Characterization of the singing voice from polyphonic recordings

Christine Smit

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2011

Abstract

Characterization of the singing voice from polyphonic recordings

Christine Smit

In order to study the singing voice, researchers have traditionally relied upon lab-based experiments and/or simplified models. Neither of these methods can reasonably be expected to always capture the essence of true performances in environmentally valid settings. Unfortunately, true performances are generally much more difficult to work with because they lack precisely the controls that lab setups and artificial models afford. In particular, true performances are generally polyphonic, making it much more difficult to analyze individual voices than if the voices can be studied in isolation.

This thesis approaches the problem of polyphony head on, using a time-aligned electronic score to guide estimation of the vocal line characteristics. First, the exact fundamental frequency track for the voice is estimated using the score notes as guides. Next, the harmonic strengths are estimated using the fundamental frequency information. Third, estimates in notes are automatically validated or discarded based on characteristics of the frequency tracks. Lastly, good harmonic estimates are smoothed across time in order to improve the harmonic strength estimates.

These final harmonic estimates, along with the fundamental frequency track estimates, parameterize the essential characteristics of what we hear singers' voices. To explore the potential power of this parameterization, the algorithms are applied to a real data consisting of five sopranos singing six arias. Vowel modification and evidence for the singer's formant are explored.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank the many people who helped me through the process of developing the ideas in this thesis and through the actual writing process. First and foremost, I would like to thank my advisor Dan Ellis for all the guidance and assistance he has provided while I have been a student. He showed me how to take a germ of an idea and make a thesis out of it. Many thanks to my thesis committee for their comments and suggestions. I would also like to thank Martha Randall, who fostered my fascination with the mechanics of the voice while I was an undergraduate.

I feel I must also give a shout out to the members of LabROSA. Thank you, all of you, for the helpful discussions and the general sense of fun around the lab. I would especially like to thank Ron Weiss for lending me his thesis template, which made the process of writing my thesis infinitely easier than it would have been otherwise.

Finally, I must thank my husband Jeff who helped to keep my spirits and energy up when they threatened to sag.

Chapter 1

Introduction

Of all music instruments, the voice is arguably the most complex and most expressive. Although it may not have the range or power of some other instruments, the voice does have an incredible ability to produce different kinds of tones.

This thesis is motivated by the idea that a parametric description of the voice in an artistic performances would allow researchers to answer new questions about performance practice that they have been unable to answer before. Perhaps just as importantly, such a parametric model would allow researchers to answer old questions with more quantitative authority than was possible in the past.

The complexity of the voice is, of course, tied intimately with our ability to speak and produce all the sounds that speech requires. Researchers have studied the mechanics of the vocal system that allow us to articulate speech (e.g. [29]) from the characteristics of the glottal source to how movements of the tongue, jaw, soft palate, etc. shape the words we speak. Singing requires us to adapt this sophisticated speech system to the temporal, pitch, and other constraints of music. One of the most obvious changes is the general practice of elongating vowels on sustained notes. Indeed, these sustained vowels often make up a majority of what we recognize as

'singing.' This thesis confines itself to these vowels, which are voiced and therefore have the characteristic harmonic structure of musical notes (section 2.1) and can be described in terms of time-varying magnitudes of harmonically-related pure tones.

1.1 Contributions

There were two ways to think about acquiring data about singers. The first way essentially involves recruiting singers to come and perform in a room with a microphone. This data is expensive, difficult to collect, and ultimately means that only singers who are physically and temporally close to researchers will ever be studied. The second way involves digging into the enormous and varied corpus of commercially recorded music. The size of this corpus is appealing, but the reasons for using artistic recordings rather than laboratory recording go beyond simply the quantity of data. Laboratory recordings tend to be very unnatural. Researchers often ask their subjects to sing in an unusual way, such as without vibrato, or use unusual equipment, such a electroglottograph. In contrast, artistic recordings are made only with artistic constraints. The performers are singing to express their musical ideas, not to stay within the confines of the experimental constraints. The algorithms described in this thesis provide a toolbox for quantitatively studying truly musical performances.

There is a difficulty peculiar to music that does not generally affect study of speech, namely the fact that music, at least Western music, is generally polyphonic. Speech researchers can access huge databases of 'clean' speech where the speaker has been recorded by a microphone in a quiet room. Although there is also, of course, a great deal of speech recorded in noisy environments, 'clean' recordings are the generally desired form of speech by both the public and researchers. Anyone who releases audio recordings with a great deal of background noise will be accused of

poor production values. So, in some sense, a single voice, recorded with very little noise, is a natural and desired kind of speech.

In contrast, Western music is overwhelmingly polyphonic. Even 'solo' music is polyphonic. The lead or 'solo' instrument is generally backed by one or more other instruments. So polyphony is the truly desired and natural form of recorded music. This form of music is what musicians know, work to, and feel comfortable performing. Of course, in practice, professional recordings are generally mixes of tracks of individually recorded instruments. So musicians do sit in sound-isolated rooms with headphones and a microphone. Still, at some very fundamental level, music is truly meant to exist as polyphony.

What this thesis does at its most fundamental level is try to give researchers tools for pulling out the important characteristics of the voice from polyphonic recordings so that the vast collections of professional recordings can be analyzed for the kind of information that was previously available most easily from controlled laboratory recordings:

- Exact pitch estimation. This thesis discusses algorithms for finding exact fundamental frequency estimates using electronic score information as a guide.
- Harmonic strength estimation. This thesis discusses using the fundamental frequency estimates to estimate the magnitudes of the harmonics.
- Vowel parameterization. This thesis discusses averaging the raw harmonic strength estimates across one or more notes sung on the same vowel and pitch in order to characterize how a particular vowel is sung.

1.2 Thesis organization

The remainder of the thesis is organized as follows. Chapter 2 gives a background discussion of the topics covered later in the thesis. The qualities of the singing voice are discussed as well as how they have been estimated in the past. There is also a brief discussion of MIDI files as this is the electronic score format used in the rest of the thesis. Finally, a data set used in later chapters is described.

Chapter 3 is a very brief chapter that covers converting lyrics into phones. The lyrics, taken from midi files, are aligned to notes at the syllable level. Pronunciation dictionaries generally give pronunciations of entire words. This chapter covers breaking up those pronunciations into syllables.

Chapter 4 discusses an initial approach taken to finding snippets of monophony in largely polyphonic music. Essentially, the easiest way to isolate a voice is to find it in isolation already. This chapter explores finding these isolated sections.

Chapter 5 starts down a new path of characterizing voices in polyphony by examining two different approaches to very exact frequency estimation. The idea is to start by aligning an electronic score with the audio. This aligned electronic score provides very good information about approximately where each voice and note is. The task then becomes homing in on exactly where the note is near the nominal note frequency provided by the score.

Chapter 6 discusses using the frequency estimates from the previous chapter to make initial estimates of the harmonic strengths of notes being sung on a frame-by-frame basis. These initial estimates essentially look for energy where the fundamental frequency indicates the energy should be.

Chapter 7 investigates the problem of removing bad data estimates. The frequency estimates can fail, and when they do, the initial amplitude estimates also fail. This

chapter looks at certain characteristics of the frequency track in order to decide what part of each note, if any, contains good information.

Chapter 8 discusses improving the harmonic amplitude estimates by smoothing across time. The initial harmonic amplitude estimates are extremely noisy because they rely on a few Discrete Fourier Transform (DFT) bins. Smoothing across time allows for better estimates of the characteristics of the vowel across time.

Chapter 9 uses the algorithms explored in chapters 3 and 5 to 8 to explore a small data set consisting of six arias from five operettas. Finally, chapter 10 offers concluding remarks about the thesis as a whole.

Chapter 2

Background

This chapter lays the background for the work in the remaining chapters. It covers aspects of the voice that are discussed in the thesis (section 2.1), modeling the voice (section 2.2), Musical Instrument Digital Interface (MIDI) issues (section 2.3), and a data set that is used extensively throughout the thesis (section 2.4).

2.1 The singing voice

The voice is often thought of in terms of a source-filter model [22]. In vowels, the focus of later chapters in this thesis, the source is the vibration of the vocal folds, and the filter is created by everything above the vocal folds, namely the pharynx, mouth, and nasal cavities. The source created by the vocal folds is approximately periodic over short time intervals. In the frequency domain, this glottal source is a harmonic series of partials where each partial is at an integer multiple of the fundamental frequency [47]. Since the source is harmonic, what we hear is also harmonic, with partials whose intensities have been changed by the vocal tract filter.

2.1.1 Vowel formants

The vocal tract filter characteristics for vowels are generally defined in terms of formants, or peaks in the vocal tract filter, which strengthen near-by glottal harmonics. Peterson and Barney [40] showed that the first and second harmonics are sufficient for characterizing vowels. They plotted the first versus the second formant frequencies of the utterances of 76 speakers speaking a variety of English vowels and showed that these vowels formed reasonably separate clusters. Technically, Peterson and Barney were plotting what they called "energy concentrations" rather than formants because formants cannot be directly observed from just the vocal utterances. Since the glottal source is harmonic, spoken vowels can only sample the vocal tract filter at multiples of the fundamental frequency.

When singing, these vowel formants can create a problem. For example, Peterson and Barney characterize the vowel /u/ as having a first formant frequency somewhere between about 200 and 400 Hz. Singers, particularly sopranos, regularly sing above 400 Hz, which is about G4. Because the intensity of the voice is a function of both the glottal vibrations and the resonances of the vocal tract, the harmonic frequencies need to be near resonances to maximize intensity. So, to accommodate these higher pitches, sopranos change the formant frequencies to get more power out of their voices. This process is called vowel modification or formant tuning.

Sundberg made the argument in 1987 that the lower position of the jaw on higher notes is a strong arguments for vowel modification since the lower jaw position is associated with a higher first formant [47]. Unfortunately, measuring these formant frequencies directly from the recorded voice is difficult or impossible due to the wide spacing of harmonics at higher pitches. So, Sundberg and Skoog measured jaw openings as a proxy for formant tuning of the first formant in a variety of voices and concluded that formant tuning seemed likely in notes higher than the first formant

[50].

Of course, formant tuning does not necessarily have to be limited to high notes. Singers may want to take advantage of the energy output increase due to tuning formants to higher partials even when formant tuning is not required by the low fundamental. Miller and Schutte found some direct evidence of formant tuning in a baritone singer in 1990 by recording the electroglottographic, subglottal and supraglottal pressures simultaneously with audio [36]. A couple years later, however, Carlsson and Sundberg used an artificial singing synthesizer to simulate three simple formant-tuning strategies which were then evaluated by professional singing teachers [12]. As with the baritone, the notes were low enough that the first formant was above F_0 (C4 down to C3, about 262 Hz down to 131 Hz). Interestingly, the singing teachers overwhelmingly preferred the fixed formant scales rather than the tuned formant scales, suggesting that the energy increase associated with tuning formants to upper partials was not sufficient to overcome the intelligibility problems associated with moving the formants.

Returning once again to sopranos and high notes, Joliveau et al. [25] were able to make a reasonably accurate estimate of the entire vocal tract filter of nine sopranos singing four vowels using broad band acoustic excitation of the vocal tract at the mouth while the singers sang. They showed strong evidence that the first formant remained essentially constant until F_0 reached the first formant, at which point the first formant began to track F_0. As the authors point it out, it may be that intelligibility ceases to even really be possible at sufficiently high pitches due to the widely spaced partials, and so increasing energy becomes the primary goal instead.

2.1.2 Singer's formant

In 1934, Bartholomew reported an energy peak in the range of 2800-3200 Hz in good quality male singing voices [7]. Sundberg coined the term "singer's formant" in 1974 to describe this phenomenon and developed a physical model to explain the resonance pattern [48], namely that it is a clustering of the third, fourth, and fifth formants. This definition physically limits the singer's formant to lower voices. In contrast, Bloothooft and Plomp performed experiments in 1986 on a wide variety of voices from bass to soprano [10]. They defined the singer's formant in terms of energy in one third of an octave centered around 2.5 kHz for men and 3.16 kHz for women. In their experiments on 14 singers, the singer's formant met an energy threshold in almost all cases except for fundamental frequencies above 392 Hz. Barnes and Davis et al. [6] stuck with Sundberg's formant clustering definition of the singer's formant, but showed that sopranos with more prestigious careers have more energy in 3 kHz range, suggesting that energy in this range is also important to sopranos even if the mechanism by which they achieve this energy is different.

2.1.3 Vibrato

Vibrato, a regular undulation in the pitch of the voice, can be characterized in terms of rate, depth, regularity, and waveform [49]. The rate of the vibrato is generally somewhere between 5.5 and 7.5 Hz and the depth is about $\pm$ 1 or 2 semitones [49]. While vibrato is nominally a sinusoidal shape, some variations in the shape [49] and depth [34] do occur.

With an unmodulated tone, the output frequency spectrum only has energy at the harmonics of the single fundamental frequency. Vibrato, on the other hand, sweeps out a small section of frequencies and thus illuminates small sections of the vocal tract filter around the center frequency of each of the partials, causing amplitude

modulation synchronized with the vibrato. Unfortunately, amplitude modulation may not be entirely due to the vocal tract filter. First of all, the vocal tract shape can change in synchrony with the vibrato [49], in which case the vocal tract filter is not constant across the sung note. Second of all, the glottal source itself may cause some of the amplitude variation due to changes in subglottal pressure or glottal adjustment [49].

A second question is whether this amplitude modulation is perceivable. If the amplitude modulation tells listeners something useful about the location of formants, then we would expect vibrato to improve intelligibility of vowels. Sundberg, using synthesized vowels with synthesized vibrato, found the intelligibility did not change between vowels with and without vibrato, suggesting that these subtle amplitude cues may not be perceptually relavant [45]. Still, amplitude modulation seems important enough for researchers to include it in their models (e.g. [2, 34]).

2.2 Modeling the voice

In order to parameterize the voice, a model is first needed. A few common approaches to voice modeling are physical models, source-filter models, and sinusoidal models.

2.2.1 Physical modeling

Perhaps the most obvious way to model the voice is to create either a mathematical or physical model for the physical system that produces the voice, namely the vocal folds and vocal tract. In mathematical models, the vocal folds are usually modeled as one [23] or more masses [51]. The vocal tract is then modeled as a series of connected cylindrical tubes of differing cross-sections, known as a waveguide digital filter [27, 14]. Unfortunately, the number of cylinders needed is generally very high and the models can become unstable [28]. Furthermore, estimating the model parameters can be

difficult [27]. Physical models, such as [24], get around the instability and waveguide complexity issues. They are also able to easily model non-linear effects [27].

2.2.2 Source-filter modeling

As mentioned earlier, the source-filter model assumes that the vocal fold source is linearly filtered through the vocal track. A popular technique for estimating the vocal tract filter based on this model is Linear Predictive Coding (LPC) [5]. This method posits that $s[n]$, the current audio sample, can be estimated from a weighted sum of the previous P samples and the glottal source. The transfer function of this equation, which corresponds to the model of the vocal tract, then becomes an all-pole filter. The coefficients of the filter, which are the weights in the sum, are estimated by minimizing the distance between $s[n]$ and the weighted sum plus glottal source. The simplicity of the model and the ease with which the filter coefficients can be calculated makes this model appealing[27].

2.2.3 Sinusoidal modeling

Sinusoidal modeling means estimating the frequency, amplitude, and phase of the partials. This is essentially the approach taken in this thesis. A very simple model for a stationary harmonic tone is a sum of weighted partials[1],

$$x(t) = \sum_{i=1}^{N_h} A_i \cos\left(2\pi t i f_0 + \phi_i\right) \tag{2.1}$$

where t is the current time, N_h is the number of partials, A_i is the amplitude of partial i, f_0 is the fundamental frequency, and ϕ_i is the phase of partial i. To completely reconstruct $x(t)$ in this case, the fundamental frequency, harmonic amplitudes, and

[1]Not all musical tones are perfectly harmonic [54], but a healthy voice is very close to perfectly harmonic [33].

harmonic phases would have to be estimated. However, it has long been known that the phases of the partials are relatively unimportant to perception [35], so this thesis will primarily focus on estimating the fundamental frequency and harmonic amplitudes.

The most basic problem of single frequency estimation has been covered extensively using, for example, instantaneous frequency estimations [11] or autocorrelation [16]. These methods either will not work for musical tones with harmonics [11] or they ignore the extra information available in the higher harmonics [16]. Linear predictive filtering has been used to tackle harmonic sinusoids [13], but this method does not address the issue of multiple musical notes at once. Transcription (e.g. [9]) involves finding multiple simultaneous musical notes, but is more often concerned with looking for note names, not exact pitches or amplitudes. Coding via harmonic pitch objects [53] is more interested in exact pitches and amplitudes, but aims mostly more for good reconstruction.

The phase vocoder, used for analysis and synthesis, can give frequency, amplitude, and phase estimates [19], but will not work in a polyphonic context because it will generally be impossible to isolate the partials of the voice from the instrumental partials.

Given the difficulty of dealing with mixtures of voices, it would be nice to separate the voice from the mix first before trying to estimate its characteristics. Ozerov et al. [39] attempt to pull the singer out of the mix in pop music using binary masking, which assumes that the sources are reasonably separated in time and frequency. Bins in the spectrogram of the signal are assigned exclusively to one source or another. This model may not be a good fit to the data, particularly in the case of music where notes in musically consonant relations have harmonics that overlap. There, even after separation, the problem of interfering partials from other instruments may be a

problem.

Other approaches to source separation use directional cues from multiple microphones to perform source separation. Independent component analysis (ICA) assumes that the source at each microphone is a linear mixture of the sources present. Such algorithms can have problems in reverberant environments, where echoes can look like additional sources [41]. Since most music recordings have reverberation, this can be a problem.

2.3 MIDI

Much of this thesis relies on information from electronic scores either directly (chapters 5, 7 and 9) or indirectly. Although these electronic scores can in theory be in any convenient form, in practice the scores used in this thesis are all MIDI files. This section is a catch-all description of the issues that arise from having to work with MIDI.

The complete MIDI standard, of which the MIDI file format is a small part, can be found on MIDI Manufacturers Association website [3]. In brief, MIDI files can store most, but not all, of the information that a typical score contains. Notes, numbered by half step from 0 to 127, can be turned on and off. The amplitude or 'velocity' of the notes can be set or changed. The note tuning can be changed or 'bent.' The length of the quarter note can be changed to speed up or slow down the music using tempo commands. Timed lyric commands can specify lyrics at the syllable level. MIDI files can also contain events that specify key and time signature, but these events are purely informational and do not affect the interpretations of the tempo or note on/off commands. However, because the MIDI standard was originally developed to allow communication between MIDI devices rather than creation of electronic scores, MIDI files cannot specify many of the score markings

that musicians are accustomed to seeing. So there is no way of including standard dynamic markings (e.g. forte, piano), articulation marks (e.g. staccato, tenuto), tempo indications (e.g. andante, allegro, presto), or mood indications (e.g. dolce, contabile) in MIDI files.

A type 1 MIDI file essentially contains a stream of time-stamped MIDI events on tracks. These events can either be channel-based events or meta-events. A channel in a track generally identifies a single device or instrument, so channel-based events include commands associated with a particular device such as note on and note off. Confusingly, meta-events can apply just to the track they are on (e.g. track name commands) or the entire stream (e.g. tempo and time signature commands). So why are tracks needed if there are channels? The standard only allows 4 bits to specify the channel, which limits a single track to 16 devices. Adding additional tracks allows for more devices. Tracks can also be used to group or organize channels together.

2.3.1 Tools for reading MIDI files

In Matlab, the MIDI Toolbox [20] is a well known free set of MIDI-related functions. It's two low level functions for reading and writing MIDI files unfortunately suffer from several problems. First, they rely on out-dated external executables that no longer run under recent versions of Windows. Second, they are unable to deal with tempo changes, so the timing information is very poor for tracks including such commands. Third, they cannot pull out lyrics information.

Ken Schutte has also published Matlab MIDI tools[2]. These tools run reliably and deal with tempo changes, but do not do all the lyrics processing that is needed for this thesis.

[2]http://www.kenschutte.com/midi. Accessed on 2010-11-01.

Matlab can run java natively and the javax.sound.midi package [3] provides some useful functionality. It parses MIDI files into MidiEvent objects that each contain a MidiMessage, which simply has the raw bytes of the MIDI event. Unfortunately, this package does not do much to parse these raw bytes into anything meaningful.

Because none of the existing tools did exactly what this thesis work needed, new tools were created. At the time that this thesis's MIDI work started, Ken Schutte's tools were not as advanced as they are now. So, the Java package seemed like the best place to start. These developed tools have been released[4] and can handle tempo changes, run on multiple operating system, and can easily grab lyrics out of midi files.

2.3.2 Time alignment

Although MIDI scores contain timing information, the MIDI timing will obviously not line up particularly well with an actual performance of a particular piece of music. The problem of automatic MIDI-to-audio alignment is not new and has been studied extensively, for example [21, 52, 18, 15]. The general approach is to break the audio into frames, break the MIDI up into frames (if the MIDI is synthesized) or collections of simultaneous notes (for features directly calculated from the MIDI), calculate features for the audio and MIDI, calculate distances or matching costs between every frame in the audio and every frame or note collection in the MIDI, and then use a dynamic time warp to find the best alignment.

The dynamic time warp works on a matrix of cost values. Each cost value at index (i, j) represents the cost of matching frame i in the audio to frame or note collection j in the MIDI. There is also often a transition cost associated with moving

[3] http://download.oracle.com/javase/1.4.2/docs/api/index.html version 1.4.2. Accessed on 2010-11-01.

[4] http://www.mathworks.com/matlabcentral/fileexchange/27470-midi-tools. Accessed 2010-05-04.

from one location on the matrix to another. A common transition cost scheme might assign a cost of 1 to moving diagonally $(i, j) \rightarrow (i + 1, j + 1)$, forward in the audio $(i, j) \rightarrow (i + 1, j)$, or forward in the MIDI $(i, j) \rightarrow (i, j + 1)$, and an infinite cost to moving in any other direction. The cost of the path through the matrix is then a combination of all the matching costs and transition costs and the best path is the lowest-cost path through the matrix.

Clearly the choice of both the matching costs and the transition costs can have a significant effect on the final solution. For example, if the piece of music contains repeated notes and the features are based on the stationary short time characteristics (e.g. harmony) rather than on transient characteristics (e.g. onsets), the dynamic time warp can have difficulty deciding where to put the boundary between the notes. For similar reasons, silences in the music can cause problems. The transition costs can come into play particularly if the number of frames in the audio is significantly different from the number of frames or note collections in the MIDI. In this case, if the movement costs are biased toward the diagonal $(i, j) \rightarrow (i + 1, j + 1)$, then the shortest path will want to incorporate as many diagonal elements as possible even though the correct path will clearly require more lateral or upward movement.

These issues with the dynamic time warp are neither new nor unknown, but still cause problems. So, for the work in this thesis, all alignments are calculated by hand in order to ensure accuracy.

2.4 The Gilbert and Sullivan data set

The Gilbert and Sullivan data set, which is used in several different experiments, consists of six soprano arias from five different operettas written by the musical collaborators Arthur Sullivan (music) and W.S. Gilbert (libretto). Each aria has a

full orchestra recording from the D'Oyly Carte Opera Company, the same company that gave the original performances of all the works in the late 1800s.

The soprano lines in each recording have been aligned on a note-by-note basis[5] to midi scores from the Gilbert and Sullivan Archive [1]. All note onsets are individually annotated and note offsets are assumed to be the next note's onset. For notes where this is not the case, such as notes right before a breath, the note offset is also annotated. Some notes from the arias are ignored because they are difficult or impossible to align, such as cadenzas with extremely fast notes. After removing these notes, there are approximately 1500 labeled notes in the six recordings.

These midi scores contain both the notes and lyrics of these arias. So, after note alignment, the midi scores provide syllable-level word alignment.

[5]Specifically, the author tapped along with the notes using Sonic Visualizer (http://www.sonicvisualiser.org/), which can play the music and record the timing of the taps. A musical score was used for reference. The taps were then reviewed for accuracy and corrected, again using Sonic Visualizer, which can play back the music and add clicks at the annotated times.

Chapter 3

Determining vowels from lyrics embedded in scores

The ultimate goal of this work is to characterize sung voices. One part of that characterization involves looking at how singers form vowels (chapter 8). While it would certainly be possible to hand label notes with the vowels the performers are singing, an automated solution would be much preferable.

As mentioned in section 2.3, MIDI files can have embedded lyrics. These lyrics are generally time-tagged at the syllable level. In essentially every case, syllables have only a single vowel. Diphthongs and glides are an obvious complication, but in these cases singers generally sustain on the longer vowel. So, if the syllables can be transcribed in phones, finding the vowel for each syllable is very straightforward.

Pronunciation dictionaries provide word-level phone transcriptions, not syllable-level transcriptions. So the question is how to break up the word-level string of phones into the phones associated with each syllable. One approach would be to look at the all pronunciation rules in English and then try to map these pronunciation rules onto the words and their corresponding phone sequences. This chapter takes a

much simpler, probabilistic approach instead.

The organization of the rest of this chapter is as follows. Section 3.1 covers calculating the alignment, section 3.2 discusses how the alignment works, and section 3.3 summarizes the chapter as a whole.

3.1 Calculating syllable to phone alignment

The basic idea behind this model is that the relationship between letters and phones can be estimated probabilistically from a large data set. In other words, a letters-to-phones matrix can be calculated where each entry is the probability that a particular letter will map to a particular phone.

In this case, the large data set is the British English Example Pronunciation (BEEP) dictionary[1], version 1.0. An initial, best-guess approximation of the alphabet-to-phones map is calculated by assuming a linear mapping from letters to phones in the dictionary and averaging across all words. This initial map is then improved upon iteratively.

In each iteration, the alphabet-to-phones map is updated using the previous iteration's map. For each word-phones pair, the most likely mapping from letters to phones is calculated using a dynamic time warp on a word-level letter-to-phone mapping from the previous iteration. Note that since dynamic time warping finds the lowest cost path while this algorithm needs the highest probability path, the dynamic time warp is actually given $1 - probabilitymap$. The new letter-to-phones map is then the average of all these dynamic time warp-derived mappings from letters to phones.

Figure 3.1 shows the initial and iterated versions of the alphabet-to-phone map. The iterated map is a little more sparse, which makes sense because each letter in a

[1] Compiled by Tony Robinson. ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries/beep.tar.gz.

word is being assigned to a single phone. The initial linear approximation means that if the number of letters in a word does not match the number of phones, the letters will be assigned fractions of one or more phones.

With this probabilistic map in hand, finding the syllable-to-phones relationship for a new word is simple. First, the phones for the word are found in the BEEP dictionary. Then, a dynamic time warp on the alphabet-to-phone map is used to associate letters with phones. Finally, the phones are broken up into syllables based on which syllable they map to.

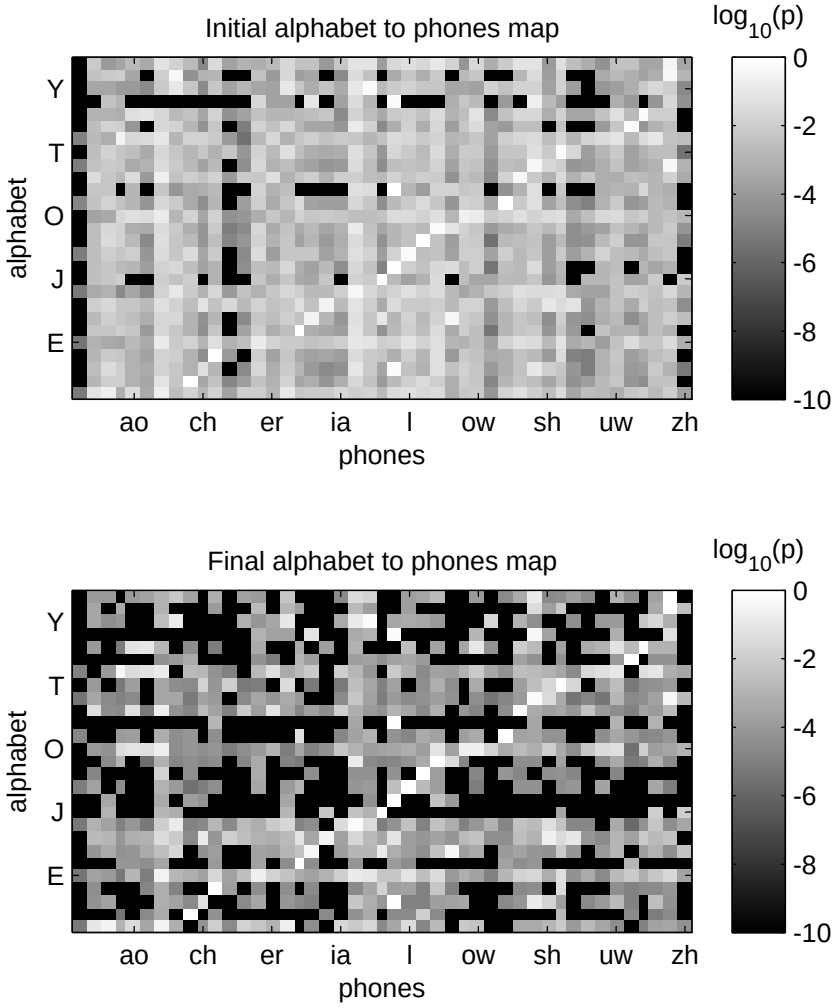

Figure 3.1: Alphabet-to-phones probability map using a log scale. The top plot shows the initial estimate of the alphabet-to-phones map using a linear mapping from letters to phones in the database. The bottom plot shows the alphabet-to-phones mapping after the algorithm converges.

word	pronunciation
a-blaze	[ax] - [b l ey z]
a-go	[ax] - [g ow]
an-ces-tral	[ae n] - [s eh s] - [t r ax l]
a-ny	[eh] - [n iy]
be-side	[b ih] - [s ay d]
ea-sy	[iy] - [z iy]
glo-ries	[g l ao] - [r ih z]
guil-ty	[g ih l] - [t iy]
low-ly	[l ow] - [l iy]
Mar-co's	[m aa] - [k ow z]
mat-ter	[m ae t] - [t ax]
pass-ing	[p aa s] - [ih ng]
pil-lows	[p ih l] - [l ow z]
Utt-ered	[ah t] - [ax d]
wi-dow	[w ih] - [d ow]

Table 3.1: Syllable-level pronunciation results. All these pronunciations seem to have been broken up sensibly.

3.2 Discussion

Table 3.1 shows the syllable-level break up of a random selection of 10% of the multi-syllable words found in the MIDI files from the Gilbert and Sullivan data set (section 2.4). All of the words are broken up correctly.

Since the results are so good at the syllable level, it is interesting to look at the alignment for a word where there is not a one-to-one correspondence between letters and phones. Figure 3.2 shows the dynamic time warp for the word 'passing,' which is interesting because of the double s and the 'ing.' As might be expected, both s letters map to the same 's' phone. Interestingly, the 'ing' ending is a different story. A human would probably want to match $i \rightarrow ih$ and $ng \rightarrow ng$. Looking at the probability map, the likelihoods $n \rightarrow ih$ and $n \rightarrow ng$ are both reasonably high, but the time warp picks the higher value and maps $n \rightarrow ih$.

So interestingly, the full letters-to-phones path may not be completely accurate or satisfying, but this does not necessarily hurt the syllabification. Although table 3.1 shows only a tenth of the entire word set, other words have also been examined and the syllabification is fine. The explanation for this disconnect may lie in the way English uses certain combinations of letters. There are lots of these letter combinations, such as 'ing' or 'ion.' In many cases, the pronunciation of the letters in these letter combinations is quite different from their nominal pronunciations. So it is not surprising when the simplistic letters-to-phones model used in this chapter has problems. Fortunately, the process of syllabification is not arbitrary. Even though different people may have slightly different ideas about where to place syllable boundaries, it is unlikely that anyone would place a boundary in the middle of one of these letter combinations. Since the probabilistic model only really needs to be accurate at the syllable boundaries, it can make mistakes elsewhere and still produce the correct results.

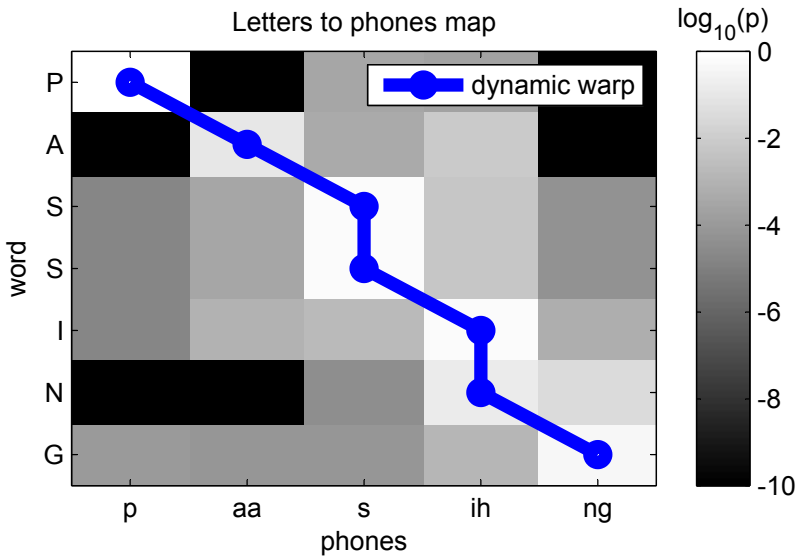

Figure 3.2: Mapping 'passing' to 'p aa s ih ng'. The letter-to-phones probability map for 'passing' and 'p aa s ih ng' is a generated by taking the appropriate columns and rows from the converged alphabet-to-phones map (figure 3.1, bottom plot). The line shows the results of the dynamic warp. Once again, the probabilities are shown in the log scale.

3.3 Summary

Lyrics in MIDI files are often aligned by syllable. Pronunciation dictionaries generally give word-level pronunciations. This chapter outlines a method by which word-level phones can be translated into syllable-level phones using a probabilistic mapping from letters to phones. Since syllables generally have only one vowel, knowing the pronunciation for the syllable means that finding the vowel for a note is simple.

Chapter 4

Solo voice detection

The end goal of this work is to characterize professional singers' voices. To be able to make this characterization, a fair amount of data is needed, but getting the data is not an easy task. Collecting data manually by asking singers to perform in a laboratory provides very clean, but limited data. In contrast, commercial recordings offer the promise of enormous amounts of data from environmentally valid sources, but the generally polyphonic nature of the recordings makes extracting an individual singer's voice difficult. The approach laid out in this chapter, which was first published in [43], attempts to combine the advantages of both laboratory recordings and commercial recordings by searching for unobstructed or "solo" sections in the middle of otherwise polyphonic recordings.

A solo musical passage will for the most part consist of a single note (pitch) sounding at any time. The spectral structure of an isolated pitch is characteristically simple, consisting of well-defined, regularly spaced harmonic spectral peaks(as illustrated in the top pane of figure 4.1) and this should allow a classifier to distinguish these frames from either multiple simultaneous voices (middle pane) which exhibit a much more complex pattern of superimposed and interacting harmonic series, or

silent gaps (bottom pane) which reveal a frequency-dependent noise floor.

Several approaches were attempted. The baseline system adopts the same approach used for detecting when a singer is active during accompanied music [8] by training a classifier on the ubiquitous Mel-frequency cepstral coefficients (MFCCs) borrowed from speech recognition.

A second approach involved seeing if the specific structural details visible in figure 4.1 could be employed directly. The first idea was to attempt to spot the 'gaps' between the harmonics of the solo voice which might be expected to revert to the noise floor. However, it was difficult to make this approach work, particularly as the voice pitch became lower and the 'gaps' became smaller.

The most successful approach is based on the idea of attempting to model a short-time frame of the signal as consisting of a single periodicity, canceling energy at that period with an appropriate comb filter (i.e. subtracting the signal delayed by one period from itself), then seeing what proportion of the total signal energy is removed. When the signal consists largely or wholly of a single periodicity, it should be possible to cancel virtually all of the periodic (tonal) energy, leading to a very large drop in energy after the filter.

In general, however, the optimal period will not be an integer number of samples, so a fractional-delay filter is required. The next section describes the approach to finding this filter, then section 4.3 describes the experiments with this detector, comparing it to the MFCC-based baseline. Section 4.4 discusses the filter and how it works in practice and finally section 4.5 summarizes the chapter.

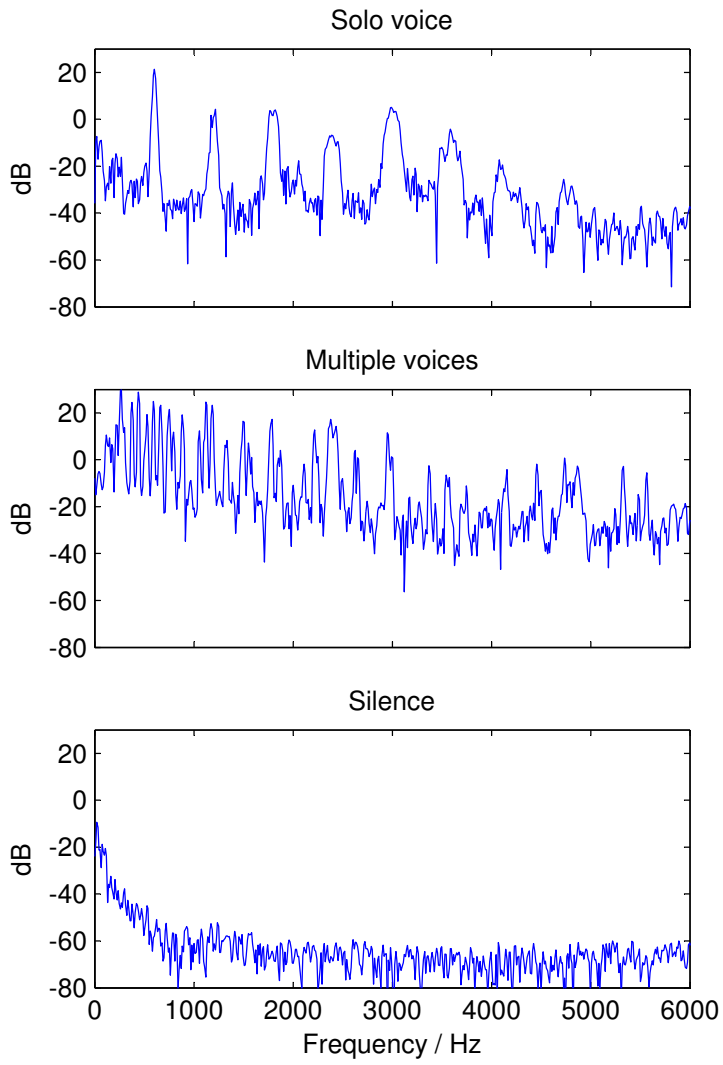

Figure 4.1: Example spectra taken from solo voice (top pane), ensemble accompaniment (middle pane), and background silence (bottom pane).

4.1 Optimal Periodicity Cancellation

By definition, a single voice has a single pitch (in the sense of a fundamental frequency), which, for musical voices, will often be relatively stationary. To detect if only a single voice is present, this approach is to find the best-fitting single period, cancel its energy, and see how completely the cancellation has removed the energy of the frame. Solo voices will have only their aperiodic energy left, resulting in a large drop in energy. Polyphonies consisting of several instruments playing different pitches will have canceled only one of the periodicities, leading to a much smaller drop in energy.

After breaking up the sound files into 93 ms frames (4096 samples at 44.1 kHz sampling rate), autocorrelation is used to obtain an initial estimate, τ, of the dominant fundamental period for each frame by finding the largest peak in the autocorrelation. A simple filter (figure 4.2, top) might then be able to remove that frequency and all its harmonics:

$$\epsilon[n] = x[n] - x[n - \tau]. \tag{4.1}$$

If τ exactly matches the period of a purely periodic waveform within the frame, $\epsilon[n]$ should be identically zero. This is exactly the same as an approach taken by Kim and Whitman to find sections of singing in pop music [26].

The problem with this scheme is that, in general, the period of an acoustic source will not correspond to an integer number of samples. This problem has been encountered in many previous circumstances including the "long-term predictor" of traditional vocoders [4] and the delay lines at the heart of physical modeling music synthesis [42]. To get around this limitation, a slightly more complicated filter is

used to optimally remove the voice (figure 4.2, bottom),

$$\epsilon[n] = x[n] - \sum_{i=-k}^{k} a_i \cdot x[n - (\tau + i)] \tag{4.2}$$

or

$$\mathbf{e} = \mathbf{x} - \mathbf{Z}\mathbf{a} \tag{4.3}$$

where $\mathbf{e}_i = \epsilon[i]$, $\mathbf{x}_i = x[i]$, $\mathbf{Z}_{i,j} = x[i-(\tau+j)]$, $\mathbf{a}_j = a[j]$; $i\epsilon[0, N-1]$ and $j\epsilon[-k, k]$. The filter uses $k = 3$ for a seven-coefficient filter as a more or less arbitrary compromise between computational complexity and flexibility of the cancellation filter.

The question is how to calculate the coefficients $a[j]$. One solution is to pick them to optimally reduce the energy of $\epsilon[n]$,

$$\hat{\mathbf{a}} = (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{x}. \tag{4.4}$$

This will not generally create exactly a sub-sample delay filter (see section 4.4), but it should do a reasonable job of removing the energy of non-integer wavelength sources. Having solved for these coefficients within each frame, the filter is applied to find the energy of the residual $\epsilon[n]$ within the frame, then the ratio of the residual energy to the energy of the original signal $x[n]$ is calculated. In the case of a purely periodic signal whose period is a non-integer number of samples, $\hat{\mathbf{a}}$ should approximate an ideal fractional delay filter (sinc interpolator) which can exactly cancel the periodic signal, leading to a residual-to-original ratio close to zero. When the signal consists of many periodicities, only a small proportion of the energy will be canceled by eliminating just one dominant periodicity.

In frames consisting of "silence" (noise floor), however, a single spectral peak may account for a large proportion of the very small amount of energy. In this case, the

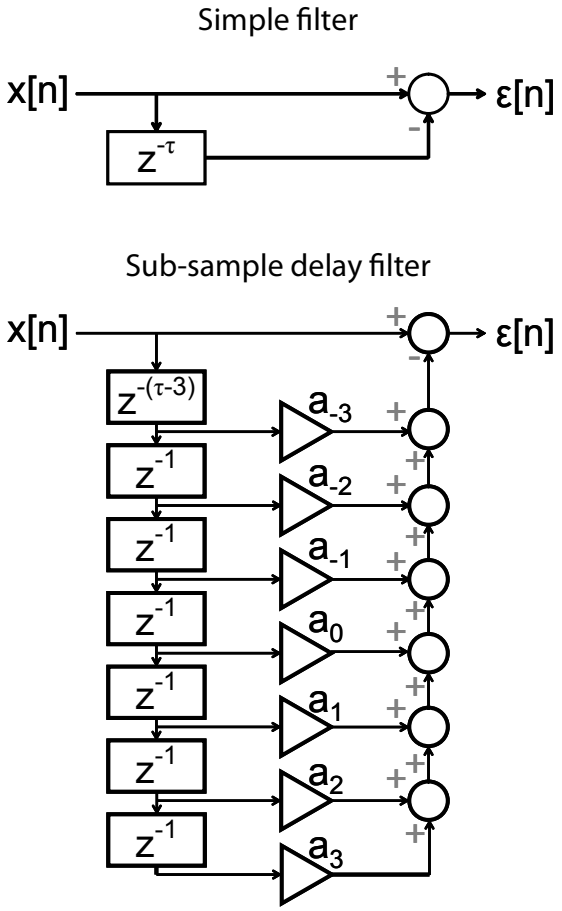

Figure 4.2: Optimal cancellation filters. Top: for signals with integer periodicities; bottom: a filter able to cancel non-integer periodicities.

optimal cancellation filter may also be able to remove a large proportion of the energy. To differentiate between silent frames and single voice frames, a second feature is added with a value related to each frame's original energy. To avoid any issues arising from global scaling of the original sound files, the entire waveform is normalized to make the 98th percentile of the short-time Fourier transform magnitude equal to 1.

4.2 Classifier

The two-dimensional feature consists of the residual-to-original energy ratio and the normalized absolute energy as a two-dimensional feature. These features are fed into a simple Bayesian classifier to estimate the probability that each frame belongs to each of three classes – solo voice, multiple voices, and silence. The model distribution parameters for each class are calculated from a small amount of hand-labeled training data (see section 4.3). The normalized absolute energy is fit with a Gaussian in the log (dB) domain. The residual-to-original energy ratio, however, always lies between 0 and 1, and is heavily skewed toward 0 in the solo class. A Gaussian is thus a poor fit, and no simple transformation will make all the classes appear Gaussian. Instead, a Beta distribution is used for each category. The Beta distribution is defined over $[0, 1]$ and has two parameters to fit both the mode and spread of the observed class-conditional values. The two features are treated as independent, so the overall likelihood is of a particular observation frame under each class is calculated by multiplying the Gaussian and Beta likelihoods together for that class. Simple ML classification then is used to label according to the largest posterior.

Independent classification of each time frame can result in rapid alternation between class labels, whereas real data changes state relatively infrequently. A three-state hidden Markov model (HMM) with transition probabilities set to match the empirical frame transition counts in the training data is used to model the actual transition probabilities. The single most likely label sequence given this transition model and the class-dependent likelihoods is calculated using the Viterbi algorithm[1]. Figure 4.3 shows an example of label sequences before and after HMM smoothing, compared to the ground-truth labels.

[1]Kevin Murphy's Matlab implementation [37] is used for these calculations.

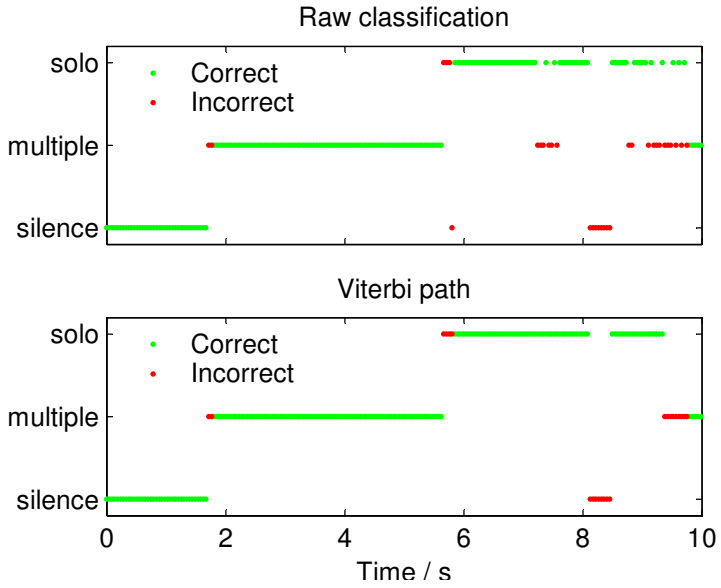

Figure 4.3: Example output from the cancellation-based classifier. Top pane shows raw frame-level results, bottom pane shows the result of HMM smoothing to remove rapid switching between states.

To trade precision for recall, the model can be biased to generate more or fewer "solo" labels simply by scaling the solo model likelihood by a constant value. Smaller likelihoods for the "solo" class result in fewer, more confidently "solo" labels. In this application, assuming a very large underlying archive to search, having a low recall (only a small portion of all possible solo regions are identified) in order to achieve a higher precision (nearly all of the identified regions are, in fact, solo regions) is probably acceptable.

4.3 Experiments and results

The data set consists of twenty 1 minutes samples that are hand-labeled as silence, solo, or multiple voices. The samples are taken from a variety of professional folk and classical recordings. About 28% of the frames in the data set contain a solo

voice. The other 72% of data frames contain multiple voices or silence. Ten samples are used in calculating the distribution and Viterbi path parameters. The remaining ten samples are used for testing.

The baseline classifier, as mentioned in the introduction, is a 'generic' audio classifier based on the MFCC feature vectors commonly used in speech recognition and that have also shown themselves very successful in many music classification tasks [32, 8]. The first 13 cepstral coefficients, normalized so their means are zero within each track to eliminate any fixed filtering effects, are used. The same Bayesian classifier structure is used, but in this case each of the three classes is fit by a full-covariance multidimensional Gaussian. Netlab is used for this modeling [38].

Figure 4.4 shows the results of the cancellation- and MFCC-based classifiers on all the test data combined. The different precision/recall points are obtained by manipulating the solo likelihood. Above about 90% precision, the MFCC and cancelation systems perform approximately equally. At lower precision levels, however, the cancelation algorithm has a much better recall. At 80% precision and below, the cancelation algorithm has at least 10% higher recall than the MFCC system.

The cancellation system also exhibits more consistent performance. When comparing the frame labeling accuracy on individual tracks in the test set, the variance of the cancellation system performance is half that of the MFCC system. This is probably because the pitch cancellation algorithm has many fewer learned parameters (4 per class, compared to 104 per class for the MFCC) and thus is less susceptible to over fitting.

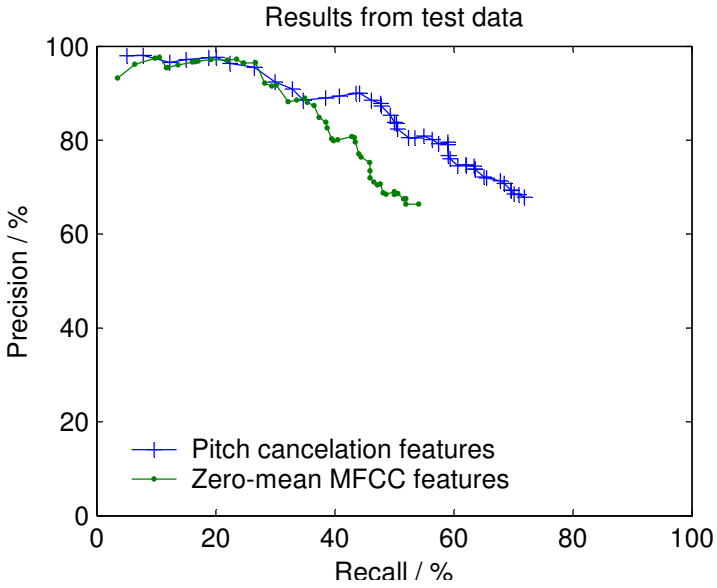

Figure 4.4: Solo voice detection precision/recall trade off for the two approaches.

4.4 Discussion

Although the least-squares optimal filter is used simply as a way to achieve precise fitting of non-integer-period signals, it is interesting to consider what we get as a result. The filter is optimized to minimize output energy, subject to the constraints that (a) the first value of the impulse response is 1; (b) the next $\tau - 4$ are zero; and (c) the total length is $\tau + 3$, where τ is the initial, coarse estimate of the dominant period. This filter is not constrained to include an exact unit-gain, linear-phase fractional delay, and in general energy will be minimized by a more complex response subject to the constraints. The seven free parameters of the system allow a certain error tolerance in the coarse period estimation as well as making it possible to match an ideal sync fractional delay more accurately, but they also permit a more complex range of solutions; solving for longer filter blocks would result in filters that deviate increasingly far from our intention of a simple comb canceling a single period.

Results from running the optimal cancellation algorithm on a partial music track can be seen in figure 4.5. Silence starts the track and then the full orchestra enters ($\sim$ 2 seconds). The orchestra drops out ($\sim$ 6 seconds) for the soloist's entrance and then joins back in the mix ($\sim$ 10 seconds). While the orchestra is playing, the spectrum is quite dense and the comb filter cannot adapt to remove energy effectively. As soon as the soloist enters, however, the filter takes advantage of the harmonic structure to remove a significant portion of the energy, particularly in the lower, higher energy bands. As mentioned previously, the 'silent' frames also have a low original-to-residual ratio because the optimal cancellation algorithm is able to cancel a large portion of the small amount of energy present. These silent frames, which have almost no energy originally, are differentiated from the more energetic solo frames by the second feature, which is related to the original energy.

As discussed in the introduction, the goal of this chapter was to find a way to accurately and automatically identify solo excerpts within a large music corpus in order to collect training data for solo source models. This cancellation system seems very suitable for this task, and the next step would be to apply the system to a large music archive to see what it can find. The ability of the system to detect periodicity without a more detailed model of the particular voice to be found is both a strength and a weakness – it's useful to be able to detect solos for instruments not in the training set, but it means that the returns from the solo detection data mining will themselves need to be clustered and classified to build separate models for distinct instruments.

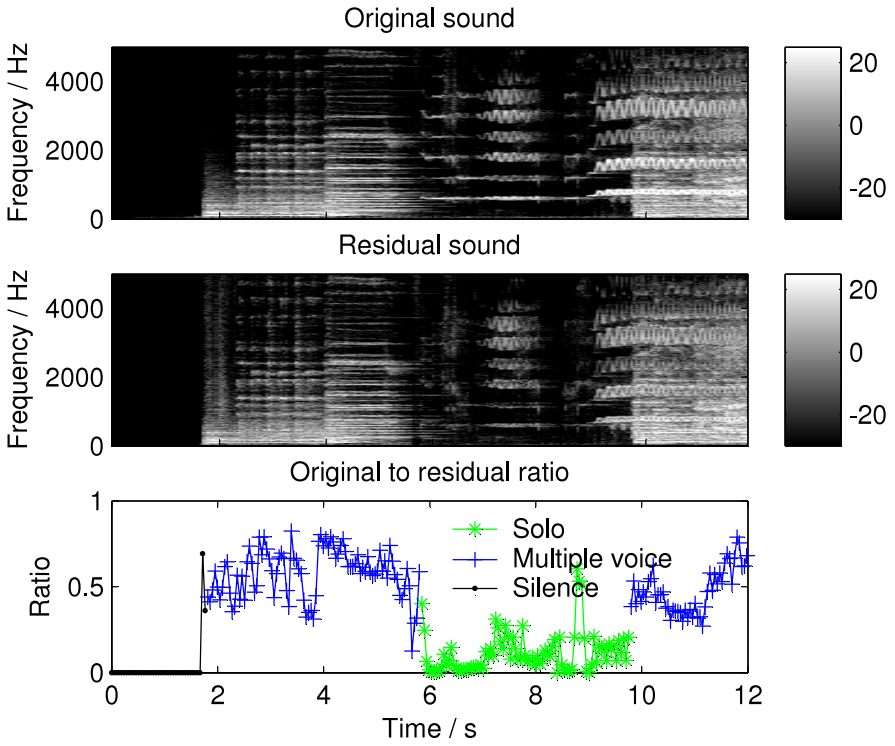

Figure 4.5: Comparison of a wave file before (top) and after (middle) filtering out the main pitch showing a significant reduction in energy in the area labeled 'solo' voice.

4.5 Summary

Most recordings of western music are essentially polyphonic. However, in these polyphonic recordings, there may be windows of true "solos." These windows offer unobstructed views of a voice that can be analyzed further.

This chapter addresses the issue of how to automatically find these solo passages. The basic strategy is to do the best job possible of canceling a single voice in a short time window. If the canceled energy accounts for a substantial portion of the energy in that window, then the window is most probably monophonic. In practice, essentially silent windows can sometimes appear to by monophonic using

this calculation, so the overall energy of the initial window is also needs to be taken into account. This strategy works reasonably well, but can be noisy, so the estimates are smoothed using an HMM.

Chapter 5

Frequency estimation

The end goal of this research is to characterize sung vowels in commercial recordings which offer the promise of huge, environmentally valid data sets. Unfortunately, the very thing that makes these recordings interesting, the fact that they are created with artistic constraints rather than research constraints, makes them difficult to work with. Most commercial recordings are polyphonic mixes of instruments and voices. The previous chapter tried to get around this difficulty by searching for 'solo' voices hidden in the middle of otherwise polyphonic recordings. Unfortunately, there simply is not much 'solo' time to find. So this chapter starts down a different path where the polyphony found in most recordings is tackled head-on. The necessary trick is to isolate the voice of interest in the recordings enough to calculate its most important characteristics, namely the power in the voice overall and the relative power in each harmonic of the voice.

Isolating the voice really boils down to fundamental frequency or pitch estimation. Once the exact fundamental frequency is known, the frequencies of the harmonics are also known. Assuming these harmonics are sufficiently isolated in frequency and time, the strength of the harmonics can be estimated.

Finding notes in a polyphonic mix is difficult when there are no constraints on the pitch, the number of notes, or the location of the notes in time. The question is how to add constraints sensibly so that the problem becomes more manageable. A large portion of recorded music is derived from a score of some sort. Many of these scores can be found in electronic form, such as MIDI files. With this electronic score, there is suddenly a lot of really good information about approximate fundamental frequencies of the notes in the form of note names (A4, C3, etc.). If the score is time-aligned with the recording, then there is also very good information on when each notes occur. Furthermore, since scores are generally broken up by part, it is easy to get all the notes associated with just the voice of interest.

The rest of this chapter covers two different approaches to using approximate pitch and timing information from aligned electronic scores in order to accurately estimate fundamental frequencies in polyphonic recordings. Section 5.1 covers the basic signal model that both estimates use. Section 5.2 covers the noise model used by the Bayesian estimator (section 5.3), which was first presented in [44]. This estimator is reasonably accurate, but very slow. Section 5.4 covers a template-matching approach that is much faster, but not as accurate. Section 5.5 describes the experiments used to evaluate the two approaches. Section 5.6 discusses the results from the experiments. Finally, section 5.7 summarizes the chapter.

5.1 Basic signal model

A harmonic signal can be written as a sum of harmonics,

$$x[n] = \sum_{i \in H} h_i[n] \tag{5.1}$$

where H is the set of harmonic numbers (say $\{1, 2\}$ for the first and second harmonics) and each harmonic is a sinusoid[1],

$$h_i[n] = \begin{cases} A_i \cos(\omega i n + \phi_i) & -\frac{N}{2} + 1 \leq n < \frac{N}{2} - 1 \\ 0 & n = \frac{N}{2}, \end{cases} \tag{5.2}$$

where A_i is the strength of harmonic i, ω is the fundamental frequency in radians per sample, ϕ_i is the phase offset of harmonic i, and N is the window size.

In this analysis, $h_i[n]$ simply repeats outside of the range of the window $n \in \{-\frac{N}{2} + 1, \cdots \frac{N}{2}\}$, so $h_i[m] = h_i[m + N]$. In particular, the Fourier transforms are all calculated over the range $n = 0 \cdots N - 1$.

To reduce side lobe effects, a Hann window is used,

$$w[n] = \frac{1}{2} \cdot \left(1 + \cos\left(\frac{2\pi n}{N}\right)\right), \tag{5.3}$$

so

$$x_w[n] = \sum_{i \in H} (w[n] \cdot h_i[n]) = \sum_{i \in H} h_{i,w}[n]. \tag{5.4}$$

Moving to the frequency domain next makes sense because it separates out the harmonics. Consider a set of indexes

$$k_{h,i} \in \{k_{0,i} - d, \cdots, k_{0,i}, \cdots, k_{0,i} + d\} \tag{5.5}$$

centered around the harmonic's nominal frequency ω_0

$$k_{0,i} = round\left(\frac{\omega_0 N i}{2\pi}\right), \tag{5.6}$$

[1] A zero point is added to the definition of $h_i[n]$ to make the expression for $H_i[k]$ (equation (5.7)) simpler. Note that the windowing function, equation (5.3), is also zero at $n = \frac{N}{2}$, so artificially adding a zero to $h_i[n]$ here has no actual effect on the final windowed signal.

where i is the harmonic number and d is some reasonable range around $k_{0,i}$. In this range, $X_w[k_{h,i}]$ is dominated by the one harmonic in question and is essentially equal to $H_i[k_{h,i}]$.

In the Fourier domain, calculated from $n = 0$ to $n = N - 1$, the harmonic is

$$H_i[k] = \frac{A_i}{2}\left(e^{j\phi_i}F_{\frac{N}{2}-1}\left(\frac{2\pi k}{N} - \omega i\right) + e^{-j\phi_i}F_{\frac{N}{2}-1}\left(\frac{2\pi k}{N} + \omega i\right)\right), \tag{5.7}$$

where

$$F_M(\theta) = 2\cos\left(\theta \cdot \frac{M}{2}\right) sinc_{M+1}\left(\frac{\theta}{2}\right) - 1 \tag{5.8}$$

and *sinc* is a periodic sinc function,

$$sinc_M(\theta) = \frac{\sin(\theta M)}{\sin(\theta)}. \tag{5.9}$$

The Hann window has a simple 3-point Fourier Transform, so the windowed harmonic is simply

$$H_{w,i}[k] = \frac{1}{2}H_i[k] + \frac{1}{4}H_i[k-1] + \frac{1}{4}H_i[k+1]. \tag{5.10}$$

5.2 Noise model

Around each harmonic, the signal is corrupted by noise,

$$Y_w[k_{h,i}] = H_{w,i}[k_{h,i}] + N_{w,i}[k_{h,i}] \tag{5.11}$$

where $Y_w[k_{h,i}]$ is the Fourier transform of the full input around harmonic i and $N_{w,i}[k_{h,i}]$ is the noise around harmonic i.

To derive $N_{w,i}[k_{h,i}]$, start with broad-band white noise in the time domain,

$$n_i[n] \sim N(0, \sigma_{n,i}^2). \tag{5.12}$$

In the Fourier domain, it is easier to notate $N_i[k]$ as a vector, $\boldsymbol{N}_i$, and so

$$Pr\left(\boldsymbol{N}_i\right) = Pr\left(\Re\left(\boldsymbol{N}_i\right)\right) \cdot Pr\left(\Im\left(\boldsymbol{N}_i\right)\right) \tag{5.13}$$

because the real and imaginary parts of $\boldsymbol{N}_i$ are independent. Furthermore,

$$Pr\left(\Re\left(\boldsymbol{N}_i\right)\right) \sim N\left(\mathbf{0}, \sigma_{n,i}^2 \cdot \boldsymbol{\Sigma}_R\right) \tag{5.14}$$

and

$$Pr\left(\Im\left(\boldsymbol{N}_i\right)\right) \sim N\left(\mathbf{0}, \sigma_{n,i}^2 \cdot \boldsymbol{\Sigma}_I\right), \tag{5.15}$$

where $\boldsymbol{\Sigma}_R$ and $\boldsymbol{\Sigma}_R$ are both $N \times N$ dual diagonal matrices offset by one row and one column. Using zero-indexing, where k is the row and l is the column,

$$\boldsymbol{\Sigma}_R(k,l) = \begin{cases} N & \text{if } k = l = 0 \text{ or } \frac{N}{2} \\ \frac{1}{2}N & \text{if } k = l, k \neq 0, k \neq \frac{N}{2} \\ \frac{1}{2}N & \text{if } k = N - l, k \neq 0, k \neq \frac{N}{2} \\ 0 & \text{elsewhere} \end{cases} \tag{5.16}$$

and

$$\boldsymbol{\Sigma}_I(k,l) = \begin{cases} \frac{1}{2}N & \text{if } k = l, k \neq 0, k \neq \frac{N}{2} \\ -\frac{1}{2}N & \text{if } k = N - l, k \neq 0, k \neq \frac{N}{2} \\ 0 & \text{elsewhere.} \end{cases} \tag{5.17}$$

For the windowed noise, $\boldsymbol{N}_{w,i}$, the real and imaginary parts are similarly independent and

$$Pr\left(\Re\left(\boldsymbol{N}_{w,i}\right)\right) \sim N\left(\mathbf{0}, \sigma_{n,i}^2 \cdot \boldsymbol{B}\boldsymbol{\Sigma}_R\boldsymbol{B}^{\mathrm{T}}\right) \tag{5.18}$$

and

$$Pr\left(\Im\left(\boldsymbol{N}_{w,i}\right)\right) \sim N\left(\mathbf{0}, \sigma_{n,i}^2 \cdot \boldsymbol{B}\boldsymbol{\Sigma}_I\boldsymbol{B}^{\mathrm{T}}\right) \tag{5.19}$$

where $\boldsymbol{B}$ is an $N \times N$ matrix,

$$\boldsymbol{B}(k,l) = \begin{cases} \frac{1}{2} & \text{if } k = l \\ \frac{1}{4} & \text{if } k = l-1 \text{ or } l+1 \pmod N \\ 0 & \text{elsewhere.} \end{cases} \tag{5.20}$$

$N_{w,i}$ only of interest around harmonic i. To calculate just $Pr(N_{w,i}[k_{h,i}])$, simply use the submatrices $\boldsymbol{\Sigma}_R(k_{h,i}, k_{h,i})$ and $\boldsymbol{\Sigma}_I(k_{h,i}, k_{h,i})$ as the covariances of the real and imaginary parts of the noise. Thus, the noise model is a narrow-band piece of broad-band white Gaussian noise.

5.3 Bayesian estimation

This estimator calculates a maximum a posteriori (MAP) estimate of the pitch, ω. Because there is no reasonable way of knowing in advance the harmonic strengths, $A_{i\in H}$, and phases, $\phi_{i\in H}$, these are also estimated in addition to the noise parameters, $\sigma_{n,i\in H}$. So,

$$\hat{\boldsymbol{\theta}}_{MAP} = \underset{\theta}{\operatorname{argmax}}\, Pr\left(\boldsymbol{\theta}|\boldsymbol{Y}_w, \omega_0\right), \tag{5.21}$$

where $\boldsymbol{\theta}$ is a vector of parameters, [ω, $A_{i\in H}$, $\phi_{i\in H}$, $\sigma_{n,i\in H}$], $\boldsymbol{Y}_w$ is the Discrete Fourier Transform (DFT) of the windowed input, and ω_0 is the nominal pitch in the current window.

5.3.1 Defining probabilities

Using Bayes,

$$Pr\left(\boldsymbol{\theta}|\boldsymbol{Y}_w, \omega_0\right) \propto Pr\left(\boldsymbol{Y}_w|\boldsymbol{\theta}, \omega_0\right) Pr\left(\boldsymbol{\theta}|\omega_0\right). \tag{5.22}$$

The prior, $Pr\left(\boldsymbol{\theta}|\omega_0\right)$, needs to capture the idea that the pitch of the current window, ω, should be close to the nominal pitch from our score, ω_0, so assign

$$Pr\left(\boldsymbol{\theta}|\omega_0\right) \propto Pr\left(\omega|\omega_0\right) \sim N(\omega_0, \sigma_\omega^2) \tag{5.23}$$

where σ_ω^2 is the variance of the pitch around ω_0.

Returning to equation (5.22), and assuming that $\boldsymbol{Y}_w$ only depends on the nominal pitch ω_0 via the actual pitch ω, define

$$Pr\left(\boldsymbol{Y}_w|\boldsymbol{\theta}, \omega_0\right) = \prod_{i \in H} Pr\left(Y_w[k_{h,i}] \mid A_i, \phi_i, \sigma_{n,i}, \omega\right). \tag{5.24}$$

This essentially combines the information from different harmonics by assuming that the conditional probabilities associated with each harmonic, $Pr\left(Y_w[k_{h,i}] \mid A_i, \phi_i, \sigma_{n,i}, \omega\right)$, are independent. Since the signal spectrum $H_i[k_{h,i}]$ is completely specified by the givens in equation (5.24), $N_{w,i}[k_{h,i}] = Y_w[k_{h,i}] - H_i[k_{h,i}]$ can be calculated. So

$$Pr\left(Y_w[k_{h,i}] \mid A_i, \phi_i, \sigma_{n,i}, \omega\right) = Pr\left(N_{w,i}[k_{h,i}] \mid \sigma_{n,i}\right) \tag{5.25}$$

where $Pr\left(N_{w,i}[k_{h,i}] \mid \sigma_{n,i}\right)$ is specified in section 5.2.

5.3.2 Finding the maximum

Finding the maximum of equation (5.21) directly is difficult. The Nelder-Mead Simplex Method [31] as implemented by Matlab's fminsearch() is used here to search

for a maximum, but any suitable maximization algorithm could be used. The main difficulty lies in seeding the search algorithm sufficiently close to the global optimum to avoid local maxima.

The first step is to calculate approximate values for ω and $A_{i\in H}$. Note that

$$H_i[k_{h,i}] \approx \frac{A_i}{2} F_{\frac{N}{2}-1}\left(\frac{2\pi k}{N} - \omega i\right) e^{j\phi_i}, \tag{5.26}$$

and so

$$\begin{aligned} H_{w,i}[k_{h,i}] \approx e^{j\phi_i}\Bigg(&\frac{A_i}{4} F_{\frac{N}{2}-1}\left(\frac{2\pi k}{N} - \omega i\right) \\ &+ \frac{A_i}{8} F_{\frac{N}{2}-1}\left(\frac{2\pi (k-1)}{N} - \omega i\right) \\ &+ \frac{A_i}{8} F_{\frac{N}{2}-1}\left(\frac{2\pi (k+1)}{N} - \omega i\right)\Bigg) \end{aligned} \tag{5.27}$$

using equation (5.10). Since $|X_w[k_{h,i}]| \approx |H_{w,i}[k_{h,i}]|$ there is an expression which is only a function of ω and $A_{i\in H}$. Also note that near the peak values of $|Y_w[k]|$,

$$|Y_w[k_{peak,i}]| \approx |X_w[k_{peak,i}]|, \tag{5.28}$$

where $k_{peak,i}$ consisted of the two largest values next to each other in $|Y_w[k_{h,i}]|$ for each harmonic i. The initial estimates are thus a best fit of ω and $A_{i\in H}$ to the approximation in equation (5.28).

Initial phases ϕ_i are simply the phases of the peak value of $Y_w[k_{h,i}]$,

$$\phi_i \approx \angle Y_w[\underset{k\in k_{h,i}}{\operatorname{argmax}} |Y_w[k]|]. \tag{5.29}$$

For the estimate of $\sigma_{n,i}$, note that in equations (5.18) and (5.19), the noise covariance scales linearly with $\sigma_{n,i}^2$. The estimates of ω, $A_{i\in H}$, and $\phi_{i\in H}$ are not

always good enough to calculate $N_w[k_{h,i}]$ accurately from $Y_w[k_{h,i}]$, particularly around the peaks. So, instead estimate each $\sigma^2_{n,i}$ from the non-peak values of $Y_w[k_{h,i}]$,

$$\sigma^2_{n,i} \approx \frac{\sum_{i \in k_{h,i} \setminus k_{peak,i}} |N_w[i]|^2}{\sum_{i \in k_{h,i} \setminus k_{peak,i}} (\boldsymbol{B}\boldsymbol{\Sigma}_R\boldsymbol{B}^{\mathrm{T}}) + \sum_{i \in k_{h,i} \setminus k_{peak,i}} (\boldsymbol{B}\boldsymbol{\Sigma}_I\boldsymbol{B}^{\mathrm{T}})}. \tag{5.30}$$

Thus, for a given window of the original signal, the initial estimates of ω and $A_{i \in H}$ are derived using equations (5.26) to (5.28), $\phi_{i \in H}$ is derived from equation (5.29), and $\sigma_{n,i \in H}$ is derived using equation (5.30). These values are passed to the optimizer to maximize equation (5.22), as defined in equations (5.23) to (5.25).

5.4 Template estimation

The main problem with the Bayesian estimator is that the optimization step simply takes a long time. This template-matching algorithm is an order of magnitude faster, although it is not as accurate.

In the frequency domain, a harmonic signal is essentially a set of peaks. This algorithm slides a template, also composed of peaks, over the DFT of the signal to find the best match.

The template consists of a series of Gaussian-shaped peaks

$$T_\omega[k] = \frac{1}{B} \sum_{i \in H} (G_{i,\omega}[k]) \tag{5.31}$$

where B is a normalizing constant so that $\sum_k T[k] = 1$ and $G_i[k]$ is a peak centered around ωi,

$$G_{i,\omega}[k] = exp\left\{ -\frac{\left(k - \frac{\omega i N}{2\pi}\right)^2}{2\sigma^2} \right\}, \tag{5.32}$$

where ω is the fundamental frequency of the template and σ controls the width of the peaks.

$Y_w[k]$, the DFT of the signal with noise, is symmetric because $y_i[n]$ is real. So the fit between the template and the $Y_w[k]$ only needs to be calculated over half the DFT,

$$FIT(\omega) = \sum_{k_{half}} \left| Y_w[k] \cdot T_\omega[k] \right|, \tag{5.33}$$

where $k_{half} = \left\{ k \,|\, 0 \leq k < \frac{N}{2} \right\}$.

To find the best fundamental frequency ω, equation (5.33) must be maximized,

$$\hat{\omega} = \underset{\omega}{\operatorname{argmax}}\, FIT(\omega). \tag{5.34}$$

Recall that ω_0, the nominal note frequency from the electronic score, is known and should be near ω, the true fundamental frequency. So the simplest way to find $\hat{\omega}$ is to try out a reasonable number of frequencies near ω_0 and take the best value.

5.5 Experiments and results

Experiments in this section are performed on real and simulated data. In all cases, the window size is set to 4096 (93 ms at 44.1 kHz) and the distance between successive windows is set to 1024 (23 ms) for a 75% overlap. For the bayesian estimates, the first and third harmonics are used to calculate the fundamental frequency on the multi-track data (section 5.5.2). The third harmonic is used rather than the second to reduce the chance that the harmonic because the second harmonic is only an octave away, which is a common musical interval. The third harmonic, while still low enough to contain a reasonable proportion of the energy, should also hopefully be less likely to be near interferring partials from other lines. The fundamental frequency's variance parameter, σ_ω^2 (equation (5.23)), is set to one third of 2 half steps down from the nominal frequency. This essentially means that the prior over the fundamental

frequency is nearly zero more than 2 half steps away from the nominal frequency. For the template estimates, which take essentially the same amount of time regardless of the number of harmonics used, the first three harmonics are used on the multi-track data. The template's σ parameter, which controls the width of the template peaks, is set to 50 cents down from the nominal note frequency derived from the score. The search range for the template function starts 2 half steps below the nominal fundamental frequency and ends the same number of Hz above the nominal frequency. Within this search range, 100 linearly-spaced frequencies are tested.

5.5.1 Experiments with simulated data

One of the key arguments for using multiple harmonics is that it allows the algorithms to correctly estimate pitch even when one or more harmonics are obscured. To test this hypothesis, simulated data is generated with two harmonics. The power in each harmonic ($0.5 \cdot A_i^2$) is varied so the total signal power remains constant while the power moves incrementally from the first harmonic to the second. At each harmonic power level, 100 test windows are generated with a random fundamental frequencies near 440 Hz, the nominal fundamental frequency given to each estimator. Each window has added broadband white Gaussian noise so the composite signal to noise ratio is -20 dB. Each noisy window is then evaluated by each estimator three times, once with both harmonics, once with just the first harmonic, and once with just the second harmonic.

Figure 5.1 shows the results of the test using the Bayesian estimator. As can be seen, the tests using a single harmonic struggle when that harmonic disappears into the noise, but combining the harmonics provides a consistently good estimate. The same pattern is true for the template estimator (figure 5.2), although the estimates are not as good overall as the Bayesian estimates.

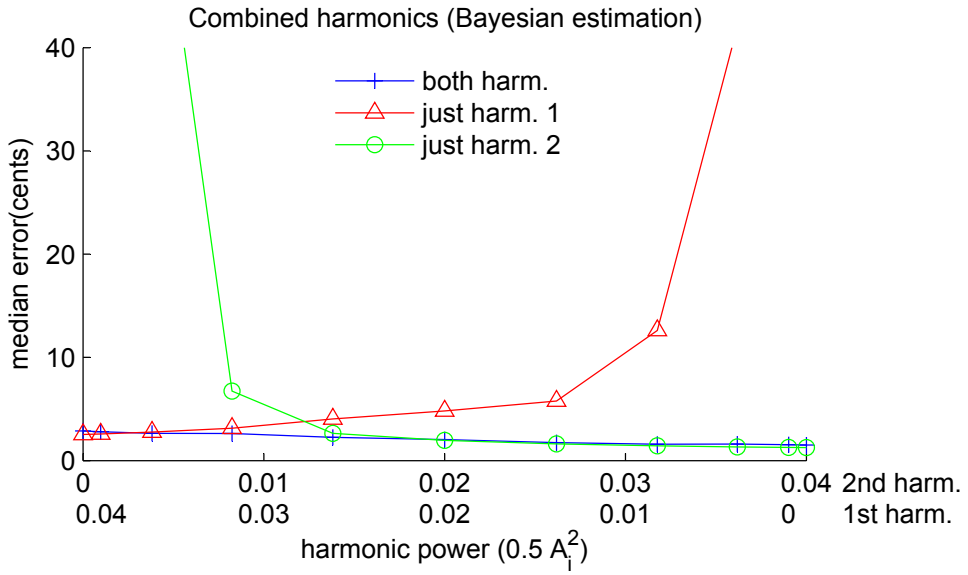

Figure 5.1: The Bayesian estimate on two harmonics as power shifts from the first harmonic to the second. The blue line with crosses represents the results using both harmonics for frequency estimation, the green line with circles represents the results using just the first harmonic, and the red line with triangles represents the results using just the second harmonic.

5.5.2 Experiments with multi-track data

Multi-track recordings, where a single line is recorded on each track, offer a simple way to get truth data for pitch tracking in polyphonic recordings. For these tests, the data consists of a multi-track recording[2] [17] of the opening to Guillaume de Machaut's Kyrie in Messe de Nostre Dame (c. 1365). Each of the four voices has been recorded individually and has been labeled with accurate start and stop times for each note. Overall, there are about 1000 windows in notes for each track.

YIN [16], a well known fundamental frequency estimation algorithm, has been run over each individual tracks to obtain truth data. For comparison purposes, the Bayesian and template algorithms are run over the individual tracks as well as on

[2]Thanks to Johanna Devaney for providing these recordings.

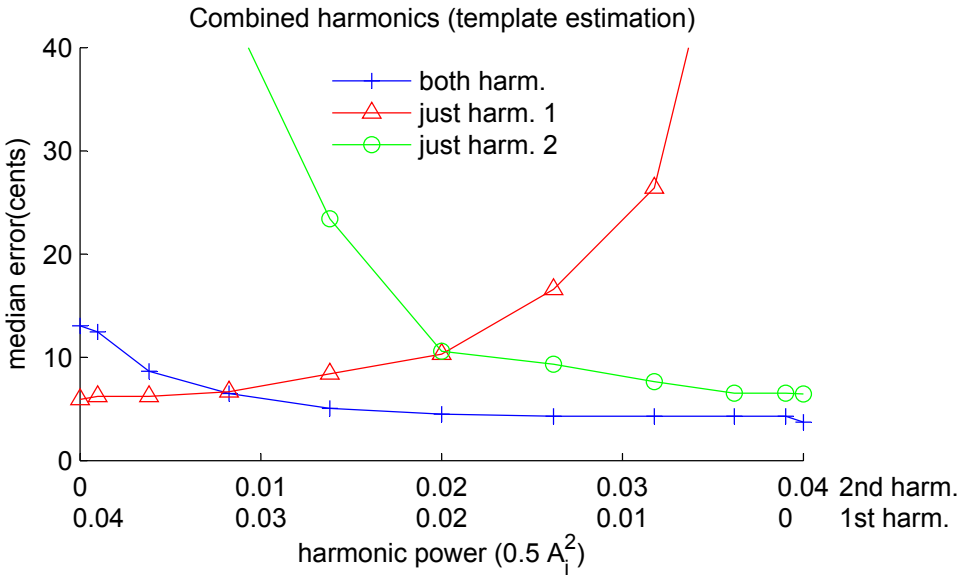

Figure 5.2: The template estimate on two harmonics as power shifts from the first harmonic to the second. Once again, the blue line with crosses represents the results using both harmonics for frequency estimation, the green line with circles represents the results using just the first harmonic, and the red line with triangles represents the results using just the second harmonic.

the full mix. The YIN algorithm actually permits specification of a frequency search range, so YIN is also run on the full mix for comparison using the same guided search range that the other algorithms use.

Figure 5.3 shows histograms of the frequency estimation errors relative to the truth data from YIN and table 5.1 summarizes these results. The Bayesian estimator agrees substantially with YIN when run on the individual vocal lines (blue line). In 93% of windows, the Bayesian estimator is within 10 cents of YIN's frequency. When run on the complete mix (green line), the Bayesian estimator still manages to get the frequency within 10 cents of the correct frequency 69% of the time.

The template algorithm is less accurate. When run on the individual lines, it is only within 10 cents 86% of the time (purple line). That being said, the granularity

of the template frequency estimate is on the order of a few cents, so some of these errors are simply rounding errors. When run on the mix, the template algorithm still fairs worse than the Bayesian estimator and is only within 10 cents in 58% of windows (black line).

Not unsurprisingly, the YIN estimate is way off in the multi-track environment. Given that YIN is only designed to work on single pitches, it is somewhat surprising that YIN worked at all. An obvious base line for comparison is to simply guess the nominal note frequency as the fundamental frequency in every window. This f_0 estimator only gets within 10 cents of the correct frequency in 4% of windows, where as YIN gets 21% of windows. So YIN is clearly able to find the fundamental frequency in at least some cases.

Figure 5.4 shows a 5 second excerpt from the recording. The top plot shows a spectrogram of the triplum or top line by itself with the YIN truth data overlaid in red. The bottom plot shows a spectrogram of the full mix with the YIN (red), Bayes (blue), and template (purple) full-mix estimates. For the most part, the estimates are actually all very close initially. Even the vibrato at about 8.5 seconds is traced out. But then at about 9 seconds, YIN starts to struggle. Note that when YIN fails to find the correct harmonic, it fails in an unpredictable fashion. In contrast, the Bayes and template algorithms seem to fail by finding the wrong harmonic. For the Bayes estimator this is clear just after 9 seconds, and for the template estimator after 9.5 seconds.

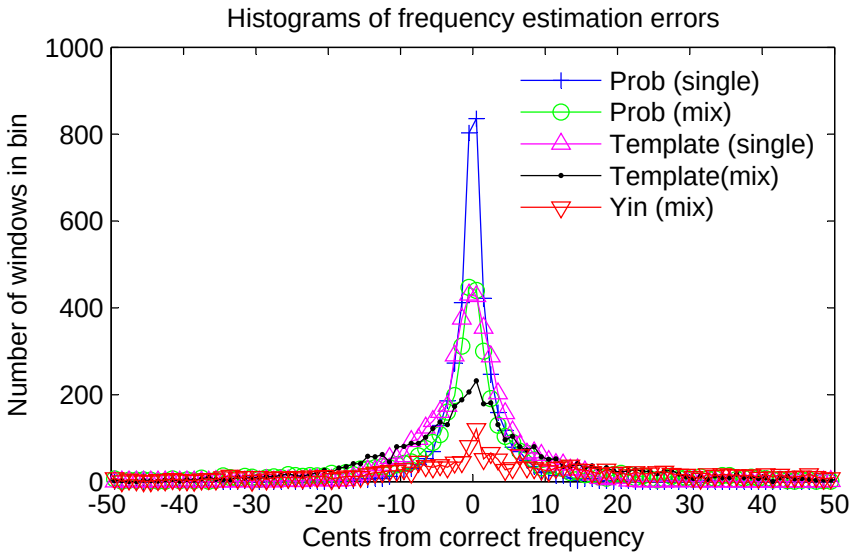

Figure 5.3: Histogram of frequency estimation errors relative to the truth data from YIN. Note that substantial counts lie outside of the range of this plot. Each estimate is run on the individual lines and the full mix.

	RMSE (Hz)	RMSE (cents)	% within 10 cents	% within 50 cents
Bayes (single)	0.743	4.77	92.9	98.9
Bayes (mix)	4.15	24.8	69.2	91.4
Template (single)	0.823	5.19	86.4	99.7
Template (mix)	3.6	21.1	58	92.6
YIN (mix)	31.1	162	21	45.9
f_0	8.45	49.7	3.88	52.4

Table 5.1: Summary of estimate results from multi-track experiments. The results are relative to the truth data generated by YIN from the single track recordings. The Bayes and template estimate errors are given for both the individual lines and the full mix. The f_0 estimate basically uses the nominal note frequency as the estimate in all windows. The high errors for this estimate are partly due to the fact that the singers are all slightly flat relative to the A440 tuning standard, although they are in tune relative to one another.

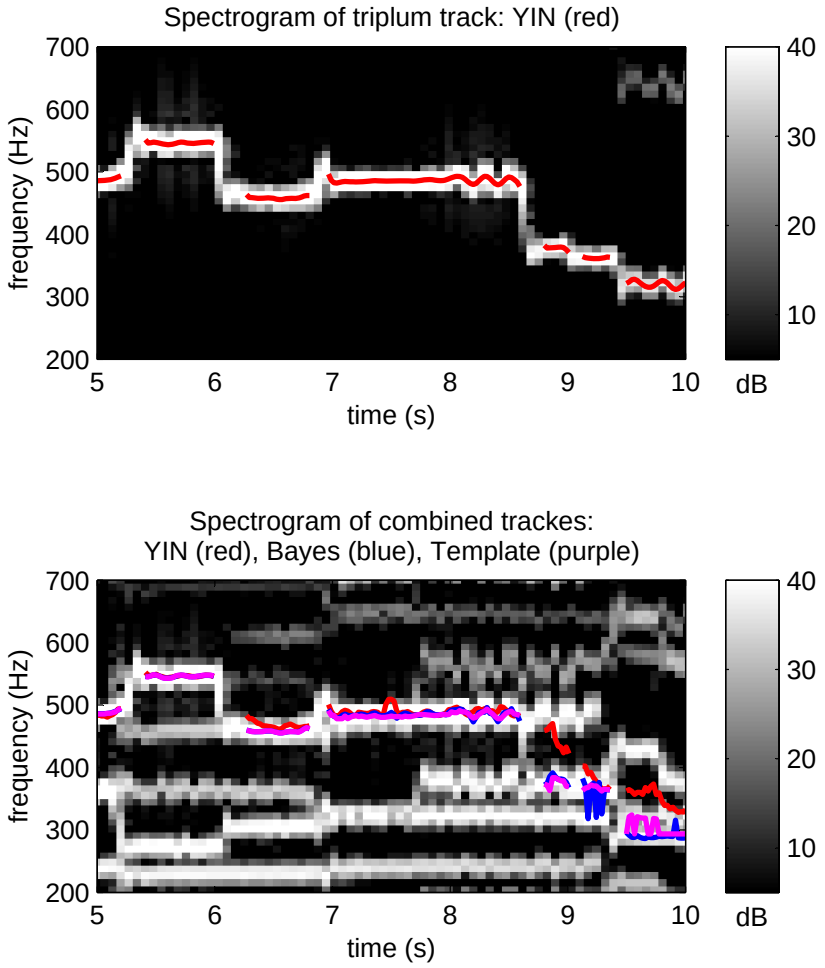

Figure 5.4: Frequency estimates against a spectrogram of a 5 second excerpt. The top plot shows the spectrogram of a the triplum line with the YIN truth data. The bottom plot shows the spectrogram of the full mix along with the YIN, Bayes, and template estimates.

5.6 Discussion

In table 5.1, the template estimator actually has a slightly lower root mean square error (RMSE) than the Bayes estimator. This is simply because the Bayesian estimator has the freedom to converge on frequencies far away from the nominal note frequency during optimization. These really far off estimates do not happen very often, but they increase the mean error. Fortunately, these extreme outliers are easy to identify and ignore. In these rare cases, either the window could be completely ignored or some other estimate, such as the nominal note frequency, could be substituted.

Otherwise, the clear winner here is the Bayesian estimator. This is not terribly surprising since it incorporates a lot of information into its estimates including local noise, harmonic phase, and amplitude. In comparison, the template matching algorithm is very crude and so the fact that the template matching algorithm works less well is not unexpected.

Having said that, the template estimator has some major points in its favor. First of all, it is an order of magnitude faster to run than the Bayesian estimator. What takes a day with the Bayesian estimator takes an hour with the template estimator, making the template estimator much more practical for larger data sets. Secondly, the Bayesian estimator does not scale well to multiple harmonics. Scaling up to three harmonics from two makes the optimization routine much less likely to converge. The template algorithm, however, takes more or less the same time whether it uses 1 harmonic or 12. So if higher harmonics are expected to be of use, the template algorithm can easily take advantage of these harmonics. For example, if a singer is performing with orchestral accompaniment, then using harmonics that fall in the singer's formant, a well known peak in the voice at around 3 kHz [46], may improve estimation.

So the decision to use the Bayesian estimator or template estimator really depends on the task. The Machaut recording used to test the estimators is actually quite difficult. Even though there are only four parts, these four parts are more or less equally loud and the parts are relatively close together. This means that that interfering harmonics are a very real problem because they are nearby and they are prominent. Furthermore, the voices are all about a quarter step flat relative to the A440 tuning standard, which makes it difficult for the estimators to distinguish between the correct but flat note and a nearby note's harmonic that looks like it could be the correct note, but sharp. In this case, using the Bayesian estimator may make sense because of its generally higher accuracy.

Some data sets, however, are a great deal easier. If the task is to pull out a soloist against background accompaniment, the solo voice will tend to be much more prominent and louder than the accompaniment. In this case, the template estimator is probably a better choice because it will do a reasonable job of finding the soloist and it will be much more efficient than the Bayesian estimator. This is indeed the case for a Gilbert and Sullivan data set used in chapters 8 and 9, where the estimator is needed to pull out a soprano soloist. So for this data set, the template estimator will be used.

Ideally, it would be nice to compare these two algorithms to prior work. Unfortunately, this is difficult to do because there aren't really any algorithms that have tried to do guided frequency estimation before. A somewhat similar task is the Music Information Retrieval Evaluation eXchange (MIREX) 2010 audio melody extraction task[3]. Here, the task is to report the melodic pitch in every frame that contains melody. One statistic reported is the percentage of frames in which the

[3]The task is described at `http://www.music-ir.org/mirex/wiki/2010:Audio_Melody_Extraction`. The results are reported at `http://www.music-ir.org/mirex/wiki/2010:MIREX2010_Results`

reported melodic pitch is within 50 cents of the truth data. The five algorithms tested ranged from 61.8% to 89.6% accurate, depending on the data set and the algorithm. These are somewhat below the 91.4% (Bayes) and 92.6% (template) reported in table 5.1, but then then the comparison is not completely straightforward. Apart from the fact the the MIREX competetion used completely different data, the algorithms described in this thesis use score information, whereas the melody extraction algorithms obviously have to pick a note frequency without this guidance.

5.7 Summary

Music recordings offer the promise of large, environmentally valid data sets. Since Western music is almost entirely polyphonic, the best approaches to using music recordings have to be willing to deal with polyphony. This chapter addresses the problem of exact frequency estimation in a polyphonic context. Unconstrained polyphony is difficult to deal with, so this chapter takes the approach of using aligned electronic scores to provide good information on the approximate pitch being performed.

Two different algorithms are presented which hone in on the exact pitch in a frame using the note frequency as a guide. The first approach is Bayesian. It finds the best frequency by maximizing a probability distribution whose prior is a function of the note frequency. The second approach finds the best frequency by finding the best match to a frequency template. In both cases, multiple harmonics are used in order to improve the pitch estimate.

Both algorithms have their strengths. The Bayesian algorithm is somewhat more accurate, but also much more time consuming. The template algorithm is much simpler and faster, but not as accurate. For difficult cases, the Bayesian approach's higher accuracy may be important. In less difficult cases, where the voice whose

frequency is being estimated is quite prominent, the speed of the template algorithm may be more important. For the remainder of this thesis, the template algorithm is used because it provides sufficient accuracy for the data sets used and much greater speed.

Chapter 6

Initial harmonic amplitude estimation

In the previous chapter, the best frequency estimates for a short window in a note was estimated based on the note's nominal pitch using two approaches. The Bayesian approach actually estimated the harmonic strengths as well as the fundamental frequency. Unfortunately, this approach is very time consuming and does not scale well to be used with many harmonics. If most of the interesting information is in harmonics under about 4000 Hz, this translates to about 9 harmonics for A4, which would simply require far too many iterations to converge. The template approach, in contrast, is much faster, but only estimates the fundamental frequency and does not even attempt to estimate the harmonic strengths.

So, some new technique is needed to estimate these harmonic strengths. The amplitude estimates promise to be more difficult than the frequency estimates because there is less information to work with. The frequency estimates rely on multiple harmonics to inform the estimate of the fundamental frequency. So the relatively short 23 ms windows and their attendant poor frequency resolution in the Fourier

domain are less of a problem than they might be otherwise. In contrast, there is nothing in particular linking the amplitude estimate of one harmonic to the amplitude estimate of another harmonic and so they must all be estimated separately.

The rest of this chapter is organized as follows. Sections 6.1 to 6.3 cover three different methods for estimating the harmonic amplitudes, section 6.4 covers experiments and results, section 6.5 discusses the results of the experiments, and section 6.6 summarizes the chapter.

6.1 Energy estimation

This method estimates the magnitude of a harmonic from its energy. Consider a simple sinusoid,

$$z(t) = A\cos\left(2\pi ft + \phi\right), \tag{6.1}$$

where A is the amplitude, f is the frequency in Hz, and ϕ, the phase, is pulled uniformly at random from $(-\pi, \pi]$. The average energy of $z(t)$ is $\frac{A^2}{2}$:

$$E_\phi\left(z(t)^2\right) = A^2 E_\phi\left(\cos(4\pi ft + 2\phi) + \frac{1}{2}\right) \tag{6.2}$$

$$= \frac{A^2}{2} + A^2 E_\phi\left(cos(4\pi ft + 2\phi)\right) \tag{6.3}$$

$$= \frac{A^2}{2}. \tag{6.4}$$

If $z(t)$ is discretized and windowed with a rectangular window of length N, then

$$\frac{1}{N}\sum_n z[n]^2 \approx \frac{A^2}{2}, \tag{6.5}$$

as long as the window is sufficiently long and the sampling rate is sufficiently high.

Recall from section 5.1 that the model for a harmonic signal is

$$x[n] = \sum_i h_i[n] \tag{6.6}$$

where each harmonic is modeled as a sinusoid with frequency ωi in radians per second for harmonic i,

$$h_i[n] = \begin{cases} A_i \cos(\omega \cdot i \cdot n + \phi_i) & -\frac{N}{2} + 1 \leq n < \frac{N}{2} - 1 \\ 0 & n = \frac{N}{2}, \end{cases} \tag{6.7}$$

So, the energy in a single harmonic in a rectangular window of length N is approximately

$$\sum_n h_i[n]^2 \approx N\frac{A_i^2}{2} \tag{6.8}$$

and, using Parseval's equality,

$$\sum_k |H_i[k]|^2 \approx N^2\frac{A_i^2}{2} \tag{6.9}$$

where $H_i[k]$ is the DFT of $h_i[n]$. So, one approximation for A_i is

$$A_i \approx \frac{1}{N}\sqrt{2\sum_k |H_i[k]|^2}. \tag{6.10}$$

Except, of course, that the harmonics are added together so $|H_i[k]|$ is not directly available. But recall from equation (5.7) that the DFT of $h_i[n]$ is essentially the sum

of two periodic sinc functions weighted by A_i,

$$H_i[k] = \frac{A_i}{2}\left(e^{j\phi_i}F_{\frac{N}{2}-1}\left(\frac{2\pi k}{N} - \omega i\right) + e^{-j\phi_i}F_{\frac{N}{2}-1}\left(\frac{2\pi k}{N} + \omega i\right)\right), \tag{6.11}$$

where

$$F_M(\theta) = 2\cos\left(\theta \cdot \frac{M}{2}\right) sinc_{M+1}\left(\frac{\theta}{2}\right) - 1. \tag{6.12}$$

Notice that most of the energy in equation (6.11) is centered around $\frac{2\pi k}{N} \approx \omega i$ and $\frac{2\pi k}{N} \approx -\omega i$. Sticking with just the sinc function in the positive frequencies, for $k_{h,i} = \{k | k \approx \frac{\omega i N}{2\pi}\}$,

$$\sum_{k \in k_{h,i}} |H_i[k]|^2 \approx \frac{1}{2}\sum_{k} |H_i[k]|^2 \approx N^2\frac{A_i^2}{4}. \tag{6.13}$$

So,

$$A_i \approx \frac{1}{2N}\sqrt{\sum_{k \in k_{h,i}} |H_i[k]|^2} \approx \frac{1}{2N}\sqrt{\sum_{k \in k_{h,i}} |X[k]|^2}. \tag{6.14}$$

The question is how to pick $k_{h,i}$ to get as much energy as possible from the current harmonic without getting too much energy from neighboring harmonics. One solution is to simply look half the fundamental frequency on either side of the harmonic. For example, if the fundamental frequency is 400 Hz, look at the 200 - 600 Hz range for the first harmonic, the 600 - 1000 Hz range for the second harmonic, etc. In other words,

$$k_{h,i} = \left\{k \,|\, \omega i - \frac{\omega}{2} \leq \frac{2\pi k}{N} \leq \omega i + \frac{\omega}{2}\right\}. \tag{6.15}$$

Therefore, if $Y[k]$ is the DFT of a window of the actual data, the final estimate for each A_i is

$$\hat{A}_i = \frac{1}{2N}\sqrt{\sum_{k \in k_{h,i}} |Y[k]|^2}. \tag{6.16}$$

Now, one obvious problems with equation (6.16) is that it will include a lot of noise energy because the estimate uses the absolute value of the DFT. The best fit estimate in section 6.2 tries to address this issue.

6.2 Best fit estimation

A best fit approach can try to get around some of the limitations in the energy-based approach. Recall that in section 5.1, the harmonic model is windowed using a Hann window, rather than a square window, which yields,

$$X_w[k] = \sum_i \left(\frac{1}{2} H_i[k] + \frac{1}{4} H_i[k-1] \ + \frac{1}{4} H_i[k+1] \right). \tag{6.17}$$

This time, again look at the positive frequencies $k_{h,i} = \{k | k \approx \frac{\omega i N}{2\pi}\}$. Once again, note that only one of the two sinc functions in equation (6.11) has much energy here, so

$$H_i[k \in k_{h,i}] \approx \frac{A_i}{2} e^{j\phi_i} F_{\frac{N}{2}-1}\left(\frac{2\pi k}{N} - \omega i \right) \tag{6.18}$$

and taking the absolute value yields

$$|H_i[k \in k_{h,i}]| \approx \frac{A_i}{2} \left| F_{\frac{N}{2}-1}\left(\frac{2\pi k}{N} - \omega i \right) \right|. \tag{6.19}$$

Since the energy in the other harmonics is also nearly zero around the current harmonic,

$$|X_w[k \in k_{h,i}]| \approx \frac{A_i}{2}\left|\left(\frac{1}{2}F_{\frac{N}{2}-1}\left(\frac{2\pi k}{N}-\omega i\right)+\frac{1}{4}F_{\frac{N}{2}-1}\left(\frac{2\pi(k-1)}{N}-\omega i\right)\right.\right. \\ \left.\left.+\frac{1}{4}F_{\frac{N}{2}-1}\left(\frac{2\pi(k+1)}{N}-\omega i\right)\right)\right|, \tag{6.20}$$

combining equations (6.17) and (6.19). A_i is now simply a multiplier and everything else can be calculated from the known frequency estimate, ω, for the window. If $Y_w[k]$ is the DFT of a window from the actual recording, windowed using a Hann window, then

$$A_i \approx \frac{|Y_w[k \in k_{h,i}]|}{|X_n[k \in k_{h,i}]|} \tag{6.21}$$

where $|X_n[k]|$ is simply equation (6.20) without A_i.

Now, A_i could be estimated from the single k closest to the harmonic, but averaging usually gives a better estimate,

$$\hat{A}_i = \frac{1}{C}\sum_{k \in k_{h,i}} \frac{|Y_w[k]|}{|X_n[k]|}, \tag{6.22}$$

where C is the number of indexes in $k_{h,i}$. But since the magnitude of the DFT drops off reasonably quickly around the harmonic frequency, a weighted mean makes more sense,

$$\hat{A}_i = \frac{1}{\sum_{k \in k_{h,i}} w[k]} \sum_{k_{h,i}} \left(\frac{|Y_w[k]|}{|X_n[k]|} \cdot w[k]\right) \tag{6.23}$$

where the weights $w[k]$ are proportional to the expected strength of the signal at k,

$$w[k] = \frac{|X_n[k]|}{\sum_{k_{h,i}} |X_n[k]|}. \tag{6.24}$$

Once again, the question is how to pick $k_{h,i}$. The weights are nearly zero more than a couple of indexes away from the harmonic frequency, $\frac{\omega i N}{2\pi}$, so picking the three ks closest to the harmonic frequency allows the estimate to use the most energetic components of the DFT.

This best fit estimate should have less of a noise problem that the energy estimate. The three largest DFT values near the harmonic frequency should have the highest signal to noise ratios (SNRs) and therefore the noise should be less of a problem.

6.3 Heterodyne solution

Taking a completely different approach based on the time domain signal, recall once more that the harmonic model is essentially a simple sinusoid over a short window, but for the entire time span of the note, the frequency, $\omega[n]$, is a function of time and so

$$h_i[n] = A_i[n] \cos\left(\sum_{k=0}^{n} (\omega[k]i) + \phi_i\right). \tag{6.25}$$

Forming the analytic signal for $h_i[n]$ yields

$$h_{a,i}[n] = h_i[n] + j\hat{h}_i[n] = A_i e^{j\phi_i} exp\left\{j \sum_{k=0}^{n} (\omega[k]i)\right\}, \tag{6.26}$$

where $\hat{h}_i[n]$ is the Hilbert transform of $h_i[n]$. To demodulate,

$$h_{d,i}[n] = h_{a,i}[n] \cdot exp\left\{-j \sum_{k=0}^{n} (-\omega[k]i)\right\} = A_i e^{j\phi_i}. \tag{6.27}$$

Since the amplitudes are strictly positive,

$$A_i = |h_{d,i}[n]|. \tag{6.28}$$

Now, the actual signal is made up of the sum of many harmonics,

$$x[n] = \sum_i h_i[n] \tag{6.29}$$

and so the analytic signal will be

$$x_a[n] = x[n] + j\hat{x}[n] = \sum_i \left(A_i e^{j\phi_i} exp\left\{ j \sum_{k=0}^{n} (\omega[k]i) \right\} \right). \tag{6.30}$$

If the goal is to estimate the amplitude of harmonic ℓ, $A_\ell[n]$, then

$$\begin{aligned} x_{d,\ell}[n] &= x_a[n] \cdot exp\left\{ -j \sum_{k=0}^{n} (\omega[k]\ell) \right\} \\ &= A_\ell e^{j\phi\ell} + \sum_{i \neq \ell} \left(A_i e^{j\phi_i} exp\left\{ j \left(\sum_{k=0}^{n} (\omega[k]i - \omega[k]\ell) \right) \right\} \right). \end{aligned} \tag{6.31}$$

Note that only the desired harmonic, $h_\ell[n]$ has been moved to base-band. So a simple low pass filter will isolate $A_\ell[n]$ and the final estimate will be

$$A_\ell[n] \approx |lowpass(y_{d,\ell}[n])|, \tag{6.32}$$

where $y_{d,\ell}[n]$ is the actual signal demodulated for harmonic ℓ.

There is one small problem with equation (6.32), namely that it requires knowledge of the fundamental frequency, $\omega[n]$. Recall that the frequency estimates from section 5.4 are made on a per-window basis. Since the frequency does not change much from window to window, $\omega[n]$ can be reasonably estimated by interpolating between window values using a cubic spline for smoothness.

6.4 Experiments and results

Getting truth data for amplitudes from actual recordings would be difficult, so the experiments are all performed on simulated data with added white Gaussian noise.

For both tests, a single five-second note is synthesized with a 44.1 kHz sampling rate. Since the amplitude estimates are calculated on a per-window (energy, best-fit) or per element (heterodyne) basis, the estimates are essentially independent of each other across time and so one long note is equivalent to many shorter notes.

In order to have at least a little frequency modulation, the frequency track for the note is a ramp around the nominal frequency, A4 = 440 Hz. Starting from a half step below A4, 415 Hz, the frequency increase linearly to the same distance above A4, 465 Hz (figure 6.1).

The note is synthesized with eight harmonics and the amplitude of each harmonic is piece-wise linear, with the junctions at the second marks (figure 6.1). The magnitudes of the junction points are chosen uniformly at random between 0.5 and 1.

The window size is set to 1024 elements (23 ms) and the distance between consecutive windows is set to 512 elements (12 ms) for the energy and best fit estimations. The heterodyne algorithm works on the entire time domain signal rather than on individual windows, but needs fundamental frequency estimates, which are provided on a per-window basis. The heterodyne algorithm also requires a low-pass filter. For this experiment, a linear phase filter with 601 taps (14 ms) and a cutoff of fifth the minimum voice pitch is used.

To make comparisons with the other estimates easier, the heterodyne amplitude estimates are only examined in the middle of each window. Five seconds of 23-ms windows with 8 amplitudes and 12 ms between estimates works out to about 3400 amplitude estimates for evaluation.

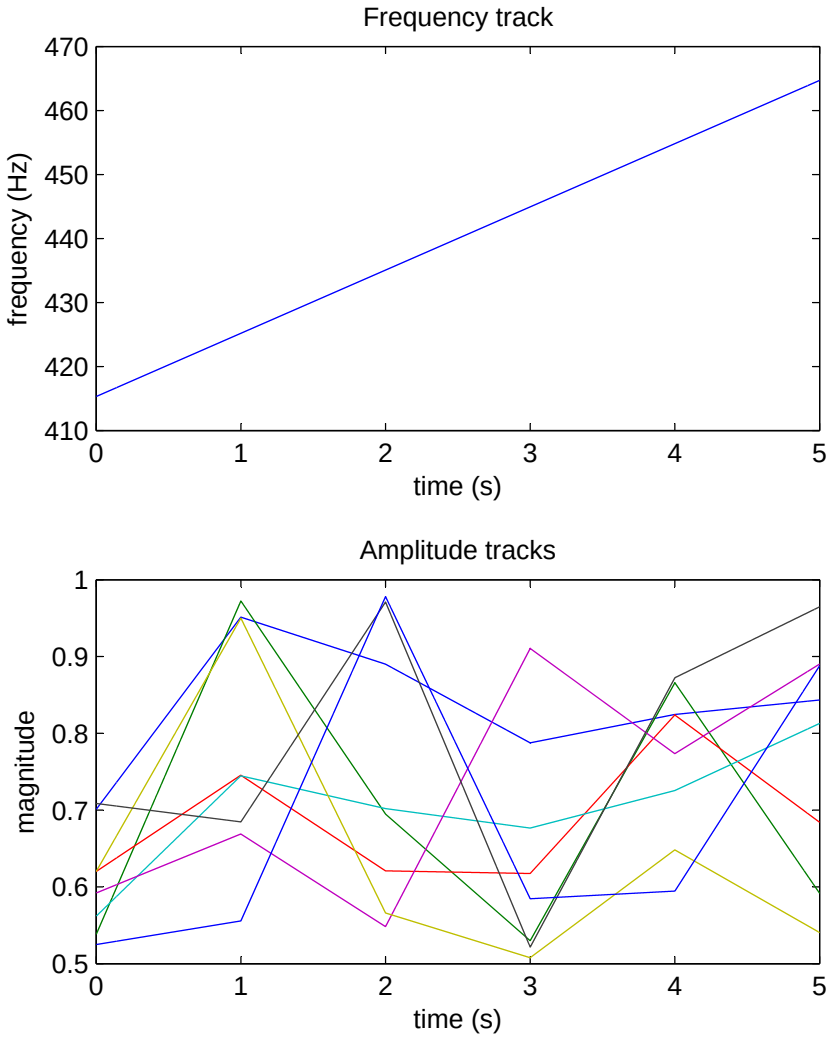

Figure 6.1: Note frequency and harmonic amplitudes. The top plot shows the note frequency ramp and the bottom plot shows the piece-wise linear harmonic amplitudes.

6.4.1 True frequency experiment

In the case of real data, the fundamental frequency would first need to be estimated, but for this initial test, all the algorithms are given the true fundamental frequencies for the middle of each window. Figure 6.2 shows the percent error averaged over

all windows and all harmonics at different SNRs. There is essentially no difference between the three methods, which have about 15% error at an SNR=-3 dB and about 7% error at SNR=3 dB. Although the energy estimate appears to perform a little better than the other methods, the difference is well within a single standard deviation of the means of the errors. In the absense of noise (right-most points), the heterodyne and best-fit solutions have very little error. The energy solution is off by about 1.5%, which may be at least partly due to the fact that it consistently underestimates the energy for each harmonic since it only calculates energy over a limited frequency range.

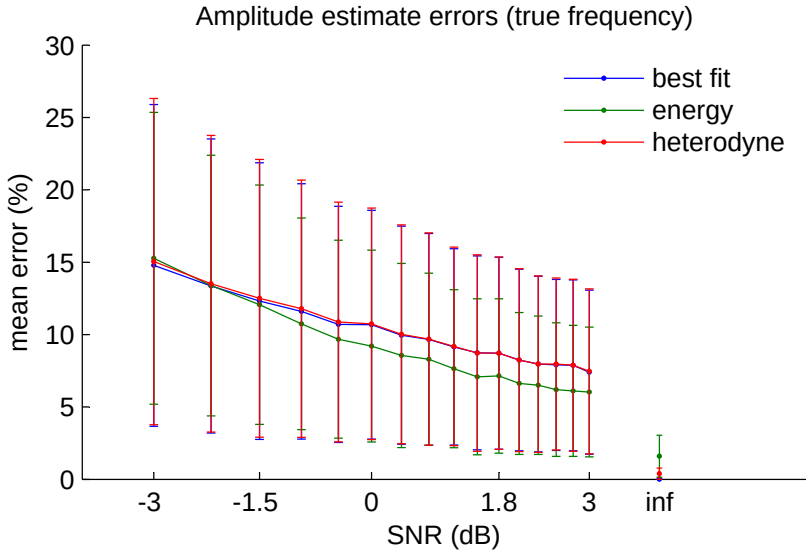

Figure 6.2: Amplitude estimation results using the true frequency. Each line represents the mean, with the vertical bars indicating one standard deviation. Results are reported as a percentage of the average magnitude of the harmonic amplitudes. The 'inf' or infinite SNR case is the test signal without noise.

6.4.2 Estimated frequency experiment

In this experiment, the fundamental frequency is estimated in each window using the template matching solution described in section 5.4 so the experiment is more realistic. For this frequency estimation, the window size is 1024 element (23 ms at 44.1 kHz sampling rate), the hop size between windows is 512 (12 ms), the first three harmonics are used, and 100 linearly spaced frequencies tested between two half steps below the nominal note frequency and the same number of Hz above. Figure 6.3 shows that on the whole, the frequency estimates are very good. The frequency errors in the no-noise case (right most point) are almost entirely a result of the fact that the template matching algorithm only searches over a fixed number of frequency values. As figure 6.4 shows, the amplitude estimation errors hardly move when the algorithms use these estimated frequencies rather than the true frequencies, so the small frequency errors are unimportant.

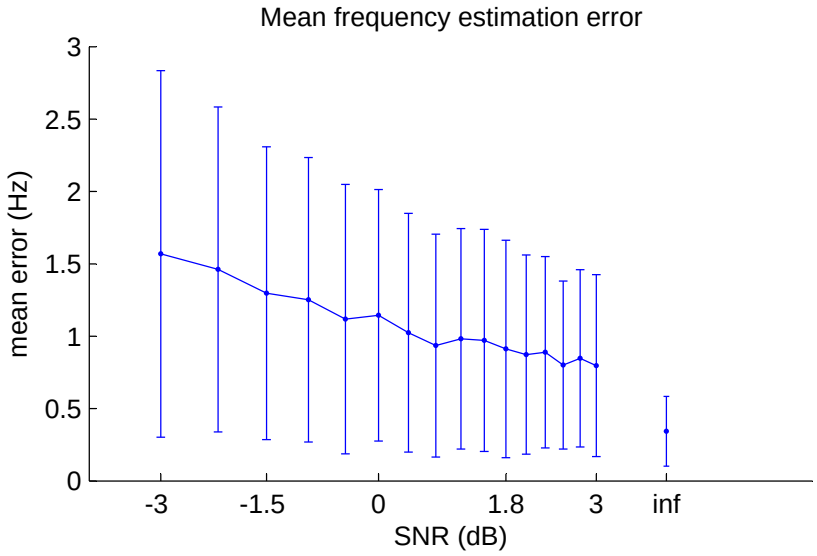

Figure 6.3: Frequency estimation error.

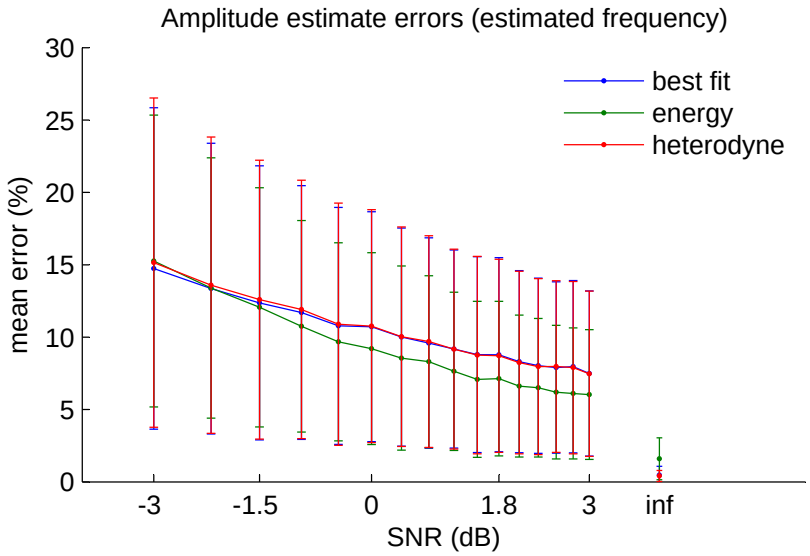

Figure 6.4: Amplitude estimation results using the estimated frequency. Once again, each line represents the mean, with the vertical bars indicating one standard deviation. Results are reported as a percentage of the average magnitude of the harmonic amplitudes. The 'inf' or infinite SNR case is the test signal without noise.

6.5 Discussion

Although the frequency errors are at worst about 0.3% (1.5 Hz/440 Hz), the worst amplitude errors are quite substantial for all three estimates at about 15%. Not only are the mean errors quite large, the spreads around the means are also large. As mentioned in the opening section of this chapter, the noise affects the amplitude estimates more because there is simply less information about the amplitudes in each window than there is about the fundamental frequency. Of course, it is reasonable to suspect that amplitude estimates across time should be quite slowly varying. Time smoothing to improve amplitude estimation is addressed later in chapter 8.

For now, sticking with estimates that are not smoothed across time, all three estimates presented here seem to perform very similarly. Computationally, however, the best fit estimate requires much less overhead. In equation (6.21), $Y[k]$ has

already been calculated in order to estimate the fundamental frequency. In contrast, the energy estimate works on the DFT of a rectangular-windowed signal and the heterodyne estimate requires that each harmonic's base-band signal be filtered.

6.6 Summary

The Bayesian algorithm from the last chapter simultaneously estimates fundamental frequency and harmonic amplitudes. Unfortunately, this algorithm converges too slowly to be useful for large data sets. Instead, the template matching algorithm from the last chapter is used to estimate the fundamental frequency, meaning that the harmonic amplitudes must be estimated separately. This chapter addresses the question of finding the harmonic amplitudes using the template fundamental frequency estimates as a guide.

Three approaches are taken to estimating the harmonic amplitudes, all of which assume that harmonics are mostly isolated in frequency. The energy-based estimate assumes that all the energy in the DFT it finds near a harmonic frequency can be attributed to that harmonic. The best fit estimate picks amplitudes A_i so that equation (6.20) fits to the absolute value of the DFT near the harmonic amplitude. Finally, the heterodyne estimate moves the harmonic to baseband, filters out all the other harmonics, and uses the absolute value of the result as the amplitude estimate.

All three approaches struggle similarly with noise, which will be addressed in chapter 8, but computationally, the best fit algorithm requires the fewest calculations. So this is the estimate that will be used in future chapters.

Chapter 7

Discarding bad data

Previous chapters have covered algorithms for finding the fundamental frequency of a note given a nominal note frequency and for finding harmonic amplitudes given the fundamental frequency. The problem is that these algorithms do not always work. If the frequency estimate is off, then the amplitude estimates will just be noise. So making sure that the fundamental frequency has been correctly estimated is important.

The alignment of the notes to the midi does not always produce the ideal alignment, even if done by hand. Recall from section 2.4 that the Gilbert and Sullivan data set labels a single boundary between notes, i.e. one note's offset is the next note's onset, unless there is a definite break between notes, in which case a separate note offset is notated. This work is really interested in the sustained vowels in words, however, rather than the entire word, consonants and all. Since there is only one boundary annotated between notes, some of the time allocated to a note will invariably include the consonants at the beginning or end of the syllable.

Furthermore, some notes are just too quiet in comparison to the orchestra to be found regardless of how well the score is aligned. So even when tracing a solo, which

should stand out from the orchestra, some notes will invariably be lost.

Figure 7.1 shows the spectrograms and frequency track for a poorly captured note. The top plot shows that the frequency track jumps around and clearly has not managed to fix on the soloist. In the bottom plot, the spectrogram shows that the note harmonics are sometimes obscured by the more powerful orchestra. This note was relatively low and short, which perhaps accounts for the difficulty. In contrast, figure 7.2 shows the same information for a well-captured note. The middle section of the note (green) is smooth and shows the vibrato very nicely, while the beginning and end of the note have poor frequency tracks. Interestingly, the word being sung is 'lot.' The 't' at the end is obviously unvoiced and so there is no solo frequency track to find. The 'l' however, is voiced, but is not nearly as loud as an open vowel.

These examples illustrate a general principal, that frequency tracks should be smooth and have regular vibrato. This principal extends to the entire data set that these notes were taken from (section 2.4). The rest of this chapter will explore how to calculate features for smoothness (section 7.1) and vibrato (section 7.2), and how to classify the data based on these features (section 7.3). Experiments performed are presented (section 7.4), a discussion of the results is given (section 7.5), and finally a brief summary is given (section 4.5).

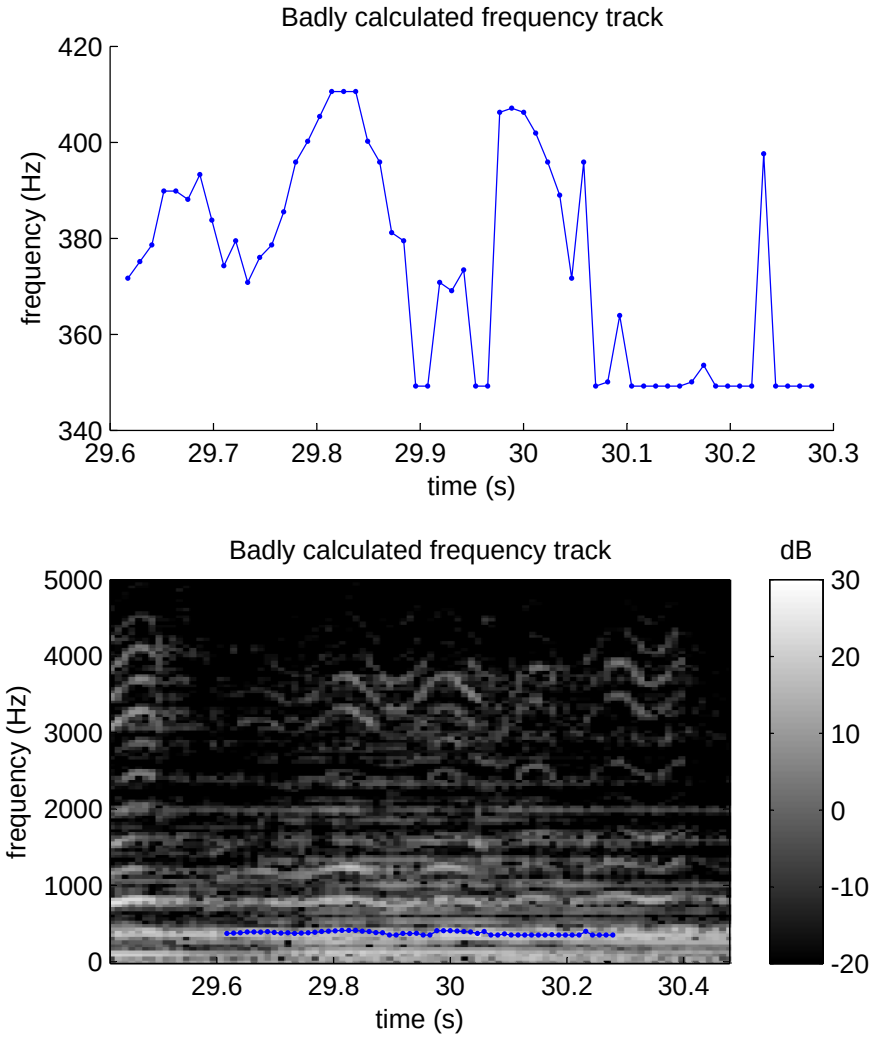

Figure 7.1: Sample of a bad note frequency track. The top plot shows just the frequency track in isolation. The frequency jumps around over a wide range of frequencies and shows no evidence of vibrato. There are also a substantial number of frequency estimates at the bottom edge of the search range ($\approx$ 350 Hz), indicating that the algorithm could not find anything substantial closer to the nominal note frequency. The bottom plot shows the same frequency track against the spectrogram it was calculated from. The note is partly obscured by the orchestra, particularly under 3000 Hz. For both plots, the window size is 1024 elements ($\sim$ 23 ms) and the hop size between windows is 512 elements ($\sim$ 12 ms).

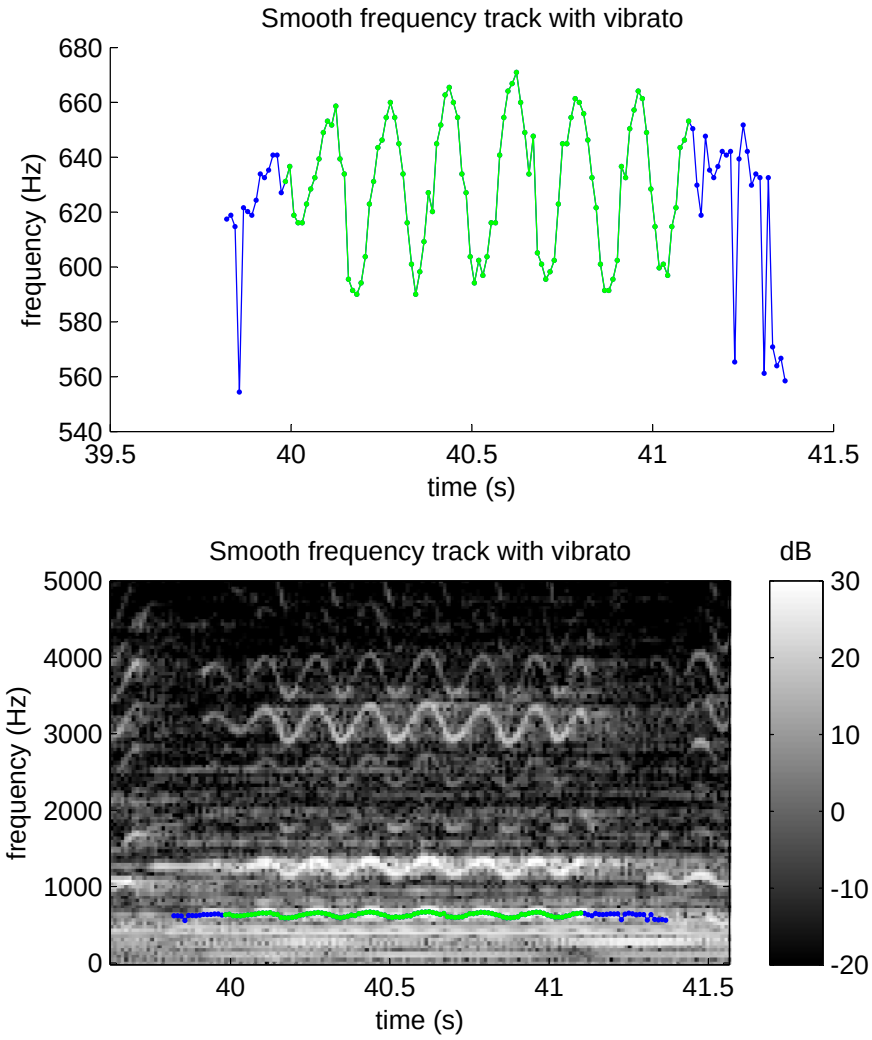

Figure 7.2: Sample of a good note frequency track. The top plot shows the frequency track. The green portion is the part that has been labeled as *good*. As can be seen, the frequency track is smooth and shows definite evidence of vibrato. The bottom plot shows the frequency track against its associated spectrogram. Throughout the *good* portion of the note, the harmonics are clearly visible. For both plots, the window size is 1024 elements ($\sim$ 23 ms) and the hop size between windows is 512 elements ($\sim$ 12 ms).

7.1 Measuring pitch track smoothness

The smoothness feature should be very local in scope to catch quickly varying values. A simple three point ratio can show how the current value compares to its neighbors:

$$s[n] = \frac{2 \cdot f[n]}{f[n-1] + f[n+1]} \tag{7.1}$$

where s is the local smoothness, n is the time index, and f is the measured frequency. This measure, along with the vibrato measure in the next section, will be used in section 7.3 to classify portions of pitch tracks as *good* or *bad*.

7.2 Measuring vibrato

Measuring the vibrato in a note is considerably more complicated that measuring pitch track smoothness because the model needs to capture longer term trends.

7.2.1 The vibrato model

To simplify the model, the frequency tracks for the notes are first normalized so that they are centered around zero,

$$f_n[n] = f[n] - middle(f[n]) \tag{7.2}$$

where $f_n[n]$ is the normalized frequency track, $f[n]$ is the original frequency track, and $middle(f[n])$ is the middle frequency of f. This middle frequency is essentially the mean of f with a few points removed. Recall from section 5.4 that $f[n]$ is the best frequency at time n out of a range of frequencies near the nominal note frequency. When the pitch information in a particular window is very weak, the algorithm often ends up selecting an extreme frequency at the edge of the frequency range

(figure 7.1). These extreme frequencies can really throw off the mean, so they are removed first. Let $R = \{n \mid f[n] \text{ is not an extreme value}\}$. Then,

$$middle(f[n]) = \frac{1}{|R|} \sum_{n \in R} f[n]. \tag{7.3}$$

The simplest model for vibrato in this normalized frequency track is a sinusoid,

$$v_s[n] = A_s \cos\left(\frac{2\pi f_{s,v}}{r} n + \phi_s\right), \tag{7.4}$$

where $v_s[n]$ is the voice pitch frequency in Hz at time index n in windows[1], A_s is the amplitude of the vibrato, $f_{s,v}$ is the frequency of the vibrato in Hz, r is the sampling rate of the raw frequency estimates in Hz, and ϕ_s is the phase of the vibrato. Unfortunately, real vibratos do not fit this simple model very well. In practice, the frequency of the vibrato modulates slowly, so a better model would be

$$v_{FM}[n] = A_s \cos\left(\frac{2\pi}{r} \sum_{k=0}^{n} f_v[k] + \phi\right), \tag{7.5}$$

where f_v, the instantaneous frequency, is a slowly varying vibrato frequency and ϕ is the initial phase. The amplitude of the vibrato also tends to change across the note, so a reasonable model of real world vibrato is

$$v_{AM}[n] = A[n] \cos\left(\frac{2\pi}{r} \sum_{k=0}^{n} f_v[k] + \phi\right), \tag{7.6}$$

where $A[n]$ modulates the amplitude of the vibrato.

Estimating all the parameters of equation (7.6) at once would be difficult, particularly in light of the 'slowly varying' constraint for $f_v[k]$. It is much easier to fit

[1]Recall that each pitch track estimate is made over a short window of the recording. As explained later in section 7.4, the windows for experiments in this chapter are 1024 samples (23 ms at 44.1 kHz sampling rate) and the hop between windows is 512 samples (12 ms).

parameters for each model and then improve the fit with the next model. Figure 7.3 shows a single note fitted to each of these models. The top plot shows the simple sinusoid model (section 7.2.2), which fails to capture many of the details of the vibrato. The middle plot shows the same model after frequency modulation (section 7.2.3). Here the peaks and troughs of the vibrato model line up much better with the peaks and troughs of the raw frequency track. Finally, the amplitude modulation (section 7.2.4) in the bottom plot produces a model that fits the vibrato sections of the note reasonably well.

7.2.2 Estimating the vibrato parameters for the simple vibrato model

For equation (7.4), the single vibrato frequency $f_{s,v}$, single amplitude A_s, and phase ϕ_s need to be estimated. Let $F_n[k]$ be the DFT of the normalized frequency track, $f_n[n]$. The estimate for $f_{s,v}$ is then simply calculated from the peak value of $F_n[k]$ in the 4 - 7 Hz range.

$$\hat{f}_{s,v} = \frac{k_{max}}{r} \tag{7.7}$$

where

$$k_{max} = \{k \mid F_n[k] = \max\left(F_n[k_{vibrato}]\right)\} \tag{7.8}$$

where $k_{vibrato}$ is simply the set of indexes of F_n in the 4 - 7 Hz range. Obviously, the frequency resolution of $F_n[k]$ can be quite low if the pitch track is short, meaning that $\hat{f}_{s,v}$ may not be very accurate. Recall, however, that the vibrato frequency will be adjusted for the frequency modulation (FM) model in equation (7.5). As detailed below in section 7.2.3, the FM vibrato frequency will be permitted to take on the full range of frequency values, not just the quantized frequencies used in this estimate.

Once $\hat{f}_{s,v}$ is known, the phase can simply be estimated from this peak as

$$\hat{\phi}_s = \angle F_n[k_{max}]. \tag{7.9}$$

To estimate A_s, the the DFT of $v_s[n]$ (equation (7.4)) is needed. Note that if $A_s = 1$, then the DFT of $v_s[n]$ is

$$V_{s,norm}[k] = \frac{1}{2}e^{j\phi_s}Z_N\left(\frac{2\pi f_{s,v}}{r} - \frac{2\pi k}{N}\right) + \frac{1}{2}e^{-j\phi_s}Z_N\left(-\frac{2\pi f_{s,v}}{r} - \frac{2\pi k}{N}\right) \tag{7.10}$$

where

$$Z_N(\omega) = e^{j\frac{\omega(N-1)}{2}}sinc_N\left(\frac{\omega}{2}\right) \tag{7.11}$$

and

$$sinc_N(\gamma) = \begin{cases} \frac{\sin(\gamma N)}{\sin(\gamma)} & \text{if } \gamma \neq \pi, 2\pi, \dots \\ N & \text{otherwise} \end{cases}. \tag{7.12}$$

Since A_s is simply a constant multiplier in equation (7.4), the general equation for the DFT of $v_s[n]$ is

$$DFT(v_s[n]) = V_s[k] = A_s \cdot V_{s,norm}[k] \tag{7.13}$$

Once $\hat{f}_{s,v}$ and $\hat{\phi}_s$ have been calculated, the ratio

$$ratio[k] = \frac{|F_n[k]|}{|V_{s,norm}[k]|} \tag{7.14}$$

will equal A_s if the data, $F_n[k]$, perfectly fits the model, $V_s[k]$. Because the data will not in general fit the model, the estimate $\hat{A}_s$ can be calculated with a weighted

average

$$\hat{A}_s = \frac{1}{\sum_{k_{est}} w[k]} \left(\sum_{k_{est}} ratio[k_{est}] \cdot w[k_{est}] \right) \tag{7.15}$$

where $k_{est} = \{k_{max} - 1, k_{max}, k_{max} + 1\}$ and the weights are calculated based on the magnitude of the theoretical signal

$$w[k] = \frac{|V_{s,norm}[k]|^2}{\sum_{k_{est}} V_{s,norm}[k]} \tag{7.16}$$

7.2.3 Estimating the frequency modulation

The instantaneous frequency of the vibrato should be slowly varying. So the instantaneous frequency $f_v[n]$ is modeled as a piece-wise continuous linear function where each piece is three wavelengths of $f_{s,v}$, the single approximate frequency found for the simple model (section 7.2.2).

The best $f_v[n]$ should decrease the distance between $f_n[n]$ and $v_{FM}[n]$, but it should not vary far from $f_{s,v}$. Define the distance between $f_n[n]$ and $v_{FM}[n]$ as

$$d = \sum_{k_{dist}} |F_n[k_{dist}] - V_{FM}[k_{dist}]| \tag{7.17}$$

where $F_n[k]$ is the DFT of $f_n[n]$, $V_{FM}[k]$ is the DFT of $v_{FM}[n]$, and k_{dist} is the set of three indexes that are closest $f_{s,v}$. The task then is to minimize equation (7.17) by changing the anchor points or ends of the pieces of $f_v[n]$ and the phase ϕ. An analytic solution to this problem is not clear, but the Nelder-Mead simplex algorithm [31] can iteratively find the solution. The initial estimate for ϕ is ϕ_s and for the frequency anchor points, $f_{v,s}$.

7.2.4 Estimating the amplitude modulation

At this point, the FM-model's peaks and troughs should roughly line up with the peaks and troughs of the frame-based estimates, $f_n[n]$. The problem is that the peaks and troughs may not go high or low enough because the amplitude is fixed to a single value. So, for this model, $A[n]$ takes on a single value per peak/trough so as to minimize the square error between $f_n[n]$ and $v_{AM}[n]$.

Suppose that $y[n] \approx Ax[n]$, where A is picked to reduce the square error:

$$E(A) = \sum_n \left(y[n] - Ax[n]\right)^2 \tag{7.18}$$

Taking the derivative of $E(A)$ with respect to A and setting that derivative equal to zero yields

$$A = \frac{\sum_n y[n]x[n]}{\sum_n x[n]^2}. \tag{7.19}$$

So, if $\{n_z\}$ is the set of indexes associated with a particular peak or trough (i.e. the set of indexes between successive zero-crossings), then

$$A[n_z] = \frac{\sum f_n[n_z] \cdot v_n[n_z]}{\sum v_n[n_z]^2} \tag{7.20}$$

where

$$v_n[n] = \frac{v_{FM}[n]}{A_s} = \cos\left(\frac{2\pi}{r}\sum_{k=0}^{n} f_v[k] + \phi\right) \tag{7.21}$$

will give this least-squares solution.

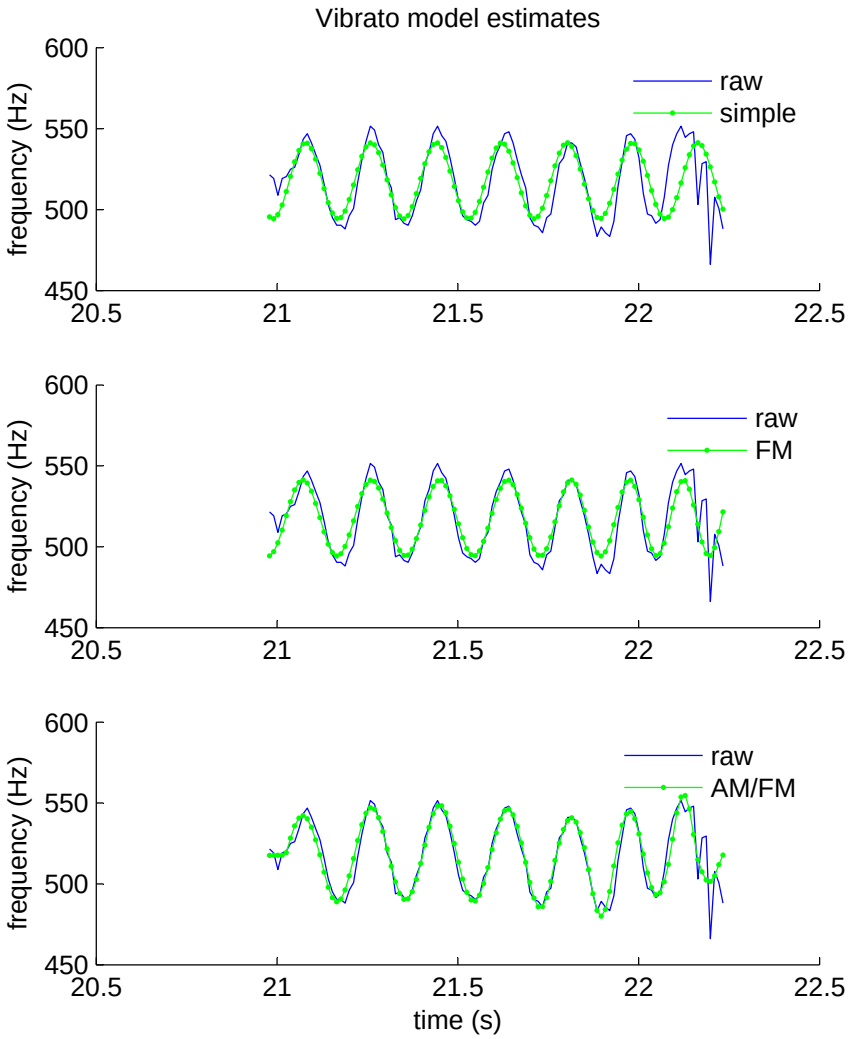

Figure 7.3: Vibrato models example. The top, middle, and bottom plots show the simple (equation (7.4)), frequency modulated (FM) (equation (7.5)), and amplitude modulated (AM/FM) (equation (7.6)) models fit to a note. In each case, the blue line shows the raw frame-based frequency estimate and the green dotted line shows the best model found.

7.3 Classification

The kinds of problems that tend to affect frequency estimates tend to either cause problems over the entire note, as is the case with notes that are too quiet in comparison to the rest of the orchestra, or they tend to cause problems at just the beginning and end of notes, as is the case with consonants. So a typical frequency track for a note with some good data starts with noise, settles into a stable vibrato pattern, and then ends with noise (figure 7.2, top plot). A typical frequency track for a note without any good data is just noise throughout. This can be summarized in a state machine (figure 7.4) where notes with good data start on the left and move right and notes without good data never leave the first state.

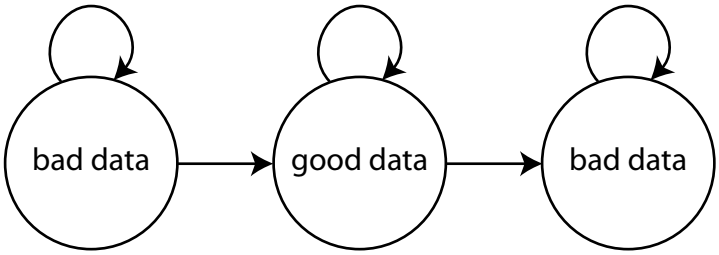

Figure 7.4: Frequency track state machine.

Setting up the problem probabilistically means calculating probabilities for each state given the measured frequency track. Once these probabilities are set up, the optimal solution is the Viterbi path. Since the smoothness and amplitude features have no obvious inherent distributions, distributions were picked that seemed to fit the truth data (section 2.4) well. Note that in all cases, the two 'bad data' states are modeled with the same parameter values.

7.3.1 Calculating smoothness likelihoods

The smoothness feature (equation (7.1)) is modeled as a normal distribution with a mean μ and variance σ for the *good* and *bad* data states. So the likelihood is

$$L_s(s[n]) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(s[n]-\mu)^2}{2\sigma^2}} \tag{7.22}$$

where σ and μ are estimated separately for each state using the sample variance and mean of the truth data.

7.3.2 Calculating vibrato model likelihoods

The vibrato likelihood is calculated over the vibrato magnitude values estimated in equation (7.20). Occasionally, the calculated $A[n]$ is negative or zero when the frequency curve is not a good match for the model. The likelihood of occurrence for these non-positive values is modeled as a Bernoulli distribution where

$$L_{A,B}(A[n] \text{ is in state 'x' } \mid A[n] <= 0) = p_x \tag{7.23}$$

and 'x' is either *good* data or *bad* data and p_x is estimated from the truth data.

Positive-valued amplitudes are modeled as gamma distributions:

$$L_{A,G}(A[n] \mid A[n] > 0) = A[n]^{(k-1)} \frac{e^{\frac{-A[n]}{\theta}}}{\Gamma(k)\theta^k} \tag{7.24}$$

where k and θ are estimated using maximum likelihood estimates from truth data.

In order to calculate the likelihood over all the values in $A[n]$, the two likelihoods must be combined. For state $x \in \{good, bad\}$, this gives

$$L_A(A[n]) = \begin{cases} p_x & \text{if } A[n] <= 0 \\ (1 - p_x)L_{A,G} & \text{if } A[n] > 0 \end{cases} \tag{7.25}$$

7.3.3 Combining likelihoods

The smoothness features and the vibrato magnitude features hopefully provide different kinds of information about the overall likelihood that a particular frequency measurement is *good* or *bad*. For simplicity, assume the features are independent, so

$$L_c(s[n], A[n]) = L_s(s[n]) \cdot L_A(A[n]). \tag{7.26}$$

7.3.4 The Viterbi path

Equations (7.22), (7.25) and (7.26) calculate likelihoods of data in particular states, but the Viterbi path algorithm also requires prior probabilities,

$$Pr(f[0] \text{ is in state } x), \tag{7.27}$$

and transition probabilities,

$$Pr(f[n+1] \text{ is in state } x \mid f[n] \text{ is in state } y). \tag{7.28}$$

These can also be estimated from the truth data. There are three states - $\{bad, good, bad\}$. Notes can either begin in the first state, *bad*, or the second state, *good*. Notes that start with good data start in the second state, all other notes start in the first state. So the prior is zero for the third state. Notes can transition from one state to the

next state or stay in the same state, but they cannot go back to a previous state. So many of the transition probabilities are also zero. The non-zero values are estimated directly from the numbers in the truth data, as described below.

7.4 Experiments and results

There are three defined distributions: one over vibrato magnitude, one over frequency smoothness, and one that combines the two (equations (7.22), (7.25) and (7.26)). The question is which distribution does the best job of distinguishing between the vibrato and non-vibrato sections. To answer this question, the distributions are tested on a subset of the Gilbert and Sullivan data set (section 2.4).

Recall that the Gilbert and Sullivan data set consists of arias whose soprano lines have been hand-aligned with a midi score. For the purposes of this these experiments, the frequency tracks for the notes in 'Sorry her lot' from *H.M.S. Pinafore* were then calculated using the algorithm described in section 5.4 with the window size set to 1024 samples (23 ms at 44.1 kHz sampling rate) and the hop size between windows was set to 512 samples (12 ms). The first three harmonics were used to calculate the frequency estimate. The frequency search region around each nominal note frequency from the MIDI was set to two half steps (i.e. 200 cents) down from the nominal note frequency and an equal number of Hz above the nominal note frequency. A hundred linearly-spaced frequencies in this range were tested for the best fit.

For truth data, all the notes that were at least half a second long were hand labeled for *good* and *bad* data. These 110 notes were labeled by looking at the frequency track and making a somewhat subjective decision about what looked reasonable (i.e. was smooth and had vibrato) and what did not.

With this truth data in hand, the three distributions are tested using 10-fold cross validation. In each of the 10 folds, 11 samples are held out for testing and

	naive	vibrato mag.	smoothness	combined
accuracy (%)	63 (4)	77 (6)	74 (9)	78 (5)
precision (%)	69 (8)	84 (7)	75 (8)	85 (6)
recall (%)	80 (5)	86 (6)	93 (6)	87 (6)

Table 7.1: 10-fold cross validation results. Each column represents a different classification scheme. The accuracy, precision, and recall are calculated for each fold. Each box has the mean across the 10 folds and the standard deviation in parentheses.

99 samples are used to calculate distribution and Viterbi parameters. As a straw man, the results from the three distributions are compared to a 'naive' classifier. Obviously, most of the notes start with bad data, move to good data, and end with bad data. So labeling the first few windows as *bad*, the middle windows as *good*, and the last windows at *bad* will give a pretty good level of classification without even looking at the actual note data. This is essentially what the 'naive' classifier does after first calculating the average proportions of state one (*bad*), state two (*good*), and state three (*bad*) from the truth data.

The test results are all calculated with reference to the desired second state (*good*) over window-level labels. Table 7.1 shows the results for these four classifiers. The naive classifier does surprisingly well with 63% accuracy, but all the probabilistic approaches work better. Furthermore, the approaches that use the vibrato magnitude estimates clearly out-perform the smoothness-only estimate. The combined estimate seems to be a slight improvement on the vibrato magnitude-only estimate.

This cross-fold validation does not tell the entire story. Ideally, the classifier should get as many correct data points as possible (recall) while maintaining a sufficiently high precision. A simply way to change the precision of classifiers is to manipulate the likelihoods before the Viterbi path algorithm is run. If the *good* state's likelihood is multiplied by a number less than 1, the Viterbi algorithm will

tend to be more conservative about picking this state and thus will raise the precision. A multiplier greater than 1 should make the Viterbi algorithm less cautious and thus will raise the recall.

Figure 7.5 shows the results of manipulating the *good* state likelihood, testing and training over all the truth data. As expected, the combined likelihood classifier is consistently better than the other classifiers, although it is not much better than the vibrato magnitude-only classifier. At 90% precision, the combined classifier has about 82% recall to the vibrato magnitude-only classifier's 77%.

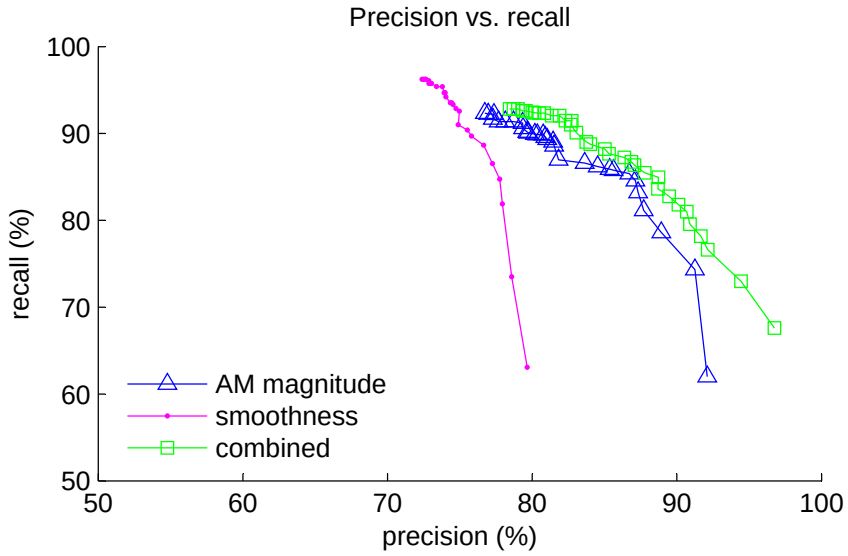

Figure 7.5: Precision vs. recall for good frequency data detection.

7.5 Discussion

Only two features - vibrato magnitude and smoothness - are presented in this chapter. Other features were tried, but did not work as well. As mentioned earlier, the frame-based frequency estimates often pick edge values when there isn't anything to find near the note frequency. A true-or-false feature was tried based on whether or not

the frequency estimate in a window was an edge value. In the end, the number of windows with edge estimates was simply to small to make this feature very useful.

It might seem strange that the vibrato magnitude estimates were used as a feature rather than some kind of fit or distance between the frequency track and the vibrato model. The problem is that the vibrato model could often fit the noise reasonably well, especially if the 'bad' data was near the mean frequency and the vibrato magnitude could be made close to zero. Instead, the magnitudes themselves, when they are reasonably large, are a much better indication that the model has found a genuine vibrato pattern.

Both the vibrato magnitude-only classifier and the combined classifier do a reasonable job of classification. Given the relatively small data set, it is hard to firmly say that the combined classifier is better. It requires more parameters and features. Furthermore, the differences in the means from the 10-fold cross validation test are smaller than the standard deviations. Most of the strength of the combined classifier clearly comes just from the amplitude estimates.

The truth data also has limitations. It was labeled by making a subjective decision about whether the frequency tracks made sense. Figures 7.1 and 7.2 show two obvious examples, but some other notes were much more difficult label, particularly at the boundaries between *good* and *bad* data. So it is not clear that the truth data itself is accurate to within a few percent.

Having said that, however, the combined classifier does have a few things going for it. First, there are only four extra parameters, so the danger of over fitting seems very small. Secondly, the combined classifier does consistently seem to out perform the amplitude-only classifier, at least on average. The combined classifier also takes into account the kind of features that were used initially to create the truth data. So at best, the combined classifier is a genuinely better classifier and at worst, it

should do no worse than the AM magnitude-only classifier.

7.6 Summary

The frequency estimates from section 5.4 are not always correct. When these incorrect frequency estimates are used to estimate harmonic amplitudes (chapter 6), the result is simply noise. This chapter outlines an algorithm for estimating which frequency estimates in a note are reliable.

Two features are used to estimate reliability, smoothness and vibrato magnitude. The smoothness feature essentially makes the assumption that the fundamental frequency of the voice should not change drastically from one frame to the next. The vibrato magnitude feature essentially shows how well the frequency estimates across a note match to the typical characteristics of vibrato in western classical singing. Once the features have been calculated, frame-level likelihoods are calculated for both the 'good' and 'bad' states. Finally an HMM is used to calculate the frame level labels for each note based on the likelihoods.

The vibrato magnitude feature is much better at distinguishing good from bad data than the smoothness feature, but the two features combined seem to do the best job. So, both are used to label the data before it is used in the next chapter.

Chapter 8

Estimates of vowel characteristics

Previous chapters have discussed estimating the fundamental frequency of a note (chapter 5), using that fundamental frequency to make a localized estimate of harmonic amplitudes (chapter 6), and finally deciding which parts of the note as whole are most likely to have good data (chapter 7), but recall that the localized harmonic amplitude estimates are prone to error. In a single short window, the amplitude is estimated from a few DFT coefficients and so the noise can have a significant effect on the estimate.

The obvious solution to the problem of localized noise is to average across time. Unfortunately, a simple average of the absolute harmonic amplitudes will not work because the note as a whole may get louder or softer. This chapter discusses two models for improving and normalizing the harmonic amplitude estimates across all the available data. Sections 8.1 to 8.3 develop the models. Section 8.4 discusses estimate reliability. Section 8.5 contains experiments on simulated data. Section 8.6 discusses the models and their uses. Finally, section 8.7 summarizes the chapter.

8.1 Developing vowel models from the source-filter model

The source-filter model for the vocal tract (section 2.1) holds that the vocal folds produce the excitation signal which is filtered by the vocal tract. So a signal $y_k(t)$ at time k in a window of length T seconds, can be modeled as

$$Y_k(j\Omega) = H_k(j\Omega)X_k(j\Omega) \tag{8.1}$$

where $H_k(j\Omega)$ is the filter and $X_k(j\Omega)$ is the source excitation. For simplicity, assume that the filter $H_k(j\Omega)$ essentially does not change with time, since the notes are being sung and held on a single vowel, i.e. $H_k(j\Omega) = H(j\Omega)$.

The excitation pattern, or vocal fold source, can be modeled as a sum of harmonic sinusoids. If $E_i(kT)$ is the (positive, real) magnitude of the i^{th} harmonic, constant across frame k, and $\Omega_f(kT)$ is the fundamental frequency, also constant across frame k, then the Fourier Transform of the excitation pattern consists of symmetric delta functions at multiples of the fundamental frequency,

$$|X_k(j\Omega)| = \sum_{i=1}^{N_H} E_i(kT)\,(\,\delta(\Omega - i\cdot\Omega_f(kT)) + \delta(\Omega + i\cdot\Omega_f(kT))\,) \tag{8.2}$$

where N_H is the number of harmonics. Around each harmonic frequency, i.e. for $\Omega \sim i\cdot\Omega_f$, the Fourier Transform of $y_k(t)$ consists of a delta function weighted by the filter and the magnitude of excitation harmonic,

$$|Y_k(\Omega \sim i\cdot\Omega_f)| = |H(j\Omega)|\cdot E_i(kT)\cdot\delta(\Omega - i\cdot\Omega_f(kT)). \tag{8.3}$$

This presents a problem because if only $|Y_k(\Omega \sim i\cdot\Omega_f)|$ can be measured, there is an inherent ambiguity between the magnitude effect of the filter, $H(j\Omega)$, and the magnitude of the excitation pattern $E_i(kT)$.

To get around this difficulty, assume that the excitation frequency is at least constant across harmonics, if not across time, i.e. $E_i(kT) = E(kT)$:

$$|X_k(j\Omega)| = E(kT) \sum_{i=1}^{N_H} \left(\delta\left(\Omega - i \cdot \Omega_f(kT)\right) + \delta\left(\Omega + i \cdot \Omega_f(kT)\right)\right) \tag{8.4}$$

and so

$$|Y_k(\Omega \sim i \cdot \Omega_f)| = |H(j\Omega)| \cdot E(kT) \cdot \delta(\Omega - i \cdot \Omega_f(kT)). \tag{8.5}$$

Now $E(kT)$ represents a time-varying energy curve that might be thought of as the breath energy. Since the glottal source is not actually made up of equally energetic harmonics, this assumption will push the glottal harmonic strength patterns into the 'filter' $H(j\Omega)$, which will now become something very similar to the composite transfer function used by Mellody, Herseth, and Wakefield in [34].

The question is how to model $H(j\Omega)$. During the course of a note, the singer will sweep out a small range of frequencies around the nominal note frequency, Ω_0 and so a small section of $H(j\Omega)$ will also be swept out near each harmonic of Ω_0.

One possible 'log model' uses exponents so everything becomes linear in the log domain and assumes

$$|H(j\Omega)| \approx |H(ji \cdot \Omega_0)| \exp\left\{(\Omega - \Omega_0) \left.\frac{\partial \log |H(ji \cdot \Omega')|}{\partial \Omega'}\right|_{\Omega'=\Omega_0}\right\} \quad \text{for} \quad \Omega \sim i \cdot \Omega_0, \tag{8.6}$$

where $|H(ji \cdot \Omega_0)|$ is the gain for each harmonic and $\left.\frac{\partial \log |H(ji \cdot \Omega')|}{\partial \Omega'}\right|_{\Omega'=\Omega_0}$ is the local log slope. In the log domain this becomes,

$$\log |H(j\Omega)| \approx \log |H(ji \cdot \Omega_0)| + (\Omega - \Omega_0) \left.\frac{\partial \log |H(ji \cdot \Omega')|}{\partial \Omega'}\right|_{\Omega'=\Omega_0} \quad \text{for} \quad \Omega \sim i \cdot \Omega_0. \tag{8.7}$$

So this model essentially assumes that the filter can be modeled in the log domain as a line around each harmonic with a harmonic gain and local slope.

Another possible model is in the linear domain,

$$|H(j\Omega)| \approx |H(ji \cdot \Omega_0)| + (\Omega - i\Omega_0)\frac{\partial |H(ji \cdot \Omega')|}{\partial \Omega'}\Big|_{\Omega'=\Omega_0} \quad \text{for} \quad \Omega \sim i \cdot \Omega_0, \tag{8.8}$$

where $|H(ji \cdot \Omega_0)|$ is again the gain, and $\frac{\partial |H(ji \cdot \Omega')|}{\partial \Omega'}\Big|_{\Omega'=\Omega_0}$ is now the local slope in the linear domain.

These two models are further explored in section section 8.2 (log model) and section 8.3 (linear model).

8.2 Log model

To convert equation (8.7) from the continuous domain to the discrete domain, note that during the initial frame-based amplitude estimation (section 6.2), the measured amplitudes are effectively trying to estimate

$$A_i(kT) = \int_{i\cdot\Omega_f(kT)-\epsilon}^{i\cdot\Omega_f(kT)+\epsilon} |Y_k(j\Omega)| d\Omega \tag{8.9}$$

If the average value $|H(ji \cdot \Omega_0)| = g_i$ and the local log-slope $\frac{\partial \log |H(ji \cdot \Omega')|}{\partial \Omega'}\Big|_{\Omega'=\Omega_0} = s_i$, then combining equations equations (8.5), (8.6) and (8.9) and calling $A_i(kT) = A_i[n]$, $E(kT) = E[n]$, $\Omega = f$, and $\Omega_0 = f_0$ yields

$$\log A_i[n] = \log E[n] + \log g_i + s_i \cdot i\,(f[n] - f_0)\,. \tag{8.10}$$

The question is how to best selection $E[n]$, g_i, and s_i so that $A_i[n]$, the model amplitudes, best fit $M_i[n]$, the noisy measured amplitudes. The least-squares solution to this problem is relatively straightforward, but unfortunately, this model does not

yield a unique solution for $E[n]$, g_i, and s_i. Note that

$$\begin{aligned}\log E[n] + \log g_i &= \log\left(E[n]g_i\right) \\ &= \log\left(\frac{1}{m}E[n] \cdot mg_i\right) \\ &= \log\left(\frac{1}{m}E[n]\right) + \log\left(mg_i\right),\end{aligned} \tag{8.11}$$

which essentially means that there is trade off between putting energy in the underlying energy curve and putting energy in the individual harmonic gains. Note further that

$$\log E[n] + s_i \cdot i\left(f[n] - f_0\right) = \left(\log E[n] - a(f[n] - f_0)\right) + \left((s_i + \frac{a}{i}) \cdot i\left(f[n] - f_0\right)\right). \tag{8.12}$$

So there is a subtle trade off between the slope and the underlying energy curve as well.

These two degrees of freedom present a problem. The trade-off between the gains, g_i, and the energy curve, $E[n]$, is relatively easy to deal with because it is simply a scaling problem. One of the gains can be set to a constant for the purposes of solving equation (8.10) and then the gains and energy curve can be normalized later to something sensible. The slope and energy curve trade-off is more difficult. Really what is happening here is that changing the slopes moves some of the vibrato in $(f[n] - f_0)$ into the energy curve. If the energy curve is supposed to be a slowly-varying function, then all the vibrato variation should be captured by the slopes. Enforcing this constraint in equation (8.10) is much more difficult.

Note, however, that if $E[n]$ is known, then the log model becomes a very simple linear equation with gains g_i, slopes s_i and no extra degrees of freedom,

$$\log A_i[n] - \log E[n] = \log g_i + s_i \cdot i \left(f[n] - f_0 \right). \tag{8.13}$$

So if $E[n]$ can be estimated, the the least squares solution to $\log g_i$ and s_i is simply a polynomial best fit.

8.2.1 Estimating the underlying energy curve

$E[n]$ represents the breath energy or dynamic intensity, so it should be slowly varying relative to the vibrato. The most obvious way of estimating this curve is to use a low pass filter. Looking back at the model in equation (8.10), note that

$$\begin{aligned} \sum_i \log A_i[n] = & N_H \log E[n] + \sum_i \log g_i \\ & + \sum_i \left(s_i \cdot i (f[n] + f_0) \right) \end{aligned} \tag{8.14}$$

Now suppose that

$$f[n] - f_0 = f_c + f_z[n] \tag{8.15}$$

where $f_z[n] = f[n] - f_0 - f_c$ and $f_c = mean(f[n] - f0)$. So, essentially $f_z[n]$ is the vibrato undulation centered around zero and f_c is a constant offset. Recall that f_0 is simply the nominal note frequency and may not actually be the center frequency. Now

$$\begin{aligned} \sum_i \log A_i[n] = & N_H \log E[n] + \sum_i \log g_i \\ & + \sum_i s_i \cdot i \cdot f_c + \sum_i s_i \cdot i \cdot f_z[n] \end{aligned} \tag{8.16}$$

where $\sum_i s_i \cdot i \cdot f_z[n]$ is the only term that contains quickly varying vibrato. So, low-pass filtering will yield

$$\begin{aligned} L[n] \equiv lowpass(\sum_i \log A_i[n]) \approx & N_H \log E[n] + \sum_i \log g_i \\ & + \sum_i s_i \cdot i \cdot f_c \end{aligned} \tag{8.17}$$

where $\sum_i \log g_i + \sum_i s_i \cdot i \cdot f_c$ is simply some currently unknown constant and essentially,

$$L[n] \approx N_H \log E[n] + N_H \log c = N_H \log cE[n]. \tag{8.18}$$

This is very close to giving an estimate for $E[n]$ as it is only off by a constant multiplier. For now, let's call this the unnormalized estimate for $E[n]$,

$$\log \hat{E}_u[n] = \frac{1}{N_H} L[n] \tag{8.19}$$

and worry about normalizing later to get rid of the constant (section 8.2.2).

Since in practice filters have a finite time support, not all of $E[n]$ can be estimated. In particular, for a linear phase filter of length N_f, $\frac{N_f-1}{2}$ data points on either end of $E[n]$ are lost. So the trick is to come up with a filter that is as short as possible, but removes the vibrato. A standard low-pass linear phase filter is probably a bad choice because one with an appropriately sharp cutoff will be too long.

Although the filter needs to be essentially a low-pass filter so that it can remove noise as well the vibrato, the most energetic part of the frequency spectrum that needs to be removed is the vibrato. Suppose, for the sake of argument, that the

vibrato wavelength is exactly N_v windows[1] long. A moving average filter with uniform gain $\frac{1}{N_v}$ and length N_v will exactly cancel out the vibrato and will remove a fair amount of high frequency noise.

Remember that vibrato is usually about 6 Hz. In order to capture the frequency detail of the vibrato, the time between frequency and amplitude estimates has been set to about 12 ms. This works out to roughly 14 windows per vibrato cycle, which means a loss of about 7 points on either end of the note after filtering. A general low pass filter would have to be much longer to attenuate the vibrato frequency as well.

Figuring out the vibrato frequency is not a problem as that has already been taken care of in chapter 7, where a model of vibrato is used in each note to remove badly estimated data. The question is what do about the fact that the vibrato rate is not completely constant and the fact that the vibrato wavelength is not usually an integer multiple of the inter-window time. In practice, (section 8.5) the moving average filter only needs to be approximately the correct length in order to produce sufficiently accurate results. So, the filtered can be set to

$$f_v[n] = \frac{1}{N_v} \sum_{i=1}^{N_v} \delta[n - i - 1] \tag{8.20}$$

where N_v is the odd integer closest to the average vibrato wavelength in the note.

8.2.2 Normalizing the gains and energy curve

The initial energy curve estimate $\hat{E}_u[n]$ (equation (8.19)) needs to be normalized. One way to approach this normalization is to trade energy between the energy curve and the gains. As noted previously, $\log\left(\frac{1}{m}E[n]\right) + \log m g_i = \log E[n] + \log g_i$. One

[1]Recall that the pitch track estimates are made for short windows. For the experiments in this chapter, the window length is 1024 samples (23 ms at a 44.1 kHz sampling rate) and the hop between windows is 512 samples (12 ms) (section 8.5.1).

simple normalization is to say that

$$\sum_i g_i^2 = 1. \tag{8.21}$$

Normalizing the gains in this way has the advantage that it makes gains comparable across notes.

So let us suppose that the log model (equation (8.10)) has been solved with parameters $E_u[n]$, $g_{u,i}$, and s_i, where the energy curve and gains are both off by some constant. If $g_i = mg_{u,i}$ and therefore $E[n] = \frac{1}{m}E_u[n]$, then

$$\sum_i g_i^2 = \sum_i (mg_{u,i})^2 = 1 \tag{8.22}$$

and the normalization factor m is

$$m = \frac{1}{\sqrt{\sum_i g_{u,i}^2}}. \tag{8.23}$$

8.2.3 Summary

The complete steps for estimating the log model parameters are

1. Estimate the unnormalized energy curve $E_u[n]$ values using equation (8.19) with the measured amplitudes $M_i[n] \approx A_i[n]$.

2. Find the least squares best fit to equation (8.13) of the gains $g_{u,i}$ and slopes s_i using $\hat{E}_u[n]$ from the previous step and measured amplitudes $M_i[n] \approx A_i[n]$.

3. Calculate the normalizing factor m (equation (8.23)) to normalize $\hat{g}_{u,i}$ and $\hat{E}_u[n]$ to $\hat{g}_i$ and $\hat{E}[n]$.

8.3 Linear model

To convert the linear model (equation (8.8)) from the continuous domain to the discrete domain, call the the average value $|H(ji \cdot \Omega_0)| = g_i$ and the local slope $\frac{\partial |H(ji \cdot \Omega')|}{\partial \Omega'}\Big|_{\Omega'=\Omega_0} = s_i$. Then combining equations equations (8.5), (8.8) and (8.9) and once more calling $A_i(kT) = A_i[n]$, $E(kT) = E[n]$, $\Omega = f$, and $\Omega_0 = f_0$ yields

$$A_i[n] = E[n]\left(g_i + s_i \cdot i\left(f[n] - f_0\right)\right), \tag{8.24}$$

For this model, there is once again a trade off between the energy in $E[n]$ and the energy in the gains and slopes,

$$E[n]\left(g_i + s_i \cdot i\left(f[n] - f_0\right)\right) = \frac{1}{m}E[n]\left(mg_i + ms_i \cdot i\left(f[n] - f_0\right)\right). \tag{8.25}$$

Furthermore, finding a simple solution to equation (8.24) is difficult because taking the log no longer separates the energy curve, gains, and slopes nicely. But, as with the log model, knowing $E[n]$ makes solving for the least squares solution to the gains and slopes extremely easy since

$$\frac{A_i[n]}{E[n]} = g_i + s_i \cdot i\left(f[n] - f_0\right). \tag{8.26}$$

So once again, the best way to solve for $E[n]$, g_i, and s_i is to first estimate the underlying energy curve.

8.3.1 Initial estimate of the energy curve

As with the log model, the underlying energy curve should be slowly varying. To disentangle this slowly varying signal from the rest of the signal, consider

$$\sum_i A_i[n]^2 = \sum_i \left(E[n]\left(g_i + s_i \cdot i(f[n] - f_0)\right)\right)^2 \tag{8.27}$$

$$= E[n]^2 \sum_i \left(c + highpass[n]\right) \tag{8.28}$$

$$\tag{8.29}$$

where c is some constant and $highpass[n]$ is a high frequency signal involving the vibrato. So, when equation (8.29) is low pass filtered,

$$lowpass\left(\sum_i A_i[n]^2\right) \approx E[n]^2 \sum_i c \tag{8.30}$$

Similar to the log model, equation (8.30) can be used to estimate $E[n]$ to within a constant multiplier,

$$\hat{E}_u[n] \approx \sqrt{lowpass\left(\sum_i M_i[n]^2\right)}, \tag{8.31}$$

which can then be normalized later. Once again, the question is how to design the low pass filter. This time, since $M_i[n]$ is squared, the vibrato is also squared. This effectively means that $highpass[n]$ contains energy at both the vibrato frequency and twice the vibrato frequency[2]. Fortunately, the moving average filter described in equation (8.20) is actually a comb filter that best attenuates integer multiples of the lowest frequency $1/N_v$.

[2]Essentially, the vibrato is a sinusoid plus an offset. Note that $(\cos(x) + a)^2 = \cos^2(x) + 2a\cos(x) + a^2 = \frac{1}{2}cos(2x) + \frac{1}{2} + 2a\cos(x) + a^2$.

8.3.2 Normalizing the gains, slopes, and energy curve

Estimating the energy curve using equation (8.31) yields $\hat{E}_u[n]$. Recall that equation (8.25) shows that if $E[n]$ is off my a constant multiplier, g_i and s_i will be off by the inverse of that multiplier. So, using the unnormalized estimates $\hat{E}_u[n]$ in equation (8.26) to estimate the gains and slopes will yield unnormalized estimates $\hat{g}_{u,i}$ and $\hat{s}_{u,i}$ (equation (8.25)). In other words,

$$\hat{g}_i = m \cdot \hat{g}_{u,i} \tag{8.32}$$

$$\hat{s}_i = m \cdot \hat{s}_{u,i} \tag{8.33}$$

$$\hat{E}[n] = \frac{1}{m}\hat{E}[n] \tag{8.34}$$

where m is a constant multiplier.

The question is how to pick m. Once again, normalizing the gain energy to sum to one would allow easy comparison between gains calculated on different notes. So, if $\sum_i g_i^2 = 1$, then

$$\sum_i g_i^2 = \sum_i (m g_{u,i})^2 = 1. \tag{8.35}$$

Solving for m gives

$$m = \frac{1}{\sqrt{\sum_i g_{u,i}^2}}. \tag{8.36}$$

8.3.3 Summary

The complete steps for the linear model are

1. Calculate the energy curve estimate $\hat{E}_u[n]$ using equation (8.31).

2. Estimate the gains and slopes using equation (8.26) and the standard polynomial least-squares equations.

3. Calculate m (equation (8.36)) to normalize the energy curve, gains, and slopes.

8.4 Estimate reliability

While it is simple to measure errors when truth data is known, it is obviously useful to be able to estimate how good the gain and slope estimates are in the absence of truth data. In other words, it is useful to have 'error bars' on the estimates. Since it is not necessarily immediately obvious what those error bars should be, this section develops a definition for error bars based on a probability model of the noise.

8.4.1 Log model

Starting with the log model, recall that

$$\log A_i[n] - \log E[n] = \log g_i + s_i i \cdot (f[n] - f_0). \tag{8.37}$$

Let's define $norm(A_i[n]) \equiv \log A_i[n] - \log E[n]$ as the amplitudes normalized by the energy curve, which should yield a simple straight line when plotted against $f[n]$. Note that after calculating estimate $\hat{E}[n]$ from the noisy measured amplitudes $M_i[n]$, the idea is that

$$\log M_i[n] - \log \hat{E}[n] \equiv norm(M_i[n]) \approx \log \hat{g}_i + \hat{s}_i i \cdot (f - f_0). \tag{8.38}$$

where the "$\approx$" is in the least-squares sense. So, define a set of points

$$p_{\hat{g}_i,\hat{s}_i}[n] \equiv norm(M_i[n]) - (\log \hat{g}_i + \hat{s}_i i \cdot (f - f_0)). \tag{8.39}$$

If the model fit the data perfectly, then $p_{\hat{g}_i,\hat{s}_i}[n]$ would be zero for all time instants n and harmonics i. Of course, since the amplitude estimates are noisy, the points

$p_{\hat{g}_i,\hat{s}_i}[n]$ are unlikely to really be zero, but the numbers should be close to zero if the model is a good fit to the data.

Modeling the noise in p as a Gaussian would yield

$$p_{\hat{g}_i,\hat{s}_i}[n] \sim N(0, \sigma_i^2), \tag{8.40}$$

where the mean of the Gaussian is zero since the gain has been subtracted out and the variance, σ_i^2, indicates the spread of the data for a particular harmonic. The log likelihood of a data point under this model would then be

$$LL(p_{\hat{g}_i,\hat{s}_i}[n]) = -\frac{1}{2}\ln(2\pi\sigma_i^2) - \frac{1}{2\sigma_i^2}p_{\hat{g}_i,\hat{s}_i}[n]^2. \tag{8.41}$$

Now, for each harmonic i, there are N points. The average log likelihood of all these points under the Gaussian assumption would be

$$\begin{aligned}\overline{LL(p_{\hat{g}_i,\hat{s}_i}[n])} &= \frac{1}{N}\sum_{n=1}^{N} LL(p_{\hat{g}_i,\hat{s}_i}[n]) \\ &= -\frac{1}{2}\ln(2\pi\sigma_i^2) - \frac{1}{2N\sigma^2}\sum_{n=1}^{N} p_{\hat{g}_i,\hat{s}_i}[n]^2.\end{aligned} \tag{8.42}$$

This average log likelihood by itself is not terribly useful since it merely measures how well the measured amplitudes fit to the model. It does not directly measure how good a gain or slope estimate is.

One way to think about how good the gain or slope estimates are is ask how much they would have to change in order to reduce the likelihood in a meaningful way. In other words, look at how the log likelihood changes when one of the parameters changes

$$\overline{LL\left(p_{\tilde{g}_i,\hat{s}_i}[n]\right)} - \overline{LL\left(p_{\hat{g}_i,\hat{s}_i}[n]\right)} = \ln\alpha \tag{8.43}$$

for a new gain $\tilde{g}_i$ or

$$\overline{LL\left(p_{\hat{g}_i,\tilde{s}_i}[n]\right)} - \overline{LL\left(p_{\hat{g}_i,\hat{s}_i}[n]\right)} = \ln \alpha \tag{8.44}$$

for a new slope $\tilde{s}_i$.

Now, getting 'error bars' for the gains and slopes is simple. First, estimate σ_i for each harmonic from $p_{\hat{g}_i,\hat{s}_i}[n]$ using the sample variance. Then, select an appropriate α, $0 < \alpha \leq 1$, and solve equation (8.43) or equation (8.44) for $\tilde{g}_i$ or $\tilde{s}_i$. The solutions to these equations are quadratic, and so there will be two solutions to $\tilde{g}_i$ for each $\hat{g}_i$ estimate and two solution to $\tilde{s}_i$ for each $\hat{s}_i$, which makes perfect sense. One solution will be higher than the estimate and one will be lower. Additionally, since these equations are based on reducing the average log likelihood rather than just the log likelihood, the number of points will generally not affect the size of the error bars.

The solutions to equations (8.43) and (8.44) are of the form,

$$0 = (\tilde{c}_i)^2 \cdot \sum_{i=1}^{N} z_i[n]^2 + \tilde{c}_i \cdot 2 \sum_{i=1}^{N} r_i[n] z_i[n] + \sum_{i=1}^{N} r_i[n]^2 - \sum_{i=1}^{N} p_{\hat{g}_i,\hat{s}_i}[n]^2 - 2\sigma^2 N \ln \alpha \tag{8.45}$$

where $\tilde{c}_i$ is the parameter being estimated. If the gain is being estimated, then $\tilde{c}_i = \log \tilde{g}_i$ and

$$r_i[n] = \log M_i[n] - \log \hat{E}[n] - \hat{s}_i i \cdot (f[n] - f_0) \tag{8.46}$$

$$z_i[n] = -1. \tag{8.47}$$

If the slope is being estimated, then $\tilde{c}_i = \tilde{s}_i$ and

$$r_i[n] = \log M_i[n] - \log \hat{E}[n] - \log \hat{g}_i \tag{8.48}$$

$$z_i[n] = -i \cdot (f[n] - f_0). \tag{8.49}$$

8.4.2 Linear model

The story here is very similar. Recall that

$$\frac{A_i[n]}{E[n]} = g_i + s_i i \cdot (f[n] - f_0). \tag{8.50}$$

So, for the linear model, define $norm(A_i[n]) \equiv \frac{A_i[n]}{E[n]}$ and for the estimated amplitudes

$$\frac{M_i[n]}{\hat{E}[n]} \equiv norm(M_i[n]) \approx \hat{g}_i + \hat{s}_i i \cdot (f[n] - f_0). \tag{8.51}$$

Now the set of points that should be close to zero are defined as

$$p_{\hat{g}_i,\hat{s}_i}[n] = norm(M_i[n]) - (\hat{g}_i + \hat{s}_i i \cdot (f[n] - f_0)). \tag{8.52}$$

Once again, noise can be modeled as a Gaussian,

$$p_{\hat{g}_i,\hat{s}_i}[n] \sim N(0, \sigma_i^2). \tag{8.53}$$

The log likelihood and average log likelihood can be defined exactly as they were in the previous section (equations (8.41) and (8.42)) and the reduction in the average log likelihood (equations (8.43) and (8.44)) can again be calculated. The same quadratic form for the error bars holds (equation (8.45)), but the details are slightly different. For the gains, $\tilde{c}_i = \tilde{g}_i$,

$$r_i[n] = M_i[n] - E[n]\hat{s}_i i \cdot (f[n] - f_0) \tag{8.54}$$

$$z_i[n] = -E[n] \tag{8.55}$$

and for the slopes, $\tilde{c}_i = \tilde{s}_i$,

$$r_i[n] = M_i[n] - E[n]\hat{g} - i \tag{8.56}$$

$$z_[n] = -E[n]i \cdot (f[n] - f_0). \tag{8.57}$$

8.5 Experiments and results

Because it is unclear how to get truth data from real recordings, these experiments all use simulated data with noise. Since the log and linear models make different assumptions about the nature of the data, they must also use different simulated data. However, in both cases, the simulated data is combined with white Gaussian noise to achieve different SNRs.

8.5.1 Creating the simulated voices

In both cases, the frequency track $f[n]$ is simulated as

$$f[n] = f_0 + A_s \cos\left(2\pi v \frac{n}{r} + \phi_v\right) \tag{8.58}$$

where f_0 is the note frequency in Hz, v is the vibrato rate in Hz, r is the sampling rate in Hz, ϕ_v is the vibrato phase in radians, and A_s is the amplitude of the vibrato. For simplicity, f_0 is set to 440 Hz (A4), A_s is set to two half steps down from f_0, v is set to 6 Hz, and ϕ_v is pulled randomly from $(0, 2\pi)$.

Also in both cases, the underlying energy curve, $E[n]$, needs to be low frequency. So, the simulated $E[n]$ consists of a simple line with a gentle random slope,

$$E_s[n] = m_E \cdot \frac{n}{r} + b_E \tag{8.59}$$

where m_E is between -0.25 and 0.25, r is the sampling rate, and b_E is $2{\cdot}max\left(|m_E \cdot \frac{n}{r}|\right)$ to make sure $E_s[n]$ is positive and never too small.

The length of each simulated clip is set to one second with a 44.1 kHz sampling rate. Nine harmonics, which is the number that fit under 4000 Hz, are simulated.

The gains, g_i, are linearly spaced between 1 and 0.2 and then normalized so that $\sum_i g_i^2 = 1$ for both the log and linear model. The slopes, s_i, are set so the maximum deviation is between 0% and 20% of $\log g_i$ (log model) or g_i (linear model) given the range of frequencies in $i(f[n] - f_0)$.

Once the harmonic amplitudes $A_i[n]$ are calculated using equation (8.10) or equation (8.24), the AM-modulated FM signal

$$x[n] = \sum_i A_i[n] \cos\left(2\pi i \cdot y[n]\right) \tag{8.60}$$

where

$$y[n] = \sum_{k=0}^{n} \left(\frac{f[n]}{r}\right). \tag{8.61}$$

In each experiment, the SNRs are varied from -3 dB to 3 dB. At each SNR level, ten simulated one-second tracks are created. Frequency estimates for each track are calculated using the algorithm described previously in 5.4 and the amplitude estimates from the algorithm described in 6.2. For frequency estimation, the window size is set to 1024 (23 ms) and the hop size is 512 (12 ms). A hundred fundamental frequencies, linearly spaced from two half steps below the nominal note frequency to the same number of Hz above, are tested using all nine harmonics in order to find the best fundamental frequency estimate.

8.5.2 Results and discussion from the log experiments

Just to get a feeling for what the data looks like, figure 8.1 shows a plot of the amplitudes simulated for the first track in the first experiment. Figure 8.2 shows a plot of the first harmonic of this first track along with the raw estimated harmonic amplitudes and the smooth amplitudes. In this experiment, the SNR is -3 dB. For the most part, the smoothed amplitudes track the truth data pretty well, in spite of the fact that the raw estimates are pretty noisy.

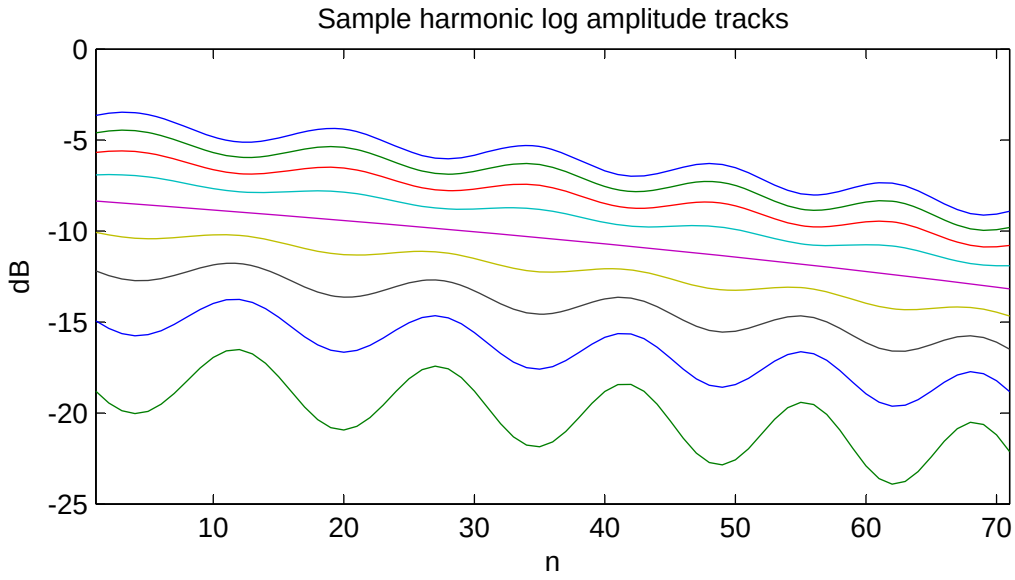

Figure 8.1: Plot of sample amplitudes generated for a single log model experiment. The top line represents the first harmonic, which is the most powerful. The second line from the top is the second harmonic, and so on. Note that the top four harmonics have a negative slope, the middle harmonic has a slope of zero, and the bottom four harmonics have a positive slope.

The amplitude estimates for the log model, which are calculated from the $\hat{E}[n]$, $\hat{g}_i$, and $\hat{s}_i$ estimates, are reasonably close to the truth (figure 8.3). Even with a low SNR of -3 dB, the mean errors are within about 2% of the true values. Unsurprisingly,

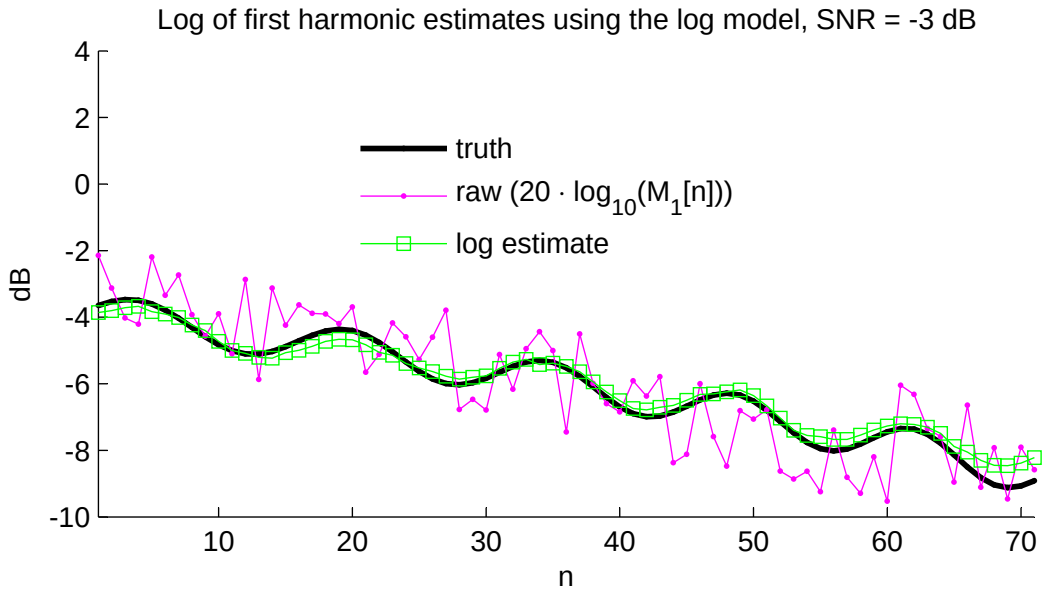

Figure 8.2: Plot of the calculated first harmonic amplitudes for the first track at SNR = -3 dB using the log model.

the error of log the model estimates with respect to the truth data goes down as a function of SNR.

Looking at the parameters themselves, however, the picture is more mixed. Figure 8.4 shows the same error information for the parameters themselves. The gains around each harmonic, g_i, are estimated to within 5% of their true values. The slopes, s_i, and the energy curve, $E[n]$, however, seem to be way off target except in the no-noise case.

Figure 8.5, which gives a more detailed picture of the errors from the energy-based estimate, sheds some light on where the errors are coming from. In all cases, the error goes down as SNR goes down, but this is to be expected.

Consider the energy curve (top right). Here, the vertical axis is n, the time index. Notice that errors are actually spread out pretty evenly across time. In contrast, the largest slope errors (middle right) seem to be associated with the largest slopes,

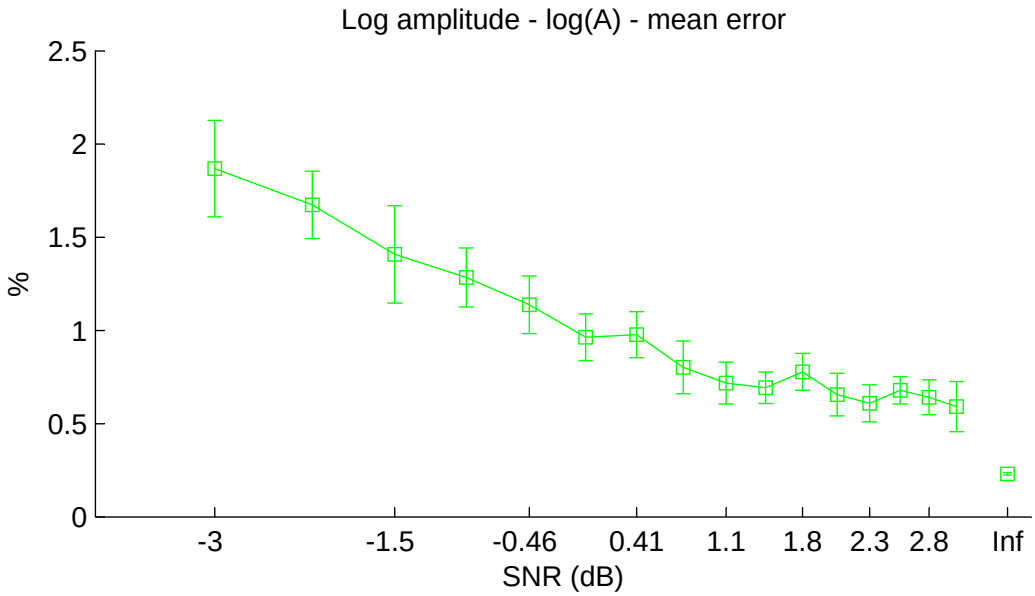

Figure 8.3: Plot of the mean error of the log amplitudes for the log model. The mean is take across all iterations, all harmonics, and all time points relative to the square root of the mean energy of the signal. The error bars represent the standard deviation in this mean calculated across iterations. The last point on the right shows the errors when there is no added

which are associated with the lowest and highest harmonics. Finally, consider the gain errors (middle left). These errors are clustered around the highest harmonics, which also have the lowest gains. Interestingly, this is the pattern that plays out in the amplitude errors (top left), although the amplitude errors are lower than any of the other parameter errors. Recall that the normalization step ensures that the sum of the squares of the gains is one and pushes off the other energy into the underlying energy curve. If this normalization step is off, both the energy curve and gain estimates will have high errors even though the amplitude errors will not change. As far as the slopes are concerned, it is perhaps not all that surprising that large slope errors do not seem to have much of an effect on the amplitude errors.

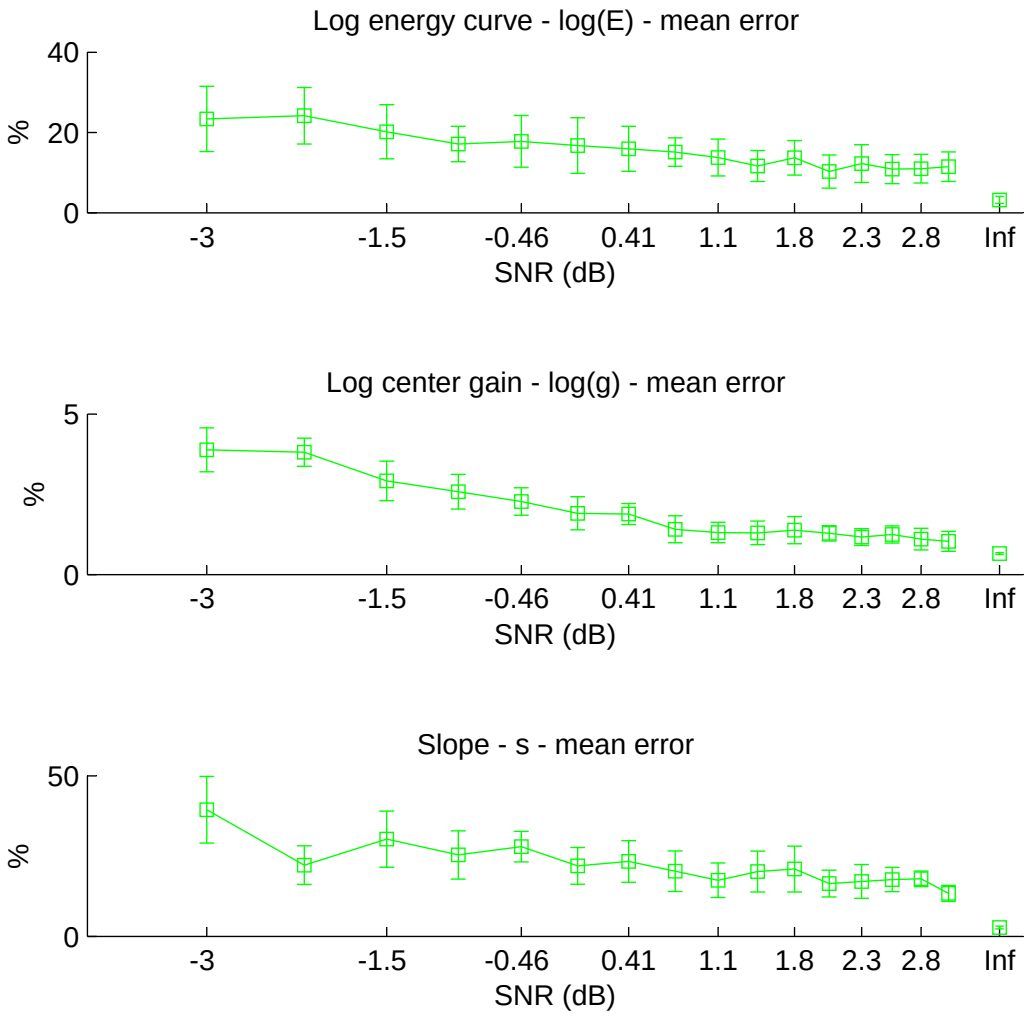

Figure 8.4: Plots of the mean error of the parameters of the log model. The mean is take across all iterations, all harmonics (g_c, s) or all time points (E) relative to the square root of the mean square of the true values. The error bars represent the standard deviation in this mean calculated across iterations.

The slopes can essentially do two things; they can change the phase of the amplitude relative to the fundamental frequency and they can change the 'peakiness' of the amplitude. Altering the 'peakiness' of the amplitude estimate is simply not going to cause large amplitude errors in the same way that the other parameters can cause

errors.

The bottom two plots in figure 8.5 show the mean sizes of the 'error bars' from section 8.4 with $\alpha = 0.95$. Ideally, the error bars would track the actual errors well. In reality, the gain and slope error bars roll off much more slowly than the true error. However, they do seem to capture the general trends reasonably well, so they should make a reasonable proxy to the actual error when the actual error cannot be calculated.

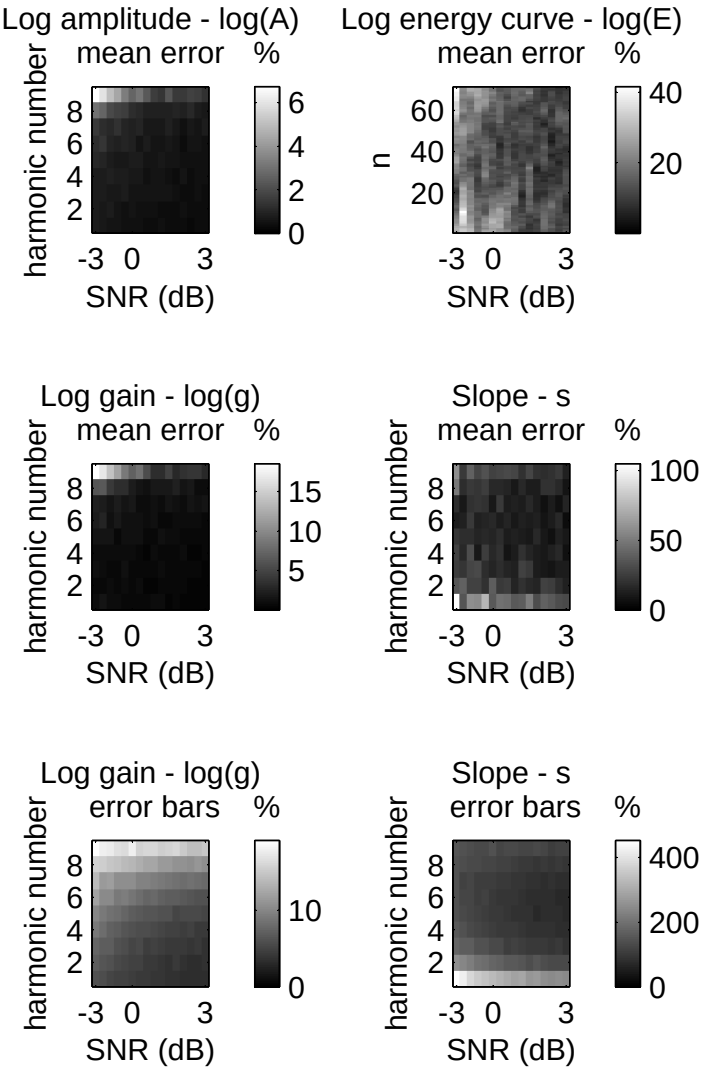

Figure 8.5: The top four plots are plots of the mean error of the amplitudes and parameters of the log model using just the energy-based estimates. The bottom two plots show the mean size of the 'error bars' calculated with $\alpha = 0.95$. This time, the mean is take only across iterations, but the normalization is the same as in figures 8.3 and 8.4.

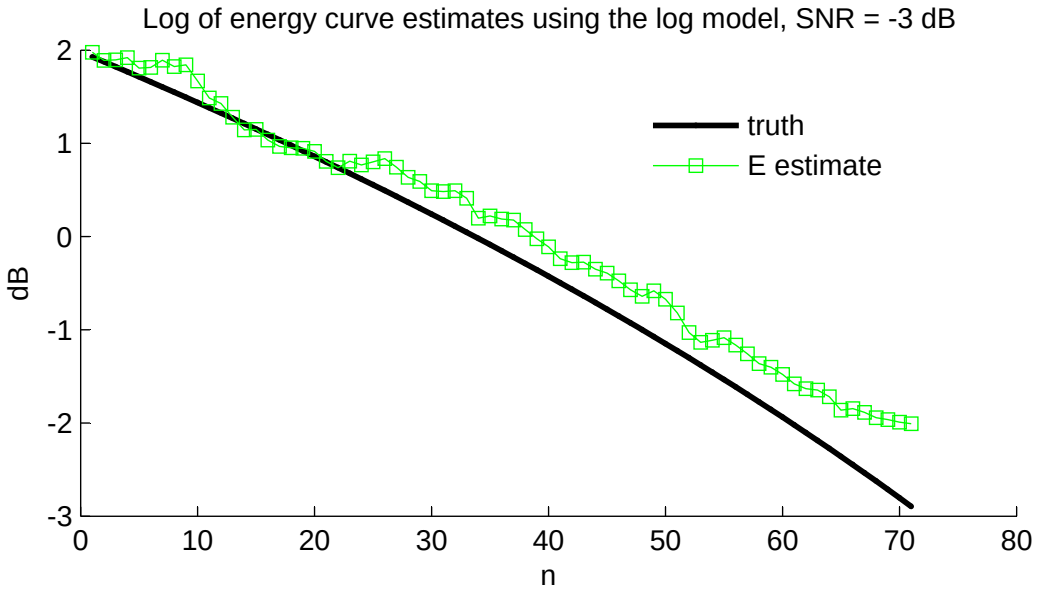

Figure 8.6: Plot of the calculated energy curve for the first track at SNR = -3 dB.

8.5.3 Results and discussion of the linear model experiments

Figures 8.7 and 8.8 once again give a feel of the kind of data produced by the model. Figure 8.7 shows a set of simulated harmonic amplitude tracks from a single test and figure 8.8 shows just the first harmonic amplitude track from the same test along with the raw amplitude measurements $M_1[n]$ and the linear model estimates.

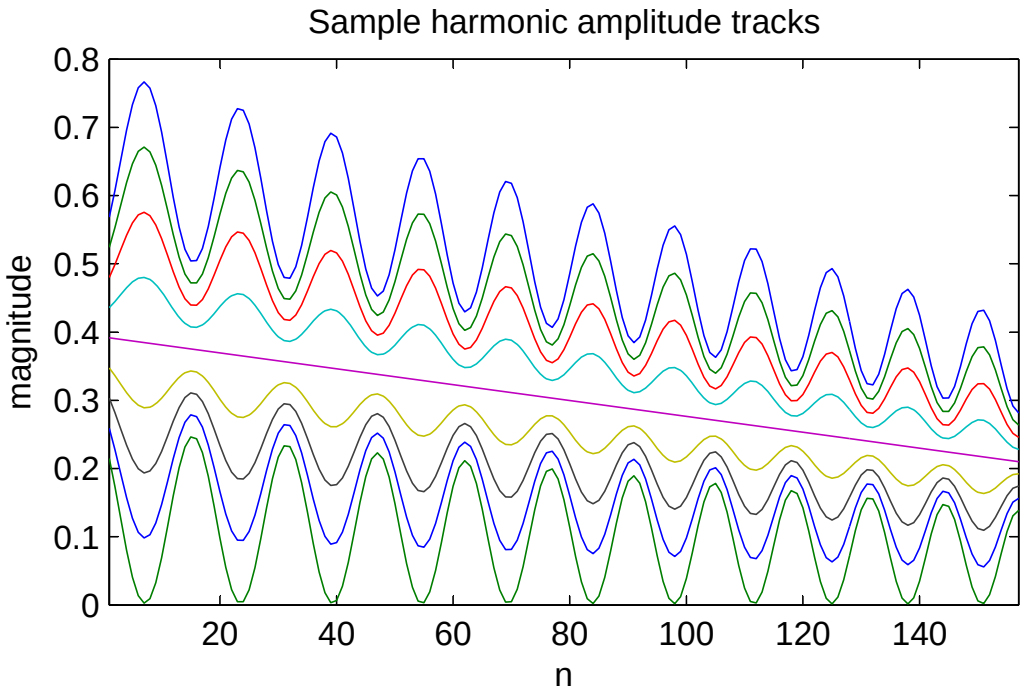

Figure 8.7: Plot of sample amplitudes generate for a single linear model experiment. The top line represents the first harmonic, which is the most powerful. The second line from the top is the second harmonic, and so on. Once again, the top four harmonics have a negative slope, the middle harmonic has a slope of zero, and the bottom harmonics have positive slopes.

As with the log model, the amplitude estimates for the linear model are reasonably close to the truth (figure 8.9) and under 3% at an SNR of -3 dB. Once again, the error of the model with respect to the truth data goes down as a function of SNR.

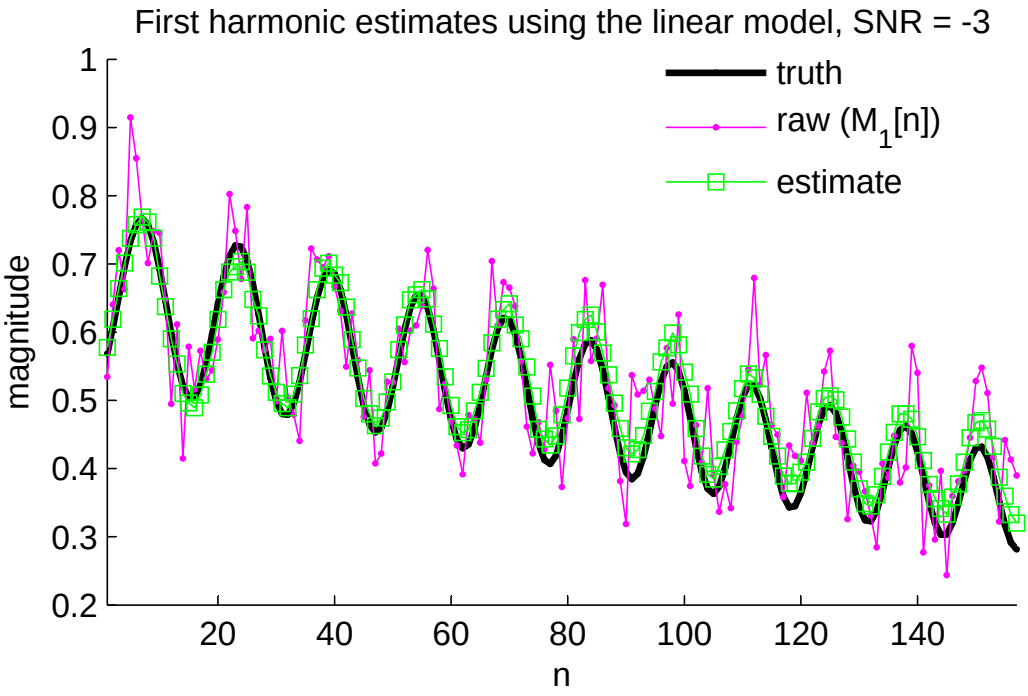

Figure 8.8: Plot of the calculated first harmonic amplitudes for the first track at SNR = -3 dB using the linear model.

Looking at the parameters themselves, however, the picture is different from the log model. Figure 8.10 shows the same error information for the parameters themselves. All the parameters are reasonably estimated, although the slopes still have the highest error.

Figure 8.12 shows the error information broken out across time (energy curve) or harmonics (amplitudes, gains, and slopes). This picture for the linear model is very similar to the picture for the log model. Energy curve errors are spread more or less evenly across time. Gain errors are concentrated around the less energetic higher harmonics, and slope errors seem to be worst around larger slopes. The bottom two plots, which show the error bars, are once again not perfect proxies for the actual error, but they do at least capture the trend of larger errors for higher SNRs.

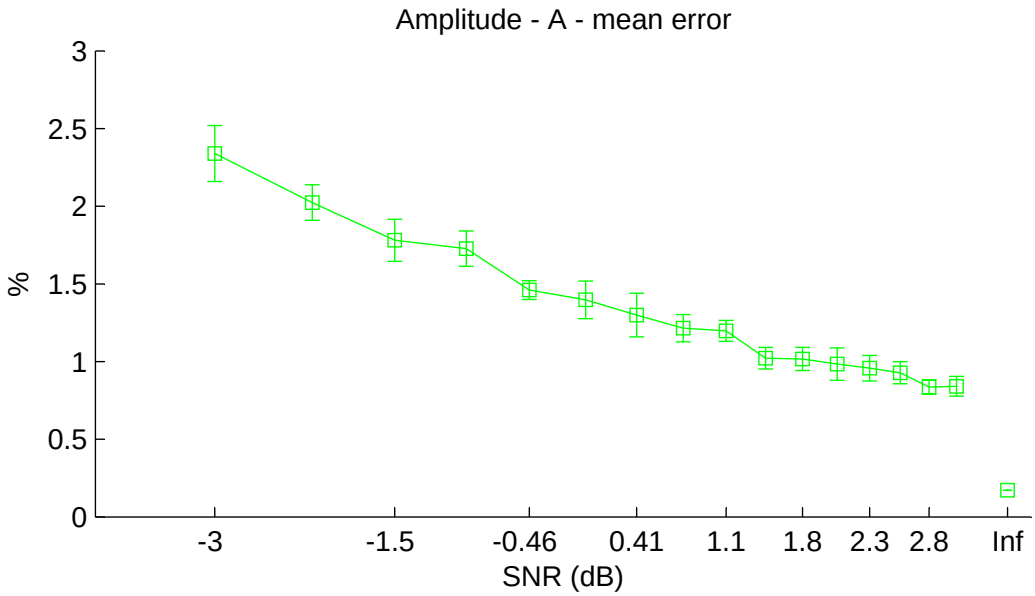

Figure 8.9: Plot of the mean error of the amplitudes for the linear model. The mean is take across all iterations, all harmonics, and all time points relative to the square root of the mean energy of the signal. The error bars represent the standard deviation in this mean calculated across iterations. The last point on the right shows the errors when there is no added noise.

Figure 8.12 once again shows the errors in estimating the amplitudes and parameters across iterations. The pictures is somewhat different in the linear model. Once again, the error tends to go down with higher SNRs. But this time, it seems that the middle harmonics are better estimated than the higher-energy lower harmonics. The slopes in the synthetic data are kept the same through all the tests. The slopes s_i run from negative to positive with the middle harmonics slopes smaller than the outer harmonic slopes. It appears that this model is better able to calculate parameters when the slope is small.

Once again, the bottom two plots show the error bars for the gains and slopes. The error bars do not capture all the detail of the actual errors, although they do at least seem to capture the SNR trend. The error bars actually seem to work less

well over all for the linear model than they did for the log model. Note that the gain errors actually go down for the higher harmonics even though that is where the errors are concentrated. Having said that, the most interesting information is in strong harmonics, not weak ones. So as long as the SNR trend is shown reasonably well, the error bars should have some value.

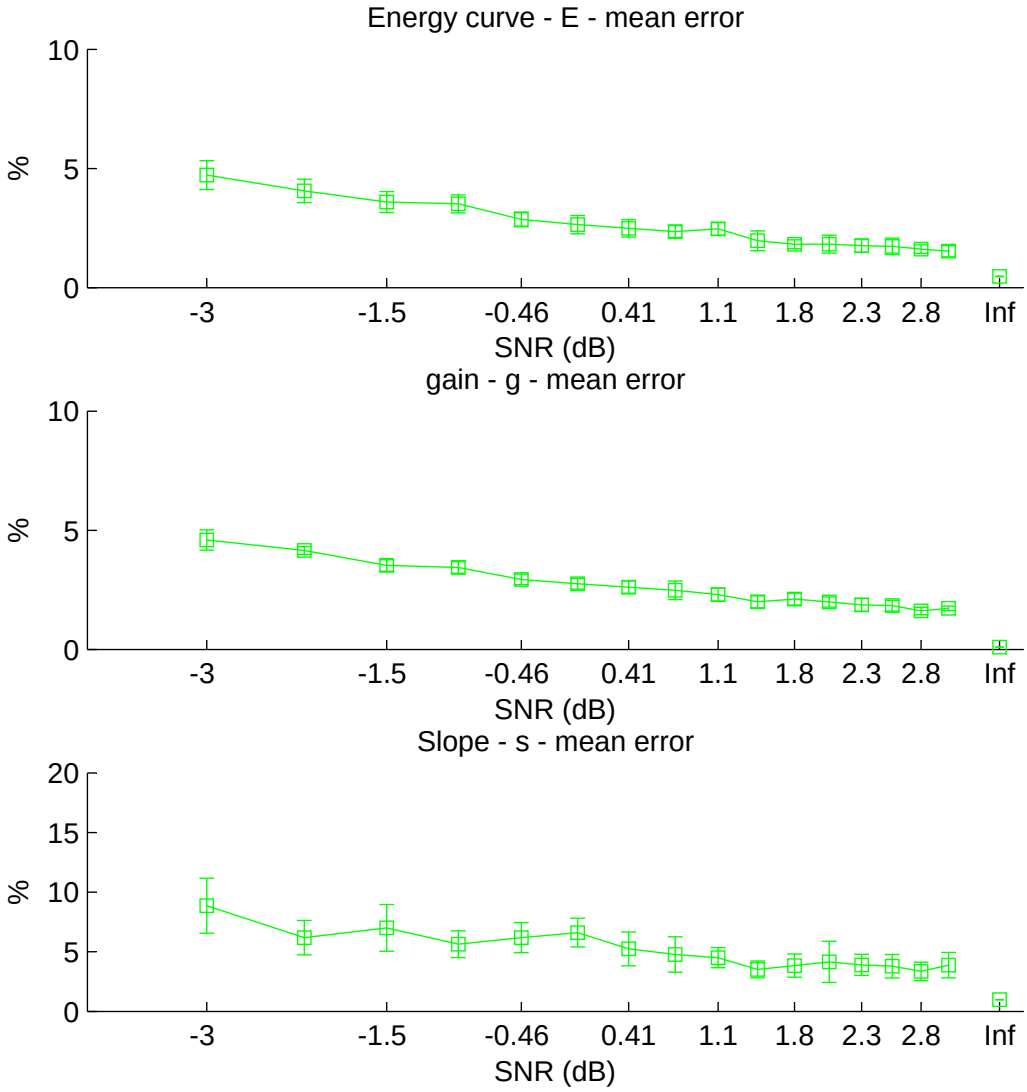

Figure 8.10: Plots of the mean error of the parameters of the linear model. The mean is take across all iterations, all harmonics (g_i, s_i) or all time points (E) relative to the square root of the mean square of the true values. The error bars represent the standard deviation in this mean calculated across iterations.

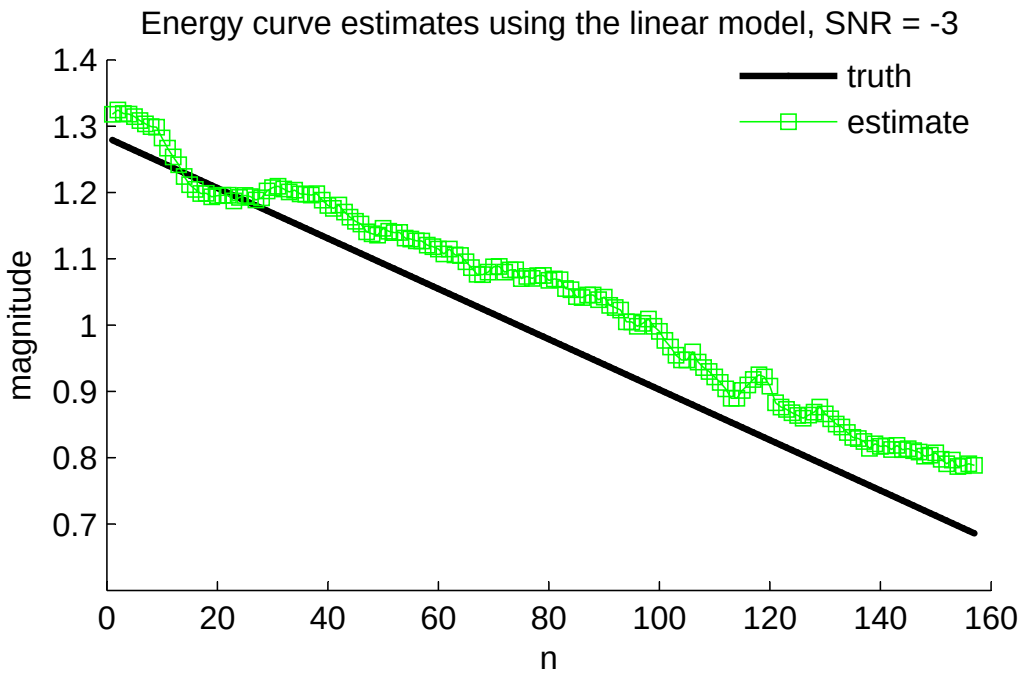

Figure 8.11: Plot of the calculated energy curve for the first track at SNR = -3 dB for the linear model.

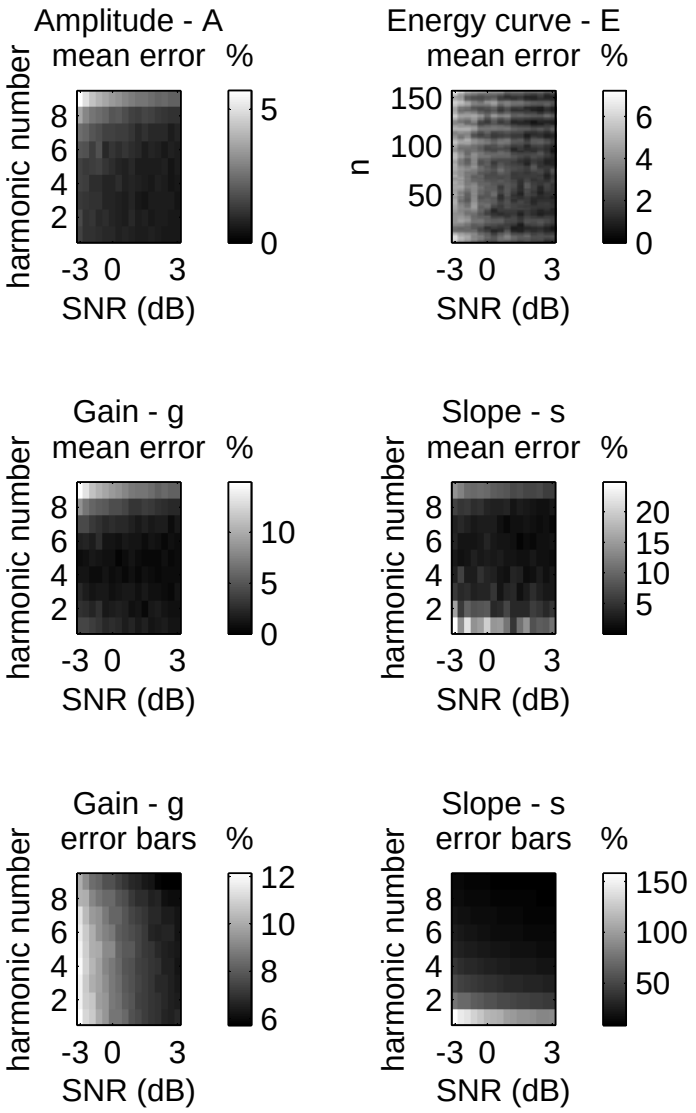

Figure 8.12: Plots of the mean error of the amplitudes and parameters of the linear model and error. The top four plots show the errors in the amplitude and parameter estimates. The bottom two plots show the 'error bar' sizes with $\alpha = 0.95$. Although the mean is take only across iterations, the normalization is the same as in figures 8.9 and 8.10.

8.6 Discussion

These models are not good physical models for the voice. The 'energy' that comes into the vocal tract filter from the vocal folds is not necessarily slowly varying, and the harmonics are certainly not equally energetic. Furthermore, the vocal tract filter is not necessarily stable across the note, so the slope estimate is not really an estimate of the local vocal tract filter.

Instead of trying to estimate the vocal tract, these models are really trying to estimate the parameters of what is important to people listening to the voice. Listeners obviously cannot measure the glottal pressure or the vocal tract shape. They are only able to work with what they can hear, which basically involves groups of harmonically-related frequencies. The absolute loudness of these collections of frequencies is generally unimportant. It is the relative strengths of the harmonic frequencies which carries information, and that is essentially what that gains in both the log and linear models attempt to measure.

It would be interesting to see how the algorithms described in this chapter compare to other algorithms in the literature for estimating voice characteristics. Unfortunately, there aren't really any algorithms that can be directly compared with the ones described here. LPC (section 2.2.2), for example, has been used to estimate the vocal tract filter as an all-pole filter. Unfortunately, LPC is unlikely to work well in the polyphonic case because it would be confused by the other voices. Other algorithms would undoubtedly suffer for the same reason.

8.7 Summary

This chapter takes the estimates described in previous chapters and synthesizes them into useful measures of vowels characteristics. Since the raw amplitude estimates

from chapter 6 suffer from a great deal of noise, the models in this chapter smooth across time in order to come up with more reasonable estimates. Both the linear and log models assume an essentially stable vocal tract filter and a slowly-varying source energy. After estimating this energy source, they then estimate the gain associated with each harmonic and a slope near the harmonic, which is swept out by the vibrato. In the next chapter, these vowel models will be applied to the Gilbert and Sullivan data set (section 2.4).

Chapter 9

Experiments with the Gilbert and Sullivan Data Set

Chapter 8 demonstrated that the log and linear models can do a reasonable job of estimating vowel parameters, particularly harmonic gains. This chapter will apply these models to a real data from the Gilbert and Sullivan data set (section 2.4). Recall that the set consists of six arias from five operettas. These arias have now been time aligned with MIDI by hand, the phones have been assigned (chapter 3), the frequency tracks (section 5.4)[1] and initial frame-based amplitudes (section 6.2) have been calculated, and the bad data has been discarded (chapter 7)[2].

The log and linear models from chapter 8 essentially average the frame-based harmonic amplitude estimates across time after the amplitudes have been normalized by a per-note energy curve. The question is which data points to average over, for

[1] As in section 7.4, a window size of 1024 (23 ms at 44100 sampling rate) and hop size of 512 samples (12 ms) are used in order to capture the the vibrato with sufficient accuracy. The first three harmonics are used to calculate the frequency estimate. The frequency search region around the nominal note frequency is two half steps down from the nominal note frequency and the same number of Hz above. A hundred linearly spaced frequencies are tested in this range.

[2] Based on calculations from the training data, the algorithm's bias is set for 90% precision (section 7.4).

instance to estimate the spectrum of a vowel. Most simply, the harmonic gains and slopes can be calculated from the data in a single note. Hopefully the vocal tract will be reasonably stable and consistent across a single note, so the gain and slope estimates will not have to deal with too much noise. Unfortunately, an individual note may be rather short and not provide many data points over which the average can be calculated.

All the notes sung in a particular operetta are sung by the same soprano. So another option is calculate gains and slopes across many notes sung on the same vowel by the same soprano. Recall, however, that singers adjust the location of vowel formants as the pitch rises, a process called vowel modification (section 2.1.1). So a sensible approach would be to calculate the gains and slopes for a particular nominal note frequency (say all C4s or G5s) and vowel (say AY) combination across all instances of that note frequency and vowel in each operetta. The same soprano should pronounce the same vowel in a similar manner across the entire operetta, so the gain and slope estimates should be able to benefit from more data points.

Once again, however, there may not be sufficient data in a single operetta. So the last option is to calculate the gains and slopes for a particular nominal note frequency and vowel across all instances of that note frequency and vowel in all operettas. After all, the reason why we can understand each other is that we all pronounce the same vowels in similar ways. Furthermore, in a classical music data set such as this one, the accent should remain reasonably consistent across different performances with different singers because performance practice dictates a certain kind of British accent for Gilbert and Sullivan.

The remainder of this chapter will switch between averaging over notes, operettas, and over all data, depending on what makes most sense for the particular task at hand. Section 9.1 characterizes the data from the Gilbert and Sullivan data set.

Sections 9.2 to 9.4 discuss the harmonic strengths of AA, AH, and IY across different pitches. Section 9.5 discusses evidence for the singer's formant. Section 9.6 discusses the chapter as a whole. Finally, section 9.7 summarizes the chapter.

9.1 Characterizing the data

The log and linear models described in the previous chapter need long, sustained vowels whose frequency tracks have been captured successfully. Looking at just the notes with good data (chapter 7), the first and last 200 ms of data are discarded in order to make sure the voice has settled. Then only notes that have at least 20 data points (232 ms) left after filtering[3] are kept. Out of all six arias, this leaves 183 notes distributed across 18 vowels and 18 nominal note frequencies. Figure 9.1 shows the distribution of data. Some vowels such as AA (/ɑ/ as in father) and AY (/aɪ/ in my) are much more popular and have much more data than UW (/u/ as in you). There is only about a one and a half octave range of frequencies and most of the data is concentrated in the middle of that range, from about A4 to F5. Obviously, many notes are missing simply because they are not in the keys of the six arias from which the data is calculated.

In order to get some idea of the quality of the measurements calculated from the Gilbert and Sullivan data sets, tables 9.1 to 9.4 show the error bars (section 8.4) for the first two harmonics for both the log and linear models. Parameters are calculated across each operetta with one set of gains and slopes per vowel/operetta/nominal note frequency combination. The median is taken across all nominal note frequencies. Not unsurprisingly, given the results with test data, the slope values are mostly unreliable for both the log and linear models. The error bars are of similar magnitudes as

[3]Recall from chapter 8 that the energy curve estimate for both the log and linear model is found by filtering.

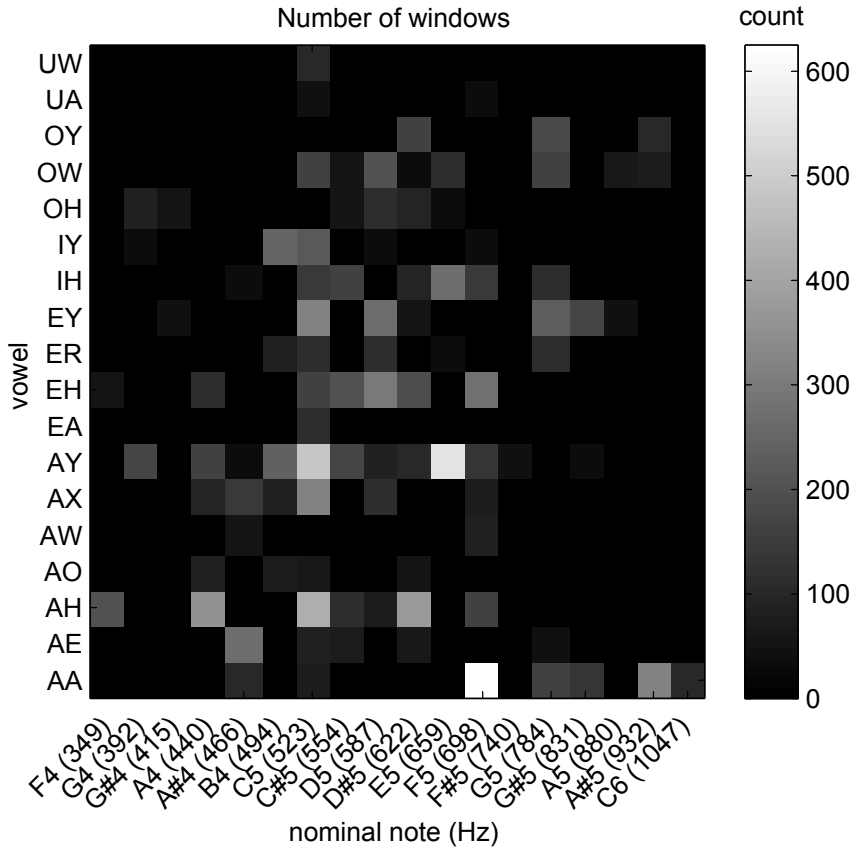

Figure 9.1: Plot of the distribution of data across notes and vowels. The counts refer to the number of windows for a particular vowel and frequency across all six arias. Recall that the notes are chopped up into windows and that for each window, one raw amplitude estimate is made for each harmonic.

the slopes themselves, indicating that the calculated slope parameters are probably mostly noise. Clearly, the data is simply too spread out to calculate a reliable slope. The gains, on the other hand, seem much more reliable. Recall that the gains have been normalized so that the sum of the squares of the gains is one. Most of note energy is generally found in the first couple of harmonics, the harmonics used for these calculations, so error bars on the order of 1 dB seem reasonable.

Since the data is limited and some vowels are better represented than others, it

vowel/op.	HMS	Mikado	Pirates	Gondoliers	Yeomen
AA	0.99/-0.99		0.83/-3.1	0.73/-1.3	1.3/-1.3
AE	0.82/-1.1			2.5/-0.72	1.1/-1.1
AH	0.82/-1	1.5/-0.81	1.5/-0.8	2.4/-1.2	1/-1
AO	0.95/-0.95	0.97/-0.97			0.81/-0.81
AW					1.1/-1.1
AX	1.2/-1.2	0.74/-1.2		1.2/-1.2	
AY	1/-1.1	1.5/-1.6	1.1/-1.1	1.2/-1.2	1.1/-1.1
EA				2.4/-0.68	
EH	1.3/-0.98			1.1/-1.1	
ER	0.62/-0.62	0.98/-0.98		0.25/-6.4	
EY	1/-1.1	1.6/-0.81	2.5/-0.54		
IH	0.78/-1.1		1.2/-1.2	1.1/-1.9	0.99/-0.99
IY	0.93/-0.99	1.3/-1.3		0.77/-0.77	1.7/-0.95
OH	0.79/-0.79		0.82/-0.82		1.2/-1.2
OW	0.81/-0.98	1.2/-1.2			1/-1
OY					0.96/-1.1
UA	1/-1		0.67/-0.67		
UW	1.4/-1.4				

Table 9.1: Median error bars for the log-model gain estimates on the Gilbert and Sullivan data set. Entries are the median distance above (positive) and below (negative) the nominal gain in dB for the first two harmonics. Blank cells indicate there is no data for this particular vowel and operetta combination. "HMS" refers to *H.M.S. Pinafore*, "Mikado" refers to *The Mikado*, "Pirates" refers to *The Pirates of Penzance*, "Gondoliers" refers to *The Gondoliers*, "Yeomen" refers to *The Yeomen of the Guard*.

vowel/op.	HMS	Mikado	Pirates	Gondoliers	Yeomen
AA	0.0031 (0.0014)		0.0046 (0.0059)	0.0055 (0.0036)	0.0077 (0.0029)
AE	0.0033 (0.0014)			0.0098 (0.004)	0.0057 (0.0038)
AH	0.0037 (0.0012)	0.0043 (0.0026)	0.0044 (0.0042)	0.006 (0.0016)	0.0042 (0.0027)
AO	0.0038 (0.0088)	0.0058 (0.0035)			0.004 (0.0061)
AW					0.0053 (0.0023)
AX	0.0042 (0.0033)	0.0042 (0.0041)		0.0073 (0.002)	
AY	0.0034 (0.0014)	0.0062 (0.0025)	0.0041 (0.0026)	0.0048 (0.0016)	0.0031 (0.0016)
EA				0.0041 (0.0022)	
EH	0.0051 (0.0019)			0.0037 (0.0019)	
ER	0.0018 (0.0025)	0.0027 (0.0017)		0.01 (0.0022)	
EY	0.0034 (0.0026)	0.003 (0.0031)	0.0034 (0.0023)		
IH	0.0042 (0.00093)		0.0057 (0.0035)	0.0094 (0.00045)	0.0038 (0.0033)
IY	0.0021 (0.00071)	0.0057 (0.0035)		0.0027 (0.0028)	0.0086 (0.0046)
OH	0.0025 (0.0028)		0.0044 (0.0027)		0.0079 (0.0019)
OW	0.0031 (0.0023)	0.0038 (0.0062)			0.0036 (0.0023)
OY					0.0044 (0.0011)
UA	0.0058 (0.0042)		0.0044 (0.0071)		
UW	0.006 (0.0032)				

Table 9.2: Median error bars for the log-model slope estimates on the Gilbert and Sullivan data set. Entries are the median error bar size for the slope estimates of the first two harmonics (above) and the median of the absolute value of the slope for reference (below) in db/Hz.

vowel/op.	HMS	Mikado	Pirates	Gondoliers	Yeomen
AA	0.69/-1.1		2.3/-0.58	0.91/-1	0.95/-1.1
AE	0.7/-1.1			1.2/-1.1	0.96/-1.1
AH	0.72/-0.83	0.88/-1.1	1/-1.4	1.3/-0.84	0.85/-0.87
AO	0.72/-0.83	0.77/-0.94			0.78/-0.64
AW					0.91/-0.92
AX	0.69/-1.5	0.89/-1.2		0.88/-1.2	
AY	0.79/-0.98	0.85/-1.2	0.9/-1.2	0.8/-1.4	0.83/-1.1
EA				0.71/-2.5	
EH	1.1/-1.1			0.89/-1	
ER	0.57/-0.6	0.82/-0.98		0.27/-5.6	
EY	0.85/-0.9	0.82/-1.1	0.6/-0.85		
IH	0.72/-0.94		1.1/-0.99	2.1/-0.89	0.67/-1
IY	0.68/-1.5	0.86/-1.3		0.64/-0.77	1.1/-1
OH	0.61/-1		0.74/-0.85		0.91/-1.1
OW	0.72/-0.79	0.99/-0.99			0.65/-0.73
OY					0.81/-0.9
UA	0.84/-1		1/-0.66		
UW	1.1/-1.3				

Table 9.3: Median error bars for the linear-model gain estimates on the Gilbert and Sullivan data set. Values are calculated in the same way as in table 9.1. Note that even though this is the linear model, the results are posted here in dB so they can be compared more easily with the log model results.

is difficult to draw really meaningful conclusions across all vowels and notes. Instead, the rest of this chapter will discuss the notes and vowels with the most and best data. It will also only show the log model results since they are essentially the same as the linear model results.

vowel/op.	HMS	Mikado	Pirates	Gondoliers	Yeomen
AA	0.00287 (0.000939)		0.00336 (0.00526)	0.0043 (0.00341)	0.00582 (0.00249)
AE	0.00276 (0.00112)			0.00747 (0.00366)	0.00509 (0.00367)
AH	0.0032 (0.000857)	0.00376 (0.00204)	0.00447 (0.00399)	0.00491 (0.000552)	0.00345 (0.00232)
AO	0.00307 (0.00789)	0.00482 (0.00319)			0.00352 (0.00588)
AW					0.0044 (0.00171)
AX	0.0036 (0.00275)	0.00415 (0.00357)		0.00686 (0.00184)	
AY	0.00295 (0.00101)	0.00516 (0.00188)	0.00395 (0.00305)	0.00383 (0.00132)	0.00391 (0.00169)
EA				0.00483 (0.00225)	
EH	0.00391 (0.00118)			0.00314 (0.00181)	
ER	0.00171 (0.00202)	0.00247 (0.00201)		0.00723 (0.0021)	
EY	0.0031 (0.00203)	0.00231 (0.00254)	0.00275 (0.0017)		
IH	0.00394 (0.0011)		0.00484 (0.0043)	0.00811 (0.00113)	0.00327 (0.00211)
IY	0.002 (0.000519)	0.00431 (0.00324)		0.00247 (0.00263)	0.00744 (0.00505)
OH	0.00224 (0.00218)		0.00364 (0.00269)		0.00567 (0.00164)
OW	0.0038 (0.00183)	0.00327 (0.00443)			0.00302 (0.000851)
OY					0.0037 (0.00139)
UA	0.00527 (0.00336)		0.00512 (0.00793)		
UW	0.00513 (0.00297)				

Table 9.4: Median error bars for the linear-model slope estimates on the Gilbert and Sullivan data set. Values are calculated in the same way as in table 9.2. Note that even though this is the linear model, the slopes are reported in dB/Hz here so they can be compared more easily with the log model results.

9.2 The vowel AA

The vowel AA is characterized by a first formant somewhere around 600 - 1200 Hz and a second slightly higher formant in the range of 1000-1500 Hz [40]. This relatively high first formant helps to make AA a popular choice for singing on higher notes.

Figure 9.2 shows the model parameters for AA calculated across all notes with the same nominal note frequency in all operettas. Perhaps the most interesting thing to note from these plots is how the energy is distributed as the pitch increases. For A#4 and C5, the bottom two plots, the first harmonic is well below the first formant, so the singers place most of the energy in this second harmonic, which is in the correct range. As the pitch goes up, the first harmonic enters the region of the first formant and so the first formant becomes more energetic. On the highest notes, A#5 and C6, the first harmonic has the lion's share of the energy.

Now, in this stack of vowel plots, G#5 stands out because it breaks the pattern. Its second harmonic has slightly more energy than its first harmonic. It is possible that this anomaly is simply the result of noise. However, it is interesting to break up the parameter estimates by operetta and thus by singer. Most of this data comes from notes that are sung either in *H.M.S. Pinafore* (figure 9.3) or *The Gondoliers* (figure 9.4). In *H.M.S. Pinafore*, there is a clear pattern where the energy moves smoothly from the second to the first harmonic as the pitch rises. The trade-off clearly takes place somewhere between C5 and F5. In contrast, *The Gondoliers* seems to show that this trade-off happens at a slightly higher frequency, around A#5. It could be that these two sopranos simply have slightly different vowel modification strategies. The nominal formants for vowels are relatively wide because different people pronounce them slightly differently. It makes sense that these differences would translate into sung vowels as well.

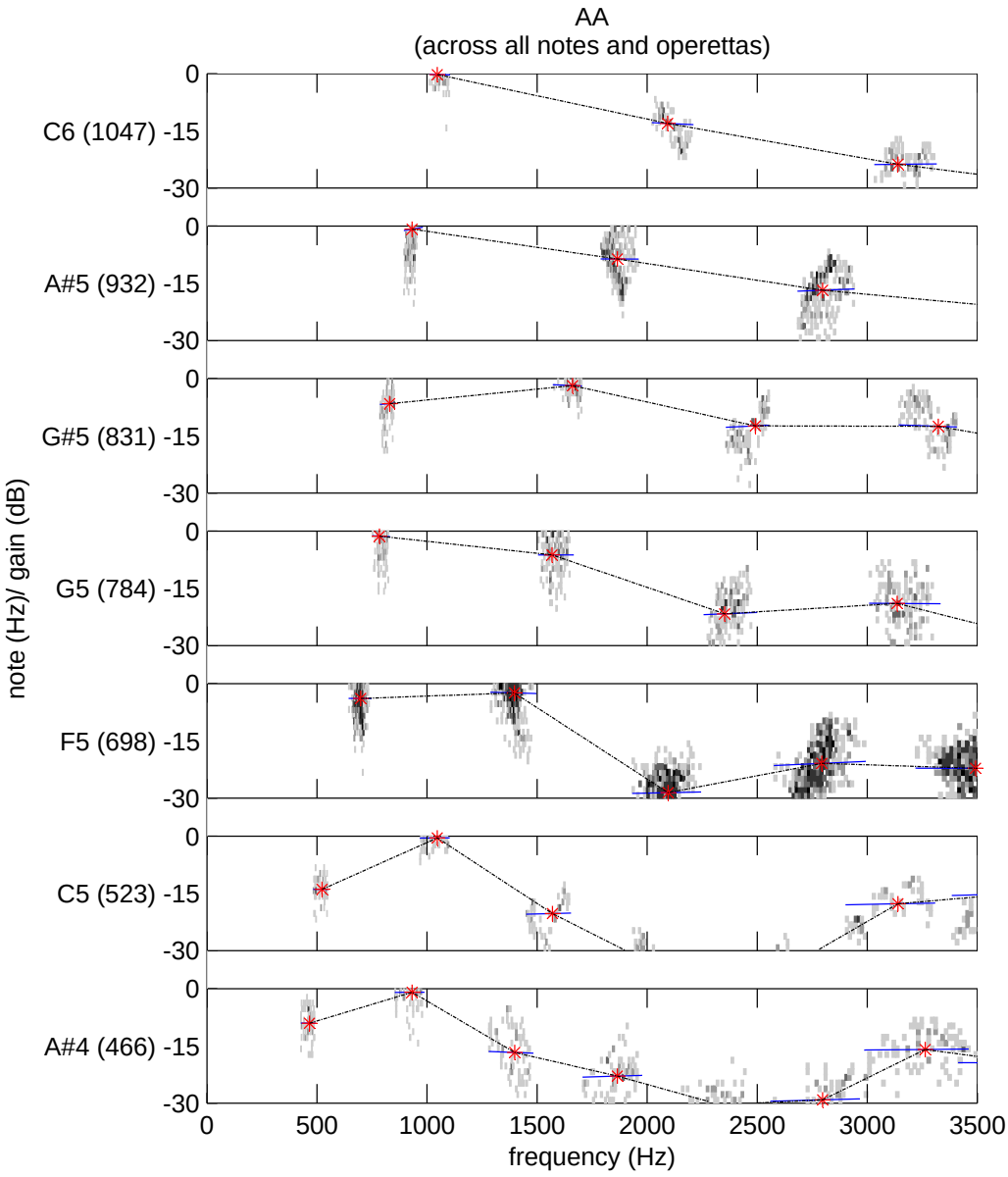

Figure 9.2: AA harmonics across all notes in all operettas. Each plot shows the log model gains (red stars) and slopes (blue lines) for the vowel AA sung on the note indicated to the left of the plot. The clouds behind each gain are histograms of the actual points used to calculate the slopes and gains (raw amplitudes normalized by the energy curve - equation (8.38)). Black points in the histogram represent at least 5 points.

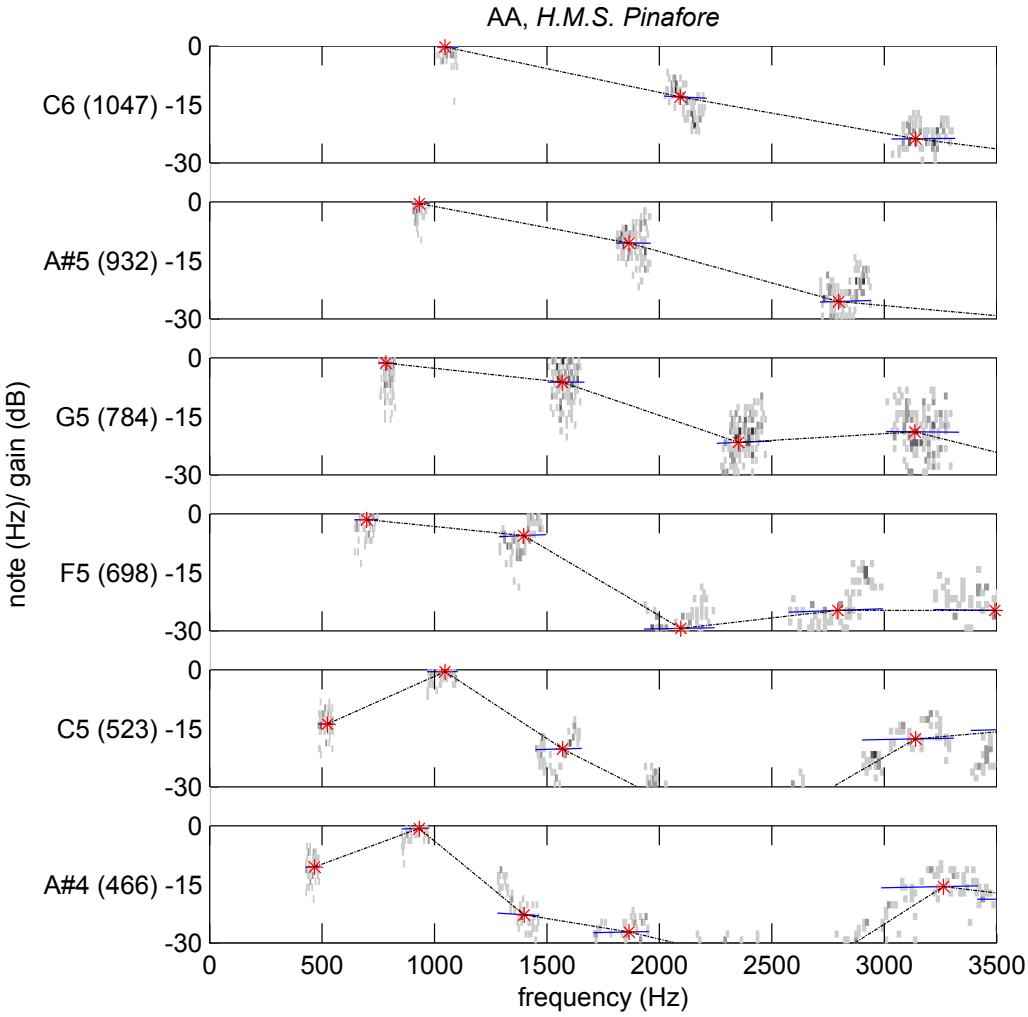

Figure 9.3: AA harmonics across all notes in *H.M.S. Pinafore*. As in figure 9.2, the gains are red stars, the slopes are blue lines, and the histograms show the points used to calculate parameters.

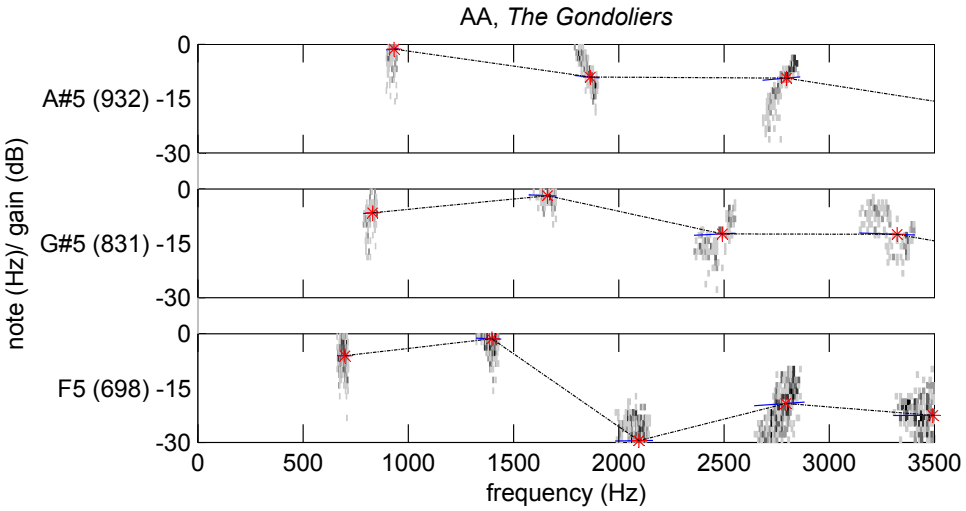

Figure 9.4: AA harmonics across all notes in *The Gondoliers*. As in figure 9.2, the gains are red stars, the slopes are blue lines, and the histograms show the points used to calculate parameters.

9.3 The vowel AH

The vowel AH (/ʌ/ as in up) is characterized by a first formant somewhere around 600 - 1100 Hz and a second slightly higher formant in the range of 1100-1700 Hz [40]. So the formants are just a little higher than those for AA.

The gains really follow the same pattern as the AA gains. In figure 9.5, the energy shifts from higher harmonics to lower harmonics as the pitch increases. At high pitches, it is hard to imagine that anyone would be able to distinguish between vowels because they look very, very similar.

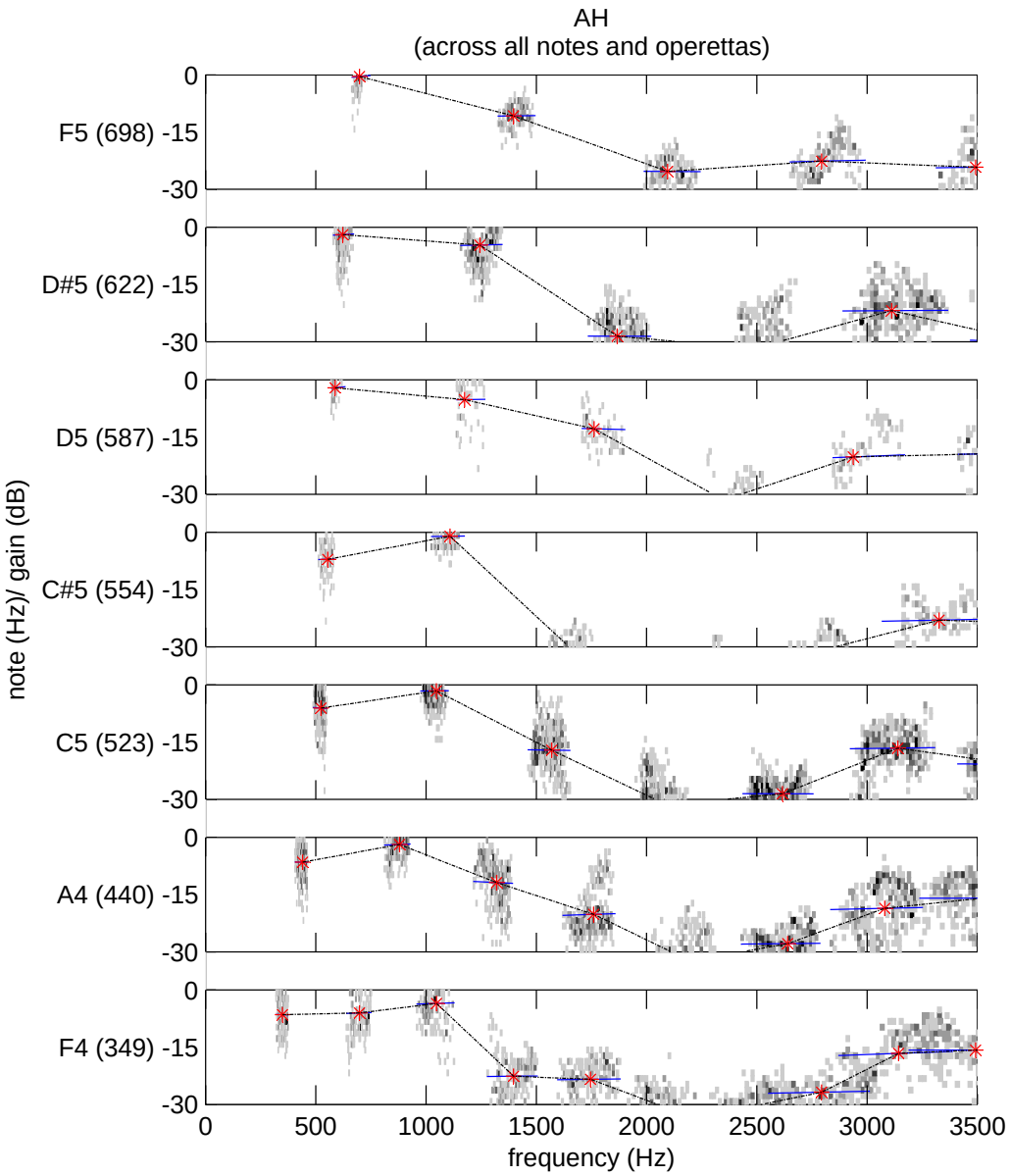

Figure 9.5: AH harmonics across all notes in all operettas. As in figure 9.2, the gains are red stars, the slopes are blue lines, and the histograms show the points used to calculate parameters.

9.4 The vowel IY

The vowel IY (/i/ as in she), in contrast to AA and AH, has a low first formant at about 150 to 450 Hz and a high second formant at about 2000-3700 Hz [40]. In figure 9.6, only the lowest note has even the first harmonic in range of the first formant. So rather than energy shifting between harmonics as the pitch increases, most of the energy stays firmly in the first formant. Interestingly, because the formants are better separated than the formants for AH and AA, the second formant can be seen reasonably clearly.

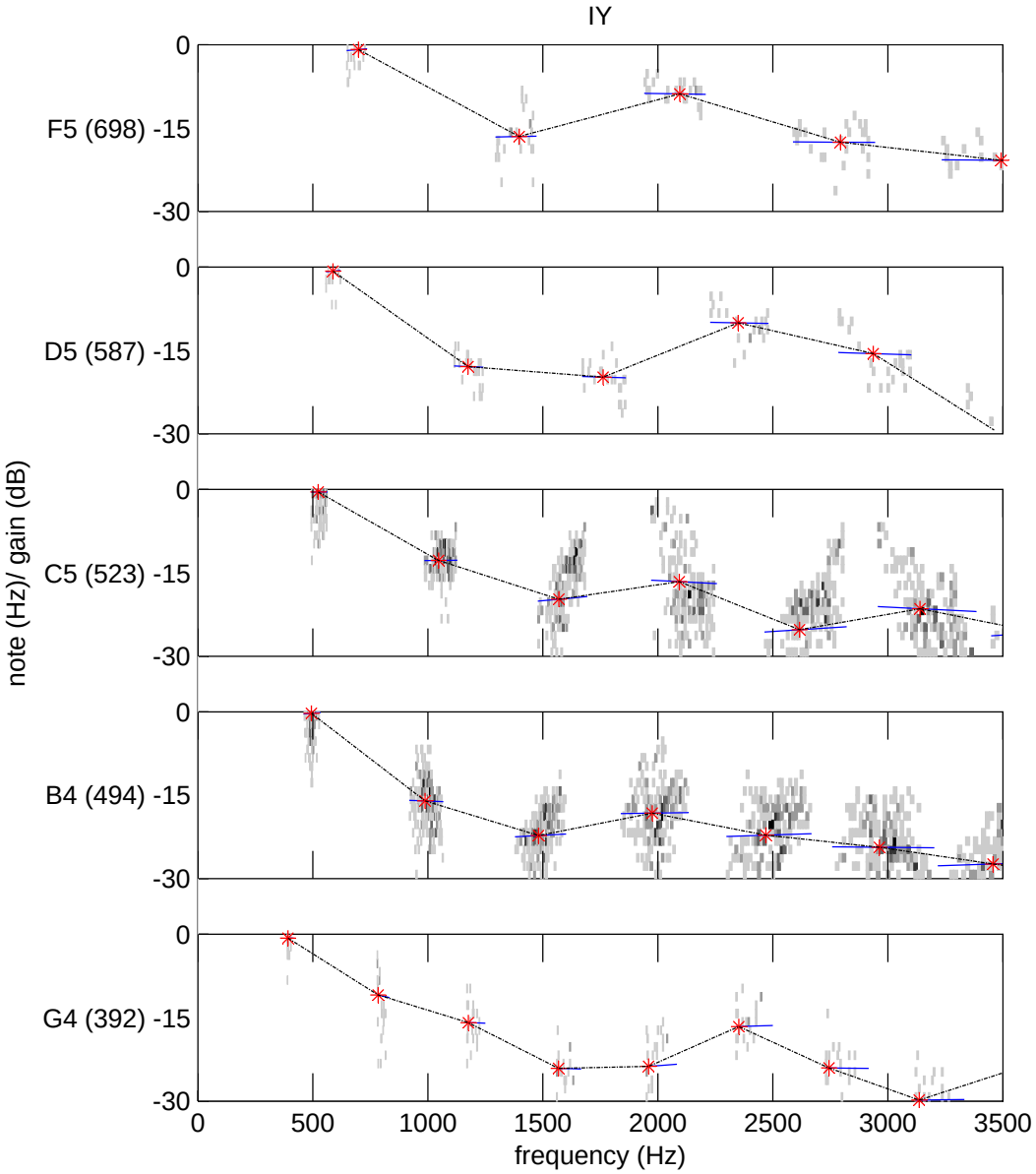

Figure 9.6: IY harmonics across all notes in all operettas. As in figure 9.2, the gains are red stars, the slopes are blue lines, and the histograms show the points used to calculate parameters.

9.5 The singer's formant

Recall from section 2.1.2 that the singer's formant is a peak in the voice near 3000 Hz. Depending on the definition, this peak may or may not apply to high voices. However, for the purposes of this section, the singer's formant will be defined as a peak, or local maximum, in a note's calculated gains (i.e. all gains and slopes will be calculated across individual notes in this section) between 2500 and 3500 Hz that is at least -20 dB. Figure 9.7 shows an example of a set of gains from one note that fits this definition (top) and an example of a set of gains that does not (bottom).

Each of the singers shows some evidence of a singer's formant (table 9.5), but different singer's employ the singer's formant with quite different frequency. The soprano from from *The Gondoliers* has evidence for the singer's formant in nearly four-fifths of her notes, while the soprano in *H.M.S. Pinafore* only uses the singer's formant in a third of notes.

Now, some vowels do have a second formant in the region of the singer's formant. Given the small size of the data set, it is possible that the differences between the singers are simply a function of the vowels they happen to be singing. This is a little difficult to assess because the BEEP dictionary used to generate phones for the lyrics is in Arpabet, but publications which specify vowel formants tend to use the International Phonetic Alphabet (i.e. [30, 40]) and sometimes translation between the two is not completely straightforward. Furthermore, many of the vowels are diphthongs. Caveats aside, it does not seem to be the case that the appearance of the singer's formant in these voices is completely tied to the vowels being sung. Sopranos sing notes without the singer's formant on vowels with a high second formant such as IH (/ɪ/ as in hid) and they sing notes with the singer's formant on vowels with a low second formant such as AH. So the use of the singer's formant seems more tied to the singer than the vowels the singers are performing.

Operetta	% of notes with singer's formant
H.M.S. Pinafore	33
The Mikado	67
The Pirates of Penzance	55
The Gondoliers	79
The Yeomen of the Guard	42

Table 9.5: Percentage of notes with evidence of the singer's formant for each Operetta.

Recall from section 2.1.2 that the purpose of the singer's formant is to help the singer to be heard over the orchestra, which has limited energy around 3000 Hz. Traditionally, the formant has been associated with lower voices and lower pitches. So it is interesting to take a look at the distribution of notes with and without the singers formant (figure 9.8). Interestingly, there is not much difference between the distribution of note frequencies with the singer's formant and without. So, this data set would tend to agree with Barnes and Davis's work [6] that use of the singer's formant is tied to singers and not to pitch. It also seems to be in direct conflict with Bloothooft and Plomp's work [10], which found no evidence of the singer's formant for notes whose fundamental frequency was higher than 392 Hz, although their definition of the singer's formant was somewhat different from the one used here.

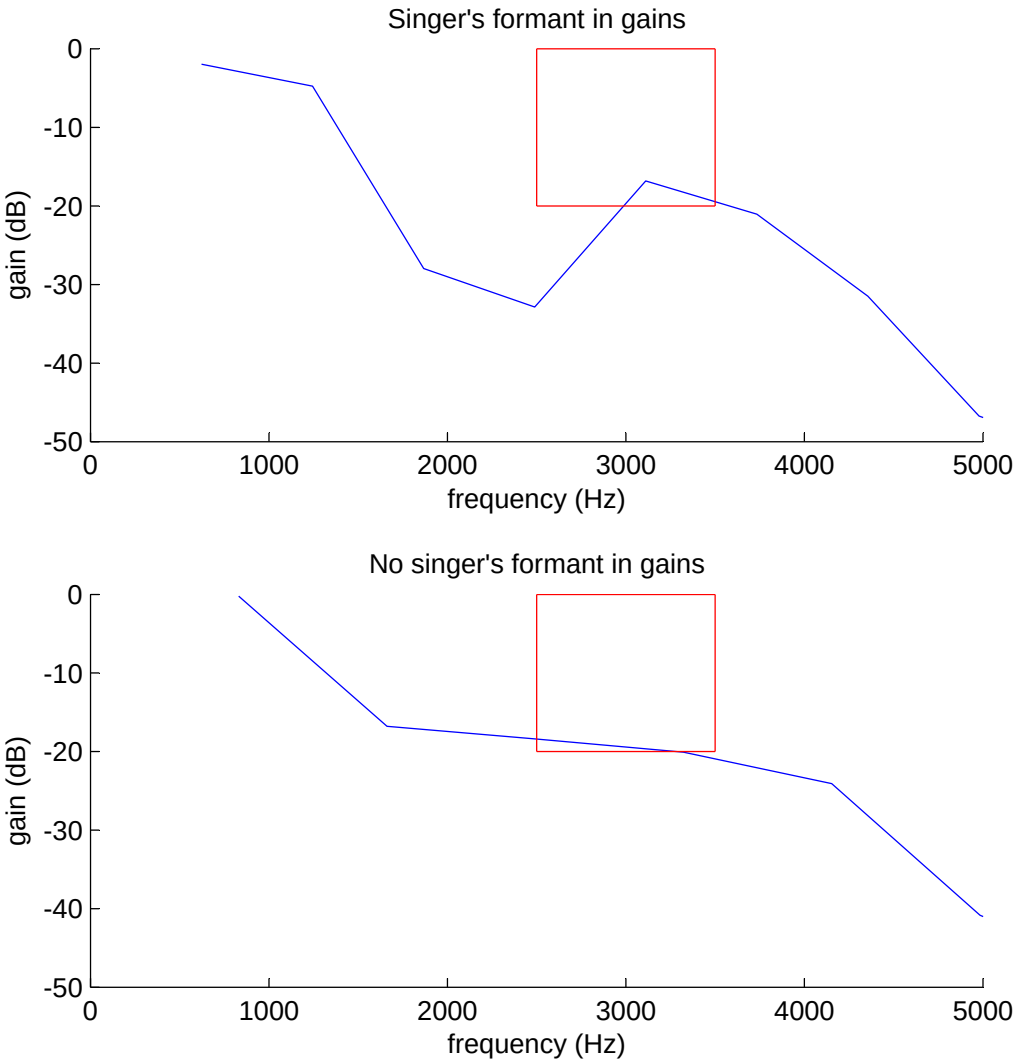

Figure 9.7: Singer's formant examples. Both plots show the gains associate with notes from *The Pirates of Penzance*. The red boxes outline the region where there needs to be a peak in the gains in order for the note to have the singer's formant. The top plot shows the gains associated with single note that shows evidence of the singer's formant and the bottom plot shows gains associated with a single note that does not.

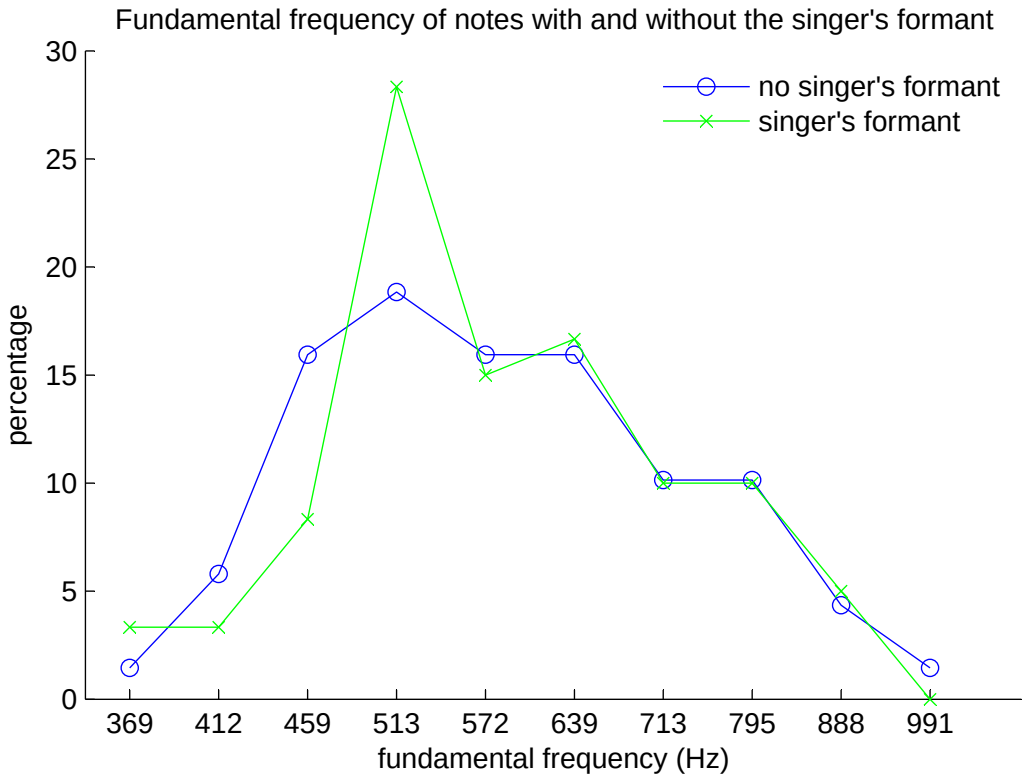

Figure 9.8: Histograms of notes with and without the singer's formant. The percentages are relative to the individual groups. In other words, just over 25% of the notes with singer's formant have a fundamental frequency around 513 Hz.

9.6 Discussion

The Gilbert and Sullivan data set is relatively small, particularly once all the short notes and notes without vibrato are removed. Ideal data would cover the $\{singer, notefrequency, vowel\}$ triple more completely and in more depth. This would obviously allow more exploration of how the harmonic gains change as function of singer, vowel, and note frequency.

However, as it stands, the Gilbert and Sullivan data set is large enough to illustrate the power of the algorithms described in previous chapters to meaningfully characterize voices pulled from a polyphonic mix.

9.7 Summary

This chapter brings together a group of algorithms described in chapters 3 and 5 to 8 to analyze a set of artistic recordings. This tool set first uses the aligned MIDI to estimate the exact pitch tracks of each note in the data set. Raw frame-based harmonic amplitudes are then estimated using these pitch tracks. Parts of the pitch tracks/harmonic amplitude tracks are then automatically discarded if the pitch track is not sufficiently smooth or if the pitch track is lacking vibrato. The tool set then goes back to the MIDI files and translates syllable-aligned lyrics into syllable-aligned phones, from which each note's vowel can be easily extracted. With the vowel information in hand, the tool set can then smooth the raw amplitudes across a single vowel and nominal note frequency combination.

These smoothed estimates essentially consist of a gain for each harmonic and a local slope around each harmonic that is swept out by the vibrato. While the harmonic gains are reasonably estimated from the data, the slopes are more suspect due to the spread of the data.

Several vowels are examined for how the vowel shape changes as pitch changes. Two of the vowels have relatively high first formants, meaning that it is possible to see the energy associated with the first formant move to lower-numbered harmonics as the pitch increases. The last vowel has a low first formant, so the first harmonic always contains the most energy.

Finally, the singer's formant is examined. Evidence is found for a singer's formant in all five sopranos in the data set, but some singers seem to use the singer's formant much more frequently than the others do. This increase in the prevalence of the singer's formant cannot be explained away by high second formants in the particular vowels the singers perform, suggesting that the use of the formant really is tied to the individual soprano's technique. Furthermore, even though the singer's formant has traditionally been associated with lower pitches, in part because of the way it was defined by Sundberg (section 2.1.2), the evidence from this data set suggests that its use is not necessarily tied to pitch.

Chapter 10

Conclusions

10.1 Thesis summary

This thesis has explored the problem of extracting the vocal characteristics of singers in commercial recordings. The initial approach was to look for unobstructed views of instruments in mostly polyphonic mixes by trying to optimally cancel energy in each window using a modified comb filter. If the window only contained a single harmonic voice, the comb filter was generally able to cancel out most of the energy in the window. Otherwise, the ratio of original to residual energy was much higher because the filter was unable to remove the majority of the energy. This approach worked reasonably well, but monophony is simply not very common in recordings. A large database of music would have been needed to extract enough data for further processing. There was also the problem that this algorithm only identified sections with harmonic tones, not the instrument that created the tones. So further processing would have been required to find just the vocal passages.

The final approach was to tackle polyphony directly. Unfortunately, polyphony without constraints is very complex problem. If nothing is known about the number

of voices, the kinds of instruments, or the musical style, then it can be difficult to constrain the problem of locating the voices sufficiently for an efficient algorithm. So this thesis approached the problem with extremely good constraints in the form of an aligned electronic score. This meant that the algorithm knew exactly when each note for the voice should appear and approximately what its pitch should be. The task now was simply to home in on the exact pitch of the voice. With the exact pitch known, the pitches of all the harmonics could be calculated and the harmonic strengths estimated.

Two approaches were taken to pitch estimation, both of which took advantage of the presence of multiple harmonics to improve the fundamental frequency estimate. The first approach was Bayesian. It attempted to find the MAP estimate of not only the fundamental frequency, but also the amplitudes and phases of the harmonics in each short window. The second approach was a much simpler template matching scheme which estimated the fundamental frequency by sliding a harmonically-shaped template over the DFT of each window. While the Bayesian solution was both more mathematically elegant and more accurate, it was extremely slow. The template matching scheme was reasonably accurate and was also at least an order of magnitude faster, which seemed like an acceptable trade off.

Once the fundamental frequency was known in each window, the amplitudes of the harmonics were estimated from the DFT coefficients nearest to each harmonic. Because the estimates were made from so few data points, they were rather noisy. The solution to this local noise was to smooth the estimates out over time. First however, the algorithms needed to know which data to use during smoothing. There was no point in smoothing data calculated from erroneous fundamental frequency estimates. So, sections of the frequency tracks were first labeled ‘good’ or ‘bad’ based on their smoothness and whether or not they contained vibrato. Only amplitudes

calculated with good frequency information were then smoothed across time.

Two models were tested for smoothing the amplitudes, both of which essentially assumed that the harmonic amplitudes at any moment were a function of an underlying 'breath' energy curve, a harmonic gain tied to the vocal tract filter, and a local slope due to fluctuations in the frequency from the vibrato. These models were shown to successfully estimate the underlying harmonic gain profiles.

10.2 Future directions

There are several obvious limitations to the approach taken in this thesis. The approach relies heavily on an accurate score. For some kinds of music, particularly classical music where musicians work from a score, this may be relatively easy. For other kinds of music which rely more on improvisation, finding a score, particularly one accurate enough for this algorithm to use, may be very difficult. However, even a partial score may be useful. At a minimum, it is not hard to imagine an algorithm that would be able to at least automatically figure out which pieces of the score are accurate and which are not.

A second limitation is the requirement that the score be hand-aligned with the recording in order to have accurate timing information. Fortunately, this limitation should go away as score-audio alignment algorithms improve.

A more fundamental limitation is that the voice being analyzed needs to be prominent. If the desired voice is not the loudest harmonic voice in a particular time/frequency block, this algorithm is liable to latch onto the wrong voice. In very dense textures with equal voices, such as quartets, this can be a definite problem. The algorithm currently works much better pulling out a solo in the solo + accompaniment case because the solo should always be the most prominent voice. One advantage of the current approach is that it does not need prior models of the voice being

pulled out, but perhaps in situations where the texture is more dense, some kind of voice-specific model would be better at differentiating between closely packed voices.

Of course, one advantage of this harmonic-basic approach is that the frequency and initial harmonic estimates are completely agnostic about what sort of instrument is creating the harmonic tone. So the frequency and initial harmonic estimates could be applied to any instrument without modification. While some sort of smoothing will still be necessary for the harmonics, the voice models presented should also apply reasonably well to instruments which also have some kind of periodic source and which also filter this source through the body of the instrument. The only part of the whole algorithm that may need to be reconsidered is the reliance on finding vibrato to label frequency track data as 'good.'

Another possible direction for investigation is singer identification. The harmonic gains provide a reasonably compact representation of how a singer produces various vowels on various notes. The singer could be represented in terms of combinations of these gains. Given that many notes that are nearly the same frequency will have similar harmonic gains, the dimensionality of the combined representation could undoubtedly be reduced. The key will be to figure out how to reduce the dimensions to emphasize the differences between singers rather than merely the similarity between notes from the same singer.

Finally, it would nice to somehow combine the speed of the template algorithm and the accuracy of the Bayesian algorithm during the frequency estimation phase. An easy starting point would be using the template algorithm's frequency estimate as the initial frequency estimate for the Bayes algorithm maximization. Since the template algorithm's frequency estimate should be closer to the actual frequency than the current initial estimate, the maximization algorithm should require fewer iterations to converge. It may also be faster to maximize the individual parameters

round-robin style rather than doing the joint maximization all at once.

10.3 Potential applications

The methods outlined in this thesis have a variety of possible applications. At a most basic level, they provide a way of creating a well-labeled data set that can be used for developing other kinds of algorithms.

Source separation is the problem of dividing a mix into its component parts. This thesis provides a way of getting extremely good information about some of those parts. If the algorithm is aimed at the lead voice, then it produces a frequency track for this voice and harmonic amplitudes across time. The frequency track could be used to carefully filter out the voice energy as much as possibly from the background. The frequency track and harmonic amplitudes, along with an artificial phase, could be used to reconstruct at least the voiced parts of the lead vocals with reasonable fidelity.

Other possibilities lie in the realm of remixing. Traditionally, remixing starts with the old multi-track recordings, from which the new mix is created. Sometimes, however, the old multi-tracks are gone. Sometimes, even if they exist, the individual multi-tracks contain multiple lines anyway because the old multi-track technology limited artists to a few tracks. This research offers the potential to do some remixing directly from when tracks for individual instruments are unavailable.

Musicology research is also an obvious application. Performance practice, the study of aspects of performing that are not necessarily notated in the music, could benefit from being able to accurately calculate these aspects of the performance directly from recordings. For example, accurate frequency tracks could allow for the exploration of subtle tuning effects. The distribution of energy among harmonics could allow musicologists to explore tone color changes.

In short, the algorithms presented in this thesis have potential applications to a variety of interesting problems in music information retrieval.

Bibliography

[1] Gilbert and sullivan archive home page. http://math.boisestate.edu/gas/. URL `http://math.boisestate.edu/gas/`. (Cited on page 17.)

[2] I. Arroabarren, M. Zivanovic, X. Rodet, and A. Carlosena. Instantaneous frequency and amplitude of vibrato in singing voice. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 5, 2003. ISBN 0780376633. (Cited on page 10.)

[3] MIDI Manufacturers Association. Tech specs & info, 2010. URL `http://www.midi.org/techspecs/index.php`. Accessed on 2010-11-01. (Cited on page 13.)

[4] B. Atal and M. Schroeder. Predictive coding of speech signals and subjective error criteria. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(3):247–254, 1979. (Cited on page 28.)

[5] B. S Atal and M. R Schroeder. Adaptive predictive coding of speech signals. *Bell System Technical Journal*, 49(8):1973b•"1986, 1970. (Cited on page 11.)

[6] J. J Barnes, P. Davis, J. Oates, and J. Chapman. The relationship between professional operatic soprano voice and high range spectral energy. *Acoustical Society of America Journal*, 116(1):530–538, July 2004. doi: 10.1121/1.1710505. (Cited on pages 9 and 142.)

[7] Wilmer T. Bartholomew. A physical definition of 'good voice-quality' in the male voice. *The Journal of the Acoustical Society of America*, 6(1):25, 1934. ISSN 00014966. doi: 10.1121/1.1915685. (Cited on page 9.)

[8] A.L. Berenzweig and D.P.W. Ellis. Locating singing voice segments within music signals. In *Proc. IEEE Workshop on Apps. of Sig. Proc. to Audio and Acous.*, pages 119–122, Mohonk, NY, October 2001. (Cited on pages 26 and 33.)

[9] Nancy Bertin, Roland Badeau, and Emmanuel Vincent. Enforcing harmonicity and smoothness in bayesian Non-Negative matrix factorization applied to polyphonic music transcription. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):538–549, 2010. ISSN 1558-7916. doi: 10.1109/TASL.2010.2041381. (Cited on page 12.)

[10] Gerrit Bloothooft. The sound level of the singer's formant in professional singing. *The Journal of the Acoustical Society of America*, 79(6):2028, 1986. ISSN 00014966. doi: 10.1121/1.393211. (Cited on pages 9 and 142.)

[11] B. Boashash. Estimating and interpreting the instantaneous frequency of a signal. i. fundamentals. *Proceedings of the IEEE*, 80(4):520–538, 1992. ISSN 0018-9219. doi: 10.1109/5.135376. (Cited on page 12.)

[12] Gunilla Carlsson and Johan Sundberg. Formant frequency tuning in singing. *Journal of Voice*, 6(3):256–260, 1992. ISSN 0892-1997. doi: 10.1016/S0892-1997(05)80150-X. (Cited on page 8.)

[13] K.W. Chan and H.C. So. Accurate frequency estimation for real harmonic sinusoids. *Signal Processing Letters, IEEE*, 11(7):609–612, 2004. ISSN 1070-9908. doi: {10.1109/LSP.2004.830115}. (Cited on page 12.)

[14] Perry R. Cook. SPASM, a Real-Time vocal tract physical model controller; and singer, the companion software synthesis system. *Computer Music Journal*, 17 (1):30–44, April 1993. ISSN 01489267. doi: 10.2307/3680568. URL `http://www.jstor.org/stable/3680568`. ArticleType: research-article / Full publication date: Spring, 1993 / Copyright B• 1993 The MIT Press. (Cited on page 10.)

[15] Roger B. Dannenberg and Christopher Raphael. Music score alignment and computer accompaniment. *Commun. ACM*, 49(8):38–43, 2006. doi: 10.1145/1145287.1145311. (Cited on page 15.)

[16] Alain de Cheveigne and Hideki Kawahara. YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4):1917–1930, April 2002. doi: 10.1121/1.1458024. (Cited on pages 12 and 49.)

[17] Johanna Devaney and Daniel P.W. Ellis. An empirical approach to studying intonation tendencies in polyphonic vocal performances. *Journal of Interdisciplinary Music Studies*, 2(1-2):141–156, 2008. (Cited on page 49.)

[18] Johanna Devaney, Michael I. Mandel, and Daniel P.W. Ellis. Improving MIDI-audio alignment with acoustic features. In *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 45–48, New Paltz, NY, USA, 2009. doi: 10.1109/ASPAA.2009.5346500. (Cited on page 15.)

[19] Mark Dolson. The phase vocoder: A tutorial. *Computer Music Journal*, 10(4): 14–27, 1986. ISSN 01489267. ArticleType: research-article / Full publication date: Winter, 1986 / Copyright B• 1986 The MIT Press. (Cited on page 12.)

[20] Tuomas Eerola and Petri Toiviainen. *MIDI Toolbox: MATLAB Tools for Music Research*. University of Jyväskylä, Jyväskylä, Finland, 2004. URL `www.jyu.fi/musica/miditoolbox/`. (Cited on page 14.)

[21] Sebastian Ewert, Meinard Müller, and Roger B. Dannenberg. Towards reliable partial music alignments using multiple synchronization strategies. In *Proceedings of the International Workshop on Adaptive Multimedia Retrieval (AMR)*, Madrid, Spain, September 2009. (Cited on page 15.)

[22] Gunnar Fant. *Acoustic Theory of Speech Production with Calculations Based on X-Ray Studies of Russian Articulations*. Mouton, The Hague, 2d ed. edition, 1970. (Cited on page 6.)

[23] J. L. Flanagan and L. Cherry. Excitation of Vocal-Tract synthesizers. *The Journal of the Acoustical Society of America*, 45(3):764–769, March 1969. doi: 10.1121/1.1911461. URL `http://link.aip.org/link/?JAS/45/764/1`. (Cited on page 10.)

[24] T. Higashimoto and H. Sawada. Speech production by a mechanical model: construction of a vocal tract and its control by neural network. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 4, page 3858b•"3863, 2002. ISBN 0780372727. (Cited on page 11.)

[25] Elodie Joliveau, John Smith, and Joe Wolfe. Vocal tract resonances in singing: The soprano voice. *Acoustical Society of America Journal*, 116(4):2434–2439, 2004. doi: 10.1121/1.1791717. (Cited on page 8.)

[26] Y. E Kim and B. Whitman. Singer identification in popular music recordings using voice coding features. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, pages 13–17, 2002. (Cited on page 28.)

[27] Youngmoo E. Kim. Singing voice analysis, synthesis, and modeling. In David Havelock, Sonoko Kuwano, and Michael Vorländer, editors, *Handbook of Signal Processing in Acoustics*, pages 259–374. Springer New York, 2008. (Cited on pages 10 and 11.)

[28] M. Kob. Singing voice modelling as we know it today. *Acta Acustica United with Acustica*, 90(4):649–661, 2004. ISSN 1610-1928. (Cited on page 10.)

[29] P. Ladefoged and K. Johnson. *A course in phonetics*, volume 127. Harcourt Brace Jovanovich New York., 1975. (Cited on page 1.)

[30] P. Ladefoged and I. Maddieson. Vowels of the world's languages. *Phonology: critical concepts*, 2001. (Cited on page 141.)

[31] Jeffrey C. Lagarias, James A. Reeds, Margaret H. Wright, and Paul E. Wright. Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM J. on Optimization*, 9(1):112–147, 1998. (Cited on pages 44 and 80.)

[32] Beth Logan. Mel frequency cepstral coefficients for music modeling,. In *Proc. International Symposium on Music Information Retrieval*, Plymouth, 2000. (Cited on page 33.)

[33] S. Matteson and F. L Lu. Vocal inharmonicity analysis: A promising approach for acoustic screening for dysphonia. *The Journal of the Acoustical Society of America*, 125(4):2638, 2009. (Cited on page 11.)

[34] Maureen Mellody, Freda Herseth, and Gregory H. Wakefield. Modal distribution analysis, synthesis, and perception of a soprano's sung vowels. *Journal of Voice*, 15(4):469–482, December 2001. ISSN 0892-1997. doi: 10.1016/S0892-1997(01)00047-9. (Cited on pages 9, 10, and 93.)

[35] Dayton Clarence Miller. *The science of musical sounds*. The Macmillan company, 1916. (Cited on page 12.)

[36] D.G. Miller and H.K. Schutte. Formant tuning in a professional baritone. *Journal of Voice*, 4(3):231–237, 1990. ISSN 0892-1997. doi: 10.1016/S0892-1997(05)80018-9. (Cited on page 8.)

[37] Kevin Murphy. Hidden markov model (HMM) toolbox for matlab. Downloadable software, 2005. URL `http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html`. (Cited on page 31.)

[38] Ian Nabney. *NETLAB: Algorithms for Pattern Recognition*. Springer-Verlag London Ltd, London, 2004. URL `http://www.ncrg.aston.ac.uk/netlab/`. (Cited on page 33.)

[39] A. Ozerov, P. Philippe, F. Bimbot, and R. Gribonval. Adaptation of bayesian models for Single-Channel source separation and its application to Voice/Music separation in popular songs. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(5):1564–1578, 2007. ISSN 1558-7916. doi: 10.1109/TASL.2007.899291. (Cited on page 12.)

[40] Gordon E. Peterson and Harold L. Barney. Control methods used in a study of the vowels. *The Journal of the Acoustical Society of America*, 24(2):175–184, March 1952. doi: 10.1121/1.1906875. (Cited on pages 7, 133, 137, 139, and 141.)

[41] M. D. Plumbley, S. A. Abdallah, J. P. Bello, M. E. Davies, G. Monti, and M. B. Sandler. Automatic music transcription and audio source separation. *Cybernetics and Systems: An International Journal*, 33(6):603, 2002. ISSN 0196-9722. doi: 10.1080/01969720290040777. (Cited on page 13.)

[42] William Putnam and Julius Smith. Design of fractional delay filters using convex optimization. *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Oct. 1997. (Cited on page 28.)

[43] Christine Smit and Daniel P.W. Ellis. Solo voice detection via optimal cancellation. In *Applications of Signal Processing to Audio and Acoustics, 2007 IEEE Workshop on*, pages 207–210, 2007. doi: 10.1109/ASPAA.2007.4393045. (Cited on page 25.)

[44] Christine Smit and Daniel P.W. Ellis. Guided harmonic sinusoid estimation in a multi-pitch environment. In *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 41–44, New Paltz, NY, USA, 2009. doi: 10.1109/ASPAA.2009.5346460. (Cited on page 39.)

[45] J. Sundberg. Vibrato and vowel identification. *Arch. Acoust*, 2:257–266, 1977. (Cited on page 10.)

[46] J Sundberg. Level and center frequency of the singer's formant. *Journal of Voice: Official Journal of the Voice Foundation*, 15(2):176–186, June 2001. ISSN 0892-1997. doi: 10.1016/S0892-1997(01)00019-4. PMID: 11411472. (Cited on page 54.)

[47] J. Sundberg et al. *The science of the singing voice*, volume 1. Northern Illinois University Press Dekalb, IL, 1987. (Cited on pages 6 and 7.)

[48] Johan Sundberg. Articulatory interpretation of the 'singing formant'. *The Journal of the Acoustical Society of America*, 55(4):838, 1974. ISSN 00014966. doi: 10.1121/1.1914609. (Cited on page 9.)

[49] Johan Sundberg. Acoustic and psychoacoustic aspects of vocal vibrato. In *Vibrato*, pages 35–62. Singular Pub., 1995. ISBN 9781565931466. (Cited on pages 9 and 10.)

[50] Johan Sundberg and JG'Aorgen Skoog. Dependence of jaw opening on pitch and vowel in singers. *Journal of Voice*, 11(3):301–306, September 1997. ISSN 0892-1997. doi: 10.1016/S0892-1997(97)80008-2. (Cited on page 8.)

[51] Ingo R. Titze. The human vocal cords: A mathematical model. *Phonetica*, 28(3-4):129–170, 1973. ISSN 1423-0321. doi: 10.1159/000259453. URL `http://content.karger.com/ProdukteDB/produkte.asp?Aktion=ShowAbstract&ArtikelNr=259453&Ausgabe=249786&ProduktNr=224275`. (Cited on page 10.)

[52] R. J Turetsky and D. P. Ellis. Force-aligning MIDI syntheses for polyphonic music transcription generation. *Proc. ISMIR, Baltimore, USA*, 2003. (Cited on page 15.)

[53] E. Vincent and M.D. Plumbley. Low Bit-Rate object coding of musical audio using bayesian harmonic models. *Audio, Speech, and Language Processing,*

IEEE Transactions on, 15(4):1273–1282, 2007. ISSN 1558-7916. doi: {10.1109/TASL.2006.889792}. (Cited on page 12.)

[54] Robert W. Young. Inharmonicity of plain wire piano strings. *The Journal of the Acoustical Society of America*, 24(3):267, 1952. ISSN 00014966. doi: 10.1121/1.1906888. (Cited on page 11.)