NORTHWESTERN UNIVERSITY

Computational Music Audio Scene Analysis

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

Zhiyao Duan

EVANSTON, ILLINOIS

August 2013

ABSTRACT

Computational Music Audio Scene Analysis

Zhiyao Duan

When playing or listening to a piece of music audio, a musician may constantly but unconsciously analyze its musically meaningful objects such as pitches and notes, and organize them according to instruments or sources into melody, bass, etc. This capability, although looks natural to musicians, is extremely difficult for machines. In my dissertation, I call this capability *Music Audio Scene Analysis (MASA)*. The goal of my research is to design computational models for MASA. More specifically, I want to answer two questions: 1) What note is played by each instrument at a given time? 2) How does each note sound? I focus on polyphonic music audio composed of harmonic sound sources.

Computational models for MASA can help people interact with music. Scholars may wish to analyze the harmonies in a recording. Musicians may be interested in correcting occasional wrong notes and beautifying their performances. Listeners may wish to amplify an instrument in order to help them follow its melodic line.

According to the availability of different modalities of the music, MASA may be approached in different scenarios. The first scenario is to perform MASA purely from audio.

I propose a system to perform multi-pitch analysis on polyphonic audio. This system first estimates pitches of all the sources in each time frame, then streams the pitch estimates across time into pitch streams. Each pitch stream corresponds to a source. Given these pitch streams, source signals can be separated with simple masking techniques. The proposed system is general enough that it can deal with polyphonic audio composed of harmonic sources other than musical instruments, such as multi-talker speech.

The second scenario is when external information such as musical score, video, lyrics, meta data is available. I focus on leveraging the musical score information for MASA. Digital musical scores (e.g. MIDI files) are widely available for many kinds of music, and are directly machine-understandable. I propose a system that first aligns the audio with the score, then uses the score-provided pitch information to separate the sources of the audio mixture. The system performs in an online fashion. Compared to solely analyzing audio, the score information does significantly improve the MASA performance, to a level that is promising for further applications.

Acknowledgements

First and foremost, I would like to thank my advisor, Prof. Bryan Pardo, for guiding, teaching, and supporting me during the past five years. Bryan's advising is comprehensive and reaches every single aspect of my academic growth. From rigorous mathematical discussions to high-level strategic thinking, to academic writing and presentation, to planning and time management, to career development and work-life balance, his advice is and will continue to be a valuable asset for me. I especially appreciate the balance between freedom and guidance that he gives to me regarding scientific exploration, and the happy atmosphere that he creates in the lab. Bryan is just a great advisor!

I owe an immerse amount of gratitude to Prof. DeLiang Wang. As a successful Chinese researcher, he sets an excellent example for me to follow. His rigor, persistence and tolerance has taught me the qualities that a good researcher must possess. I also thank him for offering me a two-months visit at his lab in Winter 2013, where I spent wonderful time and made several good friends.

Special thanks go to my dissertation readers, Prof. Thrasos Pappas and Prof. Michael Honig for serving on my dissertation committee and giving me valuable feedback since my dissertation prospectus. I also appreciate their advice on my academic job interview. That turned out to be very helpful. I thank Prof. Lance Fortnow and Prof. Jorge Nocedal for serving on the committee of my PhD qualifying exam.

I would like to thank Dr. Gautham J. Mysore and Prof. Paris Smaragdis, who were my mentors during my internship at Adobe Systems in 2011. They are excellent researchers and collaborators. I enjoyed working, chatting and drinking with them.

I would like to thank my former and present labmates, who made our Interactive Audio Lab a unique and fantastic place to work at. Special honors go to Jinyu Han, Zafar Rafii, Mark Cartwright, David Little, Arefin Huq, Jesse Bowman, and Prem Seetharaman, with whom I have had particularly fruitful discussions.

I would like to thank Dr. Eugene Kushnirsky and Prof. Michael Stein in the Mathematics Department. Taking their pure math courses is one of the most enjoyable activities I had at Northwestern.

I would like to thank my friends He Zhang, Xiang Huang, Wei Zhou, Ying Chen, Yang Xu, Jingqi Wang, and Betty Phillips, who made my life at Northwestern enjoying. Thanks also go to Xiaojia Zhao, Kun Han, Shuang Li, and Yuxuan Wang, who made my visit to the Ohio State University pleasant, and to Juhan Nam, Nick Bryan, and Brian King, who were my fellow interns at Adobe.

Finally, I thank my parents and my wife for their endless love and encouragement. They are wonderful!

To my parents Xianjun Duan and Lina Wang, and my wife Yunping Shao.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

You are listening to a violin concerto. In spite of the complex accompaniment played by the orchestra, you might be able to follow and memorize the melody of the violin. If you are also presented with the score of the music and you know how to read a score, you might be able to follow all the instrument lines. The capability of analyzing the music audio into musically meaningful objects such as pitches and notes, and organizing them according to instruments or sources into melody, bass, etc., is called *Music Audio Scene Analysis (MASA)* in this dissertation. Although it looks natural to well-trained musicians, it is extremely difficult for machines. The goal of my dissertation research is to design computational models for MASA.

1.1. Motivation

There are plenty of motivations to designing computational models for MASA. My favorite ones are about novel interactions with music. Here I describe two of them.

1.1.1. Automatic Music Accompaniment

Imagine you want to play a string quartet with your musician friends but the cello player is absent. It would be great if a computer could temporally replace your cello friend and play with you. This is the automatic music accompaniment problem. Different from

conventional accompaniment systems such as Karaoke and Music-Minus-One[1] where the human performer follows the machine, in an automatic music accompaniment scenario the machine follows the human performance and adjusts itself accordingly.

An automatic music accompaniment system clearly requires MASA techniques. It needs to analyze the human audio performance into musical objects and map them to the score. It also needs to render the accompaniment part of the score into audio at the right time. The rendering could be done using a synthesizer. But to make the accompaniment more realistic, the system often plays back a pre-recorded human audio performance of the accompaniment part. In this scenario, the system also needs to analyze the pre-recorded accompaniment audio and align it to the score, and to the human performance.

Existing automatic music accompaniment systems only follow solo audio performances [**101, 114**]. This is because they can only analyze the musical objects in monophonic music pieces robustly. They treat polyphonic performances as a whole and cannot analyze them into different sources. Due to the same reason, the pre-recorded accompaniment audio (usually polyphonic) can only be processed as a whole as well. The system cannot adaptively change the volume of some instruments, unless the accompaniment audio of different instruments is pre-recorded separately. Suppose next time your absent friend is the viola player instead of the cello player. Current systems would require another pre-recorded accompaniment of the viola, instead of using a single commercial recording of the whole quartet and adaptively playing the absent instrument(s) each time.

With more advanced MASA techniques that can deal with polyphonic music audio, one can design a new generation of automatic music accompaniment systems that are more

[1] `http://musicminusone.com/`

intelligent and versatile. These systems would be able to follow both solo or polyphonic human performances. They would simply store a single professional music recording of the quartet in the accompaniment system, and automatically mute the sources that are being performed by human. They could even alter the instrumentation and timbre of the accompaniment recording, according to human performers' instrumentation and timbre. All these novel interactions are more natural and universal than existing interactions such as Karaoke, Music-Minus-One and current automatic accompaniment systems, because they cannot analyze musical objects in polyphonic music audio.

1.1.2. Advanced Music Editing

Imagine you recorded your sister's string quartet performance with a stereo microphone. Everything was perfect but your sister's viola part was too soft. You really want to make her part louder while keeping the volume of the other parts unchanged. What can you do? Well, with existing music editing techniques, you can adjust the volume, equalization, reverberation, etc., but only on the audio mixture. You cannot just access the viola signal and increase its volume without impacting the other parts. To do so, you would need MASA to separate the mixture into different sources.

There are in fact many scenarios that require advanced music editing techniques. For example, scholars may wish to analyze the harmonies in a recording. Musicians may be interested in correcting occasional wrong notes and beautifying their performances. Audio engineers might want to separate the exciting bass drum from a popular song and remix it into another song, or make the clarinet sound somewhat like an oboe to achieve a special effect. All these musically meaningful edits require direct access to the sound

sources in the audio mixture, and cannot be fulfilled by current editing techniques unless the source signals were recorded and stored separately.

Computational models for MASA will enable these novel edits. With MASA techniques, we will be able to access and separate the sound sources from the polyphonic audio mixture, and further manipulate the source signals. In addition, MASA provides basic understanding of the musical objects such as notes and chords. Therefore, we can access and manipulate each note/chord of each source. We can modify their amplitude, pitch, time and timbre individually, and copy and paste to duplicate them. These novel editing techniques will greatly advance our interaction with music recordings.

1.2. Problem Statement

1.2.1. Computational Goals of Music Audio Scene Analysis

The problem that I am concerned with in this dissertation is music audio scene analysis (MASA). More specifically, I define its computational goals as the recognition and segregation of musically meaningful objects such as notes, chords, melodic lines and sources from music audio. Taking a wind quintet (played by flute, clarinet, oboe, horn and bassoon) as an example, the questions that I am interested in are:

(1) What note is played by each instrument at a given time?

(2) How does each note sound?

The first question, at the symbolic level, is about recognizing musical objects and transcribing them into a symbolic representation. The second question, at the physical level, is about associating physical signals to these musical objects and segregating them from the audio.

MASA is somewhat analogous to scene understanding in computer vision, which aims to recognize objects and understand their spatial relations from pictures. However, scene understanding only gives a symbolic representation of the objects and their relations. A full analogy would require an accurate image segmentation for the objects as well. MASA is also analogous to speech recognition, which aims to transcribe speech into text. However, this also only gives a symbolic representation of speech. A full analogy would be speech recognition and separation in multi-talker speech.

A music audio scene can be as simple as a nursery rhyme, or be as complex as a symphony. In what scenarios are the computational goals proposed above non-trivial but still achievable? This is an important question to bear in mind. I try to use human performance as a frame of reference.

When no information other than the music audio is available, recognizing musical objects (the first goal) is very similar to the music dictation (transcription) problem, i.e. writing down the musical score by listening to the music audio. This is one of the most important aspects of musical training [**51**], but can be very challenging when there are several simultaneous voices. There is a story about the talented musician Wolfgang Mozart that when he was fourteen-year old, he was able to transcribe Gregorio Allegri's Miserere (a piece written for nine voices) entirely with minor errors, after listening to it in the Sistine Chapel only once [**1**]. This story shows his powerful music memory and incredible music dictation ability. Although some researchers believe that the story contains some exaggerations and Mozart might have seen a version of the Miserere score before [**15**], we can view this story as providing an upper bound of what human musicians can achieve in music dictation.

On the other hand, music dictation at a certain level is a required ability for all musicians. This ability can be acquired after years of musical training [**51**]. Although the particular training procedure may vary in different institutions, it always starts from transcribing simple examples such as a single melody, isolated pitch intervals and chords, and gradually considers more complex materials. The general expectation for college music majors after four or five semesters' training in music theory is that "they can transcribe an excerpt of a quartet (e.g. four measures) with quite complex harmonies, after listening to it four or five times", according to David Temperley, an associate professor of music theory in the Eastman School of Music. These music education practices provide supports to the proposed computational goals. At least we know there exist some MASA systems, i.e. human musicians, that can achieve the recognition goal.

When the musical score is provided, human musicians can recognize even more complex music audio. For example, a piano student can match a polyphonic piano recording with its score. A conductor can follow dozens of instrument lines of a symphony and spot an error of a single note.

However, computer systems need not to follow what human musicians do to achieve the MASA goals. Tasks that are trivial for human musicians might be very challenging for computers, and vice versa. For example, Hainsworth conducted a study to ask nineteen musicians to describe how they transcribe music [**57**]. These musicians had experience in transcribing music with differen genres for different goals. It is found that their procedures are quite consistent with each other, all involving several passes of the music from high level to the details. They first write down a rough sketch of the music including the structure and pitch contours, then transcribe chords and baselines, followed

by the melody, and finally fill in the inner melodies and other details. While the iterative strategies may be due to the limited music memory of human musicians, for computers memory is not an issue and these strategies need not be followed. In addition, none of the musicians mentioned the key subtasks such as beat tracking, style detection and instrument identification, because they felt they are trivial tasks. However, these tasks are extremely challenging for computers. Take instrument identification in polyphonic music as an example, there are few computer algorithms that attempt to address the problem and their performance is far from comparable with humans [**46**].

Besides, the second goal, segregating the signals of musical objects, is even beyond any human being's ability. Musicians can perceive a sound object such as a note and reconstruct it in their heads, but no one can actually reconstruct the physical signal by hand. Computers, however, can do a fairly good job with very simple algorithms, given an accurate recognition of the note. One can design more advanced algorithms to obtain more accurate segregation.

To sum up, the proposal of the computational goals is partially inspired by the capability of human musicians. I want to design computer systems that can achieve comparable MASA performance to average college music majors. However, the design of the algorithms needs not emulate human methods. Engineering innovations should be encouraged to pursue an alternative way to achieve MASA and to go beyond human musicians.

1.2.2. Dissertation Scope: Music Audio Composed of Harmonic Sound Sources

Due to the large variety of sound sources in music and their rich timbre, the general MASA problem is clearly too big for a single dissertation. Therefore, I limit the scope

of my dissertation to Western music composed of harmonic sound sources. Examples of harmonic sound sources include wind, string and brass instruments, vocals, etc. They form the tonal basis of music.

Harmonic sounds are a special kind of periodic sounds whose dominant frequency components are approximately equally spaced. In other words, the significant spectral peaks are located at the harmonic positions (i.e. integer multiples) of the fundamental frequency (F0), which is often aligned with the first significant spectral peak. Pitch, a subjective concept of periodic sounds, is consistently perceived as the F0 of harmonic sounds. Therefore, I do not discriminate F0 and pitch, but view them as the same thing. More details about the formal definitions of harmonic sounds, pitch, and F0 can be found in the appendices.

Pitch, or F0, is a very important feature for MASA of harmonic sound sources. Dominant frequency components (i.e. harmonics) of a harmonic sound are all located at the integer multiples of the pitch. If the pitch is known, the organization of the dominant frequency components can be obtained. This is very helpful for estimating the spectrum of a harmonic sound source if it is mixed with other sources. Once the pitches of the harmonic source at different time frames are estimated, the evolution of the dominant frequency components of this source can be inferred. This will make the separation of this source from the mixture much easier. Therefore, estimating the pitches of each harmonic source at each individual time frame is an effective way to achieve MASA. In my dissertation, I adopt this method.

1.2.3. MASA Solely from Audio

When the music audio signal is the only input that a MASA system can access, it needs to analyze the music audio scene purely from the audio. The first part of my dissertation addresses this scenario. Figure 1.1 shows the general process.

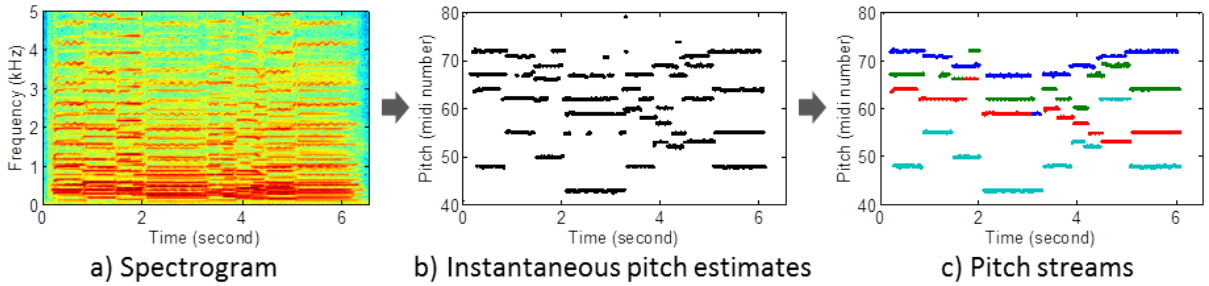


Figure 1.1. Illustration of the multi-pitch estimation and streaming process. Multi-pitch estimation first estimates concurrent pitches at each time frame of the audio spectrogram. Multi-pitch streaming then streams pitch estimates across frames according to sources. Pitch streams of different sources are shown in different colors.

The first step is to estimate the pitches from the audio. For monophonic music (e.g. a trumpet solo) where there is at most one pitch at a time, pitch estimation (or detection) is a relatively solved problem. There are a number of methods [**24, 7, 117**] that can achieve very good estimation accuracies. For polyphonic music (e.g. a string quartet) where there can be multiple pitches at a time, however, pitch estimation remains a very challenging problem. Pitch estimation from polyphonic audio is called multi-pitch estimation (MPE). It requires estimating both pitches and the number of pitches (polyphony). The difficulties of MPE mainly arise from the huge number of possible combinations of pitches and their harmonics, the rich timbre of different sources, and the issue of overlapping harmonics. In Chapter 2, I propose a novel method for MPE in each single time frame. The basic idea

is to view the problem as a maximum likelihood parameter estimation problem, where the parameters are the pitches and the observation is the magnitude spectrum of the audio frame. The parameters of the likelihood model are trained from musical chords with ground-truth pitch labels.

MPE is a first step towards analyzing music audio scenes of harmonic sound sources, but it is not enough. Although we have estimated the number and frequency values of simultaneous pitches in each frame, we still have no idea about how the pitches (and their corresponding harmonics) are organized according to sources. Without this information, we will not be able to transcribe the notes of the music or separate the sources from the audio mixture.

Multi-pitch streaming (MPS) is the key to address this issue. The idea is to group all pitch estimates from a single source into a single stream, corresponding to that source. For a piece of music audio played by M monophonic harmonic sound sources, we want to get M pitch streams, each of which corresponds to a harmonic source. The streams can be discontinuous, due to frequency jumps caused by note transitions and non-pitched frames caused by silence. In Chapter 3 I propose the first approach to perform MPS in an unsupervised way, that is, no prior training of the sources in the audio mixture is required. The basic idea is to cluster the pitches according to their timbre similarity and time-frequency proximity. Pitches with similar timbre and proximate in both time and frequency are likely from the same source. Furthermore, it can take the pitch estimates of any MPE method as input.

From the pitch trajectories estimated by MPS, a preliminary separation of the harmonic sources can be achieved by allocating the energy of the mixture signal at different frequencies to the sources whose harmonics are at the frequency.

1.2.4. MASA Informed by a Musical Score

While a computer system that takes the music audio as the only input addresses the MASA problem to some extent, in many cases the achieved results are not satisfying for further applications. What if besides the music audio, there is external information such as musical score, video, lyric, or meta data, which is related to the music? Will this information be helpful to analyze the music audio scene? Can we design computational models to leverage the external information when it is available?

In the second part of my dissertation, I consider the problem of MASA when a digital score (e.g. a MIDI file) of the music piece is also available. A digital score provides a symbolic representation (a "skeleton") of the music, and is directly machine-understandable. Digital scores for many kinds of music are widely available. They provide useful side information for computers to achieve better MASA performance. To utilize the score information, I propose to first align the music audio and the score in Chapter 5. A hidden Markov process model is proposed to infer the score location of each audio frame. Then in Chapter 6 I propose to use the pitch information provided by the aligned score to refine the multi-pitch estimation from the audio, and based on which a simple pitch-based source separation approach is used to achieve significantly better source separation results.

For humans, leveraging external information does help us analyze the music audio scene. Take the musical score as an example, it helps a conductor to identify and follow a

specific instrument part during an orchestral rehearsal. It also helps a singer in a choir to "locate" his/her voice among others' when learning this chorus. Therefore, it is reasonable to think that external information will also help a computer system to analyze the music audio scene.

Why would a musical score help analyze the audio scene? Technically speaking, the score helps reduce the search space for solutions. The score tells us what notes are supposed to be played and how they should be organized in the audio. For example, assume a system is trying to transcribe the four notes currently sounding in the audio. Instead of exploring the full four dimensional pitch space, we can look at a much smaller subset, i.e. all possible chords written in the score, and find the one that is the best match. We can also use the ordering of chords in the score to help find the best match, assuming the audio performance follows the order in the score.

Figure 1.2. Analogy between MASA with a score and 3D scene understanding with a tourist map.

MASA with a musical score is analogous to 3D scene understanding with a tourist map, as shown in Figure 1.2. When a tourist explores a place, the map, which presents an abstract representation of this place, can be very helpful. The abstract drawings of tourist spots can help the tourist localize himself by matching the drawings with the real scene. The spatial relationships between tourist spots can guide the tourist from one spot to another. However, to make the analogy more accurate, the map must also use an inconsistent scale for different places, as the performer may slow down or speed up in ways that are not precisely described in the score.

People may think that MASA becomes trivial when a score is present. However, this is not the case. The score only tells what musical objects to look for, but not where to look nor what they sound like. Think about the tourist analogy in Figure 1.2 again. With a map at hand, we still need to be able to see the real scene, match the real scene with drawings on the map, and find our way. Similarly, the proposed MASA system needs to be able to recognize potential musical objects in the audio (multi-pitch estimation), match them with those written in the score (audio-score alignment), and segregate their physical signals (score-informed source separation).

The requirement of a score does narrow the scope of the MASA system, since scores of some music styles may be hard to access and some audio performances (e.g. Jazz) may be unfaithful to their scores. However, we argue that this is not a big issue. There is still a large body of music (e.g. classical music) that has musical scores available. Tens of thousands of them are available in the digital format on the web at sources such as `http://www.classicalmidiconnection.com`. In addition, many audio performances are faithful to their scores. Even for improvised music like Jazz, the score (usually called

a lead sheet) still provides some useful information such as a musical form and chord progressions for analyzing the music scene. However, the proposed MASA system needs to use a more robust representation of audio and score [**30**]. In my dissertation, I will only address faithful audio performances, where the performers do not intentionally change the musical objects written in the score.

1.3. Related Areas

1.3.1. Auditory Scene Analysis (ASA)

ASA studies how the human auditory system organizes sound into perceptually meaningful elements from the psychoacoustic perspective [**8**]. It is believed that there is a two-stage process. The first stage is called *segmentation*, where the acoustic input is decomposed into a collection of time-frequency regions. The second stage is called *grouping*, where different time-frequency regions that are mainly from the same source are combined into perceptually meaningful structures such as a note or a word. Grouping can be operated simultaneously (*simultaneous grouping*) such as grouping harmonics into a note, or sequentially (*sequential grouping*) such as grouping several footsteps into a single walking sound source.

Researchers have discovered a number of perceptual cues that the human auditory system uses in the grouping process, including proximity in frequency and time, common periodicity, common onset and offset, common amplitude and frequency modulation, common rhythm, common spatial location, similarity in timbre, etc. However, these cues are usually discovered by presenting listeners with simple laboratory stimuli such as pure tones. Whether these cues are applicable to organize real-world sounds such as music

and speech has not been thoroughly explored in ASA research, as typically practiced in psychoacoustics labs. The work in this dissertation addresses automated grouping of real-world music audio by machine.

1.3.2. Computational Auditory Scene Analysis (CASA)

CASA is the study of constructing computer systems (or computational models) that can achieve human performance in ASA [**37, 134**]. Researchers usually start from a specific application (e.g. speech segregation [**113, 135**], sound localization [**81, 139**], or music transcription [**72**]), and then design computational models for the perceptual cues that the human auditory system is believed to use in these applications.

ASA and CASA lay out the foundation of the first part of my dissertation. Some of my proposed methods are inspired by the perceptual cues that ASA has discovered. For example, multi-pitch estimation (Chapter 2) essentially explores the common periodicity cue to estimate a fundamental frequency by grouping its harmonics simultaneously. Multi-pitch streaming (Chapter 3) can be viewed as a sequential grouping problem, where pitch estimates in different time frames are grouped according to the time-frequency proximity cue and the timbre similarity cue.

1.3.3. Music Information Retrieval (MIR)

MIR studies how to extract musically meaningful information from various kinds of music data, including music audio, MIDI files, score sheets, meta data (e.g. artist, album), music listener social network data, etc. Since the first international conference in 2000, MIR has been growing fast into a highly interdisciplinary area that integrates traditional fields

such as psychoacoustics, musical acoustics, music cognition, signal processing, machine learning, and information retrieval.

MIR is an application-oriented research area. There are many challenging real-world problems such as audio classification [**121, 75**], cover song identification [**40, 111**], multi-pitch analysis [**70, 33**], chord estimation [**76**], beat tracking [**39**], audio structural segmentation [**78**], to name a few. Among these many problems, music audio scene analysis has always been a main theme, due to the most popularity of the audio format of music.

New problems are also continuously being proposed by the research community in MIR. Recently one trend is multi-modal music processing [**88**], i.e. to fuse information extracted from different types of music data, including music audio, MIDI, score sheet, meta data, and social network data. It has the potential to achieve better results than individually processing each individual type. The second part of the dissertation concerns leveraging musical score information to help analyze the music audio scene. I will show that this does significantly improve the performance, to a level hopefully practical in the real world.

1.4. Summary of Contributions

In this section, I summarize the main contributions of my dissertation.

(1) In Chapter 2, a novel multi-pitch estimation (MPE) method is proposed to estimate pitches and polyphony from recorded mixtures of multiple concurrent sound sources, such as a string quartet. Previous frequency domain MPE methods only model the relation between pitches and spectral peaks and tend to overfit the peaks. The proposed method also models the relation between pitches and the

non-peak region of the spectrum and avoids the overfitting. The proposed method significantly reduces the computational complexity compared to existing methods. It also uses contextual information to refine single-frame pitch estimates which existing methods do not use. Experiments show the proposed method outperforms two state-of-the-art MPE methods on both pitch and polyphony estimation.

(2) In Chapter 3, a novel multi-pitch streaming (MPS) method is proposed to stream pitch estimates in individual time frames into multiple pitch trajectories, each of which corresponds to a source. This is the first MPS method that performs in an unsupervised way, i.e. no pretraining of the source models are required. The proposed method formulates the problem in a constrained clustering framework. It combines two perceptual cues in sequential grouping discovered in the auditory scene analysis (ASA) research (Section 1.3.1): timbre similarity represented by the clustering objective, and proximity in time and frequency represented by the constraints.

(3) In chapter 4, the proposed MPE and MPS methods are extended to work with harmonic sound mixtures other than polyphonic music. More specifically, a novel cepstral feature is proposed to characterize the timbre of a talker. This feature can be calculated directly from the multi-talker speech mixture signal, without the need of separation of the talker's signal. This feature may be helpful in many speech research problems such as speech/speaker recognition in multi-talker or noisy environments.

(4) In Chapter 5, a novel method is proposed to align a polyphonic music audio with its score. This is the first method that performs online alignment (i.e. score following) of polyphonic music audio with multiple instruments on a single microphone, with no prior training on the specific piece of music.

(5) In Chapter 7, a novel interactive music editing software is proposed, using the proposed score-informed source separation technique. A user can segregate the signal of a note or all notes of a source in a piece of music, by clicking on the corresponding note or source in its MIDI score. Then the user can edit the volume, pitch, timbre and timing of the note or all notes of the source. In this chapter, a real-time polyphonic music source separation application called "Soundprism" is also envisioned. Provided a MIDI score, Soundprism can separate a piece of polyphonic music audio into source signals in real time.

(6) All the code of the proposed MPE, MPS, Audio-score alignment and score-informed source separation algorithms can be freely downloaded at `http://www.cs.northwestern.edu/~zdu459/resource/Resources.html`.

1.5. Broader Impact

Besides the benefits described in Section 1.1 and 1.4, my dissertation may have impact on broader areas.

(1) Computational models for MASA can help people better organize music. In the information era, finding ways to automatically index, label and search the increasing amount of music documents is becoming one of the key problems. These operations, to be perceptually meaningful, heavily rely on computational models

for MASA. For example, automatically searching for music recordings with similar melodies requires extracting melodies from music recordings containing many simultaneous sounds (guitar, bass and drums).

(2) Working with speech data, the proposed MPE and MPS methods may help prosody analysis and speech recognition of tonal languages such as Chinese. The proposed novel cepstral feature in Chapter 4 characterizes the timbre of a talker. Since it can be directly calculated from the mixture signal, it may open an avenue for speaker recognition/identification in multi-talker or noisy environments.

(3) Besides polyphonic music (Chapter 2 and 3) and multi-talker speech (Chapter 4), the proposed MPE and MPS methods can deal with other harmonic sound mixtures such as bird songs. This could be beneficial for biological environment surveillance. In fact, I am collaborating with Jonathan Springer and Bryan Pardo on a project of concurrent bird species recognition from bird songs [**116**]. The idea is to perform MPE and MPS to estimate the pitch trajectory of each concurrent bird song, based on which to recognize the bird species.

(4) The proposed interactive music editing application in Chapter 7 can be helpful for music education. Imagine a student who studies music composition. One effective way is to analyze classical music masterpieces and try to ask oneself why it is better to compose this way. With the proposed interactive music editor, the student can modify the pitch, dynamics, timing, instrumentation and timbre of a single note or an instrumental part of the masterpieces, and play back the audio after the modification. By comparing the modified versions with the original version, he/she can have a direct feel of which is better.

(5) The envisioned real-time score-informed music source separation application Soundprism could enable novel interactions in music entertainment. For example, Soundprism can be implemented in a mobile device. An audience member in a concert equipped with this device (and a headset) could switch between enjoying the full-band performance and focusing on the solo performance of the violin, or the violin-cello duet, according his/her personal preference. He/she can also view the aligned score through the device at the same time. This adds to the traditional passive listening activity a number of active selection and attention opportunities. Similarly, this device can be used in radio, TV or any music streaming scenarios.

Part 1

Analyzing the Music Audio Scene without A Written Score

For polyphonic music audio composed of harmonic sound sources, analyzing the music audio scene solely from audio requires multi-pitch analysis. Multi-pitch analysis is the problem of estimating the pitch content of each harmonic source in each time frame from the audio mixture. With the pitch content estimated, the significant frequency components (harmonics) of the harmonic source are organized, and the source signal can be separated from the mixture. Many musical objects such as notes, chords, melody lines, etc. can be extracted as well.

Multi-pitch analysis is a fundamental problem in audio signal processing. In music information retrieval, it is of great interest to researchers working in automatic music transcription [**72**], source separation [**35**], melody extraction [**58**], etc. In speech processing, it is helpful for multi-talker speech recognition [**19**] and prosody analysis [**63**]. It is also a step towards solving the cocktail party problem [**14**].

According to MIREX[2], multi-pitch analysis can be addressed at three levels. The first (and easiest) level is to collectively estimate pitch values of all concurrent sources at each individual time frame, without determining their sources. This is also known as *multi-pitch estimation (MPE)*. Most work in multi-pitch analysis performs at this level and a number of methods have been proposed. For music, time domain methods [**120, 23, 22**] and frequency domain methods [**83, 53, 141, 93, 70, 41, 109, 33**] have been proposed. For speech, several methods [**140, 112, 3, 64**] estimate pitches of two concurrent speakers, but no existing work addresses three or more concurrent speakers.

[2]The Music Information Retrieval Evaluation eXchange (MIREX) is an annual evaluation campaign for Music Information Retrieval (MIR) algorithms. Multiple Fundamental Frequency Estimation & Tracking is one of its tasks.

The second level is called *note tracking* in music information retrieval. The task is to estimate continuous segments that typically correspond to individual notes or syllables. This is often achieved by assuming the continuity and smoothness of the pitch contours in the model, connecting pitch estimates that are close in both time and frequency. Note that each pitch contour comes from one source but each source can have many contours (e.g. one contour per musical note or spoken word). Several methods have been proposed to perform at this level, for music [**108, 95, 66, 13**] or speech [**74**].

The third (and most difficult) level is to stream pitch estimates into a single pitch trajectory over an entire conversation or music performance for each of the concurrent sources. The trajectory is much longer than those estimated at the second level, and contains a number of discontinuities that are caused by silence, non-pitched sounds and abrupt frequency changes. Therefore, techniques used at the second level to connect close pitch estimates are not enough to connect short pitch segments into streams. We argue timbre information is needed to connect discontinuous pitch segments of a single sound source. I call the third level *multi-pitch streaming*[3].

In the first part of my dissertation, I address MASA through multi-pitch analysis. In Chapter 2, I introduce my work on MPE, i.e. the first level of multi-pitch analysis. In Chapter 3, I introduce my work on MPS, i.e. the third level of multi-pitch analysis. My work on multi-pitch analysis can be applied to not only music, but also speech. I describe some results I obtained on speech data in Chapter 4.

[3]This is also sometimes called *multi-pitch tracking*, however multi-pitch tracking also refers to the first or second level in the literature. Therefore, I use streaming to refer the third level.

CHAPTER 2

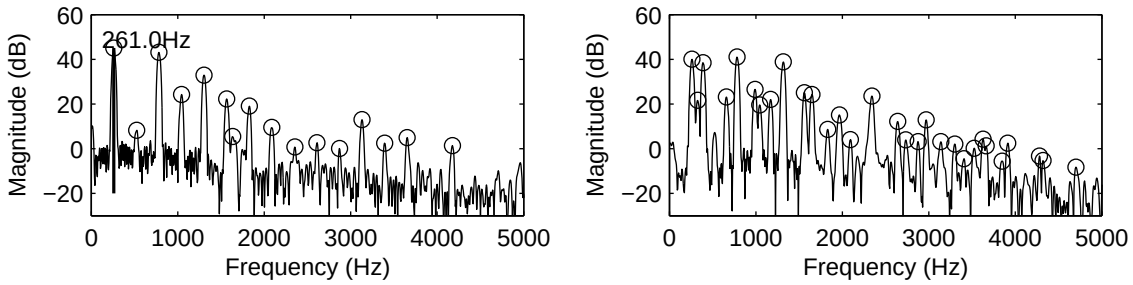
Multi-pitch Estimation

2.1. Introduction

Multi-pitch Estimation (MPE), or Multiple F0 Estimation (MF0E), is the task of estimating the pitches (fundamental frequencies, F0s) and the number of pitches (polyphony) in each time frame of polyphonic audio composed of harmonic sound sources. It is of great interest to researchers working in music audio and is useful for many applications, including automatic music transcription [**95**], source separation [**137**] and score following [**90**]. The task, however, remains challenging and existing methods do not match human ability in either accuracy or flexibility.

2.1.1. Why Is MPE So Difficult?

According to the number of concurrent sounds, music audio can be classified into monophonic and polyphonic. *Monophonic* music has at most one sound at a time, e.g. a clarinet or vocal solo. In monophonic music, there is at most one pitch at a time. Therefore, pitch estimation for monophonic music is called single pitch detection. *Polyphonic* music, however, can have more than one sounds at a time. A duet, a quintet and a symphony are all polyphonic. Some instruments such as piano and guitar are also polyphonic in nature.



(a) (b)

Figure 2.1. Monophonic and polyphonic spectra. Significant peaks are marked by circles. (a) a note by clarinet (C4); (b) a C major chord, played by clarinet (C4), oboe (E4) and flute (G4).

Single pitch detection is relatively easy and has been well solved [**24**]. As shown in Figure 2.1-(a), significant peaks in a monophonic spectrum form a clear harmonic structure. Its F0 can be easily inferred from this structure, and harmonics are well represented by spectral peaks. In Figure 2.1-(b), however, when the clarinet note is mixed with two other notes to compose a triad chord, the positions of spectral peaks become irregular. It is hard to find the original harmonic structure in Figure 2.1-(a) from a subset of peaks here. This is because of the notorious "overlapping harmonic" issue. Harmonics of concurrent sounds often overlap with each other, resulting in significant changes in amplitude, shape and even frequency of spectral peaks. Harmonics with small amplitude can even be *masked* by other harmonics or their sidelobes, and hence are not represented by peaks. Compared with general sound mixtures, this issue is especially severe in polyphonic music. This is because one of the guidelines in composing music is to make concurrent sounds consonant, which leads to small integer ratios between their F0s and therefore many overlaps between their harmonics. Take a C major chord as an

example, one can calculate that 46.7% of the harmonics of the note C4 overlap with the other two notes. E4 has 33.3% overlap and G4 has 60%.

The overlapping harmonic issue makes the correspondence between spectral peaks and harmonics noisy. This, however, is not the only problem. It can be seen that the spectrum in Figure 2.1-(b) has in total 28 significant peaks below 5000Hz. If we are not told that it is a triad chord, how many sources shall we infer from the spectrum? Let us assume each peak is caused by only a single source and not a combination of sources. Should there be 28 concurrent sources, each of which corresponds to one peak? Suppose we know there are three sources, how should the peaks be associated with the sources? If the peaks are independent to each other, there will be $3^{28} \approx 23$ trillion possible associations. Clearly we still need to use the knowledge about harmonic sources to organize these peaks and infer corresponding harmonics. However, this is still very challenging.

Besides the overlapping harmonic issue, there are two additional difficulties of MPE. First, the search space for solutions is large. Suppose there are four concurrent pitches in a frame, and each one can arbitrarily take a value from the pitch range of the piano (88 musical semitones). Suppose we want to estimate the F0 of each pitch at the precision of ten musical cents (i.e. 0.1 semitone). Then there would be 880 possible values for each pitch and $880^4 \approx 6 \times 10^{11}$ possible solutions for a single frame of audio with 4 pitched sound sources. Clearly, a brute force search would be very time-consuming. Second, polyphony, i.e. the number of concurrent pitches, usually needs to be estimated. Even in the case that the number of instruments are known in a piece of music, we are only given the maximum polyphony at different time frames, as some instruments may be inactive.

2.1.2. Related Work

All those who develop MPE systems must make certain design choices. The first of these is how to preprocess the audio data and represent it. Some researchers do not employ any preprocessing of the signal and represent it with the full time domain signal or frequency spectrum. In this category, discriminative model-based [**95**], generative model-based [**22, 126**], graphical model-based [**67**], spectrum modeling-based [**53, 66, 109, 36, 41**] and genetic algorithm-based [**104**] methods have been proposed.

Because of the high dimensionality of the original signal, researchers often preprocess the audio with some method to retain salient information, while abstracting away irrelevant details. One popular data reduction technique has been to use an auditory model to preprocess the audio. Meddis and Mard [**85**] proposed a unitary model of pitch perception for single F0 estimation. Tolonen and Karjalainen [**120**] simplified this model and applied it to multiple F0 estimation of musical sounds. de Cheveigné and Kawahara [**23**] integrated the auditory model and used a temporal cancelation method for F0 estimation. Klapuri [**69, 70**] used auditory filterbanks as a front end, and estimated F0s in an iterative spectral subtraction fashion. It was reported that [**70**] achieves the best performance among methods in this category.

Another more compact data reduction technique is to reduce the full signal (complex spectrum) to observed power spectral peaks [**52, 83, 119, 77, 34, 93, 141**]. The rationale is that peaks are very important in terms of human perception. For example, re-synthesizing a harmonic sound using only peaks causes relatively little perceived distortion [**115**]. In addition, peaks contain important information for pitch estimation

because, for harmonic sounds, they typically appear near integer multiples of the fundamental frequency. Finally, this representation makes it easy to mathematically model the signal and F0 estimation process. Given these observations, we believe this representation can be used to achieve good results.

For MPE methods in this category, a key question is how to infer F0s from peaks, or more specifically, how to model the relationship between peaks and harmonics of a F0 hypothesis. One way is to define a deterministic process to match peaks and harmonics, and define some function to measure how well they match. This can then be used as the salience of an F0 hypothesis [**83, 141, 93**]. In defining the salience function, the harmonic energy cue (i.e. that a correct F0 must have strong harmonics), the harmonic smoothness cue (i.e. that harmonics of the same F0 form a smooth spectral envelope) and the harmonic synchronization cue (i.e. that harmonics of the same F0 have similar temporal evolution) are often employed. These deterministic methods, however, suffer from several problems. First, it is very difficult to define a general matching process between peaks and harmonics in the polyphonic scenario. A deterministic one-to-one correspondence between peaks and harmonics is inherently nonrealistic because of the overlapping harmonic issue. Second, the defined salience functions usually combine different cues in a manually tuned way with a number of weighting parameters lacking a systematic framework, potentially impacting their generalizability.

Another way to model the relation between peaks and harmonics is through probabilistic models [**52, 119, 77**]. These methods view spectral peaks as being generated from F0s and their harmonics in a probabilistic manner, and they adopt the maximum likelihood framework to estimate concurrent pitches (parameters) from the spectral peaks

(observations). Although they are flexibility in modeling the noisy relation between peaks and harmonics, they have the following problems. First, their computational complexity is very high since they consider every possible peak-harmonic association in defining the likelihood, which is exponential to the number of peaks and harmonics. Second, they tend to overfit spectral peaks which leads to various pitch estimation errors. This is because they only view spectral peaks as observations and favor F0 hypotheses whose harmonics could generate the spectral peaks. However, an F0 estimate which is one octave lower than the true F0 could still generate the peaks well, using only even harmonics, although many of its odd harmonics may not find peaks to generate. Third, these methods (and the deterministic methods) all assume that the polyphony of the signal is known. This can be problematic if the polyphony is unknown or changes with time.

2.1.3. Advances of the Proposed Method

In this chapter, I address the multiple F0 estimation problem in a maximum likelihood fashion, similar to [**52, 119, 77**] and adopting the idea in [**83, 41**]. I model the observed power spectrum as a set of peaks and the non-peak region. I define the *peak region* as the set of all frequencies within d of an observed peak. The *non-peak region* is defined as the complement of the peak region (see Section 2.2 for detailed definitions). I then define a likelihood on both the peak region and the non-peak region, and the total likelihood function as their product. The peak region likelihood helps find F0s that have harmonics that explain peaks, while the non-peak region likelihood helps avoid F0s that have harmonics in the non-peak region. They act as a complementary pair. I adopt an iterative way to estimate F0s one by one to avoid the combinatorial problem of concurrent

F0 estimation. I also propose a polyphony estimation method to terminate the iterative process. Finally, I propose a postprocessing method to refine the polyphony and F0 estimates using neighboring frames. The refinement method eliminates many inconsistent estimation errors.

Table 2.1. Comparison of the proposed method with existing spectral peak modeling methods.

	Deterministic			Probabilistic			
	[**83**]	[**141**]	[**93**]	[**52**]	[**119**]	[**77**]	Proposed
Designed for multi-pitch estimation		✓	✓			✓	✓
Models peak amplitude	✓	✓	✓		✓	✓	✓
Avoids overfitting peaks	✓	✓	✓				✓
Learns parameters from data		✓		✓			✓
Computationally economic	✓	✓		✓			✓
Polyphony estimation			✓				✓
Utilizes contextual information			✓				✓
Tested on a large polyphonic dataset		✓	✓				✓

I have published the proposed method in [**33**]. Table 2.1 compares it with existing methods in the spectral peak modeling category. Among theses advances, the most important ones are:

- It avoids the peak overfitting problem in [**52, 119, 77**]. The peak-region likelihood tends to overfit peaks, but the non-peak region likelihood penalizes this overfitting. The two parts act complementarily and are organized in a unified probabilistic framework.
- It significantly reduces the computational complexity compared to [**119, 77**]. The proposed likelihood model assumes conditional independence between peaks given F0s, which avoids enumerating all possible associations between peaks and harmonics. Also, the proposed algorithm adopts a greedy search strategy in

estimating concurrent pitches. These operations reduce the complexity from exponential to linear in the number of peaks and pitches.

- It proposes a simple polyphony estimation method that shows superior performance compared to a state-of-the-art method [**70**]. Existing multi-pitch estimation methods in the spectral modeling category [**141, 77**] often require the polyphony of the audio as an input.
- It refines F0 estimates in each frame using neighboring frames, while most related methods [**141, 77**] do not use context information.

2.1.4. System Overview

Algorithm 1 shows the overview of the proposed approach. I assume an audio file has been normalized to a fixed Root Mean Square (RMS) energy and segmented into a series of (possibly overlapping) time windows called *frames*. For each frame, a Short Time Fourier Transform (STFT) is performed with a hamming window and four times zero-padding to get a power spectrum.

Spectral peaks are then detected by the peak detector described in [**35**]. Basically, there are two criteria that determine whether a power spectrum local maximum is labeled a peak. The first criterion is global: the local maximum should not be less than some threshold (e.g. 50dB) lower than the global maximum of the spectrum. The second criterion is local: the local maximum should be locally higher than a smoothed version of the spectrum by at least some threshold (e.g. 4dB). Finally, the peak amplitudes and frequencies are refined by quadratic interpolation [**115**].

Algorithm 1: Proposed Multi-pitch Estimation Algorithm.

Input : A piece of polyphonic audio composed of harmonic sound sources, segmented into overlapping time frames.
Output: A set of pitches in each audio frame.

for *each frame of audio* **do**
  Find peak frequencies and amplitudes with [**35**];
  $\mathcal{C}$ = a finite set of frequencies within d of peak frequencies;
  $\boldsymbol{\theta} = \emptyset$;
  for $N = 1 : MaxPolyphony$ **do**
    for *each* F_0 *in* $\mathcal{C}$ **do**
      Evaluate Eq. (2.2) on $\boldsymbol{\theta} \bigcup \{F_0\}$ (Section 2.2);
    end
    Add to $\boldsymbol{\theta}$ the F_0 that maximized Eq. (2.2);
  end
  Estimate actual polyphony N with Eq. (2.18) (Section 2.3);
  Return the first N estimates in $\boldsymbol{\theta} = \{F_0^1, \cdots, F_0^N\}$;
end
for *each frame of the audio* **do**
  Refine F0 estimates using neighboring frames (Section 2.4);
end

Given this set of peaks, a set C of candidate F0s is generated. To facilitate computation, I do not consider the "missing fundamental" situation in this paper. Candidate F0 values are restricted to a range of $\pm$ 6% in Hz (one semitone) of the frequency of an observed peak. I consider increments with a step size of 1% in Hz of the peak frequency. Thus, for each observed peak there are 13 candidate F0 values. In implementation, one can further reduce the search space by assuming F0s only occur around the 5 lowest frequency peaks, 5 highest amplitude peaks and 5 locally highest amplitude peaks (peak amplitudes minus the smoothed spectral envelope). This gives at most $15 \cdot 13 = 195$ candidate F0s for each frame.

A naive approach to finding the best set of F0s would have to consider the power set of these candidates: 2^{195} sets. To deal with this issue, I use a greedy search strategy, which estimates F0s one by one. This greatly reduces the time complexity (for a complexity analysis see Section 2.5).

At each iteration, a newly estimated F0 is added to the existing F0 estimates until the maximum allowed polyphony is reached. Then, a post processor (Section 2.3) determines the best polyphony using a threshold base on the likelihood improvement as each F0 estimate is added. Finally, each frame's F0 estimates are refined using information from estimates in neighboring frames (see Section 2.4).

2.2. Estimating F0s Given the Polyphony

This section describes how I approach Line 6 and 7 in Algorithm 1. Given a time frame presumed to contain N monophonic harmonic sound sources, I view the problem of estimating the fundamental frequency (F0) of each source as a Maximum Likelihood parameter estimation problem in the frequency domain,

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathcal{L}\left(\boldsymbol{\theta}\right) = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} p\left(\mathbf{O}|\boldsymbol{\theta}\right), \tag{2.1}$$

where $\boldsymbol{\theta} = \{F_0^1, \cdots, F_0^N\}$ is a set of N fundamental frequencies to be estimated, $\boldsymbol{\Theta}$ is the space of possible sets $\boldsymbol{\theta}$, and $\mathbf{O}$ represents our observation from the power spectrum.

I assume that the spectrum is analyzed by a peak detector, which returns a set of peaks. The observation to be explained is the set of peaks *and* the non-peak region of the spectrum.

I define the *peak region* as the set of all frequencies within d of an observed peak. The *non-peak region* is defined as the complement of the peak region. I currently define d as a musical quarter tone, which will be explained in Section 2.2.2. Then, similar to [**119, 77**], peaks are further categorized into normal peaks and spurious peaks. From the generative model point of view, a *normal* peak is defined as a peak that is generated by a harmonic of an F0. Other peaks are defined as *spurious* peaks, which may be generated by peak detection errors, noise, sidelobes, etc.

The peak region likelihood is defined as the probability of occurrence of the peaks, given an assumed set of F0s. The non-peak region likelihood is defined as the probability of *not* observing peaks in the non-peak region, given an assumed set of F0s. The peak region likelihood and the non-peak region likelihood act as a complementary pair. The former helps find F0s that have harmonics that explain peaks, while the latter helps avoid F0s that have harmonics in the non-peak region.

We wish to find the set $\boldsymbol{\theta}$ of F0s that maximizes the probability of having harmonics that could explain the observed peaks, and minimizes the probability of having harmonics where no peaks were observed. To simplify calculation, I assume independence between peaks and the non-peak region. Correspondingly, the likelihood is defined as the multiplication of two parts: the peak region likelihood and the non-peak region likelihood:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{\text{peak region}}(\boldsymbol{\theta}) \cdot \mathcal{L}_{\text{non-peak region}}(\boldsymbol{\theta}). \tag{2.2}$$

The parameters of the models are learned from training data, which are summarized in Table 2.2 and will be described in detail in the following.

Table 2.2. Parameters Learned From Training Data. The first four probabilities are learned from the polyphonic training data. The last one is learned from the monophonic training data.

$P(s_k)$	Prob. a peak k is normal or spurious
$p(f_k, a_k \mid s_k = 1)$	Prob. a spurious peak has frequency f_k and amplitude a_k
$p(a_k \mid f_k, h_k)$	Prob. a normal peak has amplitude a_k, given its frequency f_k and it is harmonic h_k of an F0
$p(d_k)$	Prob. a normal peak deviates from its corresponding ideal harmonic frequency by d_k
$P(e_h \mid F_0)$	Prob. the h-th harmonic of F_0 is detected

2.2.1. Peak Region Likelihood

Each detected peak k in the power spectrum is represented by its frequency f_k and amplitude a_k. Given K peaks in the spectrum, I define the peak region likelihood as

$$\mathcal{L}_{\text{peak region}}(\boldsymbol{\theta}) = p(f_1, a_1, \cdots, f_K, a_K \mid \boldsymbol{\theta}) \tag{2.3}$$

$$\approx \prod_{k=1}^{K} p(f_k, a_k \mid \boldsymbol{\theta}) . \tag{2.4}$$

Note that f_k, a_k and all other frequencies and amplitudes in this paper are measured on a logarithmic scale (musical semitones and dB, respectively)[1]. This is done for ease of manipulation and accordance with human perception. Because frequency is calculated in the semitone scale, the distance between any two frequencies related by an octave is always 12 units. I adopt the general MIDI convention of assigning the value 60 to Middle C (C4, 262Hz) and use a reference frequency of A=440Hz. The MIDI number for A=440Hz is 69, since it is 9 semitones above Middle C.

[1]FREQUENCY: MIDI number = 69+12$\times \log_2$(Hz/440); AMPLITUDE: dB = 20$\times \log_{10}$(Linear amplitude).

From Eq. (2.3) to Eq. (2.4), I assume[2] conditional independence between observed peaks, given a set of F0s. Given a harmonic sound, observed peaks ideally represent harmonics and are at integer multiples of F0s. In practice, some peaks are caused by inherent limitations of the peak detection method, non-harmonic resonances, interference between overlapping sound sources and noise. Following the practice of [**119**], I call peaks caused by harmonics *normal* peaks, and the others *spurious* peaks. We need different models for normal and spurious peaks.

For monophonic signals, there are several methods to discriminate normal and spurious peaks according to their shapes [**106, 105**]. For polyphonic signals, however, peaks from one source may overlap peaks from another. The resulting composite peaks cannot be reliably categorized using these methods. Therefore, I introduce a binary random variable s_k for each peak to represent that it is normal ($s_k = 0$) or spurious ($s_k = 1$), and consider both cases in a probabilistic way:

$$p\left(f_k, a_k|\boldsymbol{\theta}\right) = \sum_{s_k} p\left(f_k, a_k|s_k, \boldsymbol{\theta}\right) P\left(s_k|\boldsymbol{\theta}\right) \tag{2.5}$$

$P\left(s_k|\boldsymbol{\theta}\right)$ in Eq. (2.5) represents the prior probability of a detected peak being normal or spurious, given a set of F0s[3]. I would like to learn it from training data. However, the size of the space for $\boldsymbol{\theta}$ prohibits creating a data set with sufficient coverage. Instead, I neglect the effects of F0s on this probability and learn $P\left(s_k\right)$ to approximate $P\left(s_k|\boldsymbol{\theta}\right)$. This approximation is not only necessary, but also reasonable. Although $P\left(s_k|\boldsymbol{\theta}\right)$ is

[2]In this chapter, we use $\approx$ to denote "assumption".
[3]Here $P(\cdot)$ denotes probability mass function of discrete variables; $p(\cdot)$ denotes probability density of continuous variables.

influenced by factors related to F0s, it is much more influenced by the limitations of the peak detector, nonharmonic resonances and noise, all of which are independent of F0s.

I estimate $P(s_k)$ from *randomly mixed chords*, which are created using recordings of individual notes performed by a variety of instruments (See Section 2.6.1 for details). For each frame of a chord, spectral peaks are detected using the peak detector described in [**35**]. Ground-truth values for F0s are obtained by running YIN [**24**], a robust single F0 detection algorithm, on the recording of each individual note, prior to combining them to form the chord.

We need to classify normal and spurious peaks and collect their corresponding statistics in the training data. In the training data, we have the ground-truth F0s, hence the classification becomes possible. I calculate the frequency deviation of each peak from the nearest harmonic position of the reported ground-truth F0s. If the deviation d is less than a musical quarter tone (half a semitone), the peak is labeled normal, otherwise spurious. The justification for this value is: YIN is a robust F0 estimator. Hence, its reported ground-truth F0 is within a quarter tone range of the unknown true F0, and its reported harmonic positions are within a quarter tone range of the true harmonic positions. As a normal peak appears at a harmonic position of the unknown true F0, the frequency deviation of the normal peak defined above will be smaller than a quarter tone. In our training data, the proportion of normal peaks is 99.3% and is used as $P(s_k = 0)$.

In Eq. (2.5), there are two probabilities to be modeled, i.e. the conditional probability of the normal peaks $p(f_k, a_k|s_k = 0, \boldsymbol{\theta})$ and the spurious peaks $p(f_k, a_k|s_k = 1, \boldsymbol{\theta})$. I now address them in turn.

2.2.1.1. Normal Peaks. A normal peak may be a harmonic of only one F0, or several F0s when they all have a harmonic at the peak position. In the former case, $p\left(f_k, a_k | s_k = 0, \boldsymbol{\theta}\right)$ needs to only consider one F0. However, in the second case, this probability is conditioned on multiple F0s. This leads to a combinatorial problem we wish to avoid.

To do this, we adopt the assumption of binary masking [**143, 97**] used in some source separation methods. They assume the energy in each frequency bin of the mixture spectrum is caused by only one source signal. Here I use a similar assumption that each peak is generated by only one F0, the one having the largest likelihood to generate the peak.

$$p\left(f_k, a_k | s_k = 0, \boldsymbol{\theta}\right) \approx \max_{F_0 \in \boldsymbol{\theta}} p\left(f_k, a_k | F_0\right). \tag{2.6}$$

Now let us consider how to model $p\left(f_k, a_k | F_0\right)$. Since the k-th peak is supposed to represent some harmonic of F_0, it is reasonable to calculate the harmonic number h_k as the nearest harmonic position of F_0 from f_k.

Given this, I find the harmonic number of the nearest harmonic of an F0 to an observed peak as follows:

$$h_k = [2^{\frac{f_k - F_0}{12}}], \tag{2.7}$$

where $[\cdot]$ denotes rounding to the nearest integer. Now the frequency deviation d_k of the k-th peak from the nearest harmonic position of the given F0 can be calculated as:

$$d_k = f_k - F_0 - 12 \log_2 h_k. \tag{2.8}$$

Table 2.3. Correlation coefficients between several variables of normal peaks of the polyphonic training data.

	a	f	F_0	h	d
a	1.00	-0.78	**-0.11**	**-0.60**	-0.01
f	–	1.00	0.40	0.56	0.01
F_0	–	–	1.00	-0.41	-0.01
h	–	–	–	1.00	0.02
d	–	–	–	–	1.00

To gain a feel for how reasonable various independence assumptions between our variables might be, I collected statistics on the randomly mixed chord data described in Section 2.6.1. Normal peaks and their corresponding F0s are detected as described before. Their harmonic numbers and frequency deviations from corresponding ideal harmonics are also calculated. Then the correlation coefficient is calculated for each pair of these variables. Table 2.3 illustrates the correlation coefficients between f_k, a_k, h_k, d_k and F_0 on this data.

We can factorize $p\left(f_k, a_k|F_0\right)$ as:

$$p\left(f_k, a_k|F_0\right) = p\left(f_k|F_0\right) p\left(a_k|f_k, F_0\right). \tag{2.9}$$

To model $p\left(f_k|F_0\right)$, we note from Eq. (2.8) that the relationship between the frequency of a peak f_k and its deviation from a harmonic d_k is linear, given a fixed harmonic number h_k. Therefore, in each segment of f_k where h_k remains constant, we have

$$p\left(f_k|F_0\right) = p\left(d_k|F_0\right) \tag{2.10}$$

$$\approx p\left(d_k\right), \tag{2.11}$$

where in Eq. (2.11), $p\left(d_k|F_0\right)$ is approximated by $p(d_k)$. This approximation is supported by the statistics in Table 2.3, as the correlation coefficients between d and F_0 is very small, i.e. they are statistically independent.

Since I characterize $p\left(d_k\right)$ in relation to a harmonic, and I measure frequency in a log scale, I build a standard normalized histogram for d_k in relation to the nearest harmonic and use the same distribution, regardless of the harmonic number. In this work, I estimate the distribution from the randomly mixed chords data set described in Section 2.6.1. The resulting distribution is plotted in Figure 2.2.

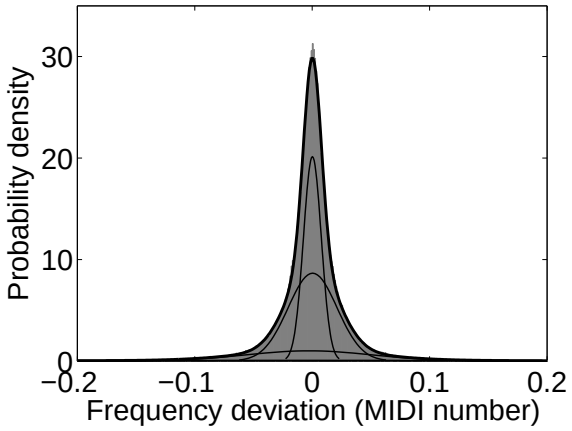


Figure 2.2. Illustration of modeling the frequency deviation of normal peaks. The probability density (bold curve) is estimated using a Gaussian Mixture Model with four kernels (thin curves) on the histogram (gray area).

It can be seen that this distribution is symmetric about zero, a little long tailed, but not very spiky. Previous methods [**52, 119, 77**] model this distribution with a single Gaussian. I found a Gaussian Mixture Model (GMM) with four kernels to be a better approximation. The probability density of the kernels and the mixture is also plotted in Figure 2.2.

To model $p\left(a_k|f_k, F_0\right)$, we observe from Table 2.3 that a_k is much more correlated with h_k than F_0 on our data set. Also, knowing two of f_k, h_k and F_0 lets one derive the third value (as in Eq. 2.8). Therefore, we can replace F_0 with h_k in the condition.

$$p\left(a_k|f_k, F_0\right) = p\left(a_k|f_k, h_k\right) = \frac{p\left(a_k, f_k, h_k\right)}{p\left(f_k, h_k\right)}. \tag{2.12}$$

I then estimate $p\left(a_k, f_k, h_k\right)$ using the Parzen window method [**92**], because it is hard to characterize this probability distribution with a parametric representation. An 11 (dB) $\times$ 11 (semitone) $\times$ 5 Gaussian window with variance 4 in each dimension is used to smooth the estimate. The size of the window is not optimized but just chosen to make the probability density look smooth.

I now turn to modeling those peaks that were not associated with a harmonic of any F0.

2.2.1.2. Spurious Peaks. By definition, a spurious peak is detected by the peak detector, but is not a harmonic of any F0 in $\boldsymbol{\theta}$, the set of F0s. The likelihood of a spurious peak from Eq. (2.4) can be written as:

$$p\left(f_k, a_k|s_k = 1, \boldsymbol{\theta}\right) = p\left(f_k, a_k|s_k = 1\right). \tag{2.13}$$

The statistics of spurious peaks in the training data are used to model Eq. (2.13). The shape of this probability density is plotted in Figure 2.3, where a 11 (semitone) $\times$ 9 (dB) Gaussian window is used to smooth it. Again, the size of the window is not optimized but just chosen to make the probability density look smooth. It is a multimodal distribution, however, since the prior probability of spurious peaks is rather small (0.007 for our training data), there is no need to model this density very precisely. Here

a 2-D Gaussian distribution is used, whose means and covariance are calculated to be $(82.1, 23.0)$ and $\begin{pmatrix} 481.6 & -89.5 \\ -89.5 & 86.8 \end{pmatrix}$.

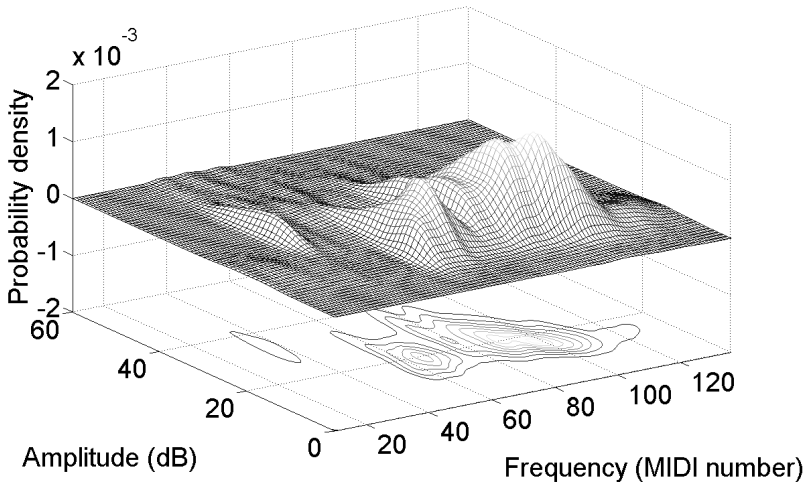


Figure 2.3. Illustration of the probability density of $p\left(f_k, a_k | s_k = 1\right)$, which is calculated from the spurious peaks of the polyphonic training data. The contours of the density is plotted at the bottom of the figure.

I have now shown how to estimate probability distributions for all the random variables used to calculate the likelihood of an observed peak region, given a set of F0s, using Eq. (2.3). I now turn to the non-peak region likelihood.

2.2.2. Non-peak Region Likelihood

As stated in the start of Section 2.2, the non-peak region also contains useful information for F0 estimation. However, how is it related to F0s or their predicted harmonics? Instead of telling us where F0s or their predicted harmonics should be, the non-peak region tells us where they should not be. A good set of F0s would predict as few harmonics as possible in the non-peak region. This is because if there is a predicted harmonic in the non-peak

region, then clearly it was not detected. From the generative model point of view, there is a probability for each harmonic being or not being detected. Therefore, I define the non-peak region likelihood in terms of the probability of *not* detecting any harmonic in the non-peak region, given an assumed set of F0s.

I assume that the probability of detecting a harmonic in the non-peak region is independent of whether or not other harmonics are detected. Therefore, the probability can be written as the multiplication of the probability for each harmonic of each F0, as in Eq. (2.14).

$$\mathcal{L}_{\text{non-peak region}}(\boldsymbol{\theta}) \approx \prod_{F_0 \in \boldsymbol{\theta}} \prod_{\substack{h \in \{1 \cdots H\} \\ F_h \in \mathcal{F}_{\text{np}}}} 1 - P\left(e_h = 1 | F_0\right), \tag{2.14}$$

where $F_h = F_0 + 12 \log_2^h$ is the frequency (in semitones) of the predicted h-th harmonic of F_0; e_h is the binary variable that indicates whether this harmonic is detected; $\mathcal{F}_{\text{np}}$ is the set of frequencies in the non-peak region; and H is the largest harmonic number we consider.

In the definition of the non-peak region in Section 2.2, there is a parameter d controlling the size of the peak region and the non-peak region. It is noted that this parameter does not affect the peak-region likelihood, but only affects the non-peak region likelihood. This is because the smaller d is, the larger the non-peak region is and the higher the probability that the set of F0 estimates predicts harmonics in the non-peak region.

Although the power spectrum is calculated with a STFT and the peak widths (main lobe width) are the same in terms of Hz for peaks with different frequencies, d should not be defined as constant in Hz. Instead, d should vary linearly with the frequency (in

Hz) of a peak. This is because d does not represent the width of each peak, but rather the possible range of frequencies in which a harmonic of a hypothesized F0 may appear. This possible range increases as frequency increases. In this work, d is set to a musical quarter tone, which is 3% of the peak frequency in Hz. This is also in accordance with the standard tolerance of measuring correctness of F0 estimation.

Now, to model $P\left(e_h = 1|F_0\right)$. There are two reasons that a harmonic may not be detected in the non-peak region: First, the corresponding peak in the source signal was too weak to be detected (e.g. high frequency harmonics of many instruments). In this case, the probability that it is not detected can be learned from monophonic training samples.

Second, there is a strong corresponding peak in the source signal, but an even stronger nearby peak of another source signal prevents its detection. I call this situation *masking*. As I am modeling the non-peak region likelihood, I only care about the masking that happens in the non-peak region.

To determine when masking may occur with our system, I generated 100,000 pairs of sinusoids with random amplitude differences from 0 to 50dB, frequency differences from 0 to 100Hz and initial phase difference from 0 to 2π. I found that as long as the amplitude difference between two peaks is less than 50dB, neither peak is masked if their frequency difference is over a certain threshold; otherwise the weaker one is always masked. The threshold is 30Hz for a 46ms frame with a 44.1kHz sampling rate. These are the values for frame size and sample rate used in our experiments.

For frequencies higher than 1030Hz, a musical quarter tone is larger than $1030 \times 2^{1/24} = 30.2$Hz. The peak region contains frequencies within a quarter tone of a peak, Therefore,

if masking takes place, it will be in the peak region. In order to account for the fact that the masking region due to the FFT bin size (30Hz) is wider than a musical quarter tone for frequencies under 1030 Hz, I also tried a definition of d that chose the maximum of either a musical quarter tone or 30Hz: $d = \max(0.5 \text{ semitone}, 30\text{Hz})$. I found the results were similar to those achieved using the simpler definition of of $d = 0.5$ semitone.

Therefore, I disregard masking in the non peak region. I estimate $P\left(e_h^n = 1|F_0\right)$, i.e. the probability of detecting the h-th harmonic of F_0 in the source signal, by running our peak detector on the set of individual notes from a variety of instruments used to compose chords in Section 2.6.1. F0s of these notes are quantized into semitones. All examples with the same quantized F0 are placed into the same group. The probability of detecting each harmonic, given a quantized F0 is estimated by the proportion of times a corresponding peak is detected in the group of examples. The probability for an arbitrary F0 is then interpolated from these probabilities for quantized F0s.

Figure 2.4 illustrates the conditional probability. It can be seen that the detection rates of lower harmonics are large, while those of the higher harmonics become smaller. This is reasonable since for many harmonic sources (e.g. most acoustic musical instruments) the energy of the higher frequency harmonics is usually lower. Hence, peaks corresponding to them are more difficult to detect. At the right corner of the figure, there is a triangular area where the detection rates are zero, because the harmonics of the F0s in that area are out of the frequency range of the spectrum.

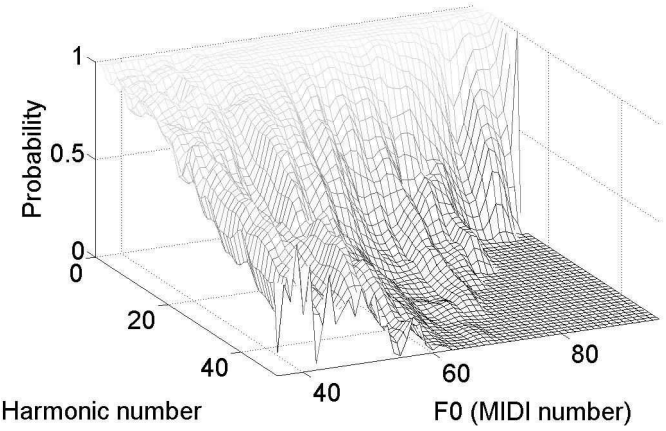


Figure 2.4. The probability of detecting the h-th harmonic, given the F0: $P\left(e_h = 1|F_0\right)$. This is calculated from monophonic training data.

2.3. Estimating the Polyphony

Polyphony estimation is a difficult subproblem of multiple F0 estimation. Researchers have proposed several methods together with their F0 estimation methods [**66, 70, 93**].

In this work, the polyphony estimation problem is closely related to the overfitting often seen with maximum likelihood methods. Note that in Eq. (2.6), the F_0 is selected from the set of estimated F0s, $\boldsymbol{\theta}$, to maximize the likelihood of each normal peak. As new F0s are added to $\boldsymbol{\theta}$, the maximum likelihood will never decrease and may increase. Therefore, the larger the polyphony, the higher the peak likelihood is:

$$\mathcal{L}_{\text{peak region}}(\hat{\boldsymbol{\theta}}^n) \leq \mathcal{L}_{\text{peak region}}(\hat{\boldsymbol{\theta}}^{n+1}), \tag{2.15}$$

where $\hat{\boldsymbol{\theta}}^n$ is the set of F0s that maximize Eq. (2.2) when the polyphony is set to n. $\hat{\boldsymbol{\theta}}^{n+1}$ is defined similarly. If one lets the size of $\boldsymbol{\theta}$ range freely, the result is that the explanation returned would be the largest set of F0s allowed by the implementation.

This problem is alleviated by the non-peak region likelihood, since in Eq. (2.14), adding one more F0 to $\boldsymbol{\theta}$ should result in a smaller value $\mathcal{L}_{\text{non-peak region}}(\boldsymbol{\theta})$:

$$\mathcal{L}_{\text{non-peak region}}(\hat{\boldsymbol{\theta}}^{n}) \geq \mathcal{L}_{\text{non-peak region}}(\hat{\boldsymbol{\theta}}^{n+1}). \tag{2.16}$$

However, experimentally I found that the total likelihood $\mathcal{L}(\boldsymbol{\theta})$ still increases when expanding the list of estimated F0s:

$$\mathcal{L}(\hat{\boldsymbol{\theta}}^{n}) \leq \mathcal{L}(\hat{\boldsymbol{\theta}}^{n+1}). \tag{2.17}$$

Another method to control the overfitting is needed. I first tried using a Bayesian Information Criterion, as in [**34**], but found that it did not work very well. Instead, I developed a simple threshold-based method to estimate the polyphony N:

$$\begin{aligned} N &= \min_{1 \leq n \leq M} n, \\ s.t. \quad & \Delta(n) \geq T \cdot \Delta(M), \end{aligned} \tag{2.18}$$

where $\Delta(n) = \ln \mathcal{L}(\hat{\boldsymbol{\theta}}^{n}) - \ln \mathcal{L}(\hat{\boldsymbol{\theta}}^{1})$; M is the maximum allowed polyphony; T is a learned threshold. For all experiments in this paper, the maximum polyphony M is set to 9. T is empirically determined to be 0.88. The method returns the minimum polyphony n that has a value $\Delta(n)$ exceeding the threshold. Figure 2.5 illustrates the method. Note that Klapuri [**70**] adopts a similar idea in polyphony estimation, although the thresholds are applied to different functions.

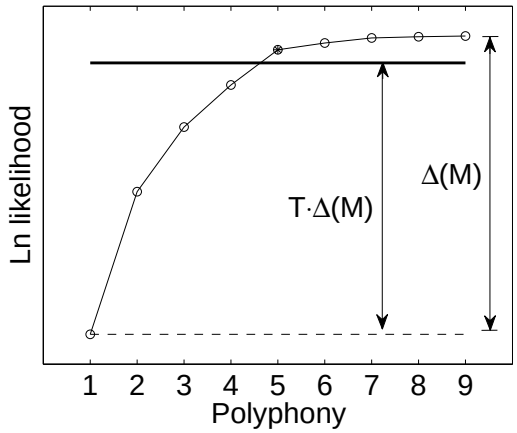


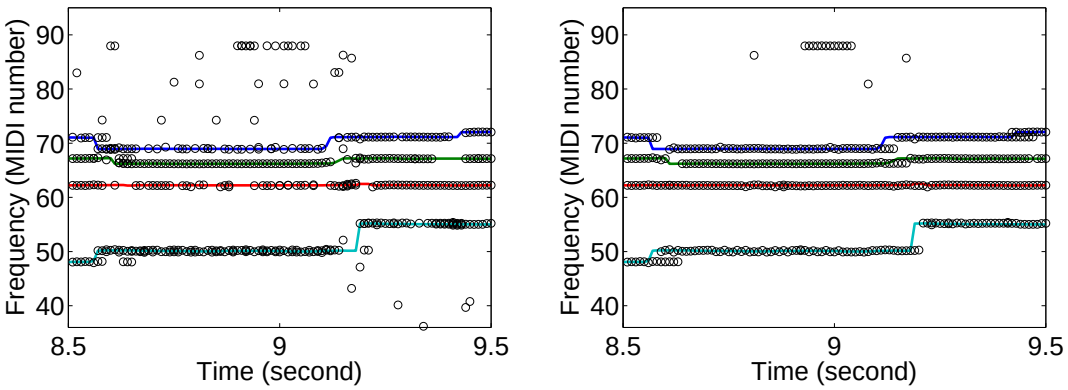
Figure 2.5. Illustration of polyphony estimation. Log likelihoods given each polyphony are depicted by circles. The solid horizontal line is the adaptive threshold. For this sound example, the method correctly estimates the polyphony, which is 5, marked with an asterisk.

2.4. Postprocessing Using Neighboring Frames

F0 and polyphony estimation in a single frame is not robust. There are often insertion, deletion and substitution errors, see Figure 2.6(a). Since pitches of music signals are locally (on the order of 100 ms) stable, it is reasonable to use F0 estimates from neighboring frames to refine F0 estimates in the current frame. In this section, I propose a refinement method with two steps: remove likely errors and reconstruct estimates.

Step 1: Remove F0 estimates inconsistent with their neighbors.

To do this, I build a weighted histogram W in the frequency domain for each time frame t. There are 60 bins in W, corresponding to the 60 semitones from C2 to B6. Then, a triangular weighting function in the time domain w centered at time t is imposed on a neighborhood of t, whose radius is R frames. Each element of W is calculated as the weighted frequency of occurrence of a quantized (rounded to the nearest semitone) F0 estimate. If the true polyphony N is known, the N bins of W with largest histogram



(a) Before refinement

(b) After refinement

Figure 2.6. F0 estimation results before and after refinement. In both figures, lines illustrate the ground-truth F0s, circles are the F0 estimates.

values are selected to form a refined list. Otherwise, I use the weighted average of the polyphony estimates in this neighborhood as the refined polyphony estimate N, and then form the refined list.

Step 2: Reconstruct the non-quantized F0 values.

I update the F0 estimates for frame t as follows. Create one F0 value for each histogram bin in the refined list. For each bin, if an original F0 estimate (unquantized) for frame t falls in that bin, simply use that value, since it was probably estimated correctly. If no original F0 estimate for frame t falls in the bin, use the weighted average of the original F0 estimates in its neighborhood in this bin.

In this work, R is set to 9 frames (90ms with 10ms frame hop). This value is not optimized. Figure 2.6 shows an example with the ground truth F0s and F0 estimates before and after this refinement. It can be seen that a number of insertion and deletion errors are removed, making the estimates more "continuous". However, consistent errors,

such as the circles in the top middle part of Figure 2.6(a), cannot be removed using this method.

It is noted that a side effect of the refinement is the removal of duplicate F0 estimates (multiple estimates within a histogram bin). This will improve precision if there are no unisons between sources in the data set, and will decrease recall if there are.

2.5. Computational Complexity

I analyze the run-time complexity of Algorithm 1 in terms of the number of observed peaks K and the maximum allowed polyphony M. We can ignore the harmonic number upper bound H and the number of neighboring frames R, because both these variables are bounded by fixed values.

The time of Line 2 through 4 is bounded by a constant value. Line 5 is a loop over Line 6 through 9 with M iterations. Line 6 through 8 involves $|\mathcal{C}| = O(K)$ likelihood calculations of Eq. (2.2). Each one consists of the peak region and the non-peak region likelihood calculation. The former costs $O(K)$, since it is decomposed into K individual peak likelihoods in Eq. (2.4) and each involves constant-time operations. The latter costs $O(M)$, since we consider MH harmonics in Eq. (2.14). Line 11 involves $O(M)$ operations to decide the polyphony. Line 12 is a constant-time operation. Line 14 through 16 involve $O(M)$ operations. Thus, total run-time complexity in each single frame is $O(MK^2 + M^2K))$. If M is fixed to a small number, the run-time can be said to be $O(K^2)$.

If the greedy search strategy is replaced by the brute force search strategy, that is, to enumerate all the possible F0 candidate combinations, then Line 5 through 10 would cost $O(2^K)$. Thus, the greedy approach saves considerable time.

Note that each likelihood calculation for Eq. (2.2) costs $O(K + M)$. This is a significant advantage compared with Thornburg and Leistikow's monophonic F0 estimation method [**119**]. In their method, to calculate the likelihood of a F0 hypothesis, they enumerate all associations between the observed peaks and the underlying true harmonics. The enumeration number is shown to be exponential in $K + H$. Although a MCMC approximation for the enumeration is used, the computational cost is still much heavier than ours.

2.6. Experiments

2.6.1. Datasets

The monophonic training data are *monophonic note recordings*, selected from the University of Iowa website[4]. In total 508 note samples from 16 instruments, including wind (flute), reed (clarinet, oboe, saxophone), brass (trumpet, horn, trombone, tuba) and arco string (violin, viola, bass) instruments were selected. They were all of dynamic "mf" and "ff" with pitches ranging from C2 (65Hz, MIDI number 36) to B6 (1976Hz, MIDI number 95). Some samples had vibrato. The polyphonic training data are *randomly mixed chords*, generated by combining these monophonic note recordings. In total 3000 chords, 500 of each polyphony from 1 to 6 were generated.

[4]http://theremin.music.uiowa.edu/

Chords were generated by first randomly allocating pitches without duplicates, then randomly assigning note samples of those pitches. Different pitches might come from the same instrument. These note samples were normalized to have the same Root-Mean-Squared (RMS) amplitude, and then mixed to generate chords. In training, each note/chord was broken into frames with length of 93 ms and overlap of 46 ms. A Short Time Fourier Transform (STFT) with four times zero padding was employed on each frame. All the frames were used to learn model parameters.

The polyphony estimation algorithm was tested using 6000 *musical chords*, 1000 of each polyphony from 1 to 6. They were generated using another 1086 monophonic notes from the Iowa data set. These were of the same instruments, pitch ranges, etc. as the training notes, but were not used to generate the training chords. Musical chords of polyphony 2, 3 and 4 were generated from commonly used note intervals. Triads were major, minor, augmented and diminished. Seventh chords were major, minor, dominant, diminished and half-diminished. Musical chords of polyphony 5 and 6 were all seventh chords, so there were always octave relations in each chord.

The proposed multiple F0 estimation method was tested on the Bach10 dataset[5]. It consists of 10 real music performances, totalling 330 seconds of audio. Each performance was of a four-part Bach chorale, performed by a quartet of instruments: violin, clarinet, tenor saxophone and bassoon. Each musician's part was recorded in isolation while the musician listened to the others through headphones. In testing, each piece was broken into frames with length of 46 ms and a 10 ms hop between frame centers. All the frames

[5]Download at `http://cs.northwestern.edu/~zdu459/resource/Resources.html`.

were processed by the algorithm. We used a shorter frame duration on this data to adapt to fast notes in the Bach chorales. The sampling rate of all the data was 44.1kHz.

2.6.2. Error Measures

The ground-truth F0s of the testing pieces were estimated using YIN [**24**] on the single-instrument recordings prior to mixing recordings into four-part monaural recordings. The results of YIN were manually corrected where necessary.

The performance of our algorithm was evaluated using several error measures. In the *Predominant-F0 estimation* (Pre-F0) situation, only the first estimated F0 was evaluated [**53**]. It was defined to be correct if it deviated less than a quarter tone (3% in Hz) from any ground-truth F0. The estimation accuracy was calculated as the amount of correct predominant F0 estimates divided by the number of testing frames.

In the *Multi-F0 estimation* (Mul-F0) situation, all F0 estimates were evaluated. For each frame, the set of F0 estimates and the set of ground-truth F0s were each sorted in ascending order of frequency. For each F0 estimate starting from the lowest, the lowest-frequency ground-truth F0 from which it deviated less than a quarter tone was matched to the F0 estimate. If a match was found, the F0 estimate was defined to be correctly estimated, and the matched ground-truth F0 was removed from its set. This was repeated for every F0 estimate. After this process terminated, unassigned elements in either the estimate set or the ground-truth set were called errors. Given this, *Precision*, *Recall* and *Accuracy* were calculated as:

$$\text{Precision} = \frac{\#\text{cor}}{\#\text{est}}, \qquad \text{Recall} = \frac{\#\text{cor}}{\#\text{ref}}, \tag{2.19}$$

$$\text{Accuracy} = \frac{\#\text{cor}}{\#\text{est} + \#\text{ref} - \#\text{cor}}, \tag{2.20}$$

where #ref is the total number of ground truth F0s in testing frames, #est is the total number of estimated F0s, and #cor is the total number of correctly estimated F0s.

Octave errors are the most common errors in multiple F0 estimation. Here we calculate octave error rates as follows: After the matching process in Mul-F0, for each unmatched ground-truth F0, we try to match it with an unmatched F0 estimate after transcribing the estimate to higher or lower octave(s). *Lower-octave error rate* is calculated as the number of these newly matched F0 estimates after a higher octave(s) transcription, divided by the number of ground-truth F0s. *Higher-octave error rate* is calculated similarly.

For polyphony estimation, a Mean Square Error (MSE) measure is defined as:

$$\text{Polyphony-MSE} = \text{Mean}\left\{(P_{\text{est}} - P_{\text{ref}})^2\right\}, \tag{2.21}$$

where P_{est} and P_{ref} are the estimated and the true polyphony in each frame, respectively.

2.6.3. Comparison Methods

Since the proposed method is related to existing methods based on modeling spectral peaks, it would be reasonable to compare to these systems. I choose one of the best performed method in this category, proposed by Pertusa and Iñesta in [**93**] (denoted as "Pertusa08") as a comparison method. I do not compare with [**52, 83, 119**], as they are all single F0 estimation methods. I do not compare with [**77**] either, as its computational complexity makes it prohibitively time-consuming, as shown in Section 2.5. Besides Pertusa08, I also compare with a method proposed by Klapuri in [**70**] (denoted

as "Klapuri06"). Both Klapuri06 and Pertusa08 were in the top 3 in the "Multiple Fundamental Frequency Estimation & Tracking" task in the Music Information Retrieval Evaluation eXchange (MIREX) in 2007 and 2008 [6].

Klapuri06 works in an iterative fashion by estimating the most significant F0 from the spectrum of the current mixture and then removing its harmonics from the mixture spectrum. It also proposes a polyphony estimator to terminate the iteration. Pertusa08 selects a set of F0 candidates in each frame from spectral peaks and generates all their possible combinations. The best combination is chosen according to their harmonic amplitudes and a proposed spectral smoothness measure. The polyphony is estimated simultaneously with the F0s. For both reference methods, we use the authors' original source code and suggested settings in our comparison.

2.6.4. Multiple F0 Estimation Results

Results reported here are for the 330 seconds of audio from ten four-part Bach chorales described in Section 2.6.1. Our method and the reference methods are all evaluated once per second, in which there are 100 frames. Statistics are then calculated from the per-second measurements.

I first compare the estimation results of the three methods in each single frame without refinement using context information. Then I compare their results with refinement. For Klapuri06, which does not have a refinement step, I apply the proposed context-based refinement method (Section 2.4) to it. I think this is reasonable because our refinement method is quite general and not coupled with our single frame F0 estimation method.

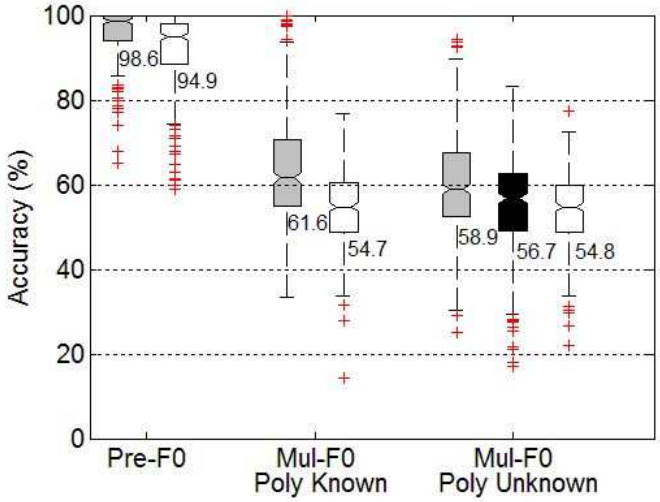
[6] http://www.music-ir.org/mirex/

Pertusa08, has its own refinement method using information across frames. Therefore, I use Pertusa08's own method. Since Pertusa08 estimates all F0s in a frame simultaneously, Pre-F0 is not a meaningful measure on this system. Also, Pertusa08's original program does not utilize the polyphony information if the true polyphony is provided, so Mul-F0 Poly Known is not evaluated for it.

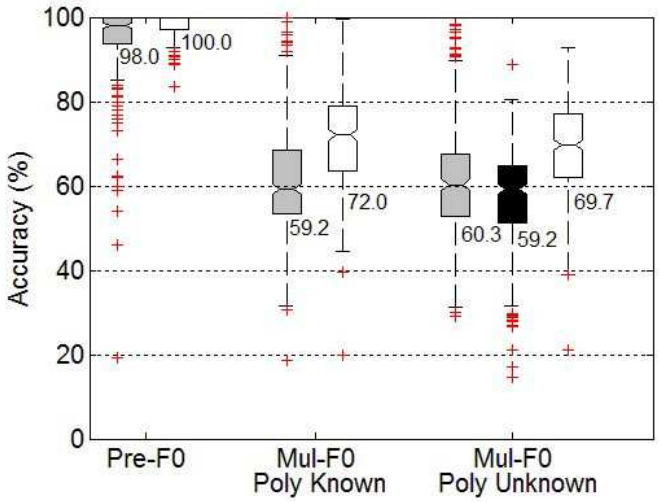
Figure 2.7 shows box plots of F0 estimation accuracy comparisons. Each box represents 330 data points. The lower and upper lines of each box show 25th and 75th percentiles of the sample. The line in the middle of each box is the sample median, which is also presented as the number below the box. The lines extending above and below each box show the extent of the rest of the samples, excluding outliers. Outliers are defined as points over 1.5 times the interquartile range from the sample median and are shown as crosses.

As expected, in both figures the Pre-F0 accuracies of both Klapuri06's and the proposed one are high, while the Mul-F0 accuracies are much lower. Before refinement, the results of the proposed system are worse than Klapuri06's and Pertusa08's. Take Mul-F0 Poly Unknown as an example, the median accuracy of our method is about 4% lower than Klapuri06's and 2% lower than Pertusa08's. This indicates that Klapuri06 and Pertusa08 both gets better single frame estimation results. A nonparametric sign test performed over all measured frames on the Mul-F0 Poly Unknown case shows that Klapuri06 and Pertusa08 obtains statistically superior results to our method with p-value $p < 10^{-9}$ and $p = 0.11$, respectively.

After the refinement, however, our results are improved significantly, while Klapuri06's results generally stay the same and Pertusa08's results are improved slightly. This makes



(a) Before refinement



(b) After refinement

Figure 2.7. F0 estimation accuracy comparisons of Klapuri06 (gray), Pertusa08 (black) and our method (white). In (b), Klapuri06 is refined with our refinement method and Pertusa08 is refined with its own method.

our results better than Klapuri06's and Pertusa08's. Take the Mul-F0 Poly Unknown example again, the median accuracy of our system is about 9% higher than Klapuri06's

Table 2.4. Mul-F0 estimation performance comparison, when the polyphony is not provided to the algorithm.

	Accuracy	Precision	Recall
Klapuri06	59.7±11.6	**86.1±9.6**	66.0±11.5
Pertusa08	57.3±11.4	84.6±13.5	63.7±9.6
Our method	**68.9±10.8**	82.7±8.1	**80.2±10.3**

and 10% higher than Pertusa08's. A nonparametric sign test shows that our results are superior to both reference methods with $p < 10^{-25}$.

Since I apply the proposed post-processing method on Klapuri06 and it removes inconsistent errors while strengthening consistent errors, I believe that the estimation errors in Klapuri06 are more consistent than the proposed method.

Remember that removing duplicates is a side effect of the proposed post-processing method. Since the proposed base method allows duplicate F0 estimates, but the data set rarely contains unisons between sources, removing duplicates accounts for about 5% of Mul-F0 accuracy improvement for the proposed method in both Poly Known and Unknown cases. Since Klapuri06 removes duplicate estimates as part of the approach, this is another reason the proposed refinement has less effect on Klapuri06.

Figure 2.7 shows a comparison of the proposed full system (white boxes in (b)) to the Kapuri06 as originally provided to us (gray boxes in (a)) and Pertusa08's system (black box in (b)). A nonparametric sign test on the Mul-F0 Poly Unknown case shows the proposed system's superior performance was statistically significant with $p < 10^{-28}$.

Table 2.4 details the performance comparisons of Mul-F0 Poly Unknown in the format of "Mean±Standard deviation" of all three systems. All systems had similar precision,

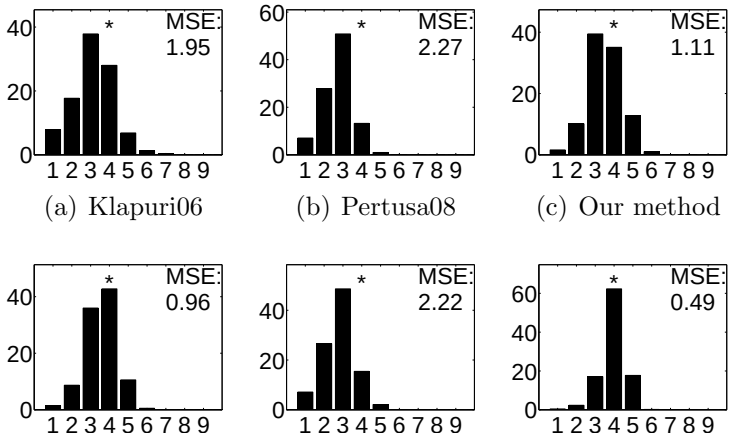
however Klapuri06 and Pertusa08 showed much lower accuracy and recall than the proposed system. This indicates both methods underestimate the number of F0s. This analysis is supported in the analysis of polyphony estimation.

2.6.5. Polyphony Estimation Results

Since polyphony estimation is a difficult task itself, I evaluated all three methods on this task. Among the 33,000 frames in Bach chorale test data, 29,687 had instruments sounding. Since all the pieces are quartets, and every instrument is active all along, the ground-truth polyphony is set to four for every frame with instruments sounding (I ignore the few frames that some performer ended or started a touch early).

Figure 2.8 shows the polyphony estimation histograms for all three methods without and with the refinement step. It can be seen that before refinement, all the methods tend to underestimate the polyphony. However, in both cases, our method obtains a better result with a lower MSE value than Klapuri06 and Pertusa08. Moreover, our refinement step improves the results for both Klapuri06 and our method, and finally our method obtains a symmetric histogram around the true polyphony as Figure 2.8 (f) shows.

In order to evaluate our polyphony estimation method (Section 2.3 without refinement) more comprehensively, I tested it on single frames of musical chords with different polyphony. Figure 2.9 shows the results. It can be seen that in most examples of polyphony from 1 to 4, the system outputs the correct polyphony. However, for polyphony 5 and 6, the polyphony estimation results are not satisfying. One of the reason is that F0s with octave relations are difficult to estimate using our algorithm. In our data set, chords of polyphony 1, 2, 3 and 4 do not have octave relations. Each chord of polyphony



(a) Klapuri06 (b) Pertusa08 (c) Our method

(d) Klapuri06 with our refinement (e) Pertusa08 with its own refinement (f) Our method with our refinement

Figure 2.8. Polyphony estimate histogram on the total 33,000 frames of the testing music pieces. X-axes represent polyphony. Y-axes represent the proportion of frames (%). The asterisk indicates the true polyphony.

5 contains a pair of pitches related by an octave. This means 40% of the pitches are in an octave relation. Each chord of polyphony 6 contains two pairs, giving 66.7% of the pitches in an octave relation. Thus, the tendency to underestimate their polyphony is not surprising.

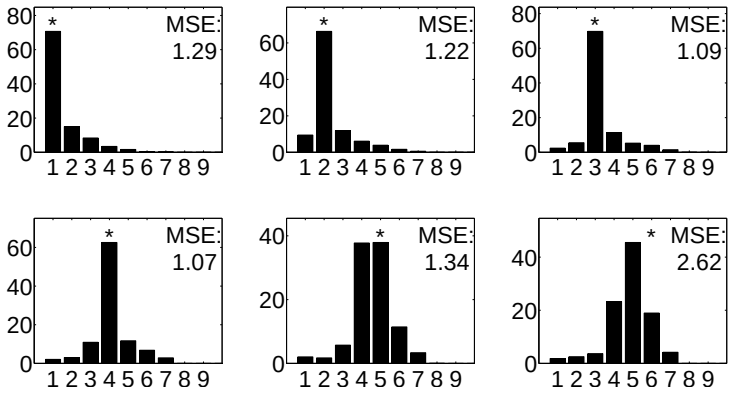


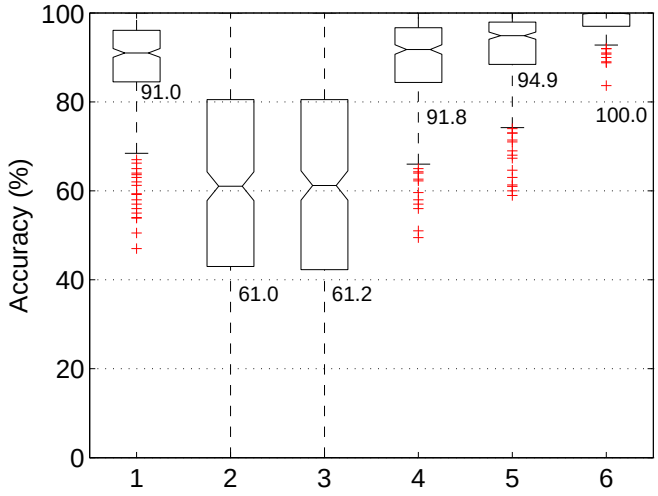
Figure 2.9. Polyphony estimation histogram of musical chords with polyphony from 1 to 6. X-axes represent polyphony. Y-axes represent the proportion of frames (%). The asterisk indicates the true polyphony.

2.6.6. Individual Analysis of System Components

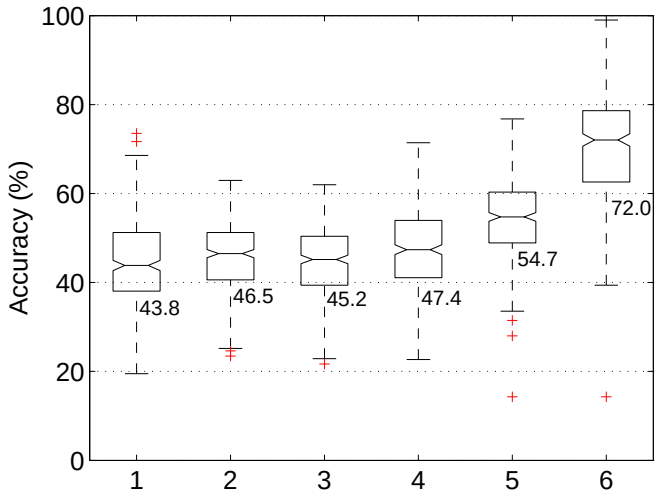
When a new approach is introduced, it may not always be clear which aspects of it contribute most strongly to its performance. I now investigate the effectiveness of different techniques that are used in our method: modeling peak frequencies and amplitudes, considering the possibility of spurious peaks, modeling the non-peak region, and refining F0 estimates using neighboring frames. In this experiment, we compare the F0 estimation accuracies with different system configurations:

- 1: models peak frequency deviations with a single Gaussian, as in [**52**].
- 2: models peak frequency deviations with a GMM model.
- 3: system 2 + models peak amplitudes with the non-parametric model in Eq. (2.12).
- 4: system 3 + considers the possibility of spurious peaks.
- 5: system 4 + models the non-peak region with Eq. (2.14).
- 6: system 5 + refines F0 estimates, as in Section 2.4.

Box plots of F0 estimation accuracies of these systems when the true polyphony is provided are illustrated in Figure 2.10. Again, each box represents 330 data points, corresponding to the 330 seconds of our testing pieces. For Pre-F0 results, systems except 2 and 3 are all higher than 90%. From System 2 to 3, the single Gaussian is replaced by a GMM to model the peak frequency deviation, which makes it possible to represent the tail of the distribution in Figure 2.2. Therefore, the frequency restriction of each F0 estimate is loosened, and the accuracy of the predominant F0 estimate is lower. However, after adding the spurious peak model in System 4 and the non-peak region model in System 5, more restrictions are added to F0 estimates and the accuracy is improved. Finally,



(a) Pre-F0



(b) Mul-F0

Figure 2.10. F0 estimation accuracy of different system configurations of our method, when the true polyphony is provided. The x-axes are system configuration numbers.

the F0 refinement technique improves the Pre-F0 median accuracy to 100.0%. It is noted that the predominant F0 estimate in a frame after the refinement may not be the same

predominant F0 estimate as before, instead, it is the best predominant F0 estimate in the neighborhood, hence is more robust.

For Mul-F0, the F0 estimation accuracy generally increases from System 1 to 6. There are three main improvements of the median accuracy: a 2.7% increase by replacing the single Gaussian with a GMM of modeling peak frequency deviations (System 1 to 2); a 7.3% increase by adding the non-peak region model (System 4 to 5); a 17.3% increase by F0 refinement (System 5 to 6). All of these improvements are statistically significant with $p < 10^{-8}$ in a nonparametric sign test. The only decrease occurs when adding the peak amplitude model (System 2 to 3). This indicates that the peak amplitude model parameters learned from randomly mixed chords are not suitable for the testing music pieces. In fact, when we train the peak likelihood parameters using 5 testing music pieces and test on all the 10 pieces, System 2 achieves 46.0% (0.5% lower), while System 3 achieves Mul-F0 accuracy median of 49.5% (4.3% higher). This indicates two things: First, the peak frequency deviation model is well learned from randomly mixed chords; Second, the peak amplitude (timbre information) modeling is helpful only if the training data are similar to the testing data. However, due to the timbral variety of music, this situation can be rare. This is in accordance with Klapuri's observation in [**69, 70**], where he employs a spectral whitening technique to remove timbre information of both training and testing signals.

As octave errors are the most frequency errors in multiple F0 estimation, Figure 2.11 shows the octave error rates of our systems. System 1 to 4 have much more lower-octave errors than higher-octave errors. This supports our claim that only modeling peaks will cause many lower octave errors. From System 4 to 5, lower-octave errors are significantly

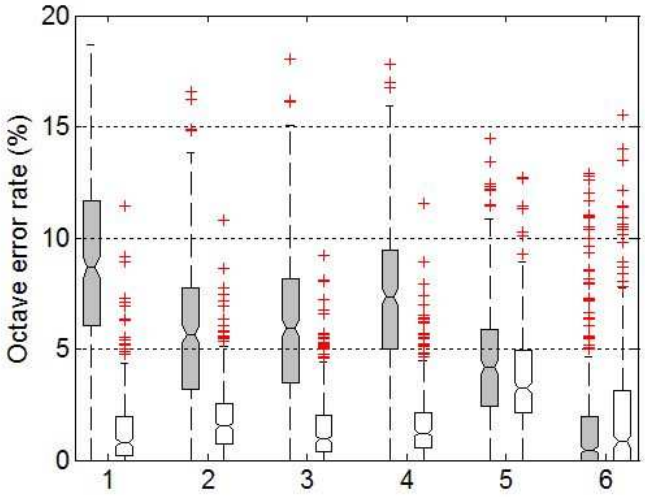


Figure 2.11. Octave error (gray: lower-octave error, white: higher-octave error) rates of different system configurations of our method, when the true polyphony is provided. The x-axis is the system configuration number.

reduced because of the non-peak region model, as they have a small non-peak region likelihood. Lower-octave and higher-octave errors are then approximately balanced. It is noted that this balance is achieved automatically by our probabilistic model, while it is achieved by manual assignment of the balancing factor ρ in [**83**]. Finally, both octave errors are significantly reduced by the refinement.

The proposed system was submitted to the "Multiple Fundamental Frequency Estimation & Tracking" task in MIREX 2009 and 2010. "DHP2" is the system we described in this paper and "DHP1" is the multi-pitch tracking system built based on "DHP2". Both systems obtained good results. The results can be accessed at `http://www.music-ir.org/mirex/wiki/2009:MIREX2009_Results` and `http://www.music-ir.org/mirex/wiki/2010:MIREX2010_Results`, respectively.

2.7. Conclusions

In this chapter, I proposed a maximum likelihood approach for multiple F0 estimation, where the power spectrum of a time frame is the observation and the F0s are the parameters to be estimated. The proposed method reduces the power spectrum into a peak region and a non-peak region, and the likelihood function is defined on both parts. The peak region likelihood is defined as the probability that a peak is detected in the spectrum given a set of F0s. The non-peak region likelihood is defined as the probability of not detecting any harmonics in the non-peak region. The two parts act as a complementary pair. To solve the combinatorial problem of simultaneously estimating F0s, I adopted an iterative estimation strategy to estimate F0s one by one. As expanding the number estimated F0s in each iteration, the total likelihood increases. I then proposed a polyphony estimation method by setting a threshold of the likelihood improvement. Finally, I proposed a refinement method to refine the F0 estimates using neighboring frames. This method removes a lot of inconsistent errors.

I tested the proposed approach on a corpus of 10 instrumental recordings of J. S. Bach quartets. The results show the proposed approach outperforms two state-of-the-art algorithms on this data set on both F0 estimation and polyphony estimation. The polyphony estimation method is also tested on 6,000 musical chords. Good results are obtained when there is no octave relation between pitches. It is noted that our system was trained using randomly mixed chords and monophonic notes, while tested on music pieces and musical chords. This indicates the generality of the proposed system.

The proposed multiple F0 estimation method not only works for music audio, it can be applied to sound mixtures composed of any kinds of harmonic sound sources, with

minimal changes. In Chapter 4 I will present some results on multi-talker speech data. The biggest change in adapting the proposed method to different data is in the training process. The maximum likelihood parameters need to be trained on the audio mixtures of the same kind of harmonic sources.

For sounds having octave-related pitches, the performance of the proposed method will deteriorate, due to the binary masking assumption adopted in the peak region likelihood definition. Since octaves are the most common intervals encountered in music, this problem should be addressed in future work. The current formulation limits the use of the method to harmonic sounds, but it should not be hard to extend it to quasi-harmonic sounds. The only change will occur in the calculation of the harmonic number of each peak.

Finally, the proposed method only estimates pitches in each individual frame. Although the post-processing module uses information from neighboring frames, it does not tell how the pitches are connected, i.e how the pitches of each source evolve. Connecting pitch estimates in individual frames into pitch trajectories across time frames is called Multi-pitch Streaming (MPS). This is an important intermediate step towards MASA of harmonic sound mixtures. In the next chapter, I will describe my work on MPS.

CHAPTER 3

Multi-pitch Streaming

3.1. Introduction

In Chapter 2 I described my work on Multi-pitch Estimation (MPE), which is the first level of the multi-pitch analysis problem. In this chapter, I propose an approach to address the third level, Multi-pitch Streaming (MPS). This approach requires three inputs: the original audio mixture, the estimated pitches at every time frame from an existing MPE algorithm, and the number of sources. This approach assumes monophonic and harmonic sound sources and streams pitch estimates into multiple pitch trajectories, each of which corresponds to an underlying source.

3.1.1. Related Work

Few perform multi-pitch analysis at the streaming level. Kashino and Murase [**67**] proposed a Bayesian network approach to integrate musicological and timbre information to stream pitches of multiple concurrent monophonic musical instruments. However, this method requires ground-truth notes (with both pitch and time information) as inputs. It has not been tested in more realistic scenarios where the inputs are estimated pitches at the frame level.

Vincent [**123**] proposed a three-layer (state, source, and mixture) Bayesian network to estimate the pitches and separate the signals of musical instruments in a stereo recording.

The approximate azimuths of the instruments are required as input. The parameters of the network need to be pre-learned from solo or mixture recordings of these instruments.

Bay *et al.* [**5**] proposed to estimate and stream pitches of polyphonic music using a probabilistic latent component analysis framework. This method also needs to pre-learn a spectral dictionary for each pitch of each instrument present in the mixture, from their isolated recordings.

Wohlmayr *et al.* [**136**] proposed a factorial hidden Markov model to estimate and stream pitches of two simultaneous talkers. The model parameters need to be trained for the talkers present in the mixture using their isolated recordings. These supervised methods prevent their usage in many scenarios when prior training on specific sources is unavailable.

Recently, Hu and Wang [**61**] proposed an unsupervised approach to estimate and stream pitches, and separate their signals of two simultaneous talkers. However, this approach was proposed only for speech and has not been tested for other kinds of audio data such as music.

In psychoacoustics, *sequential grouping* refers to the human auditory scene analysis process that streams auditory scene segments into meaningful auditory events [**8**]. Multi-pitch streaming can be viewed as a special kind of sequential grouping process, where the auditory scene segments are pitches and the meaningful auditory events are pitch trajectories of sound sources. A related concept is *simultaneous grouping*, which refers to the process of grouping simultaneous time-frequency elements into meaningful auditory events. MPE can be viewed as a kind of simultaneous grouping process.

The proposed approach (MPE + streaming) to address the third-level multi-pitch analysis problem lies in the framework of performing simultaneous grouping and sequential grouping in a sequence. This framework is feed-forward and does not use information from the streaming level to inform an existing MPE module. This would not be optimal, as the interplay between simultaneous grouping and sequential grouping is ubiquitous in human auditory scene analysis [**8**]. In addition, errors generated in the MPE stage may cause additional errors in the streaming stage. A wrong pitch estimate that is streamed to a source will prevent another correct pitch estimate in the same frame being streamed to that source.

For example, suppose there are two instruments. The first instrument plays two notes C4-D4 in a sequence. The second instrument plays G4-C5 in a sequence, and the two instruments switch notes at the same time. If the MPE stage wrongly estimates the latter part of the C4 note as C5, then these pitch estimates will probably be streamed with the C5 note of the second instrument through must-links. This error will in turn cause the G4 note of the second instrument to be excluded from the second instrument, due to its cannot-links to the latter part of the wrongly estimated and streamed C4 note.

However, the simplicity and clarity of our modular design let me build on existing work in MPE and independently optimize different levels of the system, whereas jointly determining pitch candidates and their streams may require a very complicated model and be computational intractable.

An alternate way to combine sequential and simultaneous grouping is to first do sequential grouping (partial tracking) then do simultaneous grouping (grouping partials into sources). In the literature, partial tracking is addressed by assuming the value continuity

[**84**] or the slope continuity [**25**] of the frequencies and amplitudes of partials. Therefore, a tracked partial would not be longer than a note or a syllable, and the "birth" and "death" of partials need to be addressed. In [**73**], a clustering approach based on frequency and amplitude continuity is proposed to track partials and group them into sources simultaneously, however, it still cannot group non-continuous partials since no timbre information is used.

3.1.2. Advances of the Proposed Method

I formulate the MPS problem as a constrained clustering problem, where the clustering objective is to maintain timbre consistency and the constraints are based on the relationships between pitch estimates in time and frequency. This work has been published in [**29**]. Compared to existing methods, this approach has the following advances:

- *Unsupervised.* It does not require training source models using isolated recordings of the underlying sources.
- *General.* It can deal with both music and speech, whereas existing approaches deal with either music or speech.
- *Compatible.* It can work with any MPE algorithm.

Table 3.1 summarizes the comparison of the proposed approach with existing approaches.

As a side product, I also propose a new cepstrum feature called the Uniform Discrete Cepstrum (UDC) to represent timbre of sound sources. It can be calculated from a number of points of the spectrum instead of the full spectrum, which makes it more suitable for representing timbre in multi-source mixtures than standard cepstral representations such

Table 3.1. Comparison of the proposed method with existing multi-pitch streaming methods.

	[67]	[123]	[5]	[136]	[61]	Proposed
Works on music data	✓	✓	✓			✓
Works on speech data				✓	✓	✓
Does not require pre-training source models	✓	✓			✓	✓
Tested on a large dataset			✓	✓	✓	✓

as Mel-frequency Cepstral Coefficients (MFCC). Multi-pitch streaming results using UDC outperform those using MFCC and another timbre feature.

3.2. Streaming as Constrained Clustering

I formulate the streaming problem as a constrained clustering problem, where the system takes three inputs: the original audio mixture, the set of instantaneous pitch estimates provided at each time frame by an existing multi-pitch estimation system, and the number of sources. The clustering objective is to maintain timbre consistency, based on the assumption that sound objects coming from the same source have similar timbre. Must-link constraints are imposed between pitches that are close in both time and frequency, to encourage them to be clustered into the same trajectory. I impose cannot-link constraints between pitches at the same time frame, to prevent them being assigned to the same source. We propose a novel algorithm to solve this constrained clustering problem.

3.2.1. Streaming Pitches by Clustering

I assume an audio mixture containing K monophonic sound sources. For each time frame we assume we have the output of a multi-pitch estimator that provides at most K concurrent pitch estimates. I associate the ith pitch estimate with a timbre represented as an n-dimensional vector $\mathbf{t}_i$.

I view the multi-pitch streaming problem as a pitch clustering problem, where each cluster is a pitch stream corresponding to a source. The clustering objective is defined as minimizing the total within-stream distance of the timbres of the pitch estimates:

$$f(\Pi) = \sum_{k=1}^{K} \sum_{\mathbf{t}_i \in S_k} \|\mathbf{t}_i - \mathbf{c}_k\|^2. \tag{3.1}$$

Here, Π is a partition of the pitch estimates into K streams; $\mathbf{t}_i$ is the timbre feature vector of pitch i; $\mathbf{c}_k$ is the centroid of timbres in stream S_k; and $\|\cdot\|$ denotes the Euclidean norm. This is the same as the K-means clustering objective.

To justify the clustering objective function, we note that humans use timbre to discriminate and track sound sources [**8**]. Given an appropriate timbre feature, I expect that a note (vowel) has more similar timbre to another note (vowel) produced by the same instrument (talker), than to that produced by a different instrument (talker). Different choices of timbre vectors can be found in Section 3.4.

3.2.2. Adding Locality Constraints

With an appropriate timbre feature (see Section 3.4 for the timbre features used in this work), we can cluster pitch estimates to minimize the intra-cluster timbre inconsistency in Eq. (3.1), using the K-means algorithm. However, it is not enough to provide satisfying pitch streaming results, as shown in Figure 3.1.

In the middle panel of Figure 3.1, a number of pitches are clustered into the wrong trajectory. For example, the pitches around MIDI number 55 from 14.8 sec to 15.8 sec form a continuous contour and are all played by the bassoon. However, in the resulted clustering, some of them are assigned to saxophone. In another example, from 16.8 sec

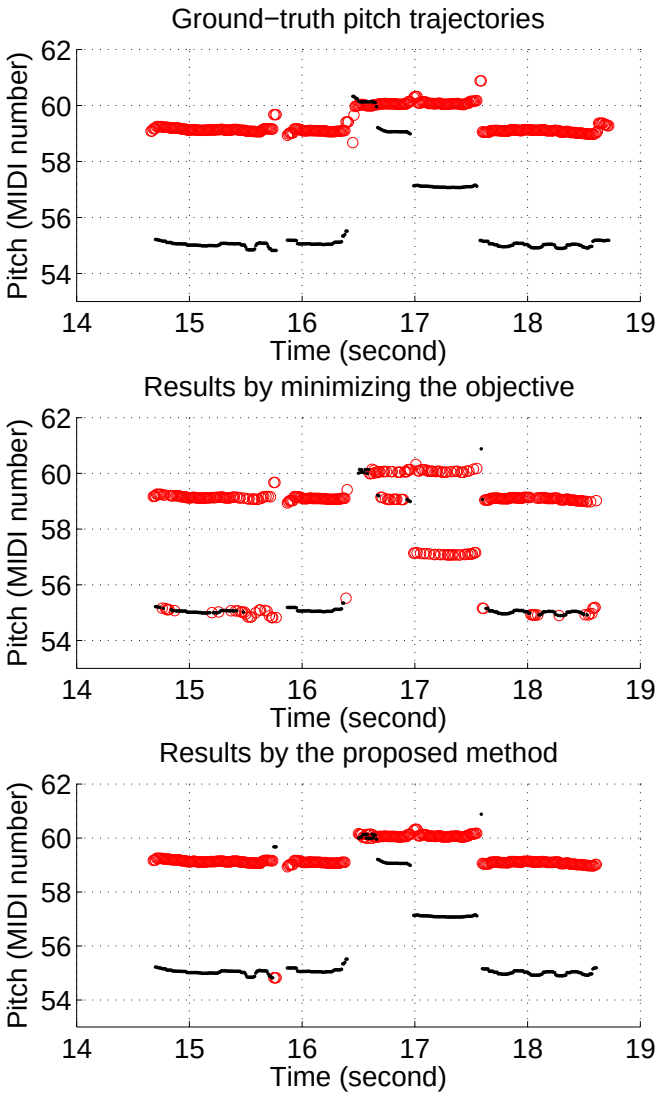


Figure 3.1. Comparison of the ground-truth pitch streams, K-means clustering ($K = 2$) results (i.e. only minimizing the objective function), and the proposed method's results (i.e. considering both objective and constraints). Both the K-means and the proposed method take the ground-truth pitches as inputs, use 50-d harmonic structure from Section 3.4 as the timbre feature, and randomly initialize their clusterings. Each point in these figures is a pitch. Different instruments are marked with different markers (circles for saxophone and dots for bassoon).

to 17.6 sec, the K-means clustering puts two simultaneous pitches into the saxophone stream. This is not reasonable, since saxophone is a monophonic instrument.

If we know that different sources do not often perform the same pitch at the same time and all sources are monophonic, we can impose two kinds of constraints on some pairs of the pitches to improve clustering: A *must-link* constraint is imposed between two pitches that differ less than Δ_t in time and Δ_f in frequency. It specifies that two pitches close in both time and frequency should be assigned to the same cluster. A *cannot-link* constraint is imposed between two pitches in the same frame. It specifies that two simultaneous pitches should be assigned to different clusters. These must-links and cannot-links form the set of all constraints C. The bottom panel of Figure 3.1 shows the result obtained from our proposed algorithm, considering both the objective and constraints.

3.2.3. Constrained Clustering and Its Properties

Given the clustering objective and constraints, the multi-pitch streaming problem becomes a constrained clustering problem with binary constraints. In seeking a good clustering, the objective function (within-stream timbre inconsistency) should be minimized while the constraints (assumptions about pitch relationship) should be satisfied.

There exist a number of constrained clustering algorithms [**131, 132, 21**] that deal with binary constraints, however, they cannot be applied due to the problem's unique properties:

- *Inconsistent Constraints:* Constraints are imposed on pitch estimates which contain errors, hence the constraints themselves also contain errors. Also, the assumptions that the constraints are based on are not always correct. Two sources

may occasionally perform the same pitch, and two pitches produced by the same monophonic source may be concurrent due to room reverberation. Therefore, the constraints may not be consistent with each other.

- *Heavily Constrained:* Since the pitch of each source often evolves smoothly over short periods (several frames), almost every pitch estimate is involved in some must-links. Also, since most of the time there are multiple sound sources playing simultaneously, almost every pitch estimate is involved in some cannot-links. This makes the clustering problem heavily constrained.

Because of the "Inconsistent Constraints" property, there may not exist any clustering satisfying all the constraints. This makes existing algorithms [**131, 132**] inapplicable, since they attempt to find a clustering minimizing the objective while satisfying all the constraints. Even if we assume all constraints are consistent, [**21**] proved that finding a feasible solution, i.e. a label assignment without violating any constraint, of a clustering problem containing cannot-links is NP-complete.

Therefore, we should not try to satisfy all the constraints. Instead, I seek an algorithm that minimizes the objective while satisfying as many constraints as possible. An *Incremental Constrained Clustering* algorithm [**21**] fits this purpose. It starts from an initial clustering of the data Π_0 that satisfies a subset of all the constraints $C_0 \subset C$ and then incrementally adds constraints in following iterations. However, I will show that [**21**] is inapplicable to our problem in Section 3.3.1. Thus, I need to design a new incremental constrained clustering algorithm for the MPS problem.

3.3. Algorithm

In this work, a point p is a pitch estimate with an associated fundamental frequency, time, and timbre. A partition Π is an assignment of each pitch estimate to exactly one of K streams (clusters). This is also referred to as a clustering. The objective function $f(\Pi)$ returns the total within-stream timbre inconsistency, as described in Eq. (3.1).

In this section I describe a novel incremental constrained clustering algorithm. It starts from an initial partition Π_0 that satisfies a subset of all the constraints $C_0 \subset C$. Then it iteratively minimizes the objective function while incrementally satisfying more constraints. Note that, although I apply it to the streaming problem, the algorithm is more general than that and may be applied to any problem of set partitioning under constraints with an objective function.

3.3.1. Forming the Initial Partition

For a general incremental constrained clustering problem, the initial partition Π_0 can be simply set by a random label assignment of all the instances. For our multi-pitch streaming problem, we can have a more meaningful initialization: I set Π_0 by sorting pitches in each frame from high to low and assigning labels from 1 to K. This is possible because, if there are K monophonic sound sources, there are at most K pitches in each frame. We call this *pitch-order initialization.*

For many audio mixtures, including much polyphonic music and two-talker speech of different genders, pitch-order initialization is more informative than random initialization. This is because pitch streams do not often interweave in these cases. Nevertheless, pitch-order initialization does not solve the streaming problem even in these cases. This is

because the algorithm takes pitch estimates as inputs, which contain many polyphony and pitch errors, and the ordering will be messed up. In the experiments, I will compare the effects of different initializations.

For pitch-order initialization Π_0, its satisfied constraints C_0 contains all cannot-links in C. This is because cannot-links are only imposed on concurrent pitches, which are assigned to different clusters (streams) in Π_0.

Given Π_0 and C_0, we want to minimize the objective function while incrementally adding constraints. Davidson et al. [**21**] showed that incrementally adding new constraints is NP-hard in general, but they identified several sufficient conditions under which the clustering could be efficiently updated to satisfy the new and old constraints. The conditions require either 1) at least one point involved in the new constraint is not currently involved in any old constraint or 2) the new constraint is a cannot-link.

For our problem, however, from the initial constraints C_0, neither of the two conditions can be met. This is because: 1) Due to the "Heavily Constrained" property, almost every pitch estimate has already been constrained by some cannot-links, so Condition 1 is not met. 2) Since all the cannot-links are already in C_0, any new constraint will be a must-link, so Condition 2 is not met. Therefore, the algorithm in [**21**] will do nothing beyond the pitch-order initialization.

3.3.2. A Novel Incremental Constrained Clustering Algorithm

Here I describe a new incremental constrained clustering algorithm (see Algorithm 2) that alternately updates the partition and set of satisfied constraints, starting from initial partition Π_0 and satisfied constraints C_0.

Suppose we are in the t-th iteration, where the previous partition is Π_{t-1} and the set of constraints that it satisfies is C_{t-1}. We first update Π_{t-1} to a new partition Π_t which strictly decreases the objective function *and* also satisfies C_{t-1} (Line 4). We then find which (if any) constraints that Π_t satisfies, which were not satisfied by Π_{t-1}. We add those constraints to the set of satisfied constraints, giving us C_t (Line 5). So we have $f(\Pi_{t-1}) > f(\Pi_t)$ and $C_{t-1} \subseteq C_t$. Although in some iterations Π_t does not satisfy more constraints than Π_{t-1} and $C_{t-1} = C_t$, in general the set of satisfied constraints will expand. The key of this algorithm is Line 4, and will be explained in Section 3.3.3 and Algorithm 3. If no new partition is returned in Line 4, Algorithm 2 will terminate. I will show that it always terminates in Section 3.3.6.

Algorithm 2: IncrementalClustering

Input : N points to be partitioned into K clusters; f: the objective function to be minimized; C: the set of all constraints; Π_0: initial partition; $C_0 \subseteq C$: constraints satisfied by Π_0.

Output: A partition Π_t and constraints it satisfies, C_t.

1 $t \leftarrow 0$;
2 **do**
3 $\quad t \leftarrow t + 1$;
4 $\quad \Pi_t =$ `FindNewPartition(`Π_{t-1},C_{t-1},f`)`;
5 $\quad C_t =$ The set of constraints satisfied by Π_t;
6 **while** $\Pi_t \neq \Pi_{t-1}$;
7 **return** Π_t and C_t;

3.3.3. Find A New Partition by Swapping Labels

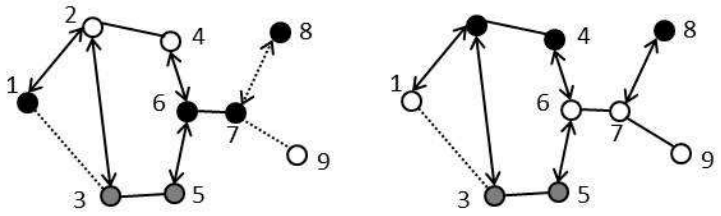
In Line 4 of Algorithm 2, we want to update Π_{t-1} to a new partition Π_t that strictly decreases the objective function and also satisfies the constraints in C_{t-1}. I do this by moving at least one point between streams in Π_{t-1}. However, if we move some point p

(recall points are pitch estimates) from cluster S_k to cluster S_l (recall clusters are streams), all the points that have a must-link to p according to C_{t-1} should be moved from S_k to S_l, because we want C_{t-1} to be satisfied by the new partition as well. Then all the points in S_l that have cannot-links to any of the above-mentioned points need also be moved out of S_l. If they are moved to another stream S_m, then the points in S_m that have cannot-links with the above-mentioned points in S_l according to C_{t-1} need to be moved, and this will cause a chain reaction.

I deal with this issue by defining the *swap set* of points that may be affected by changing the stream of p from S_k to S_l. Then I will define the *swap* operation to change the cluster label for all points in the swap set without breaking any currently-satisfied constraints in C_{t-1}.

Given a node p and two streams S_k and S_l, the swap set is the set of points from these clusters that have a path to p through the currently satisfied constraints in C_{t-1}, subject to that the path only involves points from streams S_k and S_l. Note that a currently satisfied constraint involving points from other streams is not an edge here. In other words, the swap set is the maximally connected subgraph containing p between streams S_k and S_l.

Consider the left panel of Figure 3.2. Suppose we want to move point 6 in the left panel from black to white. The swap set for point 6 in a black-white swap is the set of points 1, 2, 4, 6 and 7. They form the maximally connected graph containing the point 6 between the two clusters, using the currently satisfied constraints as edges. The swap set for point 6 in a black-gray swap is points 3, 5, 6, 7.



(a) before swap (b) after swap

Figure 3.2. An illustration of the swap operation. Here we have 9 points from 3 streams (white, gray and black). Must-links are depicted as lines without arrows, and cannot-links are lines with arrows. Constraints satisfied by the current partition are in solid lines, and those not satisfied are in dotted lines.

In our implementation, for each point we use two lists to store the currently satisfied must-links and cannot-links that involve the point. To find a swap set between S_l and S_k, we start from a point in S_l and first expand the swap set by incorporating the points that have must-links to it. We then expand the swap set by incorporating the points that are in S_k and have cannot-links to all the points in the current swap set. We then expand their must-links, then cannot-links, etc., until the swap set does not expand anymore.

The *swap* operation involves flipping the cluster for all points in the swap set. Those formerly in S_l move to S_k. Those formerly in S_k move to S_l. Figure 3.2 illustrates a white-black swap. Here, we swap these five points and get a new partition shown in the right panel. The new partition satisfies all the constraints that were satisfied before, but it also satisfies two more constraints in this example, i.e. the cannot-link between point 7 and 8, and the must-link between point 7 and 9.

3.3.4. Proof Constraints are Preserved by a Swap

The swap operation is guaranteed to preserve all currently-satisfied constraints. Proof:

Split the constraints satisfied prior to swap into those between points within the swap set, and those involving points outside the swap set. First consider the within-swap-set constraints. All satisfied must-links between points in the swap-set remain satisfied after a swap. This is true because all points in the swap set that share a cluster prior to the swap will share a cluster after the swap. Similarly, all cannot-links between points in the swap-set remain satisfied, since all points which are not in the same cluster are still not in the same cluster after the swap.

Now we address currently satisfied constraints involving points outside the swap set. Any of these constraints must be a cannot-link, and the outside point involved in this constraint must be in a third stream different from the streams that define the swap set. This is because otherwise the outside point would be in the swap set, according to the swap set definition. Since the swap operation never assigns the cluster label of the third stream to any point in the swap set, this cannot-link remains satisfied. Consider point 3 in Figure 3.2 as an illustrative example.

3.3.5. Finding a New Partition

The swap operation assures the set of satisfied constraints is expanded (or remained the same), but it does not say anything about the objective function. It is possible that the objective function is not decreased after the swap. In other words, the swap operation gives us a way to generate a new and valid partition, but the partition might not be better than the current one, given the objective function.

To make sure the objective function is also strictly decreased, I only do a swap operation that does strictly decrease the objective function. To find such a swap operation,

Algorithm 3: FindNewPartition

Input : Π_{t-1}: a K-partition of N points; C_{t-1}: constraints satisfied by Π_{t-1}; f: objective function to be minimized.

Output: Π_t: A new K-partition that also satisfies C_{t-1} and with $f(\Pi_t) \leq f(\Pi_{t-1})$.

```
f_best ← f(Π_{t-1});
Π_t ← Π_{t-1};
while f_best == f(Π_{t-1}) && not all the points p_1, ··· , p_N are traversed do
    Pick p_n at random, without replacement. Suppose p_n is in stream S_k.;
    for l ← 1, ··· , K; l ≠ k do
        Find the swap set of p_n between S_k and S_l in Π_{t-1} according to C_{t-1}; Do
        swap to get a new clustering Π_s and its centroids.;
        if f(Π_s) < f_best then
            f_best ← f(Π_s);
            Π_t ← Π_s;
        end
    end
end
return Π_t;
```

I randomly traverse all the points and try all their swap operations (i.e. try changing streams for each pitch estimate). I stop the traversal when we find any swap operation that decreases the objective function and return the new partition after the swap. If I cannot find such a swap operation after traversing all the points, then there is no new partition that strictly decreases the objective function and also satisfies the currently satisfied constraints. In this case, I return the current partition and Algorithm 2 terminates. This subroutine is described in Algorithm 3.

Figure 3.3 illustrates the process of Algorithm 2 from the perspective of solution spaces. The algorithm starts with the initial constraints C_0 and partition Π_0, where the solution space under C_0 is S_0. Then it updates to a new partition Π_1 in S_0 which decreases the objective function f in Equation 3.1. After adding all the new constraints that Π_1

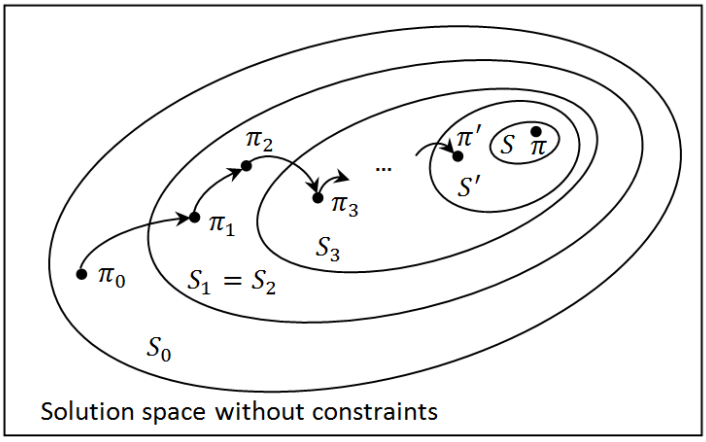


Figure 3.3. An illustration of Algorithm 2. Ellipses represent solution spaces under constraints in different iterations. Points represent clusterings. Arrows show how clusterings are updated to decrease the objective function.

satisfies, the set of satisfied constraints is expanded from C_0 to C_1, and the solution space is shrunk to S_1. Then, a new clustering Π_2 is updated in S_1, but this time there is no new constraint satisfied. Therefore, $C_2 = C_1$ and $S_2 = S_1$. This iteration terminates in Π' and C', where Π' is a local minimum of f in the solution space S' under C'. S is the solution space under all the constraints C, and Π is its optimal solution. It is noted that if the constraints are inconsistent, S will be empty.

3.3.6. Algorithm Analysis

Algorithm 2 always terminates, possibly to some local optimum, because the space of feasible partitions is finite and in every iteration the new partition found by "FindNew-Partition" strictly decreases the objective function. The only time that the objective function does not strictly decrease is when the newly found partition is actually the old one, which causes Algorithm 2 to terminate.

In each iteration of the proposed algorithm, the space of feasible partitions, given the satisfied constraints, is shrunk. Take the multi-pitch streaming problem as an example. Suppose there are K monophonic sources, T time frames. In the worst case the total number of pitches N equals to KT, then the size of the solution space without any constraint is K^{KT}. After imposing the initial constraints (all cannot-links) C_0, the space is shrunk to about $(K!)^T$. This is because, each time frame has $K!$ distinct assignments of K pitch estimates to K streams.

After imposing all the constraints C (assuming they are consistent), suppose the typical number of pitch estimates in a must-link group (a group of pitches connected by must-links) is M, then there are in total about KT/M must-link groups. Suppose also that each must-link group is involved in a K-clique with cannot-link edges (each note is overlapped by $K-1$ other notes, which can be common). Then the solution space is further reduced to $(K!)^{\frac{KT}{MK}} = (K!)^{T/M}$. A typical value of M is 20 (i.e. a must-link group spans 20 frames). With the constraints expanded, not only more domain knowledge is incorporated to refine the clustering, the shrunk space also eliminates a lot of local minima of the objective function, where Algorithm 2 can be trapped.

The worst case running time of each iteration of Algorithm 2 is $O(KN^2)$, in terms of the number of all points N and the number of clusters K. This is because in Algorithm 3, there are at most NK nested loops from Line 6 to Line 11. Line 6, 7 and 9 all cost $O(N)$ operations in the worst case (when the size of the swap set is $O(N)$). In most cases, however, the swap set is much smaller than N. Taking the multi-pitch streaming problem as an example, the size of a swap set typically does not increase with the length of the music and the number of sources. This is because breaks between notes (or words)

naturally bound the number of pitch estimates that must be considered in a swap set to a constant. In this case, each iteration of Algorithm 2 costs $O(KN)$.

How long then, does Algorithm 2 take in practice? In my experiments, a typical four-part Bach chorale (25 seconds long) from the Bach10 dataset in Section 3.5 has about 9,000 pitch estimates and 15,000 constraints. The algorithm takes about 300 iterations to terminate from pitch-order initialization. This requires about 11 minutes on one core of a 4-core 2.67GHz CPU). Assuming random initialization of the partition, the algorithm requires 2,800 iterations to terminate (43 minutes on the same computer). In practice, one can terminate the algorithm earlier, if the partition is already good enough.

3.4. Timbre Features

The constrained clustering approach described in this work depends on a clustering objective function which, in turn, depends on a timbre representation for the pitch estimates. While there are a number of approaches to representing timbre [**71, 9**], our problem formulation requires a simple approach that can be calculated from a multi-source mixture for pitch estimate in a single time frame, where time frames are on the order of 50 milliseconds in length. Here, I describe two previously-used timbre representations: harmonic structure and mel-frequency cepstral coefficients (MFCCs). I then propose a new representation: the uniform discrete cepstrum (UDC).

3.4.1. Harmonic Structure

This approach was previously used with success in [**35**]. It is defined as a vector of relative logarithmic amplitudes of the harmonics of a pitch estimate. The harmonics are at integer

multiples of the pitch. I use the first 50 harmonics to create the timbre vector $\mathbf{t}_i$. I choose this dimensionality because most instruments have less than 50 prominent harmonics. For each harmonic, I use the peak-finder from [**35**] to see if there is a significant peak within a musical quarter-tone. If no peak is associated, the magnitude of the harmonic is set to 0dB, else it is set to the value of the nearest peak. Then, the representation is normalized. This is a simple, clear baseline representation. Note that the assumptions here are that it will not be overly impacted by overlapping harmonics from different sources, and that the within-source variation in harmonic structure will be less than the between-source difference.

3.4.2. Mel-frequency Cepstral Coefficients (MFCC)

MFCCs have been widely used to represent the timbre of speech signals in many problems, including speech recognition, speaker identification, etc. To calculate an MFCC feature vector for an audio frame, the magnitude spectrum of the frame is first mapped onto the Mel-frequency scale to better approximate the frequency resolution of the human ear:

$$\mathbf{mel}(f) = \begin{cases} 3f/200 & \text{if } f \leq 1000\text{Hz} \\ 15 + \ln(f/1000)/0.0688 & \text{if } f > 1000\text{Hz} \end{cases} \tag{3.2}$$

Then, the typical steps used in creating an ordinary cepstrum (see Section 3.4.3) are applied. In this work, I use Dan Ellis's implementation [**38**], with a 40-band Mel filter bank.

To calculate the MFCC feature for an individual pitch estimate, we first need to separate its magnitude spectrum from the mixture. I do so using a simple harmonic

masking approach [**31**]. Recall that I assume a pitch estimate is associated with a single source. If there are K pitch estimates in the current time-frame, then each frequency bin in the spectrum is a harmonic of between 0 and K pitch estimates. Call this value the harmonic count, hc. For nonharmonic bins ($hc = 0$) the mixture energy is evenly distributed to all concurrent pitches. For a non-overlapping harmonic bin ($hc = 1$), the mixture energy is solely assigned to a single source. For an overlapping harmonic bin ($hc > 1$), the mixture energy is distributed among the pitch estimates it is a harmonic of. Here, the proportion of energy assigned to a pitch estimate decreases as the harmonic index increases. If the bin is the 10th harmonic of pitch p and the 2nd of pitch q, q will receive more energy. This distribution is in inverse proportion to the square of harmonic indices. It is equivalent to assuming that harmonic sources concentrate their energy in the lower partials, a reasonable assumption for many sources.

3.4.3. Uniform Discrete Cepstrum

I have described an approach to building a cepstral representation of the pitch timbre from a mixture: separate using a harmonic mask, then calculate the MFCCs. I now describe an alternate approach to calculating a cepstral representation only from points in the mixture spectrum that are likely to come from a single source, without the requirement of separation. I name this representation as Uniform Discrete Cepstrum (UDC).

Essentially, UDC is approximately equivalent to taking the discrete cosine transform (DCT) of a sparse log-amplitude spectrum. The sparse spectrum takes values of the mixture spectrum at the frequency bins that are likely to come from the source, and zeros everywhere else. For a harmonic source in this paper, these frequency bins correspond

to the harmonics of its pitch. Although the calculation of UDC is simple, its derivation and relation to other cepstral representations is not that apparent. I describe it in the following section.

3.4.4. Derivation of UDC

I first describe the basic concept of a cepstrum. I then describe the ordinary cepstrum and the discrete cepstrum proposed in [**47**], from which UDC is derived.

The concept of a cepstrum is to approximate (up to a scale) a log-amplitude spectrum $a(f)$ by a weighted sum of p sinusoids

$$a(f) \approx c_0 + \sqrt{2}\sum_{i=1}^{p-1} c_i \cos(2\pi i f), \tag{3.3}$$

where the weights $\mathbf{c} = [c_0, c_1, \cdots, c_{p-1}]^T$ form a cepstrum of order p; f is the normalized frequency (Hz divided by the sampling rate). A common approximation criterion is to minimize the Euclidean distance between both sides of Eq. (3.3), which leads to the least square solution.

The calculation of the *ordinary cepstrum (OC)* assumes that the log-amplitude spectrum $a(f)$ is observable at all frequency bins of a Fourier analysis. Suppose there are N bins and their normalized frequencies and log-amplitudes are $f_1, \cdots, f_N$ and $a_1, \cdots, a_N$, an ordinary cepstrum of order p is the first p coefficients of a DCT of the spectrum, equivalent to the least square solution of Eq. (3.3) from the whole spectrum:

$$\mathbf{c}_{\text{oc}} = (M^T M)^{-1} M^T \mathbf{a} = M^T \mathbf{a}, \tag{3.4}$$

where $\mathbf{a} = [a_1, \cdots, a_N]^T$ and

$$M = \begin{pmatrix} 1 & \sqrt{2}\cos(2\pi 1 f_1) & \cdots & \sqrt{2}\cos(2\pi(p-1) f_1) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \sqrt{2}\cos(2\pi 1 f_N) & \cdots & \sqrt{2}\cos(2\pi(p-1) f_N) \end{pmatrix}. \tag{3.5}$$

The second equality in Eq. (3.4) comes from the fact that the columns of M are orthogonal and $M^T M$ is an identity matrix. M contain the first p columns of a DCT matrix.

The calculation of the *discrete cepstrum*[1] [**47**], however, does not require observing all the frequency bins of the spectrum. It can be calculated from a sparse, possibly non-uniform, set of discrete spectral points. Suppose there are L observable frequencies $\hat{f}_1, \cdots, \hat{f}_L$, which form a subset of all the frequency bins $f_1, \cdots, f_N$[2]; and their corresponding spectral log-amplitudes are $\hat{a}_1, \cdots, \hat{a}_L$. Then the discrete cepstrum of order p is the least square solution of Eq. (3.3) at these observable frequencies[3]:

$$\mathbf{c}_{\mathrm{dc}} = (\hat{M}^T \hat{M})^{-1} \hat{M}^T \hat{\mathbf{a}}, \tag{3.6}$$

where $\hat{\mathbf{a}} = [\hat{a}_1, \cdots, \hat{a}_L]^T$ and

$$\hat{M} = \begin{pmatrix} 1 & \sqrt{2}\cos(2\pi 1 \hat{f}_1) & \cdots & \sqrt{2}\cos(2\pi(p-1) \hat{f}_1) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \sqrt{2}\cos(2\pi 1 \hat{f}_L) & \cdots & \sqrt{2}\cos(2\pi(p-1) \hat{f}_L) \end{pmatrix}. \tag{3.7}$$

[1] Its name is confusing since, "discrete" often refers to the implementation in the digital world, such as discrete cosine transform. Here, however, "discrete" refers to the fact that a discrete cepstrum can be calculated from a number of isolated analysis frequencies in a spectrum.

[2] In fact, the observable frequencies need not to be a subset of frequency bins in Fourier analysis. They can be frequencies in between the bins.

[3] Note there is a slight and indifferent difference between Eq. (3.3) and the formulation in [**47**] on the coefficients of sinusoids

Compared with $\mathbf{c}_{\text{oc}}$, $\mathbf{c}_{\text{dc}}$ has the advantage that it can be calculated from the mixture spectrum directly, from the spectral points that are likely to belong to the source. However, I found that $\mathbf{c}_{\text{dc}}$ calculated from different spectra of the same source are not similar to each other. This prevents it being used as a timbre feature of sources. In fact, $\mathbf{c}_{\text{dc}}$ was only used for the purpose of spectral envelope reconstruction when it was proposed in [**47**]. It was never used as a timbre feature for statistical comparisons. I explain this in the following.

For two spectra $\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$ of the same source, their spectral envelopes are often similar due to their similar timbre. Their spectra are two instantiations of their spectral envelopes. Therefore, their $\mathbf{c}_{\text{oc}}$'s are also similar as they are least square solutions to approximate their respective full spectra.

However, $\mathbf{c}_{\text{dc}}$ is the least square solution to only approximate the L observable frequencies, that is, the reconstructed spectral envelope from $\mathbf{c}_{\text{dc}}$ by Eq. (3.3) is very close to the original spectrum at these L frequencies, but can be arbitrary at other frequencies. The observable frequencies of the two spectra $\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$ are often quite different in their respective mixture spectra. This makes their $\mathbf{c}_{\text{dc}}$'s be quite different too, since if they were similar, their reconstructed spectral envelopes using Eq. (3.3) would also be similar at all frequencies. But this is unlikely, as the reconstructed spectral envelopes at the non-observable frequencies are arbitrary. There is no control at all for these values.

To address this problem, I define the *Universal Discrete Cepstrum (UDC)* as

$$\mathbf{c}_{\text{udc}} = \hat{M}^T \hat{\mathbf{a}} = M^T \tilde{\mathbf{a}}, \tag{3.8}$$

where $\tilde{\mathbf{a}}$ is a sparse log-amplitude spectrum of the same dimensionality with $\mathbf{a}$, but with nonzero values only at the L observable frequencies. The second equality comes from the fact that $\hat{M}$ in Eq. (3.7) is a sub-matrix (a subset of rows) of M in Eq. (3.5) at the L observable frequency bins.

Now, examining Eq. (3.4), $\mathbf{c}_{\text{udc}}$ can be viewed as an ordinary cepstrum calculated from the sparse spectrum $\tilde{\mathbf{a}}$. Remember that an ordinary cepstrum is the least square solution to reconstruct the spectral envelope. This means the reconstructed spectral envelope from $\mathbf{c}_{\text{udc}}$ using Eq. (3.3) has to be close to $\tilde{\mathbf{a}}$ not only at the L observable frequencies, but also at those non-observable frequencies, which take zero values. Being close to those zero log-amplitudes sounds useless, but that actually serves as a *regularizer* of $\mathbf{c}_{\text{udc}}$ to prevent its reconstructed spectral envelope overfitting the observable frequencies.

For the two spectra $\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$, they have many common non-observable frequencies. Therefore, their regularizers are very similar. In other words, although the observable frequencies are different, $\tilde{\mathbf{a}}^{(1)}$ and $\tilde{\mathbf{a}}^{(2)}$ are not that different, and their $\mathbf{c}_{\text{udc}}$'s will not be that different either.

From another perspective, the difference between $\mathbf{c}_{\text{udc}}$ and $\mathbf{c}_{\text{dc}}$ is that the data-dependent transformation $(\hat{M}^T\hat{M})^{-1}$ is removed. It is noted that the columns of $\hat{M}$ are not orthogonal as those of M, and $\hat{M}^T\hat{M}$ is not an identity matrix either. Multiplying by $(\hat{M}^T\hat{M})^{-1}$ is actually performing a rotation that is dependent on the observable frequencies. Since $\mathbf{c}_{\text{udc}}$'s of $\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$ are similar, their $\mathbf{c}_{\text{dc}}$'s will not be similar.

3.5. Experiments

In this section, I test the proposed multi-pitch streaming algorithm on polyphonic music recordings. Through the experiments, I want to answer the following questions:

(1) Which timbre representation (harmonic structure, MFCC or UDC) is best for streaming?

(2) How does the proposed algorithm perform on music recordings with different polyphony?

(3) What is the effect of different input MPE systems on streaming performance?

(4) Which components (e.g. initialization, timbre objective, locality constraints) of the proposed algorithm significantly affect the streaming results?

3.5.1. Dataset

I use the Bach10 dataset[4], as described in Section 2.6.1. As a reminder, this dataset consists of real musical instrumental performances of ten pieces of J.S. Bach four-part chorales. Each piece is about thirty seconds long and was performed by a quartet of instruments: violin (Track 1), clarinet (Track 2), tenor saxophone (Track 3) and bassoon (Track 4). Each musician's part was recorded in isolation while the musician listened to the others through headphones. The sampling rate was 44.1kHz. The ground-truth pitch trajectories were created using the robust single pitch detection algorithm YIN[**24**] on the isolated instrument recordings, followed by manual corrections where necessary.

[4]Download at `http://cs.northwestern.edu/~zdu459/resource/Resources.html`.

For each of the ten pieces, I created single-channel recordings of six duets, four trios and one quartet, by mixing the individual tracks with different combinations. This provided me in total 110 pieces of music with different polyphony.

3.5.2. Input Multi-pitch Estimates

As stated before, the proposed multi-pitch streaming algorithm can take frame-level pitch estimates from any MPE algorithm as inputs. Here I test it using three MPE algorithms. I provide the number of instruments in the mixture to these MPE algorithms and let them estimate the instantaneous polyphony in each frame.

The first one is our previous work [**33**], denoted by "Duan10". It is a general MPE algorithm based on probabilistic modeling of spectral peaks and non-peak regions of the amplitude spectrum.

The second one is [**70**], denoted by "Klapuri06". I use Klapuri's original implementation and suggested parameters. This is an iterative spectral subtraction approach. At each iteration, a pitch is estimated according to a salience function and its harmonics are subtracted from the mixture spectrum.

The third one is [**93**], denoted by "Pertusa08". I use Pertusa's original implementation and suggested parameters. This is a rule-based algorithm. In each time frame, it first selects a set of pitch candidates from spectral peaks, then all their possible combinations are generated. The best combination is chosen by applying a set of rules, taking into account its harmonic amplitudes and spectral smoothness.

Since pitch estimates of MPE algorithms contain errors and these errors will be propagated to the streaming results, I also use ground-truth pitches as inputs and let the proposed approach to cluster these error-free pitches into trajectories.

3.5.3. Parameter Settings

For all the MPE algorithms, the audio mixture is segmented into frames with 46ms-long frames with 10ms hope size. The pitch range of Duan10 and Klapuri08 is set to C2-B6 (65Hz-1976Hz). The pitch range of Pertusa08 is set as-is.

In imposing the must-links, I set the time and frequency difference thresholds Δ_t and Δ_f to 10ms and 0.3 semitones, respectively. 10ms is the time difference between adjacent frames, and 0.3 semitones correspond to the range that the pitch often fluctuates within a note. These thresholds are quite conservative to assure that most must-links are correct.

After clustering, I perform an additional postprocessing step. I merge two adjacent must-link groups of the same instrument if their time gap (the time interval between the offset of the previous group and the onset of the latter group) is less than 100ms. I also remove must-link groups that are shorter than 100ms. I choose this threshold because 100ms is the length of a 32nd note in a piece of music with a moderate tempo of 75 beats per minute. This step fills some small holes and removes some short notes in the pitch trajectories that are not musically meaningful.

3.5.4. Evaluation Measure

Given a polyphonic music with K monophonic instruments, the proposed multi-pitch streaming algorithm streams pitch estimates in individual frames into K pitch trajectories,

each of which corresponds to an instrument. To evaluate the streaming results, I first find the bijection between the K ground-truth pitch trajectories and the K estimated trajectories. In the experiment, I choose the bijection that gives us the best overall multi-pitch streaming accuracy. This accuracy is defined as follows. For each estimated pitch trajectory, I call a pitch estimate in a frame correct if it deviates less than 3% in Hz (a quarter-tone) from the pitch in the same frame and in the *matched* ground-truth pitch trajectory. This threshold is in accordance with the standard tolerance used in measuring correctness of pitch estimation for music [**70**]. Then the overall multi-pitch estimation and streaming accuracy is defined as:

$$\text{Acc} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}, \tag{3.9}$$

where TP (true positives) is the number of correctly estimated and streamed pitches, FP (false positives) is the number of pitches that are present in some estimated trajectory but do not belong to its matched ground-truth trajectory, and FN (false negatives) is the number of pitches that belong to some ground-truth trajectory but are not present in its matched estimated trajectory.

3.5.5. Comparison of Timbre Features

To investigate the effects of timbre features on the multi-pitch streaming performance, I run the proposed approach on the ground-truth pitch inputs, comparing system performance using three timbre representations: 50-d harmonic structure calculated from the mixture spectrum directly, 21-d MFCC feature calculated from separated signal of each pitch estimate using harmonic masking, and 21-d UDC feature calculated from the

mixture spectrum directly. To remove the effect caused by the pitch height arrangement of different tracks, I initialize all partitions randomly. Figure 3.4 shows the results.

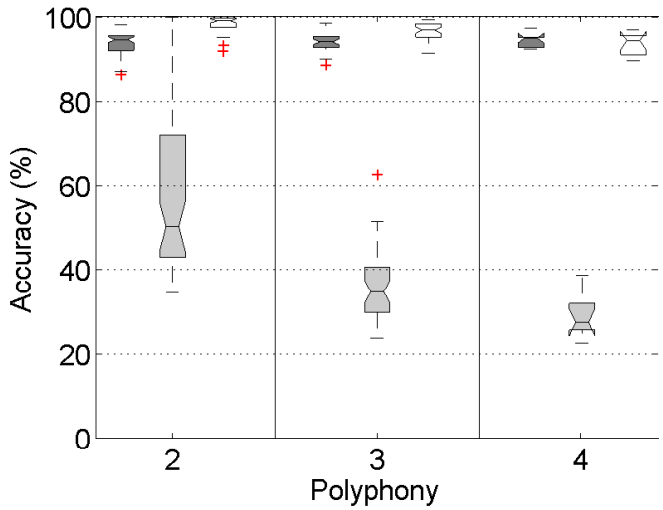


Figure 3.4. Comparison of multi-pitch streaming accuracy of the proposed approach using three kinds of timbre features: 50-d harmonic structure (dark gray), 21-d MFCC (light gray) and 21-d UDC (white). Input pitches are ground-truth pitches without track information. Clusterings are randomly initialized to remove the pitch order information.

In this and all the following box plots figures, the lower and upper lines of each box show 25th and 75th percentiles of the sample. The line in the middle of each box is the sample median. The lines extending above and below each box show the extent of the rest of the samples, excluding outliers. Outliers are defined as points over 1.5 times the interquartile range from the sample median and are shown as crosses.

For all the polyphonies, harmonic structure and UDC work well, and outperform MFCC significantly. The validity of harmonic structure for musical instruments has been validated in our previous work [**35, 28**]. It is interesting to see that UDC works even better, given the dimensionality of UDC is smaller. A nonparametric sign test shows that

the median accuracy achieved by UDC outperforms that by harmonic structure when polyphony is two or three. Although the effect is small, it is statistically significant ($p < 10^{-5}$).

On the other hand, MFCC calculated from separated spectra achieves much worse results. This can be credited to two things. First, compared to harmonic structure or UDC that only encode information at harmonics, MFCC is not that discriminative for harmonic instruments. Second, the source separation step required to use MFCC (see Section 3.4.2) may further deteriorate the performance of MFCC.

3.5.6. The Effect of the Input Multi-pitch Estimation

Given that harmonic structure and UDC performed similarly in the timbre feature evaluation, I test performance of our system in combination with several existing MPE approaches using the 50-d harmonic structure vector as the timbre feature. Figure 3.5 shows the box plots of the overall multi-pitch streaming accuracies achieved.

Note MPE accuracy is defined as the overall multi-pitch streaming accuracy except that a pitch estimate is called correct only according to the time and frequency criteria, ignoring the trajectory information. Therefore, the average overall multi-pitch streaming accuracy cannot be higher than the average MPE accuracy.

Comparing the accuracies achieved with the three MPE inputs, we see that the one taking Duan10 as inputs are much better than those taking Klapuri06 and Pertusa08 inputs. This is in accordance with their average input MPE accuracies. More accurate MPE inputs lead to more accurate multi-pitch streaming results. The median accuracy achieved by the best multi-pitch streaming configuration (using Duan10 as input) is about

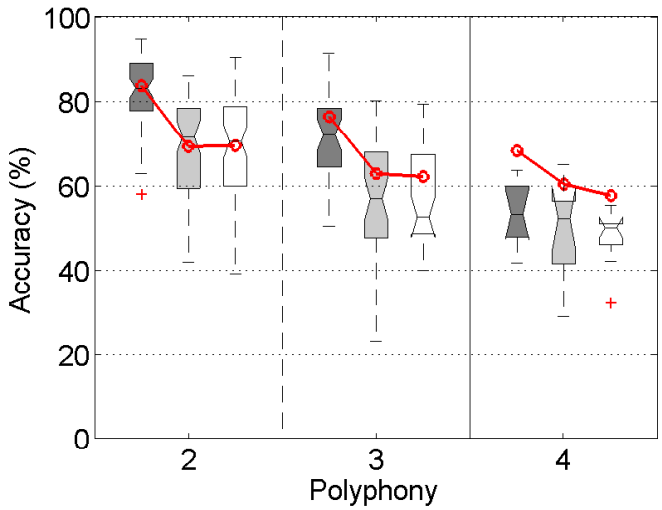


Figure 3.5. Boxplots of overall multi-pitch streaming accuracies achieved by the proposed method on the Bach chorale music pieces, taking input pitch estimates provided by three MPE algorithms: Duan10 (dark gray), Klapuri06 (light gray) and Pertusa08 (white). Each box of polyphony 2, 3 and 4 represents 60, 40 and 10 data points, respectively. The lines with circles show the average input MPE accuracy of the three MPE algorithms.

83% for duets, 72% for trios and 53% for quartets. This is promising, considering the difficulty of the task. The only information provided to the MPE algorithm and the proposed streaming algorithm about these music recordings is the number of instruments in the mixture.

To see the errors attributed only to the proposed streaming algorithm, consider Figure 3.5 where the pitch inputs are ground-truth pitches and the algorithm starts from a random clustering. We can see that the final clustering is very accurate for all polyphony when the pitch estimates are accurate.

3.5.7. Individual Analysis of System Components

As described in Section 3.2, the proposed approach utilizes two kinds of information to cluster pitch estimates. Timbre is utilized through the objective function; while pitch locality information is utilized through the constraints. I claimed that both are essential to achieve good results. In addition, I claimed that the pitch-order initialization is more informative than a random initialization in Section 3.3.1.

In this experiment, I analyze the effect caused by each individual aspect and their combinations. More specifically, I run the clustering algorithm in the following configurations, with the 50-d harmonic structure as the timbre feature:

(1) *Timbre*: from random initialization, run the algorithm to only optimize the timbre objective function; equivalent to K-means algorithm.

(2) *Locality*: from random initialization, run the algorithm to only satisfy more locality constraints.

(3) *T+L*: from random initialization, run the full version of the proposed algorithm to optimize the timbre objective as well as satisfy more locality constraints.

(4) *Order*: clustering by only pitch-order initialization.

(5) *O+T*: Configuration 1 with pitch-order initialization.

(6) *O+L*: Configuration 2 with pitch-order initialization.

(7) *O+T+L*: Configuration 3 with pitch-order initialization.

Figure 3.6 shows box plots of the multi-pitch streaming accuracy of these configurations on the ten quartets. It can be seen that the pitch-order initialization itself (Order) does not provide a satisfying clustering, even though the pitch trajectories of the Bach

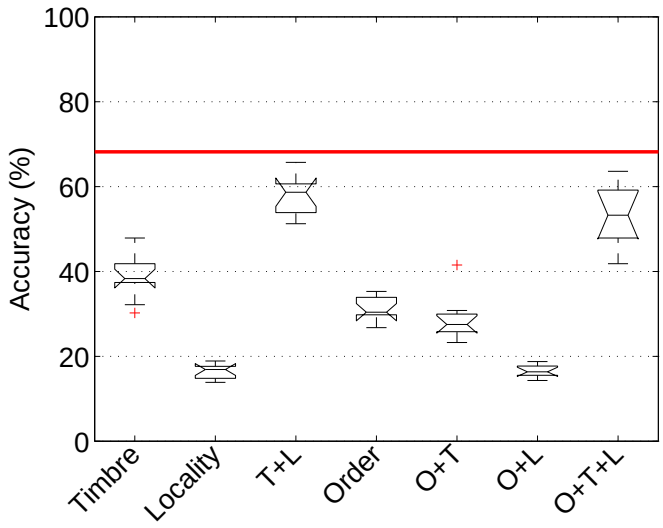


Figure 3.6. Box plots of multi-pitch streaming accuracies of the proposed approach with different system configurations, taking the same input pitch estimates from Duan10. Each box contains ten data points corresponding to the ten quartets. The horizontal line is the average input MPE accuracy.

chorales rarely interweave. This is due to the polyphony estimation and pitch estimation errors. Only using the locality constraints information, no matter what initialization (Locality and O+L), achieves the worst clustering. Only using the timbre information (Timbre and O+T) achieves better clustering but still non-satisfying. Utilizing both timbre and locality information (T+L and O+T+L) achieves significantly better clustering than only using either one of them. This supports our claim that both timbre and locality are essential for good clustering. In this case, the pitch-order initialization does not help the clustering much, as a nonparametric paired sign test favors the null hypothesis that the median difference between T+L and O+T+L is 0 ($p = 0.11$). However, the pitch-order initialization does make the algorithm converge faster, because the final clustering is "closer" (requires less swaps) from the pitch-order initialization than from a random initialization, since the pitch trajectories of the music pieces do not often interweave. For

example, the number of iterations for Algorithm 2 to terminate on the first quartet is reduced from 2781 to 313.

3.6. Conclusions

In this chapter I proposed a constrained-clustering approach for multi-pitch streaming of harmonic sound sources. Given pitch estimates in individual time frames provided by some multi-pitch estimation (MPE) algorithm, the proposed approach streams pitch estimates of the same source into a long and discontinuous pitch trajectory. This approach is unsupervised, i.e. it does not require pre-training source models on isolated recordings. It is general and can be applied to different kinds of harmonic sounds (e.g. musical instruments, speech, etc.). It is also highly compatible and can take the outputs of any MPE methods as inputs.

I also proposed a new variant of cepstrum called uniform discrete cepstrum (UDC) to represent the timbre of sound sources. UDC can be calculated from the mixture spectrum directly. Experiments show that UDC achieves better performance than ordinary cepstrum features such as MFCC, which requires source separation before feature calculation.

I evaluated the proposed approach on both polyphonic music and multi-talker speech datasets. I also compared it with several supervised and unsupervised state-of-the-art methods, which were specially designed for either music or speech. The proposed approach achieves better performance than the comparison methods on both datasets.

For future work, I would like to improve the problem formulation. Currently the constraints are binary. It may be beneficial to design soft constraints so that many existing

nonlinear optimization algorithms can be used. In addition, I would like to incorporate higher-level domain knowledge such as musicological information into the objective and constraints. I also would like to design new features and apply the proposed algorithm on more kinds of harmonic sounds and explore its broader applications. Finally, designing a method that can jointly estimate the pitches and the streams that they belong to would be an important direction to pursue to solve the multi-pitch analysis problem.

CHAPTER 4

Multi-pitch Estimation and Streaming of Multi-talker Speech

The proposed MPE and MPS methods in Chapter 2 and Chapter 3 can be applied to polyphonic audio other than music. In this chapter, I perform experiments on multi-talker speech. Estimating the pitch stream for each underlying talker in a multi-talker speech mixture would be beneficial for speech recognition, speaker identification, prosody and emotion analysis, and speech segregation.

In the following, I will first describe the main differences between music and speech signals in Section 4.1, which will affect the parameter settings in the proposed MPE and MPS methods. Then I will present the experiments in MPE and MPS in Section 4.2 and 4.3, respectively.

4.1. Differences between Music and Speech

The music data I deal with in this dissertation is polyphonic music audio composed of harmonic sound sources. No inharmonic sounds such as percussive instruments are involved. Speech data, however, are quasi-harmonic. There are both harmonic sounds (like vowels) and inharmonic sounds (like fricatives). In a multi-talker speech mixture, each audio frame can be a mixture of both harmonic signals and inharmonic signals. This adds some difficulties to the MPE problem, as the inharmonic signals are essentially noise that interfere the harmonic signals.

In addition, the pitch contours of speech are less stable than those in music. For music, most notes have a relatively constant pitch. Even if a note has vibrato, its pitch only fluctuates within a small frequency range. Speech signals, however, often have gliding pitch contours even within a vowel. In my observation, these gliding pitches can change by up to 5 semitones within 60 ms. This property affects several parameter settings of the proposed MPE and MPS systems. First, the frame length of the MPE system is changed from 46ms for music to 32ms for speech. Second, the smoothing window size in the postprocessing step of the MPE system (Section 2.4) is changed from 17 frames to 3 frames. Third, the frequency threshold Δ_f to impose must-link constraints between pitch estimates (Section 3.2.2) is changed from 0.3 semitones to 1 semitone.

Furthermore, the pitch contours of concurrent talkers can often interweave if the talkers are of the same gender. This is not that frequent in music even if two parts are played by the same kind of instrument. The interweaving properties make MPS more difficult.

Finally, the timbre of a talker is harder to model than the timbre of a musical instrument. As described in Section 3.4, the timbre of a sound source can be mostly captured by the frequency response of its resonance filter. This response corresponds to the spectral envelope, and can be approximated by different timbre features. For a musical instrument, the shape of the resonance filter (i.e. the instrument body) is fixed, hence the spectral envelope is relatively stable. For a human talker, the shape of the resonance filter (i.e. the vocal tract) varies significantly when different vowels are pronounced. This makes the spectral envelope not that stable and harder to approximate. In Section 3.5.5, I have shown that both the harmonic structure and the UDC feature can well represent the timbre of a musical instrument. For speech, however, harmonic structure captures too

many details of the spectrum and cannot be robustly used as the timbre feature. UDC, on the other hand, will be shown to work well as the timbre feature in Section 4.3.6.

There are also some properties in speech that make multi-pitch analysis easier than in music. The most important one regards the overlapping harmonic issue (Section 2.1.1). Concurrent sound sources are intentionally made harmonic to each other in music composition, which leads to much overlap between harmonics from different sources. Concurrent talkers, on the other hand, have much less dependence between each other and therefore harmonics from different talkers typically do not overlap in concurrent speech.

4.2. Multi-pitch Estimation Experiments

In this section, I test the proposed MPE method in Chapter 2 on a multi-talker speech dataset.

4.2.1. Dataset

The dataset I use is the Pitch-Tracking Database from Graz University of Technology (PTDB-TUG) [**94**]. This database consists of recordings of twenty native English speakers (ten male and ten female) from different home countries (USA, Canada, England, Ireland and South Africa), reading phonetically rich sentences from the TIMIT corpus [**50**]. The TIMIT corpus consists of 450 phonetically-compact sentences and 1890 phonetically-diverse sentences. Each sentence was read by one female and one male subject. In total there are 4680 recorded utterances, 900 of which are of phonetically-compact sentences and 3780 are of phonetically-diverse sentences. Each utterance has about four seconds of

voice and a couple of seconds of silence before and after. All the recordings were recorded in 48kHz.

Among the 3780 phonetically-diverse utterances from all twenty subjects, I selected five male and five female subjects to form the test set. This accounts for 1890 utterances. I randomly mixed these utterances, ensuring all talkers has equal root-mean-squared amplitude, to generate each multi-talker speech mixture. I considered four conditions according to the number of talkers and their gender relations: two-talker different gender (DG), two-talker same gender (SG), three-talker DG and three talker SG. I generated 100 mixtures for each condition, totalling 400 test mixtures.

The database provides a ground-truth pitch track for each utterance. Ground-truth pitch estimates were generated from individual talkers, prior to mixing using RAPT [**117**], a well known method to find pitch in voice on the filtered laryngograph signal from the utterance. The frame length and hop size were 32ms and 10ms, respectively. However, I found that these ground-truth pitch tracks contain some errors due to noise in the laryngograph signal. Therefore, I generated and used our own ground-truth pitch tracks with Praat [**7**] on the utterances[1], using the same frame length and hop size. I found that about 85% of the Praat-generated ground-truth pitches agree with the RAPT-generated ground-truth pitches. The pitch range of the utterances is between 65Hz to 370Hz.

4.2.2. Evaluation Measure

I use the multi-pitch estimation accuracy described in Section 2.6.2 to measure the pitch estimation results. Unlike music, the frequency deviation threshold to determine whether

[1]Downloadable at `http://cs.northwestern.edu/~zdu459/resource/Resources.html`

a pitch estimate is a good match to a ground-truth pitch is changed from 3% of the pitch frequency in Hz (a quarter tone) to 10%. This is a commonly used criterion in multi-pitch analysis literature for speech data [**140, 64, 136**]. To measure polyphony estimation, I use the Mean Square Error (MSE) described in Section 2.6.2.

4.2.3. Comparison Method

I compare the proposed MPE method with two state-of-the-art multi-pitch analysis methods specifically designed for speech. The first one is [**140**], denoted by "Wu03". I use their original implementation and suggested parameters. This algorithm uses a hidden Markov model (HMM) to model both the change in instantaneous polyphony and pitch values. It can estimate pitches of up to two simultaneous talkers.

The second one is [**64**], denoted by "Jin11". I use their original implementation and suggested parameters. This algorithm extends [**140**] to reverberant environments, and can also estimate pitches of up to two simultaneous talkers.

4.2.4. Parameter Settings

For the proposed method, I trained the likelihood models with 500 multi-talker mixtures using phonetically-compact utterances of the other five male and five female subjects. Similar to the test set, I randomly mixed these utterances with equal RMS levels to generate in total 400 two-talker and three-talker mixtures in both different-gender (DG) and same-gender (SG) conditions. I also included 100 single-talker utterances. These 500 training recordings were used as a whole to train the MPE model. It is noted that the

training set and the test set do not overlap in either utterances, subjects, sentences or phonetic vocabularies.

For all comparison methods, the audio mixtures were segmented into frames with length of 32ms and hop size of 10ms. The pitch range was set to 65Hz-370Hz. The number of talkers in each audio mixture was provided but the MPE methods needed to estimate the instantaneous polyphony in each frame.

4.2.5. Multi-pitch Estimation Results

Figure 4.1 shows the box plot comparisons of the three methods. It can be seen that in the two-talker DG condition, the proposed method achieves comparable performance with Wu03. This is supported by a nonparametric sign test with $p = 0.92$. In the two-talker SG condition, the proposed method is slightly worse than Wu03, but the difference is not statistically significant according to a nonparametric sign test at the significance level of 0.05. In both two-talker DG and SG conditions, Wu03 and the proposed method significantly outperform Jin11. This indicates that the proposed method, which can deal with both music and speech, is comparable to one of the best state-of-the-art MPE methods that are specifically designed for speech.

In addition, in the three-talker conditions, neither Wu03 nor Jin11 can work. They are both designed to only deal with pitch estimation for up to two simultaneous talkers. The proposed method, however, does not have this restriction. We can see that its performance in the three-talker DG condition is almost comparable to that in the two-talker SG condition, which indicates that adding another talker with the different gender to a two-talker SG mixture does not significantly increase the difficulty for the proposed

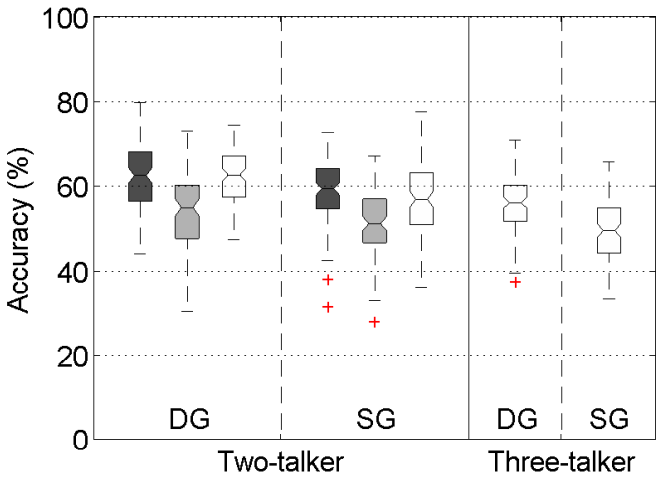


Figure 4.1. Comparison of multi-pitch estimation accuracies of Wu03 (dark gray), Jin11 (light gray) and the proposed method (white) on the multi-talker speech dataset. Here, DG means each mixture contains talkers from different genders. SG means each mixture contains talkers from only a single gender.

method. In the three-talker SG condition, the performance is worse, but the median accuracy is still around 50%. Note that pitch estimation by a random guess would achieve an accuracy very close to 0%.

4.2.6. Polyphony Estimation Results

Table 4.1 shows the comparison of instantaneous polyphony estimation of the three methods on two-talker mixtures. The true polyphony of each frame can take three values: 0, 1 or 2. The Mean Square Error (MSE) of the estimated polyphony is calculated for each true polyphony condition, as described in Section 2.6.2. Lower MSE indicates more accurate polyphony estimation. It can be seen that Wu03 achieves the lowest error when the true polyphony is 0 or 1, while the proposed method achieves the lowest error when the

Table 4.1. Mean Square Error (MSE) of instantaneous polyphony estimation of three comparison methods on two-talker mixtures.

	true polyphony	Wu03	Jin11	Proposed
DG	0	**0.051**	0.099	0.091
	1	**0.190**	0.307	0.287
	2	0.878	0.941	**0.673**
SG	0	**0.047**	0.093	0.084
	1	**0.187**	0.291	0.274
	2	1.177	1.339	**0.923**

Table 4.2. Mean Square Error (MSE) of instantaneous polyphony estimation of the proposed method on three-talker mixtures.

true polyphony	DG	SG
0	0.077	0.070
1	0.361	0.379
2	0.676	0.860
3	2.531	3.000

true polyphony is 2. Also, it can be seen that the MSEs in SG conditions are generally higher than those in the DG condition. This suggests that polyphony estimation becomes harder when the pitch ranges of simultaneous talkers overlap more.

Table 4.2 shows the MSE values of the proposed method on three-talker mixtures. It can be seent that the MSEs are not significantly different from those in Table 4.1 when there are less than or equal to two simultaneous pitches. When there are three simultaneous pitches, the MSE values are much higher, suggesting that the proposed method tends to underestimate the polyphony.

4.3. Multi-pitch Streaming Experiments

We have now seen the results for multi-pitch estimation (MPE) on speech data. How good, then, is multi-pitch streaming (MPS) on speech data. In this section, I test the proposed MPS method in Chapter 3 on a multi-talker speech dataset.

4.3.1. Dataset

I use the same speech dataset in MPE experiments as described in Section 4.2.1.

4.3.2. Input Multi-pitch Estimates

The proposed MPS approach takes pitch estimates in individual frames from an MPE approach as inputs. Here I run the proposed MPS approach with input pitch estimates from the three MPE algorithms that I compared in the MPE experiments (Section 4.2.3): Wu03 [**140**], Jin11 [**64**], and the proposed MPE algorithm denoted as Duan10. In some experiments, I also use ground-truth pitches as inputs to show how well the proposed streaming approach works with ideal inputs.

4.3.3. Parameter Settings

In imposing the must-link constraints (Section 3.2.2) on how to connect individual pitch estimates to form streams, I set the time and frequency difference thresholds Δ_t and Δ_f to 10ms and 1 semitone, respectively. The frequency threshold is larger than that used for music (0.3 semitones), since speech utterances often have fast gliding pitch contours.

For the proposed clustering algorithm, I use the proposed UDC of order 21 as the timbre feature. In Section 4.3.6 I compare its performance with two other features: 50-d harmonic structure and 21-order MFCC.

4.3.4. Evaluation Measure

I use the multi-pitch estimation and streaming accuracy defined in Section 3.5.4 to evaluate the MPS results. Similar to the MPE experiments, the frequency difference threshold to judge if a pitch estimate is matched with a ground-truth pitch is set to 10% of the ground-truth pitch frequency in Hz. This is larger than what is used for music, but is commonly used in existing multi-pitch analysis methods [**140, 64, 136**] for speech.

4.3.5. Comparison Method

I compare the proposed approach with two state-of-the-art multi-pitch estimation and streaming systems. The first one is a supervised method based on a factorial HMM [**136**], denoted by "Wohlmayr11". One HMM is used for each talker to estimate and stream the talker's pitches. The HMM parameters are trained on isolated training utterances. I use their source code and provided gender-dependent models, which give the most supervision information that we can use. The gender information gives it a small information advantage over our proposed method and the other comparison method.

The other method is an unsupervised method designed for cochannel speech separation [**61**], denoted by "Hu12". I use their source code and suggested parameters. This method estimates a pitch trajectory for each talker to construct a binary time-frequency mask to separate the mixture spectrogram. This method is built on the tandem algorithm [**60**].

Similar to the proposed approach, [**61**] also views the multi-pitch streaming problem as a constrained clustering problem, although the formulations are different. Note that [**61**] is only designed and tested for two-talker speech mixtures.

4.3.6. Comparison of Timbre Features

I first run the proposed approach with three kinds of timbre features on the ground-truth pitch inputs: 50-d harmonic structure calculated from the mixture spectrum directly, 21-d MFCC calculated from separated signal of each pitch estimate using harmonic masking, and 21-d UDC calculated from the mixture spectrum directly.

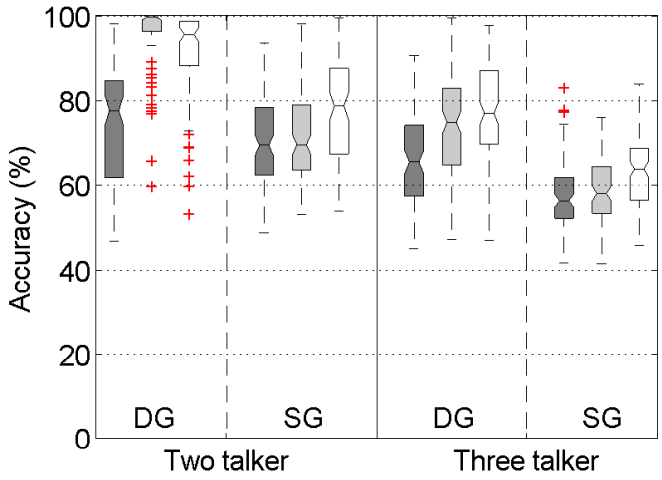


Figure 4.2. Comparison of multi-pitch streaming accuracies of the proposed approach using three kinds of timbre features: 50-d harmonic structure (dark gray), 21-d MFCC (light gray) and 21-d UDC (white). Input pitches are ground-truth pitches without track information.

The results are shown as boxplots in Figure 4.2. In three out of four conditions, the two cepstral features both significantly outperform the harmonic structure feature,

supported by a paired sign test at the 5% significance level. In the two-talker DG condition, both MFCC and UDC achieve very good streaming accuracy where MFCC achieves almost perfect results. However, when the conditions become harder, especially in the SG conditions, UDC significantly outperforms MFCC. This is because the calculation of MFCC requires source separation, which becomes less reliable when there is more overlap between concurrent sources. In contrast, the calculation of UDC is performed directly from points in the mixture spectrum that likely belong to a single source.

4.3.7. Overall Results

Figure 4.3 shows the overall comparison between Wohlmayr11, Hu12 and the proposed approach with input from three MPE algorithms, using the 21-d UDC timbre feature. It can be seen that the unsupervised methods (Hu12 and the proposed method with different inputs) significantly outperform Wohlmayr11, which uses the gender information in the mixture. It is noted, however, that Wohlmayr11 is designed to utilize supervision information and its full strength can only be shown when a model is trained for each talker in the mixture. The good results obtained by the proposed method illustrate both its compatibility with different MPE algorithms and effectiveness when combined with them to perform streaming.

In addition, the proposed multi-pitch streaming approach achieves comparable results with the state-of-the-art method Hu12. In the two-talker DG condition, the best results are obtained by Hu12, Proposed taking Duan10 and Wu03 as input. A nonparametric paired sign test shows that differences between these systems are not statistically significant at the 5% significance level. Similarly, in the two-talker SG condition, Hu12 and

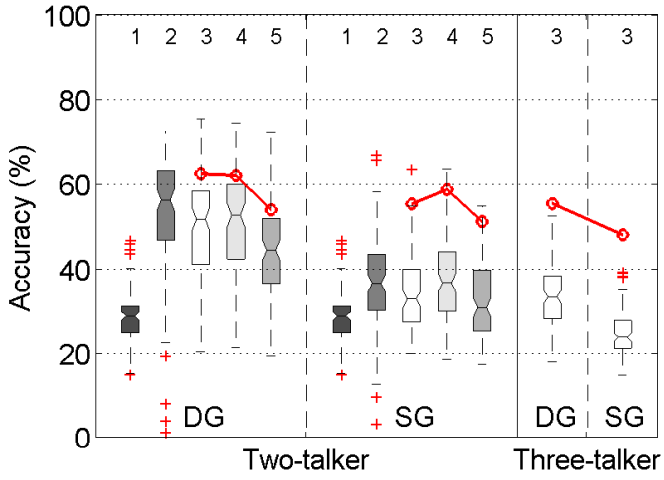


Figure 4.3. Comparison of multi-pitch streaming accuracies of 1) Wohlmayr11, 2) Hu12, and the proposed approach taking inputs from 3) Duan10, 4) Wu03 and 5) Jin11. Each box has 100 data points. The circled red lines above the boxes show the average accuracy of input pitch estimates, prior to streaming.

Proposed taking Wu03 as input obtain the best results. Their difference is not statistically significant. However, the proposed multi-pitch streaming approach is able to deal with general harmonic sounds (as shown in Section 3.5) and speech mixtures with more than two simultaneous talkers (as shown in the three-talker condition in Figure 4.3).

The errors caused by the proposed streaming approach (instead of the MPE algorithms) can be read from the gap between the box medians and the average accuracy of input pitch estimates. In the two-talker DG condition, this gap is fairly small, indicating that the proposed streaming algorithm works well. In the two-talker SG condition, this gap is significantly enlarged. This is because the pitch trajectories interweave with each other, making many must-link constraints imposed in the streaming process incorrect. The gap is further enlarged in the three-talker SG condition. One interesting thing to

notice is that there is no significant difference of the performance between two-talker SG and three-talker DG conditions. This means that adding a talker with a different gender to an existing two-talker SG mixture does not influence the pitch estimation and streaming result much. The errors made by the proposed streaming approach can also be seen in Figure 4.2, which compares clustering using different timbre features based on ground-truth pitch estimates.

4.3.8. Individual Analysis of System Components

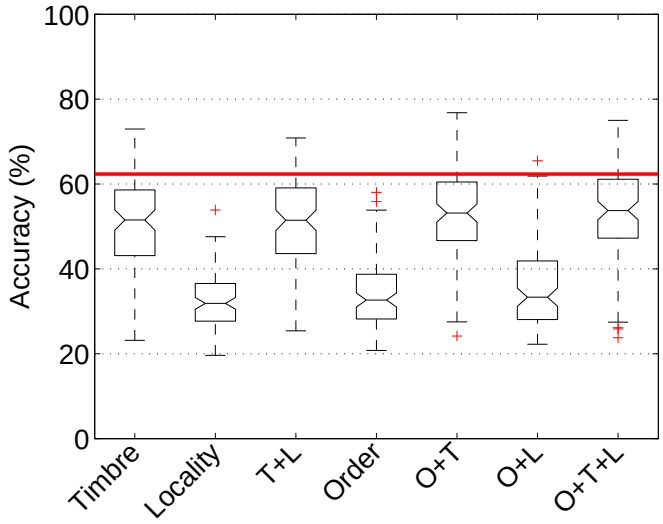


Figure 4.4. Box plots of multi-pitch streaming accuracies of the proposed approach with different system configurations, taking the same input pitch estimates from Duan10. Each box contains 100 data points corresponding to the 100 two-talker DG excerpts. The horizontal line is the average multi-pitch estimation accuracy from the best available multi-pitch estimator. The accuracy of the input pitch estimation sets an upper bound of the streaming accuracy.

I analyze the effectiveness of different system components, similar to Section 3.5.7. Figure 4.4 shows box plots of the multi-pitch streaming accuracies of these configurations

on the 100 two-talker DG excerpts using the 21-d UDC feature. It can be seen that the pitch order information (Order) does not provide results as good as in the music dataset. This is expected, as the pitch activity of the two talkers often do not overlap in time and the pitch order initialization would label almost all the pitches incorrectly to the first cluster. Only using the locality information (Locality) or combining it with the pitch order information (O+L) also does not achieve good results, which is also expected.

What I did not expect was the good performance from only using the timbre information (Timbre) or its combination with pitch order (O+T). They achieve comparable results to T+L and O+T+L. This indicates that the UDC timbre feature is good to discriminate the two talkers, while the locality information does not help much.

4.4. Conclusions

In this chapter, I tested the proposed MPE and MPS methods on a multi-talker speech dataset. I first summarized the different characteristics of speech data from music data, which affect the parameter settings of the proposed methods. I then presented experimental results on MPE and MPS individually.

Results on MPE experiments showed that the proposed MPE algorithm achieves comparable results on pitch estimation and polyphony estimation on two-talker mixtures, with state-of-the-art methods that are specifically designed for speech. The proposed method can also work for mixtures with more than two talkers while existing methods cannot.

Results on MPS experiments showed that the proposed MPS algorithm outperforms a state-of-the-art supervised method and achieves comparable results with an unsupervised

method, on two-talker mixtures. The proposed method is also able to work on three-talker mixtures, while existing methods cannot. Experiments also show the advances of the proposed UDC feature over harmonic structure and MFCC on speech data.

It is noted that although multi-pitch analysis provides important information, it is not sufficient to achieve source separation of multi-talker speech as for polyphonic music composed of harmonic sources. This is due to the inharmonic sounds in speech. How to model these sounds and combine them with multi-pitch analysis of the harmonic sounds to achieve speech separation is still a challenging problem [**61**].

Part 2

Analyzing the Music Audio Scene with A Written Score

The goal of my dissertation is to analyze a polyphonic music audio scene composed of harmonic sources, i.e. finding the musical objects and separating their signals from the mixture. One can then build on this analysis to further manipulate or interact with the music audio. In the first part, I approached this goal through multi-pitch analysis solely from audio. This approach does give us promising results on pitch estimation and streaming, based on which preliminary separation can be obtained on some sources and music. However, due to the extreme difficulty of the problem, source separation achieved in this approach in general is not satisfying for audio manipulation purposes.

There exist a number of other source separation algorithms that do not need a score. CASA-based [**82, 128, 79, 35**], spectral decomposition-based [**133, 129, 12**] and model-based methods [**107, 123, 4, 68**] have been the three main categories of this research. Again, due to the extreme difficulty of the problem, these methods usually only achieve good results in some special cases with generalization to broader scenarios being unsuccessful.

In many scenarios, the score of the music is available. One can utilize the score information to help analyze the music audio. In the second part of my dissertation, I propose algorithms to do score-informed source separation. There are two basic problems. The first one is how to align (synchronize) the audio with the score (Chapter 5). Since the audio and the score may be of different temporal dynamics, the score information cannot be used without a good alignment. The second problem is how to separate the sound sources given an audio-score alignment (Chapter 6). In Section 6.4, I will compare the score-informed source separation results with the separation results achieved from the first

part of the dissertation. Results show that the score information does help significantly. Finally, I propose two interesting applications of the proposed algorithms in Chapter 7.

CHAPTER 5

Audio-score Alignment

5.1. Introduction

Audio-score alignment is the problem of aligning (synchronizing) a piece of music audio with its score. It belongs to a broader topic called *Score Alignment*, which is to align a general musical performance (not necessarily audio) with its underlying score. In fact, the score alignment research started from addressing MIDI performances [**20, 122**], and in recent years gradually shifted to audio performances. Audio performances are much harder to deal with, since they are not directly machine understandable as MIDI.

Audio-score alignment can be addressed *offline* or *online*. An offline algorithm can use the whole performance of a music piece. The online version (also called *Score Following*) cannot "look ahead" at future performance events when aligning the current event to the score. While offline algorithms can only be used in offline applications, online algorithms can be used in both offline and online scenarios, and even be made to work in real time if they are fast enough. In my dissertation, I propose to address online audio-score alignment, which will support broader applications of music audio scene analysis.

Besides its importance to the proposed MASA system, audio-score alignment has a number of real-world applications. Offline algorithms have been used to synchronize multiple modalities (video, audio, score, etc.) of music to build a digital library [**118**].

Online algorithms have been used to make an automatic accompaniment system, where the system follows a solo performance and plays the rest parts of a band [**101**].

5.1.1. Related Work

Table 5.1 classifies existing algorithms on audio-score alignment. The rows classify them according to the music they can be applied to, from easy to hard into monophonic, single-instrument polyphonic (e.g. piano or guitar) and multi-instrument polyphonic (e.g. ensemble). Single-instrument polyphonic music is in general easier to deal with than multi-instrument polyphonic music, because the timbre complexity of the former is usually much lower than the latter. Columns classify them into offline and online. Since I am proposing an online algorithm, I will only review online algorithms in the following.

Table 5.1. Prior work on audio-score alignment.

	Offline	Online
Monophonic	[**10, 100**]	[**96, 56, 89, 101**]
Polyphonic, Single-instrument	[**102, 65**]	[**26, 16, 17**]
Polyphonic, Multi-instrument	[**91, 62, 44**]	[**55, 87, 86**]

References [**96, 56, 89, 101**] are online audio-score alignment algorithms, but only for monophonic audio performances. They cannot be applied to polyphonic performances, due to the intrinsic differences between monophonic and polyphonic music as stated in Section 2.1.1. In addition, two of these systems ([**89, 101**]) also require training of the system on prior performances of each specific piece before alignment can be performed for a new performance of the piece. This limits the applicability of these approaches to pieces with preexisting aligned audio performances.

[**26, 16, 17**] are online algorithms designed for single-instrument polyphonic audio performances. [**26**] uses an online DTW algorithm to follow piano performances, where each audio frame is represented by a spectral temporal difference vector. This onset-informed feature works well for piano performances, however, may have difficulties for instruments with smooth onsets like strings and wind. [**16**] proposed a hierarchical HMM approach to follow piano performances, where the observation likelihood is calculated by comparing the pitches at the hypothesized score position and pitches transcribed by Nonnegative Matrix Factorization (NMF) with fixed basis spectra. A basis spectrum is learned for each pitch of the same piano beforehand. This method might have difficulties in generalizing to multi-instrument polyphonic audio, as the timbre variation and tuning issues involved make it difficult to learn a general basis spectrum for each pitch. [**17**] proposed a probabilistic inference framework with two coupled audio and tempo agents to follow a polyphonic performance and estimate its tempo. This system works well on single-instrument polyphonic audio and is in theory applicable to multi-instrument polyphonic audio, however, not enough statistical results are available to evaluate the system's performance up to date.

[**55, 87, 86**] are online algorithms designed for multi-instrument polyphonic audio performances. [**55**] adopts string matching to follow a musical ensemble, where each instrument needs to be recorded by a close microphone and streamed into a monophonic pitch sequence. The system reads these pitch sequences, hence will not work in situations where the instruments are not separately recorded. [**87**] uses a finite state HMM to model the audio, where a state is a chord in the score. Each audio frame is processed by a discrete filter bank. The responses of the filters that correspond to score notes give the matchness

between the audio frame and the score position. However, the coarse frequency resolution of filter banks prevent it from differentiating close pitches. [**86**] uses a continuous-state HMM process to model the audio performance, which allows an arbitrary precision of the alignment. This idea is the same as the proposed method. In fact, they were published at the same time. However, [**86**] does not use an as accurate representation as the proposed method for audio.

5.1.2. Advances of the Proposed Method

I proposed a method to address the online audio-score alignment problem for multi-instrument polyphonic music from a single-channel input. I used a continuous state HMM process to model the audio performance, which will allow an arbitrary precision of alignment. I also used the multi-pitch information which provides a more accurate connection between audio and score than traditional representations such as chroma features.

Table 5.2. Comparison of the proposed method with existing online audio-score alignment method for multi-instrument polyphonic music.

	[**17**]	[**55**]	[**87**]	[**86**]	Proposed
Takes single-channel mixture as input	✓		✓	✓	✓
Uses a continuous state model				✓	✓
Models tempo explicitly	✓	✓		✓	✓
Uses multi-pitch information	✓		✓	✓	✓
Tested on a large multi-instrument polyphonic dataset			✓		✓

The proposed method has been published in [**32, 31**]. As shown in Table 5.2, the proposed method contributes to the online audio-score alignment field in several ways:

- It uses a continuous state HMM model, which allows each audio frame to be aligned to any score position with any precision. Existing HMM-based methods

except [**86**] all use finite states, which only permits alignment precision at the note/chord level.

- It explicitly models tempo as a parameter of the performance in the process model. This improves alignment accuracy as shown in experiments.
- It uses an state-of-the-art multi-pitch likelihood model as the observation model. Compared to the commonly used chroma representation, multi-pitch information captures more details in music and forms a better connection between audio and score. Compared to the multi-pitch models in [**17, 87, 86**], the one in the proposed method is more accurate.

5.2. A Hidden Markov Process for Score Following

5.2.1. System Overview

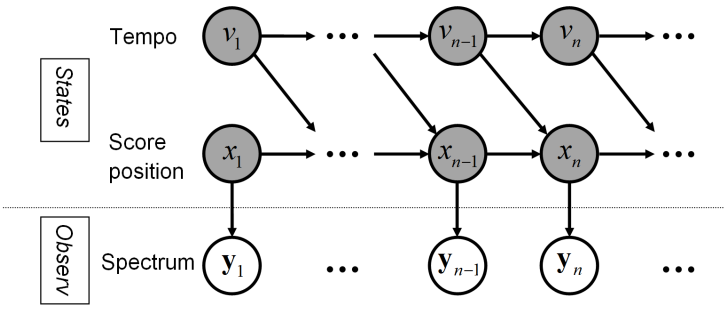


Figure 5.1. Illustration of the state space model for online audio-score alignment.

I propose a hidden Markov process model, as illustrated in Figure 5.1. I decompose the audio performance into time frames and process the frames in sequence. The n-th frame is associated with a 2-dimensional hidden state vector $\mathbf{s}_n = (x_n, v_n)^T$, where x_n is its score position (in beats), v_n is its tempo (in Beat Per Minute (BPM)) and T denotes

matrix transposition. x_n is drawn from the interval containing all score positions from the beginning to the end. v_n is drawn from the interval of all possible tempi $[v^l, v^h]$, where the lowest tempo v^l is set to half of the score tempo and the highest tempo v^h is set to twice the score tempo. These values were selected as broad limits on how far from the notated tempo a musician would be likely to deviate. Note that the the values of v^l and v^h can be chosen based on prior information about the score. In addition, for multi-tempi pieces, v^l and v^h can be changed correspondingly when a new tempo is encountered.

Each audio frame is also associated with an observation, which is a vector of PCM encoded audio, $\mathbf{y}_n$. The aim is to infer the current score position x_n from current and previous observations $\mathbf{y}_1, \cdots, \mathbf{y}_n$. To do so, I need to define a process model to describe how the states transition, an observation model to evaluate hypothesized score positions for the current audio frame, and to find a way to do the inference in an online fashion.

5.2.2. The Process Model

A process model defines the transition probability from the previous state to the current state, i.e. $p(\mathbf{s}_n|\mathbf{s}_{n-1})$. I use two dynamic equations to define this transition. To update the score position, I use

$$x_n = x_{n-1} + l \cdot v_{n-1}, \tag{5.1}$$

where l is the audio frame hop size (10 milliseconds, in this work). Thus, score position of the current audio frame is determined by the score position of the previous frame and the current tempo.

To update the tempo, I use

$$v_n = \begin{cases} v_{n-1} + n_v & \text{if } z_k \in [x_{n-1}, x_n] \text{ for some } k \\ v_{n-1} & \text{otherwise} \end{cases}, \tag{5.2}$$

where $n_v \sim \mathcal{N}(0, \sigma_v^2)$ is a Gaussian noise variable; z_k is the k-th note onset/offset time in the score. This equation states that if the current score position has just passed a note onset or offset, then the tempo makes a random walk around the previous tempo according to a Gaussian distribution; otherwise the tempo remains the same.

The noise term n_v introduces randomness to the system, which is to account for possible tempo changes of the performance. Its standard deviation σ_v is set to a quarter of the notated score tempo through this paper. I introduce the randomness through tempo in Eq. (5.2), which will affect the score position as well. But I do not introduce randomness directly in score position in Eq. (5.1). In this way, I avoid disruptive changes of score position estimates.

In addition, randomness is only introduced when the score position has just passed a note onset or offset. This is because it is rather rare that the performer changes tempo within a note. Second, on the listener's side, it is impossible to detect the tempo change before hearing an onset or offset, even if the performer does make a change within a note. Therefore, changing the tempo state in the middle of a note is not in accordance with music performance, nor does it have evidence to estimate this change.

5.2.3. The Observation Model

The observation model is to evaluate whether a hypothesized state can explain the observation, i.e. $p(\mathbf{y}_n|\mathbf{s}_n)$. Different representations of the audio frame can be used. For example, power spectra [**95**], auditory filterbank responses [**69**], chroma vectors [**62**], spectral and harmonic features [**89, 91**], multi-pitch analysis information [**16**], etc. Among these representations, multi-pitch analysis information is the most informative one to evaluate the hypothesized score position for most fully-scored musical works. This is because pitch information can be directly aligned to score information. Therefore, inspired by [**16**], I use multi-pitch observation likelihood as our preferred observation model.

In the proposed score follower, the multi-pitch observation model is adapted from the likelihood model of the proposed multi-pitch estimation described in Chapter 2. The maximum likelihood model aims to find the set of pitches that maximizes the likelihood of the power spectrum. As a reminder, this likelihood model is trained on thousands of isolated musical chords generated by different combinations of notes from 16 kinds of instruments. These chords have different chord types (major, diminished, etc.), instrumentations, pitch ranges and dynamic ranges, hence the trained likelihood model performs well in multi-instrument polyphonic music pieces as shown in 2.6.

In calculating the observation likelihood $p(\mathbf{y}_n|\mathbf{s}_n)$ in the proposed score follower, I simply extract the set of all pitches at score position x_n and use it as $\boldsymbol{\theta}$ in Eq. (2.2). Then $p(\mathbf{y}_n|\mathbf{s}_n)$ is defined as

$$p(\mathbf{y}_n|\mathbf{s}_n) = -\frac{C}{\ln \mathcal{L}(\boldsymbol{\theta})}, \tag{5.3}$$

where C is the normalization factor to make it a probability.

Note that I do not define $p(\mathbf{y}_n|\mathbf{s}_n) = \mathcal{L}(\boldsymbol{\theta})$, because it turns out that $\mathcal{L}(\boldsymbol{\theta})$ can differ by orders of magnitude for different sets of candidate pitches drawn from the score. Since, I combine the process model and observation model to infer score position, this large variation in observation model outputs can cause the observation model to gain too much importance relative to the process model. For example, two very close score position hypotheses would get very different observation likelihood, if they indicate different sets of pitches. However, the probabilities calculated from the process model are not that different. Therefore, the posterior probability of score position would be overly influenced by the observation model, while the process model would be almost ignored. If the observation model does not function well in some frame (e.g. due to a pitch-estimation glitch), the estimated score position may jump to an unreasonable position, although the process model tends to proceed from the previous score position smoothly. Eq. (5.3) compresses $\mathcal{L}(\boldsymbol{\theta})$. This balances the process model and the observation model.

Note that in constructing this observation model, I do not need to estimate pitches from the audio frame. Instead, I use the set of pitches indicated by the score. This is different from [**16**], where pitches of the audio frame are first estimated, then the observation likelihood is defined based on the differences between the estimated pitches and score-informed pitches. By skipping the pitch estimation step, I can directly evaluate the score-informed pitches at a hypothesized score position using the audio observation. This reduces model risks caused by pitch estimation errors.

The proposed observation model only considers information from the current frame, and could be improved if considering information from multiple frames. Ewert et al. [**44**]

incorporate inter-frame features to utilize note onset information and improve the alignment accuracy. Joder et al. [**65**] propose an observation model which uses observations from multiple frames for their conditional random field-based method. In the future I want to explore these directions to improve our score follower.

5.3. Inference by Particle Filtering

Given the process model and the observation model, we want to infer the state of the current frame from current and past observations. From a Bayesian point of view, this means we first estimate the posterior probability $p(\mathbf{s}_n|\mathbf{Y}_{1:n})$,then decide its value using some criterion like maximum a posterior (MAP) or minimum mean square error (MMSE). Here, $\mathbf{Y}_{1:n} = (\mathbf{y}_1, \cdots, \mathbf{y}_n)$ is a matrix whose each column denotes the observation in one frame. Recall $\mathbf{s}_n = (x_n, v_n)^T$, where x_n is score position (in beats), v_n is tempo (in Beat Per Minute (BPM)) and T denotes matrix transposition. By Bayes' rule, we have

$$\begin{aligned} & p(\mathbf{s}_n|\mathbf{Y}_{1:n}) \\ = \; & C_n p(\mathbf{y}_n|\mathbf{s}_n) \int p(\mathbf{s}_n|\mathbf{s}_{n-1}) p(\mathbf{s}_{n-1}|\mathbf{Y}_{1:n-1})\, d\mathbf{s}_{n-1}, \end{aligned} \tag{5.4}$$

where $\mathbf{y}_n$, $\mathbf{Y}_{1:n}$, $\mathbf{s}_n$ and $\mathbf{s}_{n-1}$ are all random variables; $\mathbf{s}_{n-1}$ is integrated over the whole state space; C_n is the normalization factor. $p(\mathbf{y}_n|\mathbf{s}_n)$ is the observation model defined in Eq. (5.3) and $p(\mathbf{s}_n|\mathbf{s}_{n-1})$ is the process model defined by Eq. (5.1) and (5.2).

We can see that Eq. (5.4) is a recursive equation of the posterior probability $p(\mathbf{s}_n|\mathbf{Y}_{1:n})$. It is updated from the posterior probability in the previous frame $p(\mathbf{s}_{n-1}|\mathbf{Y}_{1:n-1})$, using the

state transition probability $p(\mathbf{s}_n|\mathbf{s}_{n-1})$ and the observation probability $p(\mathbf{y}_n|\mathbf{s}_n)$. Therefore, if we can initialize $p(\mathbf{s}_1|\mathbf{y}_1)$ in the first frame and update it using Eq. (5.4) as each frame is processed, the inference can be done online.

This is the general formulation of online filtering (tracking). If all the probabilities in Eq. (5.4) are Gaussian, then we just need to update mean and variance of the posterior in each iteration. This is the Kalman filtering method. However, the observation probability $p(\mathbf{y}_n|\mathbf{s}_n)$ is very complicated. It may not be Gaussian and may not even be unimodal. Therefore, we need to update the whole probability distribution. This is not easy, since integration at each iteration is hard to calculate.

Particle filtering [**27, 2**] is a way to solve this problem, where the posterior distribution is represented and updated using a fixed number of particles together with their importance weights. In the score follower, I use the *bootstrap filter*, one variant of particle filters, which assigns equal weight to all particles in each iteration.

Algorithm 4 presents the algorithm applied to score following. In Line 1, M particles are initialized to have score positions equal to the first beat and tempi assume a uniform distribution. Line 3 starts the iteration through the frames of audio. At this point, these particles represent the posterior distribution $p(\mathbf{s}_{n-1}|\mathbf{Y}_{1:n-1})$ of $\mathbf{s}_{n-1}$. From Line 4 to 9, particles are updated according to the process model in Eq. (5.1) and (5.2) and now they represent the conditional distribution $p(\mathbf{s}_n|\mathbf{Y}_{1:n-1})$ of $\mathbf{s}_n$. In Line 10 and 11, the importance weights of particles are calculated as their observation likelihood according to Eq. (5.3), and then normalized to a discrete distribution. Then in Line 12, these particles are resampled with replacement according to their weights to generate a new set of M particles. This is the key step of a bootstrap filter, after which the new particles can be

Algorithm 4: InferenceByParticleFiltering

Input : N: number of frames of audio; $\mathbf{y}_1, \cdots, \mathbf{y}_N$: audio power spectra of all frames; v^l, v^h: the lowest and highest possible tempo of audio; z_k: onset time (beats) of the k-th score note; σ_v: standard deviation of the noise term n_v; M: number of particles in each iteration.

Output: x_n, v_n: score position and tempo of each audio frame $n = 1, \cdots, N$.

1 Initialize M particles $(x^{(1)}, v^{(1)}), \cdots, (x^{(M)}, v^{(M)})$, where $x^{(i)} = 1$ and $v^{(i)} \sim U[v^l, v^h]$;
2 **for** $n \leftarrow 1, \cdots, N$ **do**
3 $\quad x^{(i)} \leftarrow x^{(i)} + l \cdot v^{(i)}$ by Eq. (5.1);
4 $\quad$ **if** $x_{n-2} \leq z_k \leq x_{n-1}$ *for some* k **then**
5 $\quad\quad v^{(i)} \leftarrow v_{n-1} + n_v$ by Eq. (5.2);
6 $\quad$ **end**
7 $\quad$ **else**
8 $\quad\quad v^{(i)} \leftarrow v^{(i)}$;
9 $\quad$ **end**
10 $\quad w^{(i)} \leftarrow p(\mathbf{y}_n|(x^{(i)}, v^{(i)}))$ by Eq. (5.3);
11 $\quad w^{(i)} \leftarrow w^{(i)}/\Sigma w^{(i)}$;
12 $\quad$ Sample particles with replacement according to $w^{(i)}$ to get a new set of particles $(x^{(1)}, v^{(1)}) \cdots, (x^{(M)}, v^{(M)})$;
13 $\quad x_n = \frac{1}{M}\Sigma x^{(i)}$ and $v_n = \frac{1}{M}\Sigma v^{(i)}$;
14 $\quad$ Output x_n and v_n;
15 **end**
16 **return** the estimated score position x_n and tempo v_n of all frames $n = 1, \cdots, N$;

thought of having equal weights. These particles now represent the posterior distribution $p(\mathbf{s}_n|\mathbf{Y}_{1:n})$ of $\mathbf{s}_n$, and we calculate their mean as the score position and tempo estimate in the n-th frame in Line 13.

In updating the tempo of each particle in Line 6, instead of using its previous tempo $v^{(i)}$, I use the previously estimated tempo v_{n-1}, i.e. the average tempo of all particles in the previous frame. This practical choice avoids that the particles become too diverse after a number of iterations due to the accumulation of randomness of n_v.

The set of particles is not able to represent the distribution if there are too few, and is time-consuming to update if there are too many. In this paper, we tried to use 100, 1,000 and 10,000 particles. I find that with 100 particles, the score follower is often lost after a number of frames. But with 1000 particles, this rarely happens and the update is still fast enough. Therefore, 1000 particles are used in this paper.

Unlike some other particle filters, the bootstrap filter I use does not have the common problem of degeneracy, where most particles have negligible importance weights after a few iterations [**27, 2**]. This is because the resampling step (Line 12 in Algorithm 4) in each iteration eliminates those particles whose importance weights are too small, and the newly sampled particles have equal weights again. This prevents the skewness in importance weights from accumulating.

At each time step, the algorithm outputs the mean score position from the set of particles as the estimate of the current score position. Someone may suggest choosing MAP or median, since the mean value may lie in a low probability area if the distribution is not unimodal. However, I find that in practice there is not much difference in choosing mean, MAP or median. This is because the particles in each iteration generally only cover a small range of the score (usually less than 0.5 beat), and mean, MAP and median are close.

5.4. Algorithm Analysis

Algorithm 4 is an online algorithm that makes a Markovian assumption. It considers only the result of the previous time-frame and the current spectrum in calculating its output. Therefore the number of operations performed at each frame is bounded by a

constant value in terms of the number of past frames. I analyze this constant in terms of the number of particles M (on the order of 1000 in our implementation), the number of spectral peaks in the mixture K (on the order of 100), and the number of sources J (typically less than 10).

Line 4-9 and 11-13 in Algorithm 4 all involve $O(M)$ calculations. Line 10 involves M times observation likelihood calculations, each of which calculates a multi-pitch likelihood of the chord at the score position of the particle. However, these M particles usually only cover a short segment (less than 0.5 beats) of the score. Within the span of a beat there are typically few note changes (16 would be an extreme). Therefore there are usually a small number of potential pitch sets to estimate the likelihood of (less than 16). Therefore, we only need a few likelihood calculations, each of which is of $O(K + J)$ according to Section 2.5. The number of sources J is much smaller than the number of spectral peaks K and can be ignored, so the algorithm requires in total $O(M + K)$ calculations.

5.5. Experiments

5.5.1. Datasets

A dataset to evaluate the proposed score following method must have the following components: a single-channel polyphonic music audio, its MIDI score, and the ground-truth alignment between the audio and the score. In this work, I use two datasets, one synthetic and one real. The synthetic dataset is adapted from Ganseman [**49**]. It contains 20 single-line MIDI melodies made from random note sequences. Each melody is played by a different instrument (drawn from a set of sampled acoustic and electric instruments). Each melody is 10 seconds long and contains about 20 notes. Each MIDI melody has

Table 5.3. Statistics of 11 audio performances rendered from each monophonic MIDI in Ganseman's dataset [**49**].

Max tempo deviation	0%	10%	20%	30%	40%	50%
Max tempo fluctuation	1.00	1.20	1.43	1.71	2.06	2.50
Num. of performances	1	2	2	2	2	2

a single tempo but is rendered to 11 audio performances with different dynamic tempo curves, using Timidity++ with the FluidR3 GM soundfont on Linux. The statistics of the audio renditions of each melody are presented in Table 5.3. "Max tempo deviation" measures the maximal tempo deviation of the rendition from the MIDI. "Max tempo fluctuation" measures the maximal relative tempo ratio within the dynamic tempo curve.

I use these monophonic MIDI melodies and their audio renditions to generate polyphonic MIDI scores and corresponding audio performances, with polyphony ranging from 2 to 6[1]. For each polyphony, I generate 24 polyphonic MIDI pieces by randomly selecting and mixing the 20 monophonic MIDI melodies. I generate 24 corresponding audio pieces, 4 for each of the 6 classes of tempo variations. Therefore, there are in total 120 polyphonic MIDI pieces with corresponding audio renditions. Alignment between MIDI and audio can be obtained from the audio rendition process and are provided in [**49**]. Although this dataset is not musically meaningful, I use it to test the proposed algorithm on audio mixtures with different polyphonies and tempi, which are two important factors in following polyphonic music. In addition, the large variety of instruments in this dataset lets us see the adaptivity of the proposed algorithm to different timbres.

The second dataset is the Bach10 dataset, as described in Section 2.6.1. It consists of 10 J.S. Bach four-part chorales played by a quartet of instruments: violin, clarinet, tenor

[1]Note that sources in this paper are all monophonic, so polyphony equals the number of sources.

saxophone and bassoon. Each musician's part was recorded in isolation while the musician listened to the others through headphones. I also created audio files by mixing the parts to contain all duets and trios for each piece, totalling 60 duets and 40 trios. The scores were MIDI downloaded from the internet[2]. The ground-truth alignment between MIDI and audio was interpolated from annotated beat times of the audio. The annotated beats were verified by a musician through playing back the audio together with these beats. I note that, beside the general tempo difference between the audio and MIDI pieces, there is often a fermata after a musical phrase in the audio but not in the MIDI. Therefore, there are many natural tempo changes in the audio while the MIDI has a constant tempo. I use this dataset to test the proposed algorithm in a more realistic situation.

5.5.2. Error Measure

I use *Align Rate (AR)* as proposed in [**18**] to measure the audio-score alignment results. For each piece, AR is defined as the proportion of correctly aligned notes in the score. This measure ranges from 0 to 1. A score note is said to be correctly aligned if its onset is aligned to an audio time which deviates less than 50ms from the true audio time. It is noted that MIREX[3] uses average AR (called *Piecewise Precision*) of pieces in a test dataset to compare different score following systems.

I also propose another metric called *Average Alignment Error (AAE)*, which is defined as the average absolute difference between the aligned score position and the true score

[2] `http://www.jsbchorales.net/index.shtml`

[3] The Music Information Retrieval Evaluation eXchange (MIREX) is an annual evaluation campaign for Music Information Retrieval (MIR) algorithms. Score Following is one of the evaluation tasks. `http://www.music-ir.org/mirex`

position of each frame of the audio. The unit of AAE is musical beat and it ranges from 0 to the maximum number of beats in the score.

I argue that AR and AAE measure similar but different aspects of an alignment. Notice that AR is calculated over note onsets in the audio time domain, while AAE is calculated over all audio frames in the score time domain. Therefore, AR is more musically meaningful and more appropriate for applications like real-time accompaniment. For example, if an alignment error of of a note is 0.1 beats, then the corresponding alignment error in the audio time domain can be either 100ms if the tempo is 60BPM or 33.3ms if the tempo is 180BPM, which induce significantly different accompaniment perceptions. AAE, however, is more appropriate for applications like score-informed source separation, since not only note onsets but all audio frames need to be separated. In addition, AAE is well correlated with the accuracy of score-informed pitches given the typical lengths of notes in a piece of music, hence helps analyze the main factor of source separation errors. For example, suppose the shortest note is an eighth-note, then AAE of 0.2 beats will indicate a high accuracy of score-informed pitches, and the score following stage will not be the main factor causing source separation errors.

In [**18**] there is another important metric called "latency" to measure the time delay of an online score follower from detecting to reporting a score event. We do not need this metric since the score follower in Soundprism computes an alignment right after seeing the input audio frame and the computation time is negligible. Therefore, there is no inherent delay in the score follower. The only delay from the audio frame being performed to the aligned score position being output is the frame center hop size, which is 10ms in this work.

5.5.3. Comparison Methods

I compare the proposed score following method with *Scorealign*, an open-source offline audio-score alignment system[4] based on the method described in [**62**].

5.5.4. Results on the Synthetic Dataset

Table 5.4 shows the score alignment results of the proposed method and Scorealign for different polyphony on the synthetic dataset. It can be seen that Scorealign obtains higher than 50% average Align Rate (AR) and less than 0.2 beats Average Alignment Error (AAE) for all polyphony, while the results of the proposed method are significantly worse, especially for polyphony 2. However, as polyphony increases, the gap between the proposed method and Scorealign is significantly reduced. This supports our claim that the proposed method works better for high polyphony pieces.

Table 5.4. Audio-score alignment results (Average±Std) versus polyphony on the synthetic dataset. Each value is calculated from 24 musical pieces.

metric	AR (%)		AAE (beat)	
polyphony	Proposed	Scorealign	Proposed	Scorealign
2	27.6±17.3	50.1±27.4	0.60±0.64	0.15±0.08
3	36.3±16.5	51.6±24.2	0.25±0.20	0.13±0.07
4	41.4±13.7	53.9±23.3	0.21±0.09	0.15±0.09
5	47.0±18.7	60.8±20.1	0.24±0.10	0.16±0.09
6	49.8±19.6	55.5±23.8	0.30±0.23	0.18±0.09

Table 5.5 indicates that the proposed method slowly degrades as the tempo variation increases, but not as quickly as Scorealign. For the proposed method on tempo variation from 0% to 30%, AR are around 45% and AAE are around 0.25 beats. Then they degrades to about 30% of AR and 0.4 beats of AAE. Results of Scorealign, however, obtains almost

[4] `http://sourceforge.net/apps/trac/portmedia/wiki/scorealign`

Table 5.5. Audio-score alignment results versus tempo variation on the synthetic dataset.

metric	AR (%)		AAE (beat)	
tempo	Proposed	Scorealign	Proposed	Scorealign
0%	47.1±22.0	96.6±5.4	0.28±0.39	0.01±0.01
10%	51.6±18.9	38.7±16.9	0.31±0.49	0.18±0.05
20%	44.3±16.4	50.5±11.5	0.22±0.07	0.16±0.06
30%	41.5±14.2	51.6±12.5	0.25±0.12	0.16±0.04
40%	27.9±13.0	48.2±17.9	0.46±0.49	0.19±0.07
50%	30.0±15.7	40.7±14.9	0.39±0.25	0.22±0.05

perfect alignment on pieces with no tempo variation. Then it degrades suddenly to about 50% of AR and 0.18 beats of AAE. Remember that in the case of 50% tempo variation, the tempo of the fastest part of the audio performance is 2.5 times of the slowest part (refer to Table 5.3), while the score tempo is a constant. This is a very difficult case for online audio-score alignment.

5.5.5. Results on the Real music dataset

Table 5.6. Audio-score alignment results versus polyphony on the Bach chorale dataset.

metric	AR (%)		AAE (beat)	
polyphony	Proposed	Scorealign	Proposed	Scorealign
2	53.8±13.9	45.1±9.2	0.17±0.16	0.19±0.04
3	60.6±12.7	45.7±8.5	0.13±0.03	0.19±0.04
4	69.3±9.3	46.6±8.7	0.12±0.03	0.15±0.05

Table 5.6 shows audio-score alignment results versus polyphony when measured on real human performances of Bach chorales. Here, the proposed method performs better than Scorealign on both AR and AAE. This may indicate that the proposed method is more adapted for real music pieces than pieces composed of random notes. More interestingly, the average AAE of the proposed method decreases from 0.17 to 0.12 when polyphony

increases. Again, this suggests the ability of dealing with high polyphony of the proposed method. In addition, the average AAE of the proposed method is less than a quarter beat for all polyphony. Since the shortest notes in these Bach chorales are sixteenth notes, the score follower is able to find correct pitches for most frames.

5.6. Conclusions

In this chapter I proposed an online algorithm for audio-score alignment of polyphonic music composed of harmonic sources. This is an essential step towards MASA informed by a score. I use a hidden Markov process to model the audio performance. The state space is defined as a 2-d space of score position and tempo. The observation model is defined as the multi-pitch likelihood of each frame, i.e. the likelihood of seeing the audio frame given the pitches at the aligned score position. Particle filtering is employed to infer the score position and tempo of each audio frame in an online fashion.

I performed experiments on both synthetic audio and real music performances and compared it with a well-known offline audio-score alignment method. Results showed that the proposed method can deal with multi-instrument music with high polyphony and some degree of tempo variation. Interestingly, the proposed method performs better when the polyphony increases from 2 to 6. However, it degrades significantly when the tempo variation of the performance increases. Results also showed that the proposed method achieves worse results on the synthetic audio dataset, but better results on the real music dataset, than the comparison offline method.

For future work, I want to incorporate some onset-like features in the observation model of the proposed method, to improve the alignment accuracy. I also want to improve

the its robustness, to deal with the situation that performers occasionally make mistakes and deviate from the score.

CHAPTER 6

Score-informed Source Separation

6.1. Introduction

Given an alignment of the music audio and the score, we know the corresponding score position of each audio frame. If this estimate is correct, it can help us analyze the music audio scene. The problem of separating the source signals of a polyphonic music audio provided an aligned score is called Score-informed Source Separation (SISS).

As I focus on polyphonic music composed of harmonic sound sources, I look for pitch information in the score. Basically, the aligned score tells us what pitch (if any) is supposed to be played by each source in each frame. Although the score-suggested pitch would not be exactly the same as the audio-performed pitch, it can help us more accurately estimate the audio-performed pitch. A good estimate of the audio-performed pitch can then help separate the harmonic source from the mixture.

Similar to the audio-score alignment algorithm proposed in Chapter 5, I do score-informed source separation in an online fashion. With an online algorithm, both online and offline applications are possible.

6.1.1. Related Work

For score-informed source separation, there exist several approaches in the literature [**138, 103, 48, 80, 43**]. Woodruff et al. [**138**] work on stereo music, where spatial cues

are utilized together with score-informed pitch information to separate sources. This approach does not apply to my problem, since I am working on single-channel source separation. Raphael [**103**] trains a model to classify time-frequency bins that belong to solo or accompaniment using a labeled training set, then applies this model to separate the solo from the mixture. This method, however, cannot separate multiple sources from the mixture.

[**48, 80, 43**] address single-channel source separation when a well-aligned score is provided. Ganseman et al. [**48**] use Probabilistic Latent Component Analysis (PLCA) to learn a piece-specific source model from the synthesized audio of the source's score, and then apply these source models to real audio mixtures. In order to obtain good separation results, the synthesized audio should have similar timbre to the source signal, so that the learned source models can properly represent the real sources. However, this requirement is often too strong as the timbral difference between a synthesizer and a real instrument can be large for some instruments. Li et al. [**80**] first refine score-indicated pitches for the audio and then uses a least-square framework to estimate their harmonics. The key assumption is that harmonics of the same note have common amplitude modulations over time. This assumption, albeit an important human auditory perception cue, can be problematic for some instruments. For example, the decay rates of higher frequency harmonics of a piano note are much faster than those of the lower frequency harmonics.

Ewert and Müller [**43**] make less assumptions. They use Nonnegative Matrix Factorization (NMF) to separate the left and right hand performances of piano, where each pitch has a harmonic basis spectrum and an onset basis spectrum. They use the score to initialize the harmonic spectra and their activation weights, then the NMF procedure adapts

them for each specific piece. The basic assumption is that notes of the same pitch (and same instrument) share the same basis spectra. This assumption is reasonable, however, the number of parameters to model the sources is quite large. For a quartet of different instruments, suppose each instrument plays 20 different pitches in this piece and suppose the dimensionality of basis spectra is 1024 (equivalent to 46ms long window with 44.1kHz sampling rate). Then there would be in total $4 \times 20 \times 1024 = 81920$ parameters to model these sources.

Besides their individual disadvantages, [**48, 80, 43**] are all offline algorithms. They all require the entire audio piece before separation. Therefore, they cannot be applied in my scenario, where I want to build an online MASA system. In the following, I will propose an online score-informed source separation method.

6.1.2. Advances of the Proposed System

I have published the preliminary online source separation algorithm in [**31**]. Overall, the proposed method has the following advantages:

- It works online and can be easily implemented to work in real time.
- No training of the specific piece is needed before separation.

6.2. Refining Score Pitches

The pitches provided by the score $\boldsymbol{\theta} = \{F0_1, \cdots, F0_J\}$ are integer MIDI pitch numbers. MIDI pitch numbers indicate keys on the piano keyboard. Typically, MIDI 69 indicates the A above Middle C. Assuming A440-based equal temperament allows translation from MIDI pitch to frequency in Hz. The resulting frequencies are rarely equal to the real

pitches played in an audio performance. In order to extract harmonics of each source in the audio mixture, we need to refine them to get accurate estimates $\hat{\boldsymbol{\theta}} = \{\hat{F0}_1, \cdots, \hat{F0}_J\}$. I refine the pitches using the multi-pitch estimation algorithm as described in [**33**], but restricting the search space in the Cartesian product $\prod_i [F0_i - 50\text{cents}, F0_i + 50\text{cents}]$. The algorithm maximizes the multi-pitch likelihood $\mathcal{L}(\hat{\boldsymbol{\theta}})$ in Eq. (2.2) with a greedy strategy, i.e. refining (estimating) pitches one by one. The set of refined pitches $\hat{\boldsymbol{\theta}}$ starts from an empty set. In each iteration, the refined pitch that improves the likelihood most is added to $\hat{\boldsymbol{\theta}}$. Finally, I get the set of all refined pitches. In refining each pitch $F0_i$, we search $\hat{F0}_i$ in $[F0_i - 50\text{cents}, F0_i + 50\text{cents}]$ with a step of 1Hz.

6.3. Reconstruct Source Signals

For each source in the current frame, I build a ratio frequency mask and multiply it with the magnitude spectrum of the mixture signal to obtain the magnitude spectrum of the source. Then I apply the original phase of the mixture to the magnitude spectrum to calculate the source's time-domain signal. Finally, the overlap-add technique is applied to concatenate the current frame to previously generated frames. The sum of the masks of all the sources equals one in each frequency bin, so that the sources sum up to the mixture.

In order to calculate masks for sources, I first identify their harmonics and overlapping situations from the estimated pitches. For each source, I only consider the lowest 20 harmonics, each of which covers 40Hz in the magnitude spectrum. This width is assumed to be where the main lobe of each harmonic decreases 6dB from the center, when I use a 46ms Hamming window. These harmonics are then classified into overlapping harmonics

and non-overlapping harmonics, according to whether the harmonic's frequency range is overlapped with some other harmonic's frequency range of another source.

All frequency bins in the spectrum can then be classified into three kinds: a *nonharmonic bin* which does not lie in any harmonic's frequency range of any source, a *non-overlapping harmonic bin* which lies in a non-overlapping harmonic's frequency range and an *overlapping harmonic bin* which lies in an overlapping harmonic's frequency range. For different kinds of bins, masks are calculated differently.

For a nonharmonic bin, masks of all active sources are set to $1/J$, where J is the number of pitches (active sources) in the current frame. In this way the energy of the mixture is equally distributed to all active sources. Although energy in nonharmonic bins is much smaller than that in harmonic bins, experiments show that distributing the energy reduces artifacts in separated sources, compared to discarding it. For a non-overlapping harmonic bin, the mask of the source that the harmonic belongs to is set to 1 and the energy of the mixture is assigned entirely to it.

For an overlapping harmonic bin, the masks of the sources whose harmonics are involved in this overlapping situation, are set in inverse proportion to the square of their harmonic numbers (e.g. 3 is the harmonic number of the third harmonic). For example, suppose a bin is in a harmonic which is overlapped by $J-1$ harmonics from other sources. Then the mask of the i-th source in this bin is defined as

$$m_i = \frac{1/h_i^2}{\Sigma_{j=1}^{J} 1/h_j^2}, \tag{6.1}$$

where h_k is the harmonic number of the k-th source.

This simple method to resolve overlapping harmonics corresponds to the assumption that 1) overlapping sources have roughly the same amplitude; 2) all notes have harmonics amplitudes decay at 12dB per octave from the fundamental, regardless of pitch and instrument that produced the note. These assumptions are very coarse and will never be fulfilled in the real world. One can improve upon these assumptions by designing a more delicate source filter [**42**], interpolating the overlapping harmonics from non-overlapping harmonics based on the spectral smoothness assumption in each frame [**127**], or the temporal envelope similarity assumption of different harmonics of one note [**130**] or both [**142**]. Nevertheless, Eq. (6.1) gives a simple and relatively effective way to resolving overlapping harmonics as shown in experiments.

6.4. Experiments

6.4.1. Datasets

I use the same synthetic and real music performance datasets as described in Section 5.5.1 to evaluate the proposed score-informed source separation method.

6.4.2. Error Measure

I use the BSS_EVAL toolbox [**124**] to evaluate the separation results of the proposed method. Basically, each separated source is decomposed into a true source part and error parts corresponding to interferences from other sources and algorithmic artifacts. By calculating the energy ratios between different parts, the toolbox gives three metrics: *Signal-to-Interference Ratio (SIR)*, *Signal-to-Artifacts Ratio (SAR)* and *Signal-to-Distortion Ratio (SDR)* which measures both interferences and artifacts.

6.4.3. Input Alignment

I feed the proposed method with two kinds of input alignments. The first one is the estimated audio-score alignment from Chapter 5. Source separation achieved in this setting is the output of the proposed score-informed MASA system (the second part of the dissertation). It is denoted by "Soundprism" (refer Section 7.1. The second one is the ground-truth audio-score alignment. This removes the influence of the score follower and evaluates the proposed source separation method only. It is denoted by "Ideally-aligned".

6.4.4. Comparison Methods

I also compare the proposed method with three other source separation methods. *Ganseman10* is a score-informed source separation system proposed by Ganseman et al. [**48, 49**]. I use their own implementation. This system first aligns audio and score in an offline fashion, then uses a Probabilistic Latent Component Analysis (PLCA)-based method to extract sources according to source models. Each source model is learned from the MIDI-synthesized audio from the source's score. For the synthetic dataset, these audio pieces are provided by Ganseman. For the real music dataset, these audio pieces are synthesized using the Cubase 4 DAW built-in synthesis library without effects. Instruments in the synthesizer are selected to be the same as the audio mixture, to make the timbre of each synthesized source audio as similar as possible to the real source audio. However, in real scenarios that the instruments of the sources are not recognizable, the timbre similarity between the synthesized audio and the real source cannot be guaranteed and the system may degrade.

MPESS is a source separation system based on multi-pitch estimation and streaming in Chapter 2 and 3. From the pitch estimate of each source at each frame, this system applies the proposed source separation method in Section 6.3 to separate the source signal. MPESS gives source separation results based on the proposed audio-only MASA system (the first part of the dissertation). No score information is utilized here.

Oracle separation results are calculated using the BSS_Oracle toolbox [**125**]. They are the theoretically, highest achievable results of the time-frequency masking-based methods and serve as an upper bound of source separation performance. It is noted that oracle separation can only be obtained when the reference sources are available.

6.4.5. Results on the Synthetic Dataset

Figure 6.1 shows boxplots of the overall separation results of the five separation systems on pieces of polyphony 2. Each box represents 48 data points, each of which corresponds to the audio from one instrumental melody in a piece.

For pieces of polyphony 2, if the two sources are of the same loudness, then the SDR and SIR of each source in the unseparated mixture should be 0dB. It can be seen that Soundprism improves the median SDR and SIR to about 5.5dB and 12.9dB respectively. Ideal alignment further improves SDR and SIR to about 7.4dB and 15.0dB respectively. This improvement is statistically significant in a nonparametric sign test with $p < 10^{-6}$. This suggests that the score following stage of Soundprism has space to improve. Comparing Soundprism with Ganseman10, we can see that they get similar SDR ($p = 0.19$) and SAR ($p = 1$) while Ganseman10 gets significant higher SIR ($p < 10^{-7}$). But remember that Ganseman10 uses an offline audio-score alignment and needs to learn a source

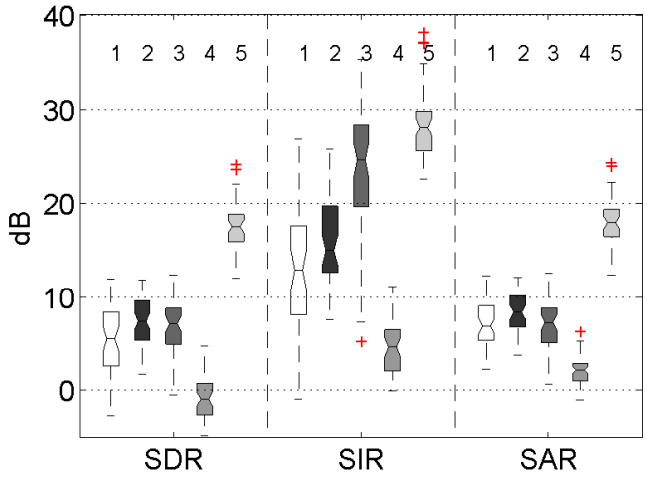


Figure 6.1. Separation results on pieces of polyphony 2 from the synthetic dataset for 1) Soundprism, 2) Ideally-aligned, 3) Ganseman10, 4) MPESS and 5) a perfect Oracle. Each box represents 48 data points, each of which corresponds to an instrumental melodic line in a musical piece from the synthetic data set. Higher values are better.

model from MIDI-synthesized audio of each source. Without using score information, MPESS obtains significantly worse results than all the three score-informed source separation systems. This supports the idea of using score information to guide separation. Finally, Oracle results are significantly better than all the other systems. Especially for Ideally-aligned, this gap of performance indicates that the separation stage of Soundprism has plenty of room to improve.

Figure 6.2 shows SDR comparisons for different polyphony. SIR and SAR comparisons are omitted as they have the same trend as SDR. It can be seen that when polyphony increases, the performance difference between Soundprism and Ideally-aligned gets smaller. This is to be expected, given that Table 5.4 shows our score following stage performs better for higher polyphony. Conversely, the difference between Soundprism and Ganseman10

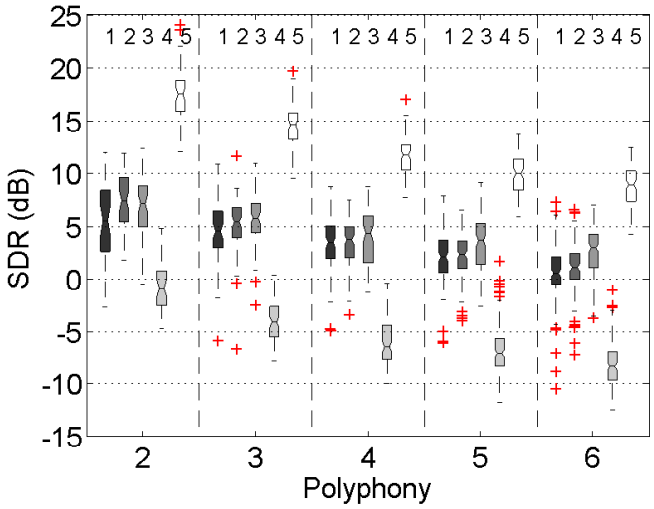


Figure 6.2. SDR versus polyphony on the synthetic dataset for 1) Soundprism, 2) Ideally-aligned, 3) Ganseman10, 4) MPESS and 5) Oracle. Each box of polyphony n represents $24n$ data points, each of which corresponds to one instrumental melodic line in a musical piece.

gets larger. This suggests that pre-trained source models are more beneficial for higher polyphony. Similarly, the performance gap from MPESS to the three score-informed separation systems gets larger. This suggests that score information is more helpful for higher polyphony pieces.

The good results obtained by Scorealign helps the separation results of Ganseman10, as they use the same audio-score alignment algorithm. However, as the SDR obtained by Soundprism and Ganseman10 in Figure 6.1 and 6.2 are similar, the performance difference of their audio-score alignment stages is not vital to the separation results.

It is also interesting to see how score-informed separation systems are influenced by the tempo variation of the audio performance. Figure 6.3 shows this result. It can be seen that the median SDR of Soundprism slowly degrades from 2.8dB to 1.9dB as the max tempo deviation increases from 0% to 50%. A two sample t-test with $\alpha = 0.05$

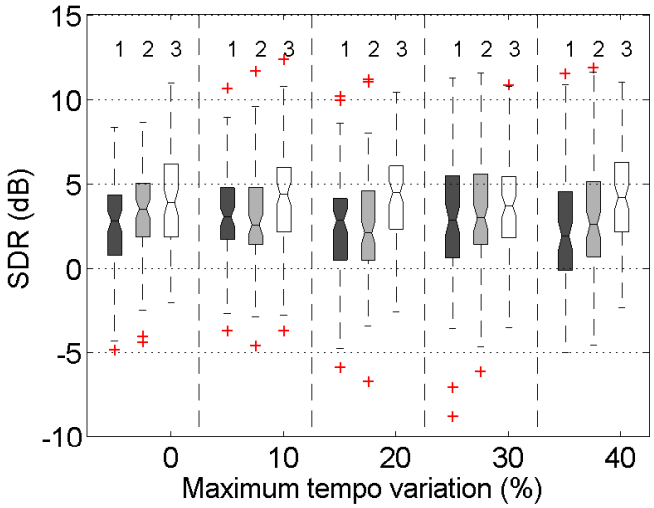


Figure 6.3. SDR versus tempo variation on the synthetic dataset for 1) Soundprism (dark gray), 2) ideally-aligned (light gray) and 3) Ganseman10 (white). Each box represents 80 data points, each of which corresponds to one instrumental melodic line in a musical piece.

shows the mean SDR of the first 5 cases are not significantly different, while the last one is significantly worse. This supports the conclusion that the score following stage of Soundprism slowly degrades as the tempo variation increases, but not much.

6.4.6. Results on the Real Music Dataset

Figure 6.4 first shows the overall results on pieces of polyphony 2. There are four differences from the results of the synthetic dataset in Figure 6.1. First, the results of Soundprism and Ideally-aligned are very similar on all measures. This suggests that the score following stage of Soundprism performs well on these pieces. Second, the difference between Soundprism/Ideally-aligned and Oracle is not that great. This indicates that the separation strategy used in Section 6.3 is suitable for the instruments in this dataset. Third, Soundprism obtains a significantly higher SDR and SAR than Ganseman10 while

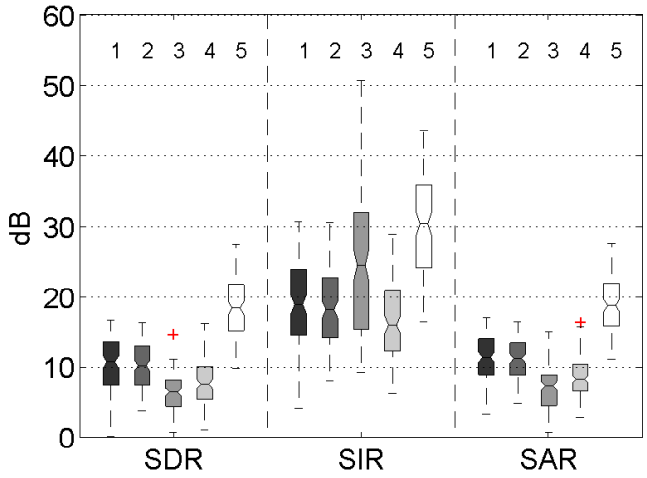


Figure 6.4. Separation results on pieces of polyphony 2 from the Bach chorale dataset for 1) Soundprism, 2) Ideally-aligned, 3) Ganseman10, 4) MPESS and 5) Oracle. Each box represents 120 data points, each of which corresponds to one instrumental melodic line in a musical piece.

a lower SIR. This indicates that Ganseman10 performs better in removing interference from other sources while Soundprism introduces less artifacts and leads to less overall distortion. Finally, the performance gap between MPESS and the 3 score-informed source separation systems is significantly reduced. This means that the multi-pitch tracking results are more reliable on real music pieces than random note pieces. But still, utilizing score information improves source separation results.

Figure 6.5 shows results for different polyphony. We can see that Soundprism and Ideally-aligned obtain very similar results for all polyphony. This suggests that the score following stage performs well enough for the separation task on this dataset. In addition, Soundprism obtains a significantly higher SDR than Ganseman10 for all polyphony ($p < 10^{-7}$). Furthermore, MPESS degrades much faster than the three score-informed

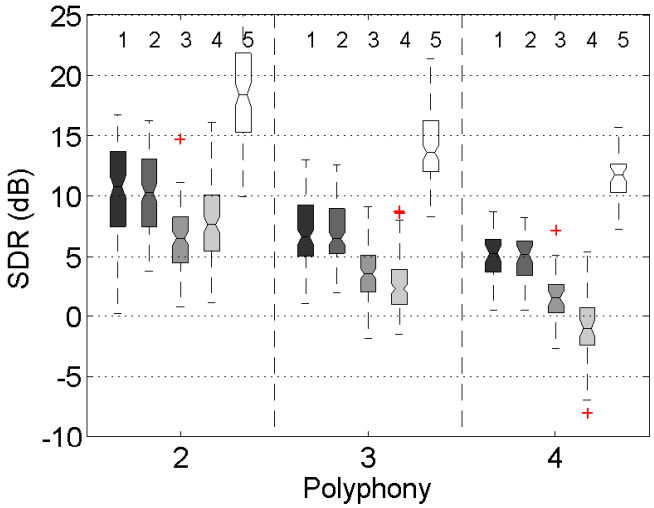


Figure 6.5. SDR versus polyphony on the Bach chorale dataset for 1) Soundprism, 2) Ideally-aligned, 3) Ganseman10, 4) MPESS and 5) Oracle. Each box of polyphony 2, 3 and 4 represents $2 \times 60 = 120$, $3 \times 40 = 120$ and $4 \times 10 = 40$ data points respectively, each of which corresponds to one instrumental melodic line in a musical piece.

separation systems, which again indicates that score information is more helpful in the pieces with higher polyphony.

The SDR of polyphony 4 showed in Figure 6.5 are calculated from all tracks of all quartets. However, for the same piece of a quartet, different instrumental tracks have different SDRs. A reasonable hypothesis is that high frequency tracks have lower SDR since they have more harmonics overlapped by other sources. However, Figure 6.6 shows opposite results. It can be seen that Track 1, 2 and 3 have similar SDRs, but Track 4 has a much lower SDR. This may suggest that the energy distribution strategy used in Section 6.3 biases to the higher-pitched source.

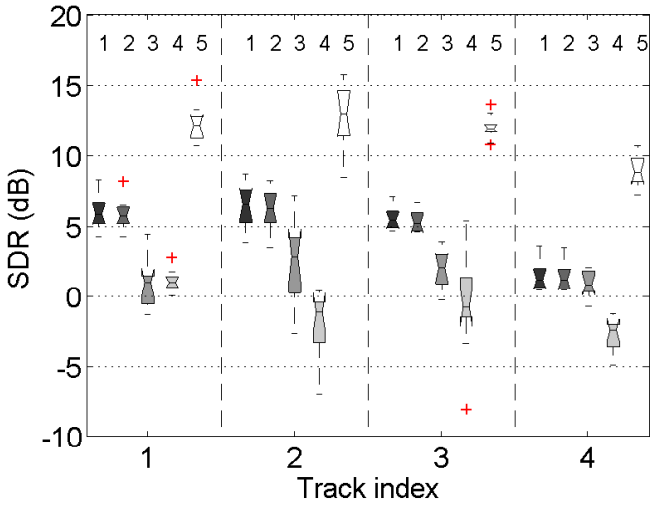


Figure 6.6. SDR versus instrumental track indices on pieces of polyphony 4 in the Bach chorale dataset for 1) Soundprism, 2) Ideally-aligned, 3) Ganseman10, 4) MPESS and 5) Oracle. Tracks are ordered by frequency, i.e., in a quartet Track 1 is soprano and Track 4 is bass.

6.4.7. Commercially recorded music examples

I test Soundprism and its comparison systems on two commercial recordings of music pieces from the RWC database [**54**]. These pieces were not mixed from individual tracks, but recorded directly as a whole from an acoustic environment. Therefore, we do not have the ground-truth sources and alignments, hence cannot calculate measures. The separated sources of these pieces can be downloaded from `http://www.cs.northwestern.edu/~zdu459/jstsp2011/examples`. This webpage also contains several examples from the Bach chorale dataset.

6.5. Conclusions

In this chapter, I proposed a simple online score-informed source separation method for polyphonic music composed of harmonic sources. Given an alignment between the

music audio and its score, I first refined the score-informed pitches in each audio frame. Then I separated the source signals from their pitches through time-frequency masking. Overlapping harmonics are resolved by assigning the mixture energy to each overlapping source in reverse proportion to the square of their harmonic numbers.

Experiments on both synthetic audio and real music performances showed that the proposed simple method achieves better separation than a state-of-the-art offline score-informed source separation algorithm. In addition, comparisons with the proposed multi-pitch estimation and streaming based source separation showed that the score information does significantly improve source separation.

Although the proposed method is simple and effective, the comparison with the Oracle separation showed that there is much room to improve its performance. One direction is to design a more advanced method to distribute the energy of overlapping harmonics. In the current method, no source models or timbre information are involved in the distribution. To improve it, one can learn a source model (e.g. a typical harmonic structure model for each source) from the mixture signal directly, by clustering all the harmonic structures of all pitches in all frames. This idea is similar to the multi-pitch streaming work proposed in Chapter 3. With a source model, the separated signals would be more realistic to the real source signals. This will be my future work.

CHAPTER 7

Applications

The proposed audio-score alignment algorithm in Chapter 5 and the score-informed source separation algorithm in Chapter 6 are both online algorithms. Online algorithms can be applied to offline scenarios as well. In this chapter, I propose two interesting applications of them, one in an online scenario and one in an offline scenario.

7.1. Online Application: Soundprism

In the online scenario, I imagine an interesting application called *Soundprism* as shown in Figure 7.1. Analogous to a prism which separates a ray of white light into multiple rays of light with different colors in real time, a soundprism separates a piece of polyphonic music audio into multiple sources in real time. To the best of my knowledge, there is no such existing system.

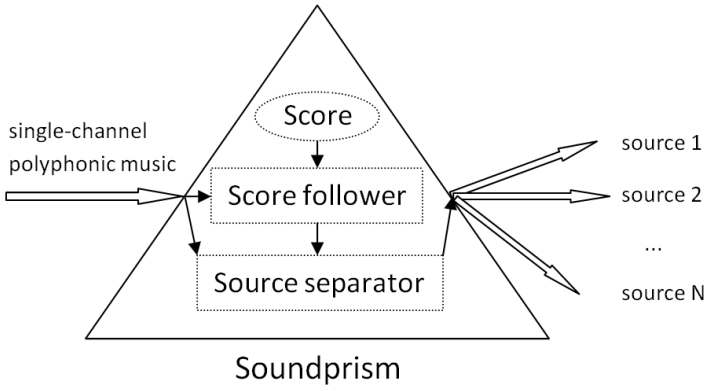


Figure 7.1. Illustration of the idea of Soundprism.

Soundprism can be helpful in many scenarios. For example, it would allow every audience member in a classical music concert to select their favorite personal mix (e.g., switch between enjoying the full performance and concentrating on the cello part) even though the instruments are not given individual microphones. Soundprism could also allow real-time remixing of CDs, live broadcasts or TV programs of monaural or stereo classical music.

Currently Soundprism is implemented as an offline system in Matlab, and it runs about 3 times slower than real time on a four-core 2.67GHz CPU under Windows 7. I expect to implement it on a mobile device and make it a truly real-time system.

7.2. Offline Application: Interactive Music Editing

In the offline scenario, one interesting application is interactive music editing. The idea is to let users edit musical objects in the polyphonic audio through a computer interface, as shown in Figure 7.2. A user can load the audio and its score, and align them through the interface. The proposed MASA system behind the interface will recognize notes indicated in the score from the audio, analyze their parameters such as loudness, pitch, onset, offset and timbre, and segregate their physical signals.

Then the user can perform two kinds of editing. The first kind is *Note-wise Editing*. The user can select a single note and change its parameters. For example, the user can correct the pitches of the out-of-tune notes, and change the timbre of the non-well-articulated notes to beautify the musical performances. The second kind is *Track-wise Editing*. The user can select a whole track of notes played by an instrument and change their parameters simultaneously. This kind of operation is interesting for audio engineers

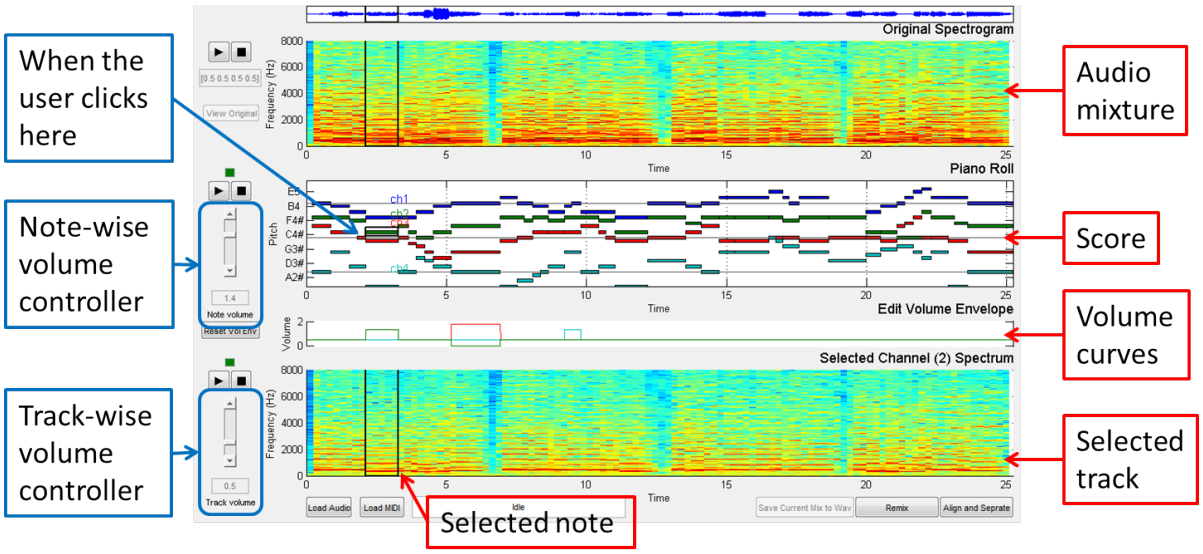


Figure 7.2. Interface of the interactive editing application. A user can edit the audio signal of a musical object (e.g. a note or an instrumental track) by selecting the object in the score and modifying its parameters such as loudness, pitch, onset, offset and timbre.

in music production. They can transpose an instrumental track from one scale to another, or change the timbre of the clarinet part to oboe. They can also copy a track and reuse it in other musical pieces.

There is a commercial interactive music editing software called "Melodyne" [1], which allows users to access individual notes of polyphonic music audio and modify their parameters. However, since Melodyne does not utilize score information, it fails in organizing notes according to sources and separating the sources when multiple instruments are playing. It can only deal with polyphonic music played by a single instrument (e.g. piano or guitar).

A prototype system of the interactive music editing application has been implemented in Matlab in a computer. Its interface is shown in Figure 7.2. It currently only supports

[1] `http://www.celemony.com/cms/`

note-wise and track-wise volume changes. I expect to extend them to more operations such as pitch, onset/offset and timbre edits.

CHAPTER 8

Conclusions

In this dissertation, I proposed computational models to address the music audio scene analysis problem. I focused on polyphonic music audio composed of harmonic sound sources, and addressed the problem in two scenarios.

In the first scenario, I developed a multi-pitch analysis system to analyze the music audio scene solely from the audio. Given a piece of polyphonic music audio, the system first estimates all the pitches of all sources in each audio frame. It then streams these pitch estimates into pitch streams, each of which corresponds to a source. From the estimated pitch streams, audio signals of each source or each note can be separated from the mixture. The system is general enough to deal with not only music audio, but any polyphonic audio composed of harmonic sources.

In the second scenario, I assumed the musical score is available. I developed a system to leverage the score information to analyze the music audio scene. The system first aligns the music audio with the score, then uses the score-provided pitch information to separate the sources of the audio mixture. The system performs in an online fashion. Compared to solely analyzing audio, the score information does significantly help improve the source separation performance, to a level that is promising for further applications. I therefore proposed two interesting applications, one being real-time music source separation and the other being interactive music editing.

8.1. Limitations

The biggest limitation of the proposed methods in my dissertation is that they were designed to only deal with harmonic sound sources. A lot of polyphonic music, however, use inharmonic sources such as percussive instruments extensively. Their spectral patterns do not show harmonic regularities, and when they are strong, they will mask the peak-harmonic-pitch relation of harmonic sources. This will deteriorate the performance of the proposed multi-pitch estimation method in Chapter 2, which is the foundation of the proposed systems. Inharmonic sources will also affect the performance of the the multi-pitch streaming method in Chapter 3. This method streams pitches through timbre modeling of harmonic sources. It cannot model the timbre of inharmonic sources, and the modeling of harmonic sources will also be interfered by the presence of inharmonic sources.

The audio-score alignment method proposed in Chapter 5 models the audio-score relation through the pitch content. It does not consider prominent inharmonic sound objects (such as drums) which could be used for the alignment. In fact, the presence of percussive instruments will interfere the modeling of the pitch content. Finally, the score-informed source separation method in Chapter 6 uses harmonic masking to separate the source signals. If there is only one inharmonic source in the mixture, its signal can be obtained after separating all the harmonic sources. But if there are more than one inharmonic sources, the proposed method would not work.

Although the proposed methods can work to some extent on quasi-inharmonic sources, as shown on the speech dataset in Chapter 4, their intrinsic design prevents their application to scenarios where strong inharmonic sources are present. They need to be combined

with other methods that are designed to model inharmonic sources to broaden their application.

The second limitation is on the first part of the dissertation. It cannot fully analyze the audio scene when polyphonic music instruments such as piano or guitar are present. The multi-pitch estimation method in Chapter 2 would still estimate simultaneous pitches in each frame, but the multi-pitch streaming method in Chapter 3 would not be able to output a single pitch trajectory of the polyphonic instrument. The cannot-link constraints prevents simultaneous pitches produced by the polyphonic instrument being clustered into the same pitch trajectory. Consequently, the source signal of the polyphonic instrument cannot be separated as a whole from the audio mixture. One way to address this problem would be removing the cannot-link constraints, but this would cause inaccuracies of pitch streaming of other monophonic instruments. It is noted that this limitation does not exist in the second part of the dissertation. This is because the pitch streaming information is provided by the score.

The third limitation is on the the audio-score alignment method proposed in Chapter 5. It fails for musical audio whose pitch content mismatches that of the score. This mismatch can be perceptually subtle, such as "staccato" notes. Staccato is a performing style of a note where the note is played shorter than then length written in the score. This affects the offset time of the note and the onset time is still performed as expected. Staccato note are common in many music pieces, and performances containing staccato notes are still considered faithful to the score. However, there does exist significant mismatch on the pitch content between the audio and score. In the score, the expected length of a note might be one beat long, but the staccato performance of this note might only last

for half a beat. Therefore, the audio frames after the offset of this staccato note would not contain the score-indicated pitch of this note. The relation between an audio frame and a score position is modeled through the multi-pitch observation model, which is very sensitive to the mismatch on the pitch content. This significantly affects the robustness of the alignment on these pieces.

8.2. Future Work

For future work, I would like to address the limitations stated in the previous section. To make the proposed multi-pitch analysis models applicable to audio with strong inharmonic sources, I would like to combine them with methods that model inharmonic sounds. In fact, I have been collaborating with my colleagues in this direction towards melody extraction of pop music. My colleagues' REPET method [**99, 98**] models repetitions of the time-frequency content of audio, which contains most of the inharmonic sounds and mostly belong to the accompaniment part. It does not model the harmonic content of music, which is important for the melody. Preliminary results have shown that combining my multi-pitch analysis method with REPET does achieve better melody extraction than using either method alone.

I would also like to extend the multi-pitch streaming work to inharmonic sounds. The basic idea of using timbre similarity to streaming sound objects of the same source should be universal for both harmonic and inharmonic sources. The challenge is to find effective timbre representations of inharmonic sounds. Take the problem of streaming a person's footsteps of a person as an example, there might be no regular pattern in the spectrum of a single footstep, but the evolution of a series of footstep spectra would probably make

them stand out from noise and characterize the person. This is because footsteps of the same person have similar temporal patterns due to his/her typical gait. The use of temporal patterns at the sound-object level (e.g. the regular pattern of footsteps) will also help stream them by sources.

To address the staccato note limitation in the audio-score alignment method, I propose to make the multi-pitch observation model be less sensitive at the perceptually less important parts such as offsets. This could be realized through an onset detection on the audio [**6**] and a weighting function on the multi-pitch observation model to weight more on the frames that are right after an onset less on the frames that are far from an onset. In fact, the onset information can also improve the alignment accuracy [**44, 65**]. Only using the multi-pitch information, the first audio frame in the sustain part of a note can be aligned to any position of the corresponding note with no difference. With onsets detected, I can require the first frame to be always aligned to the onset of a score note.

Besides addressing the limitations of my dissertation, I also would like to work on new models to deepen and broaden my research. The harmonic masking-based separation method proposed in Chapter 6 distributes the mixture spectral energy purely according the overlapping harmonic situations and harmonic indices. This method is very simple, but the separated spectra of each source may not resemble those of the original sources, and may not resemble each other either. For future work, I would like to incorporate source models to improve the separation performance. The source models could be learned beforehand from isolated recordings in a supervised way, or learned during separation in an unsupervised way (clustering). In fact, I have been collaborating with several researchers to learn a multi-excitation-filter model [**11**] for each source training recordings

in a supervised way, and to adapt the model to the test mixture. Preliminary results have shown significant improvement of the score-informed source separation results.

To broaden my research on fusing multimodal music processing, I would like to integrate other media especially visual information with audio. This will help analyze complex audio scenes which are difficult if only using audio information. For example, mouth localization has been shown helpful for speech recognition in noisy environments [**45**]. A more interesting question is whether we can use audio analysis to help with visual analysis. One scenario would be automatic pre-localization of the solo performer(s) in a concert for camera shooting purposes. Currently the cameramen need to be very familiar with the music and know how to read musical scores. With score following techniques, the computer knows where the music is on the score. It can then predict the next solo performer(s), and automatically turn the camera to the performer(s) or notify the cameramen.

References

[1] Anderson, E., Ed. *The Letters of Mozart and his Family*, 2nd ed. London: Macmillan, 1966.

[2] Arulampalam, M., Maskell, S., Gordon, N., and Clapp, T. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Processing 50*, 2 (2002), 174–188.

[3] Bach, F., and Jordan, M. Discriminative training of hidden Markov models for multiple pitch tracking. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2005), pp. 489–492.

[4] Bay, M., and Beauchamp, J. Harmonic source separation using prestored spectra. In *Proc. International Conference on Independent Component Analysis and Blind Source Separation (ICA), LNCS 3889* (2006), pp. 561–568.

[5] Bay, M., Ehmann, A. F., Beauchamp, J. W., Smaragdis, P., and Downie, J. S. Second fiddle is important too: pitch tracking individual voices in polyphonic music. In *Proc. International Society for Music Information Retrieval Conference (ISMIR)* (2012), pp. 319–324.

[6] Bello, J. P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., and Sandler, M. B. A tutorial on onset detection in music signals. *IEEE Trans. Speech Audio Process. 13*, 5 (2005), 1035–1047.

[7] Boersma, P. Praat, a system for doing phonetics by computer. *Glot International 5*, 9/10 (2001), 341–345.

[8] Bregman, A. *Auditory Scene Analysis: The Perceptual Organization of Sound*. The MIT Press, Cambridge, Massachusetts, 1990.

[9] Burred, J. J., Röbel, A., and Sikora, T. Dynamic spectral envelope modeling for timbre analysis of musical instrument sounds. *IEEE Trans. on Audio, Speech, and Language Processing 18*, 3 (2010), 663–674.

[10] Cano, P., Loscos, A., and Bonada, J. Score-performance matching using HMMs. In *Proc. International Computer Music Conference (ICMC)* (1999), pp. 441–444.

[11] Carabias-Orti, J., Virtanen, T., Vera-Candeas, P., Ruiz-Reyes, N., and Canadas-Quesada, F. Musical instrument sound multi-excitation model for non-negative spectrogram factorization. *IEEE Journal of Selected Topics in Signal Processing 5*, 6 (2011), 1144–1158.

[12] Casey, M., and Westner, A. Separation of mixed audio sources by independent subspace analysis. In *Proc. International Computer Music Conference (ICMC)* (2000), pp. 154–161.

[13] Chang, W.-C., Su, A. W. Y., Yeh, C., Robel, A., and Rodet, X. Multiple-f0 tracking based on a high-order HMM model. In *Proc. International Conference on Digital Audio Effects (DAFx)* (2008).

[14] Cherry, E. C. Some experiments on the recognition of speech, with one and two ears. *Journal of the Acoustic Society of America 25* (1953), 975–979.

[15] Chrissochoidis, I. London mozartiana: Wolfgang's disputed age & early performances of allegri's miserere. *The Musical Times 151*, 1911 (2010), 8389.

[16] Cont, A. Realtime audio to score alignment for polyphonic music instruments using sparse non-negative constraints and hierarchical HMMs. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2006), pp. 245–248.

[17] Cont, A. A coupled duration-focused architecture for real-time music-to-score alignment. *IEEE Trans. Pattern Analysis and Machine Intelligence 32*, 6 (2010), 974–987.

[18] Cont, A., Schwarz, D., Schnell, N., and Raphael, C. Evaluation of real-time audio-to-score alignment. In *Proc. International Conference on Music Information Retrieval (ISMIR)* (2007), pp. 315–316.

[19] Cooke, M., Hershey, J. R., and Rennie, S. Monaural speech separation and recognition challenge. *Computer Speech and Language 24* (2010), 1–15.

[20] Dannenberg, R. B. An on-line algorithm for real-time accompaniment. In *Proc. International Computer Music Conference (ICMC)* (1984), pp. 193–198.

[21] Davidson, I., Ravi, S., and Ester, M. Efficient incremental constrained clustering. In *Proc. ACM Conference on Knowledge Discovery and Data Mining (KDD)* (2007), pp. 240–249.

[22] Davy, M., Godsill, S. J., and Idier, J. Bayesian analysis of polyphonic Western tonal music. *Journal of the Acoustical Society of America 119* (2006), 2498–2517.

[23] de Cheveigné, A., and Kawahara, H. Multiple period estimation and pitch perception model. *Speech Commun. 27* (1999), 175–185.

[24] de Cheveigné, A., and Kawahara, H. Yin, a fundamental frequency estimator for speech and music. *Journal of the Acoustical Society of America 111* (2002), 1917–1930.

[25] Depalle, P., Garcia, G., and Rodet, X. Tracking of partials for additive sound synthesis using hidden markov models. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (1993), pp. 225–228.

[26] Dixon, S. Live tracking of musical performances using on-line time warping. In *Proc. International Conference on Digital Audio Effects (DAFx)* (Madrid, Spain, 2005).

[27] Doucet, A., de Freitas, N., and Gordon, N., Eds. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

[28] Duan, Z., Han, J., and Pardo, B. Song-level multi-pitch tracking by heavily constrained clustering. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2010), pp. 57–60.

[29] Duan, Z., Han, J., and Pardo, B. Multi-pitch streaming of harmonic sound mixtures. *IEEE Trans. Audio Speech Language Processing* (under review).

[30] Duan, Z., and Pardo, B. Aligning improvised music audio with its lead sheet. In *Proc. International Society for Music Information Retrieval Conference (ISMIR)* (2011), pp. 513–518.

[31] Duan, Z., and Pardo, B. Soundprism: an online system for score-informed source separation of music audio. *IEEE Journal of Selected Topics in Signal Processing 5*, 6 (2011), 1205–1215.

[32] Duan, Z., and Pardo, B. A state space model for online polyphonic audio-score alignment. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2011), pp. 197–200.

[33] Duan, Z., Pardo, B., and Zhang, C. Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions. *IEEE Trans. Audio Speech Language Processing 18*, 8 (2010), 2121–2133.

[34] Duan, Z., and Zhang, C. A maximum likelihood approach to multiple fundamental frequency estimation from the amplitude spectrum peaks. In *Neural Information Processing Systems workshop on Music, Brain and Cognition* (2007).

[35] Duan, Z., Zhang, Y., Zhang, C., and Shi, Z. Unsupervised single-channel music source separation by average harmonic structure modeling. *IEEE Trans. Audio Speech Language Processing 16*, 4 (2008), 766–778.

[36] Durrieu, J.-L., Richard, G., and David, B. Singer melody extraction in polyphonic signals using source separation methods. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2008), pp. 169–172.

[37] Ellis, D. P. W. *Prediction-Driven Computational Auditory Scene Analysis*. PhD thesis, MIT Department of Electrical Engineering and Computer Science, 1996.

[38] Ellis, D. P. W. PLP and RASTA (and MFCC, and inversion) in Matlab, 2005. online web resource.

[39] Ellis, D. P. W. Beat tracking by dynamic programming. *Journal of New Music Research 36*, 1 (2007), 51–60.

[40] Ellis, D. P. W., and Poliner, G. E. Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2007), pp. 1429–1432.

[41] Emiya, V., Badeau, R., and David, B. Multipitch estimation of quasi-harmonic sounds in colored noise. In *Proc. International Conference on Digital Audio Effects (DAFx)* (2007).

[42] Every, M. R., and Szymanski, J. E. Separation of synchronous pitched notes by spectral filtering of harmonics. *IEEE Trans. Audio Speech Language Processing 14*, 5 (2006), 1845–1856.

[43] Ewert, S., and Müller, M. Using score-informed constraints for NMF-based source separation. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2012), pp. 129–132.

[44] Ewert, S., Müller, M., and Grosche, P. High resolution audio synchronization using chroma onset features. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2009), pp. 1869–1872.

[45] Fanelli, G., Gall, J., and Van Gool, L. Hough transform-based mouth localization for audio-visual speech recognition. In *British Machine Vision Conference* (2009).

[46] Fuhrmann, F. *Automatic musical instrument recognition from polyphonic music audio signals*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2012.

[47] Galas, T., and Rodet, X. An improved cepstral method for deconvolution of source-filter systems with discrete spectra: Application to musical sounds. In *Proc. of International Computer Music Conference (ICMC)* (1990), pp. 82–84.

[48] Ganseman, J., Mysore, G., Scheunders, P., and Abel, J. Source separation by score synthesis. In *Proc. International Computer Music Conference (ICMC)* (June 2010).

[49] Ganseman, J., Scheunders, P., Mysore, G., and Abel, J. Evaluation of a score-informed source separation system. In *Proc. International Society for Music Information Retrieval Conference (ISMIR)* (2010), pp. 219–224.

[50] Garofolo, J., Lamel, L., Fisher, W., Fiscus, J., Pallett, D., and Dahlgren, N. The DARPA TIMIT acoustic-phonetic continuous speech corpus cdrom. NTIS, order number PB01-100354, 1993. now available from LDC.

[51] Ghezzo, M. *Solfege, Ear Training, Rhythm, Dictation, and Music Theory: A Comprehensive Course*, 3rd ed. University Alabama Press, 2005.

[52] Goldstein, J. An optimum processor theory for the central formation of the pitch of complex tones. *Journal of the Acoustical Society of America 54* (1973), 1496–1516.

[53] Goto, M. A real-time music-scene-description system: predominant-f0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication 43*, 4 (2004), 311–329.

[54] Goto, M., Hashiguchi, H., Nishimura, T., and Oka, R. Rwc music database: popular, classical, and jazz music databases. In *Proc. International Conference on Music Information Retrieval (ISMIR)* (2002), pp. 287–288.

[55] Grubb, L., and Dannenberg, R. B. Automated accompaniment of musical ensembles. In *Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)* (Menlo Park, CA, USA, 1994), AAAI '94, American Association for Artificial Intelligence, pp. 94–99.

[56] Grubb, L., and Dannenberg, R. B. A stochastic method of tracking a vocal performer. In *Proc. International Computer Music Conference (ICMC)* (1997), pp. 301–308.

[57] Hainsworth, S. *Techniques for the Automated Analysis of Musical Audio*. PhD thesis, University of Cambridge, 2004.

[58] Han, J., and Chen, C.-W. Improving melody extraction using probabilistic latent component analysis. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2011), pp. 33–36.

[59] Hartmann, W. Pitch, periodicity, and auditory organization. *Journal of the Acoustical Society of America 100*, 6 (1996), 3491–3502.

[60] Hu, G., and Wang, D. A tandem algorithm for pitch estimation and voiced speech segregation. *IEEE Trans. Audio Speech Language Processing 18*, 8 (2010), 2067–2079.

[61] Hu, K., and Wang, D. An unsupervised approach to cochannel speech separation. *IEEE Trans. Audio Speech Language Processing 21*, 1 (2013), 122–131.

[62] Hu, N., Dannenberg, R. B., and Tzanetakis, G. Polyphonic audio matching and alignment for music retrieval. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (New Paltz, New York, USA, 2003), pp. 185–188.

[63] Jiang, D.-n., Zhang, W., Shen, L.-q., and Cai, L.-h. Prosody analysis and modeling for emotional speech synthesis. In *Proc. IEEE International Conference on Audio, Speech and Signal Processing (ICASSP)* (2005), pp. 281–284.

[64] Jin, Z., and Wang, D. HMM-based multipitch tracking for noisy and reverberant speech. *IEEE Trans. Audio Speech Language Processing 19*, 5 (2011), 1091–1102.

[65] Joder, C., Essid, S., and Richard, G. A conditional random field framework for robust and scalable audio-to-score matching. *IEEE Trans. Audio Speech Language Processing 19*, 8 (2011), 2385–2397.

[66] Kameoka, H., Nishimoto, T., and Sagayama, S. A multipitch analyzer based on harmonic temporal structured clustering. *IEEE Trans. on Audio Speech and Language Processing 15*, 3 (2007), 982–994.

[67] Kashino, K., and Murase, H. A sound source identification system for ensemble music based on template adaptation and music stream extraction. *Speech Communication* (1999), 337–349.

[68] Kim, M., and Choi, S. Monaural music source separation: nonnegativity, sparseness, and shift-invariance. In *Proc. International Conference on Independent Component Analysis and Blind Source Separation (ICA)* (2006), pp. 617–624.

[69] Klapuri, A. Multiple fundamental frequency estimation based on harmonicity and spectral smoothness. *IEEE Trans. Speech Audio Processing 11*, 6 (2003), 804–815.

[70] Klapuri, A. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *Proc. ISMIR* (2006), pp. 216–221.

[71] Klapuri, A. Analysis of musical instrument sounds by source-filter-decay model. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (2007), pp. 53–56.

[72] Klapuri, A., and Davy, M., Eds. *Signal Processing Methods for Music Transcription*. Springer, 2006.

[73] Lagrange, M., and Tzanetakis, G. Sound source tracking and formation using normalized cuts. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2007), pp. 61–64.

[74] Le Roux, J., Kameoka, H., Ono, N., de Cheveigne, A., and Sagayama, S. Single and multiple f0 contour estimation through parametric spectrogram modeling of speech in noisy environments. *IEEE Trans. Audio Speech Language Processing 15*, 4 (2007), 1135–1145.

[75] Lee, H., Largman, Y., Pham, P., and Y., N. A. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Proc. Advances in Neural Information Processing Systems (NIPS)* (2009).

[76] Lee, K., and Slaney, M. Acoustic chord transcription and key extraction from audio using key-dependent HMMs trained on synthesized audio. *IEEE Trans. Audio Speech Language Process. 16*, 2 (2008), 291–301.

[77] Leistikow, R. J., Thornburg, H. D., Smith, Julius O., I., and Berger, J. Bayesian identification of closely-spaced chords from single-frame stft peaks. In *Proc. International Conference on Digital Audio Effects (DAFx'04)* (Naples, Italy, 2004), pp. 228–233.

[78] Levy, M., and Sandler, M. Structural segmentation of musical audio by constrained clustering. *IEEE Trans. Audio Speech Language Process. 16*, 2 (2008), 318–326.

[79] Li, Y., and Wang, D. Separation of singing voice from music accompaniment for monaural recordings. *IEEE Trans. Audio Speech Language Processing 15*, 4 (May 2007), 1475–1487.

[80] Li, Y., Woodruff, J., and Wang, D. Monaural musical sound separation based on pitch and common amplitude modulation. *IEEE Trans. Audio Speech Language Processing 17*, 7 (2009), 1361–1371.

[81] Lyon, R. F. A computational model of binaural locations and separation. In *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (1983), pp. 1148–1151.

[82] Maher, R. *An approach for the separation of voices in composite musical signals.* PhD thesis, University of Illinois, Urbana-Champaign, 1989.

[83] Maher, R. C., and Beauchamp, J. W. Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *Journal of the Acoustical Society of America 95*, 4 (1994), 2254–2263.

[84] McAulay, R. J., and Quatieri, T. F. Speech analysis/synthesis based on a sinusoidal representation. *IEEE Trans. Acoustics, Speech and Signal Processing 34*, 4 (1986), 744–754.

[85] Meddis, R., and O'Mard, L. A unitary model of pitch perception. *Journal of the Acoustical Society of America 102*, 3 (1997), 1811–1820.

[86] Montecchio, N., and Cont, A. A unified approach to real time audio-to-score and audio-to-audio alignment using sequential montecarlo inference techniques. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2011), pp. 193–196.

[87] Montecchio, N., and Orio, N. A discrete bank approach to audio to score matching for polyphonic music. In *Proc. International Society for Music Information Retrieval Conference (ISMIR)* (Kobe, Japan, 2009), pp. 495–500.

[88] Müller, M., Goto, M., and Schedl, M., Eds. *Multimodal Music Processing*. Dagstuhl Follow-Ups. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2012.

[89] Orio, N., and Dechelle, F. Score following using spectral analysis and hidden Markov models. In *Proc. International Computer Music Conference (ICMC)* (2001).

[90] Orio, N., Lemouton, S., and Schwarz, D. Score following: state of the art and new developments. In *Proc. International Conference on New Interfaces for Musical Expression (NIME)* (Montreal, Quebec, Canada, 2003), National University of Singapore, pp. 36–41.

[91] Orio, N., and Schwarz, D. Alignment of monophonic and polyphonic music to a score. In *Proc. International Computer Music Conference (ICMC)* (2001), pp. 155–158.

[92] Parzen, E. On the estimation of a probability density function and the mode. *Annals of Math. Stats. 33* (1962), 1065–1076.

[93] Pertusa, A., and Inesta, J. M. Multiple fundamental frequency estimation using Gaussian smoothness. In *Proc. IEEE International Conference on Acoustics, Speech Signal Processing (ICASSP)* (2008), pp. 105–108.

[94] Pirker, G., Wohlmayr, M., Petrik, S., and Pernkopf, F. A pitch tracking corpus with evaluation on multipitch tracking scenario. In *Proc. Interspeech* (2011), pp. 1509–1512.

[95] Poliner, G. E., and Ellis, D. P. W. A discriminative model for polyphonic piano transcription. In *EURASIP Journal on Advances in Signal Processing* (2007).

[96] Puckette, M. Score following using the sung voice. In *Proc. International Computer Music Conference (ICMC)* (1995), pp. 199–200.

[97] Radfar, M. H., and M., D. R. Single-channel speech separation using soft mask filtering. *IEEE Trans. Audio Speech Language Process. 15*, 8 (2007), 2299–2310.

[98] Rafii, Z., and Pardo, B. Music/voice separation using the similarity matrix. In *Proc. International Society for Music Information Retrieval (ISMIR)* (2012).

[99] Rafii, Z., and Pardo, B. Repeating pattern extraction technique (REPET): a simple method for music/voice separation. *IEEE Trans. Audio Speech Language Process. 21*, 1 (2013), 71–82.

[100] Raphael, C. Automatic segmentation of acoustic musical signals using hidden Markov models. *IEEE Trans. Pattern Analysis and Machine Intelligence 21*, 4 (1999), 360–370.

[101] Raphael, C. A Bayesian network for real-time musical accompaniment. In *Proc. Advances in Neural Information Processing Systems (NIPS)* (2001).

[102] Raphael, C. Aligning music audio with symbolic scores using a hybrid graphical model. *Machine Learning 65*, 2-3 (2006), 389–409.

[103] Raphael, C. A classifier-based approach to score-guided source separation of musical audio. *Computer Music Journal 32* (March 2008), 51–59.

[104] Reis, G., Fonseca, N., and Ferndandez, F. Genetic algorithm approach to polyphonic music transcription. In *Proc. IEEE International Symposium on Intelligent Signal Processing* (2007).

[105] Röbel, A., Zivanovic, M., and Rodet, X. Signal decomposition by means of classification of spectral peaks. In *Proc. International Computer Music Conference (ICMC)* (2004), pp. 446–449.

[106] Rodet, X. Musical sound signal analysis/synthesis: Sinusoidal+residual and elementary waveform models. In *IEEE Time-Frequency and Time-Scale Workshop (TFTS97)* (1997).

[107] Roweis, S. One microphone source separation. In *Proc. Advances in Neural Information Processing Systems (NIPS)* (2001), pp. 15–19.

[108] Ryynanen, M., and Klapuri, A. Polyphonic music transcription using note event modeling. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (2005), pp. 319–322.

[109] Saito, S., Kameoka, H., Takahashi, K., Nishimoto, T., and Sagayama, S. Specmurt analysis of polyphonic music signals. *IEEE Trans. Speech Audio Processing 16*, 3 (2008), 639–650.

[110] Schnupp, J., Nelken, I., and King, A. *Auditory Neuroscience*. MIT Press, 2011.

[111] Serra, J., Serra, X., and Andrzejak, R. G. Cross recurrence quantification for cover song identification. *New Journal of Physics 11* (2009).

[112] Sha, F., and Saul, L. Real-time pitch determination of one or more voices by nonnegative matrix factorization. In *Proc. Advances in Neural Information Processing Systems (NIPS)* (2005), pp. 1233–1240.

[113] Shao, Y., Srinivasan, S., Jin, Z., and Wang, D. A computational auditory scene analysis system for speech segregation and robust speech recognition. *Computer Speech and Language 24* (2010), 77–93.

[114] Simon, I., Morris, D., and Basu, S. Mysong: automatic accompaniment generation for vocal melodies. In *Proc. ACM SIGCHI* (2008).

[115] Smith, J., and Serra, X. Parshl: an analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation. In *Proc. Internetional Computer Music Conference (ICMC)* (1987).

[116] Springer, J., Duan, Z., and Pardo, B. Approaches to multiple concurrent species bird song recognition. In *The 2nd International Workshop on Machine Listening in Multisource Environments, ICASSP* (2013).

[117] Talkin, D. A robust algorithm for pitch tracking (RAPT). In *Speech Coding and Synthesis*, W. Kleijn, , and K. Paliwal, Eds. Elsevier Science B.V., 1995, pp. 495–518.

[118] Thomas, V., Fremerey, C., Damm, D., and Clausen, M. Slave: a score-lyrics-audio-video-explorer. In *Proc. International Society for Music Information Retrieval Conference (ISMIR)* (2009).

[119] Thornburg, H., and Leistikow, R. A new probabilistic spectral pitch estimatior: extract and mcmc-approximate strategies. *Lecture Notes in Computer Science 3310/2005* (2005), 41–60.

[120] Tolonen, T., and Karjalainen, M. A computationally efficient multipitch analysis model. *IEEE Trans. on Speech and Audio Processing 8*, 6 (2000), 708–716.

[121] Tzanetakis, G., Essl, G., and Cook, P. Automatic musical genre classification of audio signals. In *Proc. International Symposium on Music Information Retrieval (ISMIR)* (2001), pp. 293–302.

[122] Vercoe, B. The synthetic performer in the context of live performance. In *Proc. International Computer Music Conference (ICMC)* (1984), pp. 199–200.

[123] Vincent, E. Musical source separation using time-frequency source priors. *IEEE Trans. Audio Speech Language Processing 14*, 1 (2006), 91–98.

[124] Vincent, E., Gribonval, R., and Févotte, C. Performance measurement in blind audio source separation. *IEEE Trans. Audio Speech Language Processing 14*, 4 (July 2006), 14621469.

[125] Vincent, E., Gribonval, R., and M.D., P. BSS Oracle Toolbox Version 2.1.

[126] Vincent, E., and Plumbley, M. D. Efficient Bayesian inference for harmonic models via adaptive posterior factorization. *Neurocomputing 72* (December 2008), 79–87.

[127] Virtanen, T. Algorithm for the separation of harmonic sounds with time-frequency smoothness constraint. In *Proc. International Conference on Digital Audio Effects (DAFx)* (2003), pp. 35–40.

[128] Virtanen, T. *Sound source separation in monaural music signals*. PhD thesis, Tampere University of Technology, 2006.

[129] Virtanen, T. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE Trans. Audio Speech Language Processing 15*, 3 (March 2007), 1066–1074.

[130] Viste, H., and Evangelista, G. A method for separation of overlapping partials based on similarity of temporal envelopes in multi-channel mixtures. *IEEE Trans. Audio Speech and Language Process. 14*, 3 (2006), 1051–1061.

[131] Wagstaff, K., and Cardie, C. Clustering with instance-level constraints. In *Proc. International Conference on Machine Learning (ICML)* (2000), pp. 1103–1110.

[132] Wagstaff, K., Cardie, C., Rogers, S., and Schroedl, S. Constrained k-means clustering with background knowledge. In *Proc. International Conference on Machine Learning (ICML)* (2001), pp. 577–584.

[133] Wang, B., and Plumbley, M. Musical audio stream separation by non-negative matrix factorization. In *Proc. DMRN Summer Conference* (2005), pp. 23–24.

[134] Wang, D., and Brown, G., Eds. *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*. IEEE Press/Wiley-Interscience, 2006.

[135] Wang, Y., Lin, J., Chen, N., and Yuan, W. Improved monaural speech segregation based on computational auditory scene analysis. *EURASIP Journal on Audio, Speech, and Music Processing 2013*, 2 (2013).

[136] Wohlmayr, M., Stark, M., and Pernkopf, F. A probabilistic interaction model for multipitch tracking with factorial hidden Markov models. *IEEE Trans. Audio Speech Language Processing 19*, 4 (2011), 799–810.

[137] Woodruff, J., Li, Y., and Wang, D. Resolving overlapping harmonics for monaural musical sound separation using pitch and common amplitude modulation. In *Proc. International Conference on Music Information Retrieval (ISMIR)* (2008), pp. 538–543.

[138] Woodruff, J., Pardo, B., and Dannenberg, R. Remixing stereo music with score-informed source separation. In *Proc. International Conference on Music Information Retrieval (ISMIR)* (2006), pp. 314–349.

[139] Woodruff, J., and Wang, D. Binaural localization of multiple sources in reverberant and noisy environments. *IEEE Trans. Audio Speech Language Process. 20*, 5 (2012), 1503–1512.

[140] Wu, M., Wang, D., and Brown, G. J. A multipitch tracking algorithm for noisy speech. *IEEE Trans. Speech Audio Process. 11*, 3 (2003), 229–241.

[141] Yeh, C., Röbel, A., and Rodet, X. Multiple fundamental frequency estimation of polyphonic music signals. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2005), pp. 225–228.

[142] Yeh, C., Röbel, A., and Rodet, X. Multiple fundamental frequency estimation and polyphony inference of polyphonic music signals. *IEEE Trans. Audio Speech and Language Process. 18*, 6 (2010), 1116–1126.

[143] Yilmaz, O., and Rickard, S. Blind separation of speech mixtures via time-frequency masking. *IEEE Trans. Signal Process. 52*, 7 (2004), 1830–1847.

APPENDIX A

Pitch and Fundamental Frequency

Pitch is a perceptual attribute of sound. It is defined through psychoacoustic experiments as the frequency of a sinusoidal wave which is perceived as having the same frequency height as the target sound [**59**]. Pitch is subjective. Different human listeners, or even the same listener in different conditions may have inconsistent perceptions on pitch. For periodic sounds, these perceptions are usually consistent, while for non-periodic sounds, they are often inconsistent. Therefore, it is only meaningful to talk about pitch for periodic sounds.

Fundamental frequency, abbreviated as *F0*, is an objective physical term. It is defined as the reciprocal of the period of a sound, hence only defined for periodic or nearly periodic sounds. For most sounds that have a consistent pitch perception, F0 matches pitch. Therefore, in computer science and engineering fields such as music information retrieval, people often do not distinguish pitch and F0.

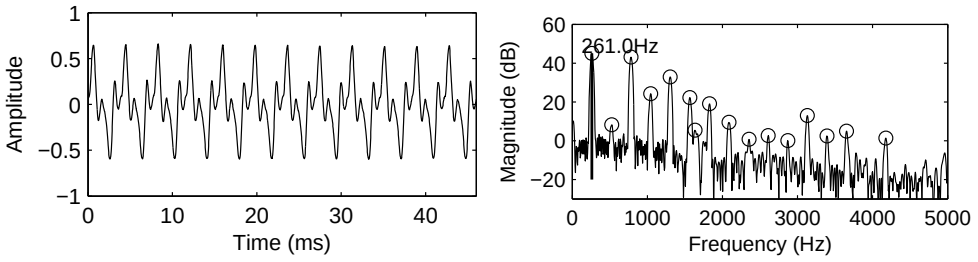
APPENDIX B

Harmonic, Quasi-harmonic and Inharmonic Sounds

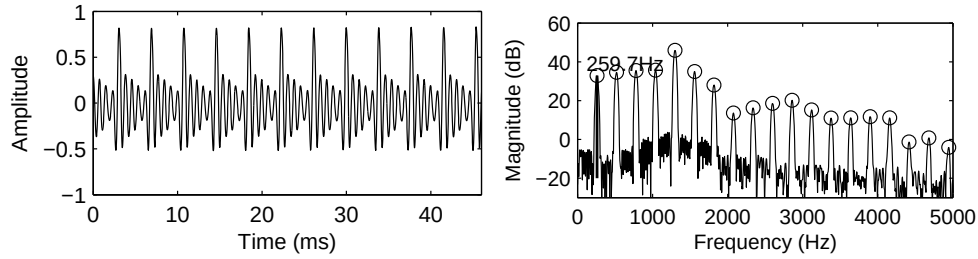
By Fourier analysis, a periodic sound can be viewed as a summation of sinusoids with different frequencies and amplitudes. In the frequency domain, these sinusoids form different spectral patterns. It is the spectral pattern that classifies periodic sounds into harmonic or quasi-harmonic.

Harmonic sounds are the sounds whose dominant frequency components (significant spectral peaks) are approximately equally spaced, and the gap between adjacent peaks equals the F0 (see Figure B.1-(a) and (b)). These spectral peaks appear at integer multiples (harmonic positions) of the F0, and they form harmonics. Harmonics have different amplitudes, and their relative amplitudes form a *harmonic structure*. It is the harmonic structure that strongly determines the timbre of a harmonic sound. Some harmonics may have too small amplitudes to be recognized, e.g. even harmonics of the clarinet. In this case, they are called "missing harmonics". Occasionally, the first harmonic, i.e. the fundamental can also be missing, without interfering the pitch perception [**110**]. This is called "the missing fundamental" situation. Many musical instruments (e.g. violin, piano, vocal, etc.) produce harmonic sounds and they are called harmonic instruments.

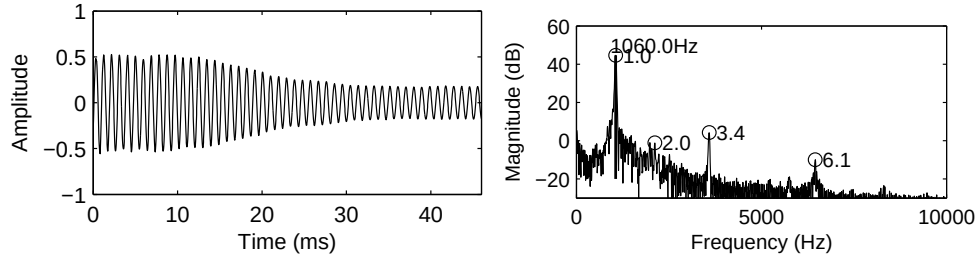
Quasi-harmonic sounds are also nearly periodic, however, their spectra do not have a harmonic structure. Figure B.1-(c) shows a marimba sound. The waveform has a clear period and the sound has a consistent pitch perception. However, many harmonics are missing and many spectral peaks do not appear at harmonic positions. Instruments that



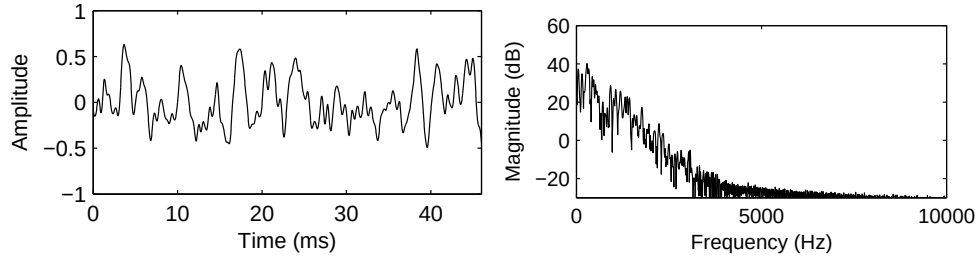
(a) Harmonic sound by clarinet C4



(b) Harmonic sound by oboe C4



(c) Quasi-harmonic sound by marimba



(d) Inharmonic sound by gong

Figure B.1. Comparisons of harmonic, quasi-harmonic and inharmonic sounds. Significant peaks in each magnitude spectrum are marked by circles. They appear at harmonic positions in harmonic sounds, but not always in quasi-harmonic sounds. Clarinet and oboe have different harmonic structures for the same note.

produce quasi-harmonic sounds include pitched percussive instruments, e.g. marimba, tabla, timpani, tuned triangle, vibraphone, xylophone, etc.

Inharmonic sounds refer to non-periodic sounds. Figure B.1-(d) shows a gong sound. It can be seen that the waveform is not periodic, nor does the spectrum have a harmonic structure. Non-pitched percussive instruments like drums, gong and tambourine are in this category.

Table B.1 classifies Western musical instruments into the three categories. It can be seen that harmonic instruments are the most popular ones in music.

Table B.1. Classification of Western musical instruments into harmonic, quasi-harmonic and inharmonic categories.

Sounds	Instrument family	Instruments
Harmonic	Woodwind	Flute, oboe, clarinet, bassoon, saxophone
	Brass	Trumpet, horn, trombone, tuba
	Arco string	Violin, viola, cello, double bass
	Pluck string	Piano, guitar, harp, celesta
	Vocal	Voiced phonemes
Quasi-harmonic	Pitched percussive	Timpani, marimba, vibraphone, xylophone
Inharmonic	Non-pitched percussive	Drums, cymbal, gong, tambourine