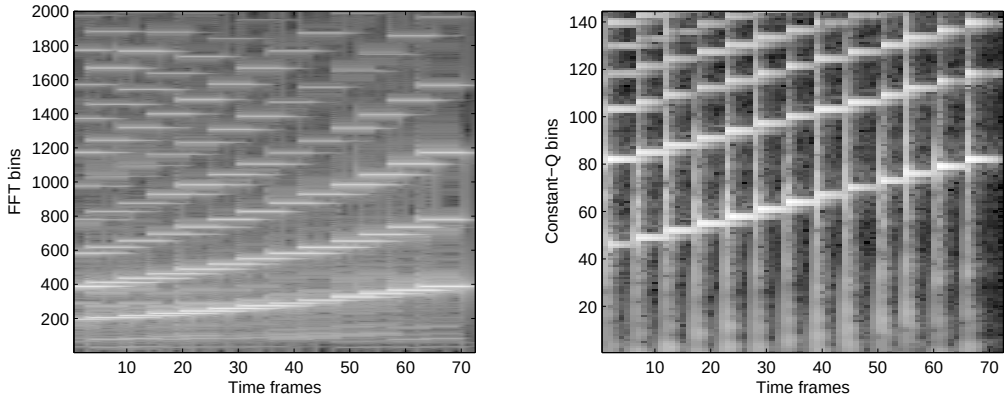


Figure 3.2: Two spectrograms of an ascending chromatic scale starting on middle C. On the left is a linear frequency spectrogram calculated with an FFT; on the right is a log spectrogram produced by the constant-Q transform. Higher energy is shown as lighter greyscales. The constant-Q spectrogram clearly shows the harmonic structure of each note in the scale.

Constant Q transform

The first stage of the system is a *Constant-Q* spectral analysis [Bro91]. This is a logarithmic frequency analysis, so named because it can be viewed as a filter bank in which each filter has a constant-Q value. The ratio of a filter's bandwidth δf to its centre frequency f is called its *quality* or Q *factor*.

$$Q = \frac{f}{\delta f} \tag{3.1}$$

Bandwidth δf is fixed in linear frequency analysis, therefore the value of Q varies in proportion to centre frequency f. In contrast, with logarithmic frequency analysis the filter bandwidths δf vary in proportion with centre frequency f, hence the quality factor Q remains constant.

For musical analysis using a constant-Q transform, we need to find a suitable value for the number of bins per octave β. The k^{th} bin centre frequency is given as:

$$f_k = \left(\sqrt[\beta]{2}\right)^k f_{\min} \tag{3.2}$$

Where frequency f varies between $f_{\min}$, the lowest frequency for which analysis is required, and a maximum frequency which is set to be below the Nyquist frequency. In equal tempered tuning $\beta = 12$, therefore the

frequencies of semitones are separated by the ratio of $\sqrt[12]{2}$ which is equivalent to 1.0595. For frequency analysis in recorded music audio where tuning may vary, quarter-tone (half a semitone) resolution at least is required to distinguish between adjacent semitones. In order to deal with audio signals where the reference tuning frequency is unknown, we opt for a resolution of $\beta = 36$ therefore $\delta f = (\sqrt[36]{2} - 1) = 0.0194$ and the required Q value is $f/0.0194f = 51.4$. Resolution of 36 bins per octave is equivalent to three bins per semitone. This ensures that it is possible to distinguish between adjacent semitone frequencies regardless of the tuning of the recording [GH04, PBO00]. The constant-Q spectrum $\mathbf{Q}_k$ of the time sequence x_n is given by the transform

$$\mathbf{Q}_k = \frac{1}{N_k} \sum_{n=0}^{N_k-1} w_{k,n} x_n e^{-j\frac{2\pi Q n}{N_k}} \tag{3.3}$$

where N_k is the analysis frame length for frequency bin k, w is a suitable window function (in our case a hamming window) and the digital frequency is $\frac{2\pi Q n}{N_k}$.

Figure 3.2 shows two spectrograms of an ascending chromatic scale starting on middle C (approximately 262Hz at concert pitch); the linear frequency spectrogram on the left, produced by the FFT, can be compared to the log frequency spectrogram on the right produced after converting to the constant-Q transform. It is clear to see from the figure that, for each note in the scale, the pattern of the harmonics is frequency invariant in the constant-Q spectrogram which makes it well suited to analysis of musical signals.

For the front end of our chord recognition system, we downsample[1] audio input data to 11025Hz then calculate a constant-Q transform across four octaves between $f_{\min}$=110Hz (A2) and $f_{\max}$=1760Hz (A6) with $\beta = 36$. We have implemented the constant-Q transform in code using the efficient algorithm described by Brown and Puckette in [BP92]. The Brown and Puckette algorithm employs a fast Fourier transform (FFT), the output of which is matrix multiplied by a set of complex frequency kernels

[1] We chose to downsample the original audio to 11025Hz in order to reduce the processing time required for our experiments. In doing so we make an assumption that frequencies above 5512.5Hz do not contain chordal information.

to convert to a log frequency scale. Using this algorithm, to obtain a constant-Q analysis with these parameters an FFT window length of 8192 samples is required. This frame size is necessary because Q complete cycles at frequency f_k must be evaluated in order to distinguish between f_k and f_{k+1}. In our 36 bins per octave system, at least 52 complete cycles of $f_{\min}$ (110Hz) are therefore necessary which is approximately 5211 samples at 11025Hz. The next power of 2 above 5211 is 8192 therefore we choose this value for the FFT window size. This is approximately 740ms which is a relatively long analysis window in terms of musical harmony. Thus, to improve time resolution, we use a hop size of $\frac{1}{8}$th of a window length between frames giving a resolution of 93ms per frame.

Harmonic Pitch Class Profile

The next stage in our system is decomposing the constant-Q spectra into a *Harmonic Pitch Class Profile* (HPCP) [Gó6]. The HPCP discards the pitch height information from the log frequency spectrum to produce a one octave feature vector representing pitch chroma (as discussed earlier in section 2.1.3). A HPCP vector $\mathbf{H}$ may be calculated from a constant-Q spectrum in the following way:

$$\mathbf{H}_b = \sum_{g=0}^{G-1} |\mathbf{Q}_{(b+\beta.g)}| \qquad \text{for} \quad 0 \leq b < \beta \tag{3.4}$$

where g is the octave index, G is the total number of octaves in the constant-Q spectrum at β bins per octave and b is the HPCP bin index. The magnitude of the constant-Q bins sharing the same pitch chroma are summed forming a HPCP vector with β bins. In our case, $\beta = 36$ and $G = 4$.

The justification for using this decomposition technique in a chord recognition system is that it removes the problem of having to deal with different voicings of the same chord. Figure 3.4 shows three alternative voicings of a C major triad chord[2]. These are: root position **I**, comprising

[2]These three voicings are shown as examples and are by no means an exhaustive list, there are in fact 1540 possible three-note voicings for a C major chord on a standard 88-key piano keyboard.

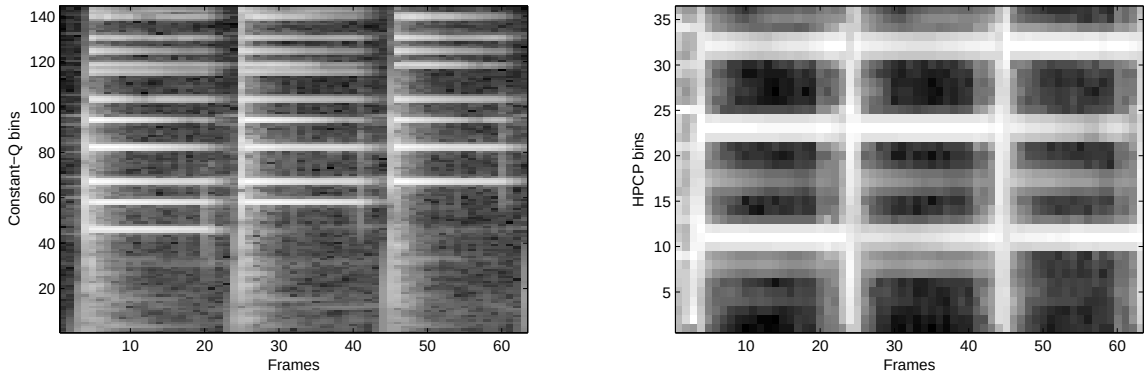


Figure 3.3: Constant-Q spectrogram (left) of three different inversions of a C Major chord compared to the HPCP (right) for the same audio signal. Higher energy is shown as lighter greyscales.

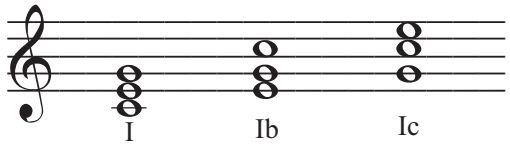


Figure 3.4: Three inversions of a C Major chord shown in the constant-Q and HPCP of figure 3.3.

pitches C4, E4 and G4; a first inversion **Ib**, comprising pitches E4, G4 and C5 and a second inversion **Ic**, comprising pitches and G4, C5 and E5. The left pane of figure 3.3 shows the four-octave constant-Q spectrogram for a piano recording of the chords in figure 3.4 where the changing voicing and the resultant pattern of harmonic overtones can be seen clearly. The right hand pane of figure 3.3 shows the HPCP vectors for the same three chords. Pitch height information discarded, the HPCP shows the three chords have very similar chromatic features. Bins corresponding to chord tones show up strongly for each one with root note C being bin 11, major third E bin 23 and perfect fifth G bin 32. Harmonic overtones of the three notes which do not add to the energy in the bins of the chord tones themselves can also be seen in the HPCP as lower energy components in other bins. The strongest of these are bins 8 and 17 which correspond to the chroma

of tones B and D respectively. B is the 3rd harmonic of chord tone E and the 5th harmonic of chord tone G; D is the 3rd harmonic of G. In this example, the energy we see in bin 8 is all contributed by the 3rd harmonic of pitch E4 which is only present in the first two voicings of the chord. This is because the top frequency $f_{\max}$ of the constant-Q spectrogram is 1760Hz (pitch A6) which is lower than the 5th harmonic of G3 which would correspond to approximately 1960Hz (pitch B6). The energy in bin 17 is contributed by the 3rd harmonic of G3 which is approximately 1176Hz (pitch D6) and is therefore present in the HPCP for all three voicings.

Use of the HPCP enables us to reduce the dimensionality of the chord recognition problem. However, the trade off is that we cannot distinguish between pitches in different octaves so an extended chord, for example one containing a major ninth interval, will appear the same as a more dissonant clustered chord containing a major second (chords discussed previously in section 2.2.2).

Tuned chromagram

For musical audio that is guaranteed to be at concert pitch (i.e. reference pitch A4 = 440Hz), a 12 bin per octave HPCP would be ideal for musical analysis. In general however, we cannot guarantee that audio we analyse from real recordings will be at concert pitch which is why we calculate the HPCP with $\beta = 36$. This is certainly true for the Beatles albums which we use as our test set in this work (discussed in chapter 6), as these contain a wide variety of tuning frequencies for different songs. To make chord recognition simpler, we wish to obtain a 12 bins per octave feature vector. The next stage of our system is therefore a tuning algorithm that will allow us to convert the 36 bin HPCP to a tuned 12 bin chromagram.

Our tuning process first requires peak picking of the 36-bin HPCP. The peak picker is a simple algorithm in which a HPCP bin $\mathbf{H}_b$ is considered to be a peak bin if

$$\mathbf{H}_b > \mathbf{H}_{(b+1 \mod 36)} \quad \text{and} \quad \mathbf{H}_b > \mathbf{H}_{(b-1 \mod 36)}. \tag{3.5}$$

Peak picking is followed by application of a quadratic interpolation [Mol04] to obtain peak positions and values (see figure 3.5).

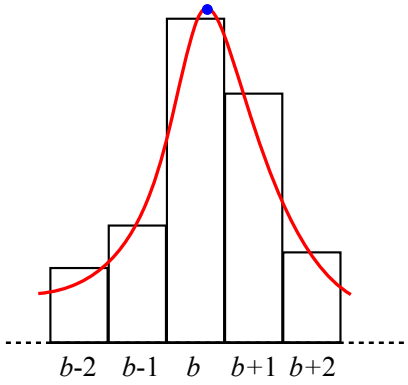


Figure 3.5: Diagram to show HPCP peak bin b and a curve fitted to the peaks used to interpolate the peak position and value.

Once the peaks have been located, we calculate the modulo 3 position values, these being equivalent to the position of each peak within one semitone. The upper plot in figure 3.6 shows the HPCP for the first 30 seconds of *Another Crossroads* by Michael Chapman. The first half of the excerpt is an instrumental introduction and the second half has the vocals from the first verse. The lower plot of figure 3.6 shows the modulo-3 HPCP peak positions for each frame. In this plot we can see that most peaks are close to zero, suggesting that the song is tuned at, or near, concert pitch. There are more 'untuned' peaks in the second half of the excerpt due to the entry of the vocals and a change in the drum pattern, however the majority still lie around zero. To identify the tuning frequency for an audio file, we calculate a histogram of the modulo 3 peak position values and find the index of the maximum peak frequency. Figure 3.7 shows the tuning histogram for the audio example from figure 3.6 where the index of the maximum peak frequency is zero. After identifying the tuning, we then discard all peaks that fall outside ± 0.2 semitones of the tuning reference in each pitch class. The remaining peaks are then allocated to 12 pitch class bins, bin centres being aligned with the tuning reference to form a 12-bin quantised chromagram as originally described in our paper [HS05]. Figure 3.8 shows the quantised chromagram for the excerpt from *Another Crossroads* by Michael Chapman.

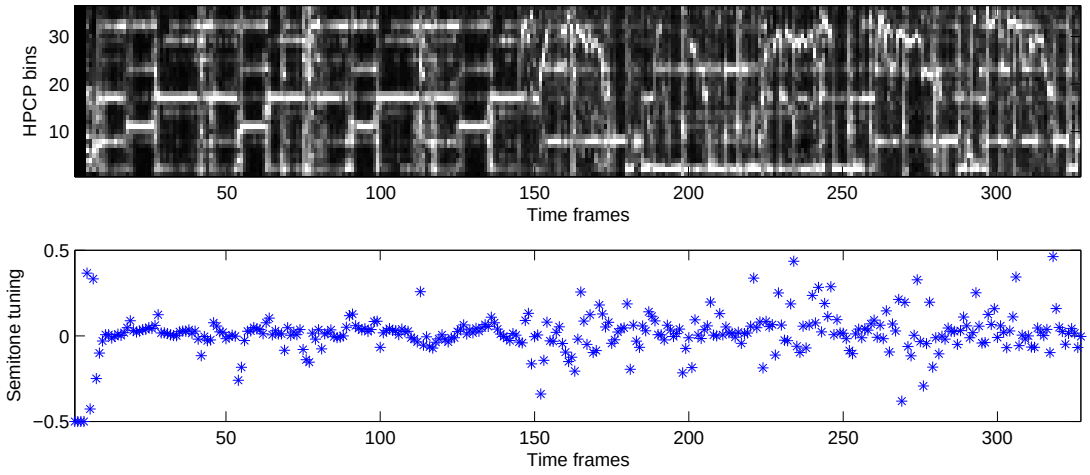


Figure 3.6: Upper plot shows a 36-bin HPCP for the first 30 seconds of *Another Crossroads* by Michael Chapman (higher energy is denoted by lighter greyscale intensity). The lower plot shows the semitone tuning of the HPCP peaks (with respect to A4 = 440Hz) for the same audio excerpt.

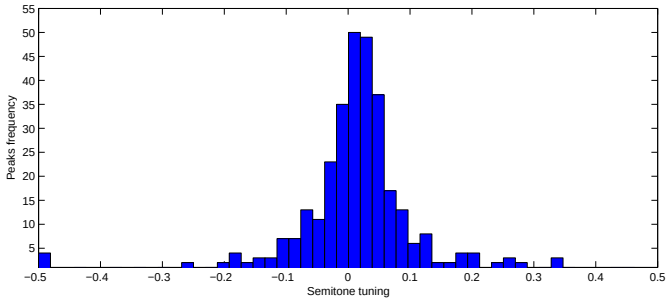


Figure 3.7: Histogram of the semitone tuning values for *Another Crossroads* by Michael Chapman. The histogram has a clear peak at just above zero showing that the tuning of the audio is close to concert pitch.

Chord recognition using pitch class templates

The next step of the process is to try to identify chord symbols from the quantised chromagram features. To do this, we use a simple system of binary chord templates which are compared with each chroma vector and the best match is recorded as the estimated chord symbol.

The chord symbols that our system is designed to recognise are major, minor, augmented and diminished triads. These four triads can be described with the binary pitchclass patterns shown in table 3.1. We form

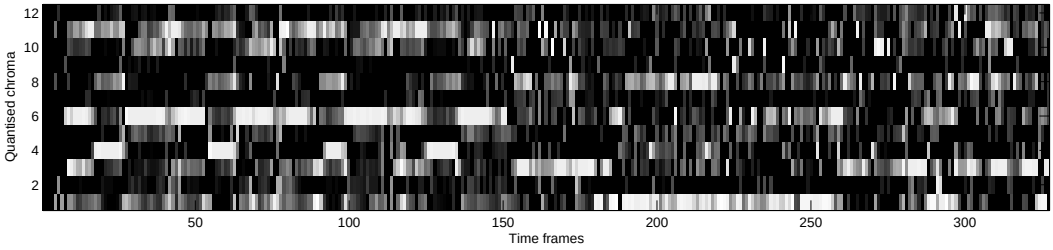


Figure 3.8: 12-bin quantised chromagram for excerpt of *Another Crossroads* by Michael Chapman. Higher energy is denoted by lighter greyscale intensity.

Table 3.1: Binary templates for the four triad types recognised by our system.

Triad	**Pitchclass template**
Major	**1** 0 0 0 **1** 0 0 **1** 0 0 0 0
Minor	**1** 0 0 **1** 0 0 0 **1** 0 0 0 0
Augmented	**1** 0 0 0 **1** 0 0 0 **1** 0 0 0
Diminished	**1** 0 0 **1** 0 0 **1** 0 0 0 0 0

a normalised template matrix $\mathbf{T}$ by concatenating the twelve different rotations of each of these bit patterns as shown in figure 3.9. The twelve rotations of four patterns make up a 12 by 48 matrix to which we add a final 49th column in which all elements are equal to identify non-chordal material, which we label '`N`'. To identify which chord is present in a particular quantised chroma vector $\mathbf{c}$, we multiply the chroma vector by the template matrix $\mathbf{T}$

$$\mathbf{W} = \mathbf{c}.\mathbf{T} \tag{3.6}$$

producing a vector $\mathbf{W}$ containing weights for each of the 49 possible chord symbol candidates. We then find the index of the maximum value in $\mathbf{W}$ to provide us with a numeric chord estimate.

Using the first 30 seconds of *Another Crossroads* again as an example, we will use a hand-transcribed chord annotation for the excerpt to compare with the output of the chord recogniser. We visualise the chord sequence as a line graph showing time against chord type which corresponds to the numeric estimates produced by our chord recogniser as shown in figure 3.10.

For our basic system, we perform chord recognition on a frame by frame

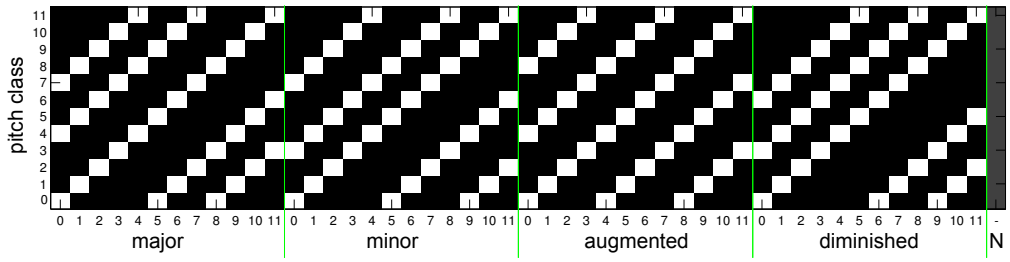


Figure 3.9: Bit patterns in the chord templates matrix.

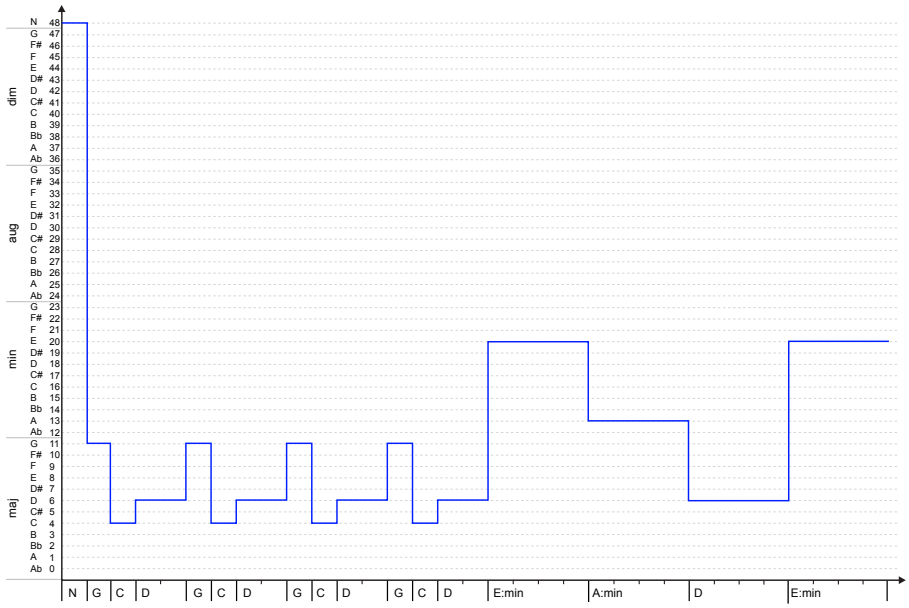


Figure 3.10: Line graph showing the hand annotated chords for the first 30 seconds of *Another Crossroads* by Michael Chapman. Gradations on the x-axis are half bars (two crotchet beats).

basis. Estimating the chord sequence directly from the quantised chroma frames produces a lot of incorrectly estimated frames due to transients and noise. Output of the system compared to the hand annotation can be seen in figure 3.11a. To reduce the problems caused by transients, we apply a low pass filter to the chromagram to pre-smooth the pitchclass information over time. This produces a cleaner output function as shown in figure 3.11b. After calculating the chord estimate values for each frame, we then use a median filter [Tuk71, Pra01] to reduce short spurious changes in the estimated sequence. Figure 3.11c shows the median filtered output of the sequence from the raw chromagram in figure 3.11a and figure 3.11d

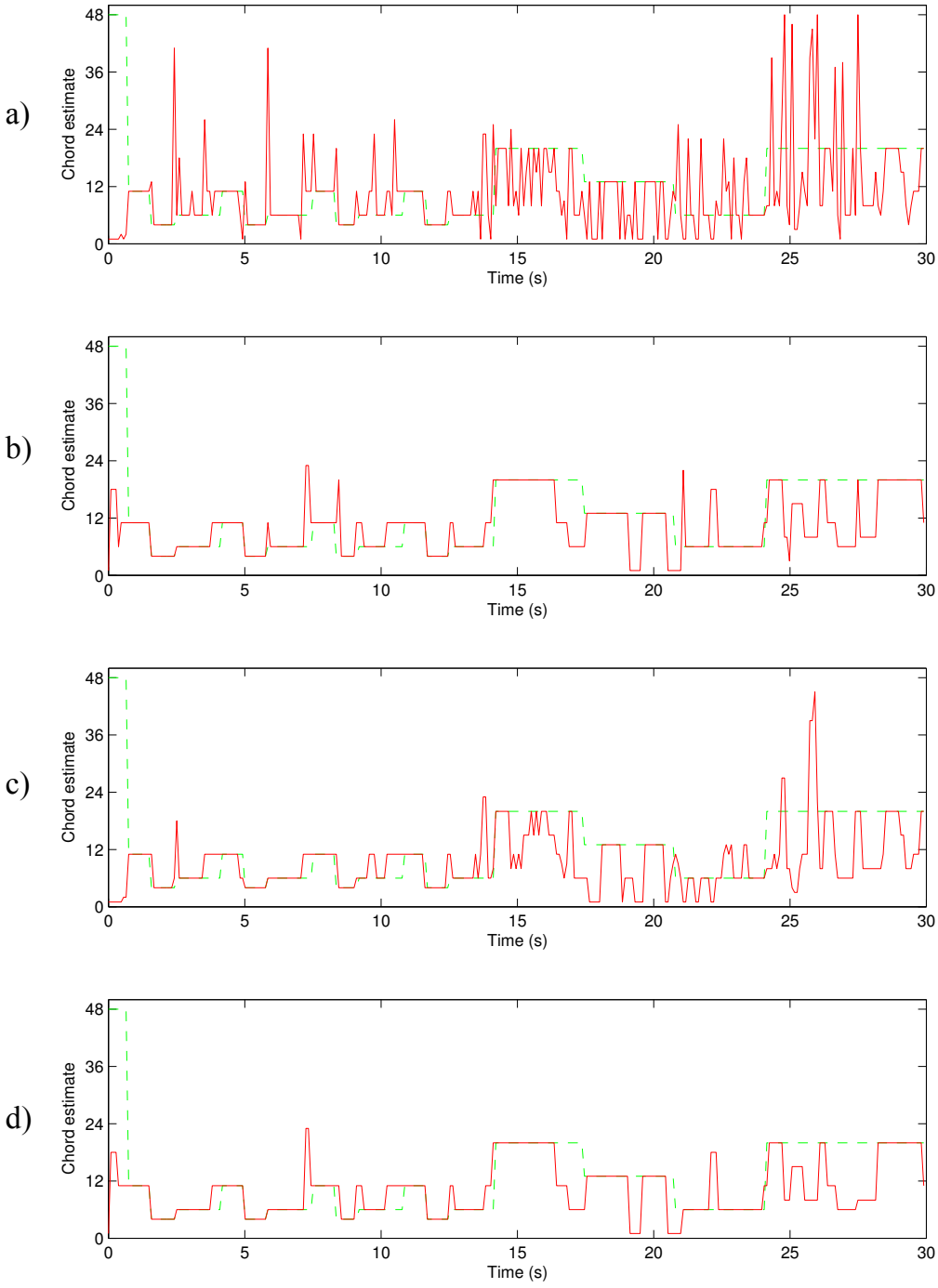


Figure 3.11: Line plots of chord recogniser output (red solid) compared with hand annotation (green dashed) for a) basic frame by frame analysis, b) chroma frames pre-filtered with a low pass function, c) median filtered frame by frame chord estimates and d) chroma frames pre-filtered with low pass function then estimates median filtered. The cleanest output is achieved by using low pass filtering and median filtering together.

shows the median filtered output of the sequence from the low pass filtered chromagram in figure 3.11b. As the figure shows, using both filtering techniques produces the cleanest estimated sequence. After calculating the numeric chord estimate values, we may use a lookup table to allocate text chord labels to the chords in order to produce a final estimated output sequence.

3.2 Improved system using segmentation algorithms

The output of the basic chord recognition system is not as good as we would like because, even after filtering, the results are still very unstable compared to the ground truth annotations. This instability is due to the frame-by-frame nature of the algorithm in which noise or transient signals in single frames can lead to many spurious outputs from the chord recogniser. To tackle this problem we have developed a chord segmentation algorithm that identifies possible chord boundaries and pre-segments the chromagram before the chord recognition stage (see figure 3.12). The chord segmentation system uses the *harmonic change detection function* (HCDF) originally presented in our paper [HSG06] which is based on the *tonal centroid* function derived from the six dimensional model for pitch space described in section 2.4.

3.2.1 Tonal centroid

In section 2.4, we introduced a six dimensional model for equal tempered pitch space in which the twelve pitch classes enclose a hypertorus with the circle of fifths wrapping around its surface three times. By using the 6-dimensional interior space contained by the surface of the hypertorus, we may apply the same technique that Chew uses to develop the 'centre of effect' in the Spiral Array to this equal tempered model for pitch space. In this case we derive a six dimensional *tonal centroid* point in the space by applying a transform function to a quantised chroma vector. As discussed in section 2.4, the six dimensional space can be visualised as the three circles in figure 3.13 where the tonal centroid for chord A major is shown.

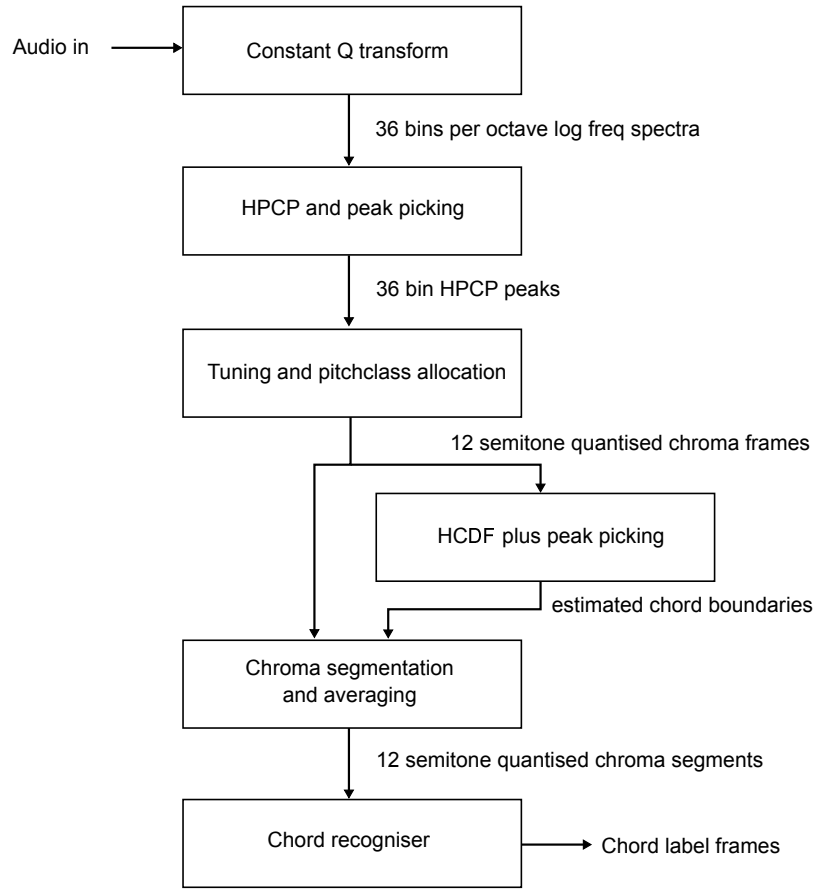


Figure 3.12: Improved chord recognition system using HCDF to segment the quantised chroma features before the chord recogniser stage.

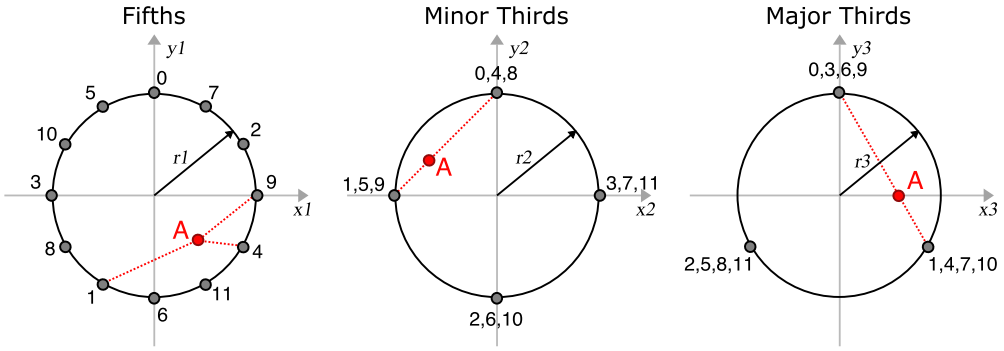


Figure 3.13: Visualising the the six-dimensional tonal space as three circles. Circles left to right: Fifths, Minor thirds and Major thirds. The Tonal Centroid for chord A Major (pitch classes 9,1 and 4) is shown at point A

The six dimensional tonal centroid vector, ζ_n, for time frame n is given by the multiplication of the 12-bin chroma vector, $\mathbf{c}$, and a transformation matrix Φ. To prevent numerical instability and ensure that the tonal centroid always lies within the 6D polytope we divide the result by the L_1 norm of $\mathbf{c}$:

$$\zeta_n(\varrho) = \frac{1}{||\mathbf{c}_n||_1} \sum_{l=0}^{11} \Phi(\varrho, l)\mathbf{c}_n(l) \qquad \begin{matrix} 0 \leq \varrho \leq 5 \\ 0 \leq l \leq 11 \end{matrix} \tag{3.7}$$

where l is the chroma vector pitch class index and ϱ denotes which of the six dimensions of ζ_n is being evaluated. The transformation matrix Φ represents the basis of the 6D space described in section 2.4 and is given as:

$$\Phi = [\phi_0, \phi_1 \ldots \phi_{11}] \tag{3.8}$$

where

$$\phi_l = \begin{bmatrix} \Phi(0,l) \\ \Phi(1,l) \\ \Phi(2,l) \\ \Phi(3,l) \\ \Phi(4,l) \\ \Phi(5,l) \end{bmatrix} = \begin{bmatrix} r_1 \sin l\frac{7\pi}{6} \\ r_1 \cos l\frac{7\pi}{6} \\ r_2 \sin l\frac{3\pi}{2} \\ r_2 \cos l\frac{3\pi}{2} \\ r_3 \sin l\frac{2\pi}{3} \\ r_3 \cos l\frac{2\pi}{3} \end{bmatrix} \qquad 0 \leq l \leq 11 \tag{3.9}$$

The values r_1, r_2 and r_3 are the radii of the three circles in figure 3.13. To ensure that the distances between pitch classes in the 6-D space correspond to our perception of harmonic relations between pitches (i.e. that the fifth is the closest relation followed by the major third then the minor third and so on) we set the r_1, r_2 and r_3 to 1, 1 and 0.5 respectively as derived in section 2.4.1.

3.2.2 Harmonic change detection function

To reduce the effects of transient frames, the sequence of tonal centroid vectors is convolved with a 19-point Gaussian with σ value of 8 in a row-by-row fashion (i.e. the individual dimensions are smoothed over time with a cutoff frequency of approximately 1Hz). We define the HCDF, ξ, as the overall rate of change of the smoothed tonal centroid signal. ξ_n is the

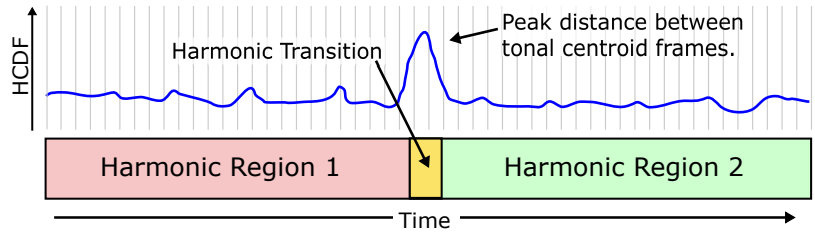


Figure 3.14: Harmonic change detection function (HCDF). Time frames are depicted by light vertical lines.

Euclidean distance between the smoothed tonal centroid vectors $\hat{\zeta}_{n-1}$ and $\hat{\zeta}_{n+1}$ (equation 3.10) where $\hat{}$ denotes vectors from the Gaussian-smoothed signal. Peaks in this signal indicate transitions between regions that are harmonically stable (see figure 3.14); an approach inspired by Chew's key modulation finding algorithm described in [Che02].

$$\xi_n = \sqrt{\sum_{d=0}^{5} \left[\hat{\zeta}_{n+1}(\varrho) - \hat{\zeta}_{n-1}(\varrho)\right]^2} \tag{3.10}$$

The HCDF has been implemented in Matlab as part of our chord recognition system and has also been implemented in C++ by Martin Gasser as a visualisation plug-in for Sonic Visualiser [CLSB06].

3.2.3 Chord segmentation with the HCDF

In order to use the HCDF for chord segmentation, we apply a peak picking algorithm to it to identify potential chord boundaries in the audio. Figure 3.15 shows the HCDF generated in Sonic Visualiser for a 13 second excerpt of *Being For The Benefit Of Mr Kite!*, by the Beatles, with peaks identified. The figure shows larger peaks that coincide with chord changes but smaller peaks are also present that may be caused by other harmonic changes such as movement in the bass line. Some other small spurious peaks appear on the sides of larger maxima. Using the peaks from the HCDF as estimated boundary positions that define chord segments, we are able to calculate average chroma vectors for each segment. These average chroma vectors may then be processed using the chord recognition algorithm, which produces an estimated chord sequence that is more stable than the frame based approach.

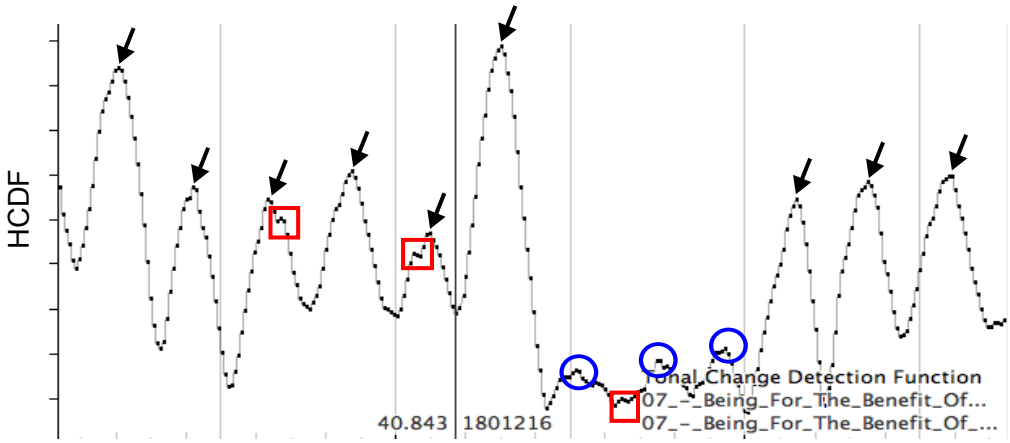


Figure 3.15: HCDF for extract of the Beatles' *Being For The Benefit Of Mr Kite!* (time 35.5 to 48.5 seconds). Arrows show peaks corresponding to chord changes, circles show peaks caused by movement in the bass line and boxes show false positives.

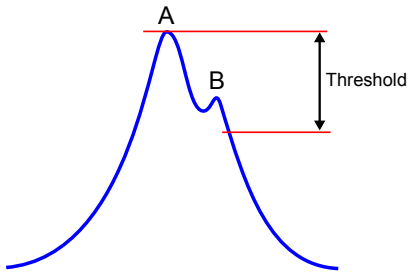


Figure 3.16: Two peaks close together. With the enhanced peak picker, peak A will be recorded as a boundary but peak B will be discarded because it is within the hysteresis threshold and is thus judged to be a spurious peak.

Improved chord segmentation

The HCDF is a nice visualisation for harmonic change in musical signals but spurious peaks in the signal mean that simple peak picking produces over-segmented results when compared with hand annotated chord changes. To reduce the number of false positives we can apply a slightly more complex peak picking algorithm to the HCDF which discards small peaks. Our enhanced peak picking algorithm uses a threshold to introduce hysteresis into the peak detection process so that small peaks and double peaks are likely to be ignored (see figure 3.16). Using this enhanced peak picker we obtain improved results, although they are still over-segmented

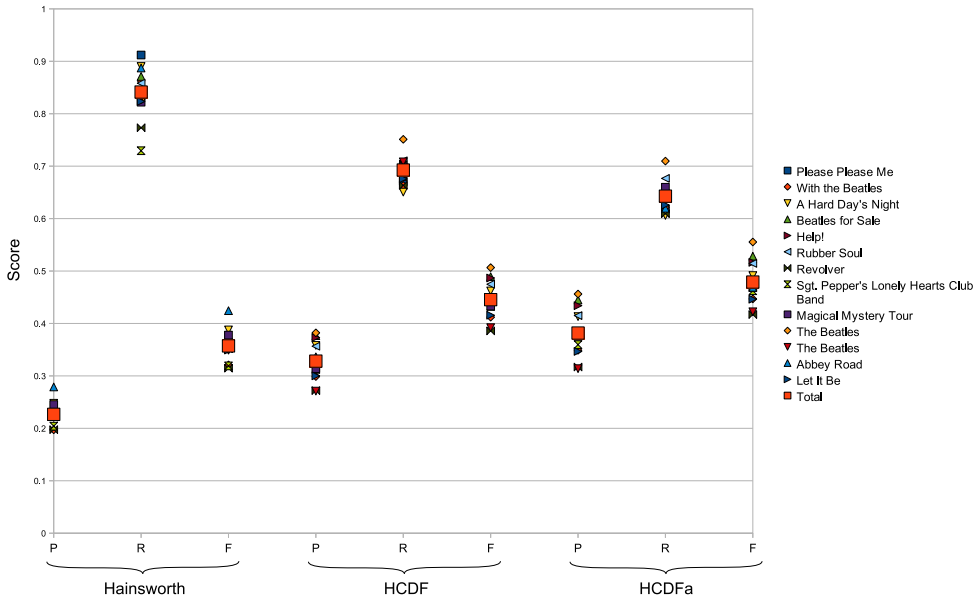


Figure 3.17: Plot showing precision, recall and f-measure results for the Hainsworth MacLeod, HCDF and HCDFa harmonic segmentation algorithms for the twelve Beatles albums.

compared to our ground truth chord annotations.

3.2.4 Analysis of HCDF performance

To quantitatively test how the HCDF performs as a chord segmentation algorithm we analysed the Beatles album collection for which we have chord transcription files (as discussed in chapter 6). HCDF peak times are compared against the times of chord changes in the transcriptions.

We present results for the standard peak picking algorithm (which we will simply label HCDF) and the enhanced peak picking algorithm (which we will label HCDFa). We also show results for a harmonic onset detection algorithm by Hainsworth and Macleod [HM03] for comparison (which we will label HM). The HM algorithm is a two-stage process in which peaks are detected in a spectral distance measure; the resulting segments are analysed for their harmonic content so that similar contiguous segments can be joined back together. The HCDF algorithm does not use such a second analysis step to obtain its results.

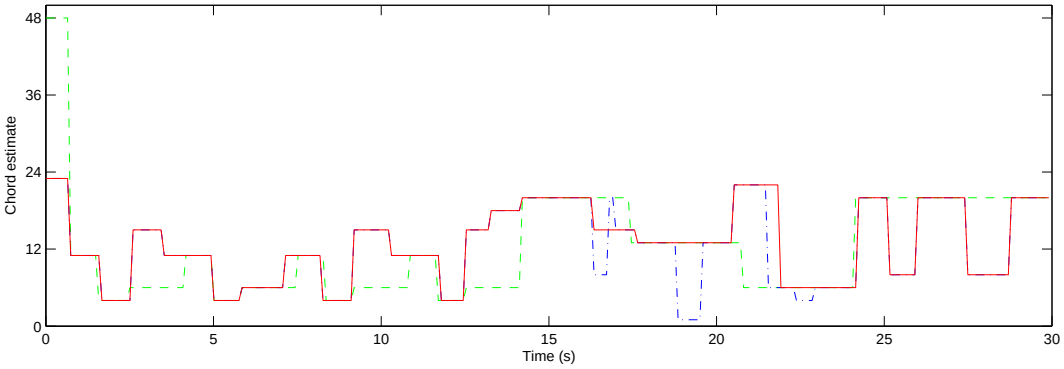


Figure 3.18: Line plot showing estimated chord sequences generated by the HCDF (blue dash and dotted line) and HCDFa (red solid line) compared with the annotated sequence (green dashed line) for the first 30 seconds of *Another Crossroads* by Michael Chapman.

The results for the experiment are shown in table 3.2 and shown graphically in figure 3.17. We defined a hit as a match within ±3 analysis frames. The three performance measures used here are *Precision* P, the ratio of hits to detected changes; *Recall* R, the ratio of hits to transcribed changes and the *f-measure* F which combines the two [Dix06] using the harmonic mean:

$$F = \frac{2RP}{R+P} \tag{3.11}$$

The HM algorithm detects any changes in the harmonic content of the signal and is not specifically designed to find chord boundaries. The results show that it achieves high recall scores around 84% but low precision scores around 22%. This gives an overall f-measure of 36% because it over detects when evaluated against annotated chord boundaries. The basic HCDF algorithm scores lower on recall with 69% but does better on precision with 33% so its f-measure is higher than HM at 45% because it is better at discriminating chord boundaries. The HCDFa algorithm improves further on the standard HCDF with 64% recall and 38% precision giving a final f-measure of 48%.

Using the HCDF and HCDFa algorithms for chord segmentation before the chord recognition stage, we get a cleaner estimated chord sequence with fewer short length spurious chords. However, this does mean that when the algorithm picks an incorrect chord, the erroneous chord symbol

will last for the whole duration of the segment. Figure 3.18 shows the output sequences generated by HCDF and HCDFa for the first 30 seconds of *Another Crossroads* by Michael Chapman. Both HCDF algorithms produce the same sequence for the first half of the excerpt but the HCDFa produces better results for the second half. The output sequences for both are more stable than those for the basic chord recognition algorithm but, as the figure shows, in some cases the chord recogniser has chosen an incorrect chord based on the average chromagram for each segment.

Full results and analysis for the three chord recognition algorithms described in this chapter are given in chapter 9 after consideration of the test set and evaluation mechanisms in chapters 6 to 8.

Table 3.2: Harmonic onset detection results: Recall (R), precision (P) and f-measure (F) for Hainsworth Macleod harmonic change algorithm (HM) compared with our peak picked HCDF and enhanced peak picking HCDFa algorithms. HM has higher precision scores than the HCDF algorithms but its recall score is low because of over detection. The HCDFa algorithm has the best recall and also best score for the combined f-measure.

Disc	Album title	P_{HM}	R_{HM}	F_{HM}	P_{HCDF}	R_{HCDF}	F_{HCDF}	P_{HCDFa}	R_{HCDFa}	F_{HCDFa}
1	Please Please Me	0.23	0.91	0.36	0.33	0.67	0.44	0.38	0.62	0.47
2	With the Beatles	0.2	0.86	0.32	0.3	0.66	0.41	0.35	0.62	0.45
3	A Hard Day's Night	0.25	0.89	0.39	0.36	0.65	0.46	0.41	0.61	0.49
4	Beatles for Sale	0.24	0.87	0.37	0.38	0.7	0.49	0.44	0.65	0.53
5	Help!	0.22	0.84	0.35	0.37	0.7	0.49	0.43	0.64	0.52
6	Rubber Soul	0.22	0.86	0.35	0.36	0.71	0.47	0.42	0.68	0.51
7	Revolver	0.2	0.77	0.31	0.27	0.66	0.39	0.32	0.61	0.42
8	Sgt. Pepper's Lonely Hearts Club Band	0.2	0.73	0.32	0.31	0.7	0.43	0.36	0.65	0.46
9	Magical Mystery Tour	0.25	0.82	0.38	0.31	0.7	0.43	0.38	0.66	0.48
10CD1	The Beatles	0.22	0.83	0.35	0.38	0.75	0.51	0.46	0.71	0.56
10CD2	The Beatles	0.22	0.83	0.35	0.27	0.71	0.39	0.31	0.65	0.42
11	Abbey Road	0.28	0.89	0.42	0.34	0.68	0.45	0.38	0.62	0.47
12	Let It Be	0.22	0.82	0.35	0.3	0.68	0.42	0.35	0.63	0.45
Total	All albums	0.23	**0.84**	0.36	0.33	0.69	0.45	**0.38**	0.64	**0.48**

Chapter 4

Representing chords in plain text

In this chapter we will discuss the issues associated with representing chord symbols in plain text. Our motivation for the work in this chapter was the lack of a standard formal way to represent chord labels in plain text annotations. The challenges associated with this issue are discussed in detail in section 4.1. We then propose a specification for such a representation in section 4.2.1 and using this specification we go on to develop a logical model and a corresponding text syntax for chord labels in sections 4.2.2-4.3. The syntax has been specifically designed to be intuitive for human users but also machine readable so that annotations may be made easily by hand but also used in computer analysis.

To make the chord symbol syntax convenient for use in experiments, a toolkit for manipulating the labels has been written in Matlab. These tools are briefly discussed in section 4.4.

Chapter Contribution

The chord symbol syntax described in this chapter provides a well defined, flexible system for labelling any chromatic chord[1] that may be encountered in music. Developing this syntax has allowed the hand annotation of the Beatles collection to be possible in an unambiguous format and thus provided a large machine-readable data set for evaluating chord recognition algorithms. The transcription process and the resulting collection of annotated data are discussed in chapter 6. The syntax is now used by many

[1] That is to say, we assume that the chord can be defined in terms of western tonal harmony and thus does not include microtonal elements.

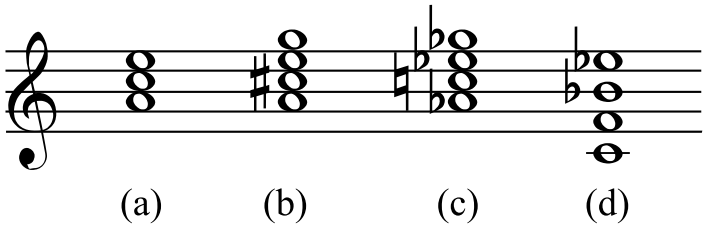


Figure 4.1: Some example chords which we may wish to label with plain text: a) 'A minor', b) 'A seven', c) 'A♭ seven', d) a quartal chord on C.

researchers involved with the MIREX chord detection track [mirb] and other transcription collections have also been made using it as part of the OMRAS II Metadata Project [MCD+09].

The logical model for chords that underpins the chord syntax (discussed in section 4.2.2) has also been used as the basis for the RDF chord ontology[2] [SRMH07] as part of the OMRAS II project.

4.1 Problems for plain text chord symbols

In this section we will look at some issues encountered when trying to express chord symbols in plain text. To start with, we will present some examples that illustrate the potential problems with ad hoc chord labelling methods.

4.1.1 Annotation style ambiguity

There are many different styles that can be used for hand writing chord notation [Cok64, Gre71]. Depending what type of musical background someone has (e.g. classical, jazz, pop etc.) they may have very different ways of notating the same chords. Indeed, even musicians from within the same genre will sometimes use different styles of chord notation depending on which school's methodology they have studied.

Consider a simple A minor triad chord comprising notes A, C and E as shown in figure 4.1a. In classical harmony analysis, minor chords are usually denoted using lower case characters so the A minor chord might

[2]URL http://motools.sourceforge.net/chord_draft_1/chord.html

simply be labelled '`a`'. In jazz, a superscript dash is quite a common notation for minor chords (e.g. A^{-}) so in plain text one might write '`A-`'. In the case of pop music, many guitar tabs and song books e.g. [Roo00] use a lower case 'm' to denote minor so the chord might be written '`Am`'. In some cases people also use the abbreviation 'min' for minor family chords so we could alternatively have '`Amin`'. With such wide variation in styles it is sometimes difficult to know what someone else's chord symbols mean if they do not explicitly specify their labelling conventions.

4.1.2 Undefined syntax ambiguity

We may be presented with the following chord symbol in plain text:

```
Ab7
```

What is this chord? If we assume that the character '`b`' denotes the flat symbol ♭, we can parse it two ways which give us very different results. Is it: '`A`∼`b7`' or '`Ab`∼`7`'? In the first instance we will assume that the root is A and that the chord is a major triad plus a flattened seventh comprising notes A, C♯, E and G as shown in figure 4.1b. However, in the second instance we could read it as an A♭ seventh chord which, if there is no key context ambiguity to also take into account (see section 4.1.4), we may assume to be notes A♭, C, E♭ and G♭ as shown in figure 4.1c giving us a chord of the same shape but all the notes shifted down one semitone.

We can see from this example that without a clear definition of the syntax for a chord symbol, we are unable to know its precise meaning.

4.1.3 Capitalisation and character choice ambiguity

When people wish to write sharp ♯ or flat ♭ signs in plain text, the characters '`#`' and '`b`' are often used. This choice is intuitive for musicians because of those characters' close resemblance to these musical symbols. In some notations lower case letters may be used to signify minor chords [Pol] and this introduces another possible ambiguity where a B minor chord may be written '`b`' but the flat sign is also '`b`'. In this case, detecting errors in a transcription becomes difficult. For example, a transcription might

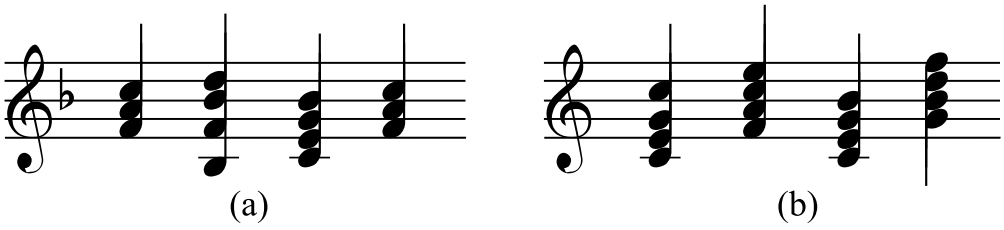


Figure 4.2: Two example chord sequences in different keys containing a diatonic seventh chord built on C: a) is in F major so the C chord is a dominant seventh, b) is in C major so the C chord is a major seventh.

contain the symbol '`b7`'. It is not possible to tell if this was truly intended to be a B minor 'seven' chord or if, for example, it might have been meant to be part of a 'X♭' seven chord symbol from which some root note 'X' has accidentally been omitted by the transcriber.

Another example of where capitalisation can lead to a difficulty in error detection is where capital '`M`' is used to signify major family chords and lower case '`m`' is used to signify minor family chords. In this case, a slip of the shift key can alter the family of the chord and because the result will also be a valid chord, there is no way to know that this has happened. In terms of making chord labels machine readable, it is important to try to avoid such potential sources of confusion.

4.1.4 Key context ambiguity

In a text file we may be presented with the following chord symbol:

```
C7
```

What chord might this symbol represent? Given this symbol alone, most musicians would probably assume that this is a 'C seven' chord (C, E, G, B♭). However, if this symbol is written as part of a classical harmony analysis [Tay89, Tay91, Tag03] then it may well be key context dependent. In this case we have a potential ambiguity in the meaning of the chord label. To illustrate, consider the two chord sequences shown in figure 4.2. The sequence in figure 4.2a might be written this way in plain text:

F major: `F Bb C7 F`

In terms of functional harmony [Tay91, Tag03, Sch54] this sequence is **I IV V7 I** in the key of F major. Here, '`C7`' represents a diatonic seventh chord built on C, the dominant note of the key, so the chord is a 'dominant seventh' chord comprising notes C, E, G and B♭.

Now consider the sequence in figure 4.2b. This time we are in the key of C major but the harmonic analysis written in plain text will contain the same chord symbol:

C major: `C F7 C7 G7`

In the new key, the functional harmony of the sequence is **I IV7 I7 V7**. This means that the '`C7`' symbol now represents a diatonic seventh chord built on the tonic (or root) note of the key so it will be a 'major seventh' chord comprising the notes C, E, G and B.

We can see in this example that one chord symbol may mean two different things depending on the key context in which it has been used. This is potentially a very big problem if there is any possibility of the vital key context information being separated from the chord symbols themselves. If this should occur, the meaning of the chord symbols can become ambiguous.

4.1.5 Chord label semantics

Some choices for chord labels can imply more information than is actually intended. A good example of this is the way that the 'dominant seventh' chord label is often inappropriately applied to chords which do not function as dominants.

In classical harmony, a dominant seventh chord is a diatonic chord built on the dominant scale degree in a given key context. In a major key, this equates to a chord comprising the 5^{th}, 7^{th}, 9^{th} and 11^{th} degrees of the major scale. These form a major triad plus a flattened seventh i.e. the chord intervals relative to the dominant chord root note are 1, 3, 5, ♭7.

Consider the chord sequence in figure 4.3a. We may see the chords in this sequence written in plain text the following way:

D major: `G Adom7 D`

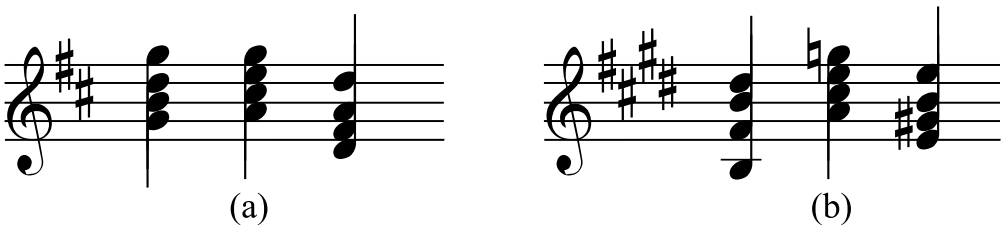


Figure 4.3: Two example chord sequences in different keys containing an A seven chord: a) is in D major so the A chord is a 'dominant seventh', b) is in E major so the A chord is a subdominant major chord with a flattened seventh.

In D major, the functional harmony of this sequence is **IV V7 I**. Because A is the dominant of D, a seventh chord built on A will be a *dominant seventh* chord comprising notes A, C♯, E and G. In this instance, labelling the chord '`Adom7`' may be considered appropriate because it is the dominant seventh chord in the given key context.

Now consider the chord sequence in figure 4.3b. The plain text representation of this sequence may also contain the same chord label because the second chord is made up of the same notes as the A chord in figure 4.3a:

E major: `B Adom7 E`

In this example however, we are now in the key of E major so the functional harmony of this sequence is **V IV(♭7) I**. In E major, A is the subdominant so the chord we wish to describe is not made entirely of diatonic notes from the current key context (G♮ is not diatonic in E major) and is *not* a dominant chord. It is correct to say that the chord has the same *shape* as a dominant seventh chord (i.e. 1, 3, 5, ♭7) but to label it as a dominant chord is incorrect in terms of harmony theory. Thus the label 'dominant seventh' is not technically correct here and a more appropriate term in this tonal context would probably be a 'chromatic seventh'.

This particular example is very common in the literature [MD08, OGF09c, Gre71], with many papers and books using the term 'dominant seventh' as a convenient generic name to describe this chord shape. In terms of the semantics of the chord label however, using this name can add an unintentional element to the meaning of the chord by implying a particular harmonic function that the chord may not actually possess. We assert

that this kind of 'semantically loaded' chord label is therefore unsuitable for the purposes of chord annotation.

In Jazz, chord labels are not key context dependent and the dominant seventh chord shape is simply known as the 'seven chord'. In this case, the chord in the two example sequences would be labelled A^{7}. In contrast to context dependent classical harmony analyses, the different types of seventh chord in jazz have separate chord symbols, for example the major seventh might be labelled A^{M7} or $A^{\triangle 7}$ and the minor seventh might be labelled A^{m7} or $A^{\triangledown 7}$. In plain text therefore, these three chords, the 'seven', 'major seven' and 'minor seven', might be written '`A7`', '`AM7`' and '`Am7`' respectively to distinguish clearly between them. In Jazz, it is understood that these symbols do not imply any specific harmonic function. Thus, in the case of the two example chord sequences, a label such as '`A7`' would be more appropriate than '`Adom7`'. This labelling would however rely on the understanding that chord symbols must therefore be specified to be context independent.

4.1.6 Inflexible labelling models

If a chord labelling scheme defines a list of chord labels that are 'allowed', what happens when you find a chord that does not fit in the list?

As an example, let us imagine a chord labelling scheme that allows triadic chords to be labelled so it assumes that all chords are based on either a major, minor, diminished, augmented or a suspended triad. For the majority of western tonal music, this system is an adequate model.

Now suppose that we have a piece of music that contains the chord shown in figure 4.1d. This chord is built entirely of perfect fourth intervals so it is not triadic and does not fit within the normal rules of western harmony. Such chords are quite common in modern jazz and are often referred to as *quartal* chords.

How can we represent this non-triadic chord within this triad-based scheme? The chord is not triadic so it does not fit the underlying model for the labelling scheme. One solution might be to define a way of modifying the chord label so we could say that this chord is based on a suspended

fourth triad type (diatonic intervals 1, 4, 5) but that it omits the 5th and adds a ♭7 and a ♭10. In doing this it appears that we have solved the problem, however we now run into the issue of semantics once again.

By specifying the chord as one of the triad types, we instantly load the chord with the label 'suspended' whether we want to or not. Because the scheme prescribes the use of a triad type as the basis for all chord labels we have no choice but to select the 'best of a bad bunch' out of the labelling options and then try to alter it to fit the new chord. This becomes a particular problem when using computers to automate analysis of annotated data sets. An algorithm running through a transcription using this scheme might only look for the triad type and ignore all other information in which case it would analyse this chord incorrectly.

4.2 Development of a chord symbol syntax

In this section we will describe the process we used to develop a flexible, machine readable yet musically intuitive chord symbol syntax. First, we outline the specification for such a syntax in section 4.2.1. In section 4.2.2 we define a logical model for describing chords. In section 4.2.3 we then develop a plain text representation for that model to which we add a system of 'shorthand' labels in section 4.2.4. We then define a formal syntax for our chord symbols in Backus Naur Form in section 4.3.

4.2.1 Specification

As we saw in the examples from section 4.1, there are many issues that need to be addressed in the specification of a flexible chord syntax for plain text. The syntax must be:

- **Unambiguous**: chord symbols must always be explicit in their meaning and it should only be possible to interpret them in one way.
- **Context independent**: chord symbols must not depend on other information such as a key context or their relative position in a progression for their precise meaning to be known. Each chord symbol

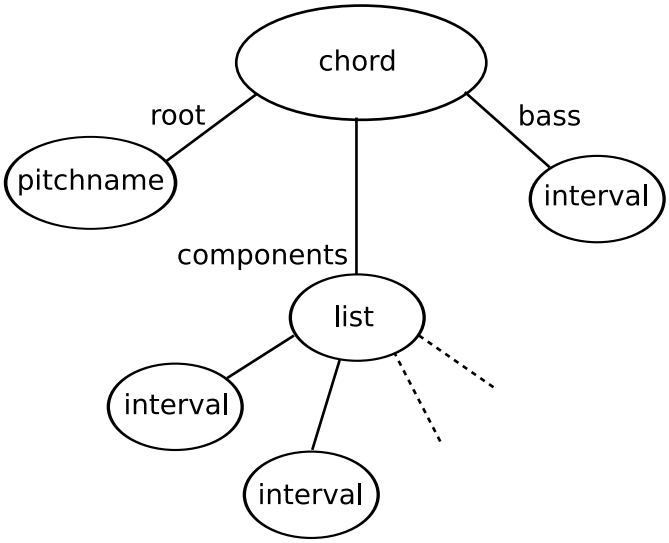


Figure 4.4: Model for chord definition

should be self contained so that it can be read and fully understod on its own.

- **Machine readable**: chord symbols should be easy to parse using computers so that they will be convenient for use in automatic data processing.

- **Human readable**: the symbols should be easy to read for human users as well as machines. The symbols should be intuitive for musicians so that they are easy to use when doing transcriptions by hand.

- **Flexible**: the syntax should allow for any chord that might be encountered. It is important that the system should not restrict the user, for example by forcing a triadic representation as seen in section 4.1.6.

4.2.2 Logical model

We will represent a chord by its three main attributes: the root, the list of component intervals and its bass note. The root is defined as a *pitchname* element which has an absolute value. The list of component intervals and

the bass note are defined as *intervals*, relative to the root. A diagram illustrating this model is shown in Figure 4.4.

Seven *natural* pitch names are defined (letters A to G as shown in eqn. 4.1), which correspond to the white keys on a piano keyboard. We define a set of *degrees*[3] in eqn. 4.2 which correspond to the degrees of the diatonic major scale (i.e. they represent either major or perfect intervals). Degrees can technically be any positive integer value greater than or equal to 1 (the tonic). In practice it is rare that a chord would ever need to contain a degree higher than a 13th (one octave plus a sixth above the tonic). To allow correct spelling of enharmonics, two *modifier* operators, **sharp** and **flat**, are included. Thus:

$$\textit{natural} = \{\text{A}|\text{B}|\text{C}|\text{D}|\text{E}|\text{F}|\text{G}\} \tag{4.1}$$

$$\textit{degree} = n \text{ where } \{n \in \mathbb{N} : n > 0\} \tag{4.2}$$

$$\textit{modifier} = \textbf{sharp} \mid \textbf{flat} \tag{4.3}$$

To obtain the full range of possible pitchnames and intervals, naturals and degrees may be operated on by the sharp and flat modifiers. In this way, *pitchnames* and *intervals* may be defined as:

$$\textit{pitchname} = \textit{natural} \mid \textit{modifier}(\textit{pitchname}) \tag{4.4}$$

$$\textit{interval} = \textit{degree} \mid \textit{modifier}(\textit{interval}) \tag{4.5}$$

The definitions of pitchname and interval are recursive allowing for different enharmonic spellings. For example, the pitchname 'C♭♭' may be modelled as the natural 'C' flattened then flattened again:

$$\text{C}\flat\flat = \textbf{flat}(\textbf{flat}(\text{C})) \tag{4.6}$$

An example model of a chord is shown in Figure 4.5. The chord in the example is a C minor-seventh chord in first inversion. The root of this chord is pitchname C. The component intervals are the root, a minor third, a perfect fifth and a minor-seventh (1, ♭3, 5, ♭7). In the case of this

[3]Please note that the definitions of *degree* and *interval* given here have been reversed compared to those defined in the ISMIR05 paper [HSAG05] to fit better with the usual meanings of these words in music theory. The term 'note' has been replaced by 'pitchname' for the same reason. This work supersedes the proposals in that paper.

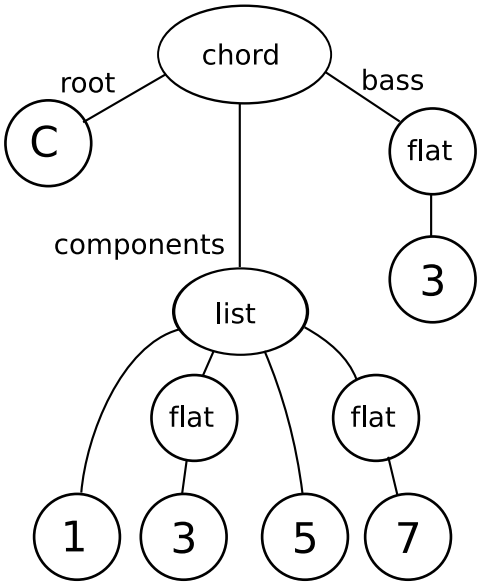


Figure 4.5: Example model of a first inversion C minor-seventh chord

example chord, these component intervals correspond to the pitchnames C, E♭, G and B♭ (see table 4.1). The bass note of a minor seventh chord in first inversion is the minor 3rd, which in this example is pitchname E♭.

Because the model is designed such that all components are relative to the root, it is easy to change this chord to any other minor-seventh chord simply by changing the value of the root pitchname. For example, if we were to change the root to a D♭, we can work out what the correct spellings of the pitchnames should be in the D♭ minor-seventh chord as shown in the middle section of table 4.1. We can see that a D♭ minor-seventh chord should contain the pitchnames D♭, F♭, A♭ and C♭. Likewise, if we were to take C♯ (the enharmonic equivalent of D♭) we can use the same process to derive the correct spellings of the pitchnames in the C♯ minor-seventh chord (see the bottom of table 4.1). In this example we can see that the pitchnames of the C♯ minor-seventh chord are C♯, E, G♯ and B. It is also worth noting that flattening a pitchname which is operated on by a sharp modifier (as in the case of the E♯ and B♯ in our C♯ minor-seventh chord) will cancel that sharp modifier:

$$\text{C}\sharp\flat = \textbf{flat}(\textbf{sharp}(\text{C})) = \text{C} \tag{4.7}$$

Table 4.1: Pitchnames for the C, D♭ and C♯ minor-seventh chords

Root	**Interval**	**(Degree,**	**Modifier)**	**Pitchname**
C	1	1 = C	none	C
	♭3	3 = E	flat	E♭
	5	5 = G	none	G
	♭7	7 = B	flat	B♭

Root	**Interval**	**(Degree,**	**Modifier)**	**Pitchname**
D♭	1	1 = D♭	none	D♭
	♭3	3 = F	flat	F♭
	5	5 = A♭	none	A♭
	♭7	7 = C	flat	C♭

Root	**Interval**	**(Degree,**	**Modifier)**	**Pitchname**
C♯	1	1 = C♯	none	C♯
	♭3	3 = E♯	flat	E
	5	5 = G♯	none	G♯
	♭7	7 = B♯	flat	B

and likewise:

$$\text{C}\flat\sharp = \textbf{sharp}(\textbf{flat}(\text{C})) = \text{C} \tag{4.8}$$

As the preceding examples demonstrate, the sharp and flat modifiers allow proper enharmonic spelling of pitchnames and intervals. This is particularly important in cases such as the diminished seventh chord (comprising the musical intervals 1, ♭3, ♭5, ♭♭7) which contains a diminished seventh interval (a major seventh interval flattened twice). Although this interval is tonally equivalent to a major sixth in equal tempered tuning (i.e. they would be the same note when played on a piano keyboard), it has a different function and meaning in music theory and as such the model has to be able to represent them both individually:

$$6 \neq \flat\flat 7 \tag{4.9}$$

likewise absolute pitchnames which are enharmonic equivalents can be distinguished from each other:

$$\text{C} \neq \text{D}\flat\flat \tag{4.10}$$

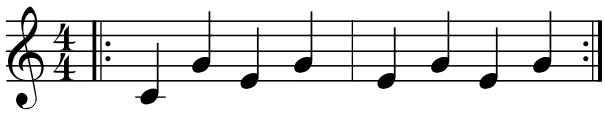


Figure 4.6: A short musical example where both bars are likely to be annotated as a C major chord although the second bar does not contain the root note C

In our ISMIR paper from 2005 [HSAG05], the chord model that we proposed assumed that the root pitchname would always be present in any chord that would be annotated and it was thus omitted from the interval list. It was subsequently found that there are in fact instances where one might wish to annotate a certain section of music as being a particular chord where its root is strongly implied by the music even though it is not actually present. A simple example of this is shown in Figure 4.6. In the absence of any other tonal context, both bars in the example are likely to be annotated as being the chord C major (which contains notes C, E and G) even though the second bar only contains the 3rd and 5th intervals from that chord. As the aim of this part of the work is to provide as flexible a chord model as possible for annotation, the assumption that the root will always be present is therefore invalid so if the root of the chord is present we now explicitly include it in the interval list.

4.2.3 Developing a syntax for plain text

With the model described in section 4.2.2, we can now represent any type of chord. We now need to define a plain text representation of that model. Thus we define the following syntax for a chord symbol:

```
root :  (interval1, interval2...)  / bass
```

The root pitchname is written first followed by a colon ‘`:`’ separator. A comma delimited list of intervals contained in parentheses is then written. This interval list notation is quite similar in appearance to the ‘chord formulas’ in Ted Green’s guitar chord text book ‘Chord Chemistry’ [Gre71] and also to John Wade Ulrich’s syntax for jazz chord analysis [Ulr77]. Finally, an optional interval may be added at the end after an oblique stroke (the forward slash character ‘`/`’) to denote the bass note if it is

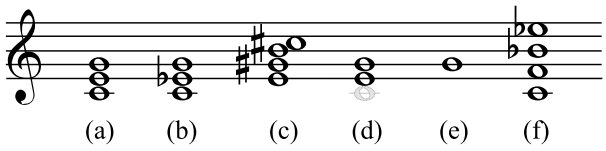


Figure 4.7: Six example chords: a) C major, b) C minor, c) C♯ minor-seventh in first inversion, d) C major with omitted root, e) Unison on G and f) a quartal chord on C

different to the root. The naturals, intervals and modifiers are defined in Table 4.3 following equations 4.1 to 4.3. The sharp and flat are signified by the hash symbol '`#`' and the lowercase '`b`' respectively.

To keep the notation musically intuitive, pitchname modifiers are placed after naturals so A♭ becomes '`Ab`'. For the same reason, interval modifiers are placed before degrees so a flattened seventh interval (♭7) is represented as '`b7`'. An extra chord state denoted by a single uppercase '`N`' is also added to signify 'no chord' marking silence or untuned, possibly percussive musical material. To resolve the possible ambiguity between the natural '`B`' and the flat modifier '`b`' the notation is necessarily case sensitive.

Following this syntax, all chords may now be described in plain text in an unambiguous manner. Figure 4.7 shows six chords which we will now describe with the new syntax. The C major chord (Figure 4.7a) can be represented as:

```
C:(1,3,5)
```

Likewise, the C minor chord (Figure 4.7b) can be written as:

```
C:(1,b3,5)
```

The C♯ minor-seventh chord in first inversion discussed in the earlier example in section 4.2.2 (see Figure 4.7b) may be represented as:

```
C#:(1,b3,5,b7)/b3
```

It is worth noting that the C♯ minor-seventh chord in first inversion could alternatively be labelled as an E major-sixth chord instead because both comprise the same four pitchnames (E, G♯, B, C♯). The only difference

between the two is which pitchname is assumed to be the root of the chord. Taking E as the root we may write the chord in this way:

```
E:(1,3,5,6)
```

Which chord spelling to use is a decision to be made by the annotator (be it a human or machine). The important point is that the syntax allows for both spellings.

This syntax also allows for the case where we have a chord with an implied root note (such as the C major chord in the example from the second bar in Figure 4.6). In the example of the C major chord where the root is not voiced (Figure 4.7d) we can represent it in this way:

```
C:(3,5)
```

This syntax is flexible and allows for many unusual cases that may need to be labelled. For example, if an ensemble all play one note in unison where there is no other tonal context with which to label the chord[4] (for example all voices play the note G as shown in Figure 4.7e), we may label it simply:

```
G:(1)
```

We may also freely label chords which do not fall into the usual triadic harmony categories common in western classical and popular music. Such chords include quartal harmonies quite often found in modern jazz. For example, a quartal chord built upon C (made up of the notes C, F, B♭, E♭ as shown in Figure 4.7f) can be represented in the following way:

```
C:(1,4,b7,b10)
```

4.2.4 Shorthand

In the previous section we described a plain text syntax for representing the chord model of section 4.2.2. This syntax, although fully capable of describing any chord we might wish to label, is still not as readily readable to

[4]It is acknowledged that a single musical note should not really be considered to be a 'chord' as such. However, it is a case which we wish to be able to annotate clearly in order to distinguish it from muliple-note chords or non-tonal events.

the human user as we would like. We must remember that the motivation for developing this chord labelling syntax is to provide a system with which large musical collections may be easily (and efficiently) labelled by hand in a machine readable format. To this end, it is necessary to introduce a further element to the syntax: we will define a list of memorable *shorthand* labels for the most commonly occurring chords. These shorthand labels are effectively a set of chord-type 'mnemonics' which map directly to pre-defined interval lists for the most common chordtypes. This makes the syntax easy for musicians to read and to write quickly by hand. Thus the syntax for a chord may also take the form:

`root : shorthand (extra-intervals) / bass`

where `shorthand` is the chord mnemonic for a given chordtype and `(extra-intervals)` is an optional further list of intervals. Provision for extra intervals in parentheses is left so that additional intervals may be added to common chords. To make the shorthand system more flexible a special 'omit interval' symbol, an asterisk `*`, is also added to denote a missing interval from a shorthand notated chord.

To illustrate, the chord C minor may be represented '`C:(1,b3,5)`' as it is built of the tonic note C, the minor third E♭ and the perfect fifth G. This chord has root C and is of type 'minor' which is always made up of the intervals 1, ♭3 and 5. Therefore we may define a shorthand mnemonic (in the case of minor we choose the shorthand '`min`') that maps to the interval list '`(1,b3,5)`'. In doing so we can now use the new shorthand to label our chord:

`C:min` → `C:(1,b3,5)`

Likewise, we will choose the label '`min7`' to denote a minor-seventh chord which implies the interval list '`(1,b3,5,b7)`'. Therefore a D♭ minor-seventh chord could be written:

`Db:min7` → `Db:(1,b3,5,b7)`

If we wish to alter the D♭ minor-seventh chord such that the 5th is no longer present but a minor 9th interval is added (i.e. the interval list becomes '`(1,b3,b7,b9)`' we could write this using the omit interval symbol

to remove the 5th and adding the ♭9 interval to the extended interval list:

`Db:min7(*5,b9)` → `Db:(1,b3,b7,b9)`

where:

`min7` → `(1,b3,5,b7)`

and

`(*5,b9)` → omit `5`, add `b9`

Choice of shorthand labels

As discussed in section 4.1.1, there are many different ways in which the same chord-type may be written and it is often quite difficult to find two musicians who agree on exactly the same conventions for all labels. It is therefore an almost impossible challenge to find a list of shorthand mnemonics that will keep all of the users happy all of the time. For this reason we try to follow a practical, pragmatic approach and give some level of justification here for the choices that have been made.

In many classical music analyses available in the form of plain text from the internet and elsewhere, authors often use the convention of uppercase letters for the root to denote major chords and lower case letters to denote minor chords (for example '`C`' for C major and '`c`' for C minor). We cannot use this convention because we have already specified that the model differentiates between the root note of the chord and other information about that chord. The system must also differentiate between the natural name '`B`' and the flat modifier '`b`' so using upper and lowercase letters for root notes would lead to problems in terms of ambiguity (see discussion in section 4.1.3) and also increase the complexity of software required to decode the syntax.

In plain text, it is quite common to see the chord family labelled with single letters for example: major '`M`', minor '`m`', augmented '`+`' and diminished '`o`' as an approximation of what would be written by hand if using pen and paper. Although we could use short labels such as these, we have chosen to use more verbose shorthands here because longer labels

Table 4.2: Definition of shorthand notations for common chords and their implied interval lists

Chordtype		**Shorthand**	**Interval List**
Triad Chords:	Major	`maj`	`(1,3,5)`
	Minor	`min`	`(1,b3,5)`
	Diminished	`dim`	`(1,b3,b5)`
	Augmented	`aug`	`(1,3,#5)`
Seventh Chords:	Major Seventh	`maj7`	`(1,3,5,7)`
	Minor Seventh	`min7`	`(1,b3,5,b7)`
	Seventh	`7`	`(1,3,5,b7)`
	Diminished Seventh	`dim7`	`(1,b3,b5,bb7)`
	Half Diminished Seventh	`hdim7`	`(1,b3,b5,b7)`
	Minor (Major Seventh)	`minmaj7`	`(1,b3,5,7)`
Sixth Chords:	Major Sixth	`maj6`	`(1,3,5,6)`
	Minor Sixth	`min6`	`(1,b3,5,6)`
Extended Chords:	Ninth	`9`	`(1,3,5,b7,9)`
	Major Ninth	`maj9`	`(1,3,5,7,9)`
	Minor Ninth	`min9`	`(1,b3,5,b7,9)`
Suspended Chords:	Suspended 2nd	`sus2`	`(1,2,5)`
	Suspended 4th	`sus4`	`(1,4,5)`

are clearer in their meaning and it also makes typographic errors in annotations easier to detect. For example, the difference (and typographic distance) between '`C:M`' and '`C:m`' is not as large as that between '`C:maj`' and '`C:min`'.

The naming scheme that we have adopted for the shorthands is close to the names commonly used in Jazz music (see section 2.2.2). Table 4.2 gives the list of shorthand label definitions that are currently defined for the system. Obviously, this list could be expanded in the future if it is felt that it is necessary to include another chordtype. The syntax itself does not need to be altered to accept other shorthand labels.

Consideration of shorthand semantics

Before moving on from this section, it is important to consider what the shorthand labels we have defined actually *mean*. As discussed in section 4.1.5, we should be careful to take into account the semantic meaning conveyed by our chord symbol shorthands and not just the set of intervals they represent. It is dangerous to assume that because a shorthand label implies a given interval list, then that interval list must therefore imply that label. For example, the label '`min7`' implies the interval list '`(1,b3,5,b7)`'. However, we cannot use this implication to assume that the interval list '`(1,b3,5,b7)`' must also imply the label '`min7`'. We know that a minor-seventh chord is made up of the intervals 1, ♭3, 5 and ♭7 therefore it is correct to say:

$$\texttt{min7} \rightarrow \texttt{(1,b3,5,b7)}$$

however, when given a list of intervals is it not necessarily safe to say the opposite:

$$\texttt{(1,b3,5,b7)} \nrightarrow \texttt{min7}$$

It is important to avoid getting into the situation where a chord is labelled with an inappropriate shorthand which is then subsequently altered to make the chord reflect the transcribed music properly. As an example, it would be *possible*, but wholly inappropriate, to use a major-seventh shorthand '`maj7`' to label a chord built upon the root note C with following interval list: 1, ♭3, 5, ♭7. This, as we have seen in previous examples, is clearly the same intervals as a C minor-seventh chord yet we could represent it using this chord label:

$$\texttt{C:maj7(*3,b3,*7,b7)} \rightarrow \texttt{C:(1,b3,5,b7)}$$

Here, we deliberately omit the major third and major seventh intervals implied by the shorthand and add in the missing minor third and minor seventh intervals. This chord label evaluates to exactly the same root and interval list as the more appropriate chord label:

$$\texttt{C:min7} \rightarrow \texttt{C:(1,b3,5,b7)}$$

Table 4.3: Syntax of Chord Notation in Backus-Naur Form

```
<chord>      ::=    <pitchname> ":" <shorthand> ["("<ilist>")"]["/"<interval>]
                  | <pitchname> ":" "("<ilist>")" ["/"<interval>]
                  | <pitchname> ["/"<interval>]
                  | "N"

<pitchname>  ::=  <natural> | <pitchname> <modifier>

<natural>    ::=  "A" | "B" | "C" | "D" | "E" | "F" | "G"

<modifier>   ::=  "b" | "#"

<ilist>      ::=  ["*"] <interval> ["," <ilist>]

<interval>   ::=  <degree> | <modifier> <interval>

<degree>     ::=  <digit> | <digit> <degree> | <degree> "0"

<digit>      ::=  "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

<shorthand>  ::=   "maj" | "min" | "dim" | "aug" | "maj7" | "min7" | "7"
                  | "dim7" | "hdim7" | "minmaj7" | "maj6" | "min6" | "9"
                  | "maj9" | "min9" | "sus2" | "sus4"
```

The semantic meaning of these two labels are very different however. To the computer, the two labels evaluate to be the same chord when following the logical model described in section 4.2.2; to the human reader, the major-seventh shorthand label distorts the meaning of the implied interval list because the label implies that the chord is major even though the actual interval list describes a minor chord.

4.3 Formal definition of chord label syntax

At this point, we now have a well defined model for representing any chord. We have also defined a way of representing this model in plain text and introduced a convenient shorthand notation for common chord labels to make the representation easier to read and write for musicians.

Before formally defining the syntax, there is one final point to include in the text representation: in the majority of hand written chord notation systems, a root pitchname written on its own (i.e. one with no other

markings attached to it to denote a chordtype) is assumed to be a major chord. To keep the system as intuitive as possible for musicians, we will follow this convention here so we may state:

$$\texttt{C} \equiv \texttt{C:maj} \rightarrow \texttt{C:(1,3,5)}$$

This convention still naturally allows for alternative bass intervals to be used with major chords labelled in this way. For example, a second inversion C major chord can be written as:

$$\texttt{C/5} \equiv \texttt{C:maj/5} \rightarrow \texttt{C:(1,3,5)/5}$$

We can now strictly define the syntax for the chord labels using Backus-Naur Form (BNF) [LM81]. Table 4.3 shows the full BNF definition for the syntax. The main symbol `<chord>` is defined as one of four alternatives:

1. A root pitchname followed by a shorthand, optional interval list (`<ilist>`), and optional bass interval.
2. A root pitchname followed by an interval list and optional bass interval.
3. A root pitchname followed by an optional bass interval.
4. An uppercase '`N`' to denote a no-chord state.

The interval list symbol `<ilist>` is a recursive definition for a comma-delimited list of intervals that can be any arbitrary length. The `<shorthand>` symbol is a member of the list of all shorthand symbols defined in table 4.2. The other symbols are BNF expressions for the definitions of pitchnames, naturals, modifiers, intervals and degrees[5] given in section 4.2.2, equations 4.1 to 4.5.

With this formal BNF definition of the syntax it is now a relatively simple task to write computer programs that deal with such chord symbols. It is also possible to transcribe the chords for large audio data sets in a consistent way using this system.

[5]It should be noted that although the BNF definition allows a degree to have an arbitrary number of digits, in reality there is no need for a degree to ever have more than two.

4.4 A reference software toolkit

We have specified a logical model for representing chords and defined a clear syntax for expressing that model in plain text. It is necessary to write software that can interpret and manipulate these chord symbols in order for the system to be useful in practical applications. In this section we will briefly describe a reference software toolkit for manipulating chords that conform to the syntax described in sections 4.2 and 4.3. The programming language chosen for the reference implementation is Matlab because that is the language used for most of the experimental work reported in this thesis. It would not be difficult, however, to transfer the algorithms used in the toolkit into other languages. Comprehensive help and comments on how to use the toolkit are provided in the Matlab code.

The chord toolkit has been released as open source software under the GPL and the source code can be downloaded from the Isophonics webpage[6].

Data types

The toolkit uses a few basic data types. There are several string-types which follow the BNF syntax definition closely including chord, interval, pitchname and natural. There are also several integer types that allow for the chord symbols to be translated from text strings into other musically useful forms. These include scale degrees, accidentals, semitones and pitchclasses. More detailed information on these data types can be found in the help files available with the toolkit code.

Basic chord symbol parsing functions

The basic functions included in the toolkit enable the user to extract various types of musical information from chord symbols or convert other musical information into text chord symbols easily. All the functions in the toolkit perform error checking on their input arguments and can warn the user if the data is incorrect. Five main functions allow basic information to

[6] http://www.isophonics.net/content/reference-annotations-beatles

be extracted from a chord symbol. These are: `parsechord`, `parsenote`, `parseinterval`, `shorthand2intervals` and `parseintervals`. Full descriptions of these and the other functions in the toolkit can be found in the help files available with the toolkit code.

Basic conversion functions

The toolkit provides a set of functions for converting between different types of musical information that can be extracted from chord symbols. This allows for users to convert the chord symbols in a transcription into local formats which may be more appropriate for their work. A good example of this is where the user may require pitch-class information (which assumes enharmonic equivalence) instead of the correctly spelled interval information contained in a chord symbol.

Transcription file utilities

The toolkit also includes a set of functions that allow easy reading and writing of data in transcription files (in the format described in section 6.2).

Higher level functions

The toolkit also provides higher level functions including implementations of the chord symbol comparison methods described in chapter 5 and the evaluation methods described in chapter 8. The inputs to the evaluation functions are plain text files in the same format as the transcriptions. Because of this, it should be possible for any researcher to use the toolkit to evaluate their own chord recognition algorithms regardless of what language the algorithm itself was implemented in.

Chapter 5

Chord symbol comparison methods

In this chapter we will discuss several methods for comparing chord symbols with each other. Our main motivation for investigating different chord comparison methods is to provide a formal basis for our chord symbol recall evaluation method discussed in chapter 8. In this case it is important that we can clearly define what constitutes a 'correct match' between a machine estimated symbol generated by a chord recognition algorithm and a hand annotated symbol in the ground truth test set.

The comparison methods detailed here will also be used to analyse the statistics of the Beatles chord transcription collection discussed in chapter 6.

Chapter contribution

The chord comparison methodology described in this chapter provides a foundation for evaluation metrics we present later in chapter 8. This approach to evaluation using clearly defined parametrised chord matching methods allows proper like-for-like comparison between different chord recognition algorithms. As such, it is potentially very useful for other researchers in their work and also for the MIREX [mirb] chord detection track. All of the comparison methods described in this chapter have been implemented in the C4DM chord tools described in section 4.4.

5.1 Manipulating chords and other musical objects

In this chapter we will discuss a number of different chord matching functions. To explain these functions clearly we will now introduce some mathematical notation and conventions which we will use for manipulating our chords and other related musical objects.

Let $\mathtt{X}$ be a character string representing a chord symbol that complies with the syntax defined in table 4.3 on page 105. We may restate the syntax algebraically such that $\mathtt{X}$ takes the form

$$\mathtt{X} = \begin{cases} \texttt{"N"} & \text{if non-chord} \\ \mathtt{R_X} \oplus \mathtt{Q_X} \oplus \mathtt{T_X} & \text{otherwise} \end{cases} \tag{5.1}$$

where $\oplus$ is the tensor addition operator[1], $\mathtt{R_X}$ is a string representing the chord root pitchname, $\mathtt{Q_X}$ is a string containing the chordtype information (i.e. shorthand, intervals, extensions etc) and $\mathtt{T_X}$ is a string containing the bass interval information if one is specified. Therefore $\mathtt{Q_X}$ will be of the form

$$\mathtt{Q_X} = \begin{cases} \texttt{":"} \oplus \mathtt{S_X} \oplus \texttt{"("} \oplus \mathtt{L_X} \oplus \texttt{")"} & \text{if } |\mathtt{L_X}| > 0 \\ \texttt{":"} \oplus \mathtt{S_X} & \text{if } |\mathtt{S_X}| > 0 \text{ and } |\mathtt{L_X}| = 0 \\ \text{empty} & \text{otherwise} \end{cases} \tag{5.2}$$

where $\mathtt{S_X}$ is a shorthand string[2] and $\mathtt{L_X}$ is a list of $M_\mathtt{L}$ interval strings $\mathtt{i}_m$ of the form

$$\mathtt{L_X} = \begin{cases} \mathtt{I}_0 \oplus \bigoplus_{m=1}^{M_\mathtt{L}-1} (\texttt{","} \oplus \mathtt{I}_m) & \text{for } M_\mathtt{L} > 1 \\ \text{empty} & \text{otherwise.} \end{cases} \tag{5.3}$$

and

$$\mathtt{I}_m = (\mathbf{o}_m \otimes \texttt{"*"}) \oplus \mathtt{i}_m \tag{5.4}$$

where $\otimes$ is the tensor multiplication operator[3], $\mathbf{o}$ is an $M_\mathtt{L}$-length binary vector such that $\mathbf{o}_m$ determines whether the interval $\mathtt{i}_m$ is omitted from

[1] e.g. $\texttt{"a"} \oplus \texttt{"b"} = \texttt{"ab"}$

[2] These are listed in table 4.2 on page 103

[3] e.g. $\texttt{"a"} \otimes 4 = \texttt{"aaaa"}$.

the chord or not. String $\mathtt{T_X}$ takes the form

$$\mathtt{T_X} = \begin{cases} \texttt{"/"} \oplus \mathtt{i}_{\text{bass}} & \text{if } |\mathtt{i}_{\text{bass}}| > 0 \\ \text{empty} & \text{otherwise} \end{cases} \tag{5.5}$$

where interval string $\mathtt{i}_{\text{bass}}$ is the bass interval that defines the bass note of the chord relative to the root pitchname $\mathtt{R_X}$.

5.1.1 Pitchnames and intervals

From the syntax definition in table 4.3, we recall that a pitchname string is defined recursively as

```
<pitchname>  ::=  <natural> | <pitchname> <modifier>
<natural>    ::=  "A" | "B" | "C" | "D" | "E" | "F" | "G"
<modifier>   ::=  "b" | "#"
```

and an interval string is defined similarly as

```
<interval>  ::=  <degree> | <modifier> <interval>
<degree>    ::=  <digit> | <digit> <degree> | <degree> "0"
<digit>     ::=  "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

We will now restate these definitions algebraically. The BNF definitions allow pitchname and interval strings to contain a variable length string of modifier characters. Therefore let $\mathtt{m}$ be a string of 0 or more modifiers where a modifier character may be a sharp (♯) denoted by the hash character '`#`' or a flat (♭) denoted by lowercase character '`b`'. We may therefore say that

$$\mathtt{m} = \begin{cases} a \otimes \texttt{"\#"} & \text{if } a \geq 0 \\ |a| \otimes \texttt{"b"} & \text{if } a < 0 \end{cases} \tag{5.6}$$

where a is the *accidental* value representing the integer number of modifier characters. If a is positive, it represents a number of sharp '`#`' characters; if it is negative, its absolute value $|a|$ denotes the number of flat '`b`' characters. If a is zero, $\mathtt{m}$ is an empty string.

String $\mathtt{m}$ may be viewed as a *multiset* [Sta97] of modifier characters described by pair (S, ν) where set $S \ni \{\texttt{"\#"}, \texttt{"b"}\}$ and operator ν denotes *multiplicity* i.e. the number of repetitions of a member of S in the multiset $\mathtt{m}$. Therefore we may calculate a for a given string $\mathtt{m}$ in the following way:

$$a = \nu_{\mathtt{m}}(\texttt{"\#"}) - \nu_{\mathtt{m}}(\texttt{"b"}) \tag{5.7}$$

That is, a is the difference between the number of sharp characters '#' and flat characters 'b' in string m.

We will now define **N** to be an ordered set of the seven natural pitchnames[4] A-G such that

$$\mathbf{N} = \{\texttt{"F"}, \texttt{"C"}, \texttt{"G"}, \texttt{"D"}, \texttt{"A"}, \texttt{"E"}, \texttt{"B"}\} \tag{5.8}$$

Therefore the general form of a pitchname string, v, will be

$$\texttt{v} = \mathbf{N}_{l_\texttt{v}} \oplus \texttt{m}_\texttt{v} \tag{5.9}$$

where $l_\texttt{v}$ is the index of the natural pitchname in set **N** and $\texttt{m}_\texttt{v}$ represents a string of modifier characters as described in equation 5.6. An interval string i may similarly be restated algebraically as

$$\texttt{i} = \texttt{m}_\texttt{i} \oplus \texttt{d}_\texttt{i} \tag{5.10}$$

where $\texttt{m}_\texttt{i}$ is a string of modifier characters as described in equation 5.6 and $\texttt{d}_\texttt{i}$ is the character string representation of a positive integer $d_\texttt{i}$ which denotes a degree of the diatonic major scale.

Vector representation of pitchnames and intervals

We know from equations 5.9 and 5.6 that a pitchname v may be represented as a natural character $\mathbf{N}_{l_\texttt{v}}$ followed by a number $a_\texttt{v}$ of modifier characters ('#' or 'b'). Therefore an alternative representation for a pitchname is the vector double

$$\texttt{v} = \{l_\texttt{v}, a_\texttt{v}\} \tag{5.11}$$

Thus pitchname 'Eb', for example, can be expressed as $\{5, -1\}$ because 'E' is element 5 of set **N** and it has one flat. Some other examples of

[4]To make the arithmetic simple when converting between intervals and pitchnames later, we order the elements of set **N** according to their positions on the line of fifths and index from 0 so $\mathbf{N}_0$ = "F", $\mathbf{N}_1$ = "C", ... $\mathbf{N}_6$ = "B". To avoid confusion, it should also be noted that we use the symbol $\mathbb{N}$ to denote the set of the natural numbers in this thesis.

pitchnames expressed in this way are:

$$\begin{aligned} \texttt{"C"} &= \{1, 0\} \\ \texttt{"F\#"} &= \{0, 1\} \\ \texttt{"G\#\#"} &= \{2, 2\} \\ \texttt{"Abb"} &= \{4, -2\}. \end{aligned} \tag{5.12}$$

In the same way, we may represent an interval `i` as a double

$$\texttt{i} = \{d_{\texttt{i}}, a_{\texttt{i}}\} \tag{5.13}$$

where $d_{\texttt{i}}$ is a degree of the diatonic major scale and $a_{\texttt{i}}$ is the number of accidentals that modify it to complete the interval. Some examples of intervals expressed this way are:

$$\begin{aligned} \texttt{"1"} &= \{1, 0\} \\ \texttt{"b3"} &= \{3, -1\} \\ \texttt{"\#4"} &= \{4, 1\}. \end{aligned} \tag{5.14}$$

Converting between pitchnames and intervals

When using the various comparison methods detailed later in this chapter, we often wish to calculate the correct enharmonic spelling of a pitchname which corresponds to an interval relative to another pitchname. This is not as straight forward as it might first appear and to find a calculation that gives a correct enharmonic spelling for any interval and pitchname combination we look to works on pitch spelling and intonation theory such as [Mer06, LH92, Reg73, Reg75, Tem01].

Figures 5.1 and 5.2 show two projections of the line of fifths with an absolute numeric index that Regener [Reg73, Reg75] calls the *quint* position[5]. The quint number line has its origin at F=0. Counting upwards one quint position ascends a perfect fifth interval and counting down descends a perfect fifth (which, assuming octave equivalence, is the same as ascending a perfect fourth). The more positive the quint number, the higher the

[5]This is similar in concept to Temperley's TPC 'tonal pitch class' [Tem01] and Longuet-Higgins' 'sharpness' [LH92] values which are alternative indices for the line of fifths [Mer06]

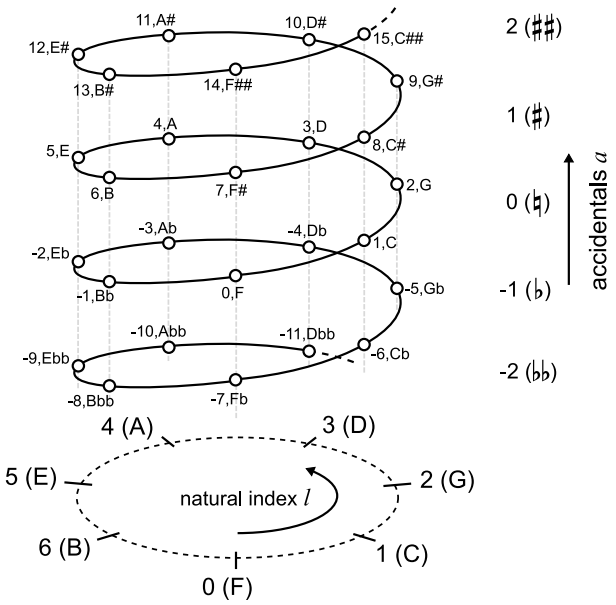


Figure 5.1: The helix of pitchnames formed when the line of fifths is wrapped around a cylinder where the seven natural names $\mathbf{N}_l$ (where $0 \leq l \leq 6$) form the circular base in the horizontal plane and the number of accidentals a form the vertical axis. The number to the left of each pitchname on the helix represents the quint position q.

Line of fifths

note n	Bbb	Fb	Cb	Gb	Db	Ab	Eb	Bb	F	C	G	D	A	E	B	F#	C#	G#	D#	A#	E#	B#	F##
quint position q	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
natural index l	6	0	1	2	3	4	5	6	0	1	2	3	4	5	6	0	1	2	3	4	5	6	0
accidentals a	-2	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2

Figure 5.2: The unwrapped line of fifths shown with the quint number line, the natural name index l and the number of accidentals a for each pitchname.

number of sharps; the more negative the quint number, the higher the number of flats. Figure 5.1 shows clearly the cyclic nature of pitchname spellings due to there being only seven natural names in set $\mathbf{N}$. Given the quint number $q_{\mathtt{v}}$ for a given pitchname $\mathtt{v}$ the index of its natural name $l_{\mathtt{v}}$ in set $\mathbf{N}$ is

$$l_{\mathtt{v}} = q_{\mathtt{v}} \bmod 7 \tag{5.15}$$

and the number of accidentals $a_{\mathtt{v}}$ is

$$a_{\mathtt{v}} = \left\lfloor \frac{q_{\mathtt{v}}}{7} \right\rfloor \tag{5.16}$$

where $\lfloor \cdot \rfloor$ denotes the floor function[6].

A pitchname has an absolute value on the quint number line and an interval is a displacement on the line. We may calculate the absolute quint position $q_{\mathtt{R}}$ of root pitchname R with the following equation:

$$q_{\mathtt{R}} = l_{\mathtt{R}} + 7a_{\mathtt{R}} \tag{5.17}$$

Similarly, the quint displacement $q_{\mathtt{i}}$ of interval i is given by

$$q_{\mathtt{i}} = \big((2d_{\mathtt{i}} - 1) \bmod 7\big) - 1 + 7a_{\mathtt{i}} \tag{5.18}$$

so to find the quint position $q_{\mathtt{v}}$ for a pitchname v corresponding to an interval i relative to a given root pitchname R, all that is necessary is to add the quint displacement of i to the quint position of R:

$$q_{\mathtt{v}} = q_{\mathtt{R}} + q_{\mathtt{i}} \tag{5.19}$$

Thus, using equations 5.15 and 5.16 we can then determine the vector double $\{l_{\mathtt{v}}, a_{\mathtt{v}}\}$ for pitchname v and then transform this to its character representation using equation 5.9.

To reverse the process and convert a pitchname v to an interval i relative to another pitchname R, we convert both pitchnames to their quint positions and take the difference to find the quint displacement $q_{\mathtt{i}}$ of the interval

$$q_{\mathtt{i}} = q_{\mathtt{v}} - q_{\mathtt{R}}, \tag{5.20}$$

then we may calculate the relative diatonic scale degree $d_{\mathtt{i}}$ of the interval with

$$d_{\mathtt{i}} = \left\lfloor \frac{q_{\mathtt{i}} \bmod 7}{2} \right\rfloor + 1 + 4\big((q_{\mathtt{i}} \bmod 7) \bmod 2\big) \tag{5.21}$$

and the accidentals with

$$a_{\mathtt{i}} = \left\lfloor \frac{q_{\mathtt{i}} + 1}{7} \right\rfloor \tag{5.22}$$

to obtain the vector representation of interval i as described in equation 5.13.

[6] i.e. round down to the closest integer below the given value.

Converting pitchnames to pitchclasses

We frequently wish to deal with enharmonic equivalents of pitchnames and intervals in our manipulation of chord symbol information. To convert a pitchname `v` into its equivalent pitchclass $\mathrm{p}_{\mathtt{v}}$ we use the elements of its vector double representation $\{l_{\mathtt{v}}, a_{\mathtt{v}}\}$. Pitchclass $\mathrm{p}_{\mathtt{v}}$ may take values in the range 0 to 11 and we use pitchname `"C"` as our absolute pitchclass reference 0. Thus pitchclass $\mathrm{p}_{\mathtt{v}}$ may be calculated as

$$\mathrm{p}_{\mathtt{v}} = \begin{cases} \big((l_{\mathtt{v}} - 1) + a_{\mathtt{v}} \bmod 12\big) & \text{if } (l_{\mathtt{v}} \bmod 2) = 1 \\ \big((l_{\mathtt{v}} + 5) + a_{\mathtt{v}} \bmod 12\big) & \text{otherwise} \end{cases} \tag{5.23}$$

and the relative pitchclass displacement (i.e. the number of semitones modulo 12) equivalent to interval `i` is given by

$$\mathrm{p}_{\mathtt{i}} = 2\big((d_{\mathtt{i}} - 1) \bmod 7\big) - \left\lfloor \frac{\big((d_{\mathtt{i}} - 1) \bmod 7\big) + 1}{4} \right\rfloor + a_{\mathtt{i}} \tag{5.24}$$

5.2 Formalising the chord matching problem

We define a chord matching function $\mathbb{M}_T$ for two chords `X` and `Y` such that

$$\mathbb{M}_T(\mathtt{X}, \mathtt{Y}) = \begin{cases} 1 & \text{if } \mathtt{X} \text{ matches } \mathtt{Y} \\ 0 & \text{otherwise} \end{cases} \tag{5.25}$$

where T denotes the type of matching method applied to `X` and `Y`. We will now look at several alternative methods for calculating the value of matching function $\mathbb{M}_T$.

Unordered matching function

Let $\tilde{\mathbf{X}}$ denote an unordered set of objects where membership is restricted so that duplicate objects are not allowed:

$$\mathrm{X}_n \neq \mathrm{X}_m \quad \forall\, \{\mathrm{X}_n, \mathrm{X}_m \in \tilde{\mathbf{X}} \colon n \neq m\} \tag{5.26}$$

We now define an unordered matching function $\mathbb{M}_{\mathrm{U}}$ that compares unordered sets $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$:

$$\mathbb{M}_{\mathrm{U}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = \begin{cases} 1 & \text{if } |\, \tilde{\mathbf{X}} \,| = |\, \tilde{\mathbf{Y}} \,| = |\, \tilde{\mathbf{X}} \cap \tilde{\mathbf{Y}} \,| \\ 0 & \text{otherwise} \end{cases} \tag{5.27}$$

Ordered matching function

Now let $\mathbf{X}$ denote a finite size ordered multiset[7] on $\tilde{\mathbf{X}}$ [Sta97] of the form

$$\mathbf{X} = \{\mathrm{X}_0, \mathrm{X}_1, ...\}. \tag{5.28}$$

where $\nu(\mathrm{X}) \geq 1$ i.e. duplicate objects are allowed so the multiplicity ν for any object $\mathrm{X} \in \mathbf{X}$ may be one or more. Duplicate objects in $\mathbf{X}$ are distinguishable from each other by their order in the set.

Now let $\mathbb{M}_{\mathrm{O}}$ be a matching function that compares two such ordered sets, $\mathbf{X}$ and $\mathbf{Y}$, element by element so

$$\mathbb{M}_{\mathrm{O}}(\mathbf{X}, \mathbf{Y}) = \begin{cases} 1 & \text{if } |\mathbf{X}| = |\mathbf{Y}| \text{ and } \mathbf{X}_n = \mathbf{Y}_n \ \forall \ \{n \in \mathbb{Z} : 0 \leq n < |\mathbf{X}| - 1\} \\ 0 & \text{otherwise} \end{cases} \tag{5.29}$$

Using ordered and unordered matching functions $\mathbb{M}_{\mathrm{O}}$ and $\mathbb{M}_{\mathrm{U}}$, we may now define several different chord matching methods that can be used for chord comparison in the evaluation of recognition algorithms and analyses of chord annotation collections.

5.2.1 String matching

The chord symbols `X` and `Y` are represented by character strings which comply with the syntax defined in chapter 4. Thus, the simplest way to compare these two chord symbols is to use a simple character by character comparison. If both strings are the same length and if each character $\mathtt{X}_n$ is the same as the corresponding character $\mathtt{Y}_n$ then the strings are the same and we may say that the two chords match. Since `X` and `Y` may be viewed as two ordered sets of character elements, we may thus apply function $\mathbb{M}_{\mathrm{O}}$ (equation 5.29) directly to the chord symbol strings to define a string-based matching function $\mathbb{M}_{\mathtt{s}}(\mathtt{X}, \mathtt{Y})$

$$\mathbb{M}_{\mathtt{s}}(\mathtt{X}, \mathtt{Y}) = \mathbb{M}_{\mathrm{O}}(\mathtt{X}, \mathtt{Y}) \tag{5.30}$$

This is a simple result and would be perfectly adequate for matching the chord symbols if it were not for the fact that the chord syntax we are using

[7] As mentioned earlier in section 5.1.1 on page 111.

allows for alternative spellings of the same chord. For example, the chord C major may be represented in at least three different ways: '`C:maj`', '`C:(1,3,5)`' or simply '`C`'. However, the string-based matching function will not evaluate these different spellings as correct matches because while

$$\mathbb{M}_{\mathbf{s}}\big(\texttt{"C:maj"}, \texttt{"C:maj"}\big) = 1$$

as we would expect, from the definition of $\mathbb{M}_{\mathbf{s}}$ we see that

$$\begin{aligned}\mathbb{M}_{\mathbf{s}}\big(\texttt{"C"}, \texttt{"C:maj"}\big) &= 0\\ \mathbb{M}_{\mathbf{s}}\big(\texttt{"C:(1,3,5)"}, \texttt{"C:maj"}\big) &= 0\\ \mathbb{M}_{\mathbf{s}}\big(\texttt{"C"}, \texttt{"C:(1,3,5)"}\big) &= 0.\end{aligned}$$

This is potentially unhelpful when working with a test set that might contain a mixture of these different spellings or worse, trying to evaluate a recognition algorithm that outputs one spelling when the test set contains an alternative spelling for the same chord.

5.2.2 Pitchname set comparison

From the definition of the chord syntax in chapter 4, we know that every chord symbol is equivalent to a root pitchname plus a number of intervals relative to that root. These intervals define the constituent pitchnames of the chord and possibly the bass note if it differs from the root.

Comparing ordered pitchname sets

Let $\overrightarrow{\mathbf{v}}_{\texttt{X}}$ be an ordered set of pitchnames (hereafter referred to as an ordered *pnset*) for chord `X` such that the pitchnames are ordered starting from the bass note in ascending cyclic order. Thus we see that the pnset for chord '`C:maj`' is

$$\overrightarrow{\mathbf{v}}_{\texttt{"C:maj"}} = \overrightarrow{\mathbf{v}}_{\texttt{"C"}} = \overrightarrow{\mathbf{v}}_{\texttt{"C:(1,3,5)"}} = \big\{\texttt{"C"}, \texttt{"E"}, \texttt{"G"}\big\} \tag{5.31}$$

whereas the enharmonic equivalent D♭♭ major triad '`Dbb:maj`' will have pnset

$$\overrightarrow{\mathbf{v}}_{\texttt{"Dbb:maj"}} = \big\{\texttt{"Dbb"}, \texttt{"Fb"}, \texttt{"Abb"}\big\}. \tag{5.32}$$

We may now define a new version of the chord matching function, $\mathbb{M}_{\overrightarrow{\mathbf{v}}}$, that compares the pnsets for chords `X` and `Y`:

$$\mathbb{M}_{\overrightarrow{\mathbf{v}}}(\mathtt{X},\mathtt{Y}) = \mathbb{M}_{\mathrm{O}}(\overrightarrow{\mathbf{v}}_{\mathtt{X}}, \overrightarrow{\mathbf{v}}_{\mathtt{Y}}) \tag{5.33}$$

Now we have a function that will correctly match alternative spellings of a particular chord. Returning to our earlier example, triad '`C:maj`', we observe that

$$\begin{aligned}\mathbb{M}_{\overrightarrow{\mathbf{v}}}\big(\texttt{"C"},\texttt{"C:maj"}\big) &= \mathbb{M}_{\overrightarrow{\mathbf{v}}}\big(\texttt{"C:(1,3,5)"},\texttt{"C:maj"}\big)\\ &= \mathbb{M}_{\overrightarrow{\mathbf{v}}}\big(\texttt{"C"},\texttt{"C:(1,3,5)"}\big)\\ &= \mathbb{M}_{\overrightarrow{\mathbf{v}}}\big(\texttt{"C:maj"},\texttt{"C:maj"}\big) = 1.\end{aligned} \tag{5.34}$$

However, the enharmonic equivalent '`Dbb:maj`' will not evaluate as a correct match:

$$\mathbb{M}_{\overrightarrow{\mathbf{v}}}\big(\texttt{"C:maj"},\texttt{"Dbb:maj"}\big) = 0. \tag{5.35}$$

Specifying an alternative bass interval in the chord symbol will affect the ordering of the pnset. For example, the pnset for a first inversion C major triad '`C:maj/3`' will be

$$\overrightarrow{\mathbf{v}}_{\texttt{"C:maj/3"}} = \{\texttt{"E"},\texttt{"G"},\texttt{"C"}\}. \tag{5.36}$$

If the bass interval is not a member of the chord's interval list, for example '`D:min/b7`', the new bass pitchname is added to the pnset as the first element

$$\overrightarrow{\mathbf{v}}_{\texttt{"D:min/b7"}} = \{\texttt{"C"},\texttt{"D"},\texttt{"F"},\texttt{"A"}\}. \tag{5.37}$$

It should be noted that pitchnames do not contain octave information so an extended interval such as a 9th will map to the same pitchname as its lower octave equivalent (which in the case of a 9th is a 2nd). However, by using ordered pnsets it is still possible to distinguish between chords such as '`C:maj9`' and '`C:maj7(2)`' because the order of pitchnames will be different:

$$\begin{aligned}\overrightarrow{\mathbf{v}}_{\texttt{"C:maj9"}} &= \{\texttt{"C"},\texttt{"E"},\texttt{"G"},\texttt{"B"},\texttt{"D"}\}\\ \overrightarrow{\mathbf{v}}_{\texttt{"C:maj7(2)"}} &= \{\texttt{"C"},\texttt{"D"},\texttt{"E"},\texttt{"G"},\texttt{"B"}\}\\ \therefore\ \overrightarrow{\mathbf{v}}_{\texttt{"C:maj9"}} &\neq \overrightarrow{\mathbf{v}}_{\texttt{"C:maj7(2)"}}\end{aligned} \tag{5.38}$$

Comparing unordered pnsets

In some instances we may wish to know if chords `X` and `Y` contain the same pitchnames regardless of the order of chord intervals.

Let $\tilde{\mathbf{v}}_{\mathtt{X}}$ be the unordered pnset for chord `X`. Since it is not ordered, we specify that set $\tilde{\mathbf{v}}_{\mathtt{X}}$ may contain only one instance of any particular pitchname so whereas the chord symbol '`C:maj9(2)`' would have the ordered pnset

$$\overrightarrow{\mathbf{v}}_{\texttt{"C:maj9(2)"}} = \{\texttt{"C"}, \texttt{"D"}, \texttt{"E"}, \texttt{"G"}, \texttt{"B"}, \texttt{"D"}\} \tag{5.39}$$

the unordered pnset[8] would be

$$\tilde{\mathbf{v}}_{\texttt{"C:maj9(2)"}} = \{\texttt{"C"}, \texttt{"D"}, \texttt{"E"}, \texttt{"G"}, \texttt{"B"}\}. \tag{5.40}$$

Thus, applying the matching function $\mathbb{M}_{\mathrm{U}}$ (equation 5.27) to the unordered pnsets for chords `X` and `Y` produces

$$\mathbb{M}_{\tilde{\mathbf{v}}}(\mathtt{X}, \mathtt{Y}) = \mathbb{M}_{\mathrm{U}}(\tilde{\mathbf{v}}_{\mathtt{X}}, \tilde{\mathbf{v}}_{\mathtt{Y}}) \tag{5.41}$$

This gives us a function that will match any pair of chords that contain the same set of pitchnames regardless of root or interval order. Therefore chords '`C:maj9`' and '`C:maj7(2)`' will be evaluated as a correct match by this function:

$$\mathbb{M}_{\tilde{\mathbf{v}}}\big(\texttt{"C:maj9"}, \texttt{"C:maj7(2)"}\big) = 1 \tag{5.42}$$

5.2.3 Pitchclass set comparison

In some cases, particularly evaluation of audio chord recognition algorithms, we may wish to consider chords which are enharmonic equivalents as being a correct match. For this we compare the constituent pitchclasses[9] of two chords rather than comparing the constituent pitchnames as we did in section 5.2.2.

Let $\overrightarrow{\mathbf{p}}_{\mathtt{X}}$ be an ordered set of absolute pitchclasses (hereafter referred to as an ordered absolute *pcset*) for chord `X` such that the pitchclasses are

[8] In this case the pitchnames in the unordered pnset have been notated in ascending interval order from the root but it would be equally valid to write them in any other order.

[9] Where a pitchclass $\mathrm{p}_{\mathtt{v}}$ is an integer between 0 and 11 that relates to pitchname `v` as described in equation 5.23, section 5.1.1 on page 116 and we reference absolute pitchclass 0 to pitchname C.

arranged in the same order as their equivalent pitchnames in pnset $\overrightarrow{\mathbf{v}}_{\mathtt{X}}$. Hence, the pcset for chord '`C:maj`' is

$$\overrightarrow{\mathbf{p}}_{\texttt{"C:maj"}} = \overrightarrow{\mathbf{p}}_{\texttt{"C"}} = \overrightarrow{\mathbf{p}}_{\texttt{"C:(1,3,5)"}} = \{0, 4, 7\} \tag{5.43}$$

and the enharmonic equivalent D𝄫 major triad '`Dbb:maj`' will also have the same pcset

$$\overrightarrow{\mathbf{p}}_{\texttt{"Dbb:maj"}} = \overrightarrow{\mathbf{p}}_{\texttt{"C:maj"}} = \{0, 4, 7\}. \tag{5.44}$$

As with the matching functions for pnsets in equations 5.33 and 5.41, we may now apply functions $\mathbb{M}_{\mathrm{O}}$ and $\mathbb{M}_{\mathrm{U}}$ to pcsets to perform ordered and unordered matches respectively so we define an ordered pcset matching function $\mathbb{M}_{\overrightarrow{\mathbf{p}}}$ as

$$\mathbb{M}_{\overrightarrow{\mathbf{p}}}(\mathtt{X}, \mathtt{Y}) = \mathbb{M}_{\mathrm{O}}(\overrightarrow{\mathbf{p}}_{\mathtt{X}}, \overrightarrow{\mathbf{p}}_{\mathtt{Y}}) \tag{5.45}$$

and unordered pcset matching function $\mathbb{M}_{\widetilde{\mathbf{p}}}$ as

$$\mathbb{M}_{\widetilde{\mathbf{p}}}(\mathtt{X}, \mathtt{Y}) = \mathbb{M}_{\mathrm{U}}(\widetilde{\mathbf{p}}_{\mathtt{X}}, \widetilde{\mathbf{p}}_{\mathtt{Y}}). \tag{5.46}$$

Using ordered matching function $\mathbb{M}_{\overrightarrow{\mathbf{p}}}$, chords of the same type that have enharmonic equivalent roots will be considered correct matches

$$\mathbb{M}_{\overrightarrow{\mathbf{p}}}\big(\texttt{"C:maj"}, \texttt{"Dbb:maj"}\big) = 1. \tag{5.47}$$

Chords with the same root but enharmonic equivalent interval lists will also be considered correct matches. For example, a diminished seventh chord (shorthand notation '`dim7`') which corresponds to the interval list `"(1,b3,b5,bb7)"` will match a chord with the different, but enharmonic equivalent, interval list `"C:(1,#2,#4,6)"`:

$$\mathbb{M}_{\overrightarrow{\mathbf{p}}}\big(\texttt{"C:dim7"}, \texttt{"C:(1,\#2,\#4,6)"}\big) = 1. \tag{5.48}$$

Likewise, if using the unordered matching function $\mathbb{M}_{\widetilde{\mathbf{p}}}$, two chords made up of the same set of absolute pitchclasses will be considered a correct match regardless of pitchclass order. For example, chords '`Db:maj6`' and '`A#:min7`' have different note sets:

$$\begin{aligned} \overrightarrow{\mathbf{v}}_{\texttt{"Db:maj6"}} &= \big\{\texttt{"Db"}, \texttt{"F"}, \texttt{"Ab"}, \texttt{"Bb"}\big\} \\ \overrightarrow{\mathbf{v}}_{\texttt{"A\#:min7"}} &= \big\{\texttt{"A\#"}, \texttt{"C\#"}, \texttt{"E\#"}, \texttt{"G\#"}\big\} \end{aligned} \tag{5.49}$$

However, their unordered pcsets comprise the same four absolute pitch-classes:

$$\tilde{\mathbf{p}}_{\texttt{"Db:maj6"}} = \tilde{\mathbf{p}}_{\texttt{"A\#:min7"}} = \{1, 5, 8, 10\} \tag{5.50}$$

thus

$$\mathbb{M}_{\tilde{\mathbf{p}}}\big(\texttt{"Db:maj6"}, \texttt{"A\#:min7"}\big) = 1. \tag{5.51}$$

5.3 Chordtype comparison

In some cases we may wish to compare two chord symbols in terms of their chordtype only, disregarding the root pitchname. This can be useful for evaluating how accurately a chord recognition algorithm detects the correct chordtype or family. It is also useful for calculating chordtype distribution statistics such as those in chapter 6 where we analyse the Beatles chord transcription collection.

5.3.1 String matching chordtypes

As discussed in section 5.2, chord symbols $\texttt{X}$ and $\texttt{Y}$ are character strings of the form shown in equation 5.1. We may discard the chord root pitchnames $\texttt{R}_\texttt{X}$ and $\texttt{R}_\texttt{Y}$ and just compare the strings $\texttt{Q}_\texttt{X} \oplus \texttt{T}_\texttt{X}$ and $\texttt{Q}_\texttt{Y} \oplus \texttt{T}_\texttt{Y}$ which describe the chordtype and inversion of the two chords. In this way we may define a string-based chordtype matching function $\mathbb{M}_\mathbf{q}$ which employs the earlier string based chord matching function $\mathbb{M}_\mathbf{s}$:

$$\mathbb{M}_\mathbf{q}(\texttt{X}, \texttt{Y}) = \mathbb{M}_\mathbf{s}(\texttt{Q}_\texttt{X} \oplus \texttt{T}_\texttt{X}, \texttt{Q}_\texttt{Y} \oplus \texttt{T}_\texttt{Y}) \tag{5.52}$$

Function $\mathbb{M}_\mathbf{q}$ is a simple method of comparing chordtypes but, as with function $\mathbb{M}_\mathbf{s}$, it will evaluate different spellings of the same chordtype as being incorrect matches so

$$\mathbb{M}_\mathbf{q}\big(\texttt{"C:maj"}, \texttt{"F:maj"}\big) = 1 \tag{5.53}$$

but

$$\mathbb{M}_\mathbf{q}\big(\texttt{"C:maj"}, \texttt{"F"}\big) = 0. \tag{5.54}$$

While this my be useful for finding the number of unique chordtype strings in an annotated collection (see chapter 6), is it not helpful when trying

to evaluate chord recognition algorithms which may output chords spelled differently to those in the ground truth annotation.

5.3.2 Interval set matching

An alternative to string-based chordtype matching is to compare the interval sets of the two chords. Let $\overrightarrow{\mathbf{i}}_{\mathtt{X}}$ be the ordered set of intervals (hereafter referred to as an ordered *rlset*) for chord `X` that describes the structure of the chord relative to its root. For example any spelling of a major triad with a given root `R` the rlset is

$$\overrightarrow{\mathbf{i}}_{\mathtt{R}} = \overrightarrow{\mathbf{i}}_{\mathtt{R}\oplus\texttt{":maj"}} = \overrightarrow{\mathbf{i}}_{\mathtt{R}\oplus\texttt{":(1,3,5)"}} = \{\texttt{"1"}, \texttt{"3"}, \texttt{"5"}\} \tag{5.55}$$

We may therefore define a chordtype matching function $\mathbb{M}_{\overrightarrow{\mathbf{i}}}$ that compares the rlsets of two chords

$$\mathbb{M}_{\overrightarrow{\mathbf{i}}}(\mathtt{X}, \mathtt{Y}) = \mathbb{M}_{\mathrm{O}}(\overrightarrow{\mathbf{i}}_{\mathtt{X}}, \overrightarrow{\mathbf{i}}_{\mathtt{Y}}). \tag{5.56}$$

Now let $\widetilde{\mathbf{i}}_{\mathtt{X}}$ be the unordered rlset for chord `X` so we may also define a matching function $\mathbb{M}_{\widetilde{\mathbf{i}}}$ for unordered rlsets

$$\mathbb{M}_{\widetilde{\mathbf{i}}}(\mathtt{X}, \mathtt{Y}) = \mathbb{M}_{\mathrm{U}}(\widetilde{\mathbf{i}}_{\mathtt{X}}, \widetilde{\mathbf{i}}_{\mathtt{Y}}). \tag{5.57}$$

We also note that we can calculate $\mathbb{M}_{\overrightarrow{\mathbf{i}}}$ and $\mathbb{M}_{\widetilde{\mathbf{i}}}$ using $\mathbb{M}_{\overrightarrow{\mathbf{v}}}$ and $\mathbb{M}_{\widetilde{\mathbf{v}}}$ respectively[10] because

$$\mathbb{M}_{\overrightarrow{\mathbf{i}}}(\mathtt{X}, \mathtt{Y}) = \mathbb{M}_{\overrightarrow{\mathbf{v}}}\big(\{\mathtt{R} \oplus \mathtt{Q}_{\mathtt{X}} \oplus \mathtt{T}_{\mathtt{X}}\}, \{\mathtt{R} \oplus \mathtt{Q}_{\mathtt{Y}} \oplus \mathtt{T}_{\mathtt{Y}}\}\big) \tag{5.58}$$

and

$$\mathbb{M}_{\widetilde{\mathbf{i}}}(\mathtt{X}, \mathtt{Y}) = \mathbb{M}_{\widetilde{\mathbf{v}}}\big(\{\mathtt{R} \oplus \mathtt{Q}_{\mathtt{X}} \oplus \mathtt{T}_{\mathtt{X}}\}, \{\mathtt{R} \oplus \mathtt{Q}_{\mathtt{Y}} \oplus \mathtt{T}_{\mathtt{Y}}\}\big). \tag{5.59}$$

where `R` is a dummy pitchname value that is used to replace the real root pitchnames of both chords. This means that intervals in both chords are translated to pitchnames relative to the dummy root `R` for the comparison.

[10] This is a potentially useful result if you want to code an implementation of these functions on a computer in a short space of time.

5.3.3 Relative pitchclass set matching

Comparing rlsets will match chordtypes of the same enharmonic spelling but will not match enharmonic equivalents. We can perform an enharmonic equivalent chordtype comparison by using relative pitchclass sets (hereafter referred to as *rcsets*) which are sets of pitchclasses relative to the chord root instead of absolute pitchclasses which we number relative to pitchname C. Let $\overrightarrow{\mathbf{r}}_{\mathtt{X}}$ be the ordered rcset for chord `X` thus the rcset for any major triad will be

$$\overrightarrow{\mathbf{r}}_{\mathtt{R}} = \overrightarrow{\mathbf{r}}_{\mathtt{R}\oplus\texttt{":maj"}} = \overrightarrow{\mathbf{r}}_{\mathtt{R}\oplus\texttt{":(1,3,5)"}} = \{0, 4, 7\} \tag{5.60}$$

and we may define a new chordtype matching function $\mathbb{M}_{\overrightarrow{\mathbf{r}}}$ based on rcset comparison

$$\mathbb{M}_{\overrightarrow{\mathbf{r}}}(\mathtt{X}, \mathtt{Y}) = \mathbb{M}_{\mathrm{O}}(\overrightarrow{\mathbf{r}}_{\mathtt{X}}, \overrightarrow{\mathbf{r}}_{\mathtt{Y}}) \tag{5.61}$$

Using $\mathbb{M}_{\overrightarrow{\mathbf{r}}}$ to compare chordtypes, enharmonic equivalent spellings will be considered correct matches so a diminished seventh chord with interval list `"(1,b3,b5,bb7)"` will match any chord with the alternative, enharmonic equivalent interval list `"(1,#2,#4,6)"`.

We may also define an unordered rcset matching function

$$\mathbb{M}_{\widetilde{\mathbf{r}}}(\mathtt{X}, \mathtt{Y}) = \mathbb{M}_{\mathrm{U}}(\widetilde{\mathbf{r}}_{\mathtt{X}}, \widetilde{\mathbf{r}}_{\mathtt{Y}}) \tag{5.62}$$

and we note that because the relationship between rcsets and rlsets is analogous to that between pcsets and pnsets, $\mathbb{M}_{\overrightarrow{\mathbf{r}}}$ and $\mathbb{M}_{\widetilde{\mathbf{r}}}$ are equivalent to

$$\mathbb{M}_{\overrightarrow{\mathbf{r}}}(\mathtt{X}, \mathtt{Y}) = \mathbb{M}_{\overrightarrow{\mathbf{p}}}\big(\{\mathtt{R} \oplus \mathtt{Q}_{\mathtt{X}} \oplus \mathtt{T}_{\mathtt{X}}\}, \{\mathtt{R} \oplus \mathtt{Q}_{\mathtt{Y}} \oplus \mathtt{T}_{\mathtt{Y}}\}\big) \tag{5.63}$$

and

$$\mathbb{M}_{\widetilde{\mathbf{r}}}(\mathtt{X}, \mathtt{Y}) = \mathbb{M}_{\widetilde{\mathbf{p}}}\big(\{\mathtt{R} \oplus \mathtt{Q}_{\mathtt{X}} \oplus \mathtt{T}_{\mathtt{X}}\}, \{\mathtt{R} \oplus \mathtt{Q}_{\mathtt{Y}} \oplus \mathtt{T}_{\mathtt{Y}}\}\big) \tag{5.64}$$

where `R` again is a dummy root that may take any pitchname value.

5.4 Cardinality-limited comparison

Matching functions $\mathbb{M}_{\mathrm{O}}$ and $\mathbb{M}_{\mathrm{U}}$ both require the two sets being compared to be of the same cardinality in order to match. When evaluating chord

recognition algorithms, it may be desirable to allow chord symbols with different cardinalities to be considered correct matches. Consider, for example, a chord recognition algorithm that has been designed to recognise triad chords. Such an algorithm will never be able to produce a tetrad chord as an estimate yet the annotated collection used as a ground truth may well contain tetrad chord symbols. If the annotated chord at a certain moment is '`D:maj7`', the recognition algorithm might output '`D:maj`'. We would generally consider this to be a correct match within the limits of the algorithm's design because the two chords share the same root and first three intervals.

With the comparison methods discussed thus far, a '`D:maj7`' and '`D:maj`' cannot be considered a correct match. We will therefore introduce a new parameter M to our general matching function from equation 5.25 to giving us $\mathbb{M}_{T,M}$ where M defines the number of elements that are required to be the same in order to consider the comparison a correct match.

Ordered cardinality-M matching

Let $\{\mathbf{X}\}_M$ denote a cardinality-M set that contains the first M elements of set $\mathbf{X}$. Thus if $\mathbf{X}$ contains three elements $\{\mathrm{X}_0, \mathrm{X}_1, \mathrm{X}_2\}$, we may write

$$\begin{aligned} \{\mathbf{X}\}_1 &= \{\mathrm{X}_0\} \\ \{\mathbf{X}\}_2 &= \{\mathrm{X}_0, \mathrm{X}_1\} \\ \{\mathbf{X}\}_3 &= \{\mathrm{X}_0, \mathrm{X}_1, \mathrm{X}_2\} \end{aligned} \tag{5.65}$$

and if M exceeds $|\mathbf{X}|$, we insert $M - |\mathbf{X}|$ null elements (denoted by '-') at the end of $\mathbf{X}$.

$$\begin{aligned} \{\mathbf{X}\}_4 &= \{\mathrm{X}_0, \mathrm{X}_1, \mathrm{X}_2, \text{-}\} \\ \{\mathbf{X}\}_5 &= \{\mathrm{X}_0, \mathrm{X}_1, \mathrm{X}_2, \text{-}, \text{-}\}. \end{aligned} \tag{5.66}$$

Now we may define a cardinality-M ordered matching function $\mathbb{M}_{\mathrm{O},M}$ such that

$$\mathbb{M}_{\mathrm{O},M}(\mathbf{X}, \mathbf{Y}) = \begin{cases} 1 & \text{if } \{\mathbf{X}\}_{M,n} = \{\mathbf{Y}\}_{M,n} \ \forall \ \{n \in \mathbb{Z} : 0 \leq n < M\} \\ 0 & \text{otherwise} \end{cases} \tag{5.67}$$

so sets $\mathbf{X}$ and $\mathbf{Y}$ are considered a correct match if the first M elements of each are the same.

We may now use this cardinality-M ordered matching function on ordered pnsets, pcsets, rlsets and rcsets to compare chords of different cardinalities. In our earlier example, we wanted to allow chords '`D:maj7`' and '`D:maj`' to be considered a correct match given that the '`D:maj`' was produced by an algorithm that could only produce triads. Using a cardinality-3 ordered pnset matching function $\mathbb{M}_{\overrightarrow{\mathbf{v}},3}$ this is possible because the pnset for '`D:maj`' is a subset of that for '`D:maj7`'

$$\overrightarrow{\mathbf{v}}_{\texttt{"D:maj"}} \subset \overrightarrow{\mathbf{v}}_{\texttt{"D:maj7"}} \tag{5.68}$$

with the first three elements in both being the same

$$\{\overrightarrow{\mathbf{v}}_{\texttt{"D:maj7"}}\}_3 = \{\overrightarrow{\mathbf{v}}_{\texttt{"D:maj"}}\}_3 = \{\texttt{"D"}, \texttt{"F\#"}, \texttt{"A"}\} \tag{5.69}$$

therefore

$$\mathbb{M}_{\overrightarrow{\mathbf{v}},3}\big(\texttt{"D:maj7"}, \texttt{"D:maj"}\big) = 1. \tag{5.70}$$

In fact,

$$\mathbb{M}_{\overrightarrow{\mathbf{v}},M}\big(\texttt{"D:maj7"}, \texttt{"D:maj"}\big) = 1 \quad \text{for } 0 \leq M \leq 3 \tag{5.71}$$

because $|\overrightarrow{\mathbf{v}}_{\texttt{"D:maj"}}| = 3$. However, for cardinality-4

$$\begin{aligned} \{\overrightarrow{\mathbf{v}}_{\texttt{"D:maj"}}\}_4 &= \{\texttt{"D"}, \texttt{"F\#"}, \texttt{"A"}, \texttt{-}\} \\ \{\overrightarrow{\mathbf{v}}_{\texttt{"D:maj7"}}\}_4 &= \{\texttt{"D"}, \texttt{"F\#"}, \texttt{"A"}, \texttt{"C\#"}\} \\ \therefore \{\overrightarrow{\mathbf{v}}_{\texttt{"D:maj"}}\}_4 &\neq \{\overrightarrow{\mathbf{v}}_{\texttt{"D:maj7"}}\}_4 \end{aligned} \tag{5.72}$$

hence

$$\mathbb{M}_{\overrightarrow{\mathbf{v}},4}\big(\texttt{"D:maj7"}, \texttt{"D:maj"}\big) = 0 \tag{5.73}$$

Using this matching function, chord '`D:maj`' will not match anything other than itself[11] for any value of M greater than $|\overrightarrow{\mathbf{v}}_{\texttt{"D:maj"}}|$.

[11] with the exception of alternative spellings such as '`D`' and '`D:(1,3,5)`'.

Unordered cardinality-M matching

We may also define a cardinality-M unordered matching function $\mathbb{M}_{\mathrm{U},M}$ such that

$$\mathbb{M}_{\mathrm{U},M}(\tilde{\mathbf{X}},\tilde{\mathbf{Y}}) = \begin{cases} 1 & \text{if } |\, \tilde{\mathbf{X}} \cap \tilde{\mathbf{Y}} \,| \geq M \\ \mathbb{M}_{\mathrm{U}}(\tilde{\mathbf{X}},\tilde{\mathbf{Y}}) & \text{otherwise.} \end{cases} \tag{5.74}$$

That is, the matching function will evaluate to 1 if at least M elements in $\tilde{\mathbf{X}}$ have matching elements in $\tilde{\mathbf{Y}}$. Otherwise, in the event that $|\, \tilde{\mathbf{X}} \,|$ and $|\, \tilde{\mathbf{Y}} \,|$ are both less than M then we can use the original (non-cardinality-limited) unordered matching function $\mathbb{M}_{\mathrm{U}}(\tilde{\mathbf{X}},\tilde{\mathbf{Y}})$ (see equation 5.27) to evaluate the match.

In this way, the chords '`D#:maj7`' and '`Eb:sus4`', which have two pitch-classes in common, could be considered a match by using a cardinality-2 unordered pcset comparison because

$$\begin{aligned} \tilde{\mathbf{p}}_{\texttt{"D\#:maj7"}} &= \{3,7,10,2\} \\ \tilde{\mathbf{p}}_{\texttt{"Eb:sus4"}} &= \{3,5,10\} \end{aligned} \tag{5.75}$$

so

$$\Big|\, \tilde{\mathbf{p}}_{\texttt{"D\#:maj7"}} \cap \tilde{\mathbf{p}}_{\texttt{"Eb:sus4"}} \,\Big| = \Big|\{3,10\}\Big| = 2 \tag{5.76}$$

therefore

$$\mathbb{M}_{\tilde{\mathbf{p}},2}\big(\texttt{"D\#:maj"},\texttt{"Eb:sus4"}\big) = 1 \tag{5.77}$$

5.5 Treatment of bass intervals

The matching functions proposed so far have assumed that the first element of ordered sets is the bass note of the chord. For most chord symbols this will be the first interval in the chord's equivalent interval list[12], which is usually '`1`' signifying the root pitchname. However, in some cases an alternative bass interval may be specified at the end of the chord symbol string. If this interval is already a member of the equivalent interval list for the chord then it signifies an inversion, causing the elements of the list

[12]Equivalent, that is, to the list of intervals defined by a shorthand string or other form of the syntax that may specify an interval list indirectly, see sections 4.2.4 to 4.3.

to change order. If the bass interval is not a member of the interval list then it adds an extra element to the chord.

In some cases, we may wish to ignore the bass interval of chords when we make comparisons with others. For example, if a chord recognition algorithm cannot recognise inversions then it could be unfair to use an ordered matching function to evaluate it when the annotated ground truth contains inverted chords. If such an algorithm outputs a '`C:maj`' symbol where the annotation contains '`C:maj/5`' then we may reasonably wish to consider this a correct match.

Because of this, we will define a 'bass-blind' version of general matching function $\mathbb{M}_{T,M}$ denoted with a prime $\mathbb{M}'_{T,M}$ such that

$$\mathbb{M}'_{T,M}(\mathtt{X},\mathtt{Y}) = \mathbb{M}_{T,M}\big(\{\mathtt{R_X} \oplus \mathtt{Q_X}\},\{\mathtt{R_Y} \oplus \mathtt{Q_Y}\}\big) \tag{5.78}$$

That is, if present, the bass interval of each chord is discarded from the chord symbol string before applying the desired matching function to the root and chordtype.

5.6 Chord likeness measure

The chord comparison methods discussed thus far in this chapter have given a binary output; either evaluating chords as a correct match or not a match at all. In some cases, it might be useful to know how alike a pair of chords are with some kind of measure that gives a range of values for chord 'likeness' rather than just 1 or 0.

A simple way to measure chord-likeness is to look at the number of shared tones between two chords. For example, figure 5.3 shows chords '`A:min`', '`A:dim`', '`C:maj`', '`E:min`', '`C:maj7`', '`C:min`', '`C:min7`' and '`G:maj`' on the pitchname tonnetz and the numbers of shared tones between each pair of these chords is shown in table 5.1. The highest number of shared tones is obviously given when comparing a chord to itself when it will equal the cardinality of the chord in question. However, when comparing chords of different cardinalities, for example '`C:maj7`' and '`E:min`', it is possible for the number of shared tones to equal the size of the smaller chord if it is a subset of the larger one.

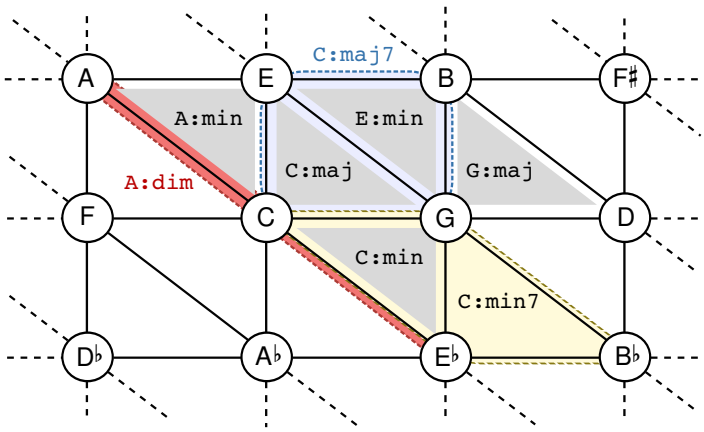


Figure 5.3: Chords '`A:min`', '`A:dim`', '`C:maj`', '`E:min`', '`C:maj7`' ,'`C:min`', '`C:min7`', and '`G:maj`' on the pitchname tonnetz.

Table 5.1: Number of shared tones for chords '`A:min`', '`A:dim`', '`C:maj`', '`E:min`', '`C:maj7`' ,'`C:min`', '`C:min7`', and '`G:maj`'.

Chord	`A:min`	`A:dim`	`C:maj`	`E:min`	`C:maj7`	`C:min`	`C:min7`	`G:maj`
`A:min`	3	2	2	1	2	1	1	0
`A:dim`	2	3	1	0	1	2	2	0
`C:maj`	2	1	3	2	3	2	2	1
`E:min`	1	0	2	3	3	1	1	2
`C:maj7`	2	1	3	3	4	2	2	2
`C:min`	1	2	2	1	2	3	3	1
`C:min7`	1	2	2	0	2	3	4	1
`G:maj`	0	0	1	2	2	1	1	3

Let us propose a likeness function $\mathbb{L}_{\mathrm{U}}$ for two unordered sets $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ where the result is the number of shared elements between the two sets divided by the total number of unique elements in the two sets

$$\mathbb{L}_{\mathrm{U}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = \begin{cases} 1 & \text{if } \tilde{\mathbf{X}}{=}\tilde{\mathbf{Y}}{=}\emptyset \\ \frac{|\tilde{\mathbf{X}} \cap \tilde{\mathbf{Y}}|}{|\tilde{\mathbf{X}} \cup \tilde{\mathbf{Y}}|} & \text{otherwise.} \end{cases} \tag{5.79}$$

An exception is made for the case where both sets are empty, in which case they are obviously the same but by definition cannot have any shared tones.

We may now use this function to calculate chord likeness values for pnsets, pcsets, rlsets and rcsets. For example, we may define a pnset

Table 5.2: Chord pnset likeness values for chords '`A:min`', '`A:dim`', '`C:maj`', '`E:min`', '`C:maj7`' ,'`C:min`', '`C:min7`', and '`G:maj`'.

Chord	`A:min`	`A:dim`	`C:maj`	`E:min`	`C:maj7`	`C:min`	`C:min7`	`G:maj`
`A:min`	1	1/2	1/2	1/5	2/5	1/5	1/6	0
`A:dim`	1/2	1	1/5	0	1/6	2/5	1/3	0
`C:maj`	1/2	1/5	1	1/2	3/4	1/2	2/5	1/5
`E:min`	1/5	0	1/2	1	3/4	1/5	1/6	1/2
`C:maj7`	2/5	1/5	3/4	3/4	1	2/5	1/3	2/5
`C:min`	1/5	2/5	1/2	1/5	2/5	1	3/4	1/5
`C:min7`	1/6	1/3	2/5	0	1/3	3/4	1	1/6
`G:maj`	0	0	1/5	1/2	2/5	1/5	1/6	1

likeness function

$$\mathbb{L}_{\widetilde{\mathbf{v}}}(\mathtt{X},\mathtt{Y}) = \mathbb{L}_{\mathrm{U}}(\widetilde{\mathbf{v}}_{\mathtt{X}},\widetilde{\mathbf{v}}_{\mathtt{Y}}) \tag{5.80}$$

Table 5.2 shows the values for the pnset likeness function $\mathbb{L}_{\widetilde{\mathbf{v}}}$ for the eight chords[13] in figure 5.3. The function gives a range of results between 0 and 1 as we require and the results also give us more information than simply looking at the number of shared tones alone. By dividing the number of shared tones by the total number of unique tones present in the two sets, we have some idea of how different the two chords are as well as how alike. For example, consider chords '`C:maj`', '`C:min`' and '`C:1,5`'. All three possible pairings of these three chords have the two shared tones C and G but it is arguable that dyad '`C:1,5`' is closer to '`C:maj`' and '`C:min`' than they are to each other because it does not contain a major or minor third and is thus a subset of both triads. The pnset likeness function reflects this

$$\begin{aligned}\mathbb{L}_{\widetilde{\mathbf{v}}}\big(\texttt{"C:maj"},\texttt{"C:(1,5)"}\big) &= \tfrac{2}{3}\\ \mathbb{L}_{\widetilde{\mathbf{v}}}\big(\texttt{"C:min"},\texttt{"C:(1,5)"}\big) &= \tfrac{2}{3}\\ \mathbb{L}_{\widetilde{\mathbf{v}}}\big(\texttt{"C:maj"},\texttt{"C:min"}\big) &= \tfrac{1}{2}\end{aligned} \tag{5.81}$$

[13]The values for pnset likeness function $\mathbb{L}_{\widetilde{\mathbf{v}}}$ are the same for a pcset likeness function $\mathbb{L}_{\widetilde{\mathbf{p}}}$ for the eight chords in the example because they are all closely spaced on the tonnetz. Replacing '`C:maj`' with its enharmonic equivalent '`B#:maj`' would alter the pnset likeness values but would not alter the pcset likeness values.

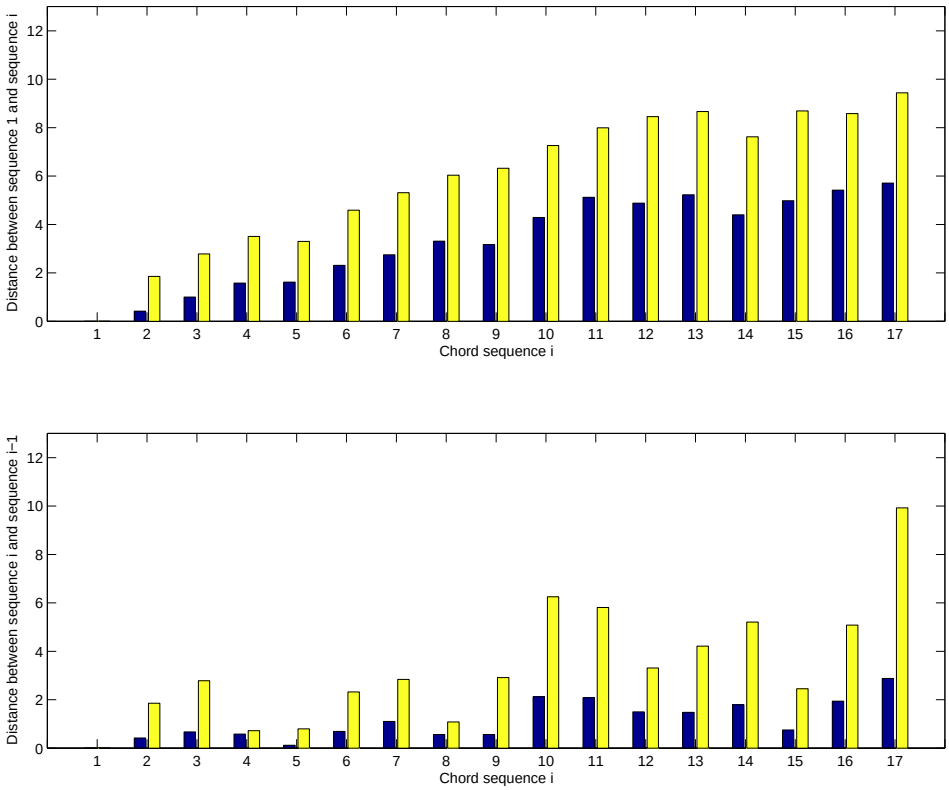


Figure 5.4: Bar charts showing results for d_{TPSD} (dark blue) and $d_{\mathbb{L}} \times 13$ (light yellow) for the seventeen chord sequences shown in table 5.4. Upper chart shows results for sequence i compared to sequence 1. Lower chart shows results for sequence i compared to sequence $i-1$ (apart from $i = 1$ where sequence 1 is compared with itself).

5.6.1 Comparison of chord likeness function with TPSD

In their paper from ISMIR08 [DHVW08], de Haas et al. propose a chord sequence distance measure called the Tonal Pitch Step Distance (TPSD) based on Lehrdal's Tonal Pitch Space (TPS) chord distance [Ler01]. In the paper, to show how the TPSD behaves in practice, they present distance results for seventeen 12-bar blues variations (the chord sequences for which are shown in table 5.4). We will use the same set of chord sequences here to briefly compare the results of pcset chord-likeness function $\mathbb{L}_{\widetilde{\mathbf{p}}}$ with their results for the TPSD.

The TPS chord distance rule is a measure of 'chord un-likeness' which gets larger as the chords become less related. It depends on the number

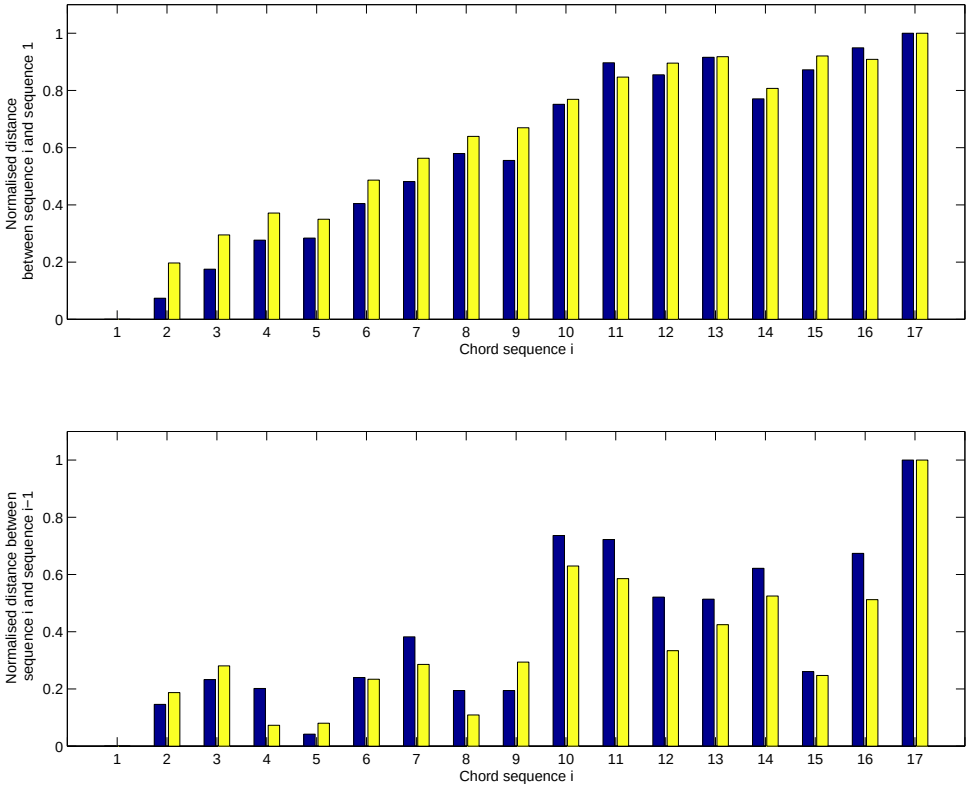


Figure 5.5: Bar charts showing normalised results for d_{TPSD} (dark blue) and $d_{\mathbb{L}}$ (light yellow) for the seventeen chord sequences shown in table 5.4. Upper chart shows results for sequence i compared to sequence 1. Lower chart shows results for sequence i compared to sequence $i-1$ (apart from $i=1$ where sequence 1 is compared with itself).

of uncommon pitchclasses between a pair chords (divided by two) and the distance between the chord roots on the circle of fifths [Ler01]. We will therefore use $1-\mathbb{L}_{\widetilde{\mathbf{p}}}$, to make a comparison.

The seventeen chord sequences in table 5.4 are each 12 bars long. Half bars may have separate chords so each sequence contains a total of 24 symbols. For two chord sequences A and B we calculate a distance measure $d_{\mathbb{L}}$ thus

$$d_{\mathbb{L}}(A,B)=\frac{\sum_{n=1}^{N_{\mathrm{seq}}}\left(1-\mathbb{L}_{\widetilde{\mathbf{p}}}(A_n,B_n)\right)}{N_{\mathrm{seq}}} \tag{5.82}$$

where N_{seq} is the length of the sequences, which in this case is 24. Table 5.3 shows the TPSD scores d_{TPSD} for the seventeen chord sequences in table 5.4 taken from [DHVW08] compared with measures for $d_{\mathbb{L}}$. The

Table 5.3: Results of d_{TPSD} and $d_{\mathbb{L}}$ functions for the seventeen chord sequences in table 5.4. Results for $d_{\mathbb{L}}$ are multiplied by 13 for direct comparison with d_{TPSD}.

Sequence i	$d_{\mathrm{TPSD}}(i,1)$	$d_{\mathbb{L}}(i,1) \times 13$	$d_{\mathrm{TPSD}}(i,i-1)$	$d_{\mathbb{L}}(i,i-1) \times 13$
1	0	0	0	0
2	0.42	1.86	0.42	1.86
3	1	2.79	0.67	2.79
4	1.58	3.51	0.58	0.72
5	1.62	3.3	0.12	0.79
6	2.31	4.59	0.69	2.32
7	2.75	5.31	1.1	2.84
8	3.31	6.04	0.56	1.08
9	3.17	6.32	0.56	2.91
10	4.29	7.26	2.12	6.25
11	5.12	7.99	2.08	5.81
12	4.88	8.46	1.5	3.31
13	5.23	8.67	1.48	4.21
14	4.4	7.62	1.79	5.21
15	4.98	8.69	0.75	2.45
16	5.42	8.58	1.94	5.08
17	5.71	9.44	2.88	9.93

distances between sequence 1 and each other sequence $d_{\mathrm{TPSD}}(i,1)$ and also between each pair of consecutive sequences $d_{\mathrm{TPSD}}(i,i-1)$ were presented in [DHVW08] so we compare these values with $d_{\mathbb{L}}(i,1)$ and $d_{\mathbb{L}}(i,i-1)$. The range of $d_{\mathbb{L}}$ is between 0 and 1 whereas the range of d_{TPSD} is 0 to 13 so we show the $d_{\mathbb{L}}$ results multiplied by 13 to compare the values directly. Using this scaling we find that the $d_{\mathbb{L}}$ gives relatively large distance values compared to d_{TPSD} as can be seen in in the bar charts in figure 5.4. However, this is not particularly surprising since $1-\mathbb{L}_{\widetilde{\mathbf{p}}}$ effectively depends on the number of uncommon tones between two chords but the TPS chord distance depends upon the number of uncommon tones divided by two. If we normalise the results by the largest value in each column and plot the two functions again (see bar charts from figure 5.5) we find that their results actually look quite similar in terms of relative distances between chord sequences.

By making this comparison with the TPSD, we aim to show empirically that function $d_{\mathbb{L}}$ measures chord distance with results comparable to

another existing method of measurement. We note however that the blues progressions presented are fairly simple and the two distance functions will behave differently when comparing more unlikely chord combinations. For example, comparing a pair of chords with d_{TPSD} can produce a maximum score (i.e. 13) when a C major chord is compared with a chord containing every note in the chromatic scale based on E (the major third of C). This combination of chords has three shared tones so $d_{\mathbb{L}}$ will be $\frac{3}{4}$ (which scales to 9.75 when multiplied by 13 if we wish to compare it directly with d_{TPSD}). In contrast, the maximum value for $d_{\mathbb{L}}$ will be produced by any pair of chords that share no common tones. It should also be noted that the TPSD is key context specific whereas $\mathbb{L}_{\widetilde{\mathbf{p}}}$ compares pairs of chord symbols independent of key. We can compare the two functions in the way presented here because all of the blues progressions in table 5.4 are notated in the same key. To compare chord sequences in different keys with $d_{\mathbb{L}}$ would require that our chord representation be made relative to a key centre. Although this is possible, it is not the primary motivation for development of the chord likeness function $\mathbb{L}_T$ and is therefore not something that we will pursue further here.

Table 5.4: Seventeen 12-bar blues chord sequence variations. Bars with no notated chord symbol repeat the chord from the previous bar.

Sequence i	**Bar 1**	**Bar 2**		**Bar 3**		**Bar 4**		**Bar 5**	**Bar 6**	**...**
1	F:7							Bb:7		
2	F:7							Bb:7		
3	F:7	Bb:7		F:7				Bb:7		
4	F:7	Bb:7		F:7				Bb:7		
5	F:7	Bb:7		F:7				Bb:7		
6	F:7	Bb:7		F:7				Bb:7	Eb:7	
7	F:7	Bb:7		F:7		C:min7	F:7	Bb:7	Eb:7	
8	F:7	Bb:7		F:7		C:min7	F:7	Bb:7	Eb:7	
9	F:7	Bb:7		F:7		C:min7	F:7	Bb:7	B:min7	E:7
10	F:maj7	E:min7	A:7	D:min7	G:7	C:min7	F:7	Bb:7	B:dim	
11	F:maj7	E:min7	Eb:min7	D:min7	Db:min7	C:min7	Cb:7	Bb:maj7	Bb:min7	
12	F:maj7	Bb:maj7		A:min7	G:min7	Gb:min7	Cb:7	Bb:maj7	Bb:min7	
13	F:maj7	Bb:maj7		A:min7	G:min7	Gb:min7	Cb:7	Bb:maj7	Bb:min7	Eb:7
14	F:maj7	E:min7	A:7	D:min7	G:7	C:min7	F:7	Bb:maj7	Bb:min7	Eb:7
15	F:maj7	E:min7	A:7	D:min7	G:7	Gb:min7	Cb:7	Bb:maj7	B:min7	E:7
16	F#:min7 B:7	E:min7	A:7	D:min7	G:7	C:min7	F:7	Bb:maj7	Bb:min7	Eb:7
17	F:maj7	F#:min7	B:7	E:maj7	Eb:maj7	Db:maj7	B:maj7	Bb:maj7	B:min7	E:7

...	**Sequence** i	**Bar 7**	**Bar 8**		**Bar 9**		**Bar 10**		**Bar 11**		**Bar 12**	
	1	F:7			C:7				F:7			
	2	F:7			C:7		Bb:7		F:7		C:7	
	3	F:7			G:7		C:7		F:7		C:7	
	4	F:7	D:7		G:7		C:7		F:7		C:7	
	5	F:7	D:7		G:min7		C:7		F:7		G:min7	C:7
	6	F:7	D:7		Db:7		C:7		F:7		Db:7	C:7
	7	F:7	A:min7	D:7	G:min7		C:7		A:min7	D:7	G:min7	C:7
	8	A:min7	D:7		G:min7		C:7		A:min7	D:7	G:min7	C:7
	9	F:7 E:7	Eb:7	D:7	G:min7		C:7	Bb:7	A:min7	D:7	G:min7	C:7
	10	A:min7 D:7	Ab:min7	Db:7	G:min7	C:7	Db:min7	Gb:7	F:7	D:7	G:min7	C:7
	11	A:min7	Ab:min7		G:min7		C:7		A:min7	Ab:min7	G:min7	Gb
	12	A:min7	Ab:min7		G:min7		Gb:7		F:maj7	Ab:min7	G:min7	Gb
	13	Ab:maj7	Ab:min7	Db:7	Gb:maj7		G:min7	C:7	A:min7	D:7	Db:min7	Gb
	14	A:min7	Ab:min7	Db:7	G:min7		C:7		A:min7	D:7	G:min7	C:7
	15	A:min7	Ab:min7	Db:7	G:min7		C:7	Bb:7	A:min7	D:7	G:min7	C:7
	16	Ab:maj7	Ab:min7	Db:7	Gb:maj7		G:min7	C:7	A:min7	D:7	G:min7	C:7
	17	A:maj7	A:min7	D:7	G:maj7		Gb:maj7		F:maj7	Ab:maj7	G:maj7	Gb

Chapter 6

A reference transcription dataset

To enable rigorous testing of chord and harmony recognition algorithms it is necessary to have a large hand-transcribed dataset which can be used as ground truth information for comparison with computer algorithm outputs. In this chapter we will describe the process of creating such a dataset and discuss the resulting collection of transcriptions.

6.1 The Beatles studio albums

The corpus chosen for the transcription project was the twelve studio albums by The Beatles. To be precise, the audio that was used was taken from the original CD releases of each album first issued in 1987, catalogue numbers for which are given in table 6.1. This collection comprises 180 songs over 13 CDs (the 'White album' is a double disc) totalling 8 hours, 8 minutes and 53 seconds of audio (or 29333 seconds). The total running time for each of the discs is shown in table 6.2.

Why use the Beatles?

We needed to select a collection of music to use as the test corpus for our chord recognition work. There are many factors to consider when making such a decision. One important one is the fact that the author has to listen to each song many, many times during the transcription process so it is a good idea to choose material that you like. The author liked the Beatles[1] and the research group owned a copy of the twelve studio albums

[1] In fact, after completing the transcriptions and also using them extensively in this research, the author *still* likes the Beatles despite the very high level of exposure to them over the course

Table 6.1: Titles and catalogue numbers for the Beatles CDs used in the transcription process (albums shown in original order of release)

Album title	**CD Catalogue number**
Please Please Me	CDP 7 46435 2
With the Beatles	CDP 7 46436 2
A Hard Day's Night	CDP 7 46437 2
Beatles For Sale	CDP 7 46438 2
Help!	CDP 7 46439 2
Rubber Soul	CDP 7 46440 2
Revolver	CDP 7 46441 2
Sgt. Pepper's Lonely Hearts Club Band	CDP 7 46442 2
Magical Mystery Tour	CDP 7 48062 2
The Beatles (the white album)	CDS 7 46443 8
Abbey Road	CDP 7 46446 2
Let It Be	CDP 7 46447 2

Table 6.2: Total run time for each Beatles CD used in the transcription process.

Disc	**Album**	**time (s)**	**time (mins:secs)**
01	Please Please Me	1965.84s	32:45
02	With the Beatles	2004.32s	33:24
03	A Hard Day's Night	1830.01s	30:30
04	Beatles for Sale	2053.41s	34:13
05	Help!	2061.06s	34:21
06	Rubber Soul	2148.1s	35:48
07	Revolver	2099.3s	34:59
08	Sgt. Pepper's Lonely Hearts Club Band	2390.57s	39:50
09	Magical Mystery Tour	2209.88s	36:49
10CD1	The Beatles	2781.91s	46:21
10CD2	The Beatles	2834.08s	47:14
11	Abbey Road	2844.08s	47:24
12	Let It Be	2110.88s	35:10

so these were quite important initial factors in choosing them as the test corpus.

There are many other practical reasons why the Beatles albums were felt to be a good corpus for the chord transcription project. The albums

of this work. This is a testament to the quality of the songwriting and production in itself.

are widely available in most parts of the world so other researchers should not have difficulty sourcing the same audio material. In addition to this, there is already a large body of research work on analysis of the music of The Beatles in terms of music theory and criticism [Pol, Ped03]. This made the transcription process a little easier because harmonic analyses were available for a large number of the songs [Pol] and these served as a useful starting point for the transcription work.

The albums cover the development of the band from their first recordings in 1963 to their final sessions in 1970 during which time they were at the forefront of new music recording and production techniques resulting in a very wide variety of sounds, styles, effects and timbres being present in a small but coherent corpus. As an example of the wide diversity that can be found in the collection, we may consider the contrasts between the traditional guitar-based rock and roll of songs like "I saw her standing there", the opening track on *Please Please Me*, and classical-music influenced arrangements of *Revolver*'s "Eleanor Rigby" or *Sgt. Pepper's Lonely Hearts Club Band*'s "She's leaving home". Likewise we can compare these styles with the psychedelic phasing sound effects of *Sgt. Pepper's Lonely Hearts Club Band*'s "Lucy in the sky with diamonds" or the proto heavy metal of "Helter Skelter" and music concrète of "Revolution nine" from *The Beatles* (more commonly known as "The white album"). Such was their impact on the world of popular music that it would be difficult for artists who have followed in the genre since to legitimately claim not to have been influenced either directly or indirectly by the Beatles in some way.

A fair criticism that can be levelled at a corpus containing only one artist's output is that it is the same voices on all songs. Although this is true, the Beatles were well known for trying to avoid what Pollack calls "foolish consistency" [Pol] throughout their work and both the vocal styles and the instruments that were used vary quite considerably through the collection.

Another aspect of the Beatles work that makes them a good candidate for chord recognition tests compared to other popular music artists is that their songwriting often includes complex harmonic progressions. It

was their particular skill in wrapping melody lines in interesting harmonic structures that produced such memorable songs. In the author's opinion it is also what makes the collection particularly interesting as a test corpus for chord recognition.

6.2 Transcription file format

The file format that was chosen for the Beatles chord transcriptions is the Wavesurfer[2] '.lab' file. This is a flat ASCII text file with each line representing a labelled time segment in an annotation. The arrangement of data in each line of a '.lab' file is:

```
start-time      end-time      label
```

where the start and end times are given in seconds and, in the case of the chord transcription files, the labels are chord symbols which conform to the syntax defined in section 4.2.

Although the '.lab' file allows unlabelled time gaps to exist between segments and also for segments to overlap in time, the Beatles transcription files contain only contiguous sequences of non-overlapping chord segments. Any non-chordal section of the music is given the label `"N"`.

6.3 The transcription process

To transcribe the Beatles songs by hand, a four stage process was followed for each song. The four stages are:

1. Familiarisation
2. Aural transcription
3. Chord boundary tapping
4. Chord segment labelling

[2] C4DM's Sonic Visualiser, a more recent audio visualisation and analysis application which was used extensively later in the transcription process is also compatible with .lab files.

In the familiarisation stage, the transcriber listens to the song a number of times until they become familiar with the structure in terms of both harmonic rhythm and also the order of sections such as verses, bridges and choruses etc.

In the aural transcription stage, detailed chord sequences for each section of the song are written out. This stage was done by carefully listening to each section of the audio and transcribing each chord individually. This time consuming job was made much easier by being able to refer to the comprehensive notes on the Beatles recordings from Alan Pollack [Pol]. Pollack's notes contain harmonic analyses of many of the songs which served as a very good starting point for this part of the task.

With the chord sequences transcribed, the chord start times can be recorded by tapping keys on the computer keyboard while listening to the audio. The keystrokes are recorded as timestamps relative to the start of the audio file and are saved in '.lab' files with dummy text labels which will be altered afterwards. The initial output lab file after tapping will look something like this:

```
0.000000 1.370000 STAMP1
1.370000 1.915668 STAMP2
1.915668 2.519387 STAMP3
...
```

After the initial recording of the timestamps, the new transcription files can be loaded into an audio annotation program and the timestamps can be altered to correct for inaccuracies. In actual fact, due to latency issues with the java application used to play the audio and record the keystrokes, it was found that all timestamps had a small offset of about +4ms. The transcriptions were therefore loaded into Matlab and all timestamps were shifted to correct for this offset before making finer adjustments by hand. The initial transcription work was done using the audio annotation tool Wavesurfer [SB00] which is why the '.lab' files were chosen as the transcription format. Later work on the transcriptions was done using Sonic Visualiser [CLSB06] which brought a greater level of timing accuracy to the transcription collection because it allows annotation boundaries to be

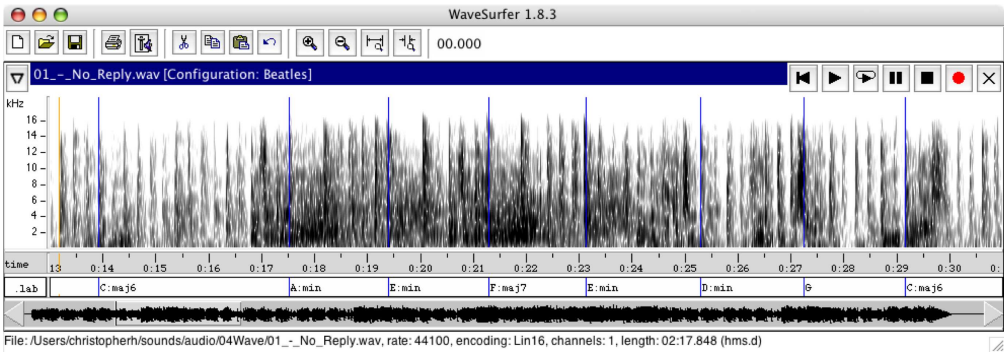


Figure 6.1: Chords from the transcription file of the Beatles' *No Reply* shown in Wavesurfer

sonified using audio samples such as drum stick clicks. Our ears are very sensitive to small timing differences (down to a resolution of around 10ms [HA96]) between the events in the audio track and these sonified chord onsets. This is a much more accurate method of aligning the chord onsets than using our eyes to try to locate and adjust timing differences represented as lines on visualisations of the audio such as waveforms and spectrograms. One reason for this is that sonifying the onsets means the timing can be judged *in time*, which is the correct dimension, instead of time being represented on the x-axis of a visualisation on the computer screen. It also means that only one sensory modality is used to detect the timing differences instead of having to try to detect differences between the auditory information from the sound playback and the visual information on the screen.

Once the timestamps are judged to be in the right places, the dummy labels can be replaced with the chord symbols worked out in the aural-transcription stage. Figure 6.1 is a screen shot showing a section of the transcription file for "No Reply", from the album *Beatles for sale*, in Wavesurfer.

6.4 Transcription policy

To transcribe such a large collection manually takes a lot of time and effort so it is important that certain transcription policies are decided at

the beginning in order to keep the annotations as consistent as possible.

6.4.1 Treatment of the melody line

In general, the chord labels in transcriptions correspond to the chord which would be written on a lead sheet for musicians to play from. That is to say, the melody line itself is considered to be separate from the harmony. For example, if the musical instruments are playing a C major chord with no sevenths or extensions but the melody line includes a B♭ (the flattened seventh) we ignore the melody and simply label the chord C major.

6.4.2 Consistent inconsistencies

The transcriptions are intended to represent which chords are perceived to be present in the audio at any given time. Each song has been listened through and analysed second by second which means that in some cases, where a musical score or lead sheet might suggest a direct repeat of material, if the audio differs slightly in the repeat then the transcription will reflect this. For example, in "Glass Onion" on the first disc of the white album there is a two bar phrase that occurs at the end of each refrain; the lead vocal sings "look into a glass onion" with chord label '`F:7`' for the first bar and '`G`' for the second. Each time the phrase occurs there is a two-crotchet fill on the drums on beats three and four of the second bar. In the first two instances of this pattern (at times 29.33 and 59.42 seconds respectively) the instruments that play the '`G`' chord fade through the drum fill and there are also clearly audible notes being played beneath it on the bass guitar which support the chord. On the third repeat however (time 1:18.7), the instruments cut off very abruptly as if faded out deliberately. This time the drum fill has no accompanying harmonic material so those two beats are labelled as an '`N`' chord. Figure 6.2 shows the three instances of the phrase viewed in Wavesurfer and all have very similar spectrograms. On first glance it would appear that the final repeat has been labelled inconsistently but by listening carefully to the audio in

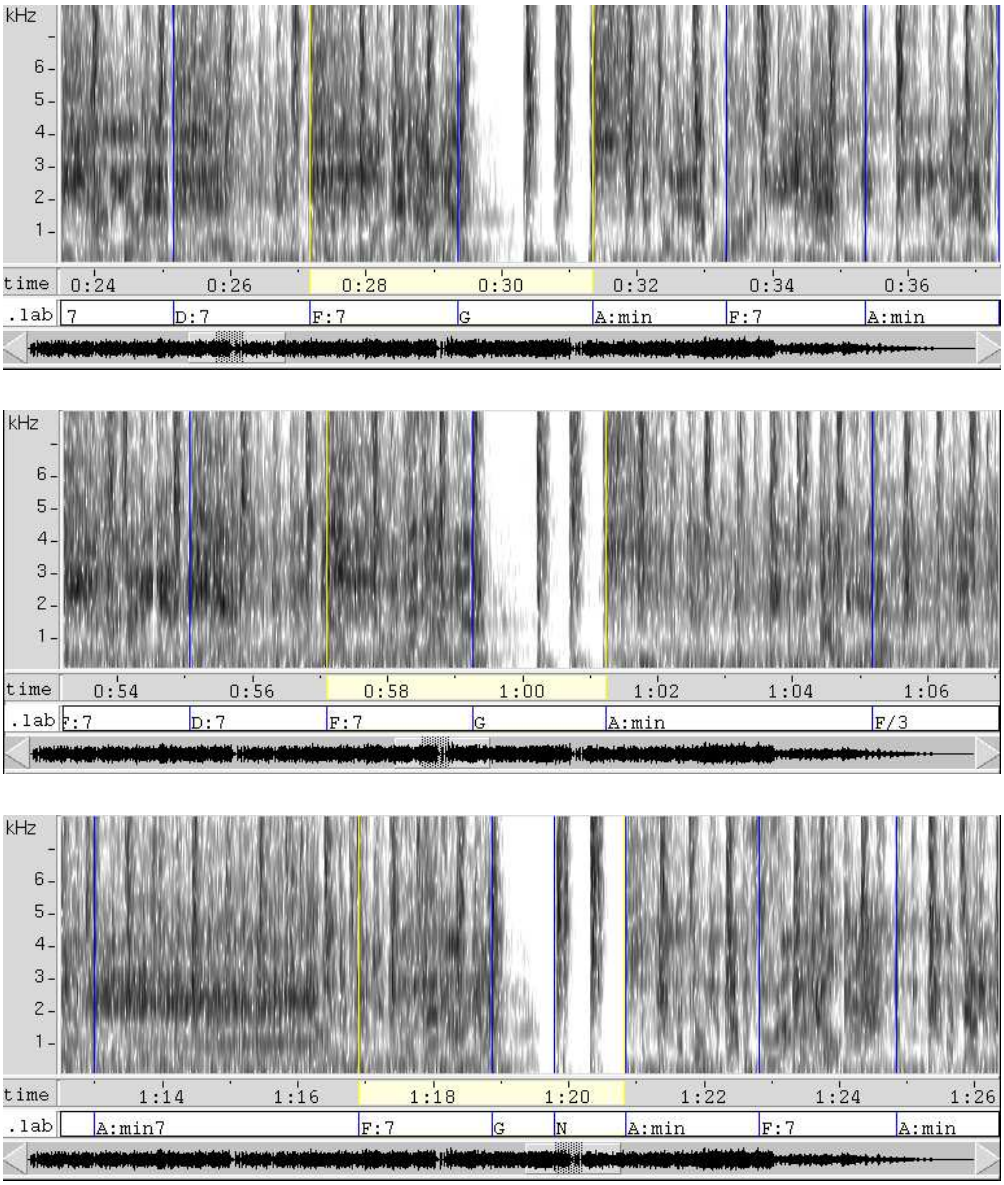


Figure 6.2: Three instances of a repeated phrase in *Glass Onion* with what, at first glance, appears to be inconsistent labelling for the final instance (bottom).

each instance we can hear that this is not the case[3].

[3]Another example of this kind of apparent inconsistency can be found in the transcription of "I'm so tired" also on the first disc of the white album at times 1:45 and 1:51.

6.5 Verification of transcriptions

The initial creation of the large collection of chord transcriptions was the first step in producing a dataset that can be used as a ground truth for evaluation. However, such a large data set created by hand is bound to contain a certain number of errors. One kind of error is typographical mistakes which can cause chord symbols to deviate from the correct format. For example, it would be easy to accidentally type the label '`C;maj7`' with a semicolon instead of the correct '`C:maj7`' with a colon separator. Identifying and fixing such errors is simple using computer programs to automatically check the syntax of the chord symbols (see the discussion of the chord tools in section 4.4). Another type of error is where the chord symbols are correctly formatted but a chord label has been missed out from the sequence causing all the following labels to be shifted out of line by one place. This kind of error can cause an otherwise accurate transcription to be completely wrong after the missing chord label and as such is sometimes possible to see when looking at the transcription in an audio annotation program. A more subtle error still is where a chord label is correctly formatted but a modifier has been missed off the root note, for example '`B:min`' instead of '`Bb:min`'. This kind of mistake is very difficult to spot by just looking at the chord labels but if you can sonify the transcriptions then they are very noticeable when compared to the original audio.

To ensure that the transcriptions were as free from mistakes as possible required a rigorous verification process. It would have taken a very long time for someone to manually check every label (effectively re-transcribing the songs) so an alternative method was devised. After the files had all been checked and corrected for chord syntax, the following process was followed for every file:

1. Convert chord transcription to standard MIDI file
2. Synthesise the MIDI file as digital audio
3. Combine synthesised chords audio with original Beatles audio
4. Conduct human verification listening test

5. Correct identified mistakes

6. Repeat stages 1 to 5 until no more mistakes are identified

6.5.1 Converting chord transcriptions to MIDI files

The chord transcription files were converted to standard MIDI files using functions from the chord tools discussed in section 4.4 and the University of Jyväskylä Matlab MIDI toolbox[4] by Tuomas Eerola and Petri Toiviainen [ET04].

6.5.2 Synthesising MIDI files as digital audio

Once the MIDI files had been created, we used the open source software synthesiser Timidity[5] to synthesise the MIDI events to digital audio. This task was not straightforward however because the aim of the process was to produce wave files of synthesised chords that could be played along with the original Beatles recordings for comparison. This meant that the final synthesised audio needed to be both in tune with the original recordings and also accurately synchronised with them in time.

The default setting for tuning in Timidity is to use the standard concert A = 440Hz as a reference. Many of the Beatles songs deviate from this tuning however; some because the band's instruments may have been tuned to themselves in the absence of any other reference and some because the song was mastered at a different tape speed to that at which it was recorded in order to produce an interesting effect[6].

In order to synthesise audio with the same tuning as the original recordings, a secondary annotation task was therefore introduced to the project: the tuning reference frequency must be determined for each song in the collection. For most songs in the collection this was a fairly straight forward task because the tuning remained constant throughout the song. However, in the case of 'Strawberry Fields Forever' from *Magical Mystery*

[4] https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/miditoolbox

[5] http://timidity.sourceforge.net/

[6] 'When I'm Sixty Four' is a good example of this. The tape speed was deliberately increased during mastering in order to make McCartney's voice sound more brittle. This raised the key by almost a semitone [Mac08]

Tour and *Abbey Road*'s 'The End', the tuning actually alters during the songs where material from different recording sessions have been spliced together to create the final track[7].

Retuning the MIDI files was accomplished using the MIDI *pitch-bend* control which alters the tuning pitch of a MIDI instrument. MIDI pitch-bend can be used as a real-time continuous control so it has a 14-bit resolution making it fine enough to be used dynamically with wide pitch-bend ranges without hearing the graduations.

To calculate the correct pitch bend value for retuning a MIDI file we first find the difference in cents between the tuning frequency used on the recording and 440Hz. To find the difference between two frequencies, f_1 and f_2, in cents we use the following equation:

$$\text{cents} = 1200.\log_2(\frac{f_1}{f_2}) \tag{6.1}$$

The standard setting for MIDI pitch-bend (and the one that Timidity defaults to) is a full scale bend range of ± 1 whole tone which is equivalent to ± 200 cents. The 14-bit resolution gives a total of 16384 possible pitch bend values with the mid-range value 8192 representing no bend. Thus we find

$$\frac{8192}{200} = 40.96 \text{ pitch bend units per cent.} \tag{6.2}$$

For the annotated tuning reference value f_{ref}, we may therefore calculate the pitch bend value to alter the MIDI file with in the following way:

$$\text{pitch bend value} = 8192 + \big(40.96 \times 1200 \times \log_2(\frac{f_{\text{ref}}}{440})\big) \tag{6.3}$$

The resulting pitch bend message is inserted into the main track of the MIDI file at the beginning of each song to alter the tuning when the wave file is synthesised.

With the tuning corrected, the MIDI file can then be synthesised as audio but it is then necessary to synchronise the result with the original

[7]In the case of 'Strawberry Fields Forever', the two separate takes that were used were actually recorded in completely different keys a whole tone apart. John Lennon decided he liked the start of one take and the end of the other which resulted in the engineers having to speed up one and slow the other one down so they were in roughly the same key at the edit. [Mac08, Lew92]

recordings. The wave files produced using Timidity begin where the first audible event happens (i.e. the first note that is synthesised). This means that where transcriptions have a silence at the beginning (denoted by `N` in the transcription), the synthesised wave file will not be synchronised properly with the original audio. To correct for this, each synthesised audio file was loaded into Matlab and the correct number of zero-amplitude samples prepended to realign it.

6.5.3 Listening tests

To verify the accuracy of the transcriptions, we combined a mono version of the original Beatles audio with the MIDI synthesised audio. The original audio was put in the left stereo channel and the synthesised audio in the right channel. These files were then given to a group of twenty volunteers to listen to and note anything that sounded incorrect to them. The volunteer listeners were all people who listened to music regularly but were not necessarily trained musicians. The listeners were not expected to provide detailed explanations of what they thought was wrong, they were simply asked to note the approximate time of what they perceived as possible errors. To prepare listeners for the task, a simple training example was produced which had two wave files of the same song; one file was completely correct but the other included incorrect chord labels and timing errors to demonstrate the kinds of mistakes to listen for.

In the first verification stage, the audio files were distributed among the listeners so that each song would be listened to once. Distribution of files was done on a track by track basis so that each volunteer would hear songs from all twelve albums. Any mistakes that were noted at this stage were found and fixed. The complete set of transcriptions were then re-synthesised as wave files and given to the volunteers again to check in a second stage. In the second stage, the tracks were distributed such that the majority of listeners did not have to check the same song twice (for two songs, this was not possible due to the availability of volunteers so the author double checked the second stage for those tracks as well). After the second stage was completed, all remaining mistakes that were identified

were fixed before the collection was released.

By following this verification process and making sure that each song was checked by at least two different people we can be fairly confident that most significant errors have been caught. In terms of audible mistakes caused by bad timing or incorrect chord labels, we may be fairly sure that the collection is reliable. However, it would be impossible to claim that the collection was perfect simply because even if all these errors have been removed, the transcriptions represent one person's opinion of what the chords are and other musicians may disagree with labelling decisions in various places. Since the collection has been made available for the whole research community to use, it is hoped that any further errors that may still exist will be reported so that the accuracy of the collection may continue to improve over time.

6.6 Transcription collection statistics

In this section we will look at some of the statistics of the transcription collection. These details are important because we wish to know something about the nature of the data in the collection before using it as a ground truth for evaluation purposes.

At the time of writing[8], the transcription collection contains a total of 14621 individual chord labels covering 8 hours, 8 minutes and 53 seconds of audio material (or 29333 seconds).

6.6.1 Chord cardinality

Before examining the statistics of unique chord symbols and chordtypes we shall look at the more general property of chord cardinality in the collection.

The distribution of cardinalities for the whole collection is given in table 6.3. We include non-chord '`N`' in the statistics as a zero cardinality category. The pie chart in figure 6.3 represents these values graphically.

[8]Some values in this section may alter very slightly over time because the collection is updated periodically if errors are identified.

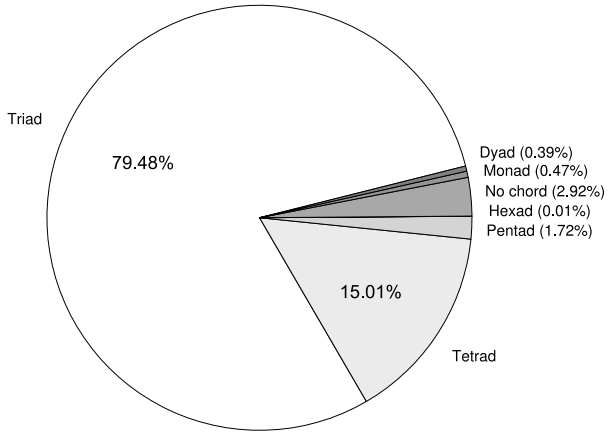


Figure 6.3: Pie chart showing the distribution of chord cardinalities for the whole collection.

Table 6.3: Numbers of chord symbols of each different cardinality in the Beatles transcription collection.

Cardinality	**Symbol Count**	**Percentage**
0 (No chord)	427	2.92%
1 (Monad)	69	0.47%
2 (Dyad)	57	0.39%
3 (Triad)	11621	79.48%
4 (Tetrad)	2194	15.01%
5 (Pentad)	252	1.72%
6 (Hexad)	1	0.01%

We find that all cardinalities between 0 and 6 are represented at least once in the collection. Nearly 80% of the chords in the collection are triads, 15% tetrads, 2.9% are the non-chord '`N`' and the remaining 2.1% is shared between the other cardinalities. Figure 6.4 shows a bar graph of the distributions of chord cardinalities for each album. In it we can see that the later albums tend to have higher numbers of tetrads than the earlier ones with the exception of disc 2 of the white album. It is also interesting to note that 'Let It Be' is the only album in the collection that contains no pentad chords.

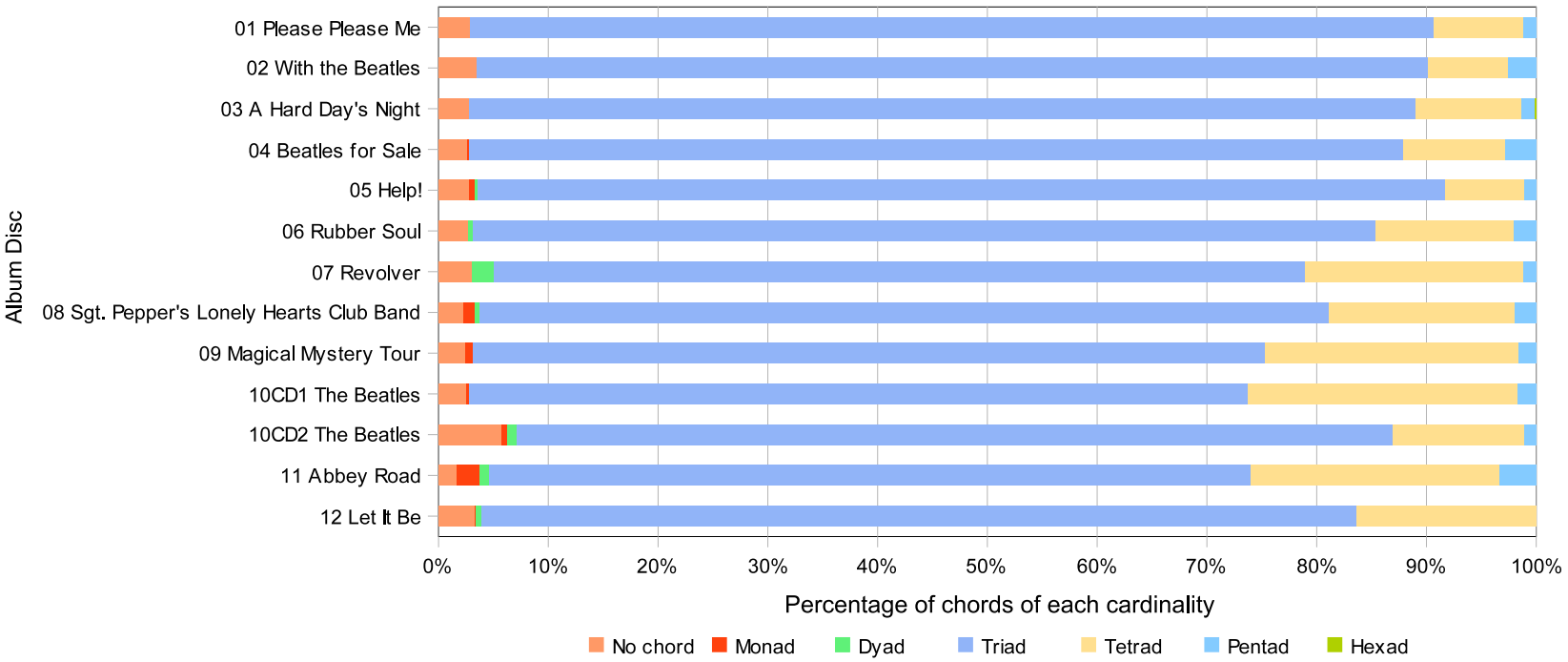


Figure 6.4: Bar chart showing the distributions of chord cardinalities in each album. Lower cardinalities are on the left, higher on the right.

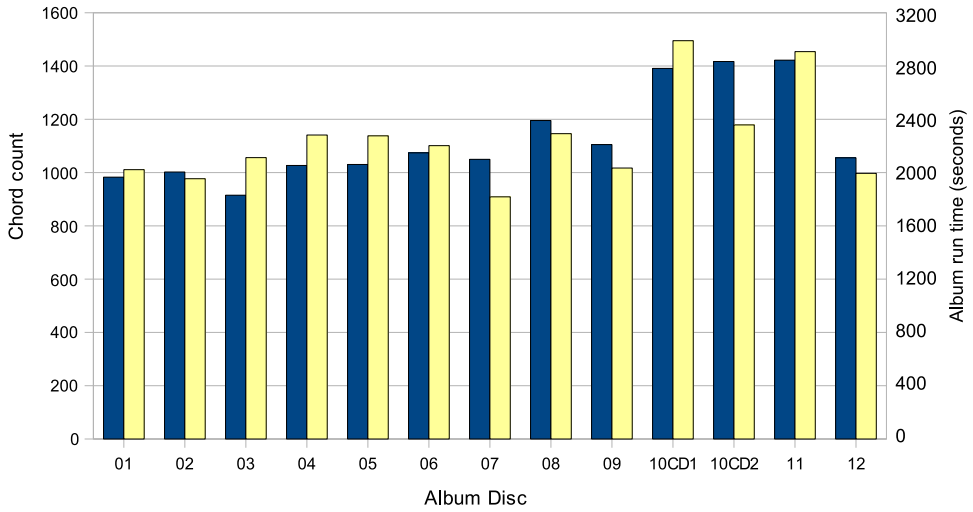


Figure 6.5: Bar chart showing total number of transcribed chord symbols (dark blue) compared to run time in seconds (light yellow) for each album.

6.6.2 Chord symbol counts: frequency and duration

The total number of transcribed chord symbols in the collection is 14621. The total chord counts for each individual album are shown in column 'TSC' of table 6.4. These values are shown graphically in the bar chart of Figure 6.5 where they are compared with the run times of the albums (presented earlier in table 6.2). The two sets of values are closely correlated with a mean time period across the whole collection of approximately 2 seconds per chord.

We will now use the chord comparison methods described in chapter 5 to calculate statisitics for the number of unique chords and chordtypes in the Beatles transcription collection.

Unique chords

The 14621 chords in the collection comprise 406 unique chord symbol strings. This value has been calculated using direct string comparison, $\mathbb{M}_{\mathbf{s}}$, of chord labels across the whole collection. The numbers of unique chord symbols for each album are shown in the $\mathbb{M}_{\mathbf{s}}$ column of table 6.4. A table showing the full 406 unique symbols with their related frequency

Table 6.4: Chord count statistics for each album in the collection.

Disc	**Album**	TSC	$\mathbb{M}_{\mathbf{s}}$	$\mathbb{M}_{\overrightarrow{\mathbf{v}}}$	$\mathbb{M}_{\overrightarrow{\mathbf{p}}}$	$\mathbb{M}_{\widetilde{\mathbf{v}}}$	$\mathbb{M}_{\widetilde{\mathbf{p}}}$	$\mathbb{M}'_{\mathbf{s}}$	$\mathbb{M}'_{\overrightarrow{\mathbf{v}}}$	$\mathbb{M}'_{\overrightarrow{\mathbf{p}}}$	$\mathbb{M}'_{\widetilde{\mathbf{v}}}$	$\mathbb{M}'_{\widetilde{\mathbf{p}}}$
01	Please Please Me	1011	45	45	45	39	39	39	39	39	38	38
02	With the Beatles	977	62	62	62	53	53	52	51	51	51	51
03	A Hard Day's Night	1056	51	51	51	42	42	42	42	42	42	42
04	Beatles for Sale	1141	62	62	62	49	49	46	46	46	44	44
05	Help!	1138	60	58	58	45	45	46	45	45	44	44
06	Rubber Soul	1101	67	67	65	58	55	59	58	56	57	54
07	Revolver	909	77	75	71	67	63	63	60	56	60	56
08	Sgt. Pepper's Lonely Hearts Club Band	1146	110	107	104	73	70	74	73	71	71	68
09	Magical Mystery Tour	1017	95	93	93	63	62	53	52	52	49	48
10CD1	The Beatles	1495	112	111	109	84	81	75	75	73	73	69
10CD2	The Beatles	1179	92	92	92	72	72	69	69	69	66	66
11	Abbey Road	1454	123	119	115	83	78	73	73	69	71	66
12	Let It Be	997	68	68	65	52	49	52	52	49	50	47
All	Whole collection	14621	406	364	346	227	202	246	234	219	206	180

Key:

TSC	Total symbol count		
$\mathbb{M}_{\mathbf{s}}$	String matching	$\mathbb{M}'_{\mathbf{s}}$	Bass-blind string matching
$\mathbb{M}_{\overrightarrow{\mathbf{v}}}$	Ordered pnset matching	$\mathbb{M}'_{\overrightarrow{\mathbf{v}}}$	Bass-blind ordered pnset matching
$\mathbb{M}_{\overrightarrow{\mathbf{p}}}$	Ordered pcset matching	$\mathbb{M}'_{\overrightarrow{\mathbf{p}}}$	Bass-blind ordered pcset matching
$\mathbb{M}_{\widetilde{\mathbf{v}}}$	Unordered pnset matching	$\mathbb{M}'_{\widetilde{\mathbf{v}}}$	Bass-blind unordered pnset matching
$\mathbb{M}_{\widetilde{\mathbf{p}}}$	Unordered pcset matching	$\mathbb{M}'_{\widetilde{\mathbf{p}}}$	Bass-blind unordered pcset matching

and duration information can be found in the accompanying online information[9].

Of course, direct string comparison does not necessarily tell us the real story in terms of tonal content because alternative spellings for the same chord (for example '`D#:min`' and '`D#:(1,b3,5)`') will be interpreted as separate symbols. When we use ordered pnset comparison $\mathbb{M}_{\overrightarrow{\mathbf{v}}}$ we find that there are in fact 364 unique pnsets in the collection therefore 42 of the 406 unique symbol strings have equivalent spellings in the collection. The numbers of unique pnsets for each album are given in the $\mathbb{M}_{\overrightarrow{\mathbf{v}}}$ column of table 6.4.

We can also use ordered pcset matching function $\mathbb{M}_{\overrightarrow{\mathbf{p}}}$ to count the number of enharmonic equivalent chord symbols in the collection. In this case, chords such as '`D#:min`' and '`Eb:min`' will be considered the same because both contain the same ordered set of pitch classes {3,6,10}. The number of unique ordered pnsets is 346 so 60 of the 406 unique chord symbol strings have enharmonic equivalents in the collection.

It is also interesting to look at the unordered pnset and pcset counts ($\mathbb{M}_{\widetilde{\mathbf{v}}}$ and $\mathbb{M}_{\widetilde{\mathbf{p}}}$) for the collection. These values tell us about the pitchname and pitchclass content of the symbols without restricting the order of the elements in pnsets and pcsets. An example of chords which would be considered equivalent by this function are '`C#:maj6`' and '`A#:min7`', both of which contain pitchnames `"A#"`,`"C#"`,`"E#"` and `"G#"` but in different orders. There are 227 unique unordered pnsets and 202 unique unordered pcsets. The values of these counts for each album are shown in the $\mathbb{M}_{\widetilde{\mathbf{v}}}$ and $\mathbb{M}_{\widetilde{\mathbf{p}}}$ columns of table 6.4.

The second half of table 6.4 contains results for the same five matching functions again but this time with each calculated bass-blind (i.e. ignoring the bass interval of chords if any are specified, see section 5.5). Counting using the bass-blind string matching function $\mathbb{M}'_{\mathbf{s}}$ gives a total of 246 unique chord symbols and thus we note that 160 of the 406 unique chord symbol strings include a specified bass interval. The bass-blind count for

[9] http://www.isophonics.net/content/reference-annotations-beatles

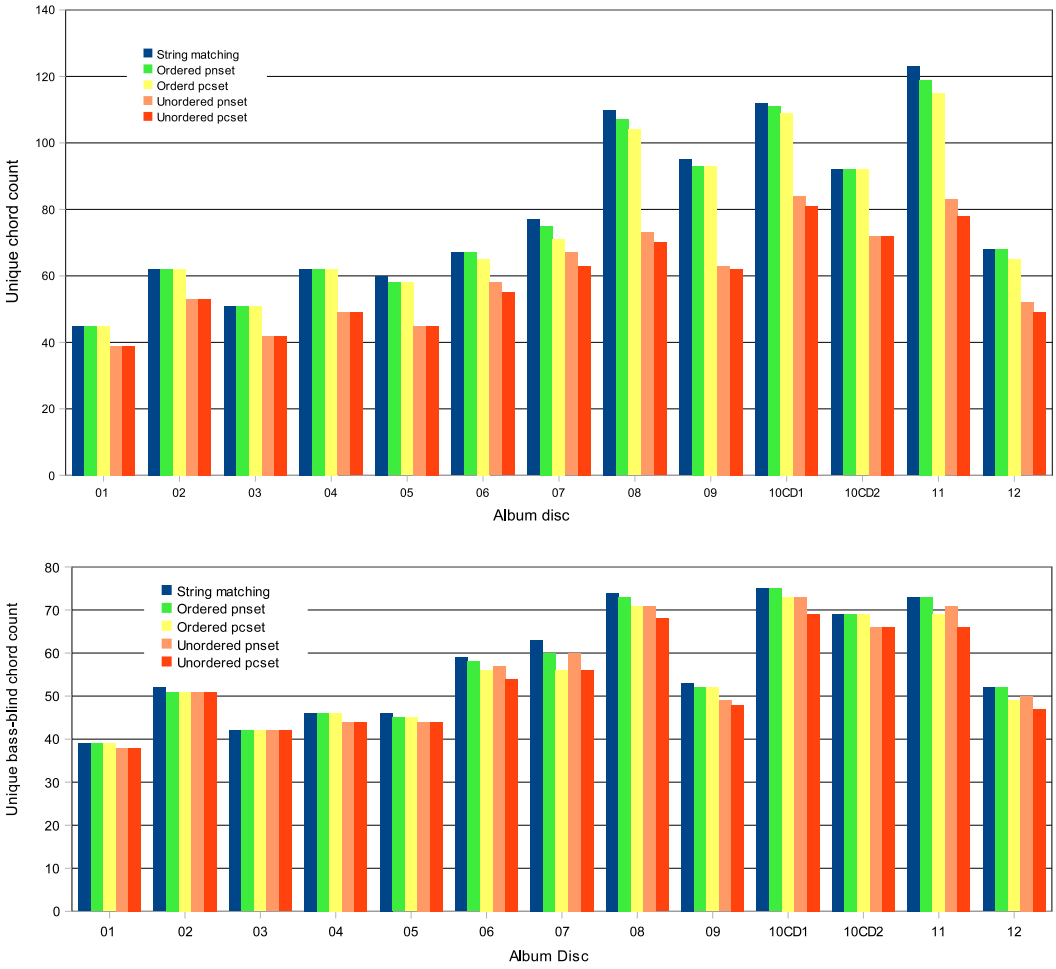


Figure 6.6: Bar charts showing unique chord symbol counts for each album using five different counting methods. Upper chart shows counts including bass intervals; lower chart shows bass-blind counts.

pnsets $\mathbb{M}'_{\overrightarrow{\mathbf{v}}}$ is 234 so there are 12 chord symbols that have pitchname equivalents out of the 246 unique bass-blind chord symbol strings. Likewise, the count for pcsets $\mathbb{M}'_{\overrightarrow{\mathbf{p}}}$ is 219 so 27 symbols have pitchclass equivalents out of the 246 chord symbols. We also calculate the values for bass-blind unordered pnsets and pcsets $\mathbb{M}'_{\widetilde{\mathbf{v}}}$ and $\mathbb{M}'_{\widetilde{\mathbf{p}}}$. The value for $\mathbb{M}'_{\widetilde{\mathbf{v}}}$ is 206 and the value for $\mathbb{M}'_{\widetilde{\mathbf{p}}}$ is 180.

The bar charts in figure 6.6 represent the unique symbol counts for each album graphically for all five counting methods; the upper chart showing counts including bass intervals and the lower chart bass-blind counts. We see that for all counting methods, the later albums generally had higher

numbers of unique chords than the earlier ones; 'Abbey Road' (disc 11) having the highest number. This trend might suggest that the Beatles use of harmony became more complex and their choice of key more varied as their songwriting developed over the lifetime of the group. In contrast, we note that 'Let It Be' has a low number of unique chords compared to the other later albums. This may reflect the 'back to basics' approach that the Beatles took on the Let It Be project [Lew92, Ped03, Mac08, Pol].

We see that the values for simple string comparison are the same or slightly higher than the ordered pnset and pcset comparisons in all cases as we would expect given that there are some equivalent spellings of chords in the collection. When we look at the unordered pnset and pcset counts in the upper chart of figure 6.6, these are lower in all cases as we would expect but we note that the difference between ordered and unordered counts are much larger for the later albums. This shows that the use of inversion was more prevalent in the later albums and supports the notion that Paul McCartney's bass lines were more inventive in terms of influencing harmonic function later in the Beatles' career [Mac08, Pol].

Looking at the lower chart in figure 6.6 we can see that the difference between ordered and unordered counts is much smaller for bass-blind comparisons. We also note that in the top chart, disc 11 (Abbey Road) has the highest number of unique chords by a margin of more than 10 chords over any other album for the ordered counting methods. However, when using unordered or bass-blind comparison for the counts, we find that discs 08 and 10CD1 (Sgt. Pepper's Lonely Hearts Club Band and the first disc of the White album) actually have very similar numbers to Abbey Road, some even slightly higher. Likewise, in the lower chart we note the comparitively low unique chord counts for disc 09 (Magical Mystery Tour). The ordered counts from the upper chart suggest that the album has a high unique chord count compared to some of the other albums but the bass-blind counts show that this is actually only due to the numbers of inverted chords on that album.

We will now look in more detail at the frequency and duration of the unique chord symbols in the collection. The statistics for both of these sets of data are shown in table 6.5 for the top 26 unique chord symbols (using

Table 6.5: Statistics for the top 26 out of the 406 unique chord symbols in the transcription collection counted using string matching of chord symbols. The final category 'Others' accounts for the other 380 unique symbols. Chords are listed in order of duration.

Chord	Symbol frequency	% frequency	Aggregate time	% time
A:maj	1568	10.72	3311.82	11.29
E:maj	1039	7.11	2714.06	9.25
G:maj	1385	9.47	2685.85	9.16
D:maj	1353	9.25	2404.14	8.20
C:maj	966	6.61	2120.08	7.23
N	427	2.92	1312.32	4.47
B:maj	503	3.44	980.80	3.34
F:maj	489	3.34	870.38	2.97
A:min	365	2.50	738.16	2.52
Bb:maj	318	2.18	641.40	2.19
F#:min	291	1.99	583.10	1.99
B:min	293	2.00	539.71	1.84
D:min	197	1.35	533.48	1.82
E:min	336	2.30	518.00	1.77
C#:min	165	1.13	408.57	1.39
G:7	150	1.03	400.13	1.36
D:7	132	0.90	377.63	1.29
C#:maj	69	0.47	374.98	1.28
F#:maj	189	1.29	349.80	1.19
A:7	116	0.79	284.71	0.97
Eb:maj	161	1.10	281.09	0.96
Ab:maj	153	1.05	246.18	0.84
E:7	90	0.62	211.09	0.72
A:min7	94	0.64	209.20	0.71
Db:maj	89	0.61	194.37	0.66
F:min	84	0.57	191.80	0.65
Others	3599	24.62	5850.15	19.95

the $\mathbb{M}_{\mathbf{s}}$ count). Values are presented in both terms of absolute frequency counts and duration totals as well as percentages of the whole collection for both. Using ordered pnset $\mathbb{M}_{\overrightarrow{\mathbf{v}}}$ to count instead, we find that the data for the top 26 chords are exactly the same as $\mathbb{M}_{\mathbf{s}}$ with the exception of '`A:min7`' and '`E:7`' which have their positions reversed. We note that pnsets can have more than one equivalent chord symbol however, and on

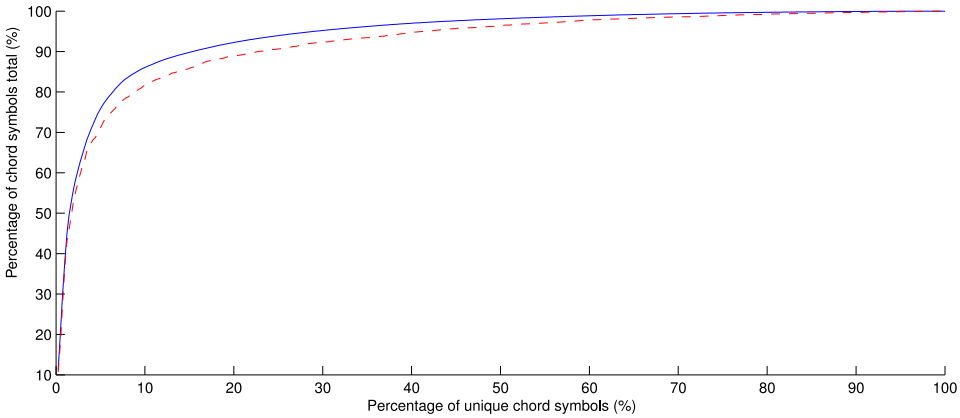


Figure 6.7: The percentages of chord count frequency (red dashed line) and duration (blue solid line) for the whole collection plotted against the percentage of unique symbols accounting for those values.

closer inspection we find that this change in the list order is because the chord '`C/6`' has the same ordered pnset as '`A:min7`' and is thus subsumed into the '`A:min7`' category.

To help visualise the values in table 6.5, pie charts for the symbol frequency and collected symbol durations are shown in Figure 6.8. What we find is that a small number of chord symbols make up quite a large proportion of the whole collection. In fact, in terms of symbol frequency, almost 50% of the chord symbols in the collection comprise just seven chords[10]: '`A:maj`', '`E:maj`', '`G:maj`', '`D:maj`', '`C:maj`', '`N`' and '`B:maj`'. For chord duration, the situation is similar with almost 50% of the time in the collection accounted for by the first six of those symbols. In other words, in both cases, roughly half the collection is accounted for by less than 2% of the total of 406 unique symbols. Likewise, ten symbols (2.5%) account for 60% of the total; sixteen symbols (4%) account for 70% and the top twenty six chords (6%) account for over 80% of the total collection. Figure 6.7 shows this relationship graphically, plotting the curves of percentages of the total for both frequency and duration against the percentage of the unique chord symbols that accounts for that part of the total. We also note

[10]It should be noted that although the transcriptions use the single root pitchnames such as '`A`' to denote major chords, we use the equivalent form '`A:maj`' here to improve clarity in the tables and graphs.

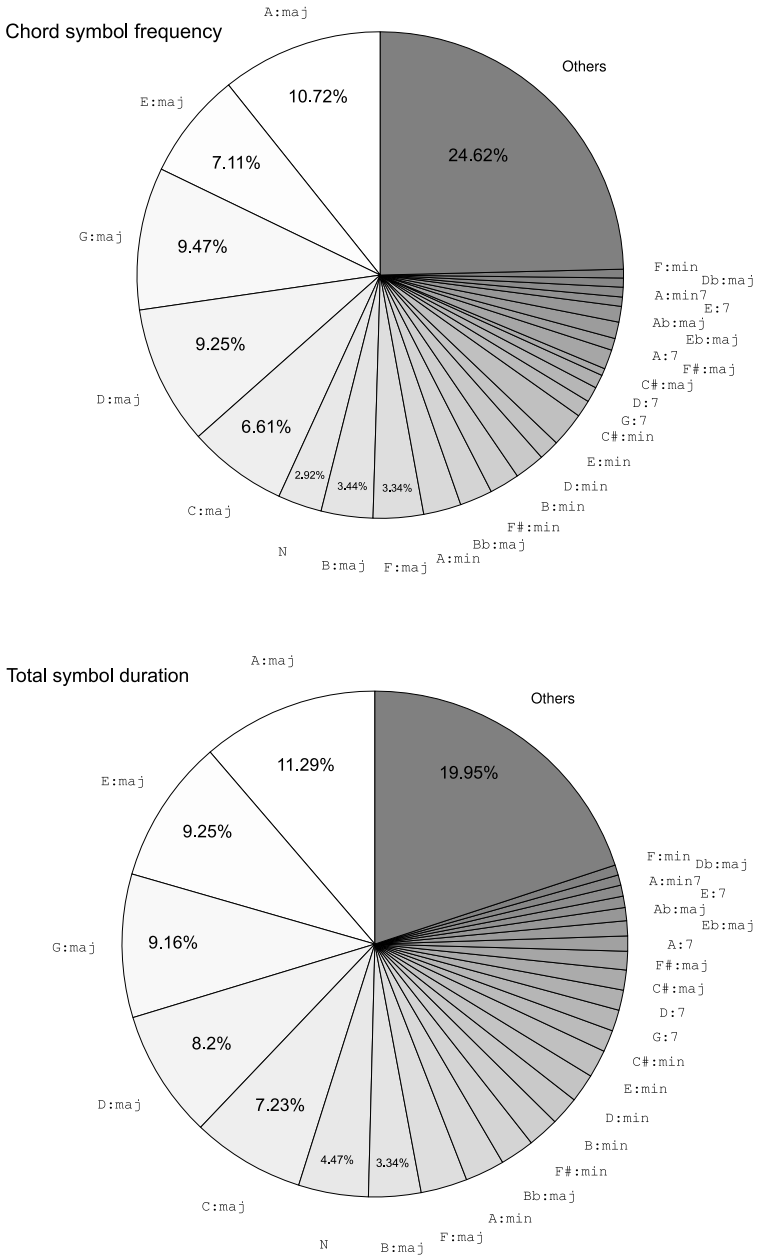


Figure 6.8: Pie charts showing chord symbol frequency (above) and allocation of time to chord symbols (below) for whole Beatles transcription collection (using direct string comparison). The top 80% (26 chord symbols) are represented individually with the remaining 380 lower frequency and shorter time chords in both cases aggregated into the category 'Others'. Segments in both pie charts are ordered according to total symbol duration. The numeric values represented in the two pie charts can be found in table 6.5.

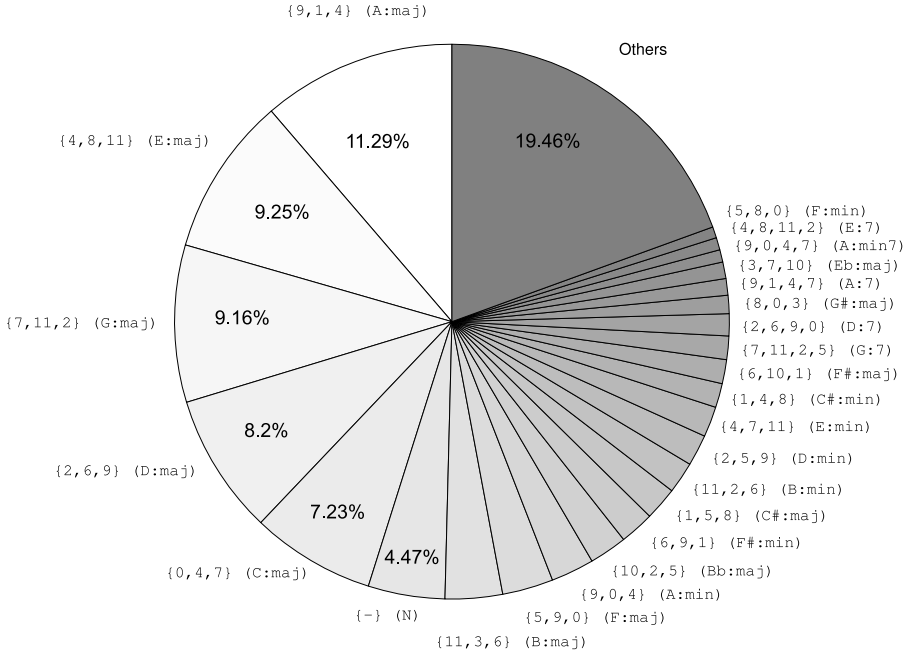


Figure 6.9: Pie chart showing allocation of time to unique ordered pcsets for the whole Beatles transcription collection. The top 80% (25 categories) are represented individually with the remaining 321 lower frequency and shorter time pcsets aggregated into the category 'Others'. Equivalent chord symbols are given next to the pcset values for clarity.

that the top five chords in the list are all easy to play in open fingerings on a guitar in standard tuning.

As we can see from the pie charts in figure 6.8, the distributions of symbol frequency and aggregate duration are quite similar to each other. This is the case for all of the counting methods that we have used. Therefore we will concentrate on the duration statistics for our further analysis because this is a more important factor than symbol frequency in evaluation of chord recognition algorithms when calculating chord symbol recall (discussed in section 8.1). It should be noted that pie charts in figures 6.8 to 6.11 have been plotted so that the categories that make up the top 80% of the collection for each different counting method are shown individually.

Figure 6.9 shows a pie chart for the total durations of unique ordered pcsets ($\mathbb{M}_{\overrightarrow{\mathbf{p}}}$) allowing quick comparision with the statistics for the string matching and ordered pnset comparison methods in figure 6.8. We see that the results for $\mathbb{M}_{\overrightarrow{\mathbf{p}}}$ are quite similar to those for $\mathbb{M}_{\mathbf{s}}$ durations for the

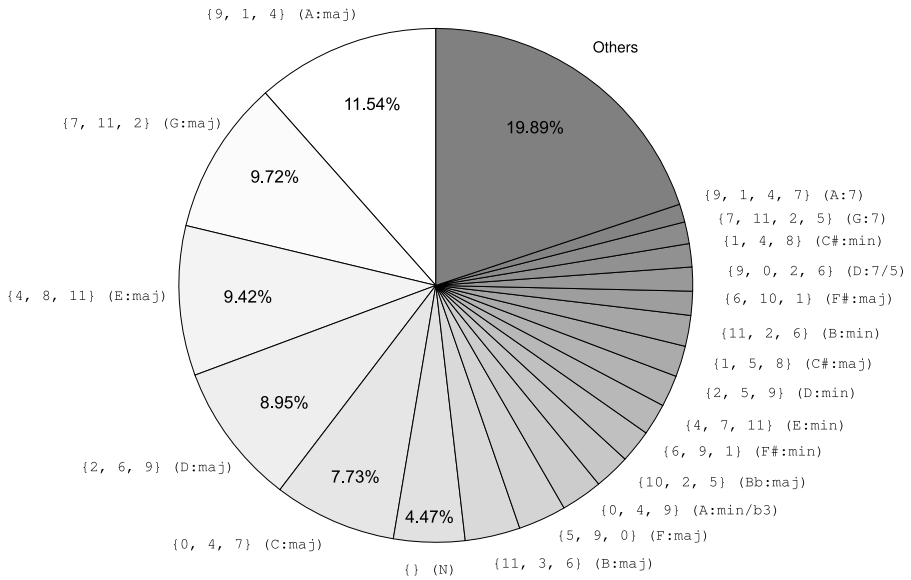


Figure 6.10: Pie chart showing allocation of time to unique unordered pcsets for whole Beatles transcription collection. The top 80% (20 categories) are represented individually with the remaining 182 lower frequency and shorter time pcsets aggregated into the category 'Others'. Equivalent chord symbols are given next to the pcset values for clarity.

top ranking chords; the top eleven chords remaining unchanged. '`C#:maj`' displaces '`B:min`' in twelth position because using an ordered pcset count, '`Db:maj`' and '`C#:maj`' are mapped to the same category.

Looking at the statistics for unordered pcsets (figure 6.10) we find that the percentages grow slightly for most of the top categories and also note that the order of the top three chords changes with '`G:maj`' moving up from third to second place. The reason for these changes is that by ignoring the order of chord elements, inversions of chords are no longer treated as separate categories. The durations for '`G:maj`' and '`E:maj`' are very similar for the $\mathbb{M}_{\mathbf{s}}$, $\mathbb{M}_{\overrightarrow{\mathbf{v}}}$ and $\mathbb{M}_{\overrightarrow{\mathbf{p}}}$ ordered counting methods (their durations differing by only 0.09%). However, using $\mathbb{M}_{\widetilde{\mathbf{p}}}$ unordered pcset counting, the three inversions of a G major triad i.e. root position (non-inverted) '`G:maj`', first-inversion '`G:maj/3`' and second-inversion '`G:maj/5`' are all combined in the {7,11,2} pcset category. The number of first and second '`G:maj`' inversions is 118 symbols which have a combined duration of 164.23 seconds (0.56% of the total). This is more than the number of first

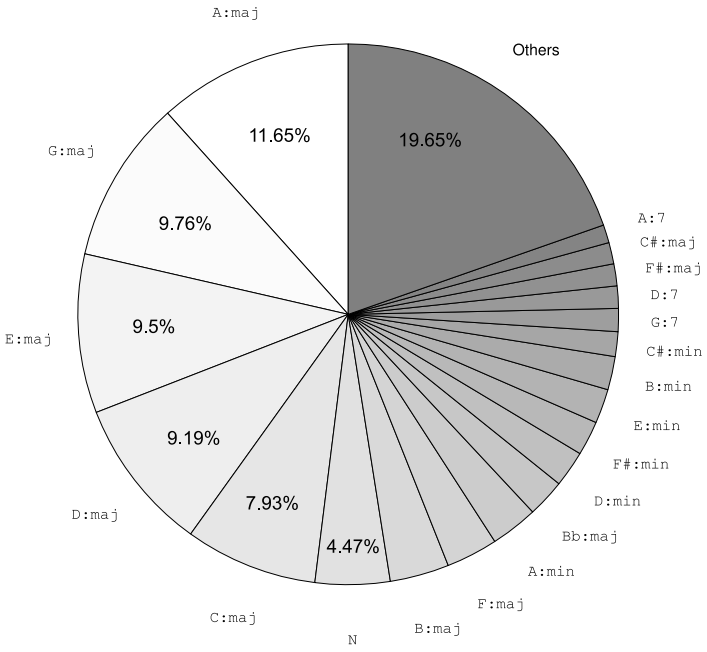


Figure 6.11: Pie chart showing allocation of time for bass-blind string comparison for whole Beatles transcription collection. The top 20 chord categories are represented individually with the remaining 226 lower frequency and shorter time chords in both cases aggregated into the category 'Others'.

and second inversions of '`E:maj`' which total 41 symbols with combined duration of 48 seconds (0.17% of the total). These values make a sufficient difference to cause the change in the list order.

Using bass-blind string comparison, $\mathbb{M}'_{\mathbf{s}}$ we find that the order of the chords in terms of duration again remains essentially the same as $\mathbb{M}_{\widetilde{\mathbf{p}}}$ for the highest categories (see pie chart in Figure 6.11). However, the top categories all grow slightly as they incorporate the symbols of the same type that previously had their own categories because of specified alternative bass intervals that were not part of the main chord interval list. This means that for $\mathbb{M}'_{\mathbf{s}}$ comparison, the top five chord symbols account for nearly 50% of the total collection duration. The distributions for the other bass-blind counts, $\mathbb{M}'_{\overrightarrow{\mathbf{v}}}$, $\mathbb{M}'_{\overrightarrow{\mathbf{p}}}$, $\mathbb{M}'_{\widetilde{\mathbf{v}}}$ and $\mathbb{M}'_{\widetilde{\mathbf{p}}}$ are all quite similar to $\mathbb{M}'_{\mathbf{s}}$ with very slight variations in the sizes of categories but no change to the order of the top fourteen chords.

Chordtypes

We now look at the numbers of unique chordtypes in the collection. The chordtype counts for each album and the collection totals are shown in table 6.6. The left side for the table shows chordtype counts using string matching $\mathbb{M}_{\mathbf{q}}$, ordered rlsets and rcsets $\mathbb{M}_{\overrightarrow{\mathbf{i}}}$ and $\mathbb{M}_{\overrightarrow{\mathbf{r}}}$ and unordered rlsets and rcsets $\mathbb{M}_{\widetilde{\mathbf{i}}}$ and $\mathbb{M}_{\widetilde{\mathbf{r}}}$. The right side of the table shows counts for the bass-blind versions of these comparison methods. The values from table 6.6 are shown graphically in the bar charts in figure 6.12; the upper chart showing the values from the left side of the table and the lower chart showing values from the bass-blind counts from the right side of the table.

Using string comparison $\mathbb{M}_{\mathbf{q}}$, we find that there are 133 individual chordtypes. We also find that both ordered rlset comparison $\mathbb{M}_{\overrightarrow{\mathbf{i}}}$ and ordered rcset comparison $\mathbb{M}_{\overrightarrow{\mathbf{r}}}$ counts total 122 individual chordtypes. The difference between the count for $\mathbb{M}_{\mathbf{q}}$ and the counts for $\mathbb{M}_{\overrightarrow{\mathbf{i}}}$ and $\mathbb{M}_{\overrightarrow{\mathbf{r}}}$ can be explained by a number of chords that have equivalent chordtype spellings. For example, chordtypes '`X:min(*b3)`', '`X:(1,5)`' and '`X:maj(*3)`' are used in various different contexts in the collection but are all equivalent to the rlset {`"1","5"`} or rcset {0,7}.

From the bar charts in figure 6.12 we can see that the values for $\mathbb{M}_{\overrightarrow{\mathbf{i}}}$ and $\mathbb{M}_{\overrightarrow{\mathbf{r}}}$ are always the same as are the counts for $\mathbb{M}_{\widetilde{\mathbf{i}}}$ and $\mathbb{M}_{\widetilde{\mathbf{r}}}$. This tells us that there are no chord symbols in the collection that are alternative enharmonic interval spellings of the same rcset (see section 5.3.3).

Following the same trend as seen in figure 6.6 with the unique chord symbols, the later albums tend to have higher numbers of unique chordtypes (with the exception of 'Let It Be').

Comparing the upper and lower charts in figure 6.12, we can see that including chord inversion has a large effect on the chordtype counts. Disc 10CD1 (the first disc of the white album) has the highest number of unique chordtypes for the ordered counting methods although it is very closely followed by discs 08 and 11 ('Sgt. Pepper's Lonely Hearts Club Band' and 'Abbey Road'). The order changes in the bass-blind counts with disc 08 having the highest number of unique chordtypes when bass intervals are ignored. We note that the counts for disc 10CD1 and 11 are both much

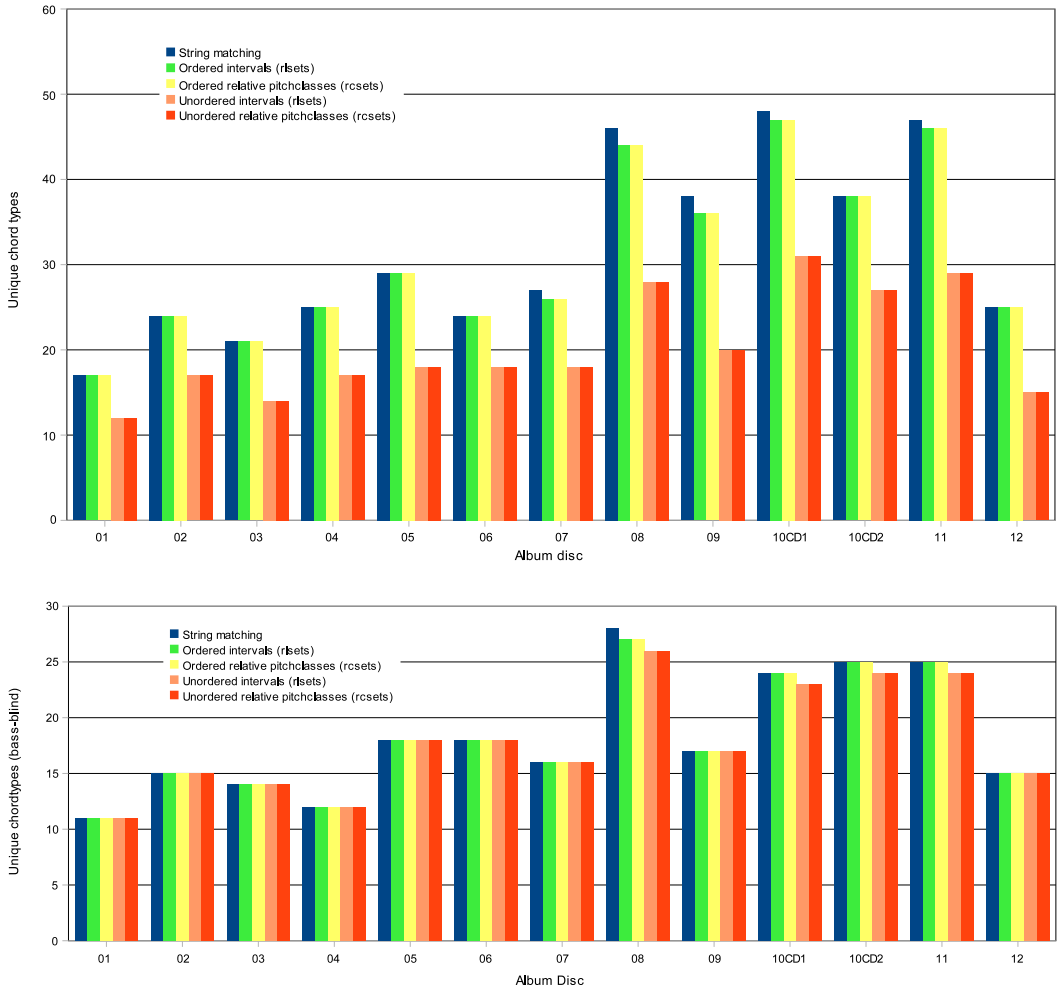


Figure 6.12: Bar charts showing unique chordtype counts for each album using five different counting methods. Upper chart shows counts including bass intervals; lower chart shows bass-blind counts.

Table 6.6: Chordtype count statistics for each album in the collection.

Disc	**Album**	$\mathbb{M}_{\mathbf{q}}$	$\mathbb{M}_{\overrightarrow{\mathbf{i}}}$	$\mathbb{M}_{\overrightarrow{\mathbf{r}}}$	$\mathbb{M}_{\widetilde{\mathbf{i}}}$	$\mathbb{M}_{\widetilde{\mathbf{r}}}$	$\mathbb{M}'_{\mathbf{q}}$	$\mathbb{M}'_{\overrightarrow{\mathbf{i}}}$	$\mathbb{M}'_{\overrightarrow{\mathbf{r}}}$	$\mathbb{M}'_{\widetilde{\mathbf{i}}}$	$\mathbb{M}'_{\widetilde{\mathbf{r}}}$
01	Please Please Me	17	17	17	12	12	11	11	11	11	11
02	With the Beatles	24	24	24	17	17	15	15	15	15	15
03	A Hard Day's Night	21	21	21	14	14	14	14	14	14	14
04	Beatles for Sale	25	25	25	17	17	12	12	12	12	12
05	Help!	29	29	29	18	18	18	18	18	18	18
06	Rubber Soul	24	24	24	18	18	18	18	18	18	18
07	Revolver	27	26	26	18	18	16	16	16	16	16
08	Sgt. Pepper's Lonely Hearts Club Band	46	44	44	28	28	28	27	27	26	26
09	Magical Mystery Tour	38	36	36	20	20	17	17	17	17	17
10CD1	The Beatles	48	47	47	31	31	24	24	24	23	23
10CD2	The Beatles	38	38	38	27	27	25	25	25	24	24
11	Abbey Road	47	46	46	29	29	25	25	25	24	24
12	Let It Be	25	25	25	15	15	15	15	15	15	15
All	Whole collection	133	122	122	63	62	66	62	62	55	55

Key:

$\mathbb{M}_{\mathbf{q}}$	String chordtype matching	$\mathbb{M}'_{\mathbf{q}}$	Bass-blind string chordtype matching
$\mathbb{M}_{\overrightarrow{\mathbf{i}}}$	Ordered rlset matching	$\mathbb{M}'_{\overrightarrow{\mathbf{i}}}$	Bass-blind ordered rlset matching
$\mathbb{M}_{\overrightarrow{\mathbf{r}}}$	Ordered rcset matching	$\mathbb{M}'_{\overrightarrow{\mathbf{r}}}$	Bass-blind ordered rcset matching
$\mathbb{M}_{\widetilde{\mathbf{i}}}$	Unordered rlset matching	$\mathbb{M}'_{\widetilde{\mathbf{i}}}$	Bass-blind unordered rlset matching
$\mathbb{M}_{\widetilde{\mathbf{r}}}$	Unordered rcset matching	$\mathbb{M}'_{\widetilde{\mathbf{r}}}$	Bass-blind unordered rcset matching

Table 6.7: Statistics for the top 20 of the 133 unique chordtypes in the transcription collection (counted using string matching of chordtypes). The final category 'Others' accounts for the other 113 unique symbols. Chordtypes are listed in order of aggregate duration.

Chordtype	**Frequency**	**% frequency**	**Aggregate time**	**% time**
`X:maj`	8367	57.23	17304.75	58.99
`X:min`	2065	14.12	4089.62	13.94
`X:7`	787	5.38	1876.51	6.4
`N`	427	2.92	1312.32	4.47
`X:min7`	312	2.13	606.62	2.07
`X:maj/5`	382	2.61	529.2	1.8
`X:9`	121	0.83	287.5	0.98
`X:maj/3`	226	1.55	258.76	0.88
`X:maj6`	106	0.72	237.65	0.81
`X:maj7`	119	0.81	216.1	0.74
`X:aug`	101	0.69	172.42	0.59
`X:min/5`	147	1.01	167.43	0.57
`X:min(*b3)`	18	0.12	149.53	0.51
`X:sus4`	118	0.81	134.48	0.46
`X:7(#9)`	43	0.29	126.07	0.43
`X:min/b3`	88	0.6	114.02	0.39
`X:maj/9`	38	0.26	89.07	0.3
`X:dim`	61	0.42	88.5	0.3
`X:dim7`	42	0.29	83.03	0.28
`X:maj/b7`	57	0.39	79.84	0.27
Others	996	6.81	1410.05	4.81

higher than those for disc 10CD2 in the upper chart with disc 10CD1 being slightly higher than disc 11. However, in the lower chart discs 10CD2 and 11 have exactly the same counts and are both slightly higher than disc 10CD1.

Table 6.7 shows the frequency and duration information for the top 20 chordtypes counted using string comparison $\mathbb{M}_{\mathbf{q}}$; these values are represented graphically by the pie charts in figure 6.13. Looking at the distribution of chordtypes we find that the top 20 chordtype categories account for 93% of the collection in terms of symbol frequency and over 95% of the duration for the whole collection. The root position major triad chordtype '`X:maj`' accounts for over half the collection in both terms of frequency and

duration. The root position minor triad '`X:min`' is second accounting for about 14% and the root position seven chord '`X:7`' is third at around 6% of the total. We note that the first and second inversions of both the major and minor triads are also found in the top 20 categories. The non-chord '`N`' is the fourth place in the list because many of the songs have silence at the start and at the end of the audio tracks.

When counted using bass-blind string comparison $\mathbb{M}'_{\mathbf{q}}$, the top categories grow further as can be seen in the pie charts in figure 6.14 (the values for which can be found in table 6.8). With this counting method the major triad chordtype category (which now subsumes first and second inversions) accounts for over 60% of the collection, the minor triad category for around 16% and '`X:7`' still accounts for around 6%. The non-chord '`N`' is still in fourth place and because it cannot be inverted or altered with an alternative bass interval, it remains at 4.47% of the total duration as before. Whereas the top 20 chords were around 95% of the collection counting with $\mathbb{M}_{\mathbf{q}}$, when we count with bass-blind $\mathbb{M}'_{\mathbf{q}}$, the top 20 chords now account for over 98% of the collection.

In their ISMIR09 paper [OGF09c], Oudre et al present a bar graph repartitioning the durations in the transcription collection into the top three chordtypes ('`X:maj`','`X:min`' and '`X:7`') and aggregating the other types into a fourth 'others' category. Figure 6.15 shows this same information calculated here using bass-blind string comparison $\mathbb{M}'_{\mathbf{q}}$. In their paper, they make the assumption that the only chordtypes used in the collection are the 17 members of the chord shorthand list given in table 4.2. However, as we have already seen from the statistics in this section, this is not the case and there are in fact 133 different chordtypes in all when using $\mathbb{M}_{\mathbf{q}}$ as the comparison method for counting. This number reduces to 66 when bass-blind string matching $\mathbb{M}'_{\mathbf{q}}$ is used and we find that fifteen of the seventeen shorthand labels are in fact present in the top 20 chordtypes using this counting method. The other five chordtypes that are in the top twenty are '`N`', '`X:(1)`', '`X:7(#9)`', '`X:maj(9)`' and '`X:min(*b3)`'. The two remaining shorthand chordtypes '`X:sus2`' and '`X:maj9`' are lower in the $\mathbb{M}'_{\mathbf{q}}$ list at positions 28 and 36 respectively.

It is interesting to look at the statistics for a wider range of chord

Table 6.8: Statistics for the top 20 of the 66 unique chordtypes in the transcription collection (counted using bass-blind string matching of chordtypes). The final category 'Others' accounts for the other 46 unique symbols. Chordtypes are listed in order of aggregate duration.

Chordtype	**Frequency**	**% frequency**	**Aggregate time**	**% time**
`X:maj`	9246	63.24	18411.11	62.76
`X:min`	2464	16.85	4543.68	15.49
`X:7`	846	5.79	1945.33	6.63
`N`	427	2.92	1312.32	4.47
`X:min7`	342	2.34	650.87	2.22
`X:9`	122	0.83	289.62	0.99
`X:maj6`	143	0.98	276.93	0.94
`X:maj7`	153	1.05	271.1	0.92
`X:aug`	105	0.72	178.25	0.61
`X:min(*b3)`	20	0.14	152.48	0.52
`X:sus4`	134	0.92	149.64	0.51
`X:7(#9)`	43	0.29	126.07	0.43
`X:maj(9)`	50	0.34	122.65	0.42
`X:dim`	78	0.53	109.9	0.37
`X:dim7`	55	0.38	101.55	0.35
`X:min6`	44	0.3	73.55	0.25
`X:(1)`	65	0.44	67.36	0.23
`X:hdim7`	41	0.28	64.26	0.22
`X:minmaj7`	30	0.21	47.95	0.16
`X:min9`	18	0.12	31.87	0.11
Others	195	1.33	406.95	1.39

symbols in the same visual representation that Oudre et al use. Figure 6.16 shows the distribution of the top twelve chordtypes for each album along with a final 'Others' category which reveals some details that are hidden in Figure 6.15's 'others' category. For example, in Figure 6.15, the second disc of the white album seems to have a very high number of 'Other' chords. When we look at the second graph we can see that almost all of the duration of 'other' chords for that disc is in fact in the '`N`' category which can be accounted for in most part by the non-tonal material in *Revolution 9* which totals 315.45 seconds.

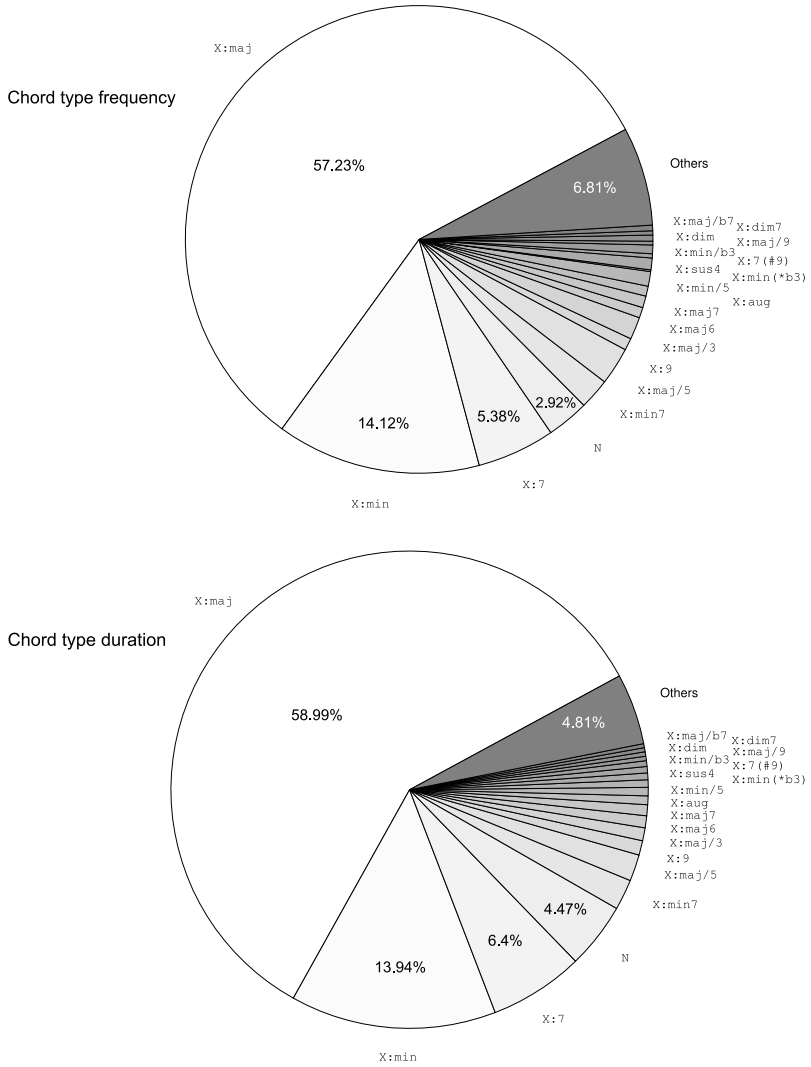


Figure 6.13: Pie charts showing chordtype frequency (above) and allocation of time to chordtypes (below) for whole Beatles transcription collection (string matching). The top 20 chord categories are represented individually with the remaining

6.6.3 Chord roots

Using a bass-blind pnset comparison with cardinality limited to 1, $\mathbb{M}'_{\overrightarrow{\mathbf{v}},1}$, we can find the distribution of enharmonically spelled chord roots (pitchnames) in the collection. Table 6.9 shows the values for the distribution of root pitchnames (excluding '`N`') and figure 6.17a shows these values graphically. There are sixteen root pitchname categories in the table. The seven

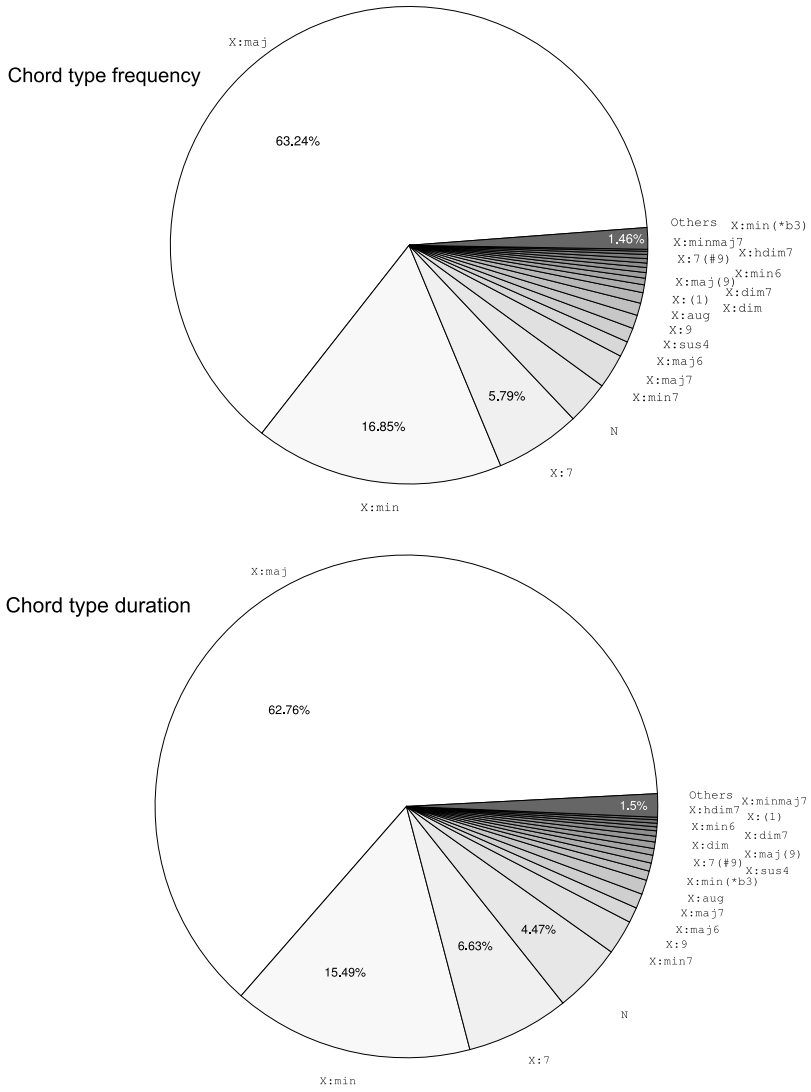


Figure 6.14: Pie charts showing chordtype frequency (above) and allocation of time to chordtypes (below) for whole Beatles transcription collection (inversion-blind string matching). The top 20 chord categories are represented individually with the remaining 47 lower frequency and shorter time chords in both cases aggregated into the category 'Others'.

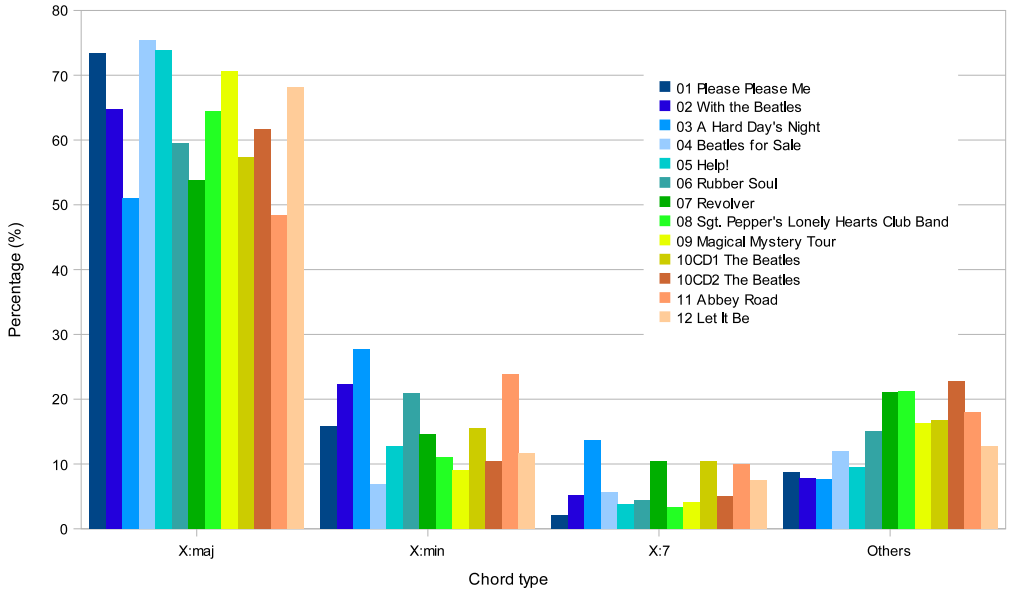


Figure 6.15: Bar chart showing distribution of time for the top three chordtypes in the transcriptions for each album with the remaining 63 types aggregated into category 'Others'.

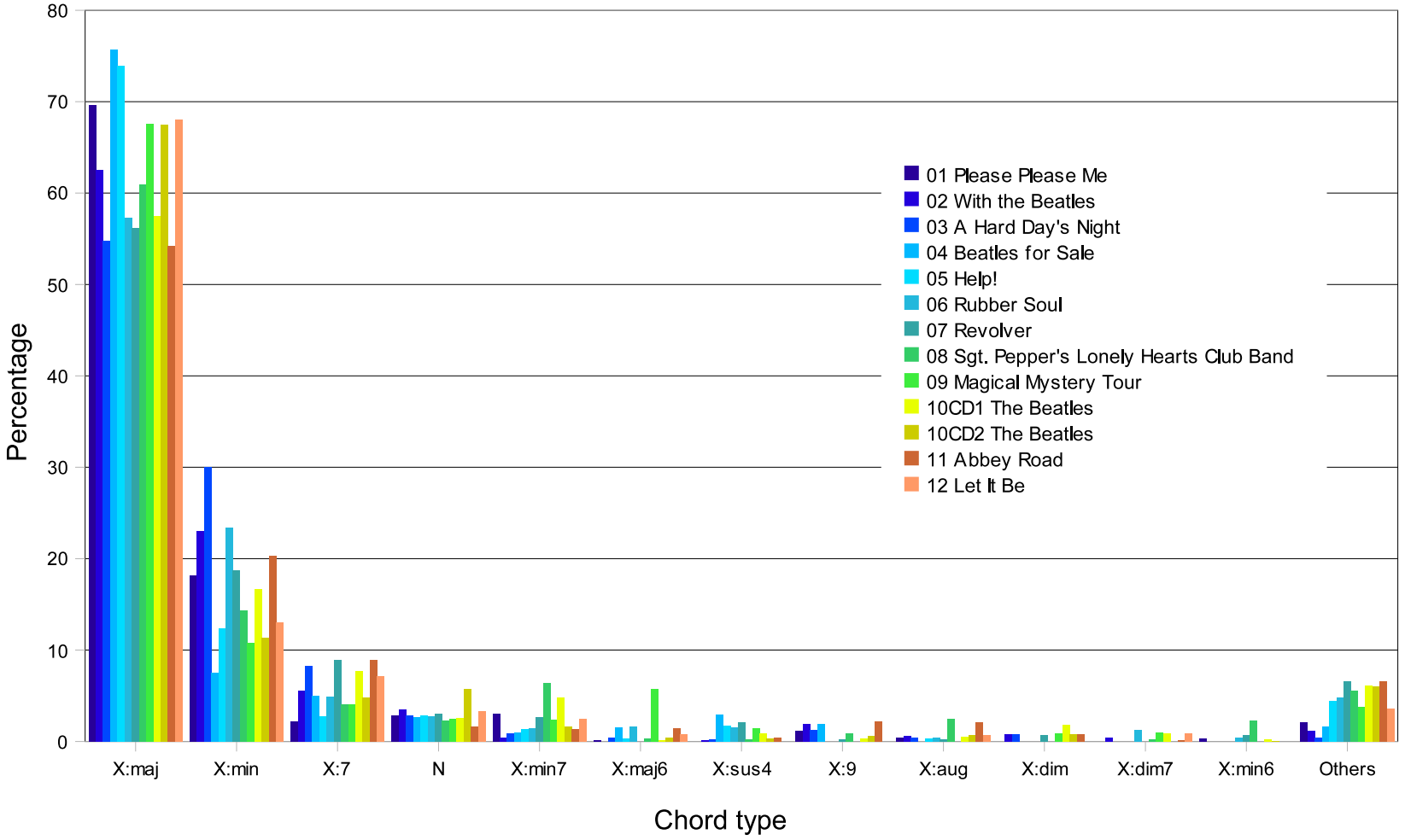


Figure 6.16: Bar chart showing distribution of time for the top twelve chordtypes in the transcriptions for each album with the remaining 54 types aggregated into category 'Others'.

Table 6.9: Pitchname distribution for chord roots for whole collection (excludes 'N' chords).

Pitchname	Frequency	% frequency	Aggregate time	% time
A	2494	17.57	5082.73	18.14
D	2254	15.88	4154.04	14.82
E	1817	12.8	4027.53	14.37
G	2030	14.3	3824.68	13.65
C	1391	9.8	2988.56	10.67
B	1091	7.69	1909.79	6.82
F	866	6.1	1594.71	5.69
F#	673	4.74	1232.99	4.4
Bb	469	3.3	936.7	3.34
C#	315	2.22	876.22	3.13
Eb	249	1.75	466.85	1.67
Ab	213	1.5	360.48	1.29
G#	151	1.06	209.79	0.75
Db	98	0.69	208.35	0.74
D#	43	0.3	80.34	0.29
Gb	40	0.28	67.38	0.24

natural pitchnames are present and are the top seven categories in the table. They are also joined by sharps F♯, C♯, G♯ and D♯ and flats B♭, E♭, A♭, D♭ and G♭.

We can also use the same approach to look at the distribution of chord root pitchclasses by using a bass-blind cardinality-1 ordered pitch class comparison $\mathbb{M}'_{\overrightarrow{\mathbf{p}},1}$. The values for this count are given in table 6.10 and these are shown graphically in figure 6.17b. Enharmonic equivalents that were separate in table 6.9 are now merged so we are left with just the 12 pitchclass categories.

6.6.4 Triad content

Using bass-blind cardinality-3 pcset comparison $\mathbb{M}'_{\overrightarrow{\mathbf{p}},3}$ for counting chord symbols allows us to map all chords into equivalent triad categories. Many current chord recognition algorithms [PP08, Bel07, WDR09, WEJ09, RUS+09, RK08] are designed to only detect major and minor triads so counting with $\mathbb{M}'_{\overrightarrow{\mathbf{p}},3}$ gives us useful information about the collection with respect to their

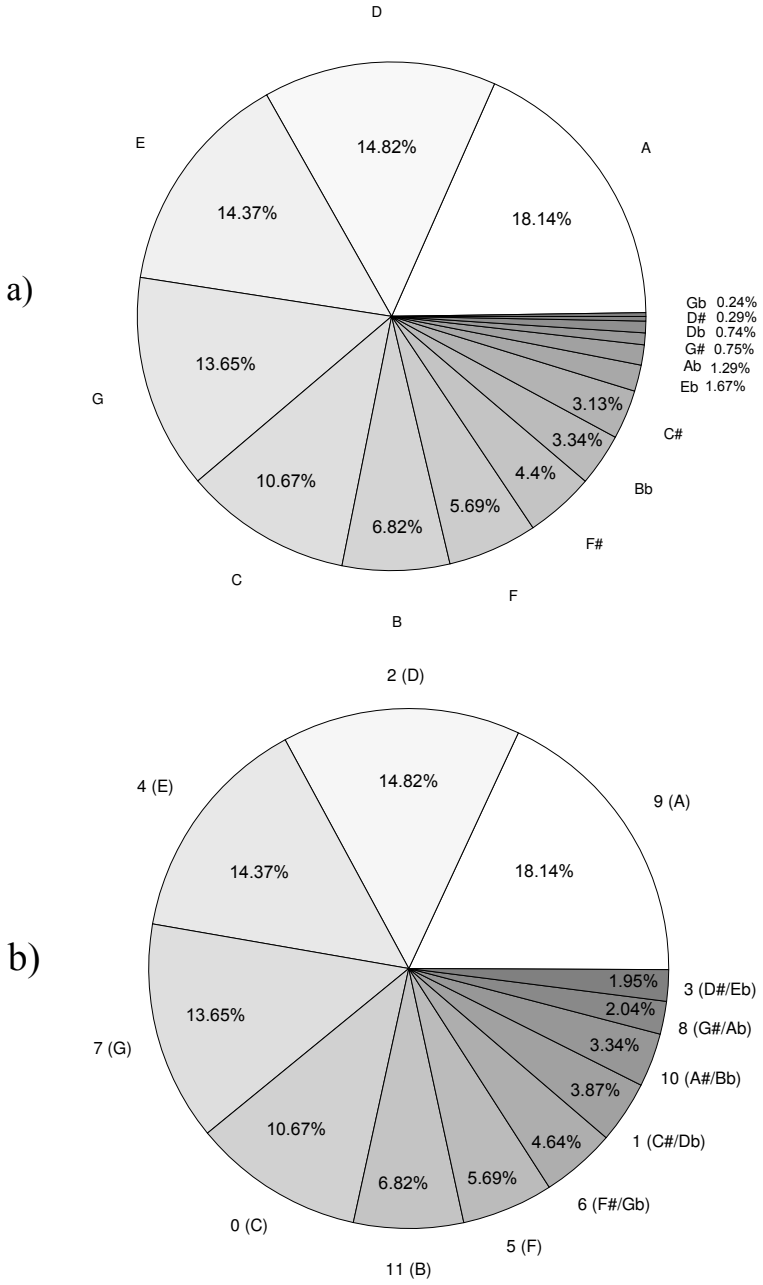


Figure 6.17: Pie charts showing distribution of durations (excluding '`N`' chords) for a) chord root pitchnames and b) chord root pitchclasses .

evaluation.

Table 6.11 shows the frequency and time statistics for the $\mathbb{M}'_{\overrightarrow{\mathbf{p}},3}$ count. We can see that the 24 major and minor triads are all represented in the collection with '`A:maj`' being the top ranking triad at 13.13% of the

Table 6.10: Pitchclass distribution for chord roots for whole collection (excludes 'N' chords).

Pitchclass	Frequency	% frequency	Aggregate time	% time
9 (A)	2494	17.57	5082.73	18.14
2 (D)	2254	15.88	4154.04	14.82
4 (E)	1817	12.8	4027.53	14.37
7 (G)	2030	14.3	3824.68	13.65
0 (C)	1391	9.8	2988.56	10.67
11 (B)	1091	7.69	1909.79	6.82
5 (F)	866	6.1	1594.71	5.69
6 (F#/Gb)	713	5.02	1300.37	4.64
1 (C#/Db)	413	2.91	1084.57	3.87
10 (A#/Bb)	469	3.3	936.7	3.34
8 (G#/Ab)	364	2.56	570.27	2.04
3 (D#/Eb)	292	2.06	547.19	1.95

duration of the collection and the pcset equivalent to 'Eb:min'/'D#:min' being the lowest ranked of the major and minor triads at 0.25% of the collection total duration. With the exception of 'N' which has its own category, all other chords that do not fit into the major or minor triad categories collectively account for 3.89% of the total collection duration.

6.6.5 Chord times

Figure 6.19 shows a histogram of chord duration for the whole collection. The upper plot is the whole histogram and the lower plot is a blown up view of the part between 0 and 10 seconds where most of the chord symbols reside. The mean length of chords in the collection is 2.0062 seconds and the median is 1.6254 seconds; the standard deviation is 2.5628. A few very long outlier chord durations pull the mean upwards; table 6.12 shows details of the longest chords in the collection. The longest chord is a 'C#:maj' in *Within you, without you* on 'Sgt.Pepper's Lonely Heart's Club Band' which has a duration of 137.4 seconds. This occurs during a part of the song where the sitar plays a solo melody and the accompaniment play a drone with no implied changes in harmony during the section. It is interesting to note that the top ten longest chords are all on the later

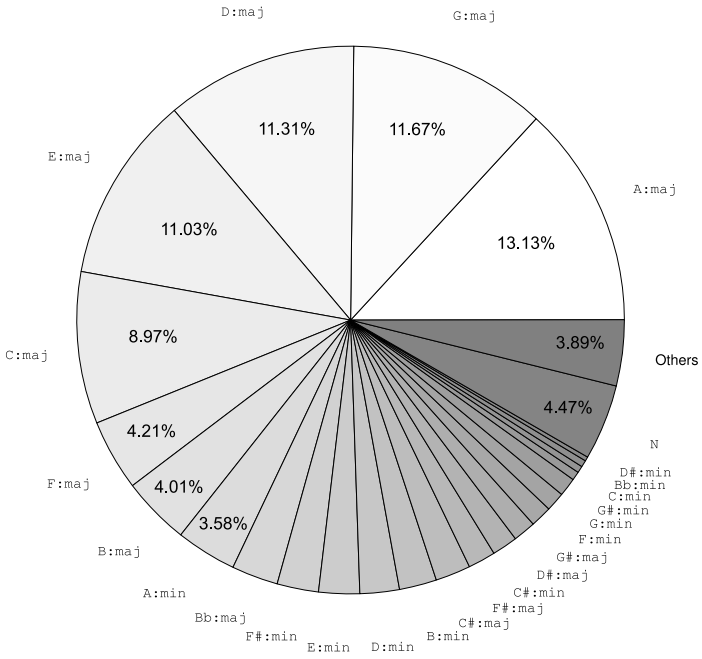


Figure 6.18: Pie chart showing distribution of equivalent triad chords in the collection counted using bass-blind cardinality-3 pcset comparison. The 24 major and minor triad categories are all represented along with the non-chord '`N`'. Other triad chords are aggregated into the 'Others' category.

albums from disc 07 (Revolver) onwards. The shortest chord symbol is the '`N`' chord at the start of *Piggies* on disc 2 of the white album which is 16.8 milliseconds long. This just describes a very short silence before the song actually starts so we may consider it to not really be part of the music as such. Discounting the non-chord '`N`' symbols at the start of songs, the shortest chord symbol in the collection is actually a 0.174 second '`C:maj`' at time 50.094 seconds in *Kansas City - Hey, Hey, Hey, Hey* from 'Beatles for Sale'.

Table 6.11: Equivalent distribution of triad chords in the collection counted using bass-blind cardinality-3 pcset comparison.

Pitchclass	Frequency	% frequency	Aggregate time	% time
A:maj	1868	12.78	3850.88	13.13
G:maj	1765	12.07	3422.54	11.67
D:maj	1838	12.57	3317.67	11.31
E:maj	1283	8.78	3235.06	11.03
C:maj	1257	8.6	2632.55	8.97
F:maj	671	4.59	1234.64	4.21
B:maj	653	4.47	1176.87	4.01
A:min	528	3.61	1050.9	3.58
Bb:maj	397	2.72	807.44	2.75
F#:min	368	2.52	717.87	2.45
E:min	474	3.24	706.14	2.41
D:min	328	2.24	679.14	2.32
B:min	369	2.52	648.96	2.21
C#:maj	188	1.29	610.86	2.08
F#:maj	256	1.75	475.39	1.62
C#:min	197	1.35	441.71	1.51
D#:maj	221	1.51	392.26	1.34
G#:maj	229	1.57	369.89	1.26
F:min	170	1.16	332.07	1.13
G:min	213	1.46	319.5	1.09
G#:min	104	0.71	150.51	0.51
C:min	57	0.39	118.66	0.4
Bb:min	63	0.43	114.69	0.39
D#:min	31	0.21	74.05	0.25
N	427	2.92	1312.32	4.47
Others	666	4.56	1140.89	3.89

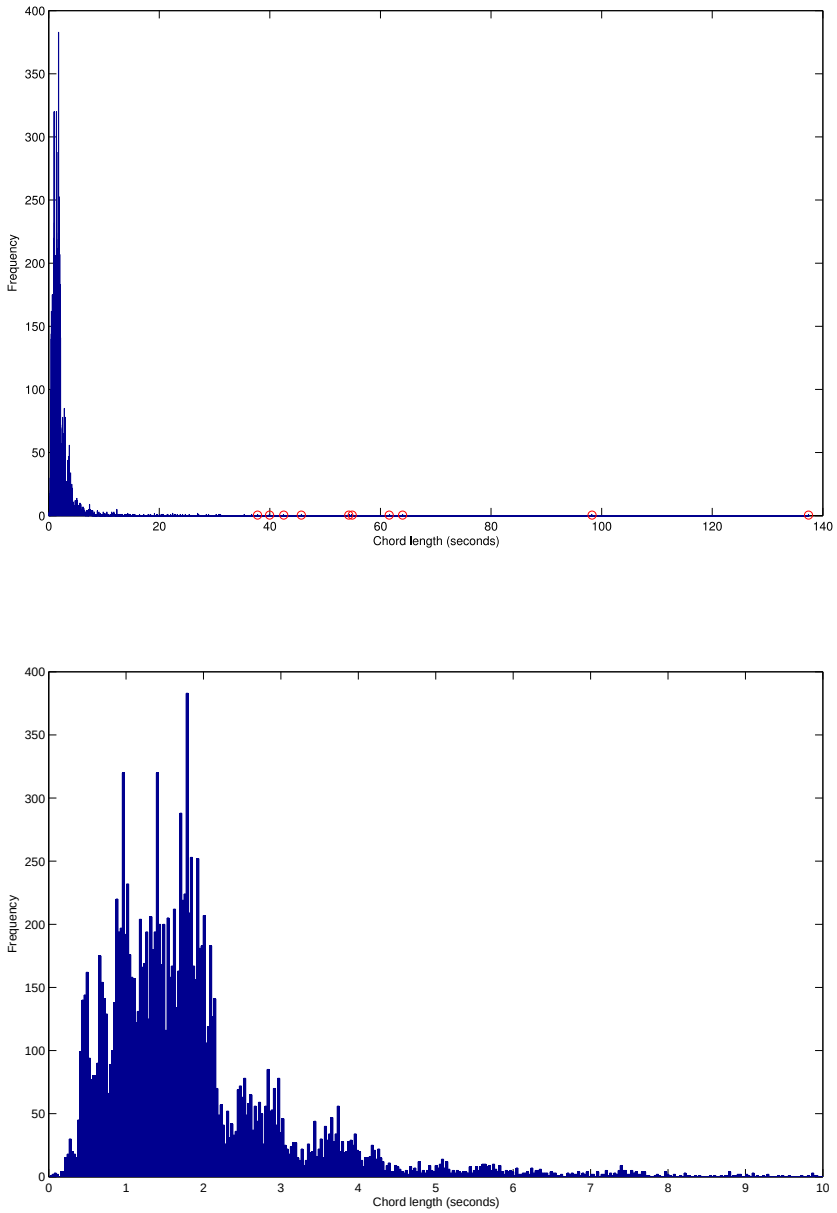


Figure 6.19: Histogram of chord times for the whole collection. The upper plot is the whole histogram including the outlier chords with very long durations (the top ten of which are marked with circles here and detailed in table 6.12). The lower plot shows the histogram between 0 and 10 seconds which is where the majority of chord times are.

Table 6.12: The top ten longest chords in the transcription collection

Album Disc	Song	Symbol	Length	Time
08	Within You Without You	`C#`	137.42s	136.06s
09	Blue Jay Way	`C`	98.23s	133.77s
12	Come Together	`D:min`	64.00s	192.27s
09	All You Need Is Love	`G`	61.64s	163.10s
10CD2	Revolution 9	`N`	54.34s	447.86s
08	A Day In The Life	`E`	42.47s	261.15s
09	Hello Goodbye	`C`	39.90s	165.63s
10CD2	Helter Skelter	`E`	37.80s	224.45s
13	Across the Universe	`Db`	35.33s	187.45s
07	Love You To	`C:min(*b3)`	30.88s	4.96s

Chapter 7

Local audio file alignment

One big problem for researchers in the MIR field is finding large annotated audio test sets which they can use as a ground truth to test their algorithms. Although several large annotation databases have been provided by various members of the community [HSAG05, GHNO02, BLEW04, Dow08] including the author's own Beatles transcriptions discussed in chapter 6, it is often still a problem for people to obtain exactly the same audio for use with this annotation data due to copyright and availability issues.

For many sets of annotation data, the times of events are given relative to the start of a particular audio file. The timing information in such annotations can only be considered to be accurate for the audio they were annotated from. However, it is not always possible to provide the actual audio files that the annotations refer to as well as the annotations themselves. Because of this, researchers often have to obtain the audio data from elsewhere making it difficult to be fully confident in the accuracy of the combination of their audio and someone else's annotations.

In this chapter we propose a solution to this problem. By using short audio fingerprints taken from the original annotated audio, it is possible to alter local audio files, aligning them accurately with the original audio. This technique provides a simple, legal way for researchers to acquire accurately aligned audio data for annotated data sets. This is a very important factor in obtaining accurate experimental results. As an example, after performing an alignment process on local copies of Beatles audio files, Ellis [Ell09] comments that a 20% increase in results was achieved for evaluations because the transcriptions were correctly aligned to the

audio material.

The system uses two short time-domain fingerprints for each song, the technical details of which are discussed in section 7.2.2. The first fingerprint is used to find the time difference between the start of the local audio and the start of the original song. This information allows the local song to be aligned accurately to the original by inserting or deleting samples at the start and end as necessary. The second fingerprint is used on the newly aligned local audio to check whether the process was successful.

The system has been implemented in Matlab and used the twelve Beatles studio albums as a test set. In experiments, we found that the alignment process is successful for all 180 songs in the collection as long as the local audio came from same mastering process as the original.

This work has many potential applications in MIR where authors of annotations containing timing information are unable to provide the original audio they used.

7.1 Access to audio for annotated data sets

The problem for researchers who provide annotated data containing time information is that such data requires the local copy of the audio file used by other researchers be accurately time-aligned with the original. This poses the question: how can other researchers obtain audio files that are properly aligned to the source audio files that were used to produce our transcription data?

Providing audio data directly

A simple answer would be to give other researchers the original source audio files that were used in the transcription process. The RWC database is an example of this model [GHNO02], providing copyright-free audio material and annotations for research work. This approach cannot be used for copyright audio such as the Beatles however, because it is illegal to copy and distribute such material.

Indirect access to copyright material

Another approach to the problem is to keep the legally owned copyright material on a secure server and allow indirect access through software tools and web services. Researchers can do experiments on the material via these services instead of having to own local copies of the material themselves. An example of an initiative that will employ this approach is the Networked Environment for Music Analysis (NEMA) project [pi08]. Although an elegant and potentially very useful solution to the problem, this approach may not be appropriate in all situations. This model requires a guaranteed network connection to the servers and researchers might be limited in what they can do depending on the tools, services and content provided on the host systems.

Providing references for copyright material

It is possible to provide references to copyright material so that other researchers may purchase the same material legally themselves. In the case of the Beatles chord transcriptions, it is easy to provide the list of catalogue numbers for the CD releases of the twelve albums so that other researchers can acquire their own copy of exactly the same audio information. This does not completely solve the problem however; the audio still needs to be extracted from the CD somehow and the result of this process is not completely predictable from computer to computer. Unfortunately, even if two researchers both use exactly the same audio CD as the source, it is not guaranteed that the audio files produced by different CD ripping software applications (or even the same application running on different platforms) will be accurately time-aligned with each other. Comparing track times for the same Beatles CDs ripped on separate computers, the times were found to differ by as much as 1838 samples (41ms) which will cause an audible delay in terms of annotations of event onsets.

7.2 Aligning wave files using audio fingerprints

To solve the problem highlighted in section 7.1 we employ a simple audio fingerprinting technique to realign local copies of audio files ensuring sample-accurate alignment to the original audio used to produce annotations. Using this technique it is possible for other researchers to legally obtain perfectly aligned audio files for use with the annotated data sets while only a very small amount of meta-data (i.e. the fingerprint information) needs to be provided by the owner of the original audio files.

7.2.1 Audio fingerprinting

Audio fingerprinting is often used in content-based retrieval of digital music. In this application, a particular audio file can be found from a large database of files when given a small segment of that same audio as a search query [CBKH05, KAH+02, HK02, RHG08]. For this reason, fingerprints must be produced for whole libraries of audio and must be robust to amplitude variation, noise, time-stretching and other distortions. The techniques used for this kind of application usually involve time-frequency analysis in order produce small fingerprints describing the features of the audio data.

Unlike the usual applications of audio fingerprinting, there is no need to search a large database for a given fingerprint in order to solve our problem. We already know which file the fingerprint belongs to; we are simply interested in aligning the local file relative to the position of the fingerprint data in the original audio. For this reason, we need a fingerprinting technique that does not distort time information so that we may achieve sample-accurate alignment.

7.2.2 Fingerprint technique

The technique that we have developed uses the sign of the first backward difference to generate fingerprint data. This simple algorithm produces a 1 or a 0 for each sample of the audio signal depending on the sign of the difference between the current audio sample and the previous sample. For

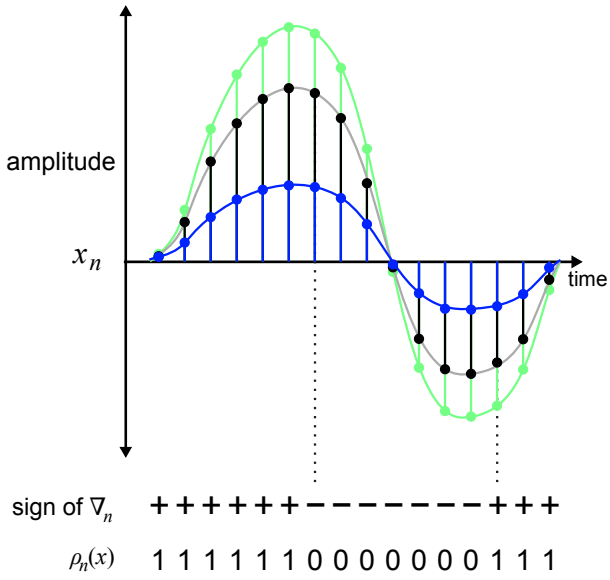


Figure 7.1: Signal x_n with corresponding sign of first backward difference ∇_n and fingerprint function $\rho_n(x)$. $\rho_n(x)$ is invariant to different amplitude scalings of x_n.

a discrete time signal x, the backwards difference ∇_n at time point n is given by:

$$\nabla_n = x_n - x_{n-1} \tag{7.1}$$

We may describe the sign of the first backward difference, function $\rho_n(x)$, mathematically in the following way:

$$\rho_n(x) = \begin{cases} 1 & \text{if} \quad \nabla_n \geq 0 \\ 0 & \text{if} \quad \nabla_n < 0 \end{cases} \tag{7.2}$$

Figure 7.1 shows that $\rho_n(x)$ is a binary function that tells us in which direction the amplitude of the signal is moving. We can see that while the signal rises in amplitude the function produces a string of ones and while it falls, it produces zeros. A transition from 1 to 0 or from 0 to 1 denotes a change in direction. No attempt is made to track the absolute signal amplitude so $\rho_n(x)$ is invariant to different amplitude scalings of the original signal.

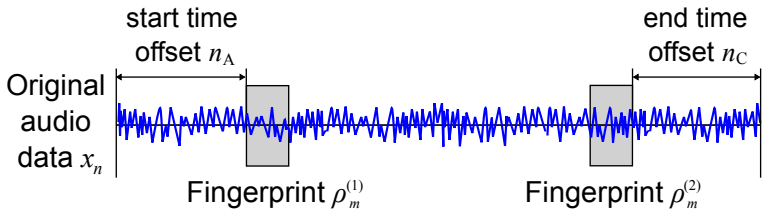


Figure 7.2: Fingerprint arrangement in original audio x_n

7.2.3 Alignment of local audio files using fingerprints

To align local audio files with the originals we use two fingerprints for each song. The first fingerprint $\rho_m^{(1)}$ is used to align the local audio then the second one $\rho_m^{(2)}$ is used to check the alignment. Figure 7.2 shows the arrangement of these two fingerprints in a song file.

If x_n is our original audio signal of length $\mathcal{N}$, we define the alignment fingerprint function $\rho_m^{(1)}$ for a particular song as:

$$\rho_m^{(1)} = \rho_{n_A+m}(x) \quad \text{for} \quad 0 \leq m < \mathcal{M} \tag{7.3}$$

where n_A is the index of x_n at which the alignment fingerprint starts (i.e. an offset from x_0) and $\mathcal{M}$ is the length of the fingerprint. We define a check fingerprint for the song, function $\rho_m^{(2)}$ as:

$$\rho_m^{(2)} = \rho_{\mathcal{N}-n_C-M+m}(x) \quad \text{for} \quad 0 \leq m < \mathcal{M} \tag{7.4}$$

where n_C is distance of fingerprint $\rho_m^{(2)}$ from the end of the song in samples (i.e. its offset from $x_{\mathcal{N}}$).

We may now define a fingerprint correlation function $C_n(x)$ to give a measure of how well a fingerprint matches segments of $\rho_n(x)$:

$$C_n(x) = \sum_m \left|\rho_{n+m}(x) - \rho_m^{(1)}\right| \tag{7.5}$$

for $0 \leq n < \mathcal{N} - \mathcal{M}$ and $0 \leq m < \mathcal{M}$. If $\mathcal{M}$ is set to be large enough to avoid false matches then we may assume that the following is true:

$$C_n(x) \begin{cases} = 0 & \text{if} \quad n = n_A \\ \neq 0 & \text{Otherwise} \end{cases} \tag{7.6}$$

That is to say where $C_n(x)$ evaluates to zero, the fingerprint exactly matches that section of $\rho_n(x)$ and this should only happen once across the original audio signal at the point where n is equal to n_A.

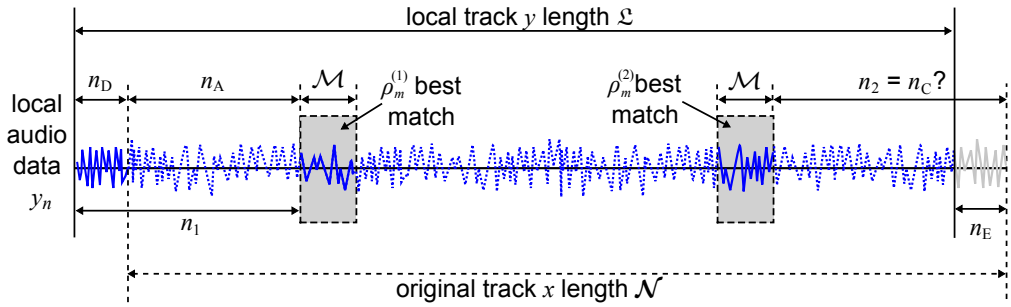


Figure 7.3: Aligning a local audio file using fingerprint $\rho_m^{(1)}$ for alignment and $\rho_m^{(2)}$ to check. In this example, n_D samples will be removed from the start of y_n and n_E samples will be appended at the end to align it with original signal x_n. After alignment using $\rho_m^{(1)}$, if $\rho_m^{(2)}$ matches such that $n_2 = n_\mathrm{C}$ then we know the process was successful.

Given this assumption, we may now use $\rho_m^{(1)}$ to align a local audio file. If y_n is a local audio signal of length $\mathfrak{L}$, we can find its fingerprint function $\rho(y_n)$ then calculate the correlation function:

$$C_n(y) = \sum_m \left|\rho(y_{n+m}) - \rho_m^{(1)}\right| \tag{7.7}$$

for $0 \leq n < \mathfrak{L} - \mathcal{M}$ and $0 \leq m < \mathcal{M}$. The minimum value of $C_n(y)$ marks the section of $\rho(y_n)$ that best matches the alignment fingerprint $\rho_m^{(1)}$. It should be noted that this value may be non-zero due to noise or distortion such as artefacts caused by compression. As long as local signal y_n is not too heavily distorted then the minimum value of $C_n(y)$ will still mark the best match. We will call the index of this value n_1 which can be expressed as:

$$n_1 = \arg\min(C_n(y)) \tag{7.8}$$

Once n_1 is determined, we must find the number of samples, n_D, that must be inserted or removed at the start of local audio y_n:

$$n_\mathrm{D} = n_\mathrm{A} - n_1 \tag{7.9}$$

To align the local audio with the original audio x_n we alter the start of y_n in the following way:

$$\text{if} \quad n_\mathrm{D} \begin{cases} > 0 & \text{insert } n_\mathrm{D} \text{ samples before } y_0 \\ = 0 & \text{do not need to alter } y_n \\ < 0 & \text{delete samples } y_0 \text{ to } y_{|n_\mathrm{D}|-1} \end{cases} \tag{7.10}$$

Figure 7.3 illustrates the relationship between n_{D}, n_{A} and n_1 at the start of a local audio signal y_n. In this example n_{D} samples will be removed at the start in order to align y_n with original signal x_n.

Once y_n has been aligned to the start of the original audio x_n, we must then ensure the local audio is the same length as x_n. This can be done by calculating a value n_{E}, the number of samples that must be appended to or removed from the end of y_n:

$$n_{\mathrm{E}} = \mathcal{N} - \mathfrak{L} + n_{\mathrm{D}} \tag{7.11}$$

The example in figure 7.3 shows that n_{E} samples will be appended to y_n in order to alter it to be length $\mathcal{N}$.

7.2.4 Checking the alignment

With the alignment complete, let $\hat{y}_n$ be the newly aligned version of y_n. We may now check that the alignment was successful by confirming that fingerprint $\rho_m^{(2)}$ matches $\hat{y}_n$ at the correct position relative to the end of the signal. To do this we calculate the correlation function $C_n(\hat{y})$ thus:

$$C_n(\hat{y}) = \sum_m \left| \rho_{\mathcal{N}-n+m}(\hat{y}) - \rho_m^{(2)} \right| \tag{7.12}$$

for $\mathcal{M} \leq n < \mathcal{N}$ and $0 \leq m < \mathcal{M}$. Then we find the best match for fingerprint $\rho_m^{(2)}$, n_2, from the correlation function:

$$n_2 = \arg\min(C_n(\hat{y})) \tag{7.13}$$

If n_2 and n_{C} are the same value (as is the case shown in figure 7.3) then the fingerprint matches at the same point as it would in the original audio signal x_n and we know that $\hat{y}_n$ is correctly aligned:

$$n_2 - n_{\mathrm{C}} \begin{cases} = 0 & \text{if } \hat{y}_n \text{ correctly aligned} \\ \neq 0 & \text{Otherwise} \end{cases} \tag{7.14}$$

If the alignment fails, the software notifies the user that it was unable to align the local audio file to the orginal.

7.3 Testing the fingerprinting technique

The fingerprinting technique described in section 7.2.2 has been implemented and tested in Matlab using the Beatles studio albums as a test set. Figure 7.4 shows correlation functions $C_n(x)$ and $C_n(y)$ generated for the first 10 seconds of the song 'I saw her standing there' from the Beatles first album 'Please Please Me'. The fingerprint $\rho_m^{(1)}$ and correlation function $C_n(x)$ were generated from the original wave file that was used for the Beatles chord transcriptions. The correlation function $C_n(y)$ was generated using the same fingerprint technique but the audio data was taken from another version of the song ripped on a separate computer. As figure 7.4a shows, the two functions are almost identical as we would expect from the same song. However, in figure 7.4b and 7.4c we can see that we can see that $C_n(x)$ matches $\rho_m^{(1)}$ at 200000 samples whereas $C_n(y)$ matches at 199300 samples (i.e. $n_{\mathrm{A}} = 200000$ and $n_1 = 199300$). This shows us that the two files are out of line by 700 samples (approximately 16ms) at the beginning so to align them we must insert that many extra samples at the start of the local audio file.

7.3.1 An alignment tool for the Beatles album collection

We have developed a set of Matlab scripts to automatically align local wave files given the correct fingerprint data. To evaluate the system we used the Beatles studio albums as a test set.

Fingerprint length and position

We experimented with different lengths of fingerprints. Figure 7.5 shows the number of alignment failures (i.e. the number of songs that failed the fingerprint $\rho_m^{(2)}$ check after alignment) against fingerprint length. We found that for prints more than 180 samples in length we could align all the 180 songs in the Beatles studio album collection successfully. For the publicly available alignment code we chose to use 200 samples (4.5ms at 44.1kHz sample rate) for the fingerprint length in order to provide a margin of error.

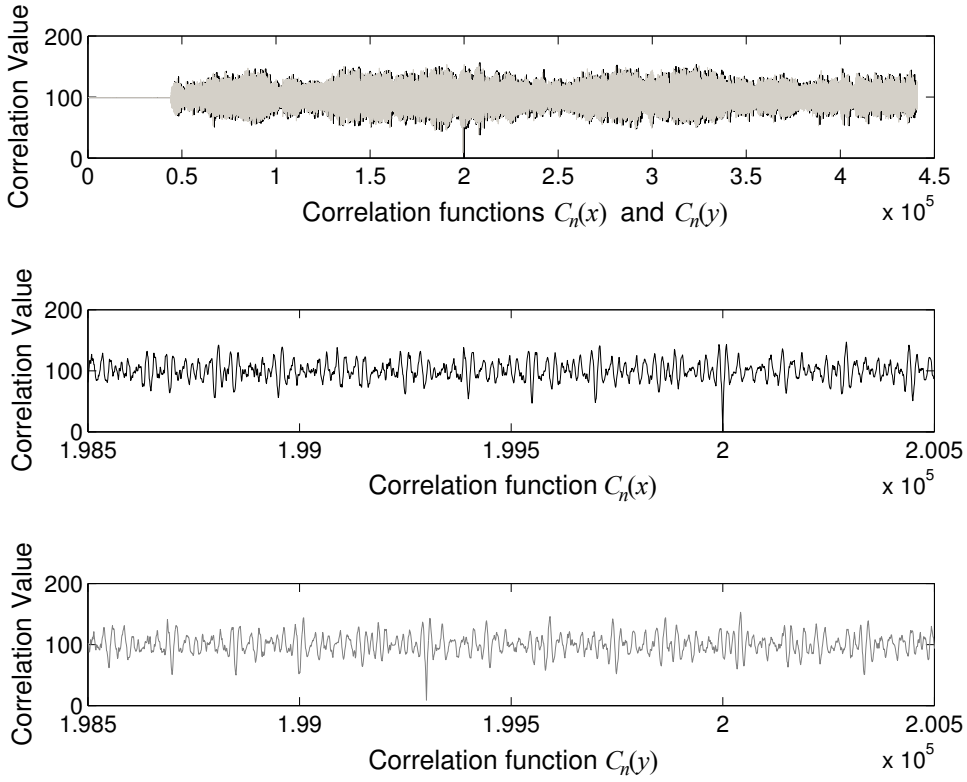


Figure 7.4: Top: fingerprint correlation functions $C_n(x)$ (black behind) and $C_n(y)$ (superimposed in light grey) for the first 10 seconds of a song; middle: close-up of $C_n(x)$ around n_A; Bottom: close-up of $C_n(y)$ near match position n_1.

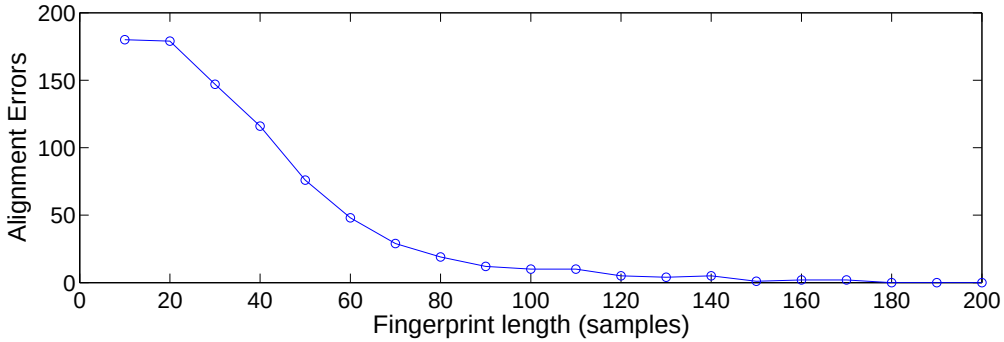


Figure 7.5: The number of alignment failures (from a total of 180 files) for different fingerprint lengths.

To make sure that there was sufficient room to accommodate any reasonable difference in start times between local and original audio files, the

length of the the start time offset n_{A} was set to 200000 samples (about 4.5 seconds into the song). The end time offset n_{C} was set to be 800000 samples (about 18 seconds) from the end of the song. This is a larger value than n_{A} because we found that for smaller values the check fingerprints were being generated from the noise or the silence at the end of songs where there were long fade outs or large gaps between tracks[1]. This made the alignment checking results unstable because the assumption from equation 7.6 does not hold for noise or long sections of silence, causing false positives in the fingerprint matching.

7.3.2 Fingerprint data files

We generated a fingerprint file for each song in the collection. The fingerprint information may easily be stored as a string of ones and zeros in a standard text file. The fingerprint files contain the following information:

- metadata about the song: song title and album title
- length of the original song file in samples ($\mathcal{N}$)
- start time alignment offset (n_{A})
- end time check offset (n_{C})
- the fingerprint data for fingerprints $\rho_m^{(1)}$ and $\rho_m^{(2)}$

Each fingerprint file is around 600 bytes long so the total size of all the files for the Beatles collection comes to 155kB and reduces to 82kB when zipped. An example of the text format for a fingerprint file is shown in table 7.1.

Padding extra samples

When extra samples need to be inserted into the local audio for alignment, we have to decide what sample values to insert. For a single wave file the only option is to pad any extra samples with zeros. For large collections

[1] In actual fact n_{C} could have been set at 400000 (9.1 seconds) except for one song, 'The End' from the album 'Abbey Road' which has a 17 second silence at the end.

Table 7.1: Example alignment fingerprint file

```
*** Beatles transcription alignment fingerprint file ***
Filename: 01_-_I_Saw_Her_Standing_There.wav
Album: 01_-_Please_Please_\ame
Samples: 7752960
Offset: 200000
Fingerprint:
00001110111000111111111000....00000011011100000111111100
Checkoffset: 800000
Checkprint:
01111111110000000001111110....00011110000011110000001111
```

such as the Beatles on the other hand, we have the option of taking extra samples required at the start of y_n from the end of the previous track on the album and extra samples at the end from the start of the next track. This technique means that songs which segue into each other (quite common on the later Beatles albums) will not contain big discontinuities in their wave files which would be caused by zero padding. This approach is obviously preferable to zero padding but is only safe if it can be guaranteed that the CD ripping process did not insert extra samples at the beginning or end of local audio tracks itself.

Practical implementation

Calculating the fingerprint correlation functions $C_n(y)$ and $C_n(\hat{y})$ is time consuming. To make the alignment and checking processes a little more efficient, we do not need to evaluate them over the whole wave file. For the alignment fingerprint $\rho_m^{(1)}$ we calculate $C_n(y)$ for the first 10 seconds of the song file. This is done because if $\rho_m^{(1)}$ cannot be found in the first 10 seconds of the file it is probably safe to assume that the local audio is not the same version of the song as the original. For checking the second fingerprint $\rho_m^{(2)}$, we already know exactly where it should be so we only need to evaluate $C_n(\hat{y})$ for a small window around sample $\hat{y}_{\mathcal{N}-n_C-M}$. If $\rho_m^{(2)}$ does not match at the expected position then we know the alignment has failed. With this implementation, all 180 songs can be aligned and

checked successfully in 20 minutes on our main test machine[2].

7.3.3 Testing

We tested the alignment system on several versions of the Beatles audio ripped from the original CDs on different computer systems. The alignment process completed successfully on all 180 songs for standard 44.1kHz wave files. We also tested the system on mp3 compressed versions of the Beatles files at the fourteen different bit rates supported by the LAME open source mp3 encoder [lam]. Although our fingerprinting technique works in the time domain, we found that the algorithm could successfully align the whole collection for all the bitrates tested, the lowest of which was 32kbps.

[2] A 3GHz Intel Core Duo PC with 4GB RAM running Ubuntu Linux 8.04.

Chapter 8

Chord recognition evaluation methods

With many researchers working in the chord recognition area it is necessary to compare results fairly with those of peers to clearly establish the state of the art. With so many different algorithms being developed in the community it is sometimes difficult to compare like with like in terms of the results they produce. The work reported in this chapter focuses on facilitating the rigorous evaluation of chord recognition algorithms.

In this chapter we examine the current evaluation techniques in use and propose a new approach to these techniques that will allow more general and fair comparisons to be drawn between algorithms.

Contribution

The main contributions we make in this chapter are:

- Formal treatment of chord recall using parameterised ordered set matching functions.
- Introduction of dictionary based evaluation.
- Development of a new chord sequence likeness measure.
- Proposal of an improved segmentation quality measure.
- Development of a transparent evaluation process separating recogniser frame rate from evaluation frame rate.

8.1 Chord symbol recall

The most obvious evaluation criterion for chord recognition is how often we can identify the correct chord. In much of the current research [PP08, LS08, RK08, VPM08, WDR09, OGF09c, KO09c, WEJ09, RUS+09], the most common performance metric that is used for this is what we shall call *chord symbol recall*, also known sometimes as the *average overlap score* [OGF09c] or *relative correct overlap* [Mau10]. This is a measure of what proportion of the time chords in the annotated ground truth chord sequence have been identified correctly in the machine estimated sequence. If both sequences comprise a number of finite length time segments, each segment corresponding to one chord symbol, the chord recall is given by

$$\mathcal{R} = \frac{|\text{estimated segments} \cap \text{annotated segments}|}{|\text{annotated segments}|} \tag{8.1}$$

where $|\cdot|$ represents the duration of a set of chord segments. That is, the summed duration of time periods where the correct chord has been identified, normalised by the total duration of the evaluation data. We may calculate the chord recall in two ways: one is to sample the chord sequences into a set of uniform length chord symbol *frames* and calculate the *frame-based chord symbol recall*; the other is to sum the durations of the continuous sections of estimated segments that correctly match the annotation and calculate the *segment-based chord symbol recall*.

8.1.1 Frame-based chord symbol recall

For a particular audio file split into a number of analysis frames, we may define the frame-based chord recognition recall $\mathcal{R}$ as:

$$\mathcal{R} = \frac{N_\text{C}}{N_\text{T}} \tag{8.2}$$

where N_T is the total number of analysis frames and N_C is the number of frames where the correct chord was identified. This measure is what Oudre [OGF09c] refers to as the 'average overlap score' and has been used by many researchers in the community including the author [OGF09c, Lee08, HS05] for chord symbol recall evaluation. It has been a popular

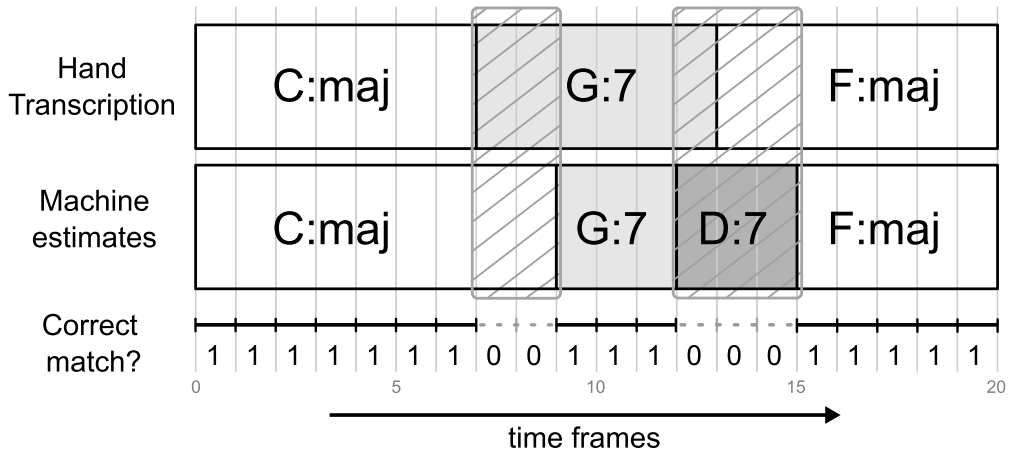


Figure 8.1: Calculating the frame-based recall measure for a set of analysis frames. There are twenty frames in total in this example, fifteen of which are correct matches so the recall score is $\mathcal{R} = \frac{15}{20} = 0.75$.

measure because most chord recognition algorithms from recent years have themselves been frame-based so it is common to sample the annotated ground truth data at the same frame rate as the recogniser to perform the recall evaluation.

A visual representation of the frame-based recall measure can be seen in figure 8.1 where fifteen out of twenty frames are correct matches giving a recall value of $\frac{15}{20} = 0.75$. The recall value provides an intuitive metric between 0 as the worst case and 1 as the best case for chord recognition.

Using a parameterised matching function for recall calculation

In chapter 5 we introduced a chord matching function $\mathbb{M}_T$ for comparing two chords $\mathtt{X}$ and $\mathtt{Y}$ which we recall from equation 5.25 takes the form

$$\mathbb{M}_T(\mathtt{X}, \mathtt{Y}) = \begin{cases} 1 & \text{if } \mathtt{X} \text{ matches } \mathtt{Y} \\ 0 & \text{otherwise} \end{cases}$$

where T denotes the comparison method used to evaluate the result of the matching function.

Using matching function $\mathbb{M}_T$, it therefore follows that to evaluate the frame based recall for a sequence of machine estimated chord symbols E compared to an annotated sequence A (where both sequences are N_{T}

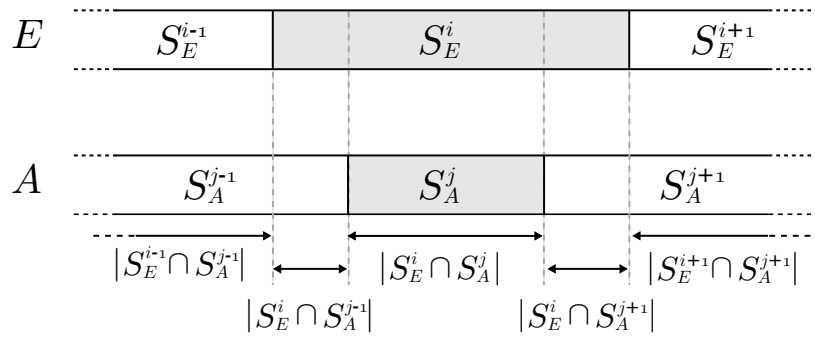


Figure 8.2: Segments of estimated sequence E compared with hand annotated sequence A.

frames in length), the number of correct matches N_{C} may be expressed as

$$N_{\mathrm{C}} = \sum_{n=0}^{N_{\mathrm{T}}-1} \mathbb{M}_T(A_n, E_n) \tag{8.3}$$

and we may therefore restate the recall equation for sequences A and E, parameterised by T as

$$\mathcal{R}_T(A, E) = \frac{\sum_{n=0}^{N_{\mathrm{T}}-1} \mathbb{M}_T(A_n, E_n)}{N_{\mathrm{T}}} \tag{8.4}$$

8.1.2 Segment-based chord symbol recall

The accuracy of the frame-based recall calculation is limited by its frame rate. As the frame rate increases, accuracy can be improved but the computation time required to perform the evaluation also increases because there are more frames to evaluate matches for. To solve this problem, we may calculate the chord symbol recall as a continuous time function by considering matches between overlapping segments in the two chord sequences instead of sampling them into discrete frames. In the limit, as frame length approaches zero, both calculations would produce the same result.

For this measure we consider hand annotated sequence A as a sequence of segments S_A and machine estimated sequence E similarly as segments S_E as shown in figure 8.2. Each segment is of finite length and contains one chord symbol. For any segment S_A^j in the annotated sequence we find all the corresponding estimate segments S_E^i for which $|S_E^i \cap S_A^j| \neq 0$ and

calculate the matching function $\mathbb{M}_T(S_E^i, S_A^j)$. The durations of overlapping segments which are correct matches are summed together to give τ_{C} which is normalised by the total duration τ_{T} thus

$$\mathcal{R}_T(S_E, S_A) = \frac{\tau_{\mathrm{C}}}{\tau_{\mathrm{T}}} = \frac{\sum_{S_A^j} \sum_{S_E^i} |S_E^i \cap S_A^j| . \mathbb{M}_T(S_E^i, S_A^j)}{\sum_{S_A^j} |S_A^j|} \tag{8.5}$$

This measure is equivalent to Mauch's relative correct overlap [Mau10].

8.1.3 Defining a correct match

The chord symbol recall metric is a good general way of measuring chord recognition performance but to use it we must be careful to define what is meant by a 'correct match' between a hand-transcribed chord and a machine-estimated one. How we choose to calculate $\mathbb{M}_T$ will have a large effect on the recall result.

In section 6.6 we saw that the Beatles chord transcription collection contains 14621 individual chord labels. Depending on the method used to count, those 14621 chord labels comprise 406 unique chord symbols which in turn comprise 66 unique chordtypes[1]. The largest number of chords that can be recognised by the algorithms from the MIREX09 competiton that are studied in this work is twelve chord types [RRHS09a] (see table 8.4 later in the chapter for a summary of the chord vocabularies of the MIREX09 algorithms studied in this work). The majority of current algorithms can recognise less than five and many still deal only with major and minor triads. This poses the question: how can we make a fair comparison between the machine estimated chords produced by our algorithms with those from the manually annotated transcription collection when the chord vocabularies of each differ so much?

Many current chord recognition algorithms produce triad chords for all of their estimates [PP08, Bel07, WDR09, WEJ09, RUS+09, RK08] with some exceptions [OGF09c, Mau10, RRHS09a] that also include seventh chords thus producing a mixture of triads and tetrads in their estimates. As discussed in section 6.6.1, the Beatles transcription collection contains

[1] These values are the counts for $\mathbb{M}_{\mathbf{s}}$ and $\mathbb{M}'_{\mathbf{q}}$ taken from tables 6.4 and 6.6 respectively.

chord symbols with cardinalities ranging from zero (the '`N`' chord) to six ('`A:9(11)`' in *Tell me why* from 'A Hard Day's Night'). Given that the cardinalities of machine-estimated chords are therefore likely to differ from that of the transcribed chords we wish to evaluate them against, we must find a fair way to compare them that takes this into account.

MIREX Chord detection evaluation

Many recent pieces of research [OGF09c, KO09c, WDR09] have adopted the evaluation method used for the MIREX chord detection task [mirb] so that they can compare their algorithm against the evaluation results which provide a good performance baseline. The MIREX08 evaluation used the Beatles transcription collection as its ground truth for evaluation and MIREX09 used the collection again plus additional annotated files including songs by Queen and Zweieck provided by Mauch et al [MCD+09].

MIREX08 dealt only with the recognition of major and minor chords and the evaluation system that year mapped all chord shorthands (defined in section 4.2.4 table 4.2) to either major or minor categories as follows:

major: `maj`, `dim`, `aug`, `maj7`, `7`, `dim7`, `hdim7`, `maj6`, `9`, `maj9`, `sus4`, `sus2`, others

minor: `min`, `min7`, `minmaj7`, `min6`, `min9`

Major family chords were mapped to the major triad and minor family chords mapped to the minor triad. All other chord qualities were also mapped to the major triad.

This mapping provides an easy way to reduce the evaluation problem to simply comparing major or minor triad categories so there is no need to worry about chord cardinality. However, there are some problems with approaching the evaluation in this way. Augmented, suspended and diminished chords ('`aug`', '`sus2`', '`sus4`', '`dim`', '`dim7`' and '`hdim7`') are mapped to the major category. In the case of the augmented triad this could be considered a reasonable mapping because its interval list `(1,3,#5)` contains a major third. However, the suspended chord types '`sus2`' and '`sus4`' do not contain a major or minor third interval and are therefore no more major than they are minor. To put them in one category or the other is

a compromise and this will unfairly favour recognition algorithms with a bias towards producing major labels when a suspended chord is present in the audio. Furthermore, the diminished chord types are arguably closer to minor than the major chord because they contain a minor third. Mapping them to major will also influence the evaluation results negatively as we might reasonably expect a major-minor chord recognition algorithm to return a minor result when a diminished chord is present in the audio.

For MIREX09, some of these issues were addressed with the mappings for the major-minor evaluation system[2] being altered to:

major: `maj`, `aug`, `maj7`, `7`, `maj6`, `9`, `maj9`, `sus4`, others
minor: `min`, `min7`, `minmaj7`, `min6`, `min9`, `dim`, `dim7`, `hdim7`, `sus2`

Again, any chord that does not fit into the seventeen shorthand categories is mapped to major by default. The diminished chords are now mapped more suitably to minor and the suspended second chord '`sus2`' has also been remapped to minor. The decision to move '`sus2`' to minor may be supported by the argument that its intervals `(1,2,5)` are closer to minor `(1,b3,5)` than major `(1,3,5)`. In doing this it evens out the problem of favouring biased algorithms to a certain extent because the suspended chords are now split between the two categories. However, the problem still remains that chords which are neither major or minor family get mapped to one or the other to enable evaluation.

We may express the MIREX chord evaluation algorithm formally in terms of matching function $\mathbb{M}_T$. Let $\mathbf{map}_{\mathrm{YY}}(\mathtt{X})$ be the MIREX chord mapping function for year YY that maps chord `X` to the major or minor category. Major category chords return 0 and minor category chords return 12 so the mapping functions for MIREX08 and MIREX09 are

$$\mathbf{map}_{08}(\mathtt{X}) = \begin{cases} 12 & \text{if } \mathtt{Q_X} \subseteq \{\mathtt{"min"}\} \\ 0 & \text{otherwise} \end{cases} \tag{8.6}$$

$$\mathbf{map}_{09}(\mathtt{X}) = \begin{cases} 12 & \text{if } \mathtt{Q_X} \subseteq \{\mathtt{"min"}|\mathtt{"sus2"}|\mathtt{"dim"}|\} \\ 0 & \text{otherwise} \end{cases} \tag{8.7}$$

[2]Many thanks to Mert Bay for providing the MIREX chord mapping script to check these details.

Table 8.1: Summary of ordered chord matching functions from chapter 5.

$\mathbb{M}_{\mathbf{s}}$	Chord symbol string matching	$\mathbb{M}_{\mathbf{q}}$	Chordtype string matching
$\mathbb{M}_{\overrightarrow{\mathbf{v}}}$	Ordered pnset matching	$\mathbb{M}_{\overrightarrow{\mathbf{i}}}$	Ordered rlset matching
$\mathbb{M}_{\overrightarrow{\mathbf{p}}}$	Ordered pcset matching	$\mathbb{M}_{\overrightarrow{\mathbf{r}}}$	Ordered rcset matching

Now let $\mathbf{mx}(\mathtt{X})$ be a function that transforms chord symbol `X` into an integer representation such that major family chords map to the values 0 to 11 (with respect to `C:maj` = 0), minor chords map to values 12 to 23 and the non-chord '`N`' mapping to 24. This may be expressed as

$$\mathbf{mx}_{\mathrm{YY}}(\mathtt{X}) = \begin{cases} \mathrm{p}_{\mathtt{R_X}} + \mathbf{map}_{\mathrm{YY}}(\mathtt{X}) & \text{if } \mathtt{X} \neq \mathtt{"N"} \\ 24 & \text{otherwise} \end{cases} \tag{8.8}$$

where $\mathrm{p}_{\mathtt{R_X}}$ is the pitchclass corresponding to root pitchname $\mathtt{R_X}$ (see equation 5.23). We may therefore state the MIREX08/09 matching function $\mathbb{M}_{\mathbf{mx}}$ as

$$\mathbb{M}_{\mathbf{mx}_{\mathrm{YY}}}(\mathtt{X}, \mathtt{Y}) = \begin{cases} 1 & \text{if } |\mathbf{mx}_{\mathrm{YY}}(\mathtt{X}) - \mathbf{mx}_{\mathrm{YY}}(\mathtt{Y})| = 0 \\ 0 & \text{otherwise} \end{cases} \tag{8.9}$$

and we can then calculate N_{C} and hence $\mathcal{R}_{\mathbf{mx}}$ using this function as described in equation 8.3.

Using an ordered set matching function for recall calculation

To solve the problems inherent in the chord mapping approach of recall evaluation algorithms like those used for MIREX08/09, we propose the use of ordered set matching functions as described in chapter 5 (a summary of these can be seen in table 8.1). As discussed before, there are multiple ways of expressing the same chord using the chord syntax described in chapter 4, so the direct string matching function $\mathbb{M}_{\mathbf{s}}$ is not suitable for use in the recall evaluation. However, the pnset and pcset matching functions, $\mathbb{M}_{\overrightarrow{\mathbf{v}}}$ and $\mathbb{M}_{\overrightarrow{\mathbf{p}}}$, do not suffer from the same problem and are therefore well suited to this task.

Table 8.2: Comparison of different chord matching functions including results for MIREX08/09 chord mapping, ordered pnset and pcset matching functions and cardinality limited pcset matching functions.

Example	chord `X`	chord `Y`	$\mathbb{M}_{\mathbf{mx}_{08}}$	$\mathbb{M}_{\mathbf{mx}_{09}}$	$\mathbb{M}_{\overrightarrow{\mathbf{v}}}$	$\mathbb{M}_{\overrightarrow{\mathbf{p}}}$	$\mathbb{M}_{\overrightarrow{\mathbf{p}},3}$	$\mathbb{M}_{\overrightarrow{\mathbf{p}},2}$
1	`C:maj`	`C:maj`	1	1	1	1	1	1
2	`C:min`	`C:min`	1	1	1	1	1	1
3	`C:dim`	`C:dim`	1	1	1	1	1	1
4	`C:aug`	`C:aug`	1	1	1	1	1	1
5	`C:maj`	`C:min`	0	0	0	0	0	0
6	`C#:maj`	`Db:maj`	1	1	0	1	1	1
7	`C:dim7`	`C:(1,#2,#4,6)`	1	0	0	1	1	1
8	`C:maj`	`C:(1,#2,#4,6)`	1	1	0	0	0	0
9	`C:maj`	`C:(1,b3,5)`	1	1	0	0	0	0
10	`C:min`	`C:(1,b3,5)`	0	0	1	1	1	1
11	`C:maj`	`C:maj7`	1	1	0	0	1	1
12	`C:dim`	`C:sus2`	1	1	0	0	0	0
13	`C:sus2`	`C:sus4`	1	0	0	0	0	0
14	`C:min`	`C:dim`	0	1	0	0	0	1
15	`C:min`	`C:aug`	1	0	0	0	0	0

Table 8.2 shows the results of several different matching functions on various pairs of chords. We will initially look at the first four columns which contain values for the MIREX matching functions $\mathbb{M}_{\mathbf{mx}_{08}}$ and $\mathbb{M}_{\mathbf{mx}_{09}}$ plus the ordered pnset and pcset matching functions $\mathbb{M}_{\overrightarrow{\mathbf{v}}}$ and $\mathbb{M}_{\overrightarrow{\mathbf{p}}}$. We see that in the first four examples (rows 1 to 4), the two chord symbols being compared are identical and all four matching functions output 1 as expected. Likewise, in row 5 the chords compared are major and minor triads so in all cases, the matching functions produce 0.

In row 6, the chords have different root pitchnames but are enharmonic equivalents. Here, functions $\mathbb{M}_{\mathbf{mx}_{08}}$, $\mathbb{M}_{\mathbf{mx}_{09}}$ and $\mathbb{M}_{\overrightarrow{\mathbf{p}}}$ output a 1 because they are blind to enharmonic spellings of the root whereas $\mathbb{M}_{\overrightarrow{\mathbf{v}}}$ outputs 0 because it compares pitchnames.

Row 7 shows two chords that are alternative interval spellings of the same set pitchclasses relative to the root. $\mathbb{M}_{\overrightarrow{\mathbf{p}}}$ produces a 1 because the pcsets of both chords are (0,3,6,9). However, $\mathbb{M}_{\overrightarrow{\mathbf{v}}}$ outputs 0 in this case because the intervals of a diminished seventh are (1,♭3,♭5,♭♭7) and will therefore evaluate to a different pnset compared with (1,♯2,♯4,6). It is

interesting to note the values of functions $\mathbb{M}_{\mathbf{mx}_{08}}$ and $\mathbb{M}_{\mathbf{mx}_{09}}$ for this example; $\mathbb{M}_{\mathbf{mx}_{08}}$ produces a 1 because, using mapping function $\mathbf{map}_{08}$ (equation 8.6), the '`dim7`' chordtype is mapped to the major category and any chord spelled with an interval list instead of a shorthand label will be also mapped to the major category regardless of its actual tonal qualities. Function $\mathbb{M}_{\mathbf{mx}_{08}}$ on the other hand produces a 0 because mapping function $\mathbf{map}_{09}$ (equation 8.7) maps '`dim7`' to minor while again treating interval list spellings as major. The unintended consequence of mapping chords to categories in this way are demonstrated more clearly in row 8 however because both $\mathbb{M}_{\mathbf{mx}_{08}}$ and $\mathbb{M}_{\mathbf{mx}_{09}}$ consider interval list spellings to be the same as the major chord label which is clearly an incorrect match in this case.

Rows 9 and 10 illustrate another problem of ignoring the interval list representation of chords. In row 9, a major triad is compared with a minor triad that has been spelled as an interval list. Both MIREX functions consider these to be the same whereas the pnset and pcset comparisons show that they are in fact different. The problem is compounded in row 10 where a correct match between a minor triad shorthand and minor triad interval list are evaluated as incorrect matches by the chord mapping functions.

Row 11 shows us a potential deficiency in the standard ordered matching functions. Where the MIREX functions consider '`maj`' and '`maj7`' chord types to be correct matches, $\mathbb{M}_{\overrightarrow{\mathbf{v}}}$ and $\mathbb{M}_{\overrightarrow{\mathbf{p}}}$ consider them different because they are of different cardinality. Depending on the algorithm being evaluated, this may not be a desirable result.

Rows 12 through 15 show the effects of the mappings where different chords are compared. In all cases, $\mathbb{M}_{\overrightarrow{\mathbf{v}}}$ and $\mathbb{M}_{\overrightarrow{\mathbf{p}}}$ produce 0 but the MIREX functions produce varying results. In row 12, both MIREX functions consider '`dim`' and '`sus2`' to be a correct match. In row 13, $\mathbb{M}_{\mathbf{mx}_{08}}$ considers '`sus2`' and '`sus4`' to be a correct match but $\mathbb{M}_{\mathbf{mx}_{09}}$ considers them to be different. In rows 14 and 15, the two functions disagree again on whether '`min`' matches '`dim`' or '`aug`'.

We can see from the results in table 8.2 that in a situation where correct enharmonic spelling is required, pnset matching is desirable. In

automatic chordal analysis of symbolic music where enharmonic spellings are known, such as [PB02], this would be a good method to use. However, for chord recognition from audio, the algorithms under evaluation cannot distinguish between different absolute enharmonic spellings[3] so we assert that an ordered pcset matching function is the best method to use in recall evaluation calculation.

Cardinality-limited recall evaluation

In row 11 of table 8.2 we saw that the straight forward matching function $\mathbb{M}_{\overrightarrow{\mathbf{p}}}$ evaluated the chords '`C:maj`' and '`C:maj7`' as an incorrect match because their cardinalities differ. This is a correct result but in many cases the chord recognition algorithm under evaluation may only be able to produce triad chord estimates so we may wish to consider '`maj`' and '`maj7`' chord types as a correct match in this situation.

In section 5.4 we introduced the cardinality parameter M to the matching function $\mathbb{M}_{T,M}$. This allows us to consider two chord symbols a correct match if a particular subset of the chords' elements are the same. Using the matching function $\mathbb{M}_{T,M}$, it therefore follows that to evaluate the frame-based recall for an annotated frame sequence A compared to an estimated sequence E (where both sequences are N_{T} frames in length), the number of correct matches N_{C} is

$$N_{\mathrm{C}} = \sum_{n=0}^{N_{\mathrm{T}}-1} \mathbb{M}_{T,M}(A_n, E_n) \tag{8.10}$$

and the recall equation becomes parameterised by T and M

$$\mathcal{R}_{T,M}(A, E) = \frac{\sum_{n=0}^{N_{\mathrm{T}}-1} \mathbb{M}_{T,M}(A_n, E_n)}{N_{\mathrm{T}}} \tag{8.11}$$

and likewise, the segment-based recall becomes

$$\mathcal{R}_{T,M}(S_E, S_A) = \frac{\sum_{S_A^j} \sum_{S_E^i} |S_E^i \cap S_A^j| . \mathbb{M}_{T,M}(S_E^i, S_A^j)}{\sum_{S_A^j} |S_A^j|} \tag{8.12}$$

[3]Some algorithms may be able to infer correct relative spellings between chords based on key analysis prior to chord recognition but even in this case the algorithm still would not be able to decide whether the original key was D♯ or E♭.

In table 8.2, the two rightmost columns show the results for the cardinality limited matching functions $\mathbb{M}_{\overrightarrow{\mathbf{p}},3}$ and $\mathbb{M}_{\overrightarrow{\mathbf{p}},2}$ which will effectively compare the triad and dyad subsets of chords respectively. We can see that these two functions produce exactly the same results as $\mathbb{M}_{\overrightarrow{\mathbf{p}}}$ in all but two rows. In row 11, both functions consider '`maj`' and '`maj7`' to be correct matches because they share the same cardinality-2 and cardinality-3 pcsets:

$$\{\overrightarrow{\mathbf{p}}_{\texttt{"C:maj"}}\}_3 = \{\overrightarrow{\mathbf{p}}_{\texttt{"C:maj7"}}\}_3 = \{0, 4, 7\} \tag{8.13}$$

so it follows that

$$\{\overrightarrow{\mathbf{p}}_{\texttt{"C:maj"}}\}_2 = \{\overrightarrow{\mathbf{p}}_{\texttt{"C:maj7"}}\}_2 = \{0, 4\} \tag{8.14}$$

We should also note that in row 14, the results for $\mathbb{M}_{\overrightarrow{\mathbf{p}},3}$ and $\mathbb{M}_{\overrightarrow{\mathbf{p}},2}$ differ. This is because the '`min`' and '`dim`' chord types are different triads but they both contain a minor third interval

$$\begin{aligned} \{\overrightarrow{\mathbf{p}}_{\texttt{"C:min"}}\}_3 &= \{0, 3, 7\} \\ \{\overrightarrow{\mathbf{p}}_{\texttt{"C:dim"}}\}_3 &= \{0, 3, 6\} \\ \therefore \{\overrightarrow{\mathbf{p}}_{\texttt{"C:min"}}\}_3 &\neq \{\overrightarrow{\mathbf{p}}_{\texttt{"C:dim"}}\}_3 \end{aligned} \tag{8.15}$$

but

$$\{\overrightarrow{\mathbf{p}}_{\texttt{"C:min"}}\}_2 = \{\overrightarrow{\mathbf{p}}_{\texttt{"C:dim"}}\}_2 = \{0, 3\} \tag{8.16}$$

By defining the cardinality of the evaluation in this way, any chord from the transcription collection can be compared with any machine-estimated chord and a reliable evaluation of a match can be made.

Another useful property of using this system is that we can run several recall evaluations on our recognition results using different parameters which can give us useful information about the algorithm performance such as how good it is at identifying the root or the first interval correctly compared to its triad recognition. This also allows true 'apples to apples' comparisons between different algorithms to be made possible. For example, if two researchers write recognition algorithms that can only produce triad estimates then a cardinality-3 recall calculation is a suitable way of comparing them. If one researcher then writes a tetrad recognition algorithm, they may wish to evaluate it with a cardinality-4 recall calculation

to see how the algorithm performs on tetrads. However, if they wish to compare the new algorithm against the triad algorithms, they can also run a cardinality-3 evaluation to compare the algorithms directly.

Treatment of bass intervals

Another chord property we may wish to include in our evaluation is inversion (discussed in section 5.5). Using pcset comparison this will be achieved automatically because, by definition, the ordering of the pcsets will naturally reflect the inversion of a chord. Caution must be exercised using this kind of approach however because it will lead to issues with the evaluation of root identification accuracy. To illustrate, we recall the example in section 4.2.3 where chords `C:min7/b3` and `Eb:maj6` were shown to be equivalent in terms of their constituent pitchnames. In a system that considers inversions, these two chords will obviously evaluate as a correct match:

$$\overrightarrow{\mathbf{p}}_{\texttt{"C:min7/b3"}} = \overrightarrow{\mathbf{p}}_{\texttt{"Eb:maj6"}} = \{3, 7, 10, 0\} \tag{8.17}$$

therefore

$$\mathbb{M}_{\overrightarrow{\mathbf{p}},M}(\texttt{"C:min7/b3"}, \texttt{"Eb:maj6"}) = 1 \tag{8.18}$$

This is obviously the correct result for a system that compares ordered pcsets but now we have the situation where two chord symbols with completely different root pitchnames (i.e. not enharmonic equivalents) can be considered to be a correct match.

The majority of chord recognition algorithms currently focus on root and chord type recognition. Few can recognise inversions so for the recall evaluations on these algorithms it is sensible to use the bass-blind matching function $\mathbb{M}'_{\overrightarrow{\mathbf{p}},M}$ (see equation 5.78). This means that if a chord recognition algorithm outputs a chord symbol estimate with the correct root and chordtype but the annotated chord is also marked as an inversion, it will still be recognised as a correct match.

In an evaluation system we now have a choice of methods for calculating the chord symbol recall. By including bass intervals we may compare different inversions of chords or we may use a bass-blind comparison instead for correct root identification. However, there is nothing to stop

Table 8.3: Mapping of chord shorthands for an inversion-blind, cardinality-3 evaluation of a major-minor algorithm. Mappings are shown for major and minor triads along with a third group that are unmatchable.

Category	Chord types
major:	`maj`, `maj7`, `7`, `maj6`, `9`, `maj9`
minor:	`min`, `min7`, `minmaj7`, `min6`, `min9`
unmatchable:	`aug`, `dim`, `dim7`, `hdim7`, `sus2`, `sus4`

us calculating more than one evaluation result by using different chord matching parameters (which should be clearly stated) for each evaluation. In fact we would argue that this is a very good approach for showing information about different aspects of an algorithm's performance.

8.1.4 Dictionary-based recall evaluation

Using ordered pcset comparison with specified parameters for maximum cardinality and treatment of inversion we can now successfully compare any two sets of chord symbols to evaluate the recall measure. However, this approach still does not completely solve the problems caused by the differing chord vocabularies between the machine estimates and hand annotated transcriptions. Table 8.4 shows the vocabularies of the different chord recognition algorithms that were entered for the MIREX09 chord detection (untrained) evaluation. These algorithms will be evaluated in section 9.

If a recognition algorithm can only recognise major or minor triad chords then we know *a priori* that there is a large set of chords that will never match either of them for any cardinality greater than 1. Given that the algorithm is blind to any chord types other than major and minor, should we necessarily penalise it for being unable to recognise chords it was not designed to 'see' to start with?

Considering a simple major-minor triad recognition algorithm; for an inversion-blind, cardinality-3 evaluation, the seventeen shorthand chord types will be mapped as shown in table 8.3. It should be noted that this table does not include the very large set of all other possible non-shorthand

Table 8.4: Chord vocabularies for MIREX09 (untrained) chord recognition algorithm entries plus two extra chord recognition algorithms by the author marked with an asterisk*

Algorithm	Abstract	Author(s)	Chord vocabulary
ch_aes	*	Harte	`N, maj, min, aug, dim`
ch_hcdf	[HS09]	"	`N, maj, min, aug, dim`
ch_hcdfa	*	"	`N, maj, min, aug, dim`
de	[Ell09]	Ellis	`N, maj, min`
ko1	[KO09a]	Khadkevich, Omologo	`N, maj, min`
ko2	"	" "	`N, maj, min`
md	[MND09a]	Mauch, Dixon	`N, maj, min, dim, 7, maj6, maj7`
ogf1	[OGF09a]	Oudre, Grenier, Fevotte	`N, maj, min`
ogf2	[OGF09b]	" " "	`N, maj, min, 7`
pp	[PP09]	Papadopoulos, Peeters	`maj, min`
pvm1	[PVM09]	Pauwels, Varewyck, Martens	`maj, min, dim, aug`
pvm2	"	" " "	`maj, min, dim, aug`
rrhs1	[RRHS09a]	Rocher, Robine, Hanna, Strandh	`N, maj, min`
rrhs2	"	" " " "	`N, maj, min`
rrhs3	"	" " " "	`N, maj, sus4, aug, 7, sus2, maj(9), min, min7, dim, min(9), dim7`

chord types that would also be mapped into these three groups for such an evaluation.

What should we do with the chords that are mapped into the unmatchable category? We propose that if we know in advance that they cannot ever be matched then frames containing these chord types should be excluded from the recall evaluation and the number of omitted frames should be recorded along with the recall result. This will provide a more accurate idea of the algorithm's performance than an evaluation that includes a large number of frames that can never possibly give correct matches. It will also give a good indication as to the suitability of a particular song for evaluation purposes. Pardo and Birmingham [PB02] make a similar argument in their approach to evaluating their symbolic chord analysis system where chord symbols that are not part of the algorithm's vocabulary are excluded from the evaluation.

Although the recall evaluation will exclude these unmatchable chord types, it will still be informative to analyse the confusion between the transcribed and machine-estimated chord symbols for the unmatchable frames even though they are not relevant to the quantitative recall performance of the algorithm.

Let D_α be a set of chordtype strings (as defined in section 5.1, equations 5.1 and 5.2) that defines the chord vocabulary for a given recognition algorithm α. We will call D_α the *dictionary* for that algorithm. For the major-minor recognition algorithm (which we will call algorithm 0 here), the dictionary is therefore:

$$D_0 = \{\texttt{"maj"}, \texttt{"min"}\} \tag{8.19}$$

If A_n is the transcribed chord symbol for frame n in a cardinality-M pcset recall evaluation for algorithm α then we will define a pcset *inclusion* function $\mathbb{I}_{\overrightarrow{\mathbf{p}},M,D_\alpha}(A_n)$ such that

$$\mathbb{I}_{\overrightarrow{\mathbf{p}},M,D_\alpha}(A_n) = \begin{cases} 1 & \text{if} \quad \exists\ \mathtt{Q_X} \in D_\alpha \quad \text{such that} \quad \mathbb{M}_{\overrightarrow{\mathbf{r}},M}\big(A_n, \mathtt{X}\big) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{8.20}$$

i.e. if the chordtype $\mathtt{Q}_{A_n}$ of symbol A_n matches one of the chordtypes $\mathtt{Q_X}$ in the dictionary D_α at the given cardinality M then frame n will be included

in the recall result; otherwise it should be excluded. We may also define a pnset inclusion function $\mathbb{I}_{\overrightarrow{\mathbf{v}},M,D_\alpha}$ in the same way:

$$\mathbb{I}_{\overrightarrow{\mathbf{v}},M,D_\alpha}(A_n) = \begin{cases} 1 & \text{if} \quad \exists\ \mathtt{Q_X} \in D_\alpha \quad \text{such that} \quad \mathbb{M}_{\overrightarrow{\mathbf{i}},M}\big(A_n, \mathtt{X}\big) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{8.21}$$

To illustrate, if we use a cardinality-3 pcset evaluation on algorithm 0, a frame containing a transcribed '`F:sus4`' chord will be excluded from the results because chordtype '`sus4`' is not a member of dictionary D_0:

$$\forall\ \mathtt{Q_X} \in D_0 \qquad \mathbb{M}_{\overrightarrow{\mathbf{r}},3}(\texttt{"F:sus4"}, \mathtt{X}) = 0 \tag{8.22}$$

therefore

$$\mathbb{I}_{\overrightarrow{\mathbf{p}},3,D_0}(\texttt{"F:sus4"}) = 0 \tag{8.23}$$

However, a transcribed '`Gb:7`' chord *would* be included in the evaluation because at cardinality-3 it is equivalent to the '`maj`' chordtype i.e.

$$\exists\ \mathtt{Q_X} \in D_0 \quad \text{for which} \quad \mathbb{M}_{\overrightarrow{\mathbf{r}},3}(\texttt{"Gb:7"}, \mathtt{X}) = 1 \tag{8.24}$$

so

$$\mathbb{I}_{\overrightarrow{\mathbf{p}},3,D_0}(\texttt{"Gb:7"}) = 1 \tag{8.25}$$

Let us consider another recognition algorithm that can detect only major, minor and major-seven chords, which we will call algorithm 1. We may define its dictionary as

$$D_1 = \{\texttt{"maj"}, \texttt{"min"}, \texttt{"maj7"}\} \tag{8.26}$$

For this algorithm, a frame containing a transcribed '`F:sus4`' will still be excluded from the result for any cardinality greater than 1. However, while the '`Gb:7`' will be still included for a cardinality-3 evaluation, it will be excluded for cardinality-4 because the three chordtypes in the dictionary do not match the '`7`' chordtype for this cardinality

$$\forall\ \mathtt{Q_X} \in D_1 \quad \mathbb{M}_{\overrightarrow{\mathbf{r}},4}(\texttt{"Gb:7"}, \mathtt{X}) = 0 \tag{8.27}$$

so

$$\mathbb{I}_{\overrightarrow{\mathbf{p}},4,D_1}(\texttt{"Gb:7"}) = 0 \tag{8.28}$$

but

$$\mathbb{I}_{\overrightarrow{\mathbf{p}},3,D_1}(\texttt{"Gb:7"}) = 1 \tag{8.29}$$

Using this dictionary-based evaluation, we may calculate the number of frames $N_{\mathcal{E}}$ of annotated sequence A that will be included in the evaluation in the following way

$$N_{\mathcal{E}} = \sum_{n=0}^{N_{\mathrm{T}}-1} \mathbb{I}_{T,M,D_\alpha}(A_n) \tag{8.30}$$

We must also alter the calculation of N_{C} to avoid possible problems caused by using a dictionary that is a subset of the chord vocabulary for the algorithm under test. This is likely to be the case when comparing several different algorithms and choosing one common dictionary to evaluate them with. In this case we wish to exclude non-dictionary chords in the estimated sequence from the summation of N_{C}:

$$N_{\mathrm{C}} = \sum_{n=0}^{N_{\mathrm{T}}-1} \mathbb{M}_{T,M}(A_n, E_n).\mathbb{I}_{T,M,D_\alpha}(E_n) \tag{8.31}$$

so the frame-based recall $\mathcal{R}_{T,M,D_\alpha}$ between annotated sequence A and machine-estimated sequence E for cardinality-M and dictionary D_α where $T \in \{\overrightarrow{\mathbf{v}}, \overrightarrow{\mathbf{p}}\}$ becomes

$$\mathcal{R}_{T,M,D_\alpha}(A, E) = \frac{N_{\mathrm{C}}}{N_{\mathcal{E}}} = \frac{\sum_{n=0}^{N_{\mathrm{T}}-1} \mathbb{M}_{T,M}(A_n, E_n).\mathbb{I}_{T,M,D_\alpha}(E_n)}{\sum_{n=0}^{N_{\mathrm{T}}-1} \mathbb{I}_{T,M,D_\alpha}(A_n)} \tag{8.32}$$

and likewise the segment-based chord symbol recall becomes

$$\mathcal{R}_{T,M,D_\alpha}(S_E, S_A) = \frac{\tau_{\mathrm{C}}}{\tau_{\mathcal{E}}} = \frac{\sum_{S_A^j} \sum_{S_E^i} |S_E^i \cap S_A^j|.\mathbb{M}_{T,M}(S_E^i, S_A^j).\mathbb{I}_{T,M,D_\alpha}(S_E^i)}{\sum_{S_A^j} |S_A^j|.\mathbb{I}_{T,M,D_\alpha}(S_A^j)} \tag{8.33}$$

It is important to note that we are now no longer evaluating across the whole hand annotated ground truth, but a subset thereof specified by the dictionary D_α. Because of this, it is important that we make a record of proportion of material discarded as 'non-dictionary'. For frame-based recall calculations the proportion of non-dictionary frames is given by $1 - \frac{N_{\mathcal{E}}}{N_{\mathrm{T}}}$ and for segment-based recall it is the proportion of non-dictionary segment duration $1 - \frac{\tau_{\mathcal{E}}}{\tau_{\mathrm{T}}}$. Figure 8.3 shows how dictionary matching could be used on the original chord symbol recall example from figure 8.1 where

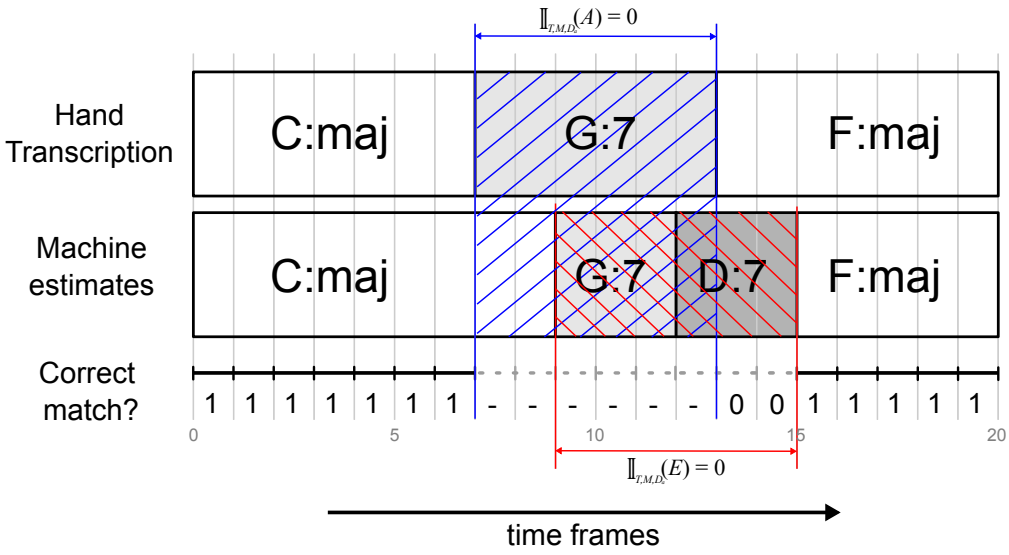


Figure 8.3: Evaluating the chord symbol recall using a dictionary that excludes the `X:7` chord type.

the recall was calculated as 0.75. In the dictionary example, the '`X:7`' chord type is not in the dictionary thus $\mathbb{I}_{T,M,D_\alpha}(A_n) = 0$ for the frames in the annotation that contain '`G:7`' and are therefore discarded from the summation of $N_{\mathcal{E}}$. $\mathbb{I}_{T,M,D_\alpha}(E_n) = 0$ for the '`G:7`' and '`D:7`' chords in the estimated sequence so these are discarded from the summation of N_{C}. In this example, the recall value would be $\frac{12}{14} = 0.86$ which is higher than before but we must remember that six frames are discarded so only 70% have actually been used in the evaluation.

This approach to recall evaluation effectively treats the annotated song data as a collection of individual test cases for an algorithm i.e. each different chord symbol and the audio data associated with it is an individual test. By specifying the dictionary for a given algorithm we decide which test cases to include from the collection. The algorithm should still be run on the whole audio file but we can then choose how to evaluate the resulting machine estimated chord sequence depending on the dictionary we define for it.

A good example of where this approach is very useful in practice is evaluating a song such as 'Mr Moonlight' from the album *Beatles for sale*.

Table 8.5: Chord type statistics for the 'Mr Moonlight' Beatles transcription. The song is 157.2 seconds long and the transcription contains 100 symbols.

Chord type	Symbol frequency	Aggregate time	% total time
`N`	2%	4.57s	2.9%
`X:maj`	56%	98.11s	62.4%
`X:sus4`	33%	31.36s	19.9%
`X:min`	1%	7.52s	4.8%
`X:7`	8%	15.64s	9.9%

Table 8.5 shows the distribution of chordtypes in terms of symbol frequency and aggregate duration for the song[4]. For this song, 33% of the transcribed symbols are chord '`F#:sus4`' accounting for just under 20% of the total song duration. All the other chord symbols in the transcription are types that will be mapped into major or minor categories for a cardinality-3 evaluation but '`F#:sus4`' will be unmatchable.

Suppose the example algorithm-0 produces 10ms frames and can successfully detect a major or minor chord with 70% accuracy for audio that is known to contain a major or minor chord type. If we run this algorithm on 'Mr. Moonlight' we will have a total of $N_{\mathrm{T}} = 15720$ estimate frames. If we evaluate the frame-based recall using a cardinality-3 pcset matching function then we already know that 20% of the total number of frames (i.e. the 3144 frames for which $A_n =$ '`F#:sus4`') cannot possibly be matched so we choose to exclude them; thus $N_{\mathcal{E}} = 15720 - 3144 = 12576$ frames. If the algorithm achieves 8803 correct matches then, using the dictionary based evaluation, the recall will be $\mathcal{R}_{\overrightarrow{\mathbf{p}},3,D_0} = \frac{8803}{12576} = 0.7$ as we would expect. However, if we calculate the recall without excluding the unmatchable chords, the recall value will be $\mathcal{R}_{\overrightarrow{\mathbf{p}},3} = \frac{8803}{15720} = 0.56$ which is a significantly worse result. It should be noted that 'Mr Moonlight' is unusual in the collection for its high percentage of '`sus4`' chords which makes it a good example here but it is not at all exceptional in containing chords which most algorithms still cannot recognise. For most songs in the collection, the effect will be less pronounced but, as the example

[4]By coincidence, the total number of chord symbols in the transcription for this song is 100 so the chord frequencies can be read directly as percentages.

demonstrates, it is definitely an issue that should be taken into account.

8.2 Chord sequence likeness measure

The frame-based recall measure $\mathcal{R}$ is an 'all or nothing' approach to evaluating the accuracy of chord symbol recognition. Each pair of chord frames (A_n,E_n) is either classified as a match, in which case it contributes to the final score, or it is deemed not to be a match and is ignored. This measure could be considered to be too polarised to give a full picture of the recognition system performance and, in discussions on the MIREX09 chord detection track wikipage [mirc], it has been suggested that an alternative, linear measure be found that shows how 'close' chords are to each other. Such a measure would be based on how many pitchnames or pitchclasses were correctly identified for each frame instead of whether the chords were an exact match.

One way to do this is to look at the shared tones in the two chord frames A_n and E_n by using the chord likeness measure $\mathbb{L}_T$ introduced in section 5.6. By using the likeness measure in place of a matching function, we can calculate an overall chord sequence likeness measure $\mathcal{L}_T$ in a similar way to calculating recall thus

$$\mathcal{L}_T(A, E) = \frac{\sum_{n=0}^{N_\mathrm{T}-1} \mathbb{L}_T(A_n, E_n)}{N_\mathrm{T}} \tag{8.34}$$

As with the chord symbol recall calculation, the chord likeness function can be computed as a function of continuous time, calculating the chord likeness for overlapping segments of chord sequence A and E:

$$\mathcal{L}_T(S_E, S_A) = \frac{\sum_{S_A^j} \sum_{S_E^i} |S_E^i \cap S_A^j| . \mathbb{L}_T(S_E^i, S_A^j)}{\tau_\mathrm{T}} \tag{8.35}$$

Using the chord sequences from figure 8.1 as an example again, we can calculate a pcset chord sequence likeness measure in the following way:

$$\begin{aligned}\mathcal{L}_{\overrightarrow{\mathbf{p}}} = \tfrac{1}{20} \times \Big(& 7 \times \mathbb{L}_{\overrightarrow{\mathbf{p}}}(\texttt{"C:maj"}, \texttt{"C:maj"}) + 2 \times \mathbb{L}_{\overrightarrow{\mathbf{p}}}(\texttt{"G:7"}, \texttt{"C:maj"}) + \\ & 3 \times \mathbb{L}_{\overrightarrow{\mathbf{p}}}(\texttt{"G:7"}, \texttt{"G:7"}) + 1 \times \mathbb{L}_{\overrightarrow{\mathbf{p}}}(\texttt{"G:7"}, \texttt{"D:7"}) + \\ & 2 \times \mathbb{L}_{\overrightarrow{\mathbf{p}}}(\texttt{"F:maj"}, \texttt{"D:7"}) + 5 \times \mathbb{L}_{\overrightarrow{\mathbf{p}}}(\texttt{"F:maj"}, \texttt{"F:maj"}) \Big)\end{aligned}$$

$$= \frac{7 \times 1 + 2 \times \frac{1}{6} + 3 \times 1 + \frac{1}{7} + 2 \times \frac{2}{5} + 5 \times 1}{20} = \frac{16.2762}{20} = 0.8138$$

So we see that measure $\mathcal{L}_{\overrightarrow{\mathbf{p}}}$ for this example is higher than the recall value $\mathcal{R}_{\overrightarrow{\mathbf{p}}}$ which was 0.75. In fact this will always be the case because when $\mathbb{M}_T$ is 1 then $\mathbb{L}_T$ will also be 1 but $\mathbb{L}_T$ can be greater than 0 where $\mathbb{M}_T$ is 0 hence in general $\mathcal{L}_T \geq \mathcal{R}_T$.

Using a chord likeness measure in this way can tell us more information about our results than using recall on its own. For example, two algorithms may get the same recall score yet one may get a higher likeness score than the other suggesting that one is better at generally detecting correct pitchclasses in the audio but perhaps its chord symbol recognition based on those features is inferior.

It should be noted that it would be possible to get quite a high score for $\mathcal{L}_T$ without ever actually correctly recognising a chord symbol at all. For example, an algorithm might consistently choose the third or fifth of a chord for the root of a chord instead of the correct note. In this case, the recall value would be very low but $\mathcal{L}_T$ would make the algorithm appear to perform well. This may be a pleasing result for algorithm designers but the whole point of designing chord recognition algorithms is to try to correctly recognise chords. The likeness measure tells us how good an algorithm is at detecting pitchclasses at a particular time in the audio source material but does not necessarily give an accurate reflection of the chord symbol recognition accuracy. For this reason we believe that although it is a potentially useful source of information about our algorithms' performance, a chord sequence likeness measure like this should only be used as well as, and not instead of the stricter chord symbol recall measure.

8.3 Chord recognition segmentation measurement

In this section we will discuss the use of a segmentation metric to evaluate chord recognition results providing an important complementary measure to the chord symbol recall measure already covered in section 8.1.

8.3.1 Problems with using recall on its own

The chord symbol recall measure provides a good way of measuring how accurate a recognition algorithm is in terms of whether the estimated chord for a given instant in the audio is correct. However, using this measure alone fails to show how consistent the recognition algorithm is. As an example, consider the annotated sequence and three estimated chord sequences compared with it in figure 8.4. We can see that all three estimated sequences have a recall value of 0.6 because each has 12 correctly matching frames out of the total of 20 frames. However, sequence 1 is more musically useful because estimation of chord boundaries is better than for the other two. In the case of sequence 2, the chord recogniser has produced a large number of short continuous chord segments creating a highly fragmented chord sequence. In sequence 3 by contrast the chord recogniser has completely missed the first chord change, staying on the same chord for the first 15 frames.

As human listeners comparing these estimated sequences to the original, we may consider sequence 1 to have one major mistake that is to incorrectly identify the '`G:7`' chord as an '`F:7`'. The second sequence on the other hand, jumps around from chord to chord making the structure of the harmony much less coherent. From a musician's perspective, estimate 1 could be used in a lead sheet with one minor correction whereas estimate 2 would be useless by comparison. Likewise, sequence 3, although better than the fragmented sequence 2, does not identify that there has been a chord change at all for the first 15 frames so a musician would have to listen for both a missing chord change *and* the correct second chord instead of just correcting the second chord in sequence 1. From these examples we can see that we require a way of measuring how good our algorithms are in terms of fragmentation or missed chord changes as well as finding the chord symbol recall in order to give a proper overview of their performance.

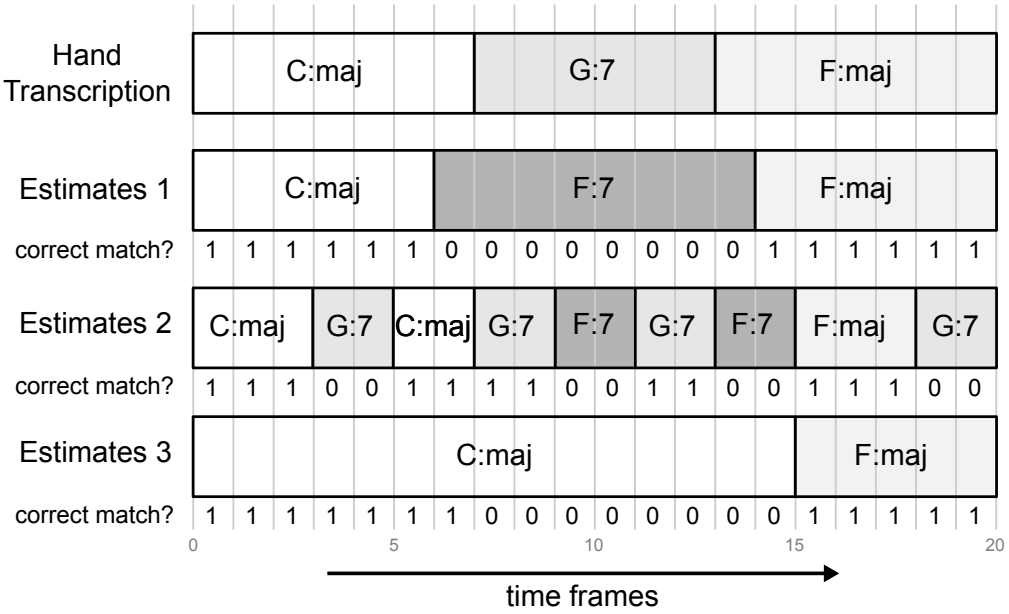


Figure 8.4: Three example estimated chord sequences compared to a hand annotated sequence. All three estimated sequences have the same recall value $\frac{12}{20} = 0.6$ but sequences 2 and 3 are clearly poorer estimates than sequence 1 due to over-segmentation and under-segmentation respectively.

Measuring chord boundary detection accuracy

To deal with the issue of segmentation, we could look at how accurately the chord recognition algorithm detects chord boundaries. A simple approach used in [MD08] and [KO09b] is the chord change rate where the number of boundaries in the hand transcription is compared to that of the machine estimated chord sequence. If the numbers differ significantly, this will tend to suggest fragmentation (i.e. more estimated changes than the ground truth) or a large number of missed chord changes (fewer changes than the ground truth). This measure, however, does not tell us if the chord segment lengths are accurate or not. For example, an estimated sequence could be uniformly fragmented all the way through so it will have a high chord change rate compared to the hand transcription. On the other hand, another estimated sequence might be quite accurate for most parts of the song but have a short period of time where the fragmentation is very high giving the same change rate.

Another approach is to define an allowable time deviation window for

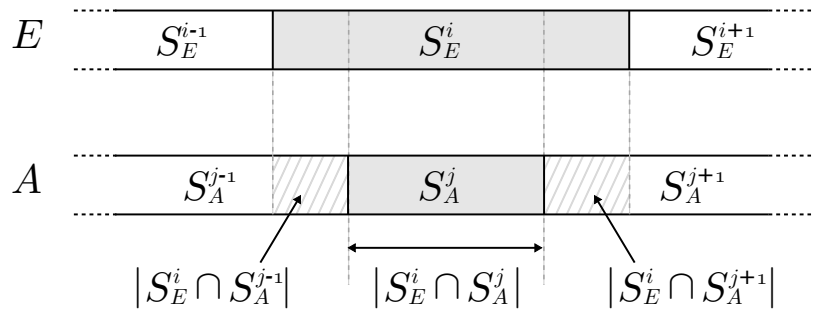


Figure 8.5: Segments of estimated sequence E compared with hand annotated sequence A. The hatched areas are summed in the calculation of directional hamming distance.

chord onsets. That is to say that an estimated chord boundary is a 'hit' if it is within a window length of a hand labelled boundary or a 'miss' otherwise. This approach can give information about how accurate the chord boundary detection is but it has the problem of how to define the size of the allowable onset deviation window.

For the evaluation to be fair, the onset time deviation window must be constant across the whole test set. However, if the time window is set rigidly to a particular number of milliseconds then it may be inappropriate for certain types of music. For example, a slow ballad may have very few chord changes, widely spaced in time so a long window would seem sensible to avoid false negative results. On the other hand, fast bebop jazz might have a chord change on every beat in which case the window would have to be very short in order for false positives to be avoided.

8.3.2 Directional hamming distance

A better solution to measuring chord segmentation, rather than looking at the boundaries themselves is to look at the gaps between them. Mauch et al [MND09b, Mau10] proposed the use of the directional hamming distance as a measure of segmentation quality for chord recognition.

The directional hamming distance (also sometimes called *hamming divergence*) is a measure originally proposed by Huang and Dom [HD95] for image segmentation evaluation and later used by Abdallah et al [ANS+05]

for evaluating audio segmentation. For this measure we consider hand annotated sequence A as a sequence of segments S_A^j and machine estimated sequence E similarly as segments S_E^i. Abdallah et al [ANS+05] show how we may compute the directional hamming distance d_{AE} by finding for each estimated segment S_E^i, the largest overlapping segment S_A^j in the annotated sequence i.e. $\max\left(|S_E^i \cap S_A^j|\right)$, then summing the difference between them:

$$d_{AE} = \sum_{S_E^i} \sum_{S_A^k \neq S_A^j} |S_E^i \cap S_A^k| \tag{8.36}$$

Figure 8.5 shows this concept visually. The hatched areas in the diagram represent the frames in A that overlap segment S_E^i but which are not part of the maximal overlapping segment S_A^j so those areas will contribute to the d_{AE} summation.

By normalising distance d_{AE} by the length τ_{T} of the sequences A and E, we can find a measure of the missed chord boundaries, or under-segmentation m

$$m = \frac{d_{AE}}{\tau_{\mathrm{T}}} \tag{8.37}$$

Likewise, we may calculate the distance d_{EA} and normalise by τ_{T} to provide a measure of the fragmentation, or over-segmentation f

$$f = \frac{d_{EA}}{\tau_{\mathrm{T}}} \tag{8.38}$$

We now have two measures m and f that tell us about the quality of a chord recognition algorithm's segmentation. In the ideal case where estimated sequence E exactly equals annotated sequence A, both measures will be 0 as there will be no over-segmentation or under-segmentation. However, in the likely event that the two sequences do not match each other perfectly, it should be noted that although in an extreme case we might be able to obtain a value of 1 for m or f, it is not possible to obtain the value 1 for both at the same time. For example, a value of 1 for f means 100% fragmentation which implies that no two contiguous frames in the estimated sequence E contain the same chord. In this case, it is not possible for any under-segmentation to exist. Likewise, a value of 1 for m means 100% missed boundaries which implies that sequence E contains

only one segment that lasts for its whole length making it impossible for any fragmentation to be present.

For a useful measure of segmentation quality that will complement the chord symbol recall, it is desirable to combine the segmentation values m and f into one measure. Mauch et al [MND09b] proposed such a combined measure **MDseg** calculated using the arithmetic mean of m and f:

$$\textbf{MDseg} = \frac{f+m}{2} = \frac{d_{EA}+d_{AE}}{2\tau_{\mathrm{T}}} \tag{8.39}$$

This measure however, suffers from the fact that m and f are not independent. It is possible that a very low value of m or f could make the segmentation results look better than they really are by hiding a high value in the other.

An alternative to this approach is to use the worst case of m or f i.e. $\max(m, f)$ as the basis for the combined measure following evaluation principles described in [MMDK07]. If both are low values, then the segmentation quality is good, if one is a high value then we know the segmentation quality is poor so we ignore the other value. It will also be useful to convert the values such that the final measure has a worst case score of 0 and a best case of 1 so it truly complements the chord symbol recall. Therefore, we propose a segmentation quality measure $\mathcal{Q}$ that fulfils these requirements calculated as

$$\mathcal{Q}(A, E) = 1 - \max(m, f) = 1 - \frac{\max(d_{AE}, d_{EA})}{\tau_{\mathrm{T}}} \tag{8.40}$$

This idea was first proposed by the author as part of a discussion on the MIREX09 wikipage for the chord detection track [mirb] and was subsequently adopted by Mauch in his recent work [Mau10]. The measure is based entirely on the arrangement of contiguous segments of frames in the chord recogniser's output. It does not depend on what the chord symbols are in each segment so it provides a truly complementary measure to the chord symbol recall $\mathcal{R}$. This being the case, we find an additional benefit of the new measure in that it can help to detect situations where the chord recognition algorithm under test has detected the correct relative chord sequence but, due to tuning issues, may produce a chord sequence that is ± 1 semitone from the hand annotation. Figure 8.6 shows a graphical

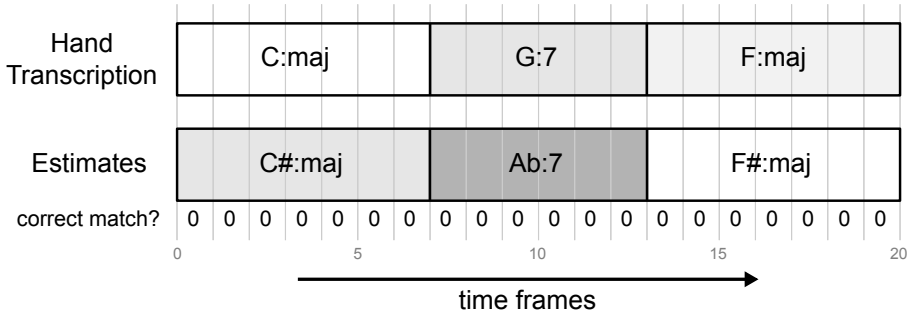


Figure 8.6: Due to tuning, an otherwise correct estimated chord sequence may be consistently 1 semitone higher than the hand annotated sequence. In this case, recall $\mathcal{R}$ will be zero but the segmentation measure $\mathcal{Q}$ will be 1.

example of this situation where all chords have been recognised as having a root note one semitone higher than those in the hand annotation. In the figure, we would achieve a zero recall score but a 100% segmentation score. In this kind of situation, whereas before we would just have had an apparently anomalous recall value approaching zero, we will now also have a very high score for $\mathcal{Q}$ which tells us we should check to see if this transposition problem has occurred. In real-world music audio recordings, it is not unlikely that the tuning frequency may deviate from A440 concert pitch; this is definitely true of the Beatles collection as discussed in section 6.5.2. If the tuning frequency of a particular song recording is a quarter tone above or below A440 then the human transcriber may well decide to annotate one way while the chord recognition algorithm goes the other and both will effectively be correct.

8.4 Practical considerations

When evaluating a chord recognition system it is simple for the researcher to write evaluation routines that use the same internal data structures such as chord frames etc. as the recognition system itself. Unfortunately, this makes the use of one person's evaluation software on another person's results very difficult because the recognition and evaluation software can be heavily intertwined making a complicated conversion process necessary. This problem can be solved if both chord recognition systems are

programmed to produce an output in a standard format and a completely separate evaluation system can be used to test results.

The Beatles transcription collection is available in the form of ASCII text files (with extension `.lab`) as described in section 6.2. This file type is an ideal output format for chord recognition systems as well because the files hold all the necessary information about the recognised chord sequence but implementation specific parameters of the recogniser such as frame rate and hop size etc are not retained. In general, the conversion of an internal chord sequence representation to `.lab` format is trivial.

We have produced an evaluation system for Matlab that can compare two `.lab` files and give results for the chord symbol recall, the chord sequence likeness and the chord segmentation quality. These functions are available in the C4DM chords toolkit discussed in section 4.4 and can be used to evaluate any chord recognition algorithm that is capable of producing results in the `.lab` format.

8.4.1 Frame-based vs segment-based recall evaluation

For calculating the chord symbol recall between two lab files, segment-based recall is a more accurate and more efficient method than frame-based recall for any reasonable frame rate. By reasonable, here we mean a frame rate high enough to give meaningful evaluation results for a chord recognition algorithm. Given that certain types of music such as fast jazz might have a chord change on each beat, even at the moderate tempo of 120bpm this equates to two chords per second so for frame-based analysis, the evaluation frame rate should definitely be above 2fps. The shortest tonal chord in the Beatles transcriptions is in fact 0.174 seconds long (see section 6.6.5) therefore the evaluation frame rate should be at least 5.7fps and ideally faster than 11.4fps to meet the Nyquist criterion.

Lee and Slaney [LS07, Lee08, LS08] used part of the Beatles transcription collection as a ground truth data set. The frame rate of their chord recogniser is 5.4fps so the chord frames are effectively 180ms long. For evaluation purposes they sampled the Beatles transcription files at this rate and quote their results for frame-based recall in two ways. The first value

is a direct frame-by-frame match which is equivalent to using frame-based recall with a frame rate of 5.4fps. The results for their direct frame-by-frame match were negatively affected by missed chord boundaries so they introduced a second measure which allowed a 1 frame tolerance either side of a chord boundary. The effective evaluation frame rate drops to 1.8fps, making the evaluation frame length 540ms with a hop of 180ms which improved their results. Their stated justification for using the larger evaluation frame size was that the hand annotated transcriptions could not be accurate because they were produced by a human listener. However, the chord annotations are accurate to approximately 10-20ms (see section 6.3) so their statement suggests the local audio files used in the experiments were probably not properly time aligned with the transcriptions (see chapter 7). We would argue that for a chord recognition algorithm with a low frame rate like this, it is better to use a high evaluation frame rate to calculate recall, effectively oversampling the algorithm's output instead of undersampling the ground truth.

8.4.2 Combined chord recognition F-measure

In many areas of information retrieval, recall and precision values are often combined into a single score called the *f-measure* which is calculated as the harmonic mean of recall and precision. For chord recognition evaluation we can combine the complementary recall and segmentation scores in a similar way to give us an overal chord recognition f-measure:

$$\mathcal{F}_{T,M,D}(A,E) = \frac{2 \times \mathcal{R}_{T,M,D}(A,E) \times \mathcal{Q}(A,E)}{\mathcal{R}_{T,M,D}(A,E) + \mathcal{Q}(A,E)} \tag{8.41}$$

This measure gives us more information about the performance of a chord recognition algorithm than using chord symbol recall on its own.

8.4.3 Which metrics to use

In this chapter we have suggested several different metrics with various parameters for chord recognition evaluation. In all cases it is recommended that both recall and segmentation quality be evaluated because, as mentioned in section 8.3, recall on its own is not sufficient to prove that an

algorithm can produce a good chord transcription.

The particular recall metric and set of parameters that should be used in an evaluation depends on the capability of the algorithm or algorithms under test. In the case where one algorithm is being evaluated in isolation, it is best to use a dictionary recall evaluation where the dictionary equals the intended vocabulary of the algorithm. To determine which cardinality value to use, we must consider whether it is acceptable to match chords that are supersets of dictionary symbols (e.g. '`C:maj7`' matching '`C:maj`') or not. For an example algorithm that can produce only major and minor triads, cardinality-3 will give results that assume all supersets of the dictionary chords will match. Using a higher cardinality value with the same algorithm and dictionary will reduce the number of possible matches for symbols in the dictionary so the evaluation score will likely be lower. However, using the cardinality parameter in this way allows us to see how good an algorithm is at recognising a specific chord label. In an evaluation where several algorithms are compared with each other, it is important to use the same recall parameters for all of them. The dictionary for evaluating a group of algorithms should therefore contain only the chordtypes that are common to all the algorithms under test in order to make the comparison fair. We demonstrate the use of these different metrics and parameters in chapter 9.

Chapter 9

Results

In this chapter we will present the results for the three chord recognition algorithms we developed in chapter 3 compared to equivalent results for the other eleven algorithms enteredto the MIREX09 chord detection evaluation [MND09a, KO09a, OGF09a, PP09, PVM09, RRHS09a]. We will evaluate these fourteen algorithms using various techniques discussed in chapter 8 and we will discuss the merits and drawbacks of these different methods. The 180 songs from the Beatles chord transcription collection described in chapter 6 are used as the test set for all the evaluations. This is a subset of the collection of annotated files used in the MIREX09 chord detection evaluation which also included songs from another collection by Mauch et al [MCD$^+$09]. We have chosen to use the Beatles alone firstly because at the time of writing we do not have access to the audio for the others and secondly because the Beatles collection are our own transcriptions which have been detailed in chapter 6 and are thus most relevant to the work in this thesis.

For results that are calculated for the whole collection we will present weighted average values as used in the official MIREX09 evaluations [mira] i.e. we sum the τ_{C} values of all songs and normalise by the total duration τ_{T} of the collection. This means that each song is weighted by its duration in the total calculation, effectively treating the whole collection as one long audio example.

9.1 Summary of the MIREX09 Entries

The algorithms we will evaluate in this chapter are the entrants to the MIREX09 chord detection evaluation plus two other algorithms by the author.

Chris Harte: 'ch_aes', 'ch_hcdf' and 'ch_hcdfa'

The three algorithms by the author have already been discussed in detail in chapter 3. The 'ch_aes' algorithm is the basic frame-based system using a tuned chromagram to provide feature vectors; 'ch_hcdf' and 'ch_hcdfa' use the same recognition technique but pre-segment the chromagram using the HCDF with simple peak-picking and enhanced thresholded peak picking respectively. The algorithms are designed to recognise chord types '`maj`', '`min`', '`aug`', '`dim`' and '`N`'.

Dan Ellis: 'de'

Dan Ellis's algorithm [Ell09] is a pre-trained system that uses Gaussian models for each chord class based on beat synchronous chroma features. The actual chord recognition part of the system is based on a Hidden Markov Model (HMM) [Rab89] using the per-chord Gaussians to calculate observation likelihoods as originally described in [SE03]. The system was trained using our Beatles transcription collection and originally submitted to the MIREX08 evaluation. Subsequently, many of the local audio files used in that work were found to be badly aligned with the transcriptions (many were taken from alternative masters so time stretching was also an issue). After correctly aligning the audio and the transcriptions, a 20% performance improvement was achieved so the updated system was submitted again to MIREX09. The algorithm is designed to recognise '`maj`' and '`min`' chords plus the '`N`' chord.

Khadkevich and Omologo: 'ko1' and 'ko2'

Khadkevich and Omologo's algorithms [KO09a] also use a HMM with chroma feature vectors. Their system was also trained using our Beatles

transcription collection. Their training process differs from Ellis's in that chord segmentation is based on the ground truth annotations rather than beat synchronous features. The algorithms are designed to recognise '`maj`' and '`min`' chords plus the '`N`' chord.

Mauch and Dixon: 'md'

Mauch and Dixon's algorithm [MND09a] combines several different elements. The system uses beat synchronous chromagrams as a basic feature vector, generated automatically using Davies' beat tracking system [Dav07]. An automatic structural segmentation is used to identify repeated sections at the verse-chorus level. The reasoning behind this is that using information about the structure of the song can help in recognising chord sequences because the same progressions are often repeated several times. Separate instances of a repeated segment in a song can be combined and a single chord progression can be inferred from the combination. The system uses a dynamic bayesian network [Mur02] which models metric position, chords and bass pitch in order to infer the most probable chord sequence from the chroma features. Mauch has continued to develop this system, further improving its performance since MIREX09, in [Mau10]. The algorithm is designed to recognise the chordtypes `N`, `maj`, `min`, `dim`, `7`, `maj6` and `maj7`.

Oudre, Grenier, Fevotte : 'ogf'

Oudre, Grenier and Fevotte's system is a frame-based signal processing approach using template matching. Their basic feature vector is also the chromagram and in their system, they use chord templates that reflect the harmonic content of the chroma vectors rather than simple binary templates of the kind used in our three algorithms. The chord recognition is achieved by minimizing a measure of fit between the template and the given chroma vector. The system they submitted to MIREX09 uses the Kullback-Liebler divergence as the measure of fit with templates containing chord notes plus single harmonics. In a similar way to our basic algorithm, they also employ low-pass filtering and median filtering

to smooth the estimated chord sequence. The 'ogf' algorithm can detect major, minor and `N` chords.

Papadopulous and Peeters: 'pp'

Papadopulous and Peeters' system is another approach that uses beat-synchronous chroma features and HMMs. Their system detects both chord sequence and the downbeats of bars simultaneously using one to help in the estimation of the other. To deal with songs in different meters, they have two time signature models, one dealing with $\frac{4}{4}$ time and the other with $\frac{3}{4}$. Their system generates a sequence of observation vectors defined by the observation probablities. Given this set of observation vectors, the algorithm chooses the meter that fits best and estimates the most likely chord sequence and set of downbeat positions using a maximum likelihood calculation.

Pauwels, Varewyck and Martens: 'pvm1' and 'pvm2'

Pauwels et al also use chroma features in their system. However, they calculate the chroma features using a technique proposed by Varewyck [PVM09] which maximally couples higher harmonics to the fundamental frequency thus reducing the proportion of non chord tone harmonics in the chromagram. They submitted two algorithms with alternative backends. Their algorithm 'pvm1' simultaneously recognises chord sequences and key context finding the best match for a key and chord label sequence with an a priori tonal model. Dynamic programming is then used to determine the optimal path of key-chord pairs. The backend of their second algorithm 'pvm2' is based on their earlier work [VPM08] using the cosine similarity between chroma vectors and a set of binary chord templates to determine chord matches in a similar way to our own chord recogniser as discussed in chapter 3.

Rocher, Robine, Hanna and Strandh: 'rrhs1', 'rrhs2' and 'rrhs3'

Rocher et al. [RRHS09a] use chroma features as the front end to their system. They then use chord templates to determine possible candidate

chord-key pairs for the extracted chroma frames. The candidate pairs for each frame are then linked to the candidates in the next frame forming a directed acyclic graph. Edges in the graph correspond to chord transitions. A cost function based on Lerdahl's chord distance [Ler01] is calculated for each edge and dynamic programming is subsequently employed to find the path with the minimum cost in order to produce the estimated chord sequence as described in [RRHS09b].

Algorithm 'rrhs1' uses the dynamic programming system but limits the chord vocabulary to major and minor. Algorithm 'rrhs2' uses the same front end but chooses major or minor triads based on frame correlation with the chord templates instead of the dynamic programming algorithm. The third algorithm, 'rrhs3' uses the same dynamic programming system as 'rrhs1' but allows for a much larger vocabulary of twelve chordtypes: '`N`', '`maj`', '`sus4`', '`aug`', '`7`', '`sus2`', '`maj(9)`', '`min`', '`min7`', '`dim`', '`min(9)`' and '`dim7`'.

9.2 Comparison of chord symbol recall values

Table 9.1 shows the weighted chord symbol recall values for the fourteen different recognition algorithms for seven different chord symbol recall methods. The recall methods that have been used here are MIREX08 mapping $\mathcal{R}'_{\mathbf{mx}_{08}}$, MIREX09 mapping $\mathcal{R}'_{\mathbf{mx}_{09}}$, cardinality-3 pcset matching $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3}$, cardinality-2 pcset matching $\mathcal{R}'_{\overrightarrow{\mathbf{p}},2}$, cardinality-3 pcset dictionary matching $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_0}$ where dictionary $D_0 = \{\texttt{maj}, \texttt{min}\}$, cardinality-3 pcset dictionary matching $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_1}$ with dictionary $D_1 = \{\texttt{N}, \texttt{maj}, \texttt{min}\}$ and finally cardinality-3 pcset dictionary matching $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_2}$ with dictionary $D_2 = \{\texttt{N}, \texttt{maj}, \texttt{min}, \texttt{aug}, \texttt{dim}\}$. At the bottom of the table, the proportion of the total duration of the annotated collection that has been included in the evaluation is also given. For the first four recall methods, the whole test set is included in the evaluation. For the dictionary matching recall calculations, unmatchable chord types are excluded from the evaluations so for dictionary D_0, 92% of the total collection duration is evaluated against and for D_1 and D_2 this rises to 96% and 98% respectively. The parameters of the five different pcset matching evaluations were chosen

Table 9.1: Weighted chord recall values for the Beatles collection.

Algorithm	$\mathcal{R}'_{\mathbf{mx}_{08}}$	$\mathcal{R}'_{\mathbf{mx}_{09}}$	$\mathcal{R}'_{\overrightarrow{\mathbf{P}},3}$	$\mathcal{R}'_{\overrightarrow{\mathbf{P}},2}$	$\mathcal{R}'_{\overrightarrow{\mathbf{P}},3,D_0}$	$\mathcal{R}'_{\overrightarrow{\mathbf{P}},3,D_1}$	$\mathcal{R}'_{\overrightarrow{\mathbf{P}},3,D_2}$
ch_aes	0.58	0.58	0.56	0.57	0.61	0.59	0.58
ch_hcdf	0.58	0.58	0.56	0.57	0.60	0.58	0.58
ch_hcdfa	0.59	0.59	0.58	0.59	0.62	0.60	0.59
de	**0.72**	**0.72**	**0.71**	**0.71**	**0.75**	**0.74**	**0.72**
ko1	0.70	0.70	0.69	0.69	0.73	0.72	0.71
ko2	0.71	0.71	0.70	0.70	0.74	0.73	**0.72**
md	0.69	0.69	0.67	0.68	0.73	0.70	0.69
ogf	0.69	0.69	0.67	0.68	0.71	0.70	0.69
pp	0.67	0.67	0.66	0.67	0.72	0.69	0.68
pvm1	0.67	0.67	0.66	0.66	0.71	0.68	0.67
pvm2	0.63	0.63	0.62	0.62	0.67	0.64	0.63
rrhs1	0.66	0.66	0.65	0.66	0.70	0.68	0.67
rrhs2	0.60	0.60	0.59	0.59	0.63	0.61	0.60
rrhs3	0.65	0.57	0.51	0.52	0.55	0.53	0.52
included %	100	100	100	100	92	96	98

because we consider them to be the fairest tests for comparing this particular set of algorithms based on their different chord vocabularies. The MIREX mapping recall methods are shown to allow comparison with our pcset evaluation results.

All algorithms score between 50% and 76% for the seven recall methods. In most cases the different recall methods produce fairly similar values for each algorithm with all but one case (rrhs3) having a range of less than 10% for the different evaluations. The results from table 9.1 are shown graphically in figures 9.1 and 9.2. We see that the recall values for the 'ch_aes' algorithm are slightly better in some cases than those for the 'ch_hcdf' algorithm although not by a large amount. The 'ch_hcdfa' algorithm however performs better than both.

Looking at all fourteen algorithms, we see that the Ellis algorithm 'de' has the best recall performance for all evaluation methods. The ranking stays constant for all recall methods apart from some slight changes in the order for $\mathcal{R}'_{\overrightarrow{\mathbf{P}},3,D_0}$ and the unusual values for algorithm 'rrhs3' which we will discuss later. Figure 9.2 shows the differences between the alternative

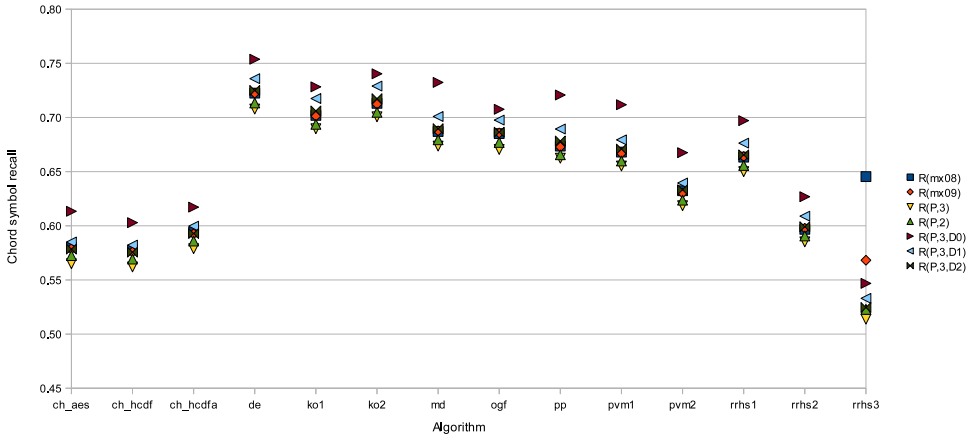


Figure 9.1: Plot showing weighted chord symbol recall values for each algorithm using seven different recall methods.

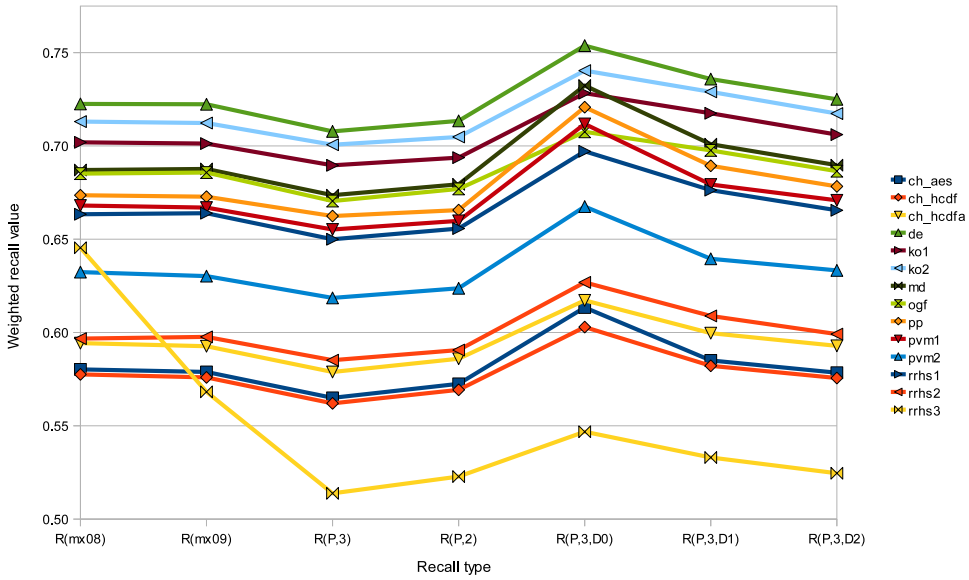


Figure 9.2: Line plot showing the weighted recall values for each algorithm against the seven different recall methods. In this plot we can see that the same ranking is maintained for all methods except for some small changes in $\mathcal{R}'_{\vec{\mathbf{p}},3,D_0}$ and the strange values for algorithm rrhs3 for the MIREX mappings.

recall calulations quite clearly. With the exception of 'rrhs3', the values for the MIREX mapping recall scores $\mathcal{R}'_{\mathbf{mx}_{08}}$ and $\mathcal{R}'_{\mathbf{mx}_{09}}$ stay roughly the same for each algorithm. This is as we might expect given that major and minor triads make up 78% of the whole collection and most of the algorithms have fairly small chord vocabularies.

The values for the pcset matching recall scores $\mathcal{R}'_{\overrightarrow{\mathbf{p}},2}$ and $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3}$ are lower than those for $\mathcal{R}'_{\mathbf{mx}_{08}}$ and $\mathcal{R}'_{\mathbf{mx}_{09}}$. This can be explained by the introduction of the 'unmatchable' category of chords which are not mapped to major or minor triads (as discussed in section 8.1.4). The MIREX mapping functions can cause false positives by mapping two different chords into the same triad category producing an inappropriate match. The unmatchable chords in the pcset matching functions will always be considered incorrect matches so the false positives are removed causing the overall scores to be lower than the MIREX recall scores.

Looking at the pcset based recall values, we can see that $\mathcal{R}'_{\overrightarrow{\mathbf{p}},2}$ is higher in all cases than $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3}$. We would expect this to be the case because $\mathcal{R}'_{\overrightarrow{\mathbf{p}},2}$ will evaluate any chords with the same root and third so major chords will be grouped with augmented chords and likewise minor with diminished.

The three dictionary-based recall evaluations $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_0}$, $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_1}$ and $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_2}$ have higher scores than $\mathcal{R}'_{\overrightarrow{\mathbf{p}},2}$ and $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3}$. The scores for $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_0}$ and $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_1}$ are also higher than $\mathcal{R}'_{\mathbf{mx}_{08}}$ and $\mathcal{R}'_{\mathbf{mx}_{09}}$ whereas $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_2}$ has similar scores to the two MIREX mapping functions. We would expect the dictionary-based scores to be higher than $\mathcal{R}'_{\overrightarrow{\mathbf{p}},2}$ and $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3}$ because unmatchable chord symbols are excluded when the dictionary is used. In the case of D_0, this means that 8% of the collection is excluded from the evaluation and we know that this 8% would definitely be incorrect matches for $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3}$. The scores for $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_1}$ are lower than $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_0}$ because D_1 includes an extra chordtype meaning than only 4% of the collection is then excluded as unmatchable. Likewise, $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_2}$ has lower values because it adds another two chordtypes taking the excluded percentage down to 2%. The rankings for $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_0}$ are slightly different to the other recall methods with algorithms 'md', 'pp', 'pvm1' and 'pvm2' all performing noticeably better when evaluated against D_0. In the case of 'pp' and the 'pvm' algorithms this can be explained because their chord vocabularies do not

include the 'N' chord therefore they can only lose out when it is added to dictionary D_1. Mauch and Dixon's 'md' algorithm on the other hand is capable of detecting 'N' so these results suggest that the detection accuracy of this algorithm is poor for this chordtype and this is confirmed in [Mau10].

As mentioned earlier, the 'rrhs3' algorithm by Rocher et al. has significantly higher values for $\mathcal{R}'_{\mathbf{mx}_{08}}$ and $\mathcal{R}'_{\mathbf{mx}_{09}}$ than it does for the five pcset based recall scores which follow the same trend as the other algorithms. Why should this algorithm have such strange results compared with all the others? Referring back to table 8.4, we can see that 'rrhs3' has a much larger chord vocabulary than the other thirteen algorithms in the study. For this reason, it is much more likely to suffer (or rather in this case gain) from the effects of inappropriate chord mappings causing false positives in the MIREX recall calculations. The other algorithms are, in general, not capable of producing symbols that will be mapped incorrectly so they will generate fewer false positives. The results support this explanation because the $\mathcal{R}'_{\mathbf{mx}_{08}}$ value is a lot higher than the $\mathcal{R}'_{\mathbf{mx}_{09}}$ value. This is to be expected given that the MIREX08 mapping converts most chordtypes to major causing more false positives than the MIREX09 mapping which is a little more evenly balanced (see section 8.1.3). These false positives are not present in the pcset matching recall scores because unmatchable chords are either treated as incorrect, in the case of $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3}$ and $\mathcal{R}'_{\overrightarrow{\mathbf{p}},2}$, or they are excluded from the evaluation where dictionary based matching is used.

Dictionary matching evaluation does not suffer from the false positives problem faced by the MIREX08 and 09 mapping approach. Unlike $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3}$ and $\mathcal{R}'_{\overrightarrow{\mathbf{p}},2}$, it excludes the unmatchable chord types from the calculation so results are not artificially lowered by chord symbols which we know cannot generate correct matches given the evaluation parameters. For this set of chord recognisers, dictionary D_0 is the most appropriate because it is the only common subset of all the chord vocabularies for all fourteen algorithms. Using the largest dictionary common to all algorithms is the fairest way to compare like with like so the optimal dictionary will always be defined by the least flexible algorithm. Because of this, we believe that

Table 9.2: Chord dictionaries for cardinality-6 test.

Dictionary	**Vocabulary**	
$D_{0.1}$	`X:maj`	
$D_{0.2}$	`X:min`	$+ D_{0.1}$
$D_{0.3}$	`X:7`	$+ D_{0.2}$
$D_{0.4}$	`N`	$+ D_{0.3}$
$D_{0.5}$	`X:min7`	$+ D_{0.4}$
$D_{0.6}$	`X:9`	$+ D_{0.5}$
$D_{0.7}$	`X:maj6`	$+ D_{0.6}$
$D_{0.8}$	`X:maj7`	$+ D_{0.7}$
$D_{0.9}$	`X:aug`	$+ D_{0.8}$
$D_{0.10}$	`X:min(*b3)`	$+ D_{0.9}$
$D_{0.11}$	`X:sus4`	$+ D_{0.10}$
$D_{0.12}$	`X:7(#9)`	$+ D_{0.11}$
$D_{0.13}$	`X:maj(9)`	$+ D_{0.12}$

the pcset dictionary based evaluation function $\mathcal{R}'_{\overrightarrow{\mathbf{P}},3,D_0}$ is the fairest measure to use for this particular set of algorithms. For future studies where several algorithms are compared such as further MIREX evaluations, an alternative dictionary may be more approriate as chord recognition algorithms become more sophisticated.

9.2.1 Effects of dictionary size

To see what effect dictionary size has on the recall results we define a set of dictionaries increasing in size from one chord to thirteen using the most common chordtypes in the Beatles collection in order of combined duration (as discussed in section 6.6.2). Table 9.2 details the chordtypes in each of these thirteen dictionaries $D_{0.1}$ to $D_{0.13}$. We now calculate the weighted recall values for each algorithm at cardinality-6 i.e. $\mathcal{R}'_{\overrightarrow{\mathbf{P}},6,D_a}$ so that we may determine how good the algorithms are at identifying the chords in each dictionary. At cardinality-6, no non-dictionary chord will ever be considered a correct match with a dictionary chord.

The results of this set of evaluations can be seen in figure 9.3 and table 9.3. We see a general trend for all algorithms that as the dictionary size increases, the recall value decreases in proportion with the number of

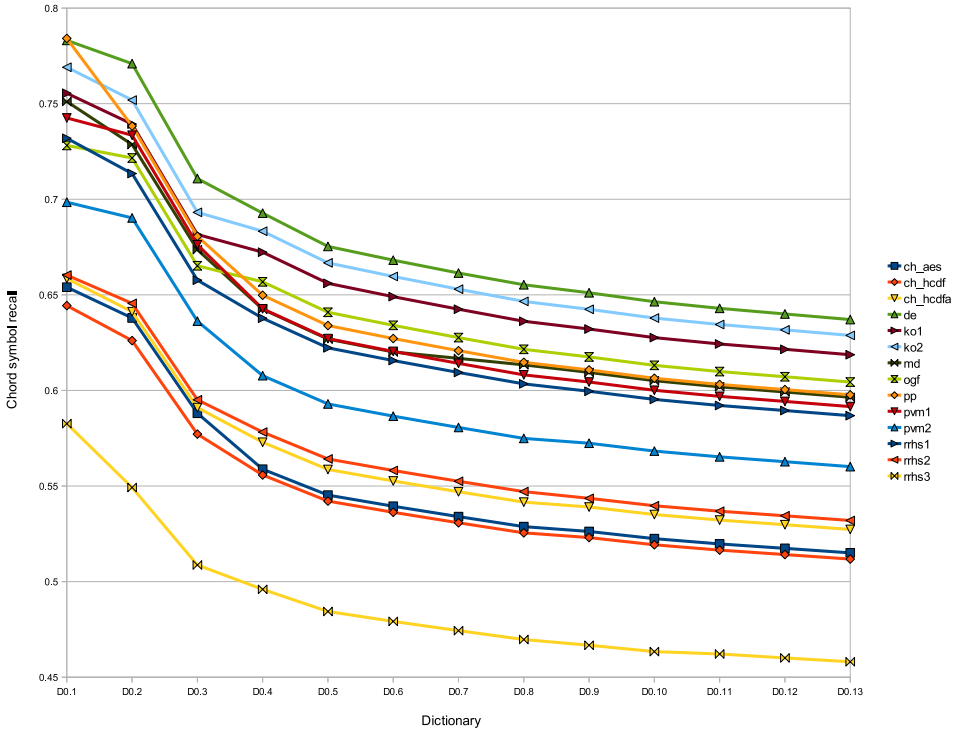


Figure 9.3: Line plot showing the cardinality-6 recall values for each algorithm for dictionaries $D_{0.1}$ to $D_{0.13}$.

chords being reincluded in the evaluation (the percentages of the collection that are included in each evaluation are plotted in figure 9.4 for comparison). This is as we would expect because for each new chord added to the dictionary, if the algorithm does not include that chord in its vocabulary then it will only have the effect of making $\tau_{\mathcal{E}}$ larger with no change in τ_{C} (see equation 8.33). It is interesting to look at the changes of fortune for various algorithms in the smaller dictionary evaluations. For dictionary $D_{0.1}$ (the single '`X:maj`' triad), Papadopoulos and Peeters' algorithm 'pp' has the top score, narrowly outperforming 'de'. However, when the '`X:min`' triad type is added in the $D_{0.2}$ evaluation, 'pp' drops very quickly compared to the other algorithms and changes rank from 1st to 4th place. Algorithms 'md' and 'rrhs1' also fall more steeply than the other algorithms both dropping one rank. This suggests that these algorithms are better at detecting the '`maj`' chordtype than they are at detecting the '`min`' type, considerably more so in the case of 'pp'.

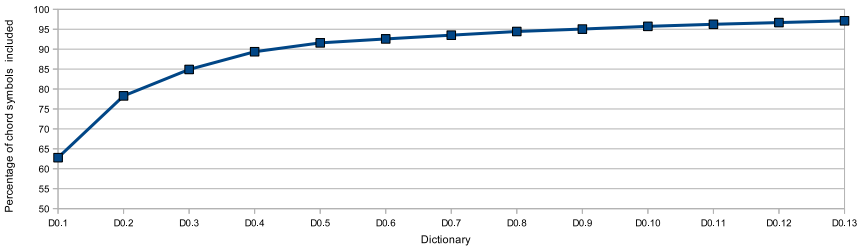


Figure 9.4: Line plot showing the proportion of the collection included in the recall evaluation for dictionaries $D_{0.1}$ to $D_{0.13}$.

Introducing the the '`X:7`' chordtype in $D_{0.3}$ makes all the algorithms scores fall steeply. This is because most of the algorithms are unable to detect the '`X:7`' type and it is the third most common chord type in the collection accounting for 6.63% of the whole duration. Strangely, the Mauch and Dixon algorithm 'md' falls at the same rate as all the others for $D_{0.3}$ despite the fact that the '`X:7`' type is included in its vocabulary. This suggests that 'md' is much better at major and minor triad detection than it is at detecting seventh chords.

The inclusion of the '`N`' chord in dictionary $D_{0.4}$ again sees some algorithms drop more steeply than others. In the case of 'pp' and the two 'pvm' algorithms, this is because '`N`' is not in their vocabulary. For all the other algorithms, it would seem that their detection performance for non-chordal material is significantly worse than that for the major and minor triads. After $D_{0.5}$, the recall performances of the algorithms all follow approximately the same trend, reducing at roughly the same rate as the percentage of included chords rises.

9.2.2 Single chordtype analysis

To prove whether our explanations of the findings for dictionaries $D_{0.1}$ to $D_{0.4}$ were borne out, we investigated the recall performance for several individual chord types. Table 9.4 shows the individual cardinality-6 pcset chord recall values[1] for chord types `X:maj`, `X:min`, `X:7`, `N`, `X:min7`, `X:aug`

[1]Cardinality-6 is the highest cardinality of any chord in the transcription collection so a chord will not match anythoung other than itself in this recall evaluation.

Table 9.3: Cardinality-6 pcset recall values for dictionaries $D_{0.1}$ to $D_{0.13}$ shown with the proportion of the collection that is included in each evaluation.

Algorithm	$D_{0.1}$	$D_{0.2}$	$D_{0.3}$	$D_{0.4}$	$D_{0.5}$	$D_{0.6}$	$D_{0.7}$	$D_{0.8}$	$D_{0.9}$	$D_{0.10}$	$D_{0.11}$	$D_{0.12}$	$D_{0.13}$
ch_aes	0.654	0.638	0.588	0.559	0.545	0.539	0.534	0.529	0.526	0.522	0.520	0.517	0.515
ch_hcdf	0.644	0.626	0.577	0.556	0.542	0.536	0.531	0.526	0.523	0.519	0.516	0.514	0.512
ch_hcdfa	0.659	0.641	0.591	0.573	0.559	0.553	0.547	0.542	0.539	0.535	0.532	0.530	0.527
de	0.783	**0.771**	**0.711**	**0.693**	**0.675**	**0.668**	**0.661**	**0.655**	**0.651**	**0.646**	**0.643**	**0.640**	**0.637**
ko1	0.755	0.739	0.682	0.672	0.656	0.649	0.642	0.636	0.632	0.628	0.624	0.622	0.619
ko2	0.769	0.752	0.693	0.683	0.667	0.660	0.653	0.647	0.642	0.638	0.634	0.632	0.629
md	0.751	0.729	0.674	0.642	0.627	0.620	0.617	0.613	0.609	0.605	0.602	0.599	0.596
ogf	0.728	0.722	0.665	0.657	0.641	0.634	0.628	0.622	0.618	0.613	0.610	0.607	0.604
pp	**0.784**	0.738	0.681	0.650	0.634	0.627	0.621	0.615	0.611	0.606	0.603	0.600	0.598
pvm1	0.743	0.734	0.676	0.643	0.627	0.620	0.614	0.608	0.604	0.600	0.597	0.594	0.592
pvm2	0.698	0.690	0.636	0.608	0.593	0.587	0.581	0.575	0.572	0.568	0.565	0.563	0.560
rrhs1	0.732	0.713	0.658	0.638	0.622	0.616	0.609	0.603	0.600	0.595	0.592	0.589	0.587
rrhs2	0.660	0.646	0.595	0.578	0.564	0.558	0.552	0.547	0.544	0.540	0.537	0.534	0.532
rrhs3	0.583	0.549	0.509	0.496	0.484	0.479	0.474	0.470	0.467	0.463	0.462	0.460	0.458
Included %	62.78	78.27	84.91	89.35	91.57	92.56	93.50	94.43	95.04	95.72	96.23	96.66	97.10

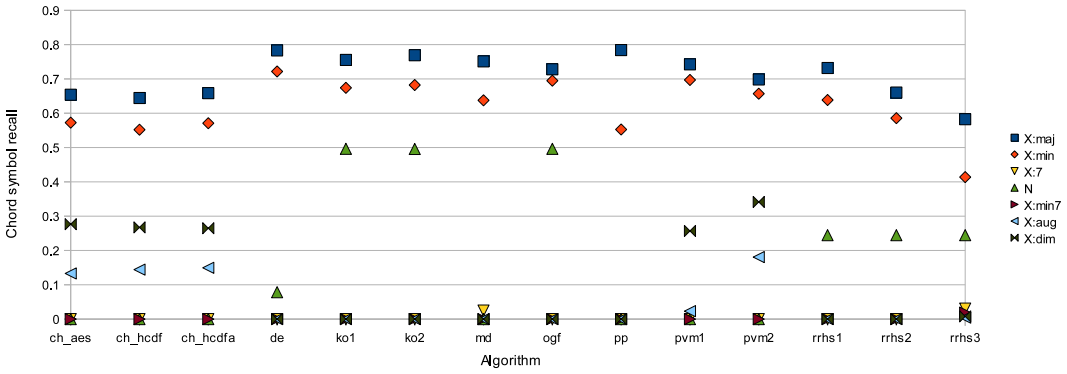


Figure 9.5: Plot showing the recall values for each algorithm for individual chord types '`X:maj`', '`X:min`', '`X:7`', '`N`', '`X:min7`', '`X:aug`' and '`X:dim`'.

Table 9.4: Cardinality-6 recall values for individual chordtypes. The chordtypes are ordered by their percentage of the total collection duration.

Algorithm	`X:maj`	`X:min`	`X:7`	`N`	`X:min7`	`X:aug`	`X:dim`
ch_aes	0.65	0.57	0	0	0	0.13	0.28
ch_hcdf	0.64	0.55	0	0	0	0.14	0.27
ch_hcdfa	0.66	0.57	0	0	0	0.15	0.26
de	**0.78**	**0.72**	0	0.08	0	0	0
ko1	0.76	0.67	0	**0.50**	0	0	0
ko2	0.77	0.68	0	**0.50**	0	0	0
md	0.75	0.64	0.02	0	0	0	0
ogf	0.73	0.69	0	**0.50**	0	0	0
pp	**0.78**	0.55	0	0	0	0	0
pvm1	0.74	0.70	0	0	0	0.02	0.26
pvm2	0.70	0.66	0	0	0	**0.18**	**0.34**
rrhs1	0.73	0.64	0	0.24	0	0	0
rrhs2	0.66	0.59	0	0.24	0	0	0
rrhs3	0.58	0.41	0.03	0.24	0.02	0	0.01
Included %	62.8	15.5	6.6	4.4	2.2	0.6	0.4

and `X:dim` i.e. $\mathcal{R}'_{\overrightarrow{\mathbf{p}},6,D_a}$ where D_a is an individual chord in each case. These values are shown graphically in figure 9.5.

The recall value for '`X:maj`' is highest for all algorithms. In most cases, the recall for '`X:min`' is slightly less but, as suggested by the results in the previous section, the '`X:min`' recall for 'pp' is much lower than their score

'X:maj'.

All of the algorithms include 'X:maj' and 'X:min' chord types in their vocabularies. However, 'X:7' is only included in the vocabularies of 'md' and 'rrhs3' so the recall values for this chord type are zero for most of the algorithms. The 'X:7' recall scores for 'md' and 'rrhs3' are very low compared to their performance for 'X:maj' and 'X:min' as suggested by the previous section's results.

For the 'N' chord recall, the results are interesting. All the algorithms apart from 'pp' and the two 'pvm' entries are supposed to be able to identify 'N chords. However, only the 'ko' and 'ogf' algorithms can do so with a 50% level of accuracy. The recalls for the three 'rrhs' algorithms are lower, all scoring 24% and 'de' is lower still at 8%. Surprisingly, despite the fact they are meant to be able to recognise the 'N' chord, 'md' and the three 'ch' algorithms all score zero for this chord type.

Only 'rrhs3' is designed to recognise the 'X:min7' type so we would expect zeros for all other algorithms. The 'X:min7' recall score for 'rrhs3' is low compared with its performance for the major and minor triad types with a score of 2%.

The 'ch' and 'pvm' algorithms are able to recognise the 'X:aug' triad type but the scores in all cases are all below 18%. The 'ch' and 'pvm' algorithms are slightly better at recognising the 'X:dim' however, with scores rising to around 30%. Algorithms 'md' and 'rrhs3' are also quoted as recognising 'X:dim' but score very poorly for this type.

9.2.3 Effects of evaluation cardinality

Our final set of results for chord symbol recall look at the effects of using different cardinalities for pcset dictionary based evaluations. Table 9.5 shows the results for the 'X:maj' chordtype for cardinalities 1 to 6; these results are shown graphically in figure 9.6. At cardinality 1, all tonal chordtypes (i.e. all chords except for 'N' which has cardinality 0) are included in the recall calculation so this is effectively a test of root identification accuracy. Since this test treats all chordtypes other than 'N' as correct matches, the total percentage of the collection that is included is 95.5%.

Table 9.5: Chord symbol recall values for chordtype '`X:maj`' with evaluation cardinality values 1 to 6.

Algorithm	$M = 1$	$M = 2$	$M = 3$	$M = 4$	$M = 5$	$M = 6$
ch_aes	0.66	0.63	0.63	0.65	0.65	0.65
ch_hcdf	0.65	0.63	0.62	0.64	0.64	0.64
ch_hcdfa	0.66	0.64	0.64	0.66	0.66	0.66
de	**0.78**	0.76	**0.77**	**0.78**	**0.78**	**0.78**
ko1	0.75	0.74	0.74	0.76	0.76	0.76
ko2	0.75	0.75	0.76	0.77	0.77	0.77
md	0.76	**0.77**	0.77	0.75	0.75	0.75
ogf	0.76	0.72	0.72	0.73	0.73	0.73
pp	0.74	0.76	**0.77**	**0.78**	**0.78**	**0.78**
pvm1	0.73	0.72	0.72	0.74	0.74	0.74
pvm2	0.71	0.68	0.68	0.70	0.70	0.70
rrhs1	0.75	0.71	0.72	0.73	0.73	0.73
rrhs2	0.68	0.64	0.64	0.66	0.66	0.66
rrhs3	0.75	0.59	0.58	0.58	0.58	0.58
Included%	95.5	74.1	73.4	62.8	62.8	62.8

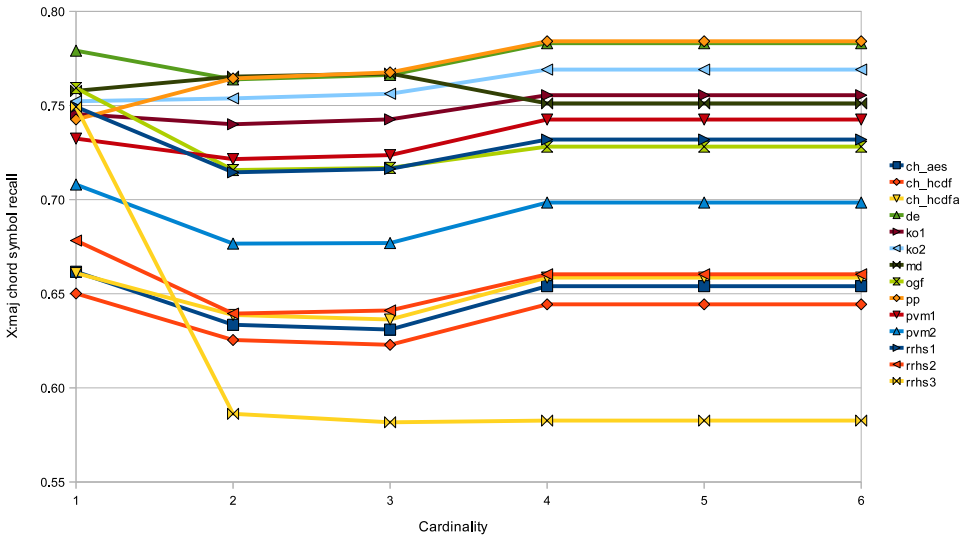


Figure 9.6: Line plot showing the recall values for each algorithm for chord type '`X:maj`' against evaluation cardinality.

As we can see clearly in the figure, the 'rrhs3' algorithm has a much higher score for this cardinality than it does for the other cardinality values, again

supporting our explanation of false positives causing the strange values it produced for $\mathcal{R}'_{\mathbf{mx}_{08}}$ and $\mathcal{R}'_{\mathbf{mx}_{09}}$ which we saw in table 9.1.

Increasing the cardinality to 2 removes all chordtypes that do not have a major third interval '`(1,3)`' as their first two elements. Most algorithms perform better at root identification than cardinality 2 but 'md', 'ko2', and 'pp' improve improve as the cardinality increases. This is because these algorithms are much better at detecting major family chords than they are at other families.

Results are fairly static for most algorithms between cardinality-2 and 3. This can be explained because the only major change between these two cardinalities is the exclusion of augmented chords at cardinality-3 and most of the algorithms cannot recognise this chord type. The 'ch' algorithms all drop in performance slightly here because '`X:aug`' is in their vocabulary but their recall perfomance for that type is poor as seen in section 9.2.2.

Moving from cardinality-3 to cardinality-4 produces a marked improvement in the scores for most algorithms. This is because at this stage all chords which are not exactly equal to '`X:maj`' are now excluded from the matching so chordtypes such as '`X:maj7`' and '`X:7`' are no longer considered matches with '`X:maj`'. The major exception to this trend is algorithm 'md' which drops in performance at this stage. This is because it is capable of producing '`X:7`' symbols and these would have added to its cardinality 3 score but their effect is removed at cardinality 4. For cardinality 5 and 6 the values all stay the same for the chord types '`X:maj`' because it is a triad chord and can therefore not be considered equal to anything but itself for cardinality 4 and above as discussed in section 5.4. The effect of excluding all tetrad chords and above at cardinality 4 is more pronounced than the effect of excluding augmented chordtypes at cardinality 3 simply because there are many more tetrads in the collection than augmented chords (as shown in section 6.6.2).

For comparison we also tested the effect of different cardinality values on the recall for chordtype '`X:min`' the values for which are shown in table 9.6 and figure 9.7. The results for cardinality 1 are exactly the same as for the '`X:maj`' results as we would expect but at cardinality 2 all

Table 9.6: Chord symbol recall values for chordtype '`X:min`' with evaluation cardinality values 1 to 6.

Algorithm	$M = 1$	$M = 2$	$M = 3$	$M = 4$	$M = 5$	$M = 6$
ch_aes	0.66	0.53	0.54	0.57	0.57	0.57
ch_hcdf	0.65	0.51	0.52	0.55	0.55	0.55
ch_hcdfa	0.66	0.53	0.54	0.57	0.57	0.57
de	**0.78**	**0.68**	**0.70**	**0.72**	**0.72**	**0.72**
ko1	0.75	0.64	0.67	0.67	0.67	0.67
ko2	0.75	0.64	0.68	0.68	0.68	0.68
md	0.76	0.57	0.59	0.64	0.64	0.64
ogf	0.76	0.64	0.67	0.69	0.69	0.69
pp	0.74	0.50	0.53	0.55	0.55	0.55
pvm1	0.73	0.65	0.67	0.70	0.70	0.70
pvm2	0.71	0.62	0.63	0.66	0.66	0.66
rrhs1	0.75	0.59	0.62	0.64	0.64	0.64
rrhs2	0.68	0.54	0.57	0.59	0.59	0.59
rrhs3	0.75	0.39	0.41	0.41	0.41	0.41
Included%	95.5	19.4	18.3	15.5	15.5	15.5

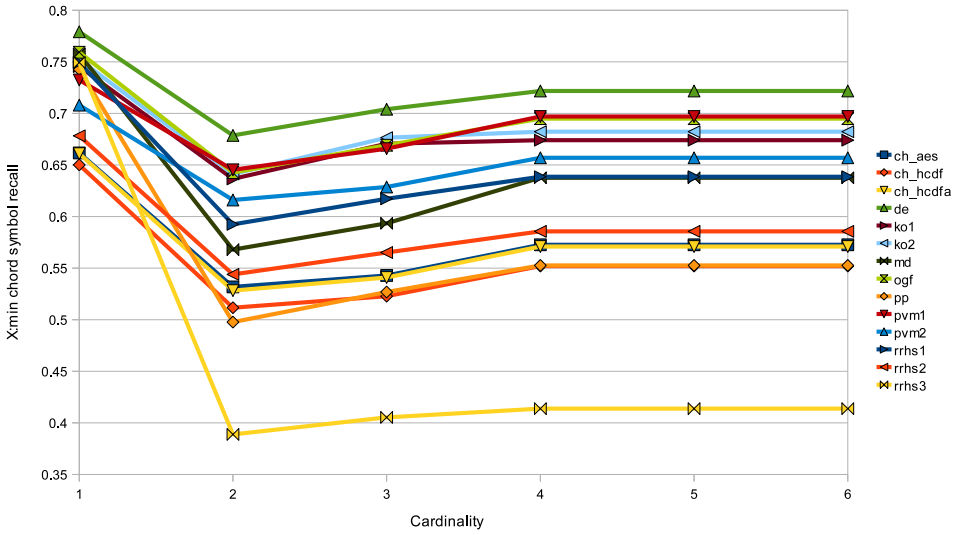


Figure 9.7: Line plot showing the recall values for each algorithm for chord type '`X:min`' against evaluation cardinality.

algorithm scores drop significantly. At cardinality 2, all chordtypes that have a minor third as their first interval are considered correct matches

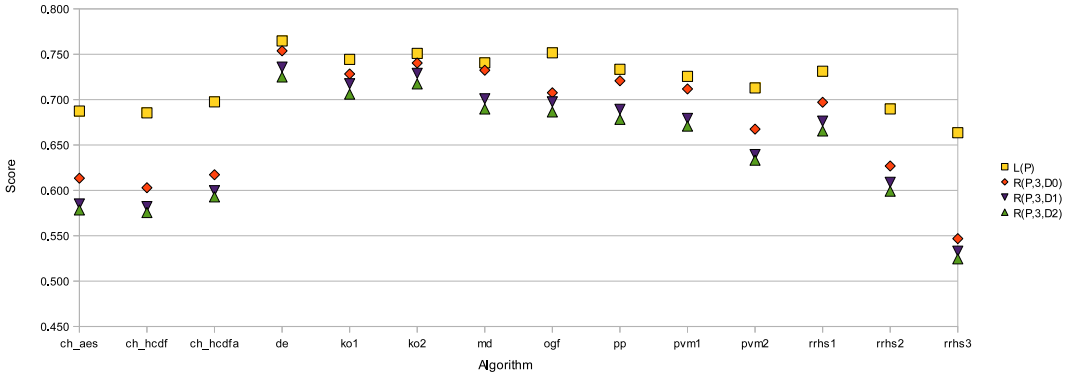


Figure 9.8: Plot showing likeness values $\mathbb{L}_{\widetilde{\mathbf{P}}}$ compared to the three sets of dictionary based recall values from table 9.1

so chordtypes such as '`X:dim`' will be matched with '`X:min`'. The two algorithms that drop the most are 'pp' and 'rrhs3' which, again, is in line with what we saw in the results from section 9.2.2. The results for 'pp' are interesting because it is the top ranking algorithm in the '`X:maj`' test but is one of the lowest ranking in this test.

All algorithms improve as the cardinality rises from 2 to 3 reflecting the removal of diminished type chords at cardinality 3. The 'ch' and 'pvm' algorithms plus 'rrhs3' see less improvement than the others at this stage because '`X:dim`' is part of their vocabulary. All algorithms improve again at cardinality 4 and above because of the removal of all tetrads and above at that stage.

9.3 Chord Likeness

The leftmost column of numbers in table 9.7 are the results for the likeness measure $\mathbb{L}_{\widetilde{\mathbf{P}}}$ between the algorithm outputs and the hand annotated files. These values are shown in figure 9.8 compared with the recall values $\mathcal{R}'_{\overrightarrow{\mathbf{P}},3,D_0}$, $\mathcal{R}'_{\overrightarrow{\mathbf{P}},3,D_1}$ and $\mathcal{R}'_{\overrightarrow{\mathbf{P}},3,D_2}$. It has been suggested by some contributors to the MIREX wiki [mirb] that this kind of measure be used in place of the recall measure for grading algorithms. The chord likeness measure gives higher scores than the stricter recall measures for all algorithms. However, it seems to have a greater effect on the algorithms that scored badly for

Table 9.7: Weighted values for chord likeness $\mathbb{L}_{\widetilde{\mathbf{p}}}$, segmentation quality $\mathcal{Q}$ and combined recall and segmentation F-measure $\mathcal{F}_{\overrightarrow{\mathbf{p}},3,D_0}$ for each algorithm across the whole Beatles collection. The F-measure is calculated using the segmentation quality $\mathcal{Q}$ and dictionary based recall measure $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_0}$ which is also shown in the table for reference.

Algorithm	$\mathbb{L}_{\widetilde{\mathbf{p}}}$	$\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_0}$	$\mathcal{Q}$	$\mathcal{F}_{\overrightarrow{\mathbf{p}},3,D_0}$
ch_aes	0.687	0.613	0.612	0.613
ch_hcdf	0.685	0.603	0.680	0.639
ch_hcdfa	0.698	0.617	0.725	0.667
de	**0.765**	**0.754**	0.831	**0.791**
ko1	0.744	0.728	0.777	0.752
ko2	0.751	0.740	0.781	0.760
md	0.741	0.732	0.806	0.768
ogf	0.752	0.708	0.757	0.731
pp	0.733	0.721	**0.832**	0.773
pvm1	0.726	0.712	0.805	0.756
pvm2	0.713	0.668	0.694	0.680
rrhs1	0.731	0.697	0.788	0.740
rrhs2	0.690	0.627	0.583	0.604
rrhs3	0.664	0.547	0.492	0.518

recall than those that scored well and thus compresses the range of values across the different algorithms.

9.4 Chord segmentation quality and F-measure

In chapter 8 we introduced the segmentation quality measure $\mathcal{Q}$ and the combined segmentation and recall F-measure $\mathcal{F}$. Figure 9.9 shows the results for $\mathcal{Q}$ compared with recall $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_0}$ and the combined measure $\mathcal{F}_{\overrightarrow{\mathbf{p}},3,D_0}$; the values for these are shown in table 9.7. We can see that the scores for $\mathcal{Q}$ are generally higher than the recall score for all algorithms except for 'rrhs2' and 'rrhs3'. In most cases, the segmentation quality values are better with algorithms 'ch_hcdfa', 'pp', 'pvm1' and 'rrhs1' all scoring much higher than their recall.

In the case of the three 'ch' algorithms, it is interesting to look at the difference between recall and segmentation scores. The 'ch_aes' algorithm

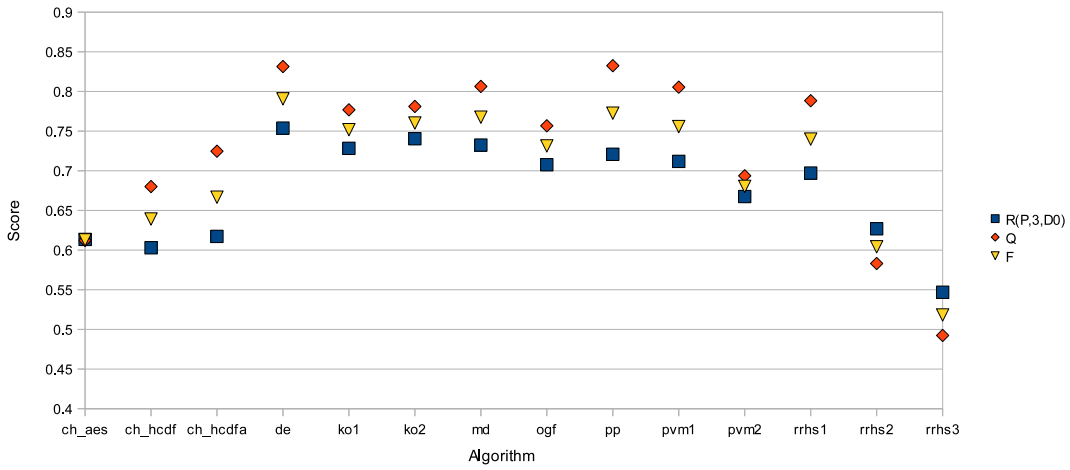


Figure 9.9: Plot showing segmentation measure $\mathcal{Q}$ and recall $\mathcal{R}'_{\vec{\mathbf{P}},3,D_0}$ compared with combined measure $\mathcal{F}_{\vec{\mathbf{P}},3,D_0}$ for each algorithm.

has no specific segmentation part except for application of a median filter to smooth the output chord symbol frames. This algorithm has almost exactly the same scores for recall and segmentation. The 'ch_hcdf' algorithm uses the same underlying chord recognition technique but employs the HCDF algorithm to segment the music before estimating chord symbols for each segment. The recall value actually falls for this algorithm compared to 'ch_aes' but the segmentation score is significantly better. The addition of the enhanced peak picking of the HCDF in algorithm 'ch_hcdfa' improves the segmentation quality significantly again and pulls the recall up slightly compared to the previous two algorithms. That the recall value falls for 'ch_hcdf' is an interesting result; this may be due to over-segmentation giving rise to some short segments with bad chord symbol estimates. With the better segmentation performance of the 'ch_hcdfa' algorithm, over-segmentation is reduced so the number of short incorrect segments falls helping the recall performance slightly.

Combined F-measure

Taking the harmonic mean of the segmentation quality $\mathcal{Q}$ and recall value $\mathcal{R}$ we can produce a single combined chord recognition score $\mathcal{F}$. In this

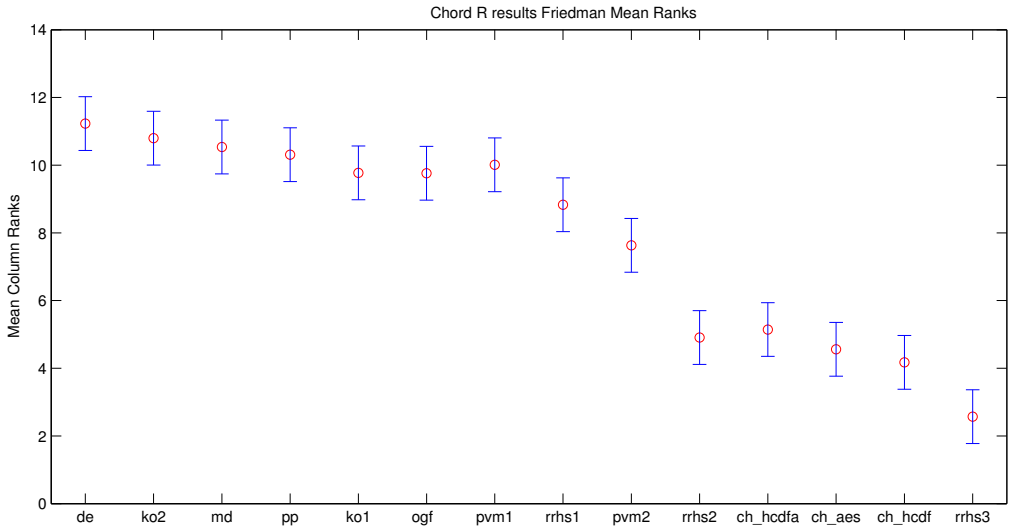


Figure 9.10: Plot showing the Friedman test mean ranks of the MIREX09 algorithms for chord recall $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_0}$.

case, the recall method we have chosen is a cardinality-3 dictionary-based pcset matching function $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_0}$ so we quote the f-measure with the same parameters $\mathcal{F}_{\overrightarrow{\mathbf{p}},3,D_0}$. Looking at figure 9.9 again, we can see that the combination of recall and segmentation actually has quite a large effect on the ranking of the algorithms. When considering recall alone, 'de' is in first place followed by 'ko2' then 'md' with 'ko1', 'pp', 'pvm1' and the 'ogf' algorithms close behind. Using the f-measure that includes the segmentation scores, the top performer is still Dan Ellis's 'de' algorithm but 'pp' jumps up to second place with 'md' again in third position. Lower down the table, the 'ch_hcdfa' algorithm benefits significantly from the inclusion of segmentation in a final combined score. Where it would be considered to have roughly the same performance if a little lower than 'rrhs1' on recall alone, it is clearly the better performing algorithm when f-measure is used.

9.4.1 Friedman test for statistical significance

Along with the weighted average values for chord symbol recall, statistical analysis of the recall results were also presented for MIREX09 [mira] using the Friedman rank test for significance [Fri40]. Mauch also chooses

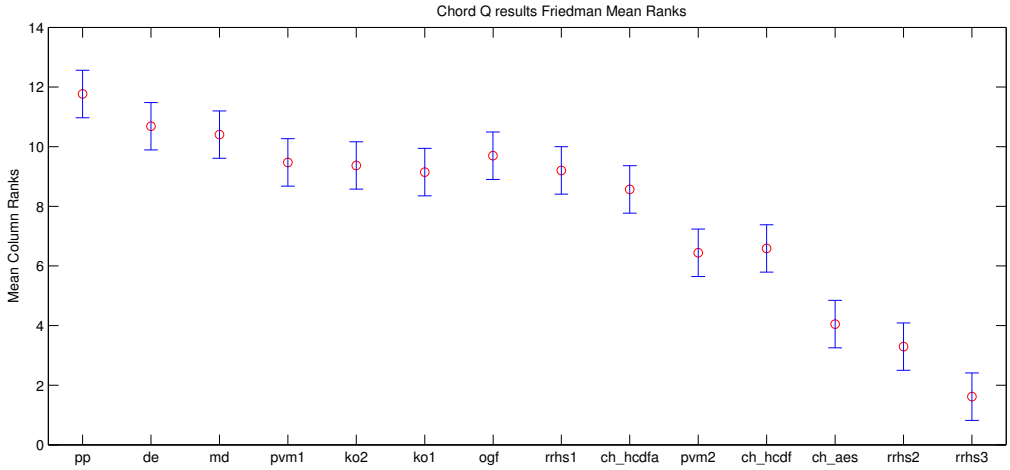


Figure 9.11: Plot showing the Friedman test mean ranks of the MIREX09 algorithms for chord segmentation $\mathcal{Q}$.

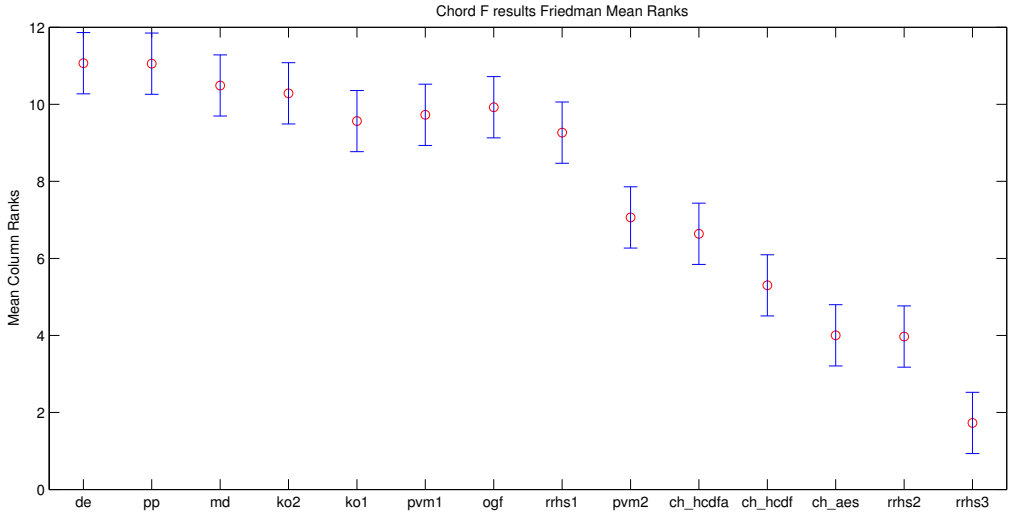


Figure 9.12: Plot showing the Friedman test mean ranks of the MIREX09 algorithms for chord f-measure $\mathcal{F}_{\overrightarrow{\mathbf{p}},3,D_0}$

to use this test to evaluate his results in [Mau10]. The Friedman test is a non-parametric test that calculates statistical significance of the difference between the algorithm results from their relative ranks for each song instead of their absolute output values. The test determines whether the mean ranking for algorithms differ significantly and because it works on ranks instead of absolute values it also has the advantage of compensating for varying difficulty of the songs in the test set.

Using the Friedman test on the song-wise results for chord symbol recall $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_0}$, chord segmentation $\mathcal{Q}$ and the combined chord f-measure $\mathcal{F}_{\overrightarrow{\mathbf{p}},3,D_0}$ we obtain the mean ranks shown in figures 9.10, 9.11 and 9.12. The ranking results for chord symbol recall put the algorithms in the same order as the weighted averages but the top performing algorithms are closer together. In fact, the test determines that for recall measure $\mathcal{R}'_{\overrightarrow{\mathbf{p}},3,D_0}$, the difference between the top algorithm 'de' and the next six algorithms 'ko2', 'md', 'ko1', 'pp', 'pvm1' and 'ogf' is not statistically significant. A similar situation is true for segmentation quality and f-measure. The results for the chord segmentation are similarly close showing that the difference between top performing algorithm 'pp' and the next two 'de' and 'md' is not statistically significant. The difference between second place 'de' and the next six 'md', 'pvm1', 'ko2', 'ko1', 'ogf', 'rrhs1' is not statistically significant. The results for the f-measure also show the difference between the top seven algorithms is not statisically significant.

For our recall evaluation, Dan Ellis' algorithm achieves the highest score. It is interesting that this is not the case for the official MIREX09 results [mira]. The MIREX09 evaluation test set included songs by Queen and Zweieck as well as the Beatles collection, whereas we used only the Beatles. Ellis used the Beatles transcriptions collection to train his recognition algorithm, so it may be that it is overfitting the training set and does not perform so well on other material.

For the authors' own algorithms, the friedman test shows that the difference between 'ch_hcdfa' and the other two 'ch_hcdf' and 'ch_aes' for recall is not statistically significant. However, the segmentation results show that 'ch_hcdf' is significantly better than 'ch_aes' and that 'ch_hcdfa' is significantly better than 'ch_hcdf'. The combined f-measure results show that 'ch_hcdfa' is not significantly better than 'ch_hcdf' but it is significantly better than 'ch_aes'.

9.5 Problem songs

The twelve Beatles albums comprise 180 songs and in this corpus there are a wide range of musical styles and timbres. Some songs are easier than

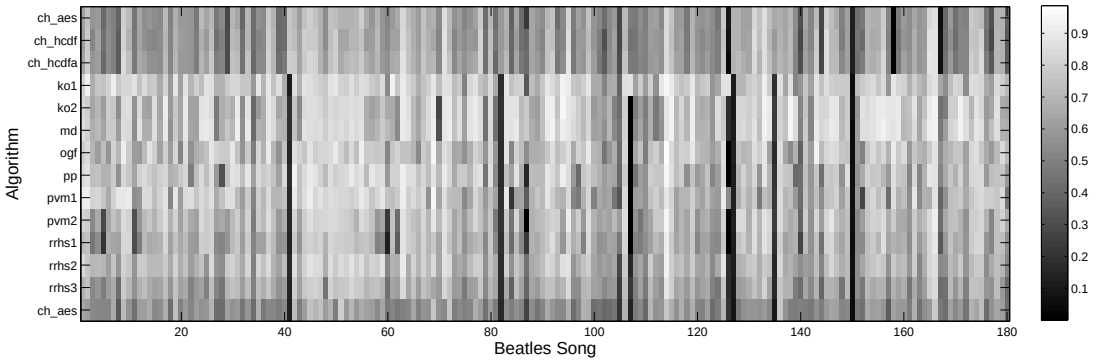


Figure 9.13: Image showing recall values as a greyscale colour intensity for each song on the x-axis for all fourteen algorithms. Dark areas show low scores. Dark vertical lines show songs that are problematic for more than one algorithm.

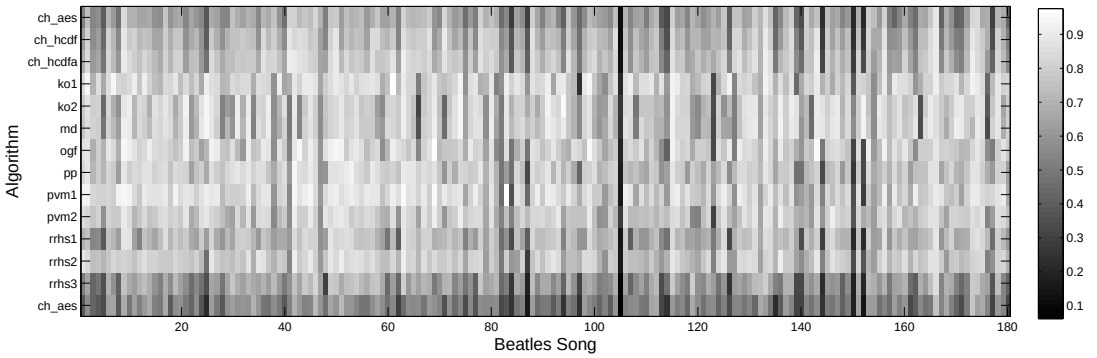


Figure 9.14: Image showing segmentation values as a greyscale colour intensity for each song on the x-axis for all fourteen algorithms. Dark areas show low scores. Dark vertical lines show songs that are problematic for more than one algorithm.

others for algorithms to perform chord recognition on as you would expect with a widely varied set of songs. However, there are a few songs in the collection for which all, or almost all the algorithms perform very badly. It is interesting to look at the individual recall and segmentation scores for all the songs for all the algorithms to see which are the 'problem' songs in the collection and why.

To get an idea of how the different algorithms performed on the songs individually we can plot average recall and segmentation scores for each song from each algorithm as greyscale colour intensities in an image map

as shown in figures 9.13 and 9.14. Certain songs are problems for all algorithms and these show up as dark vertical lines on the plots from top to bottom. The best example of this for the recall values is song 150, 'Revolution 9' from disc 1 of the white album, which shows up as a clear dark line on both the recall map and the segmentation map. This track is not a tonal pop song but rather an example of electoacoustic *musique concrète*. Originated by Schaffer [Rey96] in the late 1940s and subsequently developed by composers including Stockhausen, Varese and Xenakis [Lev02]; this style of music is created by arranging and mixing snippets of pre-recorded audio tape into a new order to build compositions. It is therefore unsurprising that chord recognition algorithms find it problematic because there is no real chord structure present in the audio except for small snatches of sampled tonal material that appear from time to time in the mix.

For MIREX09, seven songs were excluded from the final evaluation results because they had average recall scores across all thirteen algorithms less than 25%. Table 9.8 shows the details for these seven songs and for four others that we have found to have low averages in all our evaluations. 'Revolution 9' is among the songs that were excluded from MIREX09.

Some songs are difficult for some algorithms but not for others. Song 107, 'Lovely Rita' from Sgt. Pepper has very low recall scores for all algorithms except for 'de' and the three 'ch' algorithms. Algorithm 'de' actually scores 87% for the recall on this song but 'md', 'pp', both 'ko' and both 'ogf' algorithms score zero. When we look at the segmentation scores, however, the scores look perfectly normal for all algorithms. On further investigation we find that this track has been mastered with a tuning centre frequency around 425Hz which is a long way off concert pitch 440Hz. In fact, 425Hz is about a quarter tone flat which means some algorithms may well decide that the song is in the key one semitone below that of the hand annotation. Figure 9.15 shows the outputs for algorithms 'de', 'md' and 'pp' compared with the hand annotation for a short excerpt of 'Lovely Rita'. It is clear to see that 'md' and 'pp' have basically estimated the chord sequence correctly but the symbols they have produced are all one semitone lower than the annotated symbols due to this tuning

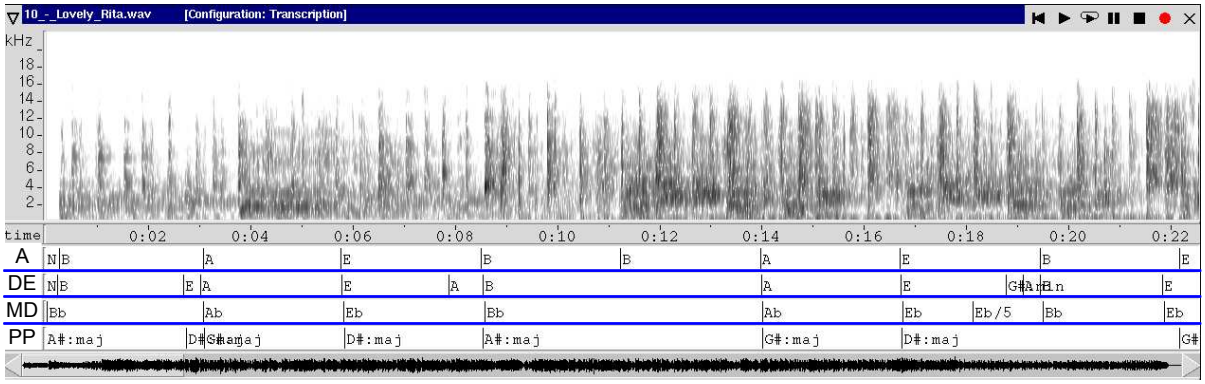


Figure 9.15: An excerpt of 'Lovely Rita' shown with the hand annotated chord symbols (A), the Dan Ellis algorithm output (DE), Mauch and Dixon output (MD) and the Papadopoulos and Peeters output (PP). MD and PP algorithms have both estimated the tuning a semitone below the hand annotation.

problem.

'Wild honey pie' is another track that many algorithms have difficulty with for chord symbol recall. This time 'de' and 'pp' score well but the others all fail. The tuning frequency for this song again is about a quartertone flat from concert pitch but an added difficulty is that all of the tonal chords in the hand annotation are in fact '`X:7`' type chords. This may mean that even if algorithms correctly guess the tuning, they may still have difficulty correctly identifying the chord family.

Looking at the image map in figure 9.14, we can see that song 105, 'Within you without you' from Sgt. Pepper, has very poor segmentation scores but the recall scores for this song do not appear to be too unusual. This song is probably the hardest for segmentation because it contains the longest continuous annotated tonal chord in the collection (as discussed in section 6.6.5). Although the implied harmony of that section of the song does not change for slightly more than 137 seconds, the movement in the sitar melody line and string arrangements will doubtless cause chord recognition algorithms to over-segment that section compared to the transcription.

Table 9.8: 'Problem songs' in the collection. These are the songs that algorithms generally have the most trouble with either in terms of recall or segmentation. Those marked with a cross in the MX09 column were excluded from the MIREX09 evaluations. It should be noted that although 'Ticket to ride' was excluded from MIREX09, it appears to have fairly normal results in our evaluations.

					$\mathcal{R}'_{\overrightarrow{\mathbf{P}},3,D_0}$				$\mathcal{Q}$			
No.	**Disc**	**Track**	**Tuning**	**MX09**	**mean**	**max**	**min**	**range**	**mean**	**max**	**min**	**range**
62	05	07 Ticket To Ride	430	✗	0.64	0.84	0.37	0.46	0.64	0.85	0.30	0.55
82	06	13 If I Needed Someone	441	✓	0.27	0.62	0.16	0.45	0.49	0.59	0.33	0.26
87	07	04 Love You To	440	✗	0.42	0.90	0.00	0.90	0.43	0.74	0.16	0.58
105	08	08 Within You Without You	440	✓	0.37	0.61	0.21	0.39	0.17	0.28	0.06	0.22
107	08	10 Lovely Rita	425	✗	0.19	0.87	0.00	0.87	0.69	0.91	0.40	0.51
126	10CD1	05 Wild Honey Pie	426	✗	0.17	0.88	0.00	0.87	0.58	0.92	0.25	0.67
127	10CD1	06 The Continuing Story of Bungalow Bill	438	✗	0.23	0.70	0.09	0.61	0.64	0.83	0.51	0.32
126	10CD1	14 Don't Pass Me By	436	✗	0.29	0.69	0.09	0.59	0.61	0.76	0.23	0.52
140	10CD2	02 Yer Blues	440	✓	0.43	0.59	0.28	0.32	0.50	0.67	0.31	0.36
144	10CD2	06 Helter Skelter	440	✓	0.54	0.86	0.29	0.57	0.45	0.88	0.23	0.65
150	10CD2	12 Revolution 9	440	✗	0.20	0.43	0.09	0.34	0.31	0.57	0.18	0.39

Chapter 10

Conclusions and future work

In this thesis we have addressed the subject of chord recognition from audio and the evaluation of algorithms designed to perform that task. After reviewing theoretical underpinnings of pitch perception, consonance and harmony theory, a novel six dimensional model for pitch space was presented in chapter 2. This model was employed in chapter 3 as the basis of a pre-segmentation stage that improves the performance of our chord recognition algorithm from [HS05]. To test our chord recognition algorithms, we require a large annotated collection of audio for use as a ground truth and at the start of this work none was available. For this reason, we designed the chord symbol model and text syntax in chapter 4 and used it in the annotation process, discussed in chapter 6, allowing us to produce the Beatles chord transcription collection. In chapter 7 we proposed a novel alignment method based on simple audio fingerprints which allows a researcher to align their local copies of the Beatles audio accurately with our transcriptions.

In chapter 5 we presented methods for comparing chord symbols in our syntax which we subsequently used for analysing the statistics of the transcription collection in chapter 6. These same chord comparison methods were also used as a basis for a novel dictionary-based chord symbol recall calculation proposed in chapter 8. Along with the new chord symbol recall method, we also presented a complementary chord segmentation measure based on directional hamming distance and proposed an f-measure score for chord recognition evaluation combining the two.

Using the measures and techniques proposed in chapter 8 we presented evaluation results for our three chord recognition algorithms compared to

the entrants of the MIREX09 chord detection evaluation in chapter 9. We used the new recall evaluation techniques to analyse the recognition algorithms for different chord dictionaries and performance on individual chord types. Analysis of the results for the fifteen algorithms we evaluated also provided us with data on the relative difficulty of songs in the Beatles collection for the task of chord recognition. This information allowed us to identify 'problem songs' in the collection for which all algorithms obtained low scores.

We will now summarise the main outcomes from the research work presented here and follow with suggested directions for further work.

10.1 Conclusions

In chapter 2, we introduced a novel six dimensional model for equal tempered pitch space based on neo-riemannian music theory principles. By projecting chords as points inside the six dimensional torus we may visualise tonal relationships in a new way. In chapter 3 we showed how this model could be used to derive a six dimensional tonal centroid feature from twelve dimensional chroma vectors generated from digital audio recordings. We use the tonal centroid as the basis for our HCDF enabling harmonic segmentation. We have demonstrated that using the HCDF as a pre-segmentation step in our chord recognition system improved the algorithm's performance. Lee and Slaney [LS07, Lee08, LS08] have also used the tonal centroid as a feature vector for chord recognition itself.

One of the major contributions of this work is the Beatles transcription collection presented in chapter 6. We needed to create this set of chord annotations because no large collection was available when we started the research presented here. In order to produce a set of transcriptions that would be accurate and convenient for use in automated evaluation processes, we have developed a novel machine readable chord syntax which was presented in chapter 4. The syntax we have developed is easy to use for human musicians because it avoids potential sources of ambiguity (discussed in section 4.1) while staying close to common chord notation conventions discussed in section 4.2. The resulting context independent

chord model and associated text syntax is very flexible. For human users' convenience, we defined a list of seventeen shorthand chordtype labels for commonly occurring chords but the system allows for the spelling of any conceivable chord that might be found in western tonal harmony through the use of an interval list notation. The shorthand labels are themselves just macro definitions of interval lists, thus the system could easily be extended to include further shorthands if required.

We have used the chord syntax for producing our transcriptions but its uses are not limited to this task alone. Our chord model has been used as the basis for the RDF chord description in the OMRAS II chord ontology [SRMH07]. A heavier encoding of the model, based on XML or RDF, may be useful for data storage and transfer, particularly in the context of a larger framework such as the Music Ontology project [RASG07] which has to handle many different classes of musical data. Entering data manually in this kind of format is cumbersome compared to lighter format such as our syntax. Furthermore, given the machine readable nature of our syntax, it is ideally suited for use as a manual entry format that can be converted to alternative formats automatically by a computer. Indeed, a very slightly altered version of our syntax (the '`#`' symbol is replaced with '`s`') is used in the OMRAS II chord ontology [SRMH07] in just this way to provide compact chord labels for the chord symbol service.

Developing the chord symbol syntax enabled us to take on the task of transcribing the Beatles collection. In order to ensure that the chord transcriptions were of high quality, we devised a process for creating and verifying the accuracy of annotations as discussed in section 6.5. We have written a Matlab toolkit for manipulating chord data in our syntax and this enabled us to automatically check the transcription collection for correct syntax. The human listening tests used in the verification process were also made possible by the use of our Matlab functions and MIDI file tools to generate audio files containing synthesised versions of the chords combined with the original Beatles recordings.

Impact of the Beatles transcription collection

The impact of our transcription collection on the MIR community has been significant. The full set of verified Beatles transcriptions have been available since 2007. At the time of writing, this means they have been in use in the community for nearly three years. Many researchers including [Bel07, BPKF07, PP08, LS08, RK08, OGF09c, KO09c, RUS+09, MND09b] have used the transcriptions in their evaluations and many more have requested copies of the collection and the Matlab tools. At the time of writing, the number of researchers who have asked us directly for copies of the transcriptions is 67. However, given that '.lab' and RDF versions of the transcriptions are publically available on the isophonics website as announced at the ISMIR09 conference [MCD+09], the number of people actually using the collection is almost certainly higher. The transcription collection is an ongoing project and if users notify us of errors, we make corrections and release updated versions.

The transcription collection has been used as the test set for the MIREX08 and MIREX09 audio chord detection evaluations. As a result, the chord syntax as been adopted as a de-facto standard for output of chord recognition algorithms in the MIREX chord detection track. For MIREX09, new transcriptions in the same format provided by Mauch et al [MCD+09] were also included in the evaluation process.

Uses for the transcriptions other than in chord recognition evaluation have also been found. Mauch's paper 'Discovering chord idioms through Beatles and real book songs' [MDH+07], on which this author is also credited, demonstrates the use of the transcription collection as a primary data source for computational musicology itself. The collection has also generated recent interest from the online community involved in the infographics project 'Charting the Beatles'[1].

[1] http://www.chartingthebeatles.com

Audio alignment

Since the initial release, a number of researchers have complained about errors in the collection, referring to large timing discrepancies between the transcription and their local copy of the Beatles audio. To solve this problem, we developed the alignment method discussed in chapter 7 that allows the user to modify their local copy of the Beatles audio such that it is accurately aligned with our original audio files. The system employs a simple audio fingerprinting technique so the alignment data for the whole collection can be sent in a file less than 200kB in size. The algorithm will fail to align a local audio file that is time-stretched compared to our original. Dan Ellis found large timing discrepancies between his local audio and the transcriptions. Some of these problems were due to bad alignment, but some songs had been taken from different releases of the material and hence were the product of an alternative mastering process. After realignment, Ellis reported a 20% improvement in performance for his pre-trained chord recognition system [Ell09]. It is likely that Lee and Slaney also had audio that was aligned poorly with the transcriptions, given their comments on the accuracy of human transcriptions in [LS07, Lee08, LS08].

In order to use the alignment system on other types of music, it is important to consider that the Beatles recordings were all performed by real musicians without the aid of computerised sequencers and digital samplers. More recent music recordings using electronically generated sounds, such as drum loops and samples, could contain multiple matches for a fingerprint and confuse the algorithm. This problem might be mitigated slightly by using an adaptive algorithm that could choose the positions of the fingerprints carefully if there is an element such as a vocal or instrumental track which does not contain repeated material. Another possibility is that more than two fingerprints could be used and a slightly more complex algorithm introduced to search for the best match of several fingerprints relative to each other.

Chord symbol comparison

In chapter 5, we restated our chord syntax algebraically allowing us to define a set of functions for converting between different musical objects including chords, intervals, pitchnames and pitchclasses. We used these definitions as a basis for formalising the chord matching problem by defining ordered and unordered set matching functions in section 5.2. Using the set matching functions on strings (ordered sets of characters), pnsets (sets of pitchnames) and pcsets (sets of pitch classes), we then defined a system for comparing chord symbols by regarding them as strings, pnsets or pcsets depending on the context of the required comparison. We extended the system to allow direct comparison of chords with different cardinalities. In the same chapter we also proposed a chord likeness measure based on the number of tones shared between two chords.

Using the chord symbol toolkit, we implemented our chord matching functions in matlab and then used them to investigate the statistics of the transcription collection in section 6.6. By varying the parameters of the chord matching functions we were able to extract interesting data about the make up of the collection with regards to specific chords and chord types. This information is very useful because it gives us an insight into the nature of the collection. This allows us to be more confident when drawing conclusions from our evaluation results.

Chord recognition evaluation

In chapter 8, we used our chord matching functions again to develop a general method for calculating chord symbol recall. By using our chord matching functions in the calculation, we were able to define a set of parameters for the recall equation: match type, cardinality and dictionary. By selecting appropriate values for these three parameters, our chord symbol recall method can be used to evaluate any set of chord recognition algorithms in a fair manner. We also argued that using chord symbol recall alone is not the best way to evaluate chord recognition systems. In section 8.3, we proposed a segmentation quality measure that complements the recall evaluation and showed how the two may be combined to give a

single chord recognition f-measure.

In chapter 9, we used the evaluation techniques presented in chapter 8 to evaluate our three chord recognition algorithms and re-evaluate the output data from the recognition systems that were entered to the MIREX09 chord detection track. We presented evaluation results for our chord symbol recall method using various different parameters and compared these results with recall values calculated with the same chord mappings that were used to produce MIREX08 and 09 evaluation results.

In our recall results we found that the ranking of the algorithms stayed fairly constant for different parameters. One exception however was the 'rrhs3' algorithm by Rocher et al. This algorithm appeared to perform significantly better when evaluated with MIREX08 and 09 chord mappings compared with our new recall evaluation methods. However, the reported chord vocabulary for the 'rrhs3' algorithm is larger than the other algorithms, containing 12 different chord types. It is therefore likely that the strange results in the MIREX mapping categories were due to incorrect chord mapping generating false positive matches.

Songs with an average score of less than 25% across all algorithms were discarded from the official MIREX09 results on the assumption that there was something wrong with their transcription or alignment. However, we found that recall values for such 'problem songs' could vary quite significantly between algorithms and, in some cases, one or two algorithms obtained very high scores where others scored very poorly. A good example of this was *Lovely Rita* for which Dan Ellis' algorithm scored 87% but several others scored zero. In that case, it was found that tuning was the cause of the problem; the reference tuning frequency for the song being almost a quartertone away from concert pitch. As a result, some algorithms generated a sequence of chord symbols that were correct in terms of relative progression, but with estimated roots one semitone lower than the ground truth annotation throughout. Some people might claim that the transcriptions are incorrect when they find this situation. However, as we can see from our results, not all algorithms will necessarily get the tuning 'wrong' so changing the transcriptions would just reverse the current situation. By including the segmentation quality metric, we were able to

show that these algorithms were actually producing good results because their segmentations were still mostly correct despite the low recall scores. We therefore recommend that attention be paid to songs where large discrepancies exist between the results for two performance measures. In cases where recall is significantly lower than the segmentation score, it is advisable to re-evaluate the recall result for that song using a transposed version of the ground truth to check for anomalous results caused by tuning errors.

As with all research, the state of the art in chord recogntion constantly advances. In MIREX08, all the entries were major/minor chord recognisers. For MIREX09, the algorithms were a mixture with some still detecting only major and minor chords, but others having larger vocabularies of up to twelve chord types. At the time of writing, preparation for MIREX10 is already underway and undoubtedly there will be new algorithms entered this year that will perform better and handle larger chord vocabularies. The evaluation measures we have demonstrated in this thesis are designed with these future possibilities in mind. As the capabilities of algorithms advance, we need only alter the parameters of the evaluation methods in order to deal with the new developments.

The statistics in section 6.6 show that the Beatles transcriptions contain a very high proportion of major and minor triad chords. As such, the test set is ideal for the algorithms that have been evaluated here in chapter 9. However, as algorithms improve and chord vocabularies grow, we will need more varied ground truth test sets in order to evaluate the new algorithms properly. It is hoped that the techniques and tools developed in this work will make the task easier for the creators of new chord transcriptions in the future.

10.2 Future work

In this section we will summarise potential directions for future work.

Computational musicology

The statisics we presented in section 6.6 only scratch the surface of the possible information that could be extracted from the Beatles transcription collection using the chord matching functions we developed in chapter 5. One area we would like to explore is the extension of chord matching functions to include *prime* pcset matching. Prime pcset analysis is popular in the music theory community [Lew82, For85, Sol82] and it would be very interesting to further analyse the Beatles transcription collection in this way.

The statistics that we presented here are all based on analysis and classification of individual chord symbols. In this work we have not analysed the transcriptions in terms of chord progression. However, it is precisely the Beatles' skilled use of harmonic progression that made so many of their songs so instantly memorable. We therefore intend to pursue further research on chord progression patterns in the collection.

It would also be interesting to analyse Mauch's chord transcriptions for the Queen and Zweieck songs using the techniques we employed in section 6.6. This will enable us to compare the properties of the different collections in a quantitative way.

Chord syntax

The chord label syntax described in chapter 4 has been specifically designed to be key context independent. It would be possible however, to extend the chord model to allow definition of a key context and then specify chord roots relative to that key context. For example, the chord progression

```
C:maj | D:min7 | F:maj | G:7 | C:maj
```

could be described as

```
I:maj | II:min7 | IV:maj | V:7 | I:maj
```

or perhaps in the cleaner, but slightly less intuitive form

```
1:maj | 2:min7 | 4:maj | 5:7 | 1:maj
```

in the key of C major. The original model was designed for use in simple text annotations where each chord symbol has to stand alone, entirely separate from others. Using an encoding such as RDF where a higher level of structure may be expressed, the ability to model chord progressions in terms of a key context and relative chord symbols may be useful for future analysis and annotation tasks.

Conversion of a chord progression stored using this alternative model to the context free form is simple. However, conversion from the context-free form to key context and relative chord symbols is not trivial. This leads to another possible area of future work, designing algorithms that can automatically infer key context from the context-free chord transcriptions that we already have.

Chord sequence comparison

The chord sequence likeness measure introduced in section 5.6 is a simple method for comparing chord sequences in terms of their component pitch-names or pitchclasses. Where one chord sequence is a transposed version of the other, this simple likeness measure will evaluate as a low score. Likewise, if two chord sequences are the same but are not time aligned, this measure will also fail to reflect their similarity.

An interesting area of possible future work is the development of a chord sequence similarity measure that can detect similar relative chord progressions in different songs. By treating each unique chord symbol as a dimension in a vector space, we may view a chord sequence as a path in this space. Moving from one chord to the next in a progression will cause a unit change in one dimension only. Ordering of the dimensions in the space might be based on an analysis of chord symbol frequency. Small differences between sequences caused by insertion or deletion of chords in one relative to the other may therefore be discarded using a principal component analysis. As a result, a pair of chord sequences that have similar relative progressions will have similar paths in the space regardless of key. Such a system would potentially be very useful in cover song detection.

Extended test set

The statistics in section 6.6 show that over three quarters of the chords in the Beatles collection are major and minor triads. As chord recognition systems become more advanced, a collection of annotated audio that contains a more balanced mixture of chord types may be more appropriate in order to keep the evaluation of chord symbol recall fair. Producing ground truth transcriptions of chords from jazz and classical music as well as further pop music recordings may help to make the test set more balanced.

An alternative approach could be to generate synthesised test audio containing more varied types of chords [WDR09]. This would allow controlled testing for recognition algorithms. However, this would not be a substitute for transcriptions of real recordings. While synthesised test data can be generated that contains any chord we want, it should be noted that in any large collection of real western tonal music, the proportion of major and minor triads will always be higher than other chordtypes due to the nature of western harmony.

Audio chord recognition

There are numerous ways in which we might improve our chord recognition algorithms in the future. Simple steps such as pre-processing to detect sections of silence would immediately have a positive impact on performance. Use of better chord templates that reflect the harmonic content of chord chroma and a different template matching mechanism should also see improved results.

Another possible direction in future work has become apparent through the analysis of our results in chapter 9. The various algorithms perform differently for different chord types. We therefore believe that it might be interesting to develop a 'mixture of experts' chord recognition system that incorporates several of these different algorithms. The separate algorithms can be used in parallel to generate chord estimates and a decision on the final estimate can be made using our knowledge of how reliable each expert is for the particular chords they produce.

Chord tools

The current version of the chord toolkit is implemented in Matlab. Johan Pauwels has also implemented a parser for the chord symbols in C++. At the time of writing, the MIREX10 evaluation system is being re-implemented in Java and our chord symbol syntax will be used as the basis for the chord detection task again. To make the tools available on open platforms we intend to produce implementations of the chord tools in Java and Python and these will be open sourced in the same way as our Matlab version.

Bibliography

[ABMD10] Amélie Anglade, Emmanouil Benetos, Matthias Mauch, and Simon Dixon. Improving music genre classification using automatically induced harmony-based rules. *Submitted to Journal of New Music Research*, 2010.

[AD08] Amélie Anglade and Simon Dixon. Characterisation of harmony with inductive logic programming. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, pages 63–68, Philadelphia, PA, 2008.

[ANS+05] S. Abdallah, K. Noland, M. Sandler, M. Casey, and C. Rhodes. Theory and evaluation of a Bayesian music structure extractor. In *Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR 2005, London, UK*, pages 420–425, 2005.

[ARD09] Amélie Anglade, Raphael Ramirez, and Simon Dixon. Genre classification using harmony rules induced from automatic chord transcriptions. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 669–674, 2009.

[Bal97] Llorenç Balsach. Application of virtual pitch theory in music analysis. *Journal of New Music Research*, 26(3):244–265, 1997.

[Bel07] Juan P. Bello. Audio-based cover song retrieval using approximate chord sequences: Testing shifts, gaps, swaps and beats. In *Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR 2007, Vienna, Austria*, 2007.

[Ben90] Arthur H. Benade, editor. *Fundamentals of Musical Acoustics: Second, Revised Edition*. Dover Publications, 1990.

[BLEW04] Adam Berenzweig, Beth Logan, Daniel P. W. Ellis, and Brian P. W. Whitman. A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, 28(2):63–76, 2004.

[BP92] Judith C. Brown and Miller S. Puckette. An Efficient Algorithm for the Calculation of a Constant Q Transform. *Journal of the Acoustical Society of America*, 92(5):2698–2701, 1992.

[BP05] Juan P. Bello and Jeremy Pickens. A Robust Mid-level Representation for Harmonic Content in Music Signals. In *Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR 2005, London, UK*, pages 304–311, 2005.

[BPKF07] John Ashley Burgoyne, Laurent Pugin, Corey Kereliuk, and Ichiro Fujinaga. A cross-validated study of modelling strategies for automatic chord recognition in audio. In *Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR 2007, Vienna, Austria*, pages 251–254, 2007.

[BPO99] Benjamin Blankertz, Hendrik Purwins, and Klaus Obermayer. Toroidal models of inter-key relations in tonal music. In *International Conference on Systematic and Comparative Musicology*, 1999.

[Bro91] Judith Brown. Calculation of a Constant Q Spectral Transform. *Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[CB09] T. Cho and J.P. Bello. Real-time implementation of HMM-based chord estimation in musical audio. In *Proceedings of ICMC 2009. Montreal, Canada.*, August 2009.

[CBKH05] Pedro Cano, Eloi Batlle, Ton Kalker, and Jaap Haitsma. A review of audio fingerprinting. *Journal of VLSI Signal Processing Systems*, 41(3):271–284, 2005.

[Che00] Elaine Chew. *Towards a Mathematical Model of Tonality*. PhD thesis, Massachusetts Institute of Technology, 2000.

[Che01] Elaine Chew. Modeling tonality: Applications to music cognition. In *Proceedings of the 23rd Annual Conference of the Cognitive Science Society, CogSci2001*, Edinburgh, Scotland, 2001.

[Che02] Elaine Chew. The spiral array: An algorithm for determining key boundaries. *Proceedings of the Second International Conference, ICMAI 2002*, pages 18–31, 2002.

[CLSB06] Chris Cannam, Christian Landone, Mark Sandler, and Juan P. Bello. The Sonic Visualiser: A Visualisation Platform for Semantic Descriptors from Musical Signals. In *Proceedings of the 7th International Conference on Music Information Retrieval, ISMIR 2006, Victoria, Canada*, 2006.

[CML07] Benoit Catteau, Jean-Pierre Martens, and Marc Leman. A probabilistic framework for audio-based tonal key and chord recognition. In Reinhold Decker and Hans-Joachim Lenz, editors, *Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation e.V., Freie Universität Berlin, March 8-10, 2006*, pages 637–644, 2007.

[Coh98] Richard Cohn. An introduction to neo-riemannian theory: A survey and historical perspective. *Journal of Music Theory*, 42(2):167–180, 1998.

[Cok64] Jerry Coker. *Improvising Jazz*. Simon and Schuster, New York, 1964.

[Coo99] Perry R. Cook. *Music Cognition and Computerized Sound - An introduction to Psychoacoustics*. The MIT Press, Cambridge, Massachusetts, 1999.

[CPB05] Giordano Cabral, François Pachet, and Jean-Pierre Briot. Automatic X traditional descriptor extraction: the case of chord recognition. In *Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR 2005, London, UK*, pages 444–449, 2005.

[Dav07] Matthew Davies. *Towards Automatic Rhythmic Accompaniment*. PhD thesis, Queen Mary University of London, London, UK, August 2007.

[DHVW08] B. De Haas, R. Veltkamp, and F. Wiering. Tonal Pitch Step Distance: a Similarity Measure for Chord Progressions. In *Proceedings of the 9th International Conference on Music Information Retrieval, ISMIR 2008, Philadelphia, USA*, pages 51–56, 2008.

[Dix06] Simon Dixon. Onset Detection Revisited. In *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx06)*, Montreal, Canada, 2006.

[Dow08] J. Stephen Downie. The Music Information Retrieval Evaluation Exchange (2005-2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[EGL09] Ravelli E., Richard G., and Daudet L. Audio signal representations for indexing in the transform domain. *IEEE Transactions on Audio, Speech, and Language Processing*, 2009.

[Ell09] Dan Ellis. The 2009 Labrosa pretrained audio chord recognition system. In *MIREX Submission Abstracts*. 2009.
`http://www.music-ir.org/mirex/abstracts/2009/DE.pdf`.

[ET04] Tuomas Eerola and Petri Toiviainen. *MIDI Toolbox: MATLAB Tools for Music Research*. University of Jyväskylä, Jyväskylä, Finland, 2004.
`www.jyu.fi/musica/miditoolbox/`.

[For85] Allen Forte. Pitch-class set analysis today. *Music Analysis*, 4(1/2):29–58, 1985.

[Fri40] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[Fuj99] Takuya Fujishima. Real time chord recognition of musical sound: a system using Common Lisp Music. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 464–467, 1999.

[GÓ6] Emilia Gómez. *Tonal Description of Audio Music Signals*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2006.

[GH04] Emilia Gómez and Perfecto Herrera. Automatic extraction of tonal metadata from polyphonic audio recordings. In *Proceedings of the AES 25th International Conference*, pages 74–80, London, UK, 2004.

[GHNO02] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Popular, classical, and jazz music databases. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, pages pp.287–288, October 2002.

[Gre71] Ted Greene. *Chord Chemistry*. Alfred Publishing, 1971.

[HA96] David M. Howard and James A. S. Angus. *Acoustics and Psychoacoustics*. Focal Press, Oxford, 1996.

[HD95] Qian Huang and B. Dom. Quantitative methods of evaluating image segmentation. In *Proceedings of the International Conference on Image Processing, 1995*, volume 3, pages 53–56, 1995.

[HK02] Jaap Haitsma and Ton Kalker. A highly robust audio fingerprinting system. In *Proceedings of the 3rd International*

Conference on Music Information Retrieval (ISMIR 2002), pages 107–115, 2002.

[HM03] S. Hainsworth and M. Macleod. Onset Detection in Musical Audio Signals. In *Proceedings of ICMC*, Singapore, 2003.

[HS05] C. A. Harte and M. Sandler. Automatic chord identification using a quantised chromagram. In *Proceedings of the 118th Convention of the AES*, 2005.

[HS09] Christopher Harte and Mark Sandler. MIREX automatic chord recognition using quantised chroma and harmonic change segmentation. In *MIREX Submission Abstracts*. 2009. `http://www.music-ir.org/mirex/abstracts/2009/harte_mirex09.pdf`.

[HSAG05] C. A. Harte, M. Sandler, S. Abdallah, and E. Gómez. Symbolic representaion of musical chords: A proposed syntax for text annotations. In *Proceedings of ISMIR*, pages 66–71. Queen Mary, University of London, 2005.

[HSG06] Christopher Harte, Mark Sandler, and Martin Gasser. Detecting harmonic change in musical audio. In *AMCMM '06: Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 21–26, New York, NY, USA, 2006. ACM.

[Hye95] Brian Hyer. Re-Imagining Riemann. *Journal of Music Theory*, 39(1):101–138, 1995.

[KAH+02] Thorsten Kastner, Eric Allamanche, Jurgen Herre, Oliver Hellmuth, Markus Cremer, and Holger Grossmann. MPEG-7 scalable robust audio fingerprinting. In *Proceedings of the 112th Audio Engineering Society Convention*, May 2002.

[Ken94] Michael Kennedy. *The Oxford Dictionary of Music*. Oxford University Press, Oxford, 1994.

[KO09a] Maksim Khadkevich and Maurizio Omologo. MIREX audio chord detection. In *MIREX Submission Abstracts*. 2009. `http://www.music-ir.org/mirex/abstracts/2009/KO.pdf`.

[KO09b] Maksim Khadkevich and Maurizio Omologo. Phase-change based tuning for automatic chord recognition. In *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09), Como, Italy*, 2009.

[KO09c] Maksim Khadkevich and Maurizio Omologo. Use of hidden Markov models and factored language models for automatic chord recognition. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[Kru90] Carol L. Krumhansl. *Cognitive Foundations of Musical Pitch*. Oxford University Press, New York, 1990.

[KS79] Carol Krumhansl and Roger Shepard. Quantification of the hierarchy of tonal functions within a diatonic context. *Journal of Experimental Psychology: Human Perception and Performance*, 5:579–594, 1979.

[lam] Lame open source mp3 encoder and decoder. `http://lame.sourceforge.net`.

[LB07] E. Li and J.P. Bello. Key-independent classification of harmonic change in musical audio. In *Proceedings of the 123rd Convention of the Audio Engineering Society. New York, NY, USA.*, October 2007.

[Lee08] Kyogu Lee. *A System for Acoustic Chord Transcription and Key Extraction from Audio Using Hidden Markov Models Trained on Synthesized Audio*. PhD thesis, Stanford University, 2008.

[Ler01] Fred Lerdahl. *Tonal Pitch Space*. Oxford University Press, 2001.

[Lev02] Benjamin R. Levy. Electroacoustic music: The continuing tradition. *Computer Music Journal*, 26(3):81–84, 2002.

[Lew82] David Lewin. Transformational techniques in atonal and other music theories. *Perspectives of New Music*, 21(1/2):312–371, 1982.

[Lew92] Mark Lewisohn. *The complete Beatles chronicle*. Chancellor Press, 1992.

[LH62a] H. Christopher Longuet-Higgins. Letter to a musical friend. *The Music Review*, 23:244–248, 1962.

[LH62b] H. Christopher Longuet-Higgins. Second letter to a musical friend. *The Music Review*, 23:271–280, 1962.

[LH92] H. Christopher Longuet-Higgins. The perception of melodies. *Machine models of music*, pages 471–495, 1992.

[LM81] Henry Ledgard and Michael Marcotty. *The Programming Language Landscape*. Science Research Associates, Inc., Chicago, 1981.

[Lon01] Justin London. Some non-isomorphisms between pitch and time. *Music Theory Midwest*, 2001.

[LS07] Kyogu Lee and Malcolm Slaney. A Unified System for Chord Transcription and Key Extraction Using Hidden Markov Models. In *Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR 2007, Vienna, Austria*, 2007.

[LS08] Kyogu Lee and Malcolm Slaney. 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, February 2008.

[Mac08] Ian MacDonald. *Revolution in the head*. Vintage books, 2008.

[Mau10] Matthias Mauch. *Automatic Chord Transcription from Audio Using Computational Models of Musical Context*. PhD thesis, Queen Mary University of London, 2010.

[MCD+09] M. Mauch, C. Cannam, M. Davies, S. Dixon, C. Harte, S. Kolozali, D. Tidhar, and M. Sandler. OMRAS2 metadata project 2009. In *Late-breaking session at the 10th International Conference on Music Information Retrieval, Kobe, Japan*, 2009.
http://ismir2009.ismir.net/proceedings/LBD-18.pdf.

[MD08] Matthias Mauch and Simon Dixon. A Discrete Mixture Model for Chord Labelling. In *Proceedings of the 9th International Conference on Music Information Retrieval, ISMIR 2008, Philadelphia, USA*, pages 45–50, 2008.

[MDH+07] Matthias Mauch, Simon Dixon, Christopher Harte, Michael Casey, and Benjamin Fields. Discovering Chord Idioms through Beatles and Real Book Songs. In *Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR 2007, Vienna, Austria*, 2007.

[Mer06] David Meredith. *Computing Pitch Names in Tonal Music: A Comparative Analysis of Pitch Spelling Algorithms*. PhD thesis, University of Oxford, 2006.

[mira] MIREX09 audio chord detection subtask results, music information retrieval evaluation exchange, Retrieved: April 2010.
http://www.music-ir.org/mirex/2009/index.php/Audio_Chord_Detection_Results.

[mirb] MIREX09 audio chord detection subtask wikipage, music information retrieval evaluation exchange, Retrieved: April 2010.
http://www.music-ir.org/mirex/2009/index.php/Audio_Chord_Detection.

[mirc] MIREX09 wiki main page, Retrieved: April 2010.
http://www.music-ir.org/mirex/2009/index.php/Main_Page.

[MMDK07] M. F. McKinney, D. Moelants, M. E. P. Davies, and A. Klapuri. Evaluation of audio beat tracking and music tempo extraction algorithms. *Journal of New Music Research*, 36(1):1–16, 2007.

[MND09a] Matthias Mauch, Katy Noland, and Simon Dixon. MIREX submissions for audio chord detection (no training) and structural segmentation. In *MIREX Submission Abstracts*. 2009. http://www.music-ir.org/mirex/abstracts/2009/ACD_SS_mauch.pdf.

[MND09b] Matthias Mauch, Katy C. Noland, and Simon Dixon. Using musical structure to enhance automatic chord transcription. In *Proceedings of the 10th International Conference on Music Information Retrieval, Kobe, Japan*, pages 231–236, 2009.

[Mol04] Cleve Moler. *Numerical Computing with MATLAB*. The MathWorks, 2004.

[Mur02] Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, 2002.

[MXKS04] N. C. Maddage, C. Xu, M. S. Kankanhalli, and X. Shao. Content-based music structure analysis with applications to music semantics understanding. In *Proceedings of the 12th annual ACM international conference on Multimedia*, 2004.

[Nol09] Katy Noland. *Computational Tonality Estimation: Signal Processing and Hidden Markov Models*. PhD thesis, Queen Mary University of London, 2009.

[OGF09a] Laurent Oudre, Yves Grenier, and Cédric Févotte. MIREX chord recognition system system 1 : Major and minor chords. In *MIREX Submission Abstracts*. 2009. http://www.music-ir.org/mirex/abstracts/2009/OGF1.pdf.

[OGF09b] Laurent Oudre, Yves Grenier, and Cédric Févotte. MIREX chord recognition system system 2 : Major and minor and dominant seventh chords. In *MIREX Submission Abstracts*.

2009.
http://www.music-ir.org/mirex/abstracts/2009/OGF2.pdf.

[OGF09c] Laurent Oudre, Yves Grenier, and Cédric Févotte. Template-based chord recognition: Influence of the chord types. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[Ols67] Harry F. Olsen. *Music, Physics and Engineering, 2nd ed.* Dover Publications, Inc. N.Y., 1967.

[PB02] Bryan Pardo and William P. Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, 26(2):27–49, 2002.

[PBO00] Hendrik Purwins, Benjamin Blankertz, and Klaus Obermayer. A new method for tracking modulations in tonal music in audio data format. In *International Joint Conference on Neural Networks (IJCNN'00)*, volume 6, pages 270–275, 2000.

[Ped03] Dominic Pedler. *The songwriting secrets of the Beatles*. Omnibus Press, 2003.

[Pee06] Geoffroy Peeters. Chroma-based Estimation of Musical Key from Audio-Signal Analysis. In *Proceedings of the 7th International Conference on Music Information Retrieval, ISMIR 2006, Victoria, Canada*, 2006.

[pi08] J. Stephen Downie (principal investigator). Networked Environment for Music Analysis (NEMA), 2008.
http://nema.lis.uiuc.edu/.

[PL65] R. Plomp and J. M. Levelt. Tonal consonance and critical bandwidth. *Journal of the Acoustical Society of America*, 38:548–560, 1965.

[Pol] Alan W. Pollack. Alan W. Pollack's Notes on... Series, The Beatles, Retrieved: April 2010.
http://www.icce.rug.nl/ soundscapes/DATABASES/AWP/awp-notes_on.shtml.

[PP07] Hélène Papadopoulos and Geoffroy Peeters. Large-Scale Study of Chord Estimation Algorithms Based on Chroma Representation and HMM. In *CBMI '07. International Workshop on Content-Based Multimedia Indexing.*, 2007.

[PP08] Hélène Papadopoulos and Geoffroy Peeters. Simultaneous estimation of chord progression and downbeats from an audio file. In *Proceedings of the 2008 ICASSP Conference*, pages 121–124, 2008.

[PP09] Hélène Papadopoulos and Geoffroy Peeters. Joint estimation of chords and downbeats. In *MIREX Submission Abstracts*. 2009.
`http://www.music-ir.org/mirex/abstracts/2009/PPupdated.pdf`.

[Pra01] William K. Pratt. *Digital Image Processing: PIKS Inside, Third Edition*. John Wiley & Sons, Inc., 2001.

[PSRI09] C. Pérez-Sancho, D. Rizo, and J. M. Inesta. Genre classification using chords and stochastic language models. *Connection Science*, 21(2 & 3):145–159, June 2009.

[Pur05] Hendrik Purwins. *Profiles of Pitch Classes — Circularity of Relative Pitch and Key: Experiments, Models, Music Analysis, and Perspectives*. PhD thesis, Technische Universität Berlin, Germany, October 2005.

[PVM09] Johan Pauwels, Matthias Varewyck, and Jean-Pierre Martens. Audio chord extraction using a probabilistic model. In *MIREX Submission Abstracts*. 2009.
`http://www.music-ir.org/mirex/abstracts/2009/PVM.pdf`.

[Rab89] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[RASG07] Y Raimond, S Abdallah, M Sandler, and F Giasson. The music ontology specification, 2007.

[Reg73] Eric Regener. *Pitch Notation and Equal Temperament: A Formal Study*. PhD thesis, University of California Press, Berkeley, CA, 1973.

[Reg75] Eric Regener. The number seven in the theory of intonation. *Journal of Music Theory*, 19(1):140–153, 1975.

[Rey96] Jean de Reydellet. Pierre Schaeffer, 1910-1995: The Founder of "Musique Concrete". *Computer Music Journal*, 20(2):10–11, 1996.

[RHG08] M. Riley, E. Heinen, and J. Ghosh. A text retrieval approach to content-based audio retrieval. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 08)*, 2008.

[Rie15] Hugo Riemann. Ideen zu einer 'Lehre von den Tonvorstellungen'. *Jahrbuch der Bibliothek Peters*, pages 21–22, 1914-1915.

[RK08] Matti P. Ryynänen and Anssi P. Klapuri. Automatic Transcription of Melody, Bass Line, and Chords in Polyphonic Music. *Computer Music Journal*, 32(3):72–86, 2008.

[Roo00] Ricky Rooksby. *The Beatles Complete Chord Songbook*. Wise Publications, 2000.

[RRD08] Emmanuel Ravelli, Gaël Richard, and Laurent Daudet. Fast MIR in a Sparse Transform Domain. In *ISMIR*, pages 527–532, 2008.

[RRHS09a] Thomas Rocher, Matthias Robine, Pierre Hanna, and Robert Strandh. Dynamic chord analysis. In *MIREX Submission Abstracts*. 2009.

`http://www.music-ir.org/mirex/abstracts/2009/RRHS.pdf`.

[RRHS09b] Thomas Rocher, Matthias Robine, Pierre Hanna, and Robert Strandh. Dynamic chord analysis for symbolic music. In *Proceedings of the International Computer Music Conference (ICMC)*, Montreal, Quebec, Canada, August 2009.

[RSN08] Johannes Reinhard, Sebastian Stober, and Andreas Nürnberger. Enhancing chord classification through neighbourhood histograms. In *Proceedings of the 6th International Workshop on Content-Based Multimedia Indexing (CBMI 2008)*, 2008.

[RUS+09] J.T. Reed, Yushi Ueda, S. Siniscalchi, Yuki Uchiyama, Shigeki Sagayama, and C.-H. Lee. Minimum classification error training to improve isolated chord recognition. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 609–614, 2009.

[SB00] Kåre Sjölander and Jonas Beskow. Wavesurfer - an open source speech tool. In *Proceedings of the International Conference on Spoken Language Processing*, 2000.

[Sch54] Arnold Schoenberg. *Structural Functions of Harmony*. Faber and Faber Limited, London, 1954.

[SE03] Alexander Sheh and Daniel P.W. Ellis. Chord Segmentation and Recognition using EM-Trained Hidden Markov Models. *Proceedings of the ICMC 2003*, 2003.

[SF] Eleanor Selfridge-Field. Music theory for computer applications, Retrieved August 2010.
`http://www.ccarh.org/courses/254/MusicTheory_ComputerApps2004.htm`.

[She64] Roger Shepard. Circularity in judgments of relative pitch. *Journal of the Acoustical Society of America*, 35:2346–2353, 1964.

[SIY+08] K. Sumi, K. Itoyama, K. Yoshii, K. Komatani, T. Ogata, and H.G. Okuno. Automatic chord recognition based on probabilistic integration of chord transition and bass pitch estimation. In *Proceedings of the 9th International Conference on Music Information Retrieval, ISMIR 2008, Philadelphia, USA*, 2008.

[SJ01] B. Su and S. Jeng. Multi-timbre chord classification using wavelet transform and self-organized map neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3377–3380, 2001.

[SMW04] Arun Shenoy, Roshni Mohapatra, and Ye Wang. Key determination of acoustic musical signals. In *International Conference on Multimedia and Expo*. 2004.

[Sol82] Larry Solomon. The list of chords, their properties and uses. *Interface, Journal of New Music Research*, 11(2):61–107, 1982.

[SRMH07] Christopher Sutton, Yves Raimond, Matthias Mauch, and Christopher Harte. The Chord Ontology, 2007. `http://purl.org/ontology/chord/`.

[Sta97] Richard P. Stanley. *Enumerative Combinatorics, Volume 1*. Cambridge University Press, Cambridge, 1997.

[Ste02] Mark Steedman. Helmholtz' and Longuet-Higgins' Theories of Consonance and Harmony. *Unpublished Tutorial Paper*, 2002.

[SVB09] Ricardo Scholz, Emmanuel Vincent, and Frédéric Bimbot. Robust modeling of musical chord sequences using probabilistic n-grams. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2009.

[Tag03] Philip Tagg. Tagg's harmony handout, available at www.tagg.org. *Version 3, (accessed April 2010)*, 2003.

[Tay89] Eric Taylor. *The AB Guide to Music Theory Part 1*. ABRSM Publishing Ltd, Portland Place, London, UK, 1989.

[Tay91] Eric Taylor. *The AB Guide to Music Theory Part 2*. ABRSM Publishing Ltd, Portland Place, London, UK, 1991.

[Tem01] D. Temperley. *The Cognition of Basic Musical Structures*. MIT Press, Cambridge, MA, 2001.

[Tuk71] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1971.

[Ulr77] John Wade Ulrich. The analysis and synthesis of jazz by computer. In *Proceedings of the 5th International Joint Conference on Artifcial Intelligence. Los Altos*, 1977.

[VPM08] Matthias Varewyck, Johan Pauwels, and Jean-Pierre Martens. A novel chroma representation of polyphonic music based on multiple pitch tracking techniques. In *Proceedings of the 16th ACM International Conference on Multimedia*, pages 667–670, 2008.

[WDR09] Jan Weil, J. L. Durrieu, and Gaël Richard. Automatic generation of lead sheets from polyphonic music signals. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[WEJ09] Adrian Weller, Dan Ellis, and Tony Jebara. Structured prediction models for chord transcription of music audio. In *MIREX Submission Abstracts*. 2009.
`http://www.cs.columbia.edu/ jebara/papers/icmla09adrian.pdf.`

[WMR92] Robert W. Wason, Elizabeth West Marvin, and Hugo Riemann. Riemann's "Ideen zu Einer 'Lehre von den Tonvorstellungen' ": An Annotated Translation. *Journal of Music Theory*, 36(1):69–79, 1992.

[YKK+04] Takuya Yoshioka, Tetsuro Kitahara, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Automatic chord transcription with concurrent recognition of chord symbols and boundaries. In *Proceedings of the 5th International Conference on Music Information Retrieval, ISMIR 2004, Barcelona, Spain*, pages 100–105, 2004.

[You39] Robert W. Young. Terminology for logarithmic frequency units. *The Journal of the Acoustical Society of America*, 11(1):134–139, 1939.

[ZR07] Veronika Zenz and Andreas Rauber. Automatic chord detection incorporating beat and key detection. In *Proceedings of the 2007 IEEE International Conference on Signal Processing and Communications (ICSPC 2007)*, 2007.

Appendix A

Derivation: inequalities for 6D space

Interval distance relations from the tonic that must be satisfied:

$$d(\text{P5,P4}) < d(\text{M3,m6}) < d(\text{m3,M6}) < d(\text{M2,m7}) < d(\text{m2,M7}) < d(\text{d5,a4}) \tag{A.1}$$

Distances in the six dimensional space in terms of radii of the circle of fifths r_1, the circle of minor thirds r_2 and the circle of major thirds r_3:

$$\begin{aligned}
d(\text{P5,P4})^2 &= (2r_1 \sin\frac{\pi}{12})^2 + 2{r_2}^2 + (2r_3 \cos\frac{\pi}{6})^2 && \text{(A.2)}\\
d(\text{M3,m6})^2 &= (2r_1 \cos\frac{\pi}{6})^2 + (2r_3 \cos\frac{\pi}{6})^2 && \text{(A.3)}\\
d(\text{m3,M6})^2 &= 2{r_1}^2 + 2{r_2}^2 && \text{(A.4)}\\
d(\text{M2,m7})^2 &= {r_1}^2 + (2r_2)^2 + (2r_3 \cos\frac{\pi}{6})^2 && \text{(A.5)}\\
d(\text{m2,M7})^2 &= (2r_1 \cos\frac{\pi}{12})^2 + 2{r_2}^2 + (2r_3 \cos\frac{\pi}{6})^2 && \text{(A.6)}\\
d(\text{d5,a4})^2 &= (2r_1)^2 + (2r_2)^2 && \text{(A.7)}
\end{aligned}$$

These simplify to

$$\begin{aligned}
d(\text{P5,P4})^2 &= (2-\sqrt{3}){r_1}^2 + 2{r_2}^2 + 3{r_3}^2 && \text{(A.8)}\\
d(\text{M3,m3})^2 &= 3{r_1}^2 + 3{r_3}^2 && \text{(A.9)}\\
d(\text{m3,M6})^2 &= 2{r_1}^2 + 2{r_2}^2 && \text{(A.10)}\\
d(\text{M2,m7})^2 &= {r_1}^2 + 4{r_2}^2 + 3{r_3}^2 && \text{(A.11)}\\
d(\text{m2,M7})^2 &= (2+\sqrt{3}){r_1}^2 + 2{r_2}^2 + 3{r_3}^2 && \text{(A.12)}\\
d(\text{d5,a4})^2 &= 4{r_1}^2 + 4{r_2}^2 && \text{(A.13)}
\end{aligned}$$

Inequality 1

Let us assume $r_1 = 1$ then

$$\begin{aligned} d(\text{P5,P4}) &< d(\text{M3,m6}) \\ (2-\sqrt{3}){r_1}^2 + 2{r_2}^2 + 3{r_3}^2 &< 3{r_1}^2 + 3{r_3}^2 \\ (2-\sqrt{3}) + 2{r_2}^2 &< 3 \\ {r_2}^2 &< \frac{1+\sqrt{3}}{2} \end{aligned}$$

therefore

$$r_2 < \sqrt{\frac{1+\sqrt{3}}{2}} \tag{A.14}$$

Inequality 2

$$\begin{aligned} d(\text{M3,m6}) &< d(\text{m3,M6}) \\ 3{r_1}^2 + 3{r_3}^2 &< 2{r_1}^2 + 2{r_2}^2 \\ 3 + 3{r_3}^2 &< 2 + 2{r_2}^2 \\ 3{r_3}^2 &< 2{r_2}^2 - 1 \\ {r_3}^2 &< \frac{2{r_2}^2 - 1}{3} \end{aligned}$$

therefore

$$r_3 < \sqrt{\frac{2{r_2}^2 - 1}{3}} \tag{A.15}$$

and since the radii cannot be negative, ${r_3}^2 \geq 0$ so

$$r_2 > \frac{1}{\sqrt{2}} \tag{A.16}$$

Inequality 3

$$\begin{aligned} d(\text{m3,M6}) &< d(\text{M2,m7}) \\ 2{r_1}^2 + 2{r_2}^2 &< {r_1}^2 + 4{r_2}^2 + 3{r_3}^2 \\ 2 + 2{r_2}^2 &< 1 + 4{r_2}^2 + 3{r_3}^2 \\ 1 - 2{r_2}^2 &< 3{r_3}^2 \end{aligned}$$

since $r_2 > \frac{1}{\sqrt{2}}$, the left hand side of this inequality will be negative so it is redundant.

Inequality 4

$$\begin{aligned} d(\text{M2,m7}) &< d(\text{m2,M7}) \\ {r_1}^2 + 4{r_2}^2 + 3{r_3}^2 &< (2+\sqrt{3}){r_1}^2 + 2{r_2}^2 + 3{r_3}^2 \\ 1 + 4{r_2}^2 &< 2 + \sqrt{3} + 2{r_2}^2 \\ 2{r_2}^2 &< 1 + \sqrt{3} \\ {r_2}^2 &< \frac{1+\sqrt{3}}{2} \end{aligned}$$

therefore

$$r_2 < \sqrt{\frac{1+\sqrt{3}}{2}} \tag{A.17}$$

This is the same result as $d(\text{P5,P4}) < d(\text{M3,m6})$.

Inequality 5

$$\begin{aligned} d(\text{m2,M7}) &< d(\text{d5,a4}) \\ (2+\sqrt{3}){r_1}^2 + 2{r_2}^2 + 3{r_3}^2 &< 4{r_1}^2 + 4{r_2}^2 \\ 2 + \sqrt{3} + 2{r_2}^2 + 3{r_3}^2 &< 4 + 4{r_2}^2 \\ 3{r_3}^2 &< (2-\sqrt{3}) + 2{r_2}^2 \\ {r_3}^2 &< \frac{(2-\sqrt{3}) + 2{r_2}^2}{3} \end{aligned}$$

therefore

$$r_3 < \sqrt{\frac{(2-\sqrt{3}) + 2{r_2}^2}{3}}. \tag{A.18}$$

This inequality is always satisfied if inequality 2 is satisfied because

$$\sqrt{\frac{2{r_2}^2 - 1}{3}} < \sqrt{\frac{(2-\sqrt{3}) + 2{r_2}^2}{3}}. \tag{A.19}$$

Therfore we are left with

$$\frac{1}{\sqrt{2}} < r_2 < \sqrt{\frac{(1-\sqrt{3})}{2}} \tag{A.20}$$

and

$$r_3 < \sqrt{\frac{2{r_2}^2 - 1}{3}}. \tag{A.21}$$