Automatic Key Detection of Musical Excerpts from Audio

Spencer Campbell

Music Technology Area, Department of Music Research
Schulich School of Music
McGill University
Montreal, Canada

August 2010

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Arts in Music Technology.

Abstract

The proliferation of large digital audio collections has motivated recent research on content-based music information retrieval. One of the primary goals of this research is to develop new systems for searching, browsing, and retrieving music. Since tonality is a primary characteristic in Western music, the ability to detect the key of an audio source would be a valuable asset for such systems as well as numerous other applications.

A typical audio key finding model is comprised of two main elements: feature extraction and key classification. Feature extraction utilizes signal processing techniques in order to obtain a set of data from the audio, usually representing information about the pitch content. The key classifier may employ a variety of strategies, but is essentially an algorithm that uses the extracted data in order to identify the key of the excerpt.

This thesis presents a review of previous audio key detection techniques, as well as an implementation of an audio key detection system. Various combinations of feature extraction algorithms and classifiers are evaluated using three different data sets of 30-second musical excerpts. The first data set consists of excerpts from the first movement of pieces from the classical period. The second data set is comprised of excerpts of popular music songs. The final set is made up of excerpts of classical music songs that have been synthesized from MIDI files. A quantitative assessment of the results leads to a system design that maximizes key identification accuracy.

Abrégé

La prolifération de grandes collections de musique numérique a récemment mené à de la recherche qui porte sur la récupération d'information musical d'après le contenu. Un des principaux objectifs de ce travail de recherche est de développer un nouveau system qui permet de chercher, feuilleter et récupérer de la musique numérique. Étant donné que la tonalité est une des principales caractéristiques de la musique occidentale, l'habilité de détecter la tonalité d'une bande sonre serait un outil indispensable pour un tel system et pourrait mener à maintes autres applications.

Un model de détection de tonalité typique comprend deux principaux éléments : l'identification des structures et la classification des tonalités. L'identification des structures comprend des techniques de traitement de signaux afin d'obtenir de l'information à partir d'une bande sonore, cette information porte typiquement sur le contenu du ton. Un classificateur de tonalité peut servir plusieurs fonctions, mais est essentiellement un algorithme qui traite l'information extraite d'une bande sonore afin d'identifier sa tonalité.

Cette thèse vise à revoir les techniques de détection de tonalité existantes, ainsi que implantation d'un tel système. Diverses combinaisons de classificateurs et d'algorithmes de télédétection et de reconnaissance seront évaluées en utilisant trois différentes bandes sonores d'une durée de 30 secondes. La première bande sonore comprend des extraits de musique classique. La deuxième bande sonore comprend des extraits de musique populaire. La troisième bande sonore comprend des extraits de musique classique créés avec un synthétiseur employant l'interface numérique des instruments de musique (MIDI). Une analyse quantitative des résultats mènera à un système qui optimise la détection de tonalité.

Acknowledgements

There were a number of people who provided support, inspiration or other assistance to help bring this thesis to fruition. First and foremost, I would like to thank my supervisor, Ichiro Fujinaga, for providing guidance, advice, and valuable feedback on all aspects of the work. This simply would not have been possible without his help.

Thank you to the various other people at the McGill Music Technology Area and elsewhere that contributed. Gary Scavone and Philippe Depalle helped me refine my proposal by suggesting improvements. Helene Papadopoulos was extremely helpful in providing valuable feedback on my background chapter. Cory McKay provided assistance with the jMIR software package. Helene Drouin was instrumental in helping me prepare for the submission of my thesis. Gilles Comeau was kind enough to translate my abstract to French and Scott Lucyk provided a great deal of assistance with proofreading and editing the text.

Contents

Chapter 1

Introduction

The proliferation of large music collections has created a need for new technology that allows users to interact with digital libraries in an efficient and meaningful manner. This need has motivated a great deal of research on content-based music information retrieval and indexing, with the aim of allowing users to more effectively locate, index, and browse digital music libraries. In light of the fact that tonality is a primary characteristic in Western music, the ability to automatically extract the tonal key from an audio source would be a valuable component for such systems.

In order to approach the problem of automatically extracting the key from audio, it is worthwhile to first define exactly what *key* is in Western music. According to the Oxford Dictionary of Music, key is "the pitch relationships that establish a single pitch-class as a tonal center or tonic (or key note), with respect to which the remaining pitches have subordinate functions" (Kennedy and Bourne 2006). There are also two primary modes for keys, known as major and minor. The tonic can be any one of the twelve different pitch-classes. So, there are a total of twenty-four distinct keys, if we are considering an equal-tempered scale and enharmonic equivalence (i.e., C# and Db have different names but the same pitch-class).

Key detection, in its simplest terms, refers to the task of automatically identifying which of the twenty-four possible keys a piece of music belongs to. Such identification may use a symbolic representation of music as input, such as a score or MIDI file. Audio key detection, on the other hand, is the more specific case of determining the key of a piece of music from an acoustic input.

This thesis studies the problem of audio key finding and presents a systematic evaluation of several audio key-finding models. The goal is to implement a system that maximizes key identification accuracy. The majority of the excerpts used to evaluate the models are of pieces from the classical period, as this is the standard set by previous studies on the subject (see Appendix A, which shows the data sets used for previous studies on audio key detection). In addition to classical music, a set of excerpts of popular music and excerpts of audio that have been synthesized from MIDI files of classical music are also used.

1.1 Motivation and Applications

As distribution and access to music becomes easier and digital music libraries continue to grow in size, it is becoming increasingly important to find new technologies that allow more effective ways to search, browse, and interact with music.

Several factors have led to unprecedented levels of dissemination and access to digital music, including ubiquity of high-capacity storage and portable media devices, technological improvements in digital audio compression, low-latency networks, and wide-spread availability of digitally distributed music (Cano, Koppenberger, and Wack 2005). It is not uncommon for a home user to have thousands of songs in their personal library. Commercial distributors may have hundreds of thousands of songs in their catalogue.

The predominant method of searching, browsing, and interacting with these collections is based on textual metadata (e.g., artist name, song name). Although expressive metadata can be sufficient for many scenarios, it is subject to several drawbacks. For instance, descriptive metadata is entered by a human and therefore represents an opinion, which makes it difficult to maintain consistency throughout large collections without editorial supervision (Casey et al. 2008).

Content-based music information retrieval (MIR) is an area of research that focuses on creating tools to extract, organize, and structure musical data. One of the fundamental goals of MIR is to provide easier ways to find music, or information about music (Casey et al. 2008). Automatic processing extracts a set of low-level features (e.g., pitch-class profile, spectral flux). The low-level features can then be used to create mid-level representations that contain a higher level of abstraction (e.g., key). Mid-level representations, such as tonal key, are useful in the context of content-based MIR because they provide musically salient information that can be used for other purposes such as audio matching, classification, music recommendation, or further musical analysis (Bello and Pickens 2005). For example, key detection is commonly used as a component in chord recognition systems.

Key finding models also play an important role in research on music perception, specifically with regards to how humans identify the key of music. A study by Temperley and Marvin (2008) used theoretical distributions of pitch-class profiles to generate random melodies and tested whether participants were able to identify the key that was used to generate the melody. They then used several types of key finding models on the same melodies and compared the results to those of the human participants in order to ascertain which model was most representative of how humans perceive key. If audio key-finding systems reach an adequate level of accuracy, then it may also be possible for them to help resolve tonal ambiguity in music.

Audio key detection can also be a practical utility for end-user applications. Mixing is a process used by DJs to create smooth transitions between songs. By way of 'beat matching,' the rhythmic elements of the songs are aligned with one another and then mixed together (Pauws 2006). Many contemporary DJs also use a technique known as 'harmonic mixing' in which the songs being mixed together are either in the same key or a closely related one (e.g., dominant, relative major/minor). In order to use this technique, the key of each song in question must first be known. So an application that automatically identifies the key of every song in a music library greatly facilitates this process.

1.2 Approaching Audio Key Detection

Key detection is the task of automatically identifying which of the twenty-four possible keys a piece of music belongs. There are two primary categories of key finding models: symbolic and audio. The first one uses a symbolic representation of music as input (e.g., a MIDI file), of which the pitch data is entirely disclosed and complete. The second category of key finding models operates on audio signal as input and uses signal processing techniques in order to extract pitch information. As a result, audio key finding has the added challenge of dealing with incomplete and ambiguous pitch data (Chuan and Chew 2007).

A typical audio key detection system is depicted in Figure 1.1. Such systems are comprised of two main elements: feature extraction and key classification. The feature extraction component can also be further subdivided into frequency analysis and pitch-class generation. Frequency analysis is the application of signal processing techniques in order to extract a frequency representation of the audio signal (e.g., FFT, Constant-Q Transform). This information is then used to generate a pitch-class distribution, representing the relative strength of each pitch-class within the signal. Finally, the key classification model uses the pitch-class distribution in order to estimate the key. Creating audio key detection systems with this type of modular design facilitates the isolation and identification of errors in each component so that they can be dealt with accordingly (Chuan and Chew 2005).

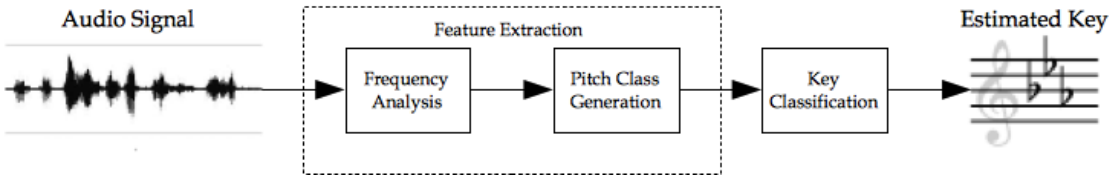

Fig 1.1: Example of a typical audio key finding system.

Signal processing techniques such as the fast Fourier transform provide an accurate and reliable means for generating a frequency representation of the audio signal. As such, most of the errors encountered during the feature extraction stage can be attributed to problems with the pitch-class generation algorithm. The following are some of the common errors encountered with pitch-class generation (Chuan and Chew 2005):

- **Tuning Variations**
  Audio recordings can sometimes contain sounds that are produced by mistuned instruments. Pitch-class generation algorithms that do not account for this possibility and use direct frequency to pitch conversions may lead to inaccurate pitch-class distributions.

- **Low Frequency Resolution**
  Humans perceive pitch on an approximately logarithmic scale. So the frequency difference between two low notes is less than the frequency difference between two high notes. As a result, the pitch-class generation algorithm must have a finer resolution for lower frequencies in order to discern the difference between adjacent pitch-classes.

- **Effect of Partials**
  In addition to the fundamental frequency, most sound waves produced by instruments also contain partials that are closely related to the harmonic series. These partials can affect the resulting pitch-class distribution.

Although there are many different types of models that have been used for key classification, several types of errors are commonly encountered. These errors are often the result of the identified key having a similar pitch-class distribution to that of the

correct key (Chuan and Chew 2005). Several common types of errors for key classification are as follows:

- **Perfect 5th Errors**

  The dominant key has only one difference in the pitch-class set from that of the tonic key, so there is a great deal of overlap in their distributions. It is often also the strongest partial, second to the actual tone.

- **Relative Major/Minor Errors**

  Relative keys have identical pitch-class sets but have different theoretical distributions. This makes distinguishing the difference very difficult in certain cases.

- **Parallel Major/Minor Errors**

  Parallel keys have the same tonic but are in different modes. A pitch-class distribution with strong tonic and dominant classes, but ambiguity for the rest of the distribution may lead to this type of error.

Audio key detection is also highly affected by the type of music that is being analyzed. The degree of tonal complexity varies greatly depending on the type of music. It is common for the key to change within a piece, which is known as a modulation. Before approaching audio key detection for music with modulations, the errors discussed in this section for single keys must first be addressed and resolved. As such, this thesis will deal exclusively with short musical excerpts in which no modulations exist.

1.3 Thesis Structure

The remainder of this thesis is organized as follows:

- **Chapter 2: Background**

  The scientific background and concepts relevant to the context of this thesis are presented. This includes an introduction to tonality as well as previous research on key detection, both from symbolic and audio sources.

- **Chapter 3: Software Design**

  The details of the software implementation created for this thesis are given. This includes the types of signal processing parameters, feature extraction algorithms, and classifiers that were used.

- **Chapter 4: Description of the Data**

  Two types of data were used to train and evaluate the software: musical excerpts and pitch-class templates. This chapter presents the details for both of these.

- **Chapter 5: Experimental Setup**

  The experiment used to parameterize and evaluate the software consisted of three phases. This chapter describes the details for each of these phases.

- **Chapter 6: Results and Discussion**

  In this chapter, the results of each phase of the experiment are reported and comments on the findings are presented.

- **Chapter 7: Conclusions**

  A review of the experiment, results, and findings is presented. Comments on the outcome and future research for the field are also presented.

Chapter 2

Background

2.1 Introduction

The purpose of this chapter is to introduce some of the concepts relevant to the context of this thesis, as well as to present previous research efforts on key detection. Section 2.2 briefly introduces some of the basic music theory for tonality and key. Section 2.3 goes on to present a scientific background on both symbolic and audio key detection.

2.2 Tonality and Key

Tonality has been thoroughly studied from many different perspectives, including music theory, music history, psychoacoustics, and music psychology (Vos 2000). As a result of this multidisciplinarity, definitions for the term vary a great deal in the literature, depending on the context. According to Hyer (2002), "Tonality most often refers to the orientation of melodies and harmonies towards a referential (or tonic) pitch-class. In the broadest possible sense, however, it refers to systematic arrangements of pitch phenomena and relations between them." Music-theory or cognitive-based geometrical models have been devised in order to represent these relationships (Noland and Sandler 2009). For example, the Harmonic Network or *Tonnetz* is a model developed by Euler that uses two-dimensional geometry to represent the harmonic relationships between the different pitch-classes (Harte et al. 2006). Within the planar representation, pitch-classes

that have stronger harmonic relationships (e.g., perfect fifths) are located closer to one another, as depicted in Figure 2.1.

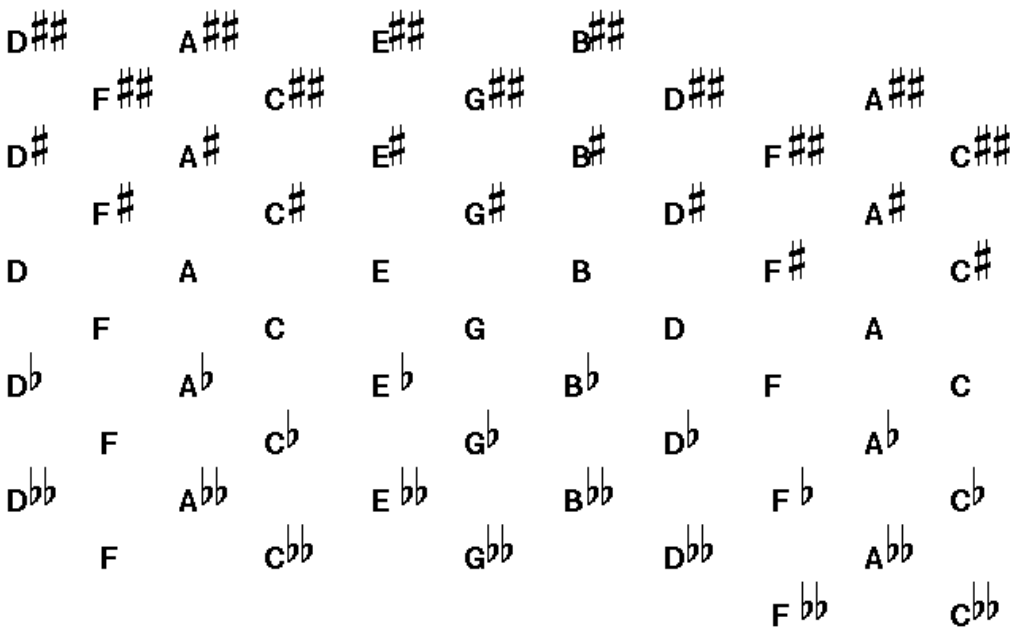

Fig. 2.1: The Harmonic Network or *Tonnetz*. Horizontally adjacent pitch-classes represent perfect fifth intervals, diagonally adjacent pitch-classes represent major/minor third intervals, and vertically adjacent pitch-classes represent semitone intervals (from Sapp 2006).

If octave equivalence is assumed (i.e., $A_1 = A_2$), then the plane of the Tonnetz can be represented as a tube with fifth intervals forming a helix on its surface. If the tube is then arranged such that the helix has the major third intervals aligned above one another, then we arrive at Chew's (2000) Spiral Array model (Harte et al. 2006). The model, illustrated in Figure 2.2, maps pitches to points on the spiral, such that pitch-classes with prominent harmonic relationships are in close proximity (e.g., chord pitches, pitch-classes for a key).

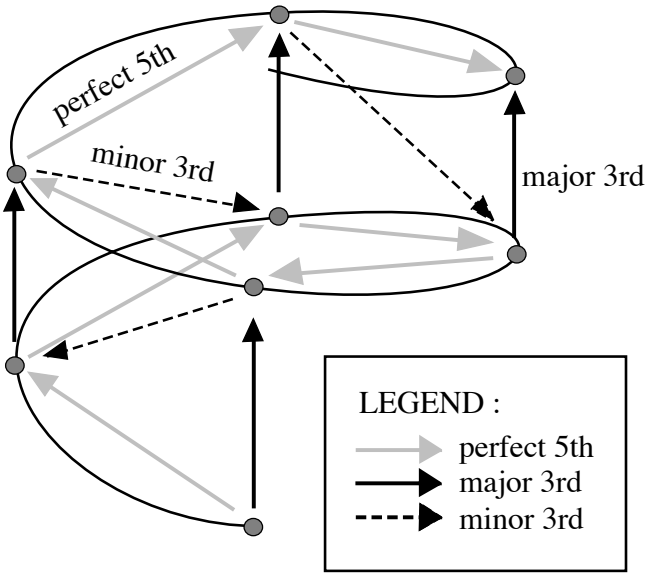

Fig. 2.2: Representations of the perfect 5th, major 3rd, and minor 3rd intervals in the Spiral Array Model (from Chew 2000).

In music theory, the terms *key* and *tonality* are often used interchangeably. However, for the context of this thesis we will primarily use the term *key*, which we define as one particular *tonic* and a *mode* (Hyer 2001). The *tonic* is the first and most stable pitch-class within the diatonic collection for the key. The *mode* governs both the melody type and scale and there are two basic modes: major and minor.

The major and natural minor are the two primary types of diatonic scales, which consist of seven notes with five whole tone intervals and two semitone intervals. The only difference between the two scales is the step sizes for the various scale degrees. Table 2.1 provides a legend of the scale degrees for the major and minor modes and Figure 2.3 shows the scale degrees and step sizes for C major and A natural minor scales.

Major		Minor	
I	Tonic	I	Tonic
II	Supertonic	II	Supertonic
III	Mediant	III	Mediant
IV	Subdominant	IV	Subdominant
V	Dominant	V	Dominant
VI	Submediant	VI	Submediant
VII	Leading Tone	#VI	Raised Submediant
VIII	Tonic	VII	Subtonic
		#VII	Leading Tone
		VIII	Tonic

Table 2.1: The scale degrees for the major and minor modes.

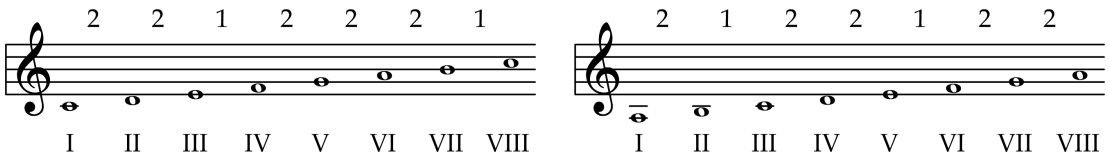

Fig. 2.3: The C major scale (left) and the A minor natural scale (right). Step sizes (in semitones) are shown above and the scale degrees are shown below.

In addition to the natural minor scale, there are two other types of minor scales: the harmonic minor and melodic minor. The harmonic minor scale is equivalent to the natural minor except the 7th degree is raised by one semitone such that the interval between the 6th and 7th degrees forms an augmented second. The ascending melodic minor scale has both the 6th and 7th scale degrees raised by one semitone and the descending melodic minor scale is equivalent to the natural minor scale. The harmonic minor and ascending melodic minor scales are depicted in Figure 2.4.

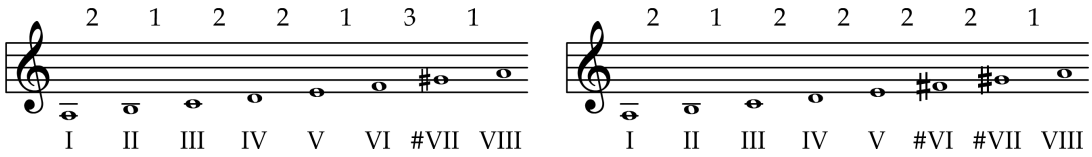

Fig. 2.4: The harmonic minor scale (left) and the ascending melodic minor scale (right). The descending melodic minor scale is identical to natural minor scale in Figure 2.3. Step sizes (in semitones) are shown above and the scale degrees are shown below.

If we consider enharmonic equivalence (i.e., C# and Db have different names but the same pitch-class), then we have a total of twenty-four distinct keys: one for each pitch-class in the major and minor modes. Each key also has harmonic relationships to other keys. Relative major/minor keys have the same pitch-class set but different modes (e.g., C major and A minor). Parallel major/minor keys have the same tonic but different modes (e.g., C major and C minor). Two keys separated by a perfect 5^{th} (e.g., C major and F major) are also closely related since they share all but one pitch-class in their diatonic collection.

2.3 Key Detection

There have been many attempts to create key-finding models in the literature and these can be separated into two distinct groups: symbolic key detection models and audio key detection models (Temperley and Marvin 2007). The first group deals with symbolic data, such as a score or MIDI file. In this case, the input is always complete and free of any ambiguity with regards to pitch and duration of events. The second category operates directly on an audio signal, requiring the extraction of data to a format that can be interpreted by the key-finding algorithm. The scope of this thesis is only concerned with the latter category of key-finding models. As such, this section presents some of the more

notable research on symbolic key detection as well as a more comprehensive review of previous research efforts on audio key detection.

2.3.1 Symbolic Key Detection

The first notable attempt to create a model that addressed the problem of key-finding was that of Longuet-Higgins and Steedman (1971). They observed that pitch-classes belonging to a key have relatively small Euclidian distances from one another on the Harmonic Network (see Figure 2.1). This observation formed the basis for their model, which used a shape-matching algorithm in order to identify the key (Chew 2000). For the purposes of the algorithm, a shape defines the mode of the key (i.e., all major keys have the same shape and all minor keys have the same shape) and location of the shape determines the tonic. Figure 2.5 shows two examples of "shapes" outlined in the Harmonic Network. The algorithm processes the notes of a melody in the order in which they appear. With the appearance of each note, the keys corresponding to shapes that do not contain that note are eliminated from consideration. If the end of the melody is reached and only one key remains, then it is chosen. If, however, more than one key remains, then a tonic-dominant rule is utilized[1]. In the case where no keys remain, then the key whose tonic is the first note in the melody is chosen[2]. The model was evaluated on the 48 fugue subjects of Bach's *Well-Tempered Clavier*. Although it correctly identified the key in every case, it should be noted that these pieces are relatively simplistic in their harmonic structure. Temperley and Marvin (2008) point out that it is relatively easy to find examples of melodies that would produce an incorrect result.

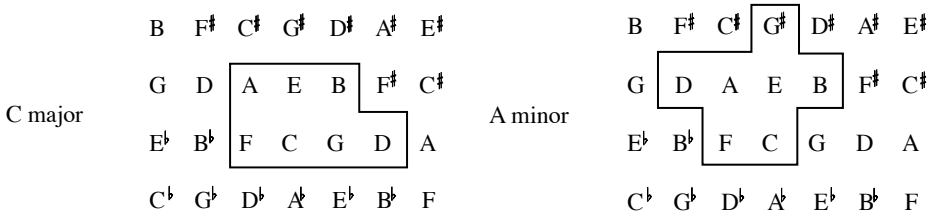

Fig. 2.5: Examples of shapes outlined in the Harmonic Network for Longuet-Higgins and Steedman's (1971) shape-matching algorithm (from Chew 2000). C major on the left and A minor (harmonic) on the right.

One of the most significant advances in symbolic key detection was made with the algorithm proposed by Krumhansl and Schmuckler, known as the Krumhansl-Schmuckler (K-S) algorithm (Krumhansl 1990). The approach is based on the set of key profiles derived from the experiments of Krumhansl and Kessler (1982). The key profiles, shown in Figure 2.6, are supposed to represent the ideal distribution of pitch-classes within a key. A key profile consists of a twelve-dimensional vector, where each value of the vector represents the relative stability of the corresponding pitch-class within the given key. There are 24 key profiles, one for each of the 12 major and minor keys.

The algorithm first calculates an *input-vector* from a MIDI file, which is a normalized representation of the total duration of each pitch-class within the piece. A correlation is then calculated between the input-vector and each of the 24 key profiles. The key corresponding to the profile with maximum correlation to the input-vector is then chosen[3].

The basic assumption of the K-S model is that the generated input-vector will correspond closely to the correct key profile. This assumption may be correct in many cases (e.g., when there is a strong presence of notes in the tonic-triad). However, there is an abundance of examples in which this assumption will lead to an incorrect result. In an effort to overcome these limitations, Temperley (1999) proposed several modifications to the K-S algorithm. Firstly, he suggests that note durations be ignored altogether, such that the values of the resulting input-vector are binary[3]. Secondly, he makes slight

modifications to the key profiles in order to help distinguish between keys with very similar pitch-class distributions.

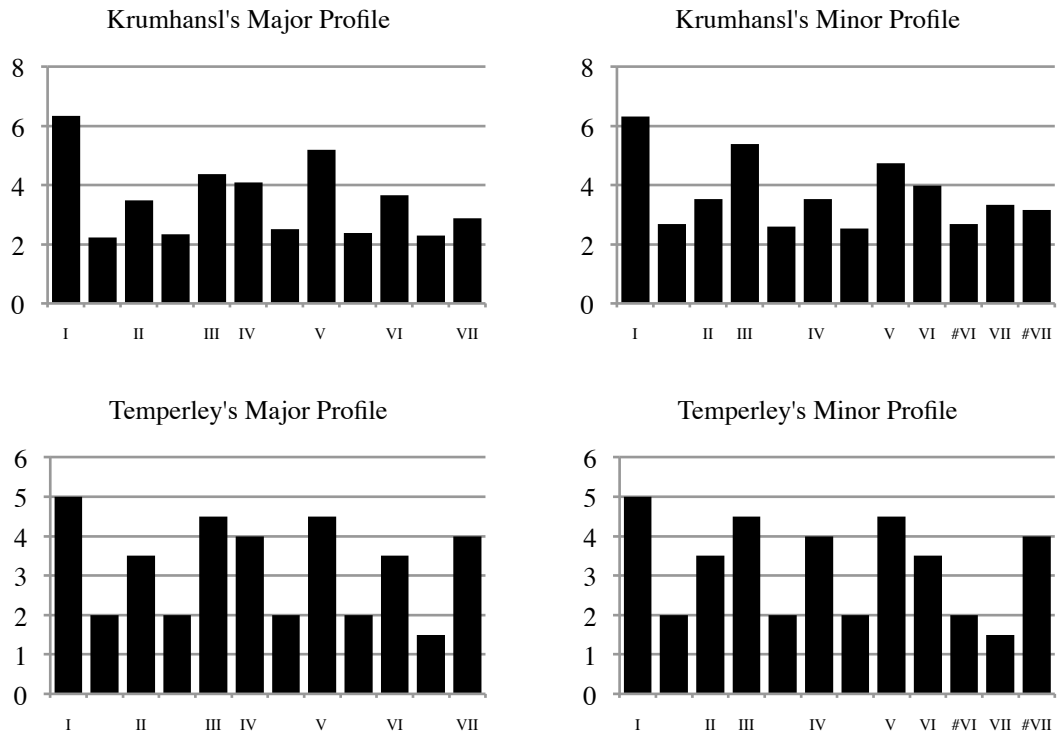

Fig. 2.6: Krumhansl and Kessler's (Krumhansl 1990) major and minor key profiles (top). Temperley's (2001) major and minor key profiles (bottom).

Based on the Spiral Array model (see Section 2.2), Chew (2000) proposes the Center of Effect Generator (CEG) key-finding method. In the CEG algorithm, a passage of music is mapped to a point within the three-dimensional space, known as the *Center of Effect*, by summing all of the pitches and determining a composite of their individual positions in the model. The algorithm then performs a nearest-neighbor search in order to locate the position of the key that is closest to the Center of Effect. The "closest key" can be

interpreted as the global key for the piece, although the proximity can be measured to several keys, which allows for tonal ambiguities.

Temperley and Marvin (2007) note that the vast majority of key-finding models take a *distributional* view, which postulates that listeners identify the key of a piece of music based solely on the distribution of pitch-classes. Based on this view, Temperley (2007) proposes a key-finding system that implements a probabilistic model. Within this model, key profiles are generated for each key, representing the probability of each pitch-class appearing. The profiles are constructed by performing a statistical analysis of a corpus of music that extracts the overall "presence" of each pitch-class. For example, the major key profile created from the opening movements of Mozart and Haydn's string quartet, is shown in Figure 2.7. Once the key profiles have been created, the model can estimate the key of a melody by calculating the probability of a melody appearing if it is in a particular key, for each of the 24 possible keys, and choosing the key with the highest value. The system was evaluated on a corpus of 65 European folk songs and had a key recognition rate of 86.15%.

Madsen and Widmer (2007) argue that in addition to the pitch-class distribution, the order of notes appearing in a piece of music may also help determine the key. They propose a key-finding system that incorporates this temporal information by also analyzing the distribution of intervals within a piece of music. *Interval Profiles* are 12x12 matrices representing the transition probability between any two scale degrees. The profiles are then learned from key annotated data for all 24 keys. Using a corpus of 8325 Finnish folk songs in MIDI format, the system was trained using 5550 songs and evaluated with the remainder. A comparison was also performed between the use of Interval Profiles and several types of pitch class profiles. The maximum key recognition rate using the Interval Profiles was 80.2%, whereas the maximum recognition rate using pitch class profiles was 71%.

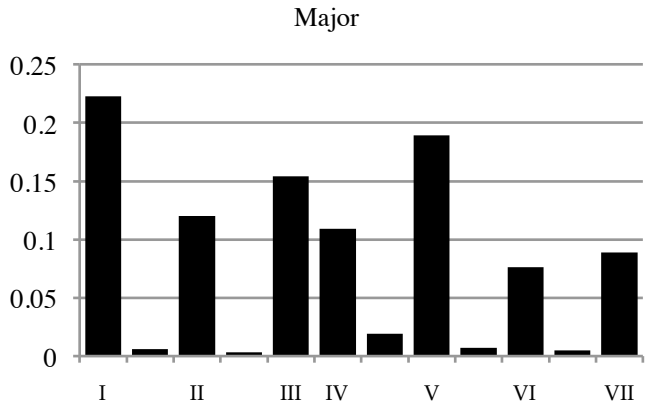

Fig. 2.7: The major key profile generated from string quartets by Mozart and Haydn (Temperley and Marvin 2007).

2.3.2 Audio Key Detection

In contrast to the extensive body of literature on symbolic key detection, there exists relatively little documented research on audio key finding. However, there appears to be four distinct types of approaches for audio key detection methods: pattern matching and score transcription methods, template-based methods, geometric models, and methods based on chord progressions or Hidden Markov Models (HMMs).

The earliest attempts at audio key detection focused on using pattern matching techniques or partial transcription of the audio signal to a score representation. The latter would seem to be the most intuitive approach, as it theoretically would allow for the application of exiting symbolic key-finding methods for audio.

The vast majority of audio key detection models circumvent the need for score transcription by implementing template-based algorithms. These models are based on correlating the global distribution of pitch-classes for a piece of music with representative templates for each key. Temperley and Marvin (2007) call this the *distributional* view for key finding. A typical system will calculate a pitch-class distribution feature[4],

representing the relative global intensity of each pitch-class within the piece. The pitch-class distribution feature is subsequently compared with pitch-class templates, representing the ideal distribution of pitch-classes for each key. The key corresponding to the template with maximum correlation to the pitch-class distribution feature is then chosen.

More recently there have been attempts to build audio key detection systems that implement Hidden Markov Models (HMMs). These types of systems will often also incorporate some form of chord detection or local key estimation (i.e., detection of modulations).

See Appendix A for a table that summarizes the audio key detection systems reviewed in this section.

2.3.2.1 Pattern-Matching and Score Transcription Methods

Leman (1991, 1995b) proposed one of the first models for audio key detection. The system is based heavily on a model of the human auditory system and consists of two stages. The first step is to extract local tone centers in a bottom-up manner for the piece of music. The second stage of the system uses a pattern-matching algorithm to compare the extracted tone center data with predetermined templates derived from self-organizing maps.

Izmirli and Bilgen (1994) proposed a system for audio key finding that implements partial score transcription in combination with a pattern-matching algorithm. In the first stage of the system, the fast Fourier transform (FFT) function is used in order to convert a single-part, melodic audio input into a sequence of note intervals with associated onset times. A second stage then employs a finite-state automata algorithm to compare the note sequences with predetermined scale patterns. The model then outputs a *tonal context* vector, where each element is known as a *tonal component*. Each tonal component represents the extent of any given scale usage within the melody for the corresponding location in time. In essence, the model provides a time-dependent tonal context for the

input melody and not an explicit estimation of the global key. Figure 2.8 depicts an example of the tonal context vector.

In 1996, Izmirli and Bilgen (1996) went on to extend their system to handle an unrestricted number of simultaneous input melodies. The first stage of the model uses a constant Q transform (CQ-transform) in order to map the input signal to the frequency domain, as opposed to the FFT function used in their earlier version (Brown 1991)[5]. A simple peak-selection algorithm is then applied in order to produce a set of notes for each time step. The second stage of the system remains roughly the same as their previous implementation, but is adapted to process simultaneous occurrence of multiple notes.

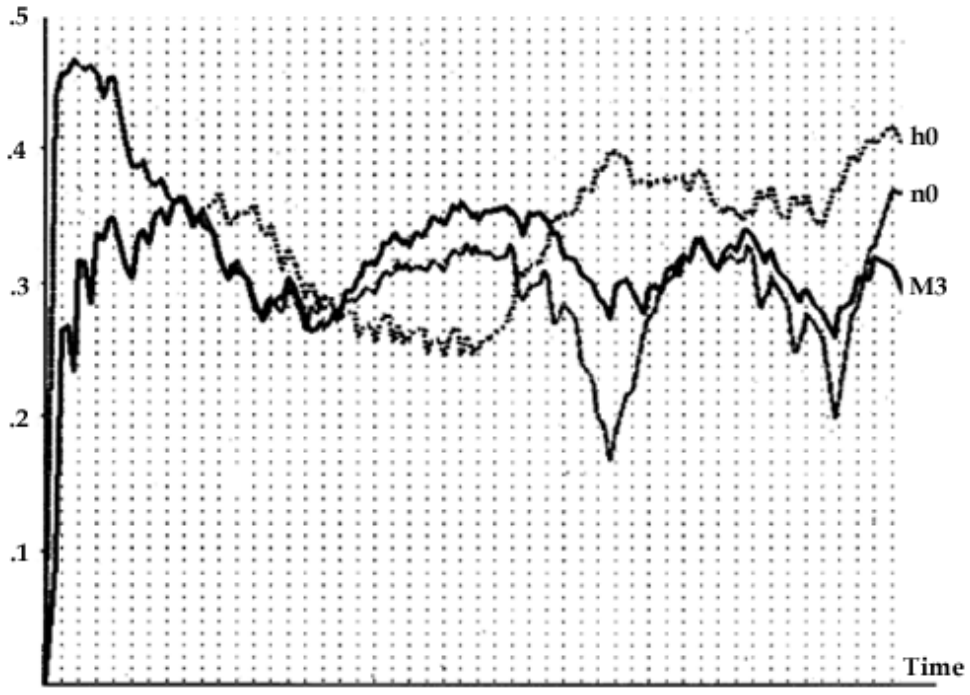

Fig. 2.8: The tonal context evolution of the three most prominent tonal components for an example melody. The x-axis denotes time and the y-axis represents the strength of the tonal components (between 0 and 1). h0 = harmonic A minor, n0 = natural A minor, and M3 = C major (from Izmirli and Bilgen 1994).

2.3.2.2 Template-Based Methods

A template-based audio key detection system typically consists of two stages. The first stage extracts a pitch-class distribution feature from the audio signal, representing the relative strength of each pitch-class within the signal. The second stage uses some form of algorithm to compare the pitch-class distribution feature with pitch-class templates in order to estimate the key. Pitch-class templates are twelve-valued vectors that represent the ideal distribution of pitch-classes for a given key.

Gómez (2006a) points out that the nomenclature for pitch-class distribution features varies a great deal in the literature: pitch pattern (Leman 2000), pitch-class profile (Fujishima 1999), Harmonic Pitch Class Profile (Gómez 2005), constant-Q profile (Purwins et al. 2000), pitch profile (Zhu et al. 2005), and chromagram (Pauws 2004)). Although the name and implementation details of the pitch-class distribution features may vary in the approaches described in this section, we will from here on refer to these with the general term of *pitch-class distributions*, for the sake of simplicity and clarity. The process of creating the pitch-class distributions from a frequency domain representation of the signal will be called *pitch-class generation* (Chuan and Chew 2007).

Similarly, there is a lack of consistency in the literature for the term used to describe pitch-class templates (e.g., key profiles, pitch-class profiles), so we will from here on refer to these only as *pitch-class templates*. There are three basic categories of pitch-class templates used for key detection: music theory-based templates, cognitive-based templates, and statistics-based templates (Noland and Sandler 2009). Music theory-based templates are constructed using some form of musical knowledge (e.g., a template with all diatonic pitch-classes having a value of one and all chromatic pitch-classes having a value of zero). Cognitive-based templates are obtained through studies on music perception and cognition (Krumhansl and Shepard 1979; Krumhansl 1990) and represent the perceptual importance of pitch-classes within a key. Statistics-based templates are derived from an empirical analysis of a corpus of music, and represent the average pitch-class distributions for that particular corpus (Gómez 2006; Noland and Sandler 2007). Pitch-class templates

can also be hybrids of these three categories (e.g., templates constructed from the cognitive experiments but weighted with statistical data).

Purwins, Blankertz, and Obermayer (2000) proposed a model based on the probe tone experiments conducted by Krumhansl and Shepard (1979)[6]. The system employs the CQ-transform to extract a pitch-class distribution from the audio signal. A fuzzy distance algorithm is then used to compare the pitch-class distribution with the cognitive-based templates. The system is able to track the key over time and thus is capable of identifying modulations in the music. An evaluation was performed using Chopin's C minor prelude, Op. 28, No. 20 and was fairly successful at tracking the key, although no quantitative results were explicitly reported.

Pauws (2004) implemented an audio key detection system that adopted the cognitive-based templates directly from Krumhansl (1990). The system incorporates signal processing techniques designed to improve the salience of the extracted pitch-class distribution. The pitch-class distribution is then used as input to the *maximum-key profile* algorithm in order to identify the key. The model was tested on a corpus of 237 classical piano sonatas, with a maximum key identification rate of 66.2%.

Van de Par et al. (2006) present an extension to the work of Pauws (2004) in which they utilize three different temporal weighting functions in the calculation of the pitch-class templates. This results in three different templates for each key. Similarly, during the actual key detection, three different pitch-class profiles are extracted, one for each temporal weighting function. Each of the three pitch-class profiles is then correlated with the corresponding templates and a final correlation value is calculated from the combined values. The system was evaluated using the same corpus of 237 classical piano sonatas as Pauws (2004) and received a maximum key recognition rate of 98.1%.

While most template-based audio key-finding systems utilize some form of Euclidian distance to compare pitch-class distributions with templates, Martens et al. (2004) implemented a model using a classification-tree for key recognition. The classification-tree was trained using 264 pitch-class templates that were constructed from Shepard sequences and chord sequences of various synthesized instruments[7]. They conducted an

experiment that compared the performance of the tree-based system with a classical distance-based model using two pieces: “Eternally” by Quadran and “Inventions No. 1 in C major” by J. S. Bach. The results led them to favor the classification-tree system due to its ability to stabilize key estimations over longer time periods. They also noted the advantage of being able to tune the system for specific types of music by using a corresponding category of music to train the model.

Gómez and Herrera (2004a) noted that the majority of audio key detection models developed up until 2004 were based on perceptual studies of tonality, which they called *cognition-inspired* models. They performed an experiment in which they directly compared an implementation of a cognition-inspired model with several machine-learning algorithms for audio key determination. The cognition-inspired model was based on the K-S algorithm but extended to handle polyphonic audio input. Numerous machine-learning techniques were implemented, including binary trees, Bayesian estimation, neural networks, and support vector machines. The various algorithms were evaluated on three criteria: estimating the “key note” (i.e., tonic), the mode, and the “tonality” (i.e., tonic and mode). A corpus of 878 excerpts of classical music from various composers was used for training and testing. The excerpts were split into two sets: 661 excerpts for training and 217 excerpts for evaluation. The results, summarized in Figure 2.9, show that for the case of estimating the “tonality,” the best machine learning algorithm (a multilayer perceptron, neural network) outperforms the cognition-inspired model, but a combination of the two approaches produces the best results.

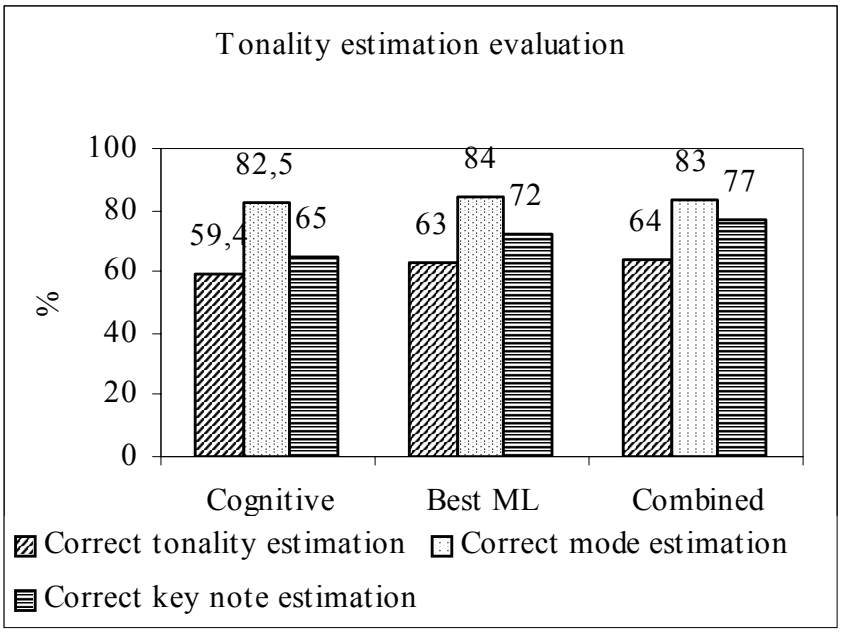

Fig. 2.9: A summary of the results of Gómez and Herrera's experiment comparing the performance of cognition-inspired models versus machine learning algorithms for audio key finding (from Gómez and Herrera 2004).

Chuan and Chew (2005c) point out the importance of segregating the sources of errors in audio key detection systems between the pitch-class generation and the key identification stages. They formulate hypotheses for sources of errors during the pitch-class generation stage and propose a modified algorithm that uses fuzzy analysis in order to eliminate some of the errors. The fuzzy analysis method consists of three main components: *clarifying low frequencies*, *adaptive level weighting*, and *flattening high and low values*[8]. They performed a direct comparison of the fuzzy analysis key-finding system with two other models: a peak detection model and a MIDI key-finding model. The evaluation utilized excerpts from a corpus of 410 classical music MIDI files, where only the first 15 seconds of the first movement was considered. The fuzzy analysis and peak detection algorithms operated on audio files that were synthesized using Winamp, and the

MIDI key-finding model operated directly on the MIDI files. The maximum key identification rates for the peak detection, fuzzy analysis, and MIDI key-finding models were 70.17%, 75.25%, and 80.34%, respectively. It is not surprising that the MIDI key-finding model had the best overall performance, considering that it operates on unambiguous and complete pitch data. However, the results do indicate that fuzzy analysis provides an effective means of improving pitch-class generation for audio key detection systems.

Signifying a recently increased interest in audio key detection, the 2005 Music Information Retrieval Evaluation eXchange (MIREX '05) featured an audio key-finding competition. Six groups participated in the event (Chuan and Chew 2005b; Gómez 2005; Izmirli 2005b; Pauws 2005; Purwins and Blankertz 2005; Zhu 2005), submitting state-of-art key-finding systems that were evaluated using a formalized scoring procedure. All of the systems were template-based and used some form of pitch-class distribution feature in combination with a key-finding model. However, the type of pitch-class templates (e.g., cognitive-based, music theory-based, statistics-based), feature extraction algorithms, and key models were varied amongst the participants. Table 2.2 summarizes the implementation details of the algorithms entered, Table 2.3 show the results of the evaluation, and Table 2.4 describes the scoring procedure that was used.

Participant	Feature Extraction		Key-finding Algorithm	Pitch-Class Template
	Frequency Analysis	Pitch-Class Generation		
Chuan & Chew	FFT	Peak detection with fuzzy analysis	Pitch spelling (maps to Spiral Array) combined with Center of Effect Generator algorithm	Music theory-based: geometric representation in the Spiral Array
Gómez	FFT	Harmonic Pitch Class Profiles with 36 bins for tuning correction	Maximal correlation with templates	Cognitive-based: modified tone profiles TM and Tm, proposed by Temperley (1999)
Izmirli	FFT	Multiple summary chroma vectors of varying window lengths	K-S correlation with confidence values for each summary chroma vector	Cognitive/Statistical/Music theory-based: composite of Temperley's (2001) tone profiles (cognitive-based) and diatonic profiles (music-theory based), combined with extracted frequency data from real instrument sounds (statistics-based)
Pauws	FFT	Subharmonic summation used to create chroma spectrum	Unknown	Statistics-based: derived from training data
Purwins & Blankertz	CQ-transform	Pitch class distributions with 36 bins for tuning correction	Maximal correlation with templates	Statistics-based: derived from training data
Zhu	CQ-transform	Pitch content classified as mono, chord or other	Rules based on training data	Music theory/Statistics-based: Music knowledge is used to create a set of rules and the parameters are derived from the training data

Table 2.2: Summary of the implementation details for the systems entered in the MIREX '05 audio key-finding competition (Chuan and Chew 2005b).

Relation to correct key	Points
Exact match	1
Perfect fifth	0.5
Relative major/minor	0.3
Parallel major/minor	0.2

Table 2.3: Summary of the metrics system used for the MIREX '05 audio key finding evaluation. Summing the total number of points and dividing by the total number of instances in the test set gives the percentage score.

Participant	Composite Percentage Score	Percentage Score	
		Winamp	Timidity
Izmirli	89.55%	89.4%	89.7%
Purwins & Blankertz	89.00%	89.6%	88.4%
Gómez (start)	86.05%	86.4%	85.7%
Gómez (global)	85.90%	86.0%	85.8%
Pauws	85.00%	84.3%	85.7%
Zhu	83.25%	85.2%	81.3%
Chuan & Chew	79.10%	80.1%	78.1%

Table 2.4: Summary of the results for the MIREX '05 audio key-finding competition. Two data sets were used for evaluation: Winamp synthesized audio and Timidity with Fusion soundfonts synthesized audio. The percentage scores are calculated using the system of metrics that was created for the competition.

Izmirli (2005a) conducted further experiments using the model that he submitted to the MIREX '05 audio key-finding competition. He evaluates the effectiveness of different types of pitch-class templates in combination with varying durations of analysis for the input signal. Two different methods are used to implement the template calculation model: the first is based purely on the spectral content of the signal and the second is a chroma-based representation that extrapolates on the spectral content. A corpus of 85

pieces from various composers, primarily from the common practice period is used to evaluate the system. The results of the experiment showed that the maximum key recognition rate of 86% was achieved using a chroma-based representation that combined the Temperley and Diatonic profiles.

Izmirli (2006) conducts an additional experiment in which he compares the model submitted to MIREX '05 with another model that utilizes dimensionality reduction. The goal is to determine the optimal number of dimensions to be used for the key-finding problem, as opposed to reducing the computational cost. An evaluation is performed using the two models on a corpus of 152 pieces from the classical period. It is shown that the performance of the key-finding system was not significantly hindered when using 6 dimensions instead of 12. The model using 6 dimensions received a composite score of 88.7%, whereas the reference model received an 88.9% composite score.

Izmirli (2007) points out that the majority of key-finding models focus on identifying the main key of a piece, as opposed to segmenting the audio based on modulations in order to perform local key-finding. A new system is proposed that uses non-negative matrix factorization in order to segment an audio signal based on modulations. A series of windowed pitch-class distributions are calculated and segments are identified based on this technique. The same correlational model as was used in (Izmirli 2005a) is employed to identify the key of any given segment. Three different data sets were used to evaluate the model: 17 pop songs with at least one modulation each, 152 excerpts from the initial portion of classical music pieces, and 17 short excerpts of classical music containing at least one modulation each. The maximum accuracy of the segmentation-based approach was 82.4%, for the pop data set.

Gómez (2006b) presents an exhaustive investigation on tonal descriptors of audio signals in her Ph.D. dissertation. She presents a thorough analysis of many of the pertinent aspects of audio key detection, including audio feature computation, evaluation strategies, and various template-based models for tonality. An evaluation of different audio key detection methods was performed for various genres of music. The study led to the conclusion that in most cases models that use cognitive-based templates outperform

those that utilize statistics-based templates. Furthermore, the results of the experiment indicated that the performance of any particular audio key detection model is heavily dependent on the genre of music that is being analyzed.

Zhu et al. (2005) propose an audio key-finding system that utilizes the CQ-transform and detects the key in two distinct steps: diatonic scale root estimation and mode determination. The system is evaluated on a corpus of 60 pop songs and 12 classical music recordings, using only the first 2 minutes of each piece. The correct scale root was detected for 91% of the pop songs but only 50% of the classical music pieces. The rate of successful mode determination for the pop songs was 90% and 83.3% for the classical pieces.

Zhu and Kankanhalli (2006) went on to further investigate the effects of mistuned recordings[9] and the effect of noisy, percussive sounds on pitch-class generation. They conducted an analysis of 185 classical and 64 popular music excerpts and determined that many of the recordings contained tuning errors. They also point out that percussive sounds should be disregarded within an audio key detection system, because they are not pitched and therefore do not contribute to tonality. However, these percussive elements still have an effect on the frequency domain representation of the signal, contributing energy to the bins used to generate the pitch-class distributions. As such, they affect the salience of the pitch-class distributions, and in turn the accuracy of key identification. They propose a system to improve on these limitations. A tuning pitch determination algorithm is used to detect a mistuned recording and adjust the pitch-class distribution accordingly. They also use consonance filtering in order to discard some of the frequency contributions from noisy, percussive elements in the signal. Figure 2.10 shows the output of the extracted note partials, with and without the consonance filtering. They perform an experiment in which they compare the proposed system with an earlier model that does not account for mistuned recordings or percussive instrumentation. They claim that the results of the experiment indicate that the use of tuning correction and consonance filtering improve the key identification accuracy.

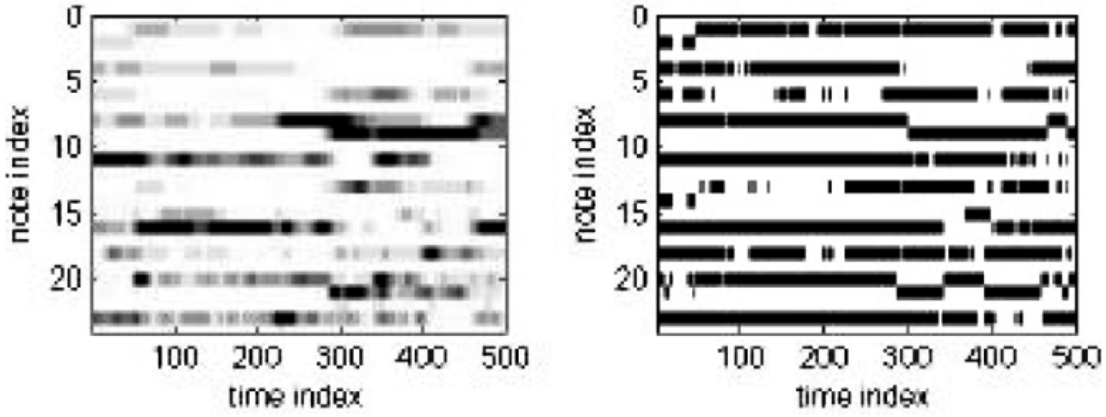

Fig. 2.10: Comparison of note partials with consonance filtering (right) and without consonance filtering (left) (from Zhu and Kankanhalli 2006).

2.3.2.3 Geometric Models

Chuan and Chew (2005a) present an audio key detection system that utilizes the Spiral Array Center of Effect Generator (CEG) algorithm (Chew 2000; Chew 2001). The system uses the standard FFT to extract pitch-class and pitch strength information from the audio signal, which is then mapped to a 3-D point in the Spiral Array. A nearest-neighbor search is then used in the Spiral Array in order to estimate the key. A comparison of this model is then made with two other template-based audio key-finding approaches: the K-S method and Temperley's modified K-S method (templates shown in Figure 2.6). All three systems were evaluated on a corpus of 61 excerpts of Mozart symphonies synthesized from MIDI. Their Spiral Array CEG model received a maximum key recognition rate of 96%, while the K-S and Temperley's modified K-S models had a maximum recognition rate of 80% and 87%, respectively.

Chuan and Chew (2007) go on to use their model in order to perform a systematic analysis of the various components of audio key detection systems, with the goal of identifying elements critical to system design. They observe that most previous evaluations of audio key-finding systems only report the overall key detection accuracy,

as opposed to a more detailed analysis of the performance of the individual system components or the effect of the type of music that is used for evaluation. They first propose a basic system using the fuzzy analysis and spiral array center of effect (FACEG) algorithm (Chuan and Chew 2005c), and evaluate it using three different key determination policies: the nearest-neighbor (NN), the relative distance (RD), and the average distance (AD). The basic system is then evaluated using excerpts of the initial fifteen seconds of 410 classical music pieces, ranging in styles from Baroque to Contemporary. The results show the average accuracy for each of the three key determination policies, revealing that the AD policy performs the best. However, analysis of the results also reveals some of the strengths and weaknesses of each policy, as well as the effect of the musical genre on key identification accuracy. They go on to propose three extensions to the basic system: the modified spiral array (mSA), fundamental frequency identification (F0), and post-weight balancing (PWB). Five different permutations using the three extensions are evaluated in a second case study using Chopin's 24 Preludes. An in-depth, qualitative, and quantitative analysis of the results also provides insight on how and why each of the extensions can be used to improve audio key identification accuracy for specific situations. The basic audio key-finding system and its three extensions are depicted in Figure 2.11.

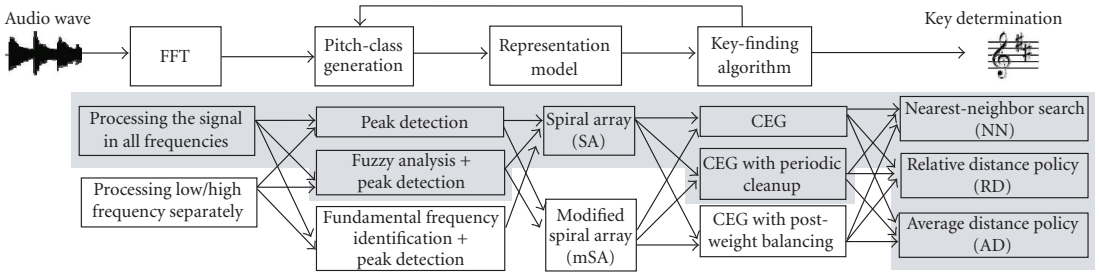

Fig. 2.11: A typical audio key finding system (top). The basic audio key-finding system (grey) and extensions (bottom) (from Chuan and Chew 2007).

Harte et al. (2006) point out that if enharmonic and octave equivalence are considered, this has the effect of joining the two ends of the Spiral Array tube to form a hypertorus. The circle of fifths is then represented as a helix that wraps around the hypertorus three times, illustrated in Figure 2.12. They then propose an audio key-finding model that is based on projecting collections of pitches onto the interior space contained by the hypertorus. This is essentially mapping to three distinct feature spaces: the circle of fifths, the circle of major thirds, and the circle of minor thirds. This 6-dimensional space is called the *tonal centroid*. The algorithm first applies the CQ-transform in order to extract the pitch-class distribution. The 12-D pitch-class distribution is then mapped to the 6-D tonal centroid with a mapping matrix. The algorithm was applied in a chord recognition system, however, the authors point out that it could be adapted for other classification tasks such as key detection.

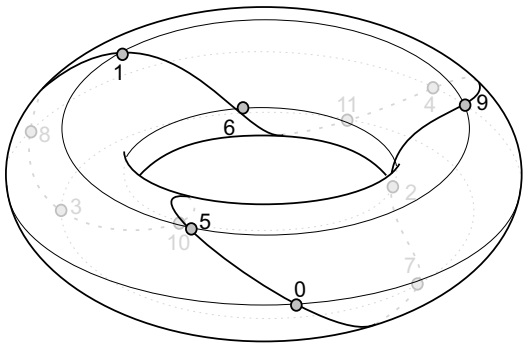

Fig. 2.12: If enharmonic and octave equivalence are considered, then the Spiral Array model can be represented as a hypertorus (from Harte et al. 2006).

Gatzsche et al. (2008) propose a novel approach to audio key finding, making use of a model based on circular pitch spaces (CPS). They introduce the music theory-based concept of a CPS and go on to present a geometric tonality model that describes the relationship between keys. Furthermore, they implement an audio key-finding system that

makes use of the model. The CQ-transform is employed in order to extract a pitch-class distribution, which is in turn input to the CPS model. The model then maps the vector to 7 different circular pitch spaces, which essentially gives 7 different predictions for the key.

2.3.2.4 Chord Progression and HMM-Based Methods

The ability to automatically identify the key and label the chords from audio would be extremely useful for the purpose of harmonic analysis. Identifying the presence of certain chords in a piece of music can lead to an improved estimate of the key. Similarly, knowing the key of a piece of music can improve the accuracy of chord identification. As such, chord recognition and key detection are two closely related problems and have been approached simultaneously by various researchers.

One of the most prominent tools for approaching this problem is the Hidden Markov Model (HMM). A HMM is a type of statistical model which is commonly used for temporal pattern recognition. It consists of a sequence of states that are hidden to the observer, which model a stochastic process. The states are observable only through another set of stochastic processes, which produce a set of time-based observations (Lee and Slaney 2007). The model is parameterized with a discrete number of states, a state transition probability distribution (i.e., the probability of each state transitioning to another one), and an observation probability distribution (i.e., the probability that each state leads to a particular observation) (Noland and Sandler 2006).

Chai and Vercoe (2005) present an HMM-based audio key detection system that segments the signal based on modulations and identifying the key of each segment. A 24-dimensional pitch-class distribution (i.e., half semitone resolution) is used, as opposed to the standard 12-dimensional vector. The proposed approach is to first detect the scale root note (the tonic) in one step and then to detect the mode of the key. Thus, two different HMMs are used, one for each step. The first HMM has 12 states (i.e., one for each key) and the second HMM has just 2 states (i.e., one for each mode: major and minor). State

transition probability distributions are represented as a 12x12 matrix for the first HMM and a 2x2 matrix for the second. The initial parameters of the HMMs were set empirically with values based on music theory. Ten classical piano pieces, manually annotated with key based on segmentation were used to evaluate the system. Three different criteria were used for the evaluation: recall (i.e., proportion of detected segmentations that are relevant), precision (i.e., proportion of relevant transitions detected), and label accuracy (i.e., proportion of correctly labeled segments). The maximum label accuracy achieved was approximately 83%.

Peeters (2006a, 2006b) proposes an audio key detection system that implements one HMM for each of the 24 possible keys. A front-end algorithm is used to extract a sequence of time-based chroma-vectors (i.e., pitch-class distributions) for each of the songs in a training set of key annotated music. All of the chroma-vectors for songs in the major mode are then mapped to C major and all of the chroma-vectors for songs in the minor mode are mapped to C minor. This data is then used to train two HMMs: one for the major mode and one for the minor mode. One HMM is then created for each of the 24 keys by applying circular permutation of the mean vectors and covariance matrices of the state observation probability. Peeters goes on to compare the HMM-based model with a template-based system that is a combination of the models proposed by Gómez (2006b) and Izmirli (2005a). A flowchart depicting both of the implemented methods is shown in Figure 2.13. In an evaluation using 302 classical music pieces, the template-based system had a maximum key recognition rate of 85.1%, whereas the HMM-based model had a maximum key recognition rate of 81%. Peeters claims that part of the reason for the lower recognition rate of the HMM-based system is due to the fact the training set included music with modulations to neighboring keys. These modulations led to perfect 5^{th}, parallel major/minor, and relative major/minor errors.

Noland and Sandler (2007) undertook an experiment in which they analyze the effect of low-level signal processing parameters on two audio key identification algorithms: one template-based algorithm and one HMM-based algorithm. The template-based algorithm uses the CQ-transform in order to extract pitch-class distributions from the signal, which

are correlated with templates derived from recordings of J. S. Bach's *The Well Tempered Clavier*. The HMM model is based on a previous implementation (Noland and Sandler 2006) and a simplified version is shown in Figure 2.14. The results of Krumhansl's probe-tone experiments (Krumhansl 1990) are used to initialize the transition and state observation probabilities. Both algorithms were evaluated using a corpus of 110 Beatles songs, testing different values for several low-level parameters: downsampling factor, window length, hop size, and highest constant-Q frequency. The results showed that the choice of parameters had different effects on the two algorithms, leading to the conclusion that an optimal choice of signal processing parameters is highly dependent on the particular algorithm that is implemented.

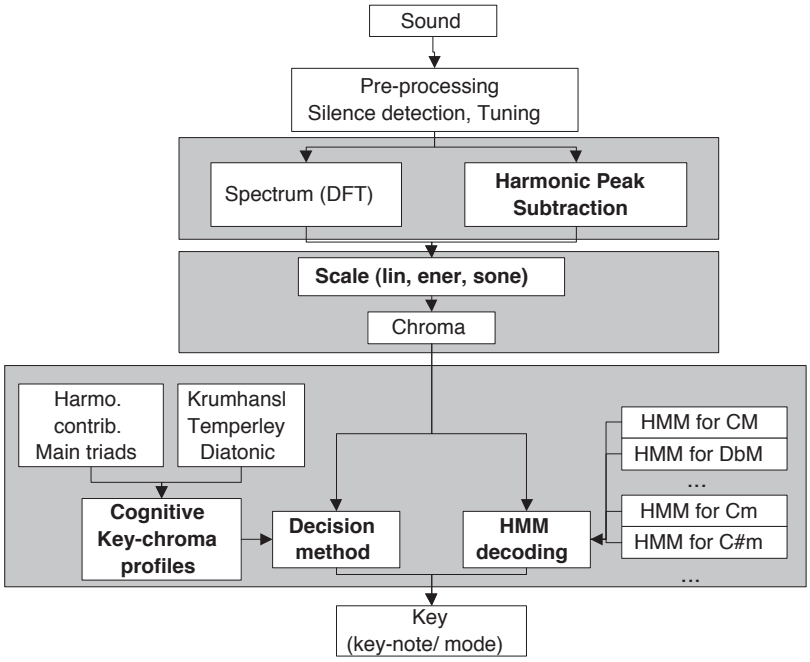

Fig. 2.13: Flowchart of the audio key estimation system (from Peeters 2006a).

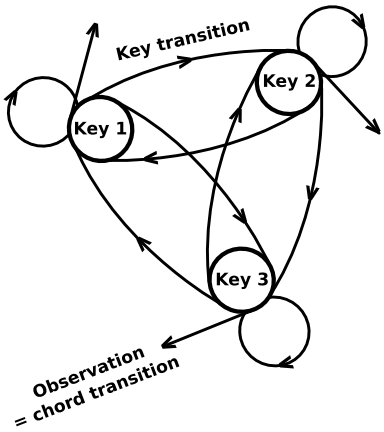

Fig. 2.14: Simplified version of the HMM, showing only three of the possible keys (from Noland and Sandler 2007).

Burgoyne and Saul (2005) present a system for tracking chords and key simultaneously, that implements a Dirichlet-based HMM. A Dirichlet distribution is a type of probability distribution that can be used in place of the more common Gaussian distribution for the implementation of an HMM. Dirichlet distributions place more emphasis on the relations of the outputs as opposed to their magnitudes. This is preferable for the case of chord detection from pitch-class distributions, because the important aspect is the presence of certain notes and not their magnitude. Burgoyne and Saul's system uses Dirichlet distributions to parameterize the observation distributions of the HMM in their system. The HMM was trained using a corpus of 5 Mozart symphonies in 15 movements, accompanied with ground truth harmonic analysis. Evaluation was then performed using a recording of Minuet from Mozart's Symphony No. 40. The correct chords were detected 83% of the time, however, the system was unable to identify the correct key.

Lee and Slaney (2007) also approach the audio key-finding problem by implementing an HMM-based system that performs chord recognition and key detection simultaneously. The system uses the Tonal Centroid vector that was proposed by Harte et al. (2006) (see Section 2.3.2.3). A separate 24-state HMM is built for each of the 24 possible keys, and

each state represents a single type of chord[10]. A corpus of 1046 audio files synthesized from MIDI was used to train the HMMs. The system was subsequently evaluated using recordings of 28 Beatles songs and the overall key detection rate was 84.62%.

Several instances have been reported in the literature of using HMMs for local key finding from audio, such that the system is able to track key modulations. Catteau et al. (2007) proposed a system of this type that performs simultaneous key and chord recognition from audio. Using music theory derived from Lerdahl (2001), the system implements a probabilistic framework, incorporating models for observation likelihood and chord/key transition. Chord and key labels are inserted on a frame-by-frame basis over the course of the audio file. An evaluation was performed using 10 polyphonic audio fragments of popular music and the correct key was labeled for 82% of the frames.

Papadopoulos and Peeters (2009) approach the local audio key estimation problem by considering combinations and extensions of previous methods for global audio key finding. The system consists of three stages: feature extraction, harmonic and metric structure estimation, and local key estimation. The feature extraction algorithm extracts a chromagram from the audio signal, consisting of a sequence of time-based pitch-class distributions (Papadopoulos and Peeters 2008). Metric structure estimation is then achieved by simultaneously detecting chord progressions and downbeats using a previously proposed method (Papadopoulos and Peeters 2008). The final stage of the system performs local key estimation using an HMM with observation probabilities that are derived from pitch-class templates. They create five different versions of the system using different types of pitch-class templates: Krumhansl (1990), Temperley (2001), diatonic, Temperley-diatonic (Peeters 2006b), and an original template where all pitch-classes have an equal value except for the tonic, which has triple the value. The system is then evaluated using five movements of Mozart piano sonatas, with manually annotated ground truth data corresponding to chords and local key. A maximum local key recognition rate of 80.22% was achieved by using the newly proposed pitch-class template.

Shenoy et al. (2004) present a novel, rule-based approach for estimating the key of an audio signal. The system utilizes a combination of pitch-class distribution information, rhythmic information, and chord progression patterns in order to estimate the key. The audio signal is first segmented into quarter note frames using onset detection and dynamic programming techniques. Once segmented, an algorithm is employed to extract the pitch-class distribution for each frame. Using this information, the system is then able to make inferences about the presence of chords over the duration of the audio signal. Finally, the chord progression patterns are used to make an estimate for the key of the piece. The system was evaluated with 20 popular English songs and had a key recognition rate of 90%.

---

[1] If the first note appearing in the melody is the tonic for a key candidate, then that candidate is chosen as the key. If the first note is not the tonic for any of the candidates, then the same process is applied using the dominant instead of the tonic.
[2] This algorithm is an example of what Temperley (1999) calls a *flat input/flat-key* approach.
[3] The original K-S algorithm is an example of what Temperley (1999) calls a *weighted-input/weighted-key* approach. The modified algorithm proposed by Temperley (1999) is an example of what he calls a *flat-input/weighted-key* approach.
[4] There are many different terms used in the literature for pitch-class distribution features. Perhaps the first reported instance of a pitch-class distribution feature was that of Fujishima (1999), who implemented the *pitch-class profile* as part of his chord recognition system. See section 2.3.2.2 for more examples of nomenclature used for pitch-class distribution features.
[5] The primary advantage provided by the CQ-transform lies in the fact that the mapping to the logarithmic frequency domain has a resolution that is geometrically proportional to the frequency. Conversely, FFT maps to the frequency domain with a constant frequency resolution (Purwins et al. 2000).
[6] The probe tone experiments were a cognitive study that derived ratings for each pitch-class within an established tonal context. Hence Purwins, Blankertz, and Obermayer's (2000) model is an example of a template-based audio key detection system that uses cognitive-based pitch-class templates.
[7] These templates are a combination of cognitive and statistics-based templates.
[8] Clarifying low frequencies is designed to overcome some of the errors attributed to the reduced resolution for lower frequencies. Fuzzy logic is used to determine the likelihood that a detected frequency component is actually attributable to a pitch-class. The adaptive level weighting scheme scales the FFT results in the various frequency ranges to improve the salience of the detected pitch content. The flattening of high and low values is a final step that sets the pitch class membership to 1 if the detected value is greater than 0.8 and sets the value to 0 if the detected value is less than 0.2.
[9] The ISO standard tuning pitch states that A = 440Hz, which is know as the concert pitch. However, there exists other historical standards for tuning, such as the diapason normal, which has A=435Hz. Furthermore, many acoustic recordings have inaccuracies in their tuning pitch. For instance, an orchestra will typically be tuned using the oboe as the reference pitch, which itself may be tuned incorrectly (Zhu and Kankanhalli 2006).
[10] In this model there are two types of chords, major and minor, for each of the 12 chromatic pitch-classes. For example, an F minor triad is considered the same type of chord as an F minor seventh. This leads to 24 different possible chord types.

Chapter 3

Software Design

3.1 Introduction

The software application implemented for this thesis is designed to automatically identify the key of musical excerpts from an audio signal. It employs signal processing techniques in order to extract salient pitch information from the signal, which is then used as input to the classifier in order to identify the key. There are four main components in the application, all of which were developed modularly (see Figure 3.1): frequency analysis, pitch-class extraction, pitch-class aggregation, and key classification. Several versions of each component were created, using a variety of parameters, techniques and algorithms. The modular approach then allowed the various component versions to be paired with one another and evaluated in order to identify the configuration with maximum accuracy. The remainder of this chapter will describe the details of each of these components as well as any other pertinent information relating to the design and implementation of the application.

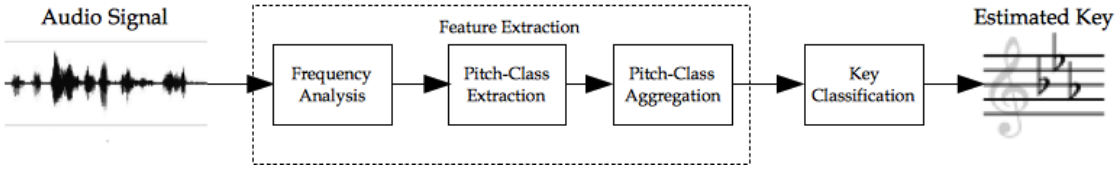

Fig. 3.1: The four primary components of the audio key detection software application.

3.2 Software Packages

jMIR is an open-source, java-based framework intended for prototyping and developing automatic music classification applications (McKay and Fujinaga 2009). Two components of the jMIR software package were used to implement the audio key detection application for this thesis: jAudio and ACE.

jAudio is an application framework for feature extraction from audio files (McEnnis et al. 2005). It is designed to reduce the duplication of effort required for developing new feature extraction algorithms. For example, the system handles the loading of files using Java's audio interface, which might otherwise be a laborious task for the researcher to implement. It also comes bundled with a number of commonly used audio features, which can either be extracted directly or used for the calculation of other features. The application has the ability to extract features for each window of an audio signal, as well as to use aggregators in order to collapse a sequence of windowed values into a single vector (e.g., mean, standard deviation). The capabilities of jAudio made it an optimal choice for the feature extraction algorithms used for this thesis.

ACE (Autonomous Classification Engine) is a meta-learning software package designed for performing and optimizing music classification tasks (McKay et al. 2005). Built on the Weka machine learning framework, ACE provides the ability to experiment with a variety of classifier algorithms, parameters, and dimensionality reduction techniques in order to determine an optimal arrangement for the particular task. The

flexibility and ease of use of ACE make it an ideal choice for experimenting with various classifier configurations for the audio key detection problem. As such, it was used for the classification portion of the software application built for this thesis.

3.3 Feature Extraction

The feature extraction component of the software involves the application of signal processing techniques in order to extract meaningful information from the audio signal that can be used to identify the key. The feature extraction algorithm implemented for this thesis can be further subdivided into three components: frequency analysis, pitch-class extraction, and pitch-class aggregation. This section will describe the implementation details for these components.

3.3.1 Frequency Analysis

The frequency analysis component consists of the application of a transform function in order to convert an audio signal from the time domain to a frequency domain representation. For the purposes of audio key detection, the FFT (Fast Fourier Transform) is the most commonly employed technique for obtaining a frequency domain representation from the audio signal. Figure 3.2 shows the time domain representation of an example audio excerpt as well as the frequency domain representation, calculated using the FFT function within Matlab.

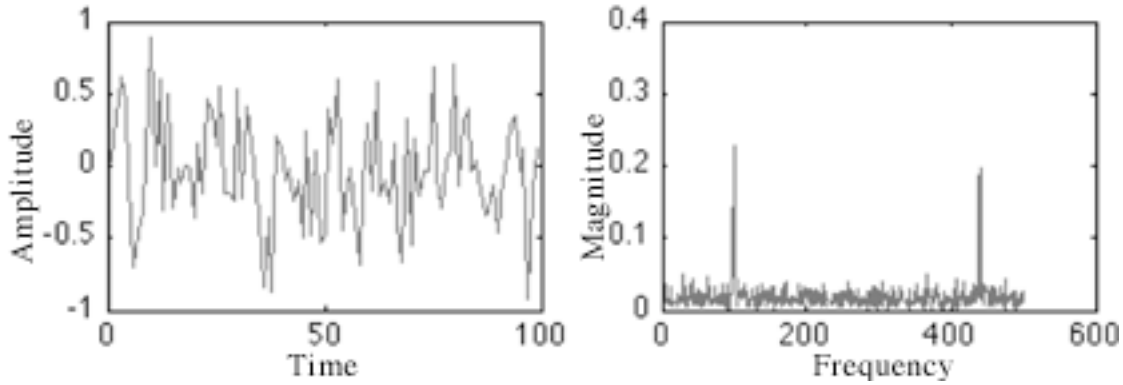

Fig 3.2: First 100 samples of an audio signal with a 100 Hz sine wave, a 440 Hz sine wave, and random noise (left). Frequency domain representation using from an FFT (right).

jAudio comes bundled with the ability to extract both magnitude an power spectrums from an audio file using a complex to complex FFT function with or without a Hanning window. It also provides the ability to easily configure several of the low-level signal processing parameters, such as the sampling rate, window size, and window overlap.

In order to compute the FFT, the audio signal must be divided into windows (also known as frames) and so it is necessary to make a choice for the window size and the amount of overlap between consecutive windows. The window size is directly proportional to the frequency resolution of the resulting frequency domain representation. However, the window size is inversely proportional to the temporal resolution. In other words, the frequency resolution increases with the window size, whereas the temporal resolution decreases. Since humans perceive pitch on a logarithmic scale, lower pitches are closer in frequency, and therefore a higher frequency resolution is required to differentiate between them. In the context of key detection it is necessary to have a high frequency resolution, so larger window sizes are often used. Although this leads to a reduced temporal resolution, increasing the window overlap amount can be used to compensate for this effect. A finer temporal resolution improves the ability of the system

to detect pitch content in the presence of dramatic temporal variations. However, larger window overlaps also lead to increased amounts of data and processing times.

The choice of sampling rate, window size, and window overlap can have a dramatic effect on the salience of pitch-class distribution that is extracted from the signal (Noland and Sandler 2009). We experiment with various values for these parameters in order to investigate how they affect key detection accuracy when paired with different classifiers. Table 3.1 summarizes the combinations of frequency analysis parameters that were tested.

Sampling Rate	Window Size	Window Overlap
11,025	1024	0
22,050	1024	0
44,100	1024	0
11,025	4096	0
22,050	4096	0
44,100	4096	0
11,025	8192	0
11,025	8192	0.5
22,050	8192	0
22,050	8192	0.5
22,050	8192	0.8
44,100	8192	0
44,100	8192	0.5
11,025	16,384	0
11,025	16,384	0.5
22,050	16,384	0
22,050	16,384	0.5
44,100	16,384	0
44,100	16,384	0.5
44,100	16,384	0.8

Table 3.1: Summary of the combinations of parameters that were used when calculating the magnitude spectrum using the FFT.

3.3.2 Pitch-Class Extraction

Once we have obtained a frequency domain representation for each window of the audio signal, it is necessary to apply an algorithm in order to extract the pitch-class distribution. We present a basic algorithm for mapping from the analysis frequency spectrum to the pitch-class distribution vector. We also present several extensions that can be used in conjunction with the basic algorithm, as well as with one another. Table 3.2 summarizes the combinations of extensions that were tested.

Extension 1	Extension 2	Extension 3
-	-	-
PD	-	-
SFM	-	-
LFC	-	-
PD	SFM	-
PD	LFC	-
SFM	LFC	-
PD	SFM	LFC

Table 3.2: Summary of the combinations of extensions that were tested in combination with the Basic Algorithm. The first row indicates a permutation in which no extensions were used. PD = Peak Detection Extension, SFM = Spectral Flatness Measure Extension, LFC = Low Frequency Clarification Extension.

3.3.2.1 Basic Mapping Algorithm

The basic algorithm uses a mapping matrix in order to translate the windowed frequency spectra into pitch-class distribution vectors. Using the standard value of 440 Hz to set the fundamental reference frequency of A_4 (i.e., $A_1 = 55$ Hz), we first utilize the function $n(f)$ to map the analysis frequency bins f_j to a semitone note scale:

$$\begin{aligned} &\mathrm{N} = \text{Window Size} \\ &j = 0,...\mathrm{N} \\ &f_0 = 55\mathrm{Hz} \\ &n(f_i) = 12\log_2\left(\frac{f_i}{f_0}\right)^2 \end{aligned} \tag{3.1}$$

An intermediate 12xN matrix *D* is then created with projected values of *n(f)* for each pitch-class in the range of -6 to +6:

$$\begin{aligned} &i = 0,...11 \\ &D_{i,j} = \big(n(f_j) - i + 6\big)(\mathrm{mod}\,12) - 6 \end{aligned} \tag{3.2}$$

A Gaussian distribution function is then employed in order to produce the mapping matrix *M*. The use of the distribution function helps to counteract the effects of any possible spectral leakage or tuning errors in the audio signal:

$$M_{i,j} = e^{-\frac{1}{2}(2D_{i,j})^2} \tag{3.3}$$

Finally, the pitch-class distribution vector *p* is obtained by multiplying the FFT spectrum values x_j by the corresponding mapping matrix entry.

$$p_i = \sum_{j=0}^{N} M_{i,j} \times x_j \tag{3.4}$$

The minimum analysis frequency to be used in the mapping is set to 55 Hz (A_1), a value based on our own preliminary experimentation as well as previous research (Noland and Sandler 2007). The maximum analysis frequency considered is set to 1760 Hz (A_6). This results in a total of five octaves to be included in the analysis, which covers the majority of fundamental note frequencies for our corpus of music (Chuan and Chew 2005b).

The final step in the algorithm is to normalize the pitch-class distribution vector such that the values of all of the elements sum to one.

3.3.2.2 Peak Detection Extension

The peak detection extension algorithm is based on the Local Maximum Selection method proposed by Chuan and Chew (2005a). Using this method, a peak is defined as any FFT bin value that is greater than the average value to both the left and right within any given semitone region in the analysis frequency range. Furthermore, only one peak may exist within any given semitone region. When used in conjunction with the basic algorithm, the only difference is in how the actual pitch-class distribution vector is created. Instead of summing every frequency component multiplied by the mapping matrix, as shown in Equation 3.4, only the peak frequencies are added to the bins of the pitch-class distribution. Here, the function *f(x)* represents the peak selection function:

$$P_i = \sum_{j=0}^{N} f(M_{i,j} \times x_j) \tag{3.5}$$

3.3.2.3 Spectral Flatness Measure Extension

The Spectral Flatness Measure (SFM) Extension employs the technique proposed by Izmirli (2005a). The SFM is defined as the ratio between the geometric mean and the arithmetic mean of any given range of values in the analysis frequency range (x_i to x_j):

$$GM = \left(\prod_{k=i}^{j} x_k \right)^{\frac{1}{j-i}} \tag{3.6}$$

$$AM = \frac{1}{j-i}\sum_{k=i}^{j} x_k \tag{3.7}$$

$$SFM = \frac{GM}{AM} \tag{3.8}$$

A SFM value that is closer to 1 indicates a flatter spectrum, whereas values closer to 0 are indicative of peaks in the signal. We calculate the SFM for half octave regions within the analysis frequency range (55 Hz to 1760 Hz). Regions that have an SFM greater than 0.6 have the values set to 0.

3.3.2.4 Low Frequency Clarification Extension

The Low Frequency Clarification Extension is based on one component of the fuzzy analysis techniques proposed by Chuan and Chew (2005c). The method is meant to counteract some of the errors produced as a result of the reduced frequency resolution in the low end of the analysis frequency spectrum. In our version, the low frequencies are considered to be those in the first two octaves of our analysis frequency range (i.e., 55 Hz to 220 Hz). First, the peak detection algorithm described in Section 3.2.2.2 is used to find the peaks in the first two octaves of the frequency spectrum. Each of these peaks is then compared to any peaks that may exist in the region one semitone above and one semitone below. If the value of any given peak is smaller than either of those found in the two neighboring semitone regions, then it is excluded from the mapping to the pitch-class distribution vector. The logic behind this step is that if a neighboring semitone region has a peak value that is greater than it's own, then it is likely that the given peak is a result of spectral leakage.

3.3.3 Pitch-Class Aggregation

After extracting the pitch-class distribution for each window of the audio signal, the data must be collapsed into a single array representing the global pitch-class distribution for the entire signal. In a typical audio key detection system, the arithmetic mean of the windowed values is used to accomplish this. However, Chuan and Chew (2005c) note that during the calculation of pitch-class distributions, errors tend to accumulate over time. In order to counteract this problem, they propose a *periodic cleanup* procedure. The pitch-class aggregator implemented for this thesis uses an adapted version of this technique. The procedure first consists of separating the windowed pitch-class distribution values into subsets of equal size. The arithmetic mean is calculated for each subset of windowed pitch-class distributions and the smallest two pitch-class values are then set to zero. Finally, the arithmetic mean is calculated for all of the subsets and then normalized (i.e., values of all indices sum to one) to give the global pitch-class distribution. Figure 3.3 illustrates this process.

We experiment with several different sizes for the subsets of windowed values, corresponding to varying period times for the cleanup procedure. We compare the results with an arithmetic mean aggregator that does not implement the periodic cleanup technique. Table 3.3 summarizes the various pitch-class aggregators that were tested.

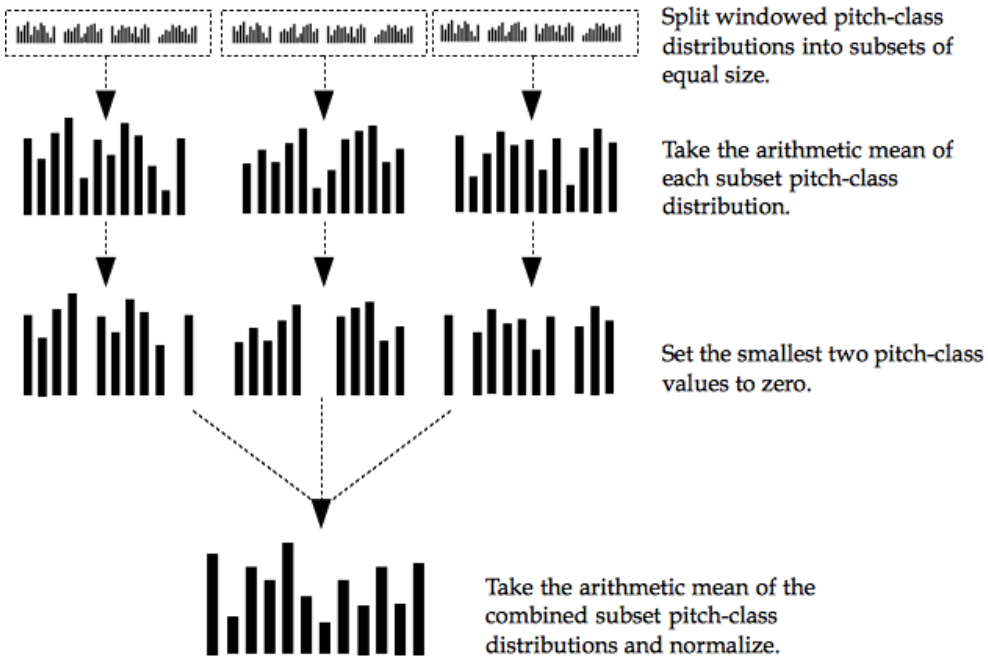

Fig. 3.3: The periodic cleanup process used when collapsing the windowed pitch-class distribution vectors into a single, global pitch-class distribution vector.

Pitch-Class Aggregator Algorithm	Period for Cleanup Procedure
Arithmetic mean	-
Periodic cleanup	~ 1 second
Periodic cleanup	~ 2 seconds
Periodic cleanup	~ 4 seconds

Table 3.3: Summary of the different pitch-class aggregator algorithms that were tested. The period for the cleanup procedure is measured in number of windows, which is dependent on the window size. As such, the approximate time value (in seconds) is given.

3.3.4 Key Classification

In order to classify a particular instance of a musical excerpt, the pitch-class distribution is first extracted using the previously described techniques. The pitch-class distribution is then used as input to a trained classifier, which identifies the instance as belonging to one of the 24 possible keys.

The classifiers are trained using two different types of data: pitch-class distributions extracted from training sets (see Section 4.2) and pitch-class templates derived from previous research (see Section 4.3).

Four different classifiers from the ACE framework are used: a neural network, a k-nearest neighbor algorithm, a support vector machine, and a naïve Bayes classifier. The remainder of this section will introduce these classifiers and provide the details of their implementation.

3.3.5 Neural Networks

The brain is composed of billions of elementary processing units, known as neurons. A single neuron is in and of itself, a relatively simple structure that acts to collect, process and propagate electrical signals throughout the brain. The immense processing power of the brain is believed to emerge only as a result of the vast interconnected network of these basic units. Early research into artificial intelligence sought to mimic these structures by creating artificial neural networks (ANNs) and has since lead to the modern field of computational neuroscience. Today, neural networks remain one of the most popular and effective forms of machine learning systems (Russel 2003).

3.3.5.1 ANN Units

In 1943 MucCulloch and Pitts devised a simple mathematical model of a neuron, illustrated in Figure 3.4. This over-simplified version of a neuron serves as the basic processing unit in an ANN. Each unit consists of three primary components: weighted input links, an activation function and output links.

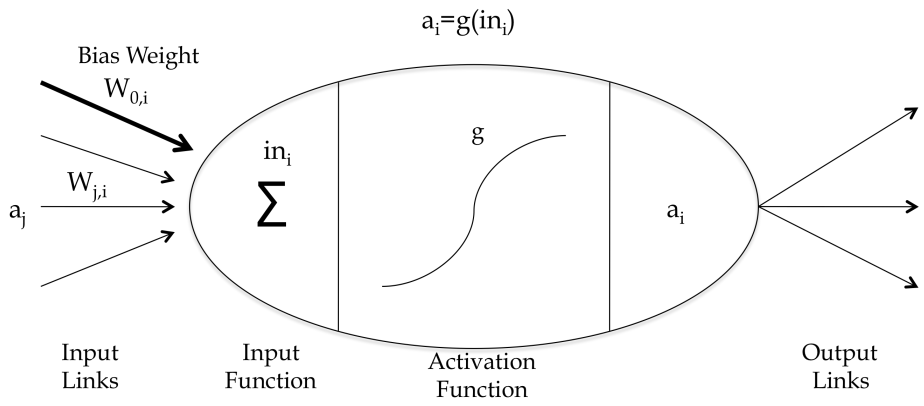

Fig. 3.4: Simple model of a neuron (Russel and Norvig 2003)

A unit receives a signal from it's weighted input links and sums the the input:

$$in_i = \sum_{j=0}^{n} W_{j,i} a_j \tag{3.9}$$

The output of the unit is then calculated from its activation function:

$$a_i = g(in_i) = g\left(\sum_{j=0}^{n} W_{j,i} a_j\right) \tag{3.10}$$

3.3.5.2 Network Topologies

The computational power of artificial neural networks is derived from the complex interconnections amongst the units and not the individual units themselves (Kostek 2005). There are two primary types of ANN topologies: feedforward and recurrent (cyclic). Recurrent topologies are not typically used for classification problems so we will restrict our attention to feedforward networks. Feedforward networks essentially represent a function of their current inputs, where the connection weights act as the function parameters (Russel 2003). Figure 3.6 shows a simple example topology for a feedforward ANN with two input units, two hidden units and one output unit. Equation 3.11 shows the function represented by the same network.

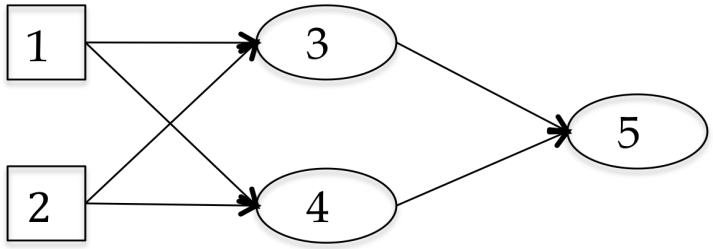

Fig. 3.6: A simple feedforward network topology with two input nodes, one hidden layer with two nodes, and one output node.

$$a_5 = g(W_{3,5}a_3 + W_{4,5}a_4)$$
$$a_5 = g(W_{3,5}g(W_{1,3}a_1 + W_{2,3}a_2) + W_{4,5}g(W_{1,4}a_1 + W_{2,4}a_2)) \quad (3.11)$$

The simplest type of feedforward ANN consists of a single input layer and a single output layer and is known as a perceptron. Perceptrons are limited by the fact that they can only represent linearly separable functions. In order to overcome this limitation, additional hidden layers must be added, which is known as a multilayer perceptron.

Mulitlayer perceptrons allow for the representation of a linear combination of perceptron threshold functions. With only one hidden layer, a multilayer perceptron is able to represent any continuous function of it's inputs. By adding additional hidden layers, any discontinuous function can be represented.

3.3.5.3 Learning Algorithms

In order to train a neural network to represent a function appropriately, a learning algorithm must be applied. The goal is to adjust the connection weights in order to reduce the error and improve performance with the training set. This essentially amounts to an optimization search in the weight space. For a single layer perceptron, the gradient descent algorithm is typically used. For a multilayer perceptron, the most common learning algorithm employed is backpropagation.

3.3.5.4 Implementation

The neural network used for this thesis consists of a feedforward multilayer perceptron with a backpropagation learning algorithm. A number of different topologies and parameter values were experimented with in preliminary testing in order to identify optimal values. Table 3.4 summarizes the final makeup that was used.

Parameter	Description	Value
Input nodes	Number of nodes in the input layer. We use one node for each pitch-class.	12
Output nodes	Number of nodes in the output layer. We use one for each possible key.	24
Hidden layers	Number of hidden layers in the network.	1
Nodes in hidden layers	Number of nodes in the hidden layer.	18
Learning rate	The amount the connection weights are updates after each epoch.	0.13
Momentum	The amount of momentum applied when updating weights. Momentum allows adjustments to persist over several epochs.	0.19
Training epochs	The number of epochs (cycles) that are used to train the network.	1000

Table 3.4: A summary of the topology and parameter values used for the neural network classifier.

3.3.6 K-Nearest Neighbor Algorithm

The k-nearest neighbor algorithm consists of classifying an instance by searching the feature space for the closest training examples. The instance is assigned to the class that the majority of its *k* nearest neighbors belongs to. If *k*=1, then the instance is simply assigned to the class of its nearest neighbor. The implementation used for this thesis uses a 12-dimensional feature space, one dimension for each pitch-class, and a Euclidean distance metric for evaluating the distance between instances. Preliminary tests indicated that a value of 1 for *k* significantly outperformed any other value and so it was the only value used.

3.3.7 Support Vector Machines

A support vector machines is a type of supervised learning classifier that belongs to the more general category of kernel machines (Russel and Norvig 2003). The fundamental idea is that if input data is mapped to a sufficiently high number of dimensions, then it

will always be linearly separable. Except in special cases, N data points will be linearly separable in an N-1 dimension space. As such, support vector machines search for ways to re-express the input data using computed features. This amounts to a quadratic programming optimization problem, searching for an optimal linear separator that maximizes the functional margin between the positive examples on the one side and the negative examples on the other side.

Supposing there are x_i input examples that can be classified as $y_i = \pm 1$, then the problem of identifying the optimal linear separator can be expressed as finding the values of α_i that maximize the value of the following expression:

$$\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$
$$\alpha_i \geq 0, \sum_i \alpha_i y_i = 0 \tag{3.12}$$

Once the optimal values for α_i have been derived, the equation that defines the linear separator is:

$$h(\mathbf{x}) = \text{sign}\left(\sum_i \alpha_i y_i (x_i \cdot x_j)\right) \tag{3.13}$$

All of the values of α_i are zero, except for those that are closest to the actual linear separator, which are known as the *support vectors*.

3.3.8 Naïve Bayes Classifiers

The naïve Bayes classifier utilizes the calculation of probabilities based on the observed sample data. This is based on the equation known as Bayes rule (Russel and Norvig 2003):

$$P(Y \mid X) = \frac{P(Y \mid X)P(Y)}{P(X)} \tag{3.14}$$

The classifier works under the assumption that a single cause influences several effects, all of which are conditionally independent. In the case of key finding, this would mean that the key (i.e., the cause) influences the twelve values in the pitch-class distribution (i.e., the effects), all of which are conditionally independent of one another. Given this, it is possible to use parameter estimation techniques (e.g., method of maximum likelihood) in a supervised learning setting in order to train the model to classify future instances.

Chapter 4

Description of the Data

4.1 Introduction

This chapter describes the data that was used for training and evaluating the audio key detection application. Two types of data are presented: musical excerpts and pitch-class templates. The pitch-class templates are used strictly as training data for a set of classifiers for Phase II of the experiment (see Section 5.3). The musical excerpts are used to parameterize the system using cross-validation in Phase I of the experiment (see Section 5.2), as well as to evaluate the trained classifiers in Phase II and III of the experiment (see Section 5.3 and 5.4). The bulk of the excerpts consist of music from the classical period, which follows the trend set by the majority of the previous studies on audio key detection (see Appendix A, which shows the data sets used for previous studies on audio key detection).

4.2 Musical Excerpts

Three different corpora of key-annotated musical excerpts are used as ground truth data for training and evaluating the classifiers: *classical*, *popular*, and *MIDI*. The first corpus is comprised of excerpts from the classical period, containing a variety of styles and instrumentation, including symphonies, sonatas, concertos, preludes, and fugues. Only the first movements are used and the key is derived from the title of the piece. The second

corpus consists of excerpts of popular music songs, primarily in the pop-rock style. The keys labels for this data set use the annotations from Mauch et al. (2009) as well as manual annotations identified by ear. The final corpus is made up of excerpts of audio that have been synthesized from MIDI files of classical music using Quicktime. All MIDI files are from the classicalarchives.com website and the instrumentation that came in the file is used. The key labels for this corpus are also derived from the titles of the pieces. The excerpts from all of the data sets consists of only the first 30 seconds of the piece, so as to encompass the portion that is most likely to establish the global key as well as to avoid modulations. Furthermore, every excerpt has been manually verified by ear to be in the key that it is annotated with.

Section 4.2.1 presents the training data sets from each of the three corpora that are used to parameterize the system in Phase I of the experiment (see Section 5.2) as well as to evaluate the system in Phase II of the experiment (see Section 5.3). Section 4.2.2 describes the data sets that are used to evaluate the trained classifiers in Phase III of the experiment (see Section 5.4).

4.2.1 Training Sets

Four different data sets are used to parameterize and evaluate the system for Phase I and II of the experiment. The first data set ("Classical") is comprised of 248 excerpts of pieces from the classical corpus, including excerpts of pieces by J. S. Bach (48), C. P. E. Bach (6), Beethoven (17), Boccherini (8), Clementi (17), Haydn (106), Mozart (18), Salieri (3), and Schubert (25). The second data set ("Popular") is made up of 150 excerpts of songs from the popular corpus, Including songs by The Beatles (126), Carole King (7), and Queen (17). The third data set (MIDI) consists of 209 excerpts from the MIDI corpus, with pieces by Bach (48), Beethoven (65), Haydn (57), Mozart (18), and Schubert (21). The final data set ("Combined") is the composite of the first three data sets (i.e., 248 excerpts from the classical corpus, 150 excerpts from the popular corpus, and 209

excerpts from the MIDI corpus). For detailed information on the excerpts used for the training sets, see Appendix A.

4.2.2 Test Sets

Four different data sets are used to evaluate the trained classifiers in Phase III of the experiment (see Section 5.4). The first data set is from the classical corpus and consists of excerpts of the first 30 seconds of each of Chopin's 24 preludes Op. 28. The second data set also consists of excerpts of the first 30 seconds of Chopin's 24 preludes Op. 28, but synthesized from MIDI files obtained from the classicalarchives.com website. The third data set is made up of 59 excerpts from a variety of artists in the popular corpus, primarily in the pop-rock style. The final data set consists of a random selection of 10 excerpts from each of the three aforementioned data sets (i.e., 10 excerpts from the classical test data set, 10 excerpts from the MIDI test data set, and 10 excerpts from the popular test data set). For detailed information on the excerpts used for the test sets, see Appendix B.

4.3 Pitch-Class Templates

Six different pitch-class templates are used to train a set of classifiers (see Section 5.3). The data used for these templates is compiled from the research of Krumhansl (1990), Temperley (2001), Izmirli (2005a), and Papadopoulos (2009). All of the templates have their values normalized so that they sum to one. Table 4.1 gives a summary of all of the templates. Table 4.2 presents the data that was used for each pitch-class value in the various templates and the remainder of the section shows graphical representations of the data.

Template	Description
Diatonic (D)	Diatonic pitch-classes have a value of one (before normalization) and all others are zero
Krumhansl (K)	Derived from the probe tone experiments of Krumhansl and Kessler (Krumhansl 1990)
Temperley (T)	Temperley modified the Krumhansl templates with improved results (Temperley 2001)
Krumhansl-Diatonic (KD)	The product of the diatonic and Krumhansl templates
Temperley-Diatonic (TD)	The product of the diatonic and Temperley templates (Izmirli 2005a)
Papadopoulos (P)	Diatonic pitch-classes have an equal value, except the tonic, which has triple the value

Table 4.1: A summary of all of the pitch-class templates.

Pitch-Class	D_M	D_m	K_M	K_m	T_M	T_m	KD_M	KD_m	TD_M	TD_m	P_M	P_m
0	0.14	0.14	0.15	0.14	0.13	0.13	0.21	0.21	0.17	0.17	0.33	0.33
1	0	0	0.05	0.06	0.05	0.05	0	0	0	0	0	0
2	0.14	0.14	0.08	0.08	0.09	0.09	0.12	0.11	0.12	0.12	0.11	0.11
3	0	0.14	0.06	0.12	0.05	0.12	0	0.18	0	0.16	0	0.11
4	0.14	0	0.10	0.06	0.12	0.05	0.15	0	0.16	0	0.11	0
5	0.14	0.14	0.10	0.08	0.10	0.10	0.14	0.12	0.14	0.14	0.11	0.11
6	0	0	0.06	0.06	0.05	0.05	0	0	0	0	0	0
7	0.14	0.14	0.12	0.10	0.12	0.12	0.17	0.15	0.16	0.16	0.11	0.11
8	0	0.14	0.06	0.09	0.05	0.09	0	0.13	0	0.12	0	0.11
9	0.14	0	0.09	0.06	0.09	0.05	0.12	0	0.12	0	0.11	0
10	0	0	0.05	0.08	0.04	0.04	0	0	0	0	0	0
11	0.14	0.14	0.07	0.07	0.10	0.10	0.10	0.10	0.14	0.14	0.11	0.11

Table 4.2: The data used for each of the pitch-class templates. D: diatonic, K: Krumhansl, T: Temperley, KD: Krumhansl-diatonic, TD: Temperley-diatonic, P: Papadopoulos, M: major, m: minor.

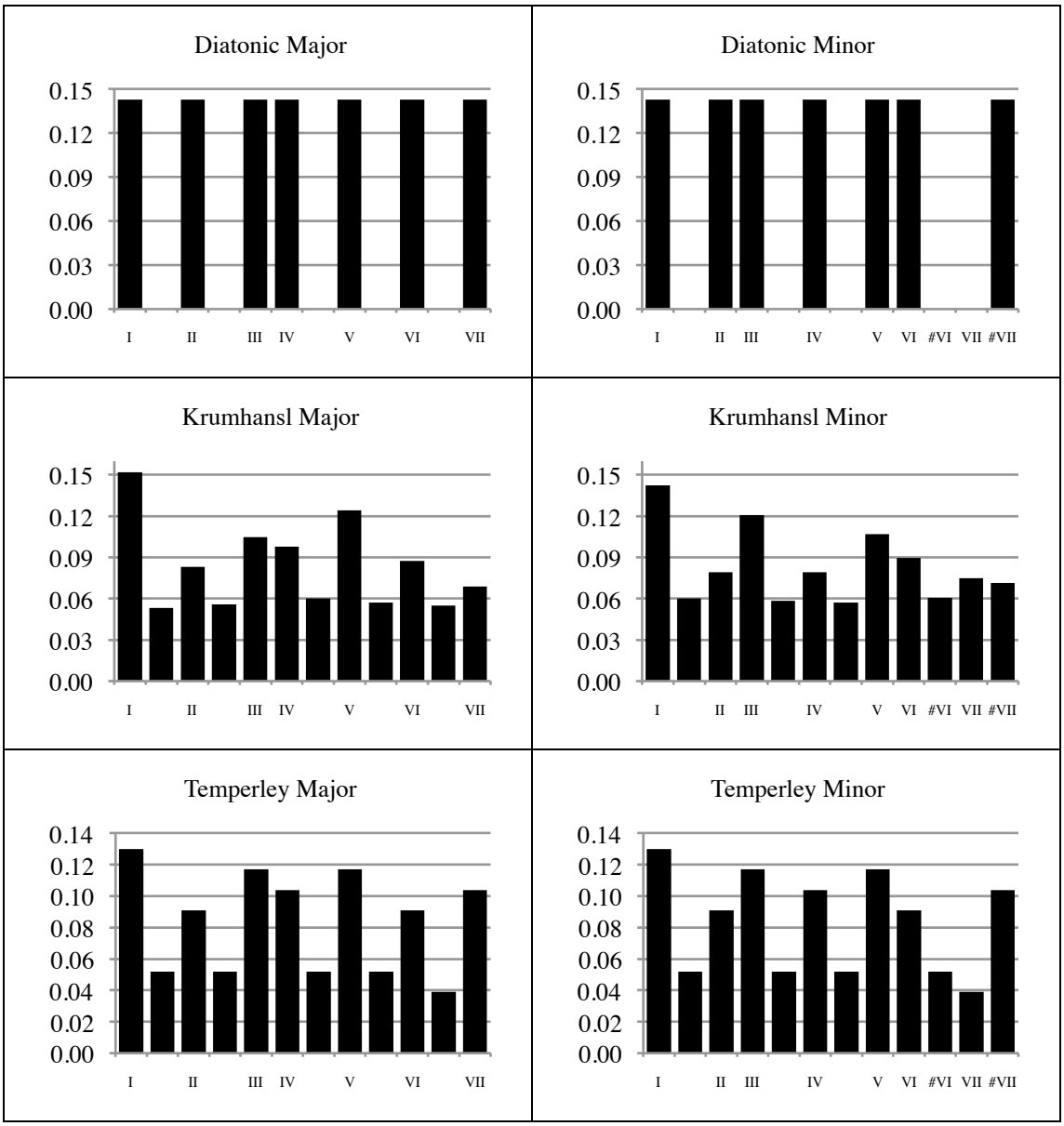

Fig. 4.1: Graphical representation of diatonic, Krumhansl, and Temperley pitch-class templates.

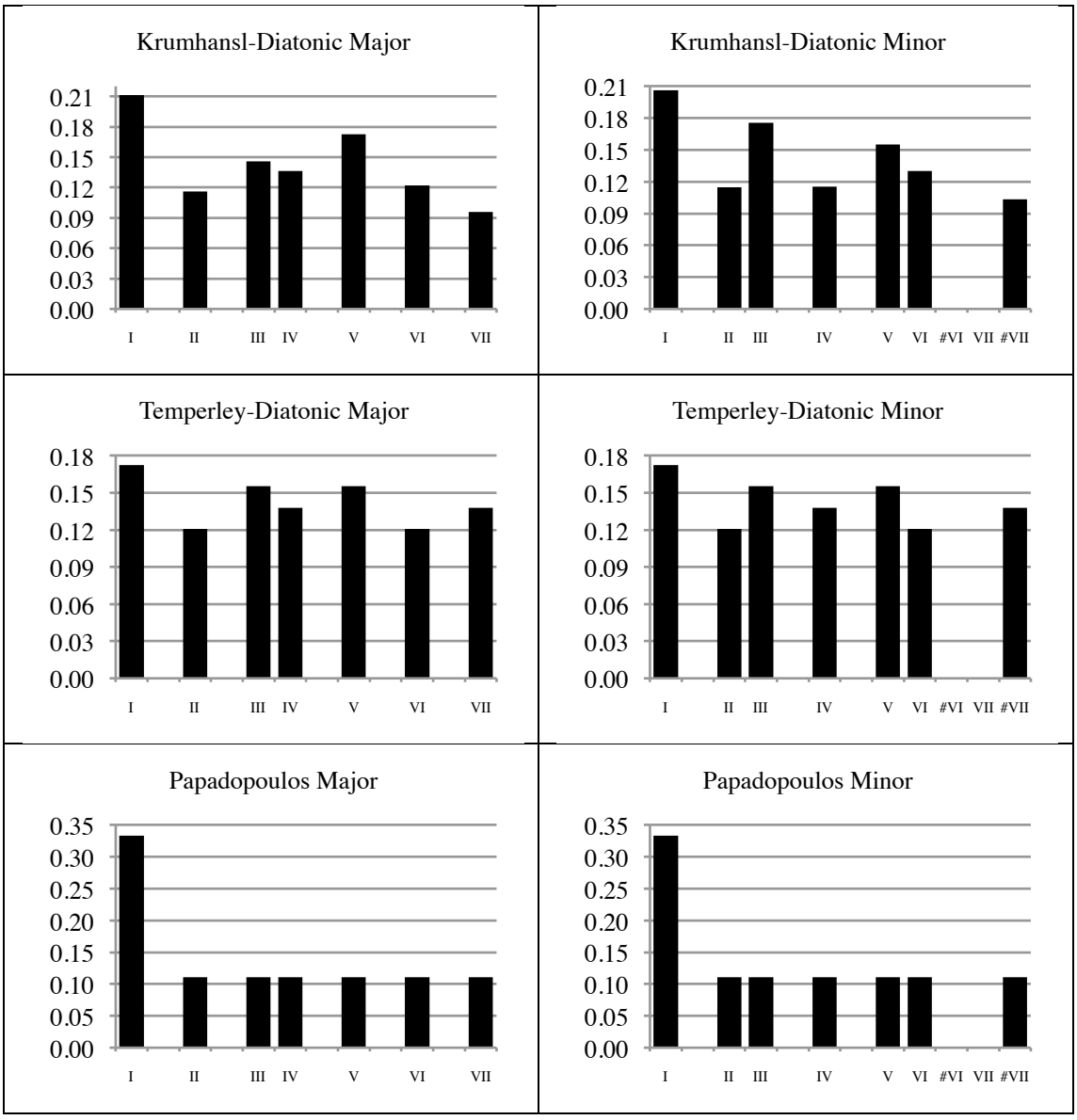

Fig. 4.2: Graphical representation of the Krumhansl-diatonic, Temperley-diatonic, and Papadopoulos pitch-class templates.

Chapter 5

Experimental Setup

5.1 Introduction

As described in Chapter 3, the software consists of four different components: frequency analysis, pitch-class extraction, pitch-class aggregation, and key classification. Several versions of each of these components were created, using a variety of parameter values and algorithms. We will refer to these various permutations as prototypes.

The experiment is comprised of three phases. Phase I is essentially a parameterization of the system. Four sub-phases are run, in which various combinations of frequency analysis parameters, pitch-class extraction algorithms, pitch-class aggregators, and classifiers are tested. The evaluation uses 10-fold cross-validation with each of the four training data sets (see Section 4.2.1). The results of Phase I are then used to select the prototype (i.e., the combination of frequency analysis parameters, pitch-class extraction algorithm, and pitch-class aggregator) to be used for the subsequent phases of the experiment. Furthermore, a set of models is trained using the data extracted with the selected prototype. One model is trained for each type of classifier (see Section 3.4) and each of the 4 training data sets.

Phase II of the experiment trains and evaluates another set of models. One model is trained for each type of classifier and each type of pitch-class template (see Section 4.3) and is subsequently evaluated using each of the 4 training data sets. This phase provides a

means of comparison between the models trained with the pitch-class templates and those that are evaluated using cross-validation in Phase I.

The final phase of the experiment uses the 4 test data sets (see Section 4.2.2) to evaluate all of the trained models from the previous two phases. There are 16 models trained from Phase I (i.e., one model of each type of classifier trained with each of the 4 training data sets) and 24 models trained from Phase II (i.e., one of each type of classifier trained with each of the 6 types of pitch-class templates).

Five different types of results are given for each phase of the experiment. One result is given for each of the four types of data sets: classical, popular, MIDI, and the combined data set. The final type of result given is the average of the results from the four different data sets.

5.2 Phase I: Cross-Validation Evaluation

Phase I entails testing the different prototype feature extraction algorithms using the k-nearest neighbor algorithm with k=1 and 10-fold cross-validation. The prototype implementations include 20 different sets of frequency analysis parameters (see Table 3.1), 8 combinations of pitch-class extraction algorithms (see Table 3.2), and 4 types of pitch-class aggregators (see Table 3.3). A final evaluation is performed using the 4 different types of classifiers: a neural network, a k-nearest neighbor algorithm, a support vector machine, and a naïve Bayes classifier. All of the evaluations for Phase I of the experiment are run with the training data sets described in Section 4.2.1.

It is clear that testing every permutation of these components would lead to a prohibitively large set of results. As such, the different versions of each component are only evaluated with a selected portion of permutations of the other components. This is accomplished by performing 4 different sub-phases of the experiment: A: frequency analysis, B: pitch-class extraction, C: pitch-class aggregation, and D: key classification.

Using the results from each sub-phase, only one set of parameters or algorithms are used for the subsequent sub-phases.

5.2.1 Sub-Phase A: Frequency Analysis

The 20 different permutations of frequency analysis parameters (see Table 3.1) are tested in combination with the basic mapping algorithm for pitch-class extraction (i.e., no extensions are used), the arithmetic mean pitch-class aggregator, and the k-nearest neighbor classifier with k=1 and 10-fold cross-validation.

5.2.2 Sub-Phase B: Pitch-Class Extraction

Each of the 8 different permutations of pitch-class extraction extensions (see Table 3.2) is evaluated with the best performing average frequency analysis parameters from Sub-phase A. The k-nearest neighbor classifier with k=1 is used with 10-fold cross-validation.

5.2.3 Sub-Phase C: Pitch-Class Aggregation

The best performing average prototype from Sub-phase B is tested in combination with the 4 different types of pitch-class aggregators (see Table 3.3). Once again, the k-nearest neighbor algorithm with k=1 is used with 10-fold cross-validation for evaluation.

5.2.4 Sub-Phase D: Key Classification

The 4 different classifiers (neural network, k-nearest neighbor, support vector machine, and naïve Bayes) are evaluated using the best performing average prototype from Sub-

phase C. This essentially provides a summary of the best performing prototype for Phase I.

5.2.5 Training Models

The data extracted (i.e., pitch-class distributions) using the best performing prototype is used to train a set of models that will be evaluated as part of Phase III of the experiment. One model is trained for each of the 4 types of classifiers and each of the 4 different training data sets, leading to a total of 16 models.

5.3 Phase II: Pitch-Class Template Evaluation

Phase II of the experiment trains each of the 4 different classifiers with each of the 6 different types of pitch-class templates described in Section 4.3. This leads to a total of 24 different models (i.e., 4 classifiers x 6 pitch-class templates for training). Each of these models is then evaluated using the each of the four training data sets described in Section 4.2.1. The purpose of this phase is to provide a means of comparison between the models trained with the pitch-class templates and those that were evaluated using cross-validation in Phase I of the experiment.

5.4 Phase III: Test Set Evaluation

Both Phase I and II of the experiment utilize the training data sets for evaluation. Since the training data sets were used to parameterize the system in Phase I, the results of these evaluations have possibly overfit the data. As such, Phase III of the experiment is designed to give a more reliable evaluation using the test data sets described in Section 4.2.2. Each of the 16 trained models from Phase I, and the 24 trained models from Phase

II, are evaluated using each of the 4 test data sets, which were not used during the training sessions.

Chapter 6

Results and Discussion

6.1 Phase I: Cross-Validation Evaluation

The first phase of the experiment is designed to parameterize the feature extraction parameters and algorithms. Sub-phase A evaluates the frequency analysis parameters, Sub-phase B evaluates the pitch-class extraction algorithms, Sub-phase C evaluates the pitch-class aggregators, and Sub-phase D shows the performance with the four different classifiers. The first three sub-phases (i.e., A, B, and C) all use the k-nearest neighbor algorithm with k=1 as the classifier. All of the sub-phases use 10-fold cross-validation on each of the four training data sets (see Section 4.2.1) for evaluation. In addition to results for each of the four data sets, the average result of the four data sets is also given.

6.1.1 Sub-Phase A: Frequency Analysis

For Sub-Phase A, three frequency analysis parameters, namely sampling rate, window size, and window overlap were varied. Table 6.1 shows the results of the evaluation using the four different training data sets (see Section 4.2.1) as well as the average of the four results.

The results show that the following frequency analysis parameters had the best performance for the average of the four training data sets: a sampling rate of 22,050 Hz, a

window size of 8192 samples, and a window overlap of 0.8. Therefore, these parameter values were selected for all subsequent phases of the experiment.

Sampling Rate (Hz)	Window Size	Window Overlap	Results (%)				
			Classical	Popular	MIDI	Combined	Average
11,025	1024	0	63.89	52.39	72.34	69.04	64.42
22,050	1024	0	55.58	42.91	61.91	55.41	53.95
44,100	1024	0	37.98	25.19	37.27	35.45	33.97
11,025	4096	0	73.36	61.96	74.68	76.01	71.50
22,050	4096	0	74.20	61.60	77.65	75.84	72.32
44,100	4096	0	66.51	53.01	73.44	67.41	65.09
11,025	8192	0	74.83	59.24	71.14	76.16	70.34
11,025	8192	0.5	75.28	60.85	75.26	77.41	72.20
22,050	8192	0	73.40	61.28	73.41	75.60	70.92
22,050	8192	0.5	75.32	64.27	75.47	77.04	73.03
22,050	8192	0.8	76.86	65.07	77.67	79.23	74.71
44,100	8192	0	70.79	64.75	76.60	76.12	72.07
44,100	8192	0.5	72.31	62.54	76.60	77.56	72.25
11,025	16,384	0	70.11	61.27	73.72	76.60	70.43
11,025	16,384	0.5	72.09	61.24	76.40	78.20	71.98
22,050	16,384	0	71.07	62.42	77.04	75.22	71.44
22,050	16,384	0.5	74.48	62.75	73.68	76.96	71.97
44,100	16,384	0	74.50	62.00	73.83	76.50	71.71
44,100	16,384	0.5	75.69	63.25	75.59	77.93	73.12
44,100	16,384	0.8	76.10	64.58	76.48	77.73	73.72

Table 6.1: The results of the frequency analysis parameter evaluation, varying the sampling rate, window size, and window overlap. Results are shown for each of the four training data sets (see Section 4.2.1) as well as the average for the four data sets. The row with the best average value is highlighted.

6.1.2 Sub-Phase B: Pitch-Class Extraction

The best average performing frequency analysis parameters identified in Sub-Phase A were used for Sub-Phase B. Table 6.2 shows the results of evaluating eight different combinations of pitch-class extraction extensions for each of the four training data sets, as well as the average of the four data sets.

The results of the evaluation indicate that the best average performance is achieved with the basic mapping algorithm in combination with the peak detection and low-frequency clarification extensions. As such, this combination of pitch-class extraction extensions was used for all subsequent phases of the experiment.

Pitch-Class Algorithm	Results				
	Classical	Popular	MIDI	Combined	Average
BA	76.86	65.07	77.67	79.23	74.71
BA + PD	77.13	67.38	79.16	77.31	75.25
BA + SFM	74.34	57.40	75.29	72.66	69.92
BA + LFC	75.56	66.23	79.84	79.14	75.19
BA + PD + SFM	71.84	59.56	74.99	75.28	70.42
BA + PD + LFC	78.34	67.35	79.45	80.42	76.39
BA + SFM + LFC	75.73	62.11	78.92	77.53	73.57
BA + PD + SFM + LFC	75.19	60.80	77.87	76.96	72.71

Table 6.2: The results of evaluating various combinations of pitch-class extraction extensions on each of the four training data sets. The average of the results from the four data sets is also given. The row with the highest average result is highlighted. BA: basic mapping algorithm, PD: peak detection, SFM: spectral flatness measure, LFC: low-frequency clarification. See Chapter 3 for details.

6.1.3 Sub-Phase C: Pitch-Class Aggregation

For Sub-Phase C, the best performing average parameters from Sub-phase A are used (i.e., a sampling rate of 22,050 Hz, a window size of 8192 samples, and a window overlap of 0.8) in combination with the best average performing pitch-class extraction algorithm from Sub-Phase B (i.e., the basic algorithm with the peak detection and low frequency clarification extensions). Each of the four variations of pitch-class aggregators is evaluated with the four training data sets. Table 6.3 presents the results for each of the four data sets, as well as the average of the four.

The results of the sub-phase show that the periodic cleanup aggregator with a period of 4.01 seconds had the best average performance. Therefore it was used for all subsequent phases of the experiment.

Pitch-Class Aggregator	Results (%)				
	Classical	Popular	MIDI	Combined	Average
AM	78.34	67.35	79.45	80.42	76.39
PC, period = 1.04 seconds	78.31	66.17	79.92	78.37	75.69
PC, period = 2.01 seconds	79.53	68.58	78.74	79.77	76.66
PC, period = 4.01 seconds	79.89	70.13	80.32	78.98	77.33

Table 6.3: The results of the pitch-class aggregator evaluation on each of the four training data sets, as well as the average result. The row with the highest average result is highlighted. AM: arithmetic mean, PC: periodic cleanup.

6.1.4 Sub-Phase D: Key Classification

For Sub-Phase D, the best performing average parameters from Sub-phase A are used (i.e., a sampling rate of 22,050 Hz, a window size of 8192 samples, and a window overlap of 0.8) in combination with the best average performing pitch-class extraction algorithm from Sub-Phase B (i.e., the basic algorithm with the peak detection and low frequency clarification extensions), and the best average performing pitch-class aggregator from Sub-Phase C (i.e., periodic cleanup with a 4.01 second period). Each of the four classifiers is evaluated using each of the four training data sets with cross-validation. Table 6.4 shows the results of the evaluation for each of the four training data sets, as well as the average result of the four data sets.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	79.89	70.13	80.32	78.98	77.33
Neural Network	76.38	63.21	77.55	79.30	74.11
Naïve Bayes	76.81	63.08	73.13	79.32	73.09
Support Vector Machine	67.01	61.37	67.61	72.06	67.01

Table 6.4: The results of the key classification using the four different classifiers with each of the four training data sets. The average result of the four data sets is also given. The best result in each column is highlighted.

6.1.5 Summary

Table 6.5 summarizes the frequency analysis parameters, pitch-class extraction algorithm, and pitch-class aggregator that were selected based on the results of Sub-phase A, B, and C of Phase I. The data (i.e., pitch-class distributions) extracted using this configuration are used to train a set of models that will be evaluated in Phase III. One of each of the four classifiers is trained with each of the four training data sets, leading to a total of sixteen trained models.

Sampling Rate (Hz)	Window Size	Window Overlap	Pitch-Class Extraction Algorithm	Pitch-Class Aggregator
22,050	8192	0.8	Basic mapping algorithm with peak detection and low-frequency clarification extensions	Periodic cleanup aggregator with a period of 4.01 seconds

Table 6.5: Summary of the frequency analysis parameters, pitch-class extraction algorithm, and pitch-class aggregator selected based on the results of Sub-phase A, B, and C of Phase I.

Table 6.6 shows the results of the best performing classifiers from Sub-phase D, for each of the four data sets as well as the average of the four data sets. In order to provide more detailed results for the best performing classifiers, the types of errors are also listed. A perfect 5th error refers to when a key is detected that is a perfect 5th (i.e., seven semitones) away from the correct key (e.g., detected key: C, correct key: G). A relative major/minor error refers to when a key is detected that is the relative major/minor of the correct key (e.g., detected key: C, correct key: Am). A parallel major/minor error refers to when a key is detected that is the parallel major/minor of the correct key (e.g., detected key: A, correct key: Am). Furthermore, the metric from the MIREX '05 audio key finding task is also given (MIREX score). This metric gives 1 point for the correct key, 0.5 points for perfect 5th errors, 0.3 points for relative major/minor errors, and 0.2 points for parallel major/minor errors. The MIREX score is then calculated by dividing the number of points by the total number of instances.

Data Set	Classifier	Perfect 5th Errors	Relative Major / Minor Errors	Parallel Major / Minor Errors	Other Errors	Raw Score (%)	MIREX Score (%)
Classical	K-NN	20	11	9	10	79.89	83.87
Popular	K-NN	15	8	8	15	70.13	77.67
MIDI	K-NN	20	10	5	6	80.32	87.08
Combined	NB	51	32	17	26	79.32	85.58
Average	K-NN	9.14%	4.89%	3.71%	5.74%	77.10	83.31

Table 6.6: The best performing classifiers for Phase I, using the parameters and algorithms listed in Table 6.5. The types of errors are shown as well as the score that is based on the MIREX '05 metric (MIREX score). The bottom row shows the best performing classifier for the average of the four data sets[1]. K-NN: k-nearest neighbor, NB: naïve Bayes.

6.1.6 Discussion

The results of Sub-phase A (Table 6.1) show a great deal of variation in performance based on the selected frequency analysis parameters. As a general trend, we see that the performance tends to increase with the amount of window overlap. This is perhaps attributable to the fact that as the temporal resolution decreases with the window size, a larger window overlap is required to compensate. A clear pattern is also apparent that the worst performance is achieved with the smallest window size of 1024 samples, most likely due to the decreased frequency resolution.

An analysis of the results of Sub-phase B (Table 6.2) indicates that the use of the spectral flatness measure extension actually decreases the performance in all cases. On the other hand, both the peak detection and low-frequency clarification extensions lead to improved results in all cases.

The results of Sub-phase C (Table 6.3) show that the periodic cleanup procedure tends to improve the performance, albeit only slightly. Only in the case where the period

was set to 1.01 seconds was the performance decreased relative to the arithmetic mean aggregator.

The results of Sub-phase D (Table 6.4) show that the k-nearest neighbor algorithm performs better than the other classifiers in almost every case (i.e., for the classical, popular, and MIDI data sets, as well as the average result). Only in the case of the combined data set did the naïve Bayes classifier and neural network outperform the k-nearest neighbor algorithm by a slight margin. It should be noted, however, that the system was parameterized using the k-nearest neighbor algorithm for the first three sub-phases.

6.2 Phase II: Pitch-Class Template Evaluation

For the second phase of the experiment, each of the four different types of classifiers (a neural network, a k-nearest neighbor algorithm, a support vector machine, and naïve Bayes classifier) is trained with each of the six different types of pitch-class templates described in Section 4.3. This leads to a total of twenty-four different trained models. Each of the trained models is then evaluated using each of the four training data sets described in Section 4.2.1. These results provide a means of comparison to the results from Phase I, which are based solely on cross-validation.

6.2.1 Results

The results are shown in tables for each type of pitch-class template that was used to train the classifiers. Each table lists the results of the evaluation for the four different training data sets, as well as the average result for the four data sets. Table 6.7 shows the results for the classifiers trained with the diatonic pitch-class template. Table 6.8 shows the results for the classifiers trained with the Krumhansl pitch-class template. Table 6.9

shows the results for the classifiers trained with the Temperley pitch-class template. Table 6.10 shows the results for the classifiers trained with the Krumhansl-diatonic pitch-class template. Table 6.11 shows the results for the classifiers trained with the Temperley-diatonic pitch-class template. Lastly, Table 6.12 shows the results for the classifiers trained with the Papadopoulos pitch-class template.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	57.66	52.67	63.16	57.17	57.67
Neural Network	45.97	40.67	51.67	46.62	46.23
Support Vector Machine	62.50	56.00	66.51	62.27	61.82
Naïve Bayes	48.79	41.33	50.24	47.45	46.95

Table 6.7: Diatonic templates used to train the classifiers.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	64.52	80.67	63.64	63.43	68.07
Neural Network	54.44	54.00	46.41	51.57	51.61
Support Vector Machine	63.71	74.67	58.37	64.58	65.33
Naïve Bayes	57.66	48.00	57.89	55.35	54.73

Table 6.8: Krumhansl templates used to train the classifiers.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	69.35	76.67	72.25	71.50	72.44
Neural Network	68.15	70.00	68.42	68.70	68.82
Support Vector Machine	71.37	80.67	70.33	73.31	73.92
Naïve Bayes	59.27	60.67	62.20	60.63	60.69

Table 6.9: Temperley templates used to train the classifiers.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	62.50	82.67	65.07	67.55	69.45
Neural Network	64.52	76.00	63.16	66.89	67.64
Support Vector Machine	65.73	81.33	64.11	69.03	70.05
Naïve Bayes	37.10	48.00	30.14	37.40	38.16

Table 6.10: Krumhansl-diatonic templates used to train the classifiers.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	68.55	71.33	70.81	69.69	70.10
Neural Network	70.16	68.00	73.21	70.68	70.51
Support Vector Machine	68.95	74.67	70.81	71.00	71.36
Naïve Bayes	58.87	55.33	53.11	56.01	55.83

Table 6.11: Temperley-diatonic templates used to train the classifiers.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	58.06	68.00	48.33	57.00	57.85
Neural Network	57.66	74.00	48.80	58.65	59.78
Support Vector Machine	56.85	75.33	48.80	58.65	59.91
Naïve Bayes	58.87	42.00	59.81	55.02	53.93

Table 6.12: Papadopoulos templates used to train the classifiers.

6.2.2 Summary

Table 6.13 gives a summary of the average result (i.e., the average of the results for the four training data sets: classical, popular, MIDI, and combined) for each type of classifier and each type of pitch-class template that was used for training. The average of the average results for each type of pitch-class template (rightmost column) and each type of classifier (bottom row) are also shown.

Table 6.14 presents the results of the best performing models (i.e., classifier and training template type) for Phase II. The types of errors are shown as well as the score using the metric from the MIREX '05 audio key finding task (MIREX score). See Section 6.15 for details on the types of errors and the MIREX metric.

Template	Results (%)				
	K-NN	NN	SVM	NB	AVERAGE
Diatonic	57.67	46.23	61.82	46.95	53.17
Krumhansl	68.07	51.61	65.33	54.73	59.94
Temperley	72.44	68.82	73.92	60.69	68.97
Krumhansl-diatonic	69.45	67.64	70.05	38.16	61.33
Temperley-diatonic	70.10	70.51	71.36	55.83	66.95
Papadopoulos	57.85	59.78	59.91	53.93	57.87
AVERAGE	79.12	72.92	80.48	62.06	67.51

Table 6.13: A summary of the average result of the four training data sets for each of the classifiers and each of the pitch-class templates. The rightmost column shows the average of the average results for each type of pitch-class template. The bottom row gives the average of the average results for each type of classifier. The bottom-right cell shows the average of all the results. K-NN: k-nearest neighbor, NN: neural network, SVM: support vector machine, NB: naïve Bayes.

Data Set	Classifier	Template	Perfect 5th Errors	Relative Major / Minor Errors	Parallel Major / Minor Errors	Other Errors	Raw Score (%)	MIREX Score (%)
Classical	SVM	Temperley	33	19	10	9	71.37	81.13
Popular	K-NN	Krumhansl-diatonic	14	4	5	3	82.67	88.80
MIDI	NN	Temperley-diatonic	23	14	7	12	73.21	81.39
Combined	SVM	Temperley	76	39	22	25	73.31	82.22
Average	SVM	Temperley	12.33%	6.78%	3.58%	4.06%	73.92	82.69

Table 6.14: The best performing models for Phase II, using the parameters and algorithms selected from Phase I (see Table 6.5). The types of errors are shown as well as the score that is based on the MIREX '05 metric (MIREX score). The bottom row shows the best performing model for the average of the four data sets[1]. K-NN: k-nearest neighbor, NN: neural network, SVM: support vector machine, NB: naïve Bayes.

6.2.3 Discussion

An analysis of the results presented in Table 6.13 reveals that the combination of classifier and pitch-class template used for training greatly influences the performance of the model. For instance, when the Krumhansl template is used to train the neural network and naïve Bayes classifiers, the average accuracy is 51.61% and 54.73%, respectively. Whereas when the Krumhansl-diatonic template is used to train the same classifiers, the average accuracy is 67.64% and 38.16%, respectively. This suggests that if any particular classifier is chosen for audio key detection, it is important to carefully consider the type of pitch-class template that is used for training, as it can greatly affect performance. However, the results also seem to indicate that the classifiers trained with the Temperley and Temperley-diatonic pitch-class templates perform well in almost every case.

We can also see from Table 6.11 that the k-nearest neighbor and support vector machines have the best average performance for the various types of training templates. In the case of the k-nearest neighbor classifier, this could be due to specialization, since the feature extraction algorithm was parameterized using the k-nearest neighbor classifier with the training data sets in Phase I. However, the fact that the support vector machine had the best overall average performance seems indicative that it is a good choice of classifier, at least in the case when pitch-class templates are used for training.

Looking at the results for the first three data sets in Table 6.14 (Classical, Popular, and MIDI), we see that the best performing models in each of the three cases consists of different classifiers and pitch-class templates for training. This suggests that the choice of classifier and training template is also heavily dependent on the corpus of music that is being analyzed, which supports the hypothesis of Gómez (2006b).

6.3 Phase III: Test Set

The third phase of the experiment consists of evaluating all of the trained models from Phase I and II of the experiment with the test data sets. There are sixteen models from Phase I trained with the training data sets (see Section 6.15) and twenty-four models from Phase II trained with the pitch-class templates (see Section 6.2). Each of the models is evaluated with each of four test data sets (see Section 4.2.2). This phase is particularly important because it provides a set of results that is completely independent from the training data.

6.3.1 Results

The results are shown in tables for each type of training data or pitch-class template that was used to train the classifiers. Each table lists the results of the evaluation for the four

different test data sets, as well as the average result of the four test data sets. Table 6.15 shows the results for the classifiers trained with the diatonic pitch-class template. Table 6.16 shows the results for the classifiers trained with the Krumhansl pitch-class template. Table 6.17 shows the results for the classifiers trained with the Temperley pitch-class template. Table 6.18 shows the results for the classifiers trained with the Krumhansl-diatonic pitch-class template. Table 6.19 shows the results for the classifiers trained with the Temperley-diatonic pitch-class template. Table 6.20 shows the results for the classifiers trained with the Papadopoulos pitch-class template. Table 6.21 shows the results for the classifiers trained with the classical training data set. Table 6.22 shows the results for the classifiers trained with the popular training data set. Table 6.23 shows the results for the classifiers trained with the MIDI training data set. Lastly, Table 6.24 shows the results for the classifiers trained with the combined training data set.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	50.00	41.07	62.50	63.33	54.23
Neural Network	45.83	23.21	62.50	50.00	45.39
Support Vector Machine	54.17	42.86	70.83	66.67	58.63
Naïve Bayes	33.33	25.00	37.50	33.33	32.29

Table 6.15: Diatonic templates used to train the classifiers.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	33.33	73.21	50.00	60.00	54.14
Neural Network	37.50	35.71	37.50	36.67	36.85
Support Vector Machine	37.50	67.86	50.00	56.67	53.01
Naïve Bayes	37.50	37.50	33.33	36.67	36.25

Table 6.16: Krumhansl templates used to train the classifiers.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	54.17	60.71	75.00	66.67	64.14
Neural Network	41.67	58.93	66.67	70.00	59.32
Support Vector Machine	50.00	73.21	70.83	70.00	66.01
Naïve Bayes	37.50	53.57	45.83	53.33	47.56

Table 6.17: Temperley templates used to train the classifiers.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	50.00	69.64	58.33	56.67	58.66
Neural Network	58.33	66.07	62.50	60.00	61.73
Support Vector Machine	45.83	67.86	58.33	60.00	58.01
Naïve Bayes	25.00	53.57	12.50	26.67	29.44

Table 6.18: Krumhansl-diatonic templates used to train the classifiers.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	66.67	66.07	75.00	66.67	68.60
Neural Network	62.50	50.00	66.67	66.67	61.46
Support Vector Machine	62.50	69.64	75.00	73.33	70.12
Naïve Bayes	45.83	51.79	50.00	53.33	50.24

Table 6.19: Temperley-diatonic templates used to train the classifiers.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	12.50	64.29	25.00	33.33	33.78
Neural Network	41.67	66.07	50.00	56.67	53.60
Support Vector Machine	12.50	67.86	25.00	33.33	34.67
Naïve Bayes	41.67	32.14	50.00	53.33	44.29

Table 6.20: Papadopoulos templates used to train the classifiers.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	54.17	50.00	58.33	60.00	55.63
Neural Network	45.83	57.14	58.33	56.67	54.49
Support Vector Machine	37.50	39.29	37.50	40.00	38.57
Naïve Bayes	41.67	37.50	48.83	50.00	44.50

Table 6.21: Extracted pitch-class distributions from the classical training data set from Phase I used to train the classifiers.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	29.17	67.86	29.17	43.33	42.38
Neural Network	25.00	60.71	37.50	40.00	40.80
Support Vector Machine	20.83	46.43	29.17	23.33	29.94
Naïve Bayes	33.33	53.57	33.33	30.00	37.56

Table 6.22: Extracted pitch-class distributions from the popular training data set from Phase I used to train the classifiers.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	66.67	41.07	79.17	60.00	61.73
Neural Network	66.67	32.14	75.00	60.00	58.45
Support Vector Machine	50.00	32.14	54.17	50.00	46.58
Naïve Bayes	66.67	44.64	62.50	60.00	58.45

Table 6.23: Extracted pitch-class distributions from the MIDI data training set from Phase I used to train the classifiers.

Classifier	Results (%)				
	Classical	Popular	MIDI	Combined	Average
K-nearest Neighbor, K=1	66.67	69.64	66.67	66.67	67.41
Neural Network	62.50	71.43	70.83	73.33	69.52
Support Vector Machine	54.17	57.14	50.00	50.00	52.83
Naïve Bayes	66.67	53.57	66.67	66.67	63.40

Table 6.24: Extracted pitch-class distributions from the combined training data set from Phase I used to train the classifiers.

6.3.2 Summary

Table 6.25 gives a summary of the average result (i.e., the average of the results for the four test data sets: classical, popular, MIDI, and combined) for each type of classifier and each type of pitch-class template or training data set. The average of the average results for each type of pitch-class template (rightmost column) and each type of classifier (bottom row) are also shown.

Table 6.26 presents the results of the best performing models (i.e., classifier and training data set or template type) for Phase III. The types of errors are shown as well as the score using the metric from the MIREX '05 audio key finding task (MIREX score). See Section 6.15 for details on the types of errors and the MIREX metric. As a means of comparison, Table 6.27 shows the results from the MIREX '05 audio key finding contest participants. The entries to the contest were evaluated with a set of 1252 pieces from the Baroque, Classical, and Romantic periods, synthesized from MIDI using two different synthesizers. This led to a total of 2504 instances in the test data set. Details on the entries can be found in Chapter 2 (see Table 2.2).

Template / Training Data	Results (%)				
	K-NN	NN	SVM	NB	AVERAGE
Diatonic	54.23	45.39	58.63	32.29	47.64
Krumhansl	54.14	36.85	53.01	36.25	45.06
Temperley	64.14	59.32	66.01	47.56	59.26
Krumhansl-diatonic	58.66	61.73	58.01	29.44	51.96
Temperley-diatonic	68.60	61.46	70.12	50.24	62.61
Papadopoulos	33.78	53.60	34.67	44.29	41.59
Classical training data set	55.63	54.49	38.57	44.50	48.30
Popular training data set	42.38	40.80	29.94	37.56	37.67
MIDI training data set	61.73	58.45	46.58	58.45	56.30
Combined training data set	67.41	69.52	52.83	63.40	63.29
AVERAGE	56.07	54.16	50.84	44.4	51.37

Table 6.25: A summary of the average of the results of the four test data sets. An average is shown for each type of classifier and each type of template or training data set. The rightmost column shows the average of the average results for each type of pitch-class template or training data set. The bottom row gives the average of the average results for each type of classifier. The bottom-right cell shows the average of all the results. K-NN: k-nearest neighbor, NN: neural network, SVM: support vector machine, NB: naïve Bayes.

Data Set	Classifier	Template / Training Data	Perfect 5th Errors	Relative Major / Minor Errors	Parallel Major / Minor Errors	Other Errors	Raw Score (%)	MIREX Score (%)
Classical	K-NN	Temperley-diatonic	2	5	0	1	66.67	77.08
	K-NN	MIDI training data set	3	2	0	3	66.67	75.42
	NN	MIDI training data set	4	1	1	2	66.67	77.08
	NB	MIDI training data set	3	1	0	4	66.67	74.17
	K-NN	Combined training data set	5	0	0	3	66.67	77.08
	NB	Combined training data set	2	2	1	3	66.67	74.17
Popular	K-NN	Krumhansl	6	5	2	2	73.21	81.96
	SVM	Temperley	4	6	0	5	73.21	80.00
MIDI	K-NN	MIDI training data set	1	1	0	3	79.17	82.50
Combined	SVM	Temperley-diatonic	2	5	0	1	73.33	81.67
	NN	Combined training data set	2	2	0	4	73.33	77.67
Average	SVM	Temperley-diatonic	6.43%	14.62%	0%	4.70%	70.12	78.96

Table 6.26: The best performing models for Phase III, using the parameters and algorithms selected from Phase I (see Table 6.5). The types of errors are shown as well as the score that is based on the MIREX '05 metric (MIREX score). The bottom row shows the best performing model for the average of the four test data sets[1]. K-NN: k-nearest neighbor, NN: neural network, SVM: support vector machine, NB: naïve Bayes.

Participant	Perfect 5th Errors	Relative Major / Minor Errors	Parallel Major / Minor Errors	Other Errors	Raw Score (%)	MIREX Score (%)
Izmirli, Ö.	78	69	35	147	86.86	89.52
Purwins & Blankertz	116	45	37	156	85.86	89.01
Gómez, E. (start)	79	81	45	217	83.15	86.05
Gómez, E. (global)	142	121	43	164	81.23	85.86
Pauws, S.	43	136	63	209	81.99	84.98
Zhu, Y.	104	75	57	270	79.79	83.22
Chuan & Chew	178	134	42	108	73.56	79.06
AVERAGE	4.22%	3.77%	1.84%	7.25%	81.78	85.39

Table 6.27: The results of the models entered in to the MIREX '05 audio key finding contest. The bottom row shows the average results for all of the entries[1]. Each entry was evaluated with a test data set of 2504 instances of audio files synthesized from MIDI. See Table 2.2 for details on the entries.

6.3.3 Discussion

Looking at the rightmost column of Table 6.25 we can compare the average performance for the different types of templates or training data sets that were used to train the classifiers. The best average performance (i.e., average of all the classifiers) of 63.29% is achieved using the combined training data set (see Section 4.2.2). This suggests that using extracted pitch-class distributions from large sets of ground truth data is a viable option for training key classifiers. It is also apparent that the Temperley and Temperley-diatonic pitch-class templates are effective at training the classifiers, as they had the next best average performance. This is further supported by the fact that these templates also

had the best average performance in Phase II of the experiment. On the other hand, the classifiers trained with the popular data set had a relatively poor average accuracy of 37.67%. This could be due to the fact that the musical excerpts from the popular training data set are too dissimilar from those in the test data sets. It may also be skewed by the fact the popular training data set contains the least number of instances compared to the other training data sets.

The bottom row of Table 6.25 reveals that the k-nearest neighbor classifier outperformed the other three classifiers on average. However, it should be noted that when considering just the classifiers trained with pitch-class templates (i.e., the trained models from Phase II, in the top six rows of results), the support vector machine seems to perform the best. Furthermore, the bottom row of Table 6.26 shows that the best average performance for all of the test data sets was achieved with the support vector machine trained with the Temperley-diatonic template. This supports the findings from Phase II of the experiment, in which the support vector machine also had the best average performance.

An analysis of Table 6.26 provides information on what combinations of classifiers and training data performed the best on each of the four test data sets. For example, there are two combinations that had the same raw score of 73.21% for the popular test data set: the k-nearest neighbor classifier trained with the Krumhansl template, and the support vector machine trained with the Temperley template. However, the MIREX score of 81.96% for the k-nearest neighbor model indicates that it actually outperformed the support vector machine. The best overall performance was achieved with the MIDI test data set using a k-nearest neighbor classifier trained with the MIDI training data set. The raw score and MIREX score are 79.17% and 82.50%, respectively. This result is perhaps to be expected, as the data sets of musical excerpts synthesized from MIDI are more simplistic than excerpts of real recordings.

The k-nearest neighbor algorithm trained with the Krumhansl templates produced the best results for the popular test data set, receiving a raw score of 73.21% and a MIREX

score of 81.96%. This is significantly better than the best results for the classical test data set, which had a raw score of 66.67% and a MIREX score of 77.08%. This may be attributable to the fact that the classical test data set is substantially smaller than the popular test data set. It may also be a result of the excerpts in the popular test data set being less harmonically complex than those in the classical set. In any event, this seems indicative that reasonable performance is achievable for audio key detection with popular music.

The bottom three rows of Table 6.26 show the best performance for the combined test data set and the best average performance for all of the test data sets. The support vector machine trained with the Temperley-diatonic template produced both of these results, suggesting that this configuration is perhaps the most robust solution when detecting the key from multiple genres of musical excerpts. However, further evaluations with more ground truth data would be needed to substantiate this hypothesis.

We can get and idea of how the models presented in this thesis perform relative to previous implementations by comparing the results in Table 6.26 with those in Table 6.27 (i.e., the MIREX '05 audio key finding contest results). Since the MIREX contest was evaluated using audio synthesized from MIDI files, the most meaningful comparison would be made with the results for the MIDI test data set. The average raw score for the MIREX contest entries was 81.78% and the average score using the MIREX metric was 85.39% (see the bottom row of Table 6.27). The best performing model for the MIDI test data set produced a raw score of 79.17% and a score of 82.50% using the MIREX metric, which suggests that the performance is comparable with the entries to the MIREX contest. This is, however, not conclusive because the MIDI test data set only consists of twenty-four instances, whereas the MIREX evaluation consisted of 2504 instances.

The results suggest that the models implemented for this thesis can perform audio key detection with a reasonable degree of accuracy. Overall, the support vector machine classifier trained with the Temperley-diatonic pitch-class template produces the most consistent accuracy across the different test data sets. Further parameterization and

evaluation with this configuration would be warranted for future research. Moreover, an expanded set of ground truth data for training and testing is needed to verify these findings, and could very well lead to improved performance.

---

[1] For the row showing the best performing model/classifier for the average of the data sets, the types of errors are shown as percentages. The percentage of each type of error is calculated for each data set (i.e. the percentage of the type of error relative to the size of the data set). The values in this row are then calculated as the average of the percentages of each type of error for all of the data sets. For example, if the four data sets have 2%, 5%, 3%, and 7% perfect 5th errors, then the value for the perfect 5th errors in this row would be 4.25%.

Chapter 7

Conclusions

This thesis has approached the audio key finding problem by evaluating a number of algorithms and classifiers with the goal of implementing a model that maximizes key identification accuracy. The methods employed are extensions of previous research in the field and are essentially built on the assumption that the key of an audio excerpt can be identified based solely on the extracted pitch-class distribution. In order to identify the key from the pitch-class distribution, a classifier that is trained with ground truth data or pitch-class templates is employed.

The systematic and modular evaluation of various parameters, algorithms, classifiers, and training data led to insights on what configurations performed well in different circumstances. It was shown that the choice of signal processing parameters and algorithms used for feature extraction had a significant impact on the results. The pairing of classifier and training data also greatly influenced the performance of the system. Moreover, it was found that certain configurations performed better depending on the corpus of music that was being used for evaluation (e.g., classical or popular). Overall, the most consistently accurate results, regardless of the corpus of music, were achieved with the support vector machine that was trained with the Temperley-diatonic pitch class template. Parameterization of the feature extraction algorithm and further evaluation with this configuration would be a worthwhile endeavor for future research. However, a larger set of ground truth data is needed in order to ascertain more conclusive observations about the performance of the models in general.

The findings of this thesis, along with previous research in the field, suggest that a distributional approach to audio key detection (i.e., using pitch-class distributions) can yield reasonable performance. However, it is possible that the accuracy of a system based on this approach is bounded by a semantic gap between the relatively simplistic pitch-class distribution feature and the high-level concept of key. Research has shown that such a semantic gap exists for other closely related tasks in the field of music information retrieval (Cano et al. 2005a). For example, Aucouturier and Pachet (2004) conducted an experiment in which they identified a glass ceiling of approximately 65% accuracy for audio similarity applications, regardless of the type of system that was employed. Similar findings have also been reported for polyphonic melody extraction (Paivo 2007), as well as timbre and rhythm recognition (Lu et al. 2006). If this is the case for audio key detection, it is important for future research to investigate new approaches that go beyond a purely distributional view.

Cognitive studies, such as that of Temperley and Marvin (2008), have suggested that in addition to the distribution of pitch-classes, listeners also use structural cues in order to identify the key of music. Although it is unclear exactly what these structural cues are, it appears that the temporal ordering and arrangement of notes do play a role in how humans perceive tonality. This is further supported by the findings of Madsen and Wilmer (2007). They show that the accuracy of their symbolic key detection system is improved by incorporating a probabilistic model for the transition between scale degrees. Incorporation of this type of temporal information could also be used to improve the performance of audio key finding systems.

Many recently proposed audio key detection systems also make use of chord recognition techniques in order to help ascertain the key. This approach has been shown to be successful, particularly for systems that perform local key estimation. However, the state-of-the-art in chord detection is still fairly rudimentary and there is much room for improvement. For example, chord detection systems, for the most part, are still unable to

identify inversions. Therefore it stands to reason that as advances are made in chord detection techniques, we could also see major improvements in audio key detection.

Regardless of the methods that are utilized, it seems that one of the most significant roadblocks to improving the accuracy of audio key detection systems is the lack of a standardized evaluation procedure and ground truth data set. Without a common method for assessing the performance of various approaches, comparison is extremely difficult. Although the MIREX audio key finding competition marks a step in the right direction, it is not without its shortcomings. Firstly, the data set used for evaluation consists solely of audio excerpts that have been synthesized from MIDI, which does not encompass the full complexity of the audio key finding problem. Furthermore, only pieces from the classical, baroque, and romantic periods are included. Ideally the data used for evaluation would include recordings of real performances from multiple genres of music. The second problem with the MIREX procedure is that the statistics reported from the results do not provide sufficient information on when and why the models fail. A more comprehensive evaluation would take a modular approach, such that the various components of the systems (e.g., frequency analysis, pitch-class extraction, key classifiers) could be assessed independently. With this type of procedure we could gain a better understanding of how each component of the system influenced the results, leading to insights that facilitate future research.

Appendix A

Previous Audio Key Detection Systems

This appendix presents a summary of the audio key detection systems that are reviewed in Section 2.3.2.

Author(s)	Algorithm Summary	Data for Training & Evaluation	Max. Accuracy	Related Work
Leman (1991, 1994)	Local tone centers extracted. Pattern-matching algorithm used to identify key.	• "Through the Keys", Bartok • Excerpts from Sextet No. 2, Brahms Excerpt of Prelude No. 20, Chopin • Excerpt of Arabesque No. 1, Debussy	N/A	
Izmirli and Bilgen (1994)	FFT to extract sequence of note intervals and onset times. Pattern-matching algorithm used to produce tonal context vector.	• "Solfege des Solfege", Lavignac, Vol. 2A, No. 4	N/A	
Izmirli and Bilgen (1996)	CQ-transform with peak selection algorithm produces set of notes for each time step. Same pattern-matching algorithm as previous implementation.	• Excerpt of Op. 34, No. 2, by Chopin	N/A	Izmirli and Bilgen (1994)
Purwins et al. (2000)	CQ-transform extracts pitch-class distributions. Fuzzy distance algorithm used to compare with templates based on probe tones.	• C minor Prelude, Op. 28, No. 20, by Chopin	N/A	
Pauws (2004)	Standard pitch-class distribution extraction with *maximum-key profile* algorithm to compare with Krumhansl templates.	• 237 classical piano sonatas	66.2%	
van de Par et al. (2006)	Extension of Pauws (2004) algorithm that uses 3 different temporal weighting functions	• 237 classical piano sonatas, composers: J.S. Bach, Shostakovich, Brahms, and Chopin	98.1%	Direct extension of Pauws (2004)
Martens et al. (2004)	Pitch patterns extracted from the audio signal using an auditory model. Classification tree is used to identify key from pitch patterns.	• Templates created from 24 sequences of Shepard chords • Evaluation performed with excerpt of "Eternally", by Quadran, a passage of "Inventions No. 1 in C major", by J.S. Bach, and an excerpt of "Children", by Robert Miles	N/A	Uses bottom-up tonal center extraction proposed by Leman (2000)

Author(s)	Algorithm Summary	Data for Training & Evaluation	Max. Accuracy	Related Work
Gómez and Herrera (2004a)	Extraction of the Harmonic Pitch Class Profile (HPCP), see Gómez (2006b) for details. Machine-learning algorithms, including binary trees, Bayesian estimation, neural networks, and support vector machines used for classification. Comparison also made with cognition inspired model as well as a combined approach (i.e., cognition inspired + machine learning)	• 878 excerpts of classical music (661 for training, 217 for evaluation), composers: Mozart, Chopin, Scarlatti, Bach, Brahms, Beethoven, Handel, Pachelbel, Tchaikovsky, Sibelius, Dvorak, Debussy, Telemann, Albinoni, Vivaldi, Pasquini, Glenn Gould, Rachmaninoff, Schubert, Shostakovich, Haydn, Benedetto, Elgar, Bizet, Liszt, Boccherini, Ravel, and Debussy	84%	HPCP based on PCP by Fujishima (1999)
Chuan and Chew (2005a)	FFT used to extract pitch-class distribution from the signal. Key classification is achieved by mapping to a point in the Spiral Array.	• 15 second excerpts of 61 renditions of 28 symphonies (1st movement only), by Mozart	96%	Based on the Spiral Array model (Chew 2002)
Chuan and Chew (2005c)	Fuzzy analysis techniques used to improve quality of extracted pitch-class distributions. CEG with Spiral Array model is used for key classification.	• 15 second excerpts from 410 classical audio files recorded from MIDI.	75.25%	Same as MIREX '05 algorithm (Chuan and Chew 2005c)
Chuan and Chew (2007)	Basic algorithm is the same as (Chuan and Chew 2005a). Three extensions are proposed: the modified spiral array, fundamental frequency identification, and post-weight balancing.	• "Twenty-Four Preludes," by Chopin	~70%	Uses same basic algorithm as (Chew 2005a)
Izmirli (2005a)	FFT is mapped to pitch-classes in order to extract the distribution. Spectral flatness measure is used to disregard frequencies not containing peaks. Pitch-class distributions are compared to various templates in order to estimate key.	• 85 classical music pieces from common practice period, composers: J. S. Bach, Beethoven, Brahms, Chopin, Clementi, Corelli, Dvorak, Handel, Haydn, Hoffman, Kraus, Mozart, Pachelbel, Scarlatti, Schubert, Scriabin, Telemann, Tchaikovsky, and Vivaldi	86%	

Author(s)	Algorithm Summary	Data for Training & Evaluation	Max. Accuracy	Related Work
Izmirli (2006)	Same model as (Izmirli 2005a) but uses dimensionality reduction techniques to reduce pitch-class distribution dimensions.	• First 30 second excerpts of 152 pieces from the classical period, composers: Albinoni, Albrechtsberger, Alkan, Bach, C. P. E. Bach, Beethoven, Bella, Brahms, Chopin, Clementi, Corelli, Dvorak, Grieg, Handel, Haydn, Hofmann, Kraus, Liszt, Mendelssohn, Mozart, Pachelbel, Paganini, Prokofiev, Rachmaninov, Scarlatti, Schubert, Scriabin, Telemann, Tchaikovsky, and Vivaldi	88.7%	(Izmirli 2005a)
Izmirli (2007)	Non-negative matrix factorization used to perform local key-finding. Same correlational model as (Izmirli 2005a) used to identify key for a series of windowed pitch-class distributions.	• First 30 second excerpts of 152 pieces from the classical period (same as Izmirli 2006) • 17 pop songs with at least one modulation • 17 short excerpts of classical pieces with at least one modulation	82.4%	(Izmirli 2005a)
Gómez (2006)	Exhaustive study on various approaches to audio key detection. Thorough analysis of pertinent aspects of audio feature computation, evaluation strategies, and various models for tonality induction.	N/A	N/A	
Zhu et al. (2005, 2006)	Four step process for feature extraction: (1) CQ-transform; (2) tuning correction; (3) extract note partials; (4) create pitch-class distribution from note partials using consonance filtering and pitch profiling process. Two step process for key detection: (1) detect scale root; (2) detect scale mode.	• 60 pop songs (pop-rock style with vocals and instrumentation, including drums) • "The Four Seasons," by Vivaldi (4 parts, 3 movements each)	91%	

Author(s)	Algorithm Summary	Data for Training & Evaluation	Max. Accuracy	Related Work
Harte et al. (2006)	CQ-transform used to extract pitch-class distribution, which is mapped to 6-D tonal centroid feature. Algorithm applied for chord recognition.	• 16 songs, by The Beatles	N/A	Pitch-class distribution created from CQ-transform based on algorithm from Harte and Sandler (2005)
Gatzsche et al. (2006)	CQ-transform employed to extract pitch-class distribution, which is used as input to the circular pitch space model in order to detect the key.	• First 80 second excerpt of "Sonate für Cello und Klavier in a-Moll (D. 821), Allegro moderato," by Schubert • "C-major prelude," BWV 846, by J. S. Bach	N/A	Harte et al. (2006)
Chai and Vercoe (2005)	24-bin pitch-class distribution used as input to two HMMs: one to detect scale root and one to detect mode.	• 10 classical piano pieces, composers: Mozart, Chopin, Dvork, Rubenstein, Paderewski, Beethoven, and Schumann	~83%	Related to chord segmentation system by Sheh and Ellis (2003)
Peeters (2006a, 2006b)	Pitch-class distributions used as input to 24 different HMMs (one for each possible key).	• 302 European baroque, classical, and romantic music pieces, composers: J. S. Bach, Corelli, Handel, Telleman, Vivaldi, Beethoven, Haydn, Mozart, Brahms, Chopin, Dvorak, Schubert, and Schuman	81%	Comparison made between the methods for interpreting global key used by Gómez (2006a) and Izmirli (2005a)
Noland and Sandler (2007)	A HMM system that is initialized based on tone-profile values is used to investigate the effects of low-level DSP parameters.	• 110 pop songs, by The Beatles • "Well-Tempered Clavier, Book I," by J. S. Bach	98%	Based on previous model by Noland and Sandler (2006), which follows the work of Bello and Pickens (2005)

Author(s)	Algorithm Summary	Data for Training & Evaluation	Max. Accuracy	Related Work
Burgoyne and Saul (2005)	A Dirichlet-based HMM model for tracking chords and key simultaneously.	• 15 movements of 5 symphonies, by Mozart (training) • "Minuet, Symphony No. 40," by Mozart (evaluation)	83%	Directly implements the PCP by Fujishima (1999)
Lee and Slaney (2007)	24 separate HMMs (one for each key), each with 24 states (representing different chords) used to track chords and key simultaneously.	• 1046 audio files (rock), synthesized from MIDI (training) • 28 pop songs by The Beatles (evaluation)	84.62%	6-D centroid vector from Harte et al. (2006) used as input to the HMM. Also related to work by Peeters (2006b).
Catteau et al. (2007)	Frame-by-frame pitch-class distributions are extracted from the audio signal. A unified probabilistic framework that predicts chord/key transitions is used to label each of the frames.	• 10 polyphonic audio excerpts (60 seconds) • 96 MIDI-to-wave synthesized audio excerpts • 144 classical cadence excerpts • 20 chord sequences	82%	System is an extension of the work by Bello and Pickens (2005)
Papadopoulos and Peeters (2009)	Local key-finding system that combines various approaches. Three stages: (1) pitch-class distribution extraction; (2) metric structure estimation by detecting chord progressions and downbeats; (3) local key estimation using an HMM with observation probabilities derived from pitch-class templates.	• 5 movements of piano sonatas, by Mozart	80.22%	Global key estimation from Gómez and Herrera (2004a)
Shenoy et al. (2004)	Audio signal segmented in quarter note frames and pitch-class distribution extracted for each frame. Rule-based model is used to infer presence of chords and key for each frame.	• 20 pop English songs	90%	Build on the idea of Goto (2001) and Goto and Muraoka (1999) to incorporate high-level music knowledge

Appendix B

Training Set Excerpts

Detailed information on the musical excerpts used for the training sets is provided in the following pages. Appendix B.1 lists the excerpts used from the classical corpus, Appendix B.2 lists the excerpts used from the popular corpus, and Appendix B.3 lists the excerpts used from the MIDI corpus.

B.1 Training Set Excerpts from the Classical Corpus

Composer	Conductor / Performer(s)	Release Title	Label, Release Date	Excerpts
J. S. Bach	Glenn Gould	The Well–Tempered Clavier II: Preludes and Fugues	Sony Classical, 1993	48 (BMV 870-893)
L. Beethoven	Claudio Arrau, János Starker, Henryk Szeryng, and Bernard Haitink conducting the Royal Concertgebouw Orchestra	The Complete Piano Sonatas & Concertos	Philips Classics, 1998	8 (Concertos No. 1–5; Piano Variations in C Op. 120; Piano variations & fugue Op. 35; Triple concerto for Piano, violin & cello Op. 56)
L. Beethoven	Conductor: Josef Krips; Performers: London Symphony Orchestra	The Nine Symphonies	Padmini Music, 1995	9 (Symphony No. 1–9)
L. Boccherini	Conductors: Pablo Casais & Raymond Leppard; Performers: Severino Gazzelloni, Pepe Romero, & Maurice Gendron	The Best of Boccherini	Philips, 1993	8 (Concerto for flute in D major Op. 27; String Quintet in E major Op. 11–5 G. 275 (Minuet); Quintet for guitar & strings in D major No. 4 G. 448; String Quartet in D major Op. 8–1 G. 165; Cello Concerto in B flat major No. 9 G. 482; Symphony in C major Op. 12–3 G. 505; Quintet for guitar & strings in C major No. 9 G. 453; Symphony in B flat major Op. 12–5 G. 507)
M. Clementi	Howard Shelley	The Complete Piano Sonatas, Vol. 1	Hyperion, 2008	17 (Piano Sonata Op. 1 No. 1–6; Piano Sonata Op. 2 No. 2, 4, 6; Piano Sonata Op. 7 No. 1–3; Piano Sonata Op. 8 No. 1–3; Harpsichord Sonata in G major WO 14; Harpsichord Sonata in A flat major WO 13)
C. P. E. Bach	Conductor: Hartmut Haenchen; Performers: Kammerorchester	Berliner Sinfonien	Brilliant Classics, 2002	5 (Wq. 174, 175, 178, 179, 181)
C. P. E.	Roland Münch,	Orgelkonzerte	Capriccio,	1 (Wq. 34 H. 444)

Composer	Conductor / Performer(s)	Release Title	Label, Release Date	Excerpts
Bach	Hartmut Haenchen, Kammerorchester		1987	
F. J. Haydn	Conductor: Adam Fischer; Performers: Austro-Hungarian Orchestra, Wolfgang Herzer, Gerhard Turetschek, Rainer Kuchl, & Michael Werba	Haydn: Complete Symphonies	Brilliant Classics, 2002	106 (Symphony No. 1–104; Sinfonia A Hob. I–107; Sinfonia B Hob. I–108; Sinfonia Concertante in B flat major Op. 84 Hob. I–105)
W. A. Mozart	Glenn Gould	The Complete Piano Sonatas	Sony Classical, 1994	18 (Piano Sonatas No. 1–18: K. 279–284, K. 309–311, K. 330–333, K. 457, K. 533/494, K. 545, K. 570, K. 576)
A. Salieri, F. Salieri	Paul Badura-Skoda, Pietro Borgonovo, & Clementine Hoogendorn	Antonio Salieri: Concertos, Francesco Salieri: Sinfonia / Scimone, I Solisti Veneti	Erato, 1999	3 (Concerto for Piano in B flat major; Concerto for Flute and Oboe in C major; Sinfonia in B flat major)
F. Schubert	Alfred Brendel	Schubert: Piano Works 1822–1828	Philips, 1989	16 (D. 850, 784, 959, 817, 783, 915, 958, 780, 845, 946, 899, 935, 894, 840, 760, 960)
F. Schubert	Conductor: Herbert Blomstedt; Performers: Dresden Staatskapelle	The Symphonies	Berlin Classics, 2010	9 (Symphony No. 1–6, 6, 9)

B.2 Training Set Excerpts from the Popular Corpus

Artist	Album	Song Title
The Beatles	Please Please Me	I Saw Her Standing There
The Beatles	Please Please Me	Misery
The Beatles	Please Please Me	Anna (Go To Him)
The Beatles	Please Please Me	Boys
The Beatles	Please Please Me	Ask Me Why
The Beatles	Please Please Me	Please Please Me
The Beatles	Please Please Me	Love Me Do
The Beatles	Please Please Me	Baby It's You
The Beatles	Please Please Me	Do You Want To Know A Secret
The Beatles	Please Please Me	A Taste Of Honey
The Beatles	Please Please Me	There's A Place

Artist	Album	Song Title
The Beatles	Please Please Me	Twist And Shout
The Beatles	With the Beatles	It Won't Be Long
The Beatles	With the Beatles	All I've Got To Do
The Beatles	With the Beatles	All My Loving
The Beatles	With the Beatles	Don't Bother Me
The Beatles	With the Beatles	Little Child
The Beatles	With the Beatles	Till There Was You
The Beatles	With the Beatles	Please Mister Postman
The Beatles	With the Beatles	Roll Over Beethoven
The Beatles	With the Beatles	Hold Me Tight
The Beatles	With the Beatles	You Really Got A Hold On Me
The Beatles	With the Beatles	I Wanna Be Your Man
The Beatles	With the Beatles	Devil In Her Heart
The Beatles	With the Beatles	Not A Second Time
The Beatles	With the Beatles	Money
The Beatles	A Hard Day's Night	A Hard Day's Night
The Beatles	A Hard Day's Night	I Should Have Known Better
The Beatles	A Hard Day's Night	If I Fell
The Beatles	A Hard Day's Night	I'm Happy Just To Dance With You
The Beatles	A Hard Day's Night	And I Love Her
The Beatles	A Hard Day's Night	Tell Me Why
The Beatles	A Hard Day's Night	Can't Buy Me Love
The Beatles	A Hard Day's Night	Any Time At All
The Beatles	A Hard Day's Night	Things We Said Today
The Beatles	A Hard Day's Night	You Can't Do That
The Beatles	A Hard Day's Night	I'll Be Back
The Beatles	Beatles for Sale	No Reply
The Beatles	Beatles for Sale	I'm a Loser
The Beatles	Beatles for Sale	Baby's In Black
The Beatles	Beatles for Sale	Rock and Roll Music
The Beatles	Beatles for Sale	I'll Follow the Sun
The Beatles	Beatles for Sale	Mr. Moonlight
The Beatles	Beatles for Sale	Kansas City– Hey, Hey, Hey, Hey
The Beatles	Beatles for Sale	Words of Love
The Beatles	Beatles for Sale	Honey Don't
The Beatles	Beatles for Sale	Every Little Thing
The Beatles	Beatles for Sale	I Don't Want to Spoil the Party
The Beatles	Beatles for Sale	What You're Doing
The Beatles	Beatles for Sale	Everybody's Trying to Be My Baby
The Beatles	Help!	Help!
The Beatles	Help!	The Night Before
The Beatles	Help!	Another Girl

Artist	Album	Song Title
The Beatles	Help!	Ticket To Ride
The Beatles	Help!	Act Naturally
The Beatles	Help!	It's Only Love
The Beatles	Help!	You Like Me Too Much
The Beatles	Help!	Tell Me What You See
The Beatles	Help!	I've Just Seen a Face
The Beatles	Help!	Yesterday
The Beatles	Help!	Dizzy Miss Lizzy
The Beatles	Rubber Soul	Norwegian Wood (This Bird Has Flown)
The Beatles	Rubber Soul	You Won't See Me
The Beatles	Rubber Soul	Nowhere Man
The Beatles	Rubber Soul	Think For Yourself
The Beatles	Rubber Soul	The Word
The Beatles	Rubber Soul	Michelle
The Beatles	Rubber Soul	What Goes On
The Beatles	Rubber Soul	Girl
The Beatles	Rubber Soul	I'm Looking Through You
The Beatles	Rubber Soul	In My Life
The Beatles	Rubber Soul	Wait
The Beatles	Rubber Soul	If I Needed Someone
The Beatles	Rubber Soul	Run For Your Life
The Beatles	Revolver	Taxman
The Beatles	Revolver	Eleanor Rigby
The Beatles	Revolver	I'm Only Sleeping
The Beatles	Revolver	Here, There And Everywhere
The Beatles	Revolver	Yellow Submarine
The Beatles	Revolver	She Said She Said
The Beatles	Revolver	Good Day Sunshine
The Beatles	Revolver	And Your Bird Can Sing
The Beatles	Revolver	For No One
The Beatles	Revolver	Doctor Robert
The Beatles	Revolver	I Want To Tell You
The Beatles	Revolver	Got To Get You Into My Life
The Beatles	Sgt. Pepper's Lonely Hearts Club Band	Sgt. Pepper's Lonely Hearts Club Band
The Beatles	Sgt. Pepper's Lonely Hearts Club Band	With A Little Help From My Friends
The Beatles	Sgt. Pepper's Lonely Hearts Club Band	Lucy In The Sky With Diamonds
The Beatles	Sgt. Pepper's Lonely Hearts Club Band	Getting Better
The Beatles	Sgt. Pepper's Lonely Hearts Club Band	She's Leaving Home
The Beatles	Sgt. Pepper's Lonely	Being For The Benefit Of Mr. Kite!

Artist	Album	Song Title
	Hearts Club Band	
The Beatles	Sgt. Pepper's Lonely Hearts Club Band	When I'm Sixty–Four
The Beatles	Sgt. Pepper's Lonely Hearts Club Band	Sgt. Pepper's Lonely Hearts Club Band (Reprise)
The Beatles	Sgt. Pepper's Lonely Hearts Club Band	A Day In The Life
The Beatles	Magical Mystery Tour	Magical Mystery Tour
The Beatles	Magical Mystery Tour	The Fool On The Hill
The Beatles	Magical Mystery Tour	Flying
The Beatles	Magical Mystery Tour	Blue Jay Way
The Beatles	Magical Mystery Tour	Your Mother Should Know
The Beatles	Magical Mystery Tour	I Am The Walrus
The Beatles	Magical Mystery Tour	Hello Goodbye
The Beatles	Magical Mystery Tour	Penny Lane
The Beatles	Magical Mystery Tour	All You Need Is Love
The Beatles	Abbey Road	Come Together
The Beatles	Abbey Road	Something
The Beatles	Abbey Road	Maxwell's Silver Hammer
The Beatles	Abbey Road	Octopus's Garden
The Beatles	Abbey Road	I Want You
The Beatles	Abbey Road	Here Comes The Sun
The Beatles	Abbey Road	Because
The Beatles	Abbey Road	You Never Give Me Your Money
The Beatles	Abbey Road	Sun King
The Beatles	Abbey Road	She Came In Through The Bathroom Window
The Beatles	Abbey Road	Golden Slumbers
The Beatles	Abbey Road	The End
The Beatles	Let It Be	Two of Us
The Beatles	Let It Be	Dig a Pony
The Beatles	Let It Be	Across the Universe
The Beatles	Let It Be	I Me Mine
The Beatles	Let It Be	Dig It
The Beatles	Let It Be	Maggie Mae
The Beatles	Let It Be	I've Got A Feeling
The Beatles	Let It Be	One After 909
The Beatles	Let It Be	The Long and Winding Road
The Beatles	Let It Be	For You Blue
Carole King	Tapestry	I Feel The Earth Move
Carole King	Tapestry	So Far Away
Carole King	Tapestry	It's Too Late
Carole King	Tapestry	Home Again
Carole King	Tapestry	Beautiful

Artist	Album	Song Title
Carole King	Tapestry	Way Over Yonder
Carole King	Tapestry	You've Got A Friend
Queen	Greatest Hits I	Bohemian Rhapsody
Queen	Greatest Hits I	Fat Bottomed Girls
Queen	Greatest Hits I	Bicycle Race
Queen	Greatest Hits I	You're My Best Friend
Queen	Greatest Hits I	Don't Stop Me Now
Queen	Greatest Hits I	Save Me
Queen	Greatest Hits I	Crazy Little Thing Called Love
Queen	Greatest Hits I	Somebody To Love
Queen	Greatest Hits I	Good Old–Fashioned Lover Boy
Queen	Greatest Hits I	Play The Game
Queen	Greatest Hits I	Seven Seas Of Rhye
Queen	Greatest Hits I	We Are The Champions
Queen	Greatest Hits II	A Kind Of Magic
Queen	Greatest Hits II	I Want It All
Queen	Greatest Hits II	I Want To Break Free
Queen	Greatest Hits II	Who Wants To Live Forever
Queen	Greatest Hits II	Friends Will Be Friends

B.3 Training Set Excerpts from the MIDI Corpus

Collection	Excerpts	Source
"The Well Tempered Clavier", BWV 870–893, J.S. Bach	48	classicalarchives.com
Concertos No. 1–5, L. Beethoven	5	classicalarchives.com
Piano Sonatas No. 1–32, L. Beethoven	32	classicalarchives.com
String Quartets No. 1–16, L. Beethoven	16	classicalarchives.com
Symphonies No. 1–9, L. Beethoven	9	classicalarchives.com
Triple Concerto for Piano, Violin, and Cello, L. Beethoven	1	classicalarchives.com
Variations & Fugue Op. 35, 120, L. Beethoven	2	classicalarchives.com
Symphonies No. 1–13, 20, 31, 36, 45–47, 49, 52, 60, 62–72, 76, 82–104, F. J. Haydn	57	classicalarchives.com
Piano Sonatas No. 1–18, W. A. Mozart	18	classicalarchives.com
Sonatas D760, D780, D784, D817, D840, D845, D850, D894, D899, D915, D935, D958, D959, D960, F. Schubert	14	classicalarchives.com
Symphonies No. 1, 3–6, 8, 9, F. Schubert	7	classicalarchives.com

Appendix C

Test Set Excerpts

Detailed information on the musical excerpts used for the test sets is provided in the following pages. Appendix C.1 lists the excerpts used from the classical corpus, Appendix C.2 lists the excerpts used from the popular corpus, and Appendix C.3 lists the excerpts used from the MIDI corpus.

C.1 Test Set Excerpts from the Classical Corpus

Composer	Conductor / Performer(s)	Release Title	Label, Release Date	Excerpts
F. Chopin	Rafał Blechacz	The Complete Preludes	Deutsche Grammophon, 2007	24 (No. 1–24)

C.2 Test Set Excerpts from the MIDI Corpus

Collection	Excerpts	Source
Preludes No. 1–24, F. Chopin	24	classicalarchives.com

C.3 Test Set Excerpts from the Popular Corpus

Artist	Album	Song Title
Bob Seger	Against the Wind	Against the Wind
Counting Crows	Shrek 2 Soundtrack	Accidentally in Love
Finger Eleven	Them vs. You vs. Me	Paralyzer
Stevie Wonder	In Square Circle	Part Time Lover
John Mayer	Heavier Things	Bigger Than My Body
Eric Clapton	Phenomenon Soundtrack	Change the World
Bryan Adams	11	I Thought I'd Seen Everything
Depeche Mode	Speak and Spell	Just Can't Get Enough
Billy Joel	Piano Man	Piano Man
Bent	Programmed To Love	Private Road
Paul McCartney & Michael Jackson	Pipes of Peace	Say Say Say
Peter Gabriel	Solsbury Hill	Solsbury Hill
Pink Floyd	The Wall	Comfortably Numb
Scissor Sisters	Scissor Sisters	Comfortably Numb
Lenny Kravitz	Baptism	Lady
Led Zeppelin	Led Zeppelin IV	Stairway to Heaven
Rihanna	Good Girl Gone Bad	Umbrella
The Police	Synchronicity	Wrapped Around Your Finger
America	America	A Horse with No Name
LCD Soundsystem	Sound Of Silver	All My Friends
Etta James	The Second Time Around	Seven Day Fool
Blink–182	Enema of the State	What's My Age Again?
Alicia Keys	The Diary of Alicia Keys	If I Ain't Got You
Weezer	Weezer	Say It Ain't So
Beck	Eternal Sunshine of the Spotless Mind Soundtrack	Everybody's Gotta Learn Somtimes
Klute	No Ones Listening Anymore	Silently
Cut Copy	Bright Like Neon	Autobahn Music Box
The Beach Boys	The Beach Boys Today!	Help Me Rhonda
Alanis Morissette	Jagged Little Pill	Ironic
Pearl Jam	Vitalogy	Better Man
Bananarama	Cruel Summer	Cruel Summer
Neil Young	Harvest	Heart of Gold
Green Day	American Idiot	Wake Me Up When September Ends

Artist	Album	Song Title
Madonna	The Immaculate Collection	Cherish
Rihanna	Good Girl Gone Bad	Take a Bow
Semisonic	Reloaded 2	Closing Time
Foreigner	No End In Sight	Waiting for a Girl Like You
Creedence Clearwater Revival	Pendulum	Have You Ever Seen the Rain?
Bloc Party	Silent Alarm	Two More Years
Alicia Keys	As I Am	No One
Boys Noize	Oi Oi Oi	Arcade Robot
Reba McEntire & Kelly Clarkson	Reba: Duets	Because of You
Billy Joel	An Innocent Man	Uptown Girl
Britney Spears	Blackout	Break the Ice
Notwist	Neon Golden	Consequence
Cyndi Lauper	She's So Unusual	Girls Just Want to Have Fun
Frente!	Lonely	Bizarre Love Triangle
Eurythmics	Touch	Here Comes the Rain Again
George Michael	Faith	Faith
José Feliciano	José Feliciano	Feliz Navidad
Fugees	The Score	Killing Me Softly
George Harrison	Cloud Nine	Got My Mind Set On You
Rick Astley	Whenver You Need Somebody	Never Gonna Give You Up
Rockwell	Somebody's Watching Me	Somebody's Watching Me
David Arnold	Casino Royale Soundtrack	The Name's Bond, James Bond
U2	All That You Can't Leave Behind	Stuck in a Moment You Can't Get Out Of

References

Aucouturier, J.-J., and F. Pachet. 2004. Improving timber similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences* 1(1).

Bartsch, M. A., and G. H. Wakefield. 2004. To catch a chorus: Using chroma-based representations for audio thumbnailing. *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. 15–8.

Bello, J. P., and J. Pickens. 2005. A robust mid-level representation for harmonic content in music signals. *Proceedings of the International Conference on Music Information Retrieval*. 304–11.

Brown, J. C. 1991. Calculation of a constant Q spectral transform. *Journal of the Acoustical Society of America* 89(1): 425–34.

Brown, J. C., and M. S. Puckette. 1992. An efficient algorithm for the calculation of a constant Q transform. *Journal of the Acoustical Society of America* 92(5): 1953–7.

Burgoyne, J. A., and L.K. Saul. 2005. Learning harmonic relationships in digital audio with Dirichlet-based hidden Markov models. *Proceedings of the International Conference on Music Information Retrieval*. 11-5.

Cano, P. 1998. Fundamental frequency estimation in the SMS analysis. *Proceedings of the Digital Audio Effects Workshop*.

Cano, P., M. Koppenberger, and N. Wack. 2005a. Content-based music audio recommendation. *Proceedings of International Conference on Multimedia*. 211–2.

Cano, P., M. Koppenberger, and N. Wack. 2005b. An industrial strength content-based music recommendation system. *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2.

Casey, M. A., R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. 2008. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE* 96(4): 668–96.

Catteau, B., J.-P. Martens, and M. Leman. 2007. A probabilistic framework for audio-based tonal key and chord recognition. In *Advances in Data Analysis*, edited by D. Reinhold, and H.-J. Lenz, 637–44. Heidelberg, Germany: Springer Berlin.

Cemgil, A. T. 2004. *Bayesian music transcription*. Ph.D. Dissertation. Radboud University Nijmegen.

Chai, W. 2005. *Automated analysis of musical structure*. Ph.D. Dissertation. Massachusetts Institute of Technology.

Chai, W., and B. Vercoe. 2005. Detection of key change in classical piano music. *Proceedings of the International Conference on Music Information Retrieval*. 468–73.

Chew, E. 2000. *Towards a mathematical model of tonality*. Ph.D. Dissertation. Massachusetts Institute of Technology.

Chew, E. 2001. Modeling tonality: Applications to music cognition. *Proceedings of the Meeting of the Cognitive Science Society*. 206–11.

Chew, E. 2002. The spiral array: An algorithm for determining key boundaries. In *Music and Artificial Intelligence*, edited by C. Anagnostopoulou, M. Ferrand, and A. Smaill, 18–31. Heidelberg, Germany: Springer Berlin.

Chew, E. 2007. Out of the grid and into the spiral: Geometric interpretations of and comparisons with the spiral-array model. *Computing in Musicology (Tonal Theory for the Digital Age)* 15: 51–72.

Chew, E., and Y.-C. Chen. 2005. Real-time pitch spelling using the spiral array. *Computer Music Journal* 29(2): 61–76.

Chuan, C. H. 2008. *Hybrid methods for music analysis and synthesis: Audio key finding and automatic style-specific accompaniment*. Ph.D. Dissertation. University of Southern California.

Chuan, C. H., and E. Chew. 2005a. Polyphonic audio key-finding using the spiral array CEG algorithm. *Proceedings of IEEE International Conference on Multimedia and Expo*. 21–4.

Chuan, C. H., and E. Chew. 2005b. Audio key finding using FACEG: Fuzzy analysis with the CEG algorithm. In *Abstract of the Annual Music Information Retrieval Evaluation eXchange*.

Chuan, C. H., and E. Chew. 2005c. Fuzzy analysis in pitch class determination for polyphonic audio key finding. *Proceedings of the International Conference on Music Information Retrieval*. 296–303.

Chuan, C. H., and E. Chew. 2007. Audio key-finding: Considerations in system design and case studies on Chopin's 24 Preludes. *EURASIP Journal on Advances in Signal Processing* 2007(1).

Cohen, A. J. 2000. Development of tonality induction: Plasticity, exposure, and training. *Music Perception* 17(4): 437–59.

Cohn, R. Introduction to Neo-Riemannian theory: A survey and a historical perspective. *The Journal of Music Theory* 42(2): 167–80.

Duan, Z., L. Lu, and C. Zhang. 2008. Audio tonality mode classification without tonic annotations. *Proceedings of the International Conference on Multimedia and Expo* (ICME). 1361–4.

Duxbury, C., M. Davies, and M. Sandler. 2001. Separation of transient information in musical audio using multiresolution analysis techniques. *Proceedings of the COST G-6 Conference on Digital Audio Effects*. 1–4.

Fujinaga, I., S. Moore, and D. S. Sullivan. 1998. Implementation of exemplar-based learning model for music cognition. *Proceedings of the International Conference on Music Perception and Cognition*. 171–9.

Fujishima, T. 1999. Realtime chord recognition of musical sound: A system using common lisp music. *Proceedings of the International Computer Music Conference (ICMC)*. 464–7.

Gatzsche, G., M. Mehnert, D. Arndt, and K. Brandenburg. 2008. Circular pitch space based musical tonality analysis. *Proceedings of the Audio Engineering Society Convention*.

Gómez E. 2005. Key estimation from polyphonic audio. In *Abstract of the Annual Music Information Retrieval Evaluation eXchange*.

Gómez E. 2006a. *Tonal description of music audio signals*. Ph.D. Dissertation. Universitat Pompeu Fabra.

Gómez E. 2006b. Tonal description of polyphonic audio for music content processing. *INFORMS Journal on Computing* 18(3): 294–304.

Gómez E., and P. Herrera. 2004a. Estimating the tonality of polyphonic audio files: Cognitive versus machine learning modelling strategies. *Proceedings of the International Conference on Music Information Retrieval*. 92–5.

Gómez E., and P. Herrera. 2004b. Automatic extraction of tonal metadata from polyphonic audio recordings. *Proceedings of the Audio Engineering Society International Conference*.

Goto, M. 2001. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research* 30(2): 159-71.

Goto, M., and Y. Muraoka. 1999. Real-time beat tracking for drumless audio signals: chord change detection for musical decisions. *Speech Communication* 27(3-4): 331-5.

Harte, C. A., and M. B. Sandler. 2005. Automatic chord identification using a quantised chromagram. *Proceedings of the Audio Engineering Society Convention*.

Harte, C. A., M. B. Sandler, and M. Gasser. 2006. Detecting harmonic change in musical audio. *Proceedings of Audio and Music Computing for Multimedia Workshop*. 21–6.

Hermes, D. J. 1988. Measurement of pitch by subharmonic summation. *Journal of Acoustical Society of America* 83(1): 257–64.

Hoashi, K., S. Hamawaki, H. Ishizaki, Y. Takishima, and J. Katto. Usability evaluation of visualization interfaces for content-based music retrieval systems. *Proceedings of the International Conference on Music Information Retrieval*. 207–12.

Holtzman, S. R. 1977. A program for key determination. *Journal of New Music Research* 6: 29–56.

Hu, D. J., and L. K. Saul. 2009. A probabilistic topic model for unsupervised learning of musical key-profiles. *Proceedings of the International Conference on Music Information Retrieval*. 441–6.

Huron, D., and R. Parncutt. 1993. An improved model of tonality perception incorporating pitch salience and echoic memory. *Psychomusicology* 12: 154–71.

Hyer, B. 2001. Key (i). In *The New Grove Dictionary of Music and Musicians*, edited by S. Sadie and J. Tyrrell. London: Macmillan.

Hyer, B. 2002. Tonality. In *The Cambridge History of Western Music Theory*, edited by T. S. Christensen, 726–52. Cambridge: Cambridge University Press.

Inoshita, T., and J. Katto. 2009. Key estimation using circle of fifths. In *Advances in Multimedia Modeling*, edited by B. Huet, A. Smeaton, K. Mayer-Patel, and Y. Avrithis, 287–97. Heidelberg, Germany: Springer Berlin.

Izmirli, Ö. 2005a. Template based key-finding from audio. *Proceedings of the International Computer Music Conference*. 211–4.

Izmirli, Ö. 2005b. An algorithm for audio key finding. In *Abstract of the Annual Music Information Retrieval Evaluation eXchange*.

Izmirli, Ö. 2006. Audio key finding using low-dimensional spaces. *Proceedings of the International Conference on Music Information Retrieval*. 127–32.

Izmirli, Ö. 2007. Localized key finding from audio using non-negative matrix factorization for segmentation. *Proceedings of the International Conference on Music Information Retrieval*. 195–200.

Izmirli, Ö. 2009. Estimating the tonalness of transpositional type pitch-class sets using learned tonal key spaces. In *Communications in Computer and Information Science*, edited by E. Chew, A. Childs, and C.-H. Chuan, 146-153. Berlin, Germany: Springer Berlin Heidelberg.

Izmirli, Ö, and S. Bilgen. 1994. Recognition of musical tonality from sound input. *Proceedings of the Mediterranean Electrotechnical Conference*. 269–71.

Izmirli, Ö, and S. Bilgen. 1996. A model for tonal context time course calculation from acoustical input. *Journal of New Music Research* 25(3): 276–88.

Jensen, J. H. 2009. *Feature extraction for music information retrieval*. Ph.D. Dissertation. Aalborg University.

Kennedy, M., and J. Bourne. 2006. *Oxford Dictionary of Music*. New York: Oxford University Press.

Kostek, B. 2005. *Perception-Based Data Processing in Acoustics*. Berlin, Germany: Springer Berlin Heidelberg.

Krumhansl, C. L. 1990. *Cognitive Foundations of Musical Pitch*. New York: Oxford University Press.

Krumhansl, C. L., and E. J. Kessler. 1982. Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys. *Psychological Review* 89(4): 334–68.

Krumhansl, C. L., and R. N. Shepard. 1979. Quantification of the hierarchy of tonal functions within a diatonic context. *Journal of Experimental Psychology: Human Perception and Performance* 5(4): 579–94.

Lee, K. 2006. Automatic chord recognition using enhanced pitch class profile. *Proceedings of the International Computer Music Conference*. 306–13.

Lee, K. 2008a. *A system for acoustic chord transcription and key extraction from audio using hidden Markov models trained on synthesized audio*. Ph.D. Dissertation. Stanford University.

Lee, K. 2008b. 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.

Lee, K., and M. Slaney. 2007. A unified system for chord transcription and key extraction using hidden Markov models. *Proceedings of the International Conference on Music Information Retrieval*. 245–50.

Leman, M. 1989. Symbolic and subsymbolic information processing in models of musical communication and cognition. *Journal of New Music Research* 18(1): 141–60.

Leman, M. 1991. *Een model van toonsemantiek: naar een theorie en discipline van de muzikale verbeelding*. Ph.D. Dissertation. University of Ghent.

Leman, M. 1994. Schema-based tone center recognition of musical signals. *Journal of New Music Research* 23(2): 169–204.

Leman, M. 1995a. A model of retroactive tone center perception. *Music Perception* 12(4): 439–71.

Leman, M. 1995b. *Music and Schema Theory: Cognitive Foundations of Systematic Musicology*. Berlin, Germany: Springer Berlin Heidelberg.

Lerdahl, F. 2001. *Tonal Pitch Space*. New York: Oxford University Press.

Longuet-Higgins, H. C., and M. J. Steedman. 1971. On interpreting Bach. In *Machine Intelligence*, edited by B. Meltzer and D. Michie, 6, 221–41. Edinburgh: University of Edinburgh Press.

Lu, L., D. Liu, and H. J. Zhang. 2006. Automatic mood detection and tracking of music signals. *IEEE Transactions on Audio, Speech, and Language Processing* 14(1): 5-18.

Madsen, S. T., and G. Widmer. 2007. Key-finding with interval profiles. *Proceedings of the International Computer Music Conference*. 212–5.

Mardirossian, A., and E. Chew. 2005. SKEFIS: A symbolic (MIDI) key finding system. In *Abstract of the Annual Music Information Retrieval Evaluation eXchange*.

Mardirossian, A., and E. Chew. 2006. Music summarization via key distributions: Analyses of similarity assessment across variations. *Proceedings of the International Conference on Music Information Retrieval*. 234-239

Marolt, M. 2006. A mid-level melody-based representation for calculating audio similarity. *Proceedings of the International Conference on Music Information Retrieval*. 280-285

Martens, G., H. D. Meyer, B. D. Baets, M. Leman, M. Lesaffre, J.-P. Martens, and T. D. Mulder. 2004. Distance-based versus tree-based key recognition in musical audio. *Soft Computing: A Fusion of Foundations, Methodologies and Applications* 9(8): 565–74.

Mauch, M., C. Cannam, M. Davies, S. Dixon, C. Harte, S. Kolozali, D. Tidhar, and M. Sandler. 2009. OMRAS2 metadata project 2009. *Late-breaking Session of the International Conference on Music Information Retrieval*.

McEnnis, D., C. McKay, I. Fujinaga, and P. Depalle. 2005. jAudio: A feature extraction library. *Proceedings of the International Conference on Music Information Retrieval*. 600–3.

McKay, C., R. Fiebrink, D. McEnnis, B. Li, and I. Fujinaga. 2005. ACE: A framework for optimizing music classification. *Proceedings of the International Conference on Music Information Retrieval*. 42–9.

McKay, C., and I. Fujinaga. 2009. JMIR: Tools for automatic music classification. *Proceedings of the International Computer Music Conference*. 65-8.

Mehnert, M., G. Gatzsche, K. Gatzsche, and K. Brandenburg. 2007. The analysis of tonal symmetries in musical audio signals. *Proceedings of the International Symposium on Musical Acoustics*.

Mitrović, D., M. Zeppelzauer, and C. Breiteneder. 2010. Features for content-based audio retrieval. *Advances in Computers* 78: 71–150.

Müller, M., F. Kurth, and M. Clausen. 2005. Chroma-based statistical audio features for audio matching. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. 275–8.

Noland, K., and M. Sandler. 2006. Key estimation using a hidden Markov model. *Proceedings of the International Conference on Music Information Retrieval*. 121-6

Noland, K., and M. Sandler. 2007. Signal processing parameters for tonality estimation. *Proceedings of the Audio Engineering Society Convention*.

Noland, K., and M. Sandler. 2009. Influences of signal processing, tone profiles, and chord progressions on a model for estimating the musical key from audio. *Computer Music Journal* 33(1): 42–56.

Ong, B. S., E. Gómez, and S. Streich. 2006. Automatic extraction of musical structure using pitch class distribution features. *Proceedings of the Workshop on Learning the Semantics of Audio Signals*. 53–65.

Paivo, R. 2007. *Melody detection in polyphonic audio*. Ph. D. Dissertation. University of Coimbra.

Papadopoulos, H., and G. Peeters. 2009. Simultaneous estimation of chord progression and downbeats from an audio file. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. 121–4.

Papadopoulos, H., and G. Peeters. 2009. Local key estimation based on harmonic and metric structures. *Proceedings of the International Conference on Digital Audio Effects*.

van de Par, S., M. McKinney, and A. Redert. 2006. Musical key extraction from audio using profile training. *Proceedings of the International Conference on Music Information Retrieval*. 328-329

Patterson, R. D. 1986. Spiral detection of periodicity and the spiral form of musical scales. *Psychology of Music* 14(1): 44–61.

Pauws, S. 2004. Musical key extraction from audio. *Proceedings of the International Conference on Music Information Retrieval*. 96–9.

Pauws, S. 2005. KEYEX: Audio key extraction. In *Abstract of the Annual Music Information Retrieval Evaluation eXchange*.

Pauws, S. 2006. Extracting the key from music. In *Intelligent Algorithms in Ambient and Biomedical Computing*, edited by W. Verhaegh, E. Aarts, and J. Korst, 119–32. Dordrecht, Netherlands: Springer.

Peeters, G. 2006a. Chroma-based estimation of musical key from audio-signal analysis. *Proceedings of the International Conference on Music Information Retrieval*. 115-120

Peeters, G. 2006b. Musical key estimation of audio signal based on hidden markov modeling of chroma vectors. *Proceedings of International Conference on Digital Audio Effects*. 127–31.

Peeters, G. 2007. Template-based estimation of time-varying tempo. *EURASIP Journal on Advances in Signal Processing*. 2007(1): 158–71.

Pickens, J., and T. Crawford. 2002. Harmonic models for polyphonic music retrieval. *Proceedings of the International Conference on Information and Knowledge Management*. 430–7.

Povel, D.-J. 1996. Exploring the elementary harmonic forces in the tonal system. *Psychological Research* 58(4): 274–83.

Purwins, H. 2005. *Profiles of pitch classes circularity of relative pitch and key – Experiments, models, computational music analysis, and perspectives*. Ph.D. Dissertation. Technischen Universität Berlin.

Purwins, H., and B. Blankertz, 2005. CQ-Profiles for key finding in audio. In *Abstract of the Annual Music Information Retrieval Evaluation eXchange*.

Purwins, H., B. Blankertz, G. Dornhege, and K. Obermayer. 2004. Scale degree profiles from audio investigated with machine learning. *Proceedings of the Audio Engineering Society Convention*.

Purwins, H., B. Blankertz, and K. Obermayer. 2000. A new method for tracking modulations in tonal music in audio data format. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*. 270–5.

Raphael, C., and J. Stoddard. 2003. Harmonic analysis with probabilistic graphical models. *Proceedings of the International Conference on Music Information Retrieval*. 177-81.

Raphael, C., and J. Stoddard. 2004. Functional harmonic analysis using probabilistic models. *Computer Music Journal* 28(3): 45-52.

Ratner, L. 1962. *Harmony: Structure and Style*. New York: McGraw-Hill.

Russel, S. and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach, Second Edition*. New Jersey: Pearson Education, Inc.

Sandler, M. 2007. Interacting with digital music. *Journal of New Music Research* 36(3): 227–39.

Sapp, C. S. 2001. Harmonic visualizations of tonal music. *Proceedings of the International Computer Music Conference*. 423–30.

Schmuckler, M. A., and R. Tomovski. Perceptual tests of an algorithm for musical key-finding. *Journal of Experimental Psychology* 31(5): 1124–49.

Sheh, A., and D. Ellis. 2003. Chord segmentation and recognition using EM-trained hidden Markov models. *Proceedings of the International Conference on Music Information Retrieval*.

Shenoy, A., R. Mohapatra, and Y. Wang. 2004. Key determination of acoustic music signals. *Proceedings of the International Conference on Multimedia and Expo*. 1771–4.

Shepard, R. N. 1964. Circularity in judgments of relative pitch. *The Journal of the Acoustical Society of America* 36(12): 2346–53.

Shmulevich, I., and O. Yli-Harja. 2000. Localized key-finding: Algorithms and applications. *Music Perception* 17(4): 531–44.

Temperley, D. 1999. What's key for key? The Krumhansl-Schmuckler key finding algorithm reconsidered. *Music Perception* 17(1): 65–100.

Temperley, D. 2001. *The Cognition of Basic Musical Structures*. Cambridge, MA: MIT Press.

Temperley, D. 2007. *Music and Probability*. Cambridge, MA: MIT Press.

Temperley, D., and E. W. Marvin. 2008. Pitch-class distribution and the identification of key. *Music Perception* 25(3): 193–212.

Tenkanen, A. 2009. Evaluating tonal distances between pitch-class sets and predicting their tonal centres by computational models. In *Communications in Computer and Information Science*, edited by E. Chew, A. Childs, and C.-H. Chuan. Berlin, Germany: Springer Berlin Heidelberg.

Varewyck, M., J. Pauwels, and J.-P. Martens. 2008. A novel chroma representation of polyphonic music based on multiple pitch tracking techniques. *Proceedings of the International Conference on Multimedia*. 667–70.

Vos, P. G. 2000. Tonality induction: theoretical problems and dilemmas. *Music Perception, Special Issue in Tonality Induction* 17(4): 402–16.

Winograd, T. 1968. Linguistics and the computer analysis of tonal harmony. *Journal of Music Theory* 12(1): 2–49.

Yoshino, I., and J.-I. Abe. 2004. Cognitive modeling of key interpretation in melody perception. *Japanese Psychological Research* 46(4): 283–97.

Zenz, V. 2007. Automatic chord detection incorporating beat and key detection. *Proceedings of the IEEE International Conference on Signal Processing and Communications*. 1175–8.

Zhu, Y. 2005. An audio key finding algorithm. In *Abstract of the Annual Music Information Retrieval Evaluation eXchange*.

Zhu, Y., and M. S. Kankanhalli. 2004. Key-based melody segmentation for popular songs. *Proceedings of the International Conference on Pattern Recognition*. 862–5.

Zhu, Y., and M. S. Kankanhalli. 2006. Precise pitch profile feature extraction form musical audio for key detection. *IEEE Transactions on Multimedia* 8(3): 575–84.

Zhu, Y., M. S. Kankanhalli, and S. Gao. 2005. Music key detection for musical audio. *Proceedings of the International Multimedia Modeling Conference*. 30–7.