FUNDAMENTAL FREQUENCY TRACKING IN MUSIC WITH MULTIPLE VOICES

BY

GEOFFREY LINDSAY HERMAN

B.S., University of Illinois at Urbana-Champaign, 2005

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2007

Urbana, Illinois

ACKNOWLEDGMENTS

First and foremost, I want to thank God for His provision and enabling. He provided me with all of the wisdom, guidance, and encouragement I needed for this thesis, and all that I have is His.

An enormous debt of thanks is also owed to my adviser James W. Beauchamp for all of his direction and help in the research and writing of this thesis. Without his advice and guidance this thesis would never have been completed. His experience greatly improved the quality of this thesis.

Many thanks to Mert Bay and Andreas Ehmann for sharing their extensive experience in music information retrieval with me, and for directing me to sources that have been critical for my research.

Thanks also to my doctoral advisers Michael C. Loui and Craig Zilles as they have waited patiently for me to finish this thesis and have provided insightful feedback on my work. Their support, guidance, and encouragement have been invaluable.

I also want to thank my friends in Grad Cru who have supported me with prayer, technical support with LaTex, and have helped kept me motivated.

Finally, I want to thank my family for all of their care, encouragement, and prayers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Research on fundamental frequency (F0) estimation, also known as pitch detection, is motivated by many different applications. F0 estimation can facilitate the classification and analysis of speech and music for cataloging and searches. Also, for speech, automatic speech recognition programs need F0 estimation to transcribe tonal languages and to process biometrics for speaker identification. For music, automatic music transcription applications use F0 estimation to simplify the difficulties of composing or analyzing music. It is also hoped that F0-estimation-based query-by-humming search engines will give users the ability to search for songs simply by humming portions of the tunes [1].

This thesis focuses on the use of F0 estimation to transcribe music. When developing an F0 estimator, music transcription tasks provide many advantages over the other applications mentioned above. First, musical scores, when available, supply approximate pitch and timing information which help to provide ground truth data for the evaluation of F0 estimation algorithms. Because speech applications lack this a priori approximate pitch and timing information, it is more difficult to compare the performance of different F0 tracking algorithms being developed [2]. Second, musical sound sources provide a wide range of timbres (spectral envelopes) to test the robustness of an algorithm. Despite the variety of musical instrument spectra, they share similar patterns (a.k.a. harmonic spectra). These patterns limit the computational expense of musically focused F0 estimation algorithms. Finally, professional and student musicians are trained to identify the pitch of musical instruments, whereas few

people are trained to identify pitch for any other application. Pitch analysis performed by musicians to refine ground truth data facilitates evaluation of F0 algorithms.

Chapter 2 of this thesis discusses the definition of pitch and frequency as well as other features of sound. Chapter 3 surveys different methods used for monophonic F0 estimators, and Chapter 4 discusses the strengths and weaknesses of the monophonic algorithms when used in polyphonic environments.

Chapter 5 discusses strategies for estimating multiple F0s in polyphonic sounds and then presents the algorithms developed by the author. In particular, the concept of antisalience, a new error correction technique developed to improve the performance of one of the F0 estimators, is presented. It was found that antisalience reduced the F0 estimator's rate of errors by over 65% in monophonic applications and as much as 45% in polyphonic applications. The performance of each of the algorithms is analyzed and compared in Chapter 6. Trade-offs between computational efficiency and accuracy, tested via a tunable parameter built into the F0 estimator, will also be discussed.

CHAPTER 2

KEY TERMS AND CONCEPTS

2.1 Introduction

Unfortunately, musicians, acousticians, and engineers as well as lay persons often attribute different meanings to the terms and measurements used to describe sound. Because of this confusion in terms, most terms used throughout this thesis will be given specific definitions. Measurements of sound can be classified into two categories: physical variables and perceptual variables. Physical variables of sound are derived from physical characteristics of sound waves, such as frequency [Hz] and sound pressure [N/m^2], that can be measured using laboratory equipment. Perceptual variables of sound (e.g., pitch, loudness, and timbre) are harder to define and are even harder to measure, because they are based on human perception instead of laboratory measurements. Perhaps the best researched and understood perceptual variable is loudness, which has multiple units of measure (e.g., sones and phons) and can be derived from the pressure and frequency content of a sound [3]. Not all perceptual variables are so well defined, and some perceptual variables, such as timbre, cannot be measured by any quantitative metric yet. Regardless of the accuracy of the units of measure, perceptual variables can only be approximations of what is truly being perceived, because perception can vary slightly every time a sound is perceived.

According to the ANSI standard (1973) definition [4], "Pitch is that attribute of auditory sensation in terms of which sounds may be ordered on a scale extending from low to high. Pitch depends mainly on the frequency content of the sound stimulus,

but it also depends on the sound pressure and the waveform of the stimulus." Pitch has several distinct units of measure to describe this low to high scale, but there is still disagreement about the mechanisms by which humans perceive pitch and the methodology for measuring perceived pitch [5]. Hartmann provides an in depth discussion of the competing theories of pitch perception [5], but for this thesis only different measures of pitch will be discussed.

The best known measure used to describe pitch is provided by music. This measure follows the convention of assigning pitch a letter, a possible accidental, and an octave number (e.g., A_4, $C^{\sharp}_3$, G_5). Music notation is convenient because sounds with the same letter have a very similar perceptual quality, and larger numbers correspond to higher pitches.

However, music notation is too coarse for many applications, and other measures of pitch have been invented to compensate for this shortcoming. Alexander Ellis invented a relative pitch scale which divided the musical octave into 1200 divisions called cents (e.g., A_3 and A_4 are 1200 cents apart) [6]. Stevens and Volkman invented the perceptually inspired mel frequency scale which quantified pitch based not only upon frequency but also on intensity [7]. In 1983 an absolute scale of pitch was invented as part of a synthesizer standard known as the Musical Instrument Digital Interface (MIDI). The original MIDI standard assigned the pitch of each musical tone to an integer (e.g., A_4 at 440 Hz is MIDI note number 69). Because MIDI pitch correlates so well with musical notation and because it is a numerical scale rather than an alphanumeric scale based strictly on F0, MIDI pitch (a.k.a. note number) has become the popular data standard for representing pitch information in music analysis and will be the pitch scale used for this thesis. MIDI pitch can also be modified so that real numbers are used instead of just integers. This continuous scale allows for finer resolution when displaying perceived pitch. Conversion from an

equivalent F0 to MIDI pitch is calculated as follows [8]:

$$MIDIPitch = 69 + 12log_2\left(\frac{F0}{440}\right). \tag{2.1}$$

MIDI pitch and Cents approximate human pitch perception with a base-two logarithmic relationship which is valid for typical sounds found in music. This approximation corresponds perfectly with equal-tempered tuning, which dictates that frequencies which are an octave apart must have a 2:1 ratio and frequencies which are a semitone apart must have a $2^{1/12}$:1 ratio.

2.2 Beating

The pitch of a single sine wave, also known as a simple tone, is typically directly related to its frequency. When two simple tones are played simultaneously, a number of different pitch sensations can occur. If the difference in frequency between the two tones is small, the two tones will sound like one pitch with a strong beating sensation. If the difference in frequency is large, it is possible that two simple tones will be perceived with little or no beating. If the two simple tones share a common divisor, it is also possible that a single complex tone will be heard. When two sine waves of nearly equal frequency are present in a signal, they will alternately constructively complement each other and deconstructively cancel each other at a rate equal to the difference between their frequencies. The instantaneous amplitude a_s of the composite partial (peak in the spectrum), known as a *beating partial*, produced by the two simple tones is

$$a_s = |a_1 + a_2 e^{j2\pi\Delta ft}| = \sqrt{a_1^2 + a_2^2 + 2a_1a_2\cos(2\pi\Delta ft)}, \tag{2.2}$$

where a_1 and a_2 are the amplitudes of the two simple tones, and $2\pi\Delta ft$ is the instantaneous difference in phase between the two simple tones. The effect of beating

can easily be seen when looking at the evolution of amplitudes in a short time fourier transform (STFT). As seen in Fig. 2.1, if two partials are close in frequency, then a single partial will be seen in each frame of the STFT with an amplitude that increases and decreases in accordance to Eq. (2.2).

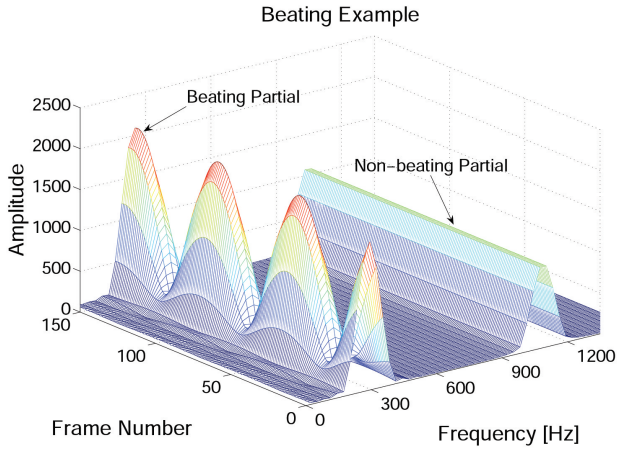

Figure 2.1 Example of a beating partial at 300 Hz.

The occurrence of beating partials challenges most F0 estimators, because beating violates some of their fundamental assumptions. Most F0 estimation algorithms assume that the spectrum of a musical or speech signal changes slowly with respect to analysis window size. Beating, which can cause rapid amplitude fluctuations from frame to frame of the STFT, can violate this assumption and make estimation of an F0 more difficult. If partials in an F0's spectrum disappear, its harmonic series is seriously modified, adding to the difficulty of F0 estimation. Time-domain based F0 estimators, which rely on detecting the periodicity of a signal, may detect the periodicity of the beating rather than the period of the simple tones. The fluctuation of amplitudes may also obscure any periodicity of the signal.

2.3 Harmonic and Inharmonic Sounds

Few musical and speech signals can be classified as simple sinusoids. Most musical and speech signals with a single discernable pitch are comprised of several partials whose frequencies are integer multiples of a fundamental frequency f_1 (i.e., they are harmonic sounds). Usually the frequency F0 corresponds to the pitch of a harmonic sound and to the short-time average of its fundamental (i.e., F0=$\bar{f}_1$). However there may be pathological cases where this is not true. In these cases we can say that the F0 for a complex tone is defined to be the frequency of a sine wave which has the same perceived pitch. A lower case f with a numerical subscript (i.e., f_1, f_2, f_3,...) corresponds to the harmonic whose frequency is an integer multiple of f_1 (e.g., f_1 is harmonic number one, $f_2 = 2f_1$ is harmonic number two, $f_3 = 3f_1$...).

A sound is classified as being harmonic if the perceived F0 can be derived from the integer relationships of a well-defined harmonic series. Most string, woodwind, and brass instruments as well as sung or spoken vowel sounds can be classified as harmonic and have partials located at all harmonics (f_1, f_2, f_3,...) even up to and beyond f_{20}. A sound can still be classified as a harmonic sound even if one or more partials are missing from its spectrum. A clarinet-like sound consisting of only odd harmonics (f_1, f_3, f_5,...) can be defined as a harmonic sound, because all of its partials are in integer relationships with F0. A sound can also still be classified as harmonic, even if the harmonic series is stretched. For example, the distance between partials of a freely vibrating string instrument's spectrum (e.g., piano or guitar) slowly increases for the higher harmonics. However, enough of the lower partials share an integer relationship with f_1 to justify calling the sound harmonic.

A special class of harmonic sounds occurs when f_1 is missing or is very weak. If f_1 is missing, the sound is described as having a missing fundamental or phantom fundamental. When the harmonic series above f_1 is sufficiently complete, the pitch

will still be perceived as corresponding to f_1. A phantom f_1 will be "heard" by the listener, even though it is absent from the sound.

A sound is classified as being strictly inharmonic if the sound consists of partials that do not correspond to a well-defined series of partials whose frequencies are integer multiples of f_1. Examples of inharmonic sounds are filtered Gaussian noise and bell tones. Filtered Gaussian noise can have a pitch but has no distinct partials. A bell tone has a well-defined partial structure, but its partial frequencies are not integer multiples of any f_1 or any F0. It is considerably more difficult to predict the pitch or lack thereof of inharmonic sounds. Moreover, discerning their pitch is less important in music transcription. Therefore, F0 estimation of inharmonic sounds is not included in the scope of this thesis and is left to future research.

2.4 Consonance and Dissonance

In 50 B.C., Pythagoras researched pitch sensations and what combinations of pitches are pleasing to listen to. While plucking two strings that are made with the same material and have the same tension, he discovered that consonant intervals of pitch (harmonious pitches) can be produced by making the lengths of the two strings to be in small integer ratios. For example, if one string is twice as long as the other, the two strings will sound an octave apart (the most consonant interval). Pythagoras's experiments led to the development of core musical concepts, such as octaves based on the 2:1 ratio in string length and the circle of fifths based on the 3:2 ratio in string length. Scientists later related string tensions, masses, and lengths to frequency, and the concept of consonant intervals of pitch was generalized so that any two sources that produce frequencies in small integer ratios are considered to be consonant. Inversely, pitches with large integer frequency ratios (e.g., 16:15) or noninteger ratios are considered to be dissonant (not harmonious) [9].

Because consonant intervals are pleasing to listen to, they are prevalent in music. The prevalence of consonant intervals complicates multiple F0 estimation tasks, because the harmonic series of two consonant pitches will overlap at many partials. For example, if two sounds are a fifth apart (3:2 ratio in frequency), every third partial of one harmonic series will overlap with every second partial of the other series. These harmonic series overlaps cause numerous problems for F0 estimators and will be discussed in detail in Chapter 5.

CHAPTER 3

OVERVIEW OF MONOPHONIC F0 ESTIMATION ALGORITHMS

3.1 Motivation

The task of monophonic F0 estimation and tracking of harmonic sounds is a problem that has been satisfactorily solved, and numerous commercial and freeware products have been produced which utilize F0 estimators [2]. Several researchers (Rabiner et al. [10], Hess [11], Gerhard [12], and deCheveigné and Kawahara [2]) have analyzed and compared these algorithms for accuracy and speed. Although these researchers have provided excellent performance analyses of the various monophonic case algorithms, none has evaluated the algorithms specifically for extension into polyphonic environments.

Despite the large number of F0 estimation algorithms developed, most of them can be classified into a few families. Early researchers classified F0 estimators into two categories: frequency-domain-based algorithms and time-domain-based algorithms [10]. More recently, Klapuri has classified F0 estimators into three major categories in his PhD dissertation [13]: spectral location type F0 estimators, *spectral interval* estimators and estimators based on perceptual models.

Spectral interval type F0 estimation algorithms estimate F0 using spectral autocorrelation to find the most prevalent *frequency interval* (frequency difference between two partials), which is then estimated to be the F0 [13]. Lahat et al. [14] and Kunieda et al. [15] have both developed accurate F0 estimators using spectral interval methods. Their algorithms are reported to perform well even when the test sounds

exhibit strong inharmonicities.

Most F0 estimators based on perceptual models separate sounds into separate frequency bands or channels to emulate the ear's basilar membrane. The separate channels are then processed according to the designer's psychoacoustic model. Models of how the brain processes these individual channels are still topics of debate, so F0 estimators based on perceptual models vary greatly. Klapuri [13], Tolonen and Karjalainen [16], Meddis and O'Mard [17], and Meddis and Hewitt [18] have developed different algorithms based on different perceptual models.

However, F0 estimators based on perceptual models and spectral interval methods go beyond the scope of this thesis. The remainder of this chapter is dedicated to giving an overview of spectral location F0 estimators. Spectral location F0 estimators were chosen as the focus of this thesis, because they are widely known and have been thoroughly peer evaluated and analyzed in the previous overviews by Rabiner et al. [10], Hess [11], Gerhard [12], and deCheveigné and Kawahara [2].

3.2 Spectral Location Methods

Spectral location F0 estimators use signal processing tools such as autocorrelation and the Fourier transform to find the locations of partials which are then used to estimate F0. The F0 of a sine wave is the frequency of its spectral location in the frequency domain or the time-inverse of the first local maximum of its autocorrelation function. By building on this observation it is reasonable to assume that the F0 of complex tones can also be estimated based on the location of peaks in the frequency domain or local maxima in the autocorrelation function.

Spectral location methods comport well with the place theory of pitch perception [5]. This theory is based on tonotopic mapping (pitch vs. location mapping) afforded by the basilar membrane, which informs the cochlear nucleus about which

neurons are firing and which are silent, much like bins in an FFT. These location data are preserved as neural signals progress through the auditory system up to the auditory cortex. Some phenomena of pitch perception provide strong evidence that pitch depends primarily on the position of basilar membrane resonances [5].

3.2.1 Periodicity analyzers

Spectral location methods can be divided into two subcategories: periodicity analyzers and harmonic pattern matchers. Periodicity analyzers attempt to find the fundamental period ($\tau 0 = 1/\mathrm{F0}$) based on a feature of the sound or a set of features. The most basic periodicity analyzer simply finds and analyzes the zero crossings of a waveform and defines $\tau 0$ to be the time interval between zero crossings with derivatives that have the same sign (interpolation techniques can be used to increase the resolution of F0 candidates). This method works well only for simple, nonnoisy signals, which of course motivates the need for methods of measuring $\tau 0$ that are applicable for complex, noisy signals.

The autocorrelation function (ACF) is capable of analyzing complex, noisy signals and is defined in the time domain as

$$R(\nu) = \sum_{n=0}^{N-1-\nu} x[n]x[n+\nu], \tag{3.1}$$

where $x[n]$ is the observed frame, N is the frame size, and ν is the amount the frame is shifted. For the sake of computational efficiency the ACF is normally calculated as

$$R(\nu) = IFFT\{|FFT[x[n]]|^2\}. \tag{3.2}$$

A sound's $\tau 0$ is found by finding the first peak in the ACF that rises above a set threshold. Because a function is always perfectly correlated with itself, the highest

peak is found at R(0). Therefore, the threshold used to find $\tau 0$ should be set in relation to R(0) to ensure that later shifted versions of the signal are sufficiently similar to the original, unshifted signal. If the threshold is set too high, there is a possibility that none of the peaks in the ACF will rise above the threshold, and the ACF will fail to detect a $\tau 0$. If the threshold is set too low, peaks in the ACF (normally due to the harmonic structure of the sound) that have a period less than the true $\tau 0$ may be erroneously detected as the correct $\tau 0$.

The autocorrelation function is appealing to developers because it is robust against noise, it is computationally inexpensive, and considering its simplicity it is surprisingly accurate. It is commonly one of the first pitch detection methods learned, and yet it is a commercially viable F0 estimator as attested by several papers written on the efficacy of ACF based algorithms [19],[20],[21]. Many of these algorithms can even be used in real-time or quasi-real-time applications.

Noll published a different periodicity analyzer called the cepstrum method [21], defined by

$$C(\nu) = IFFT\{log(|FFT[x[n]]|)\}. \tag{3.3}$$

The computation of the cepstrum is similar to ACF, except for the log compression performed on the magnitude spectrum. While logarithmic compression of the magnitude spectrum causes cepstrum-based methods to be more robust against inharmonicities and other features such as speech signal formant structures, the added robustness against inharmonicities comes at the cost of lower robustness against noise [13].

Tolonen and Karjalainen [16] examined the subtle difference in operands and some tradeoffs between the ACF and cepstrum methods and later derived a generalized autocorrelation function. It was found that the square in Eq. (3.2) can be replaced with any real exponent. Tweaking the exponent allows the user to make the function either more robust against noise or more robust against inharmonicities.

Another periodicity analyzer of note is the Average Magnitude Difference Function (AMDF)

$$d(\tau) = \sum_{n=0}^{N-1-\tau} (x'[n] - x'[n+\tau])^2, \tag{3.4}$$

where $\tau 0$ is estimated as the global nonzero minimum of $d(\tau)$. deCheivegné and Kawahara developed a specific method derived from the AMDF and the ACF known as the YIN method [22]. The AMDF calculates the magnitude of the difference between a signal and a shifted version of itself, similarly to the ACF. It is particularly noteworthy that the YIN method outperforms the basic ACF and AMDF algorithms in most applications. When using periodicity analyzers to estimate F0, the most common error is estimating the fundamental period as $2\tau 0$ (F0/2) rather than the true $\tau 0$ [13].

3.2.2 Harmonic pattern matchers

Harmonic pattern matchers attempt to find the F0 of a signal by matching an observed spectrum to a template spectrum. Template spectra can be artificially generated using knowledge about harmonic and inharmonic sounds or may be learned from training data using adaptive algorithms. The Maximum Likelihood (ML) method developed by Doval and Rodet [23], [24] is representative of the majority of harmonic pattern matchers. One method for computing the ML algorithm requires creating a matrix X of template spectra, where each ith row of the matrix contains a template spectrum $\mathbf{x_i}[k]$ corresponding to a candidate $\mathrm{F0}_i$.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1}[k] \\ \mathbf{x_2}[k] \\ \vdots \\ \mathbf{x_I}[k] \end{bmatrix}, \tag{3.5}$$

where k is the FFT bin variable. Each template ith spectrum may be considered to be a train of impulses, each of which is located at harmonic m of F0 candidate $\mathrm{F0}_i$ having tunable amplitude $A_{i,m}$, which is convolved with a smearing function g[k] (e.g., Gaussian function) to obtain

$$\mathbf{x_i}[k] = \sum_{m=1}^{M} A_{i,m} g\left[k - \frac{mKF0_i}{F_s}\right], \tag{3.6}$$

where M is the number of harmonics (taken equal to 20), K is the FFT window size, and F_s is the sampling frequency. The $\mathrm{F0}_i$ range and spacing are discussed in Sec. 3.3. The matrix $\mathbf{X}$ is then multiplied with the observed spectrum $\mathbf{O}[k]$, yielding a *salience array* or *salience function* s.

$$\mathbf{s} = \mathbf{X}\mathbf{O}^T. \tag{3.7}$$

The template spectrum that aligns best with the observed spectrum gives the global maximum of the salience function and is chosen to be the estimated F0, $\hat{F}0$, of the observed spectrum. Thus,

$$\hat{F}0 = F_i \text{ s.t. } \mathbf{s}(F_i) \geq \mathbf{s}(F_m) \forall F_m. \tag{3.8}$$

Figure 3.1 illustrates the maximum likelihood estimation method.

An alternate harmonic pattern matcher is the two-way mismatch (TWM) method developed by Maher and Beauchamp [8]. Rather than cross-correlating a predicted spectrum and an observed spectrum, the TWM uses a unique error metric to determine which template spectrum best matches the observed spectrum. Using only the frequency location and amplitude of the partials, the TWM measures the difference in frequency between each observed partial and the template partial nearest to it, and then measures the difference in frequency between each template partial and the

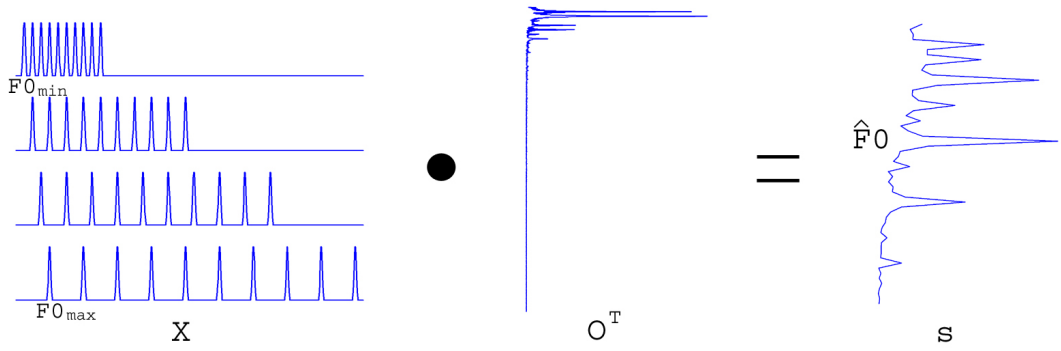

Figure 3.1 ML matrix multiplication.

observed partial nearest to it. Summations of these differences in frequency, weighted by the amplitudes of their associated partials, are then evaluated by the error metric. The F0 corresponding to the template spectrum yielding the least error is chosen to be the estimated F0.

Both the TWM and ML suffer from octave errors, much like the ACF, but rather than mistakenly choosing F0/2 to be $\hat{F}0$, they tend to choose 2F0. Periodicity analyzers make octave errors in estimation, because a function with period $\tau 0$ is also periodic with period $2\tau 0$, but not $\tau 0/2$. On the other hand, harmonic pattern matchers can find a complete harmonic series above both F0 and 2F0, but not above F0/2. Estimating F0 incorrectly to be 2F0 more commonly occurs if f_2 is the strongest harmonic.

3.3 Implementation Details

The author's ACF, AMDF, and YIN implementations corresponded to algorithms found in deCheveigné's evaluation of the algorithms [2]. The TWM algorithm implemented followed the description given by Maher and Beauchamp [8] as closely as possible. How spectral peaks were defined was not described in the 1994 paper, so for this implementation of the TWM a peak of the observed spectrum was defined to be

any peak above a threshold that has twice the amplitude of its neighboring frequency bins two bins away. All other formulas for the TWM can be found in the 1994 paper. Because of the uncertain definition of a spectral peak, error rate data found in this thesis may not accurately reflect Maher's implementation and may account for the TWM's overall slightly higher error rates. Since robustness of the algorithms rather than minimal error rates was the chief concern of the investigation, the slightly raised error rates are acceptable.

However, for the ML implementation described in this thesis many more design choice liberties were taken. A hanning function was used both to window the STFT frames and as the ML smearing function. Also, rather than using strict impulses as a basis for the template partials, rectangular boxcars were used, centered at $\frac{mK}{\tau_i}=\frac{mKF0_i}{F_s}$ with bandwidths

$$\Delta k_{\tau_i,m} = \left\langle \frac{mK/2}{\tau_i - \Delta\tau_i} - \frac{mK/2}{\tau_i + \Delta\tau_i} \right\rangle = \left\langle \frac{mK\Delta\tau_i}{\tau_i^2 - \Delta\tau_i^2} \right\rangle \approx \left\langle \frac{mK\Delta\tau}{\tau_i^2} \right\rangle, \tag{3.9}$$

where $\langle \cdots \rangle$ means rounded to the nearest integer and $\Delta\tau$ is a tunable parameter between zero and one. It was decided empirically to set $\Delta\tau$=0.5. It is important to note that the bandwidth of each boxcar increases linearly with m. This increase in bandwidth is used to compensate for inharmonicities. It is also important to note that since the spacing between the center of the boxcars is $\frac{K}{\tau_i}$, $\tau_i > 20$, and $m \leq 20$, $\frac{K}{\tau_i} > \frac{mK\Delta\tau}{\tau_i^2}$ so the boxcars will never overlap. All frequency bins within the range of $\Delta k_{\tau_i,m}$ are given amplitude $1/m$ before the smearing function is applied. One template spectrum per F0 candidate is created using 20 partials, limited by half the sampling rate. The template spectra unique to this implementation are labeled as $z_{\tau_i}[k]$ to distinguish them from the template spectra in Eq. (3.6):

$$z_{\tau_i}[k] = \sum_{m=1}^{M} \frac{1}{m} g[k] * boxcar\left[k - \frac{mKF0_i}{F_s}; \Delta k_{\tau_i,m}\right], \tag{3.10}$$

where '$*$' indicates convolution

The range of candidate F0s is between 40 Hz and 2100 Hz (E_1 - C_8), and there is one F0 candidate per fundamental period that is an integer number N of samples long $\left(\text{i.e., } F0_i = \frac{F_s}{N}\right)$. Using the sample fundamental periods offers a number of advantages. The number of distinguishable pitches is denser in the lower frequencies (E_1 and F_1 are only 2 Hz apart while E_6 and F_6 are 80 Hz apart), so it is better to have the list of candidate F0s be more densely populated for the lower pitches. Using fundamental periods corresponding to an integer number of samples provides this type of density. For example, with a 4096 sample window at 44.1 kHz, there are 29 F0 candidates between E_1 and F_1, but only 10 F0 candidates between $B_7^{\flat}$ and C_8. Finer resolution between the lower frequency F0s is also helpful when a sound has inharmonicities. Because lower F0s have more harmonics below half the sampling frequency and inharmonicities (e.g., for stretched string sounds) increase with each higher harmonic, the effect of any inharmonicity in the sound will be more pronounced for the lower F0s. Therefore, having several F0 candidates for the lower F0s helps ensure that at least one F0 candidate for the lower F0s will align well with the exhibited inharmonicities.

CHAPTER 4

EVALUATION OF TECHNIQUES FOR EXTENSION TO POLYPHONY

4.1 Selection of Test Signals and Algorithms

Test sounds were gathered from the University of Iowa Theremin database [25] as well from the University of Illinois Computer Music Project collection of sounds which have been used for previous pitch detection experiments [8]. All recordings were recorded at 16 bit, 44.1 kHz. Each algorithm used a short-time Fourier transform (STFT) using a 2048 sample (46.4 ms) frames windowed by a hanning function and a 256 sample (5.8 ms) hop size. The ground truth used for measuring error rates for each test signal was based on visual examination of the signals' spectrograms aided by musical scores and listening. The ground truth was confirmed by running all algorithms on each test signal. If there was a unanimous consensus of the estimated pitch within a semitone between all algorithms and the visual examination, the consensus pitch was established as the ground truth. If there was any disagreement between the algorithms and the visual examination, the visual examination was reevaluated in light of the results of the automatic F0 estimators. In most cases, the estimated pitch gathered by visual examination would remain as the ground truth or if it was adjusted it would be by less than a semitone.

No pre- or postprocessing algorithms (e.g., spectral whitening, median filtering, music theory based heuristics, etc.) were included in any of the final implementations of the chosen algorithms, because adding these algorithms would make it more difficult to critically evaluate the strengths and weaknesses of each general algorithm. The

lack of pre- or postprocessing in addition to the forced use of a hanning window for all algorithms (some algorithms may work better with a different window) may account for the slightly higher error rate data found in this thesis when compared with the literature for these algorithms. However, the primary concern of this research was to measure the overall robustness of the algorithms, and not their optimal performance.

In order to be chosen as the best F0 algorithm to use as the basis for a multiple-F0 estimator, the chosen algorithm must be able to reliably estimate F0 under certain criteria. The first requirement is that the chosen algorithm must be)(1) robust against transient signals. Because polyphonic signals have more sources than monophonic signals they will invariably have more transient or nonstationary elements. Sources of transients are instruments starting or ceasing to play, changing notes, rapid intensity fluctuations due to beating, tremolos, and rapid frequency changes due to vibrato. The two other requirements, which are closely related, are that the algorithm must be (2) robust against noise and (3) robust against reverberation. In polyphony there will always be spectral data that is unrelated to any particular F0. These unrelated spectral data can either, much like noise, bury the partials of one voice under the spectral content of the other voices, or the spectral data can take the form of spurious partials which increase the salience of some candidate F0s that are not truly in the sound. Adding white Gaussian noise to a monophonic sound can simulate a voice being buried under unrelated voices. This simulation necessitates the second requirement. Reverberation in a nonstationary signal can add a limited number of extra partials (coming from previous notes) unrelated to the predominant F0 and mislead the F0 estimator. The addition of unrelated partials necessitates the third requirement.

Of the requirements described above for a multiple-F0 estimation basis algorithm, only robustness against noise has been documented [2],[13]. Because the cepstrum and generalized autocorrelation methods are less robust against noise than the autocorre-

lation function (ACF) methods, it was decided not to investigate their performance any further. The Average Magnitude Difference Function (AMDF) and ACF are both required to implement the YIN, so all three methods were chosen to be evaluated. It was also chosen to evaluate both the Maximum Likelihood (ML) and the Two-Way Mismatch (TWM) methods.

To test the requirements described above, each algorithm was tested with four sets of monophonic sounds. The performance of each algorithm was evaluated by tabulating the number of errors - defined to be estimated F0s ($\hat{F}$0s) that were more than half a semitone away from the ground truth - generated from frame-by-frame analysis of the sound. For each algorithm, the error rates shown in the tables below were calculate as the percentage of frames whose F0s were estimated in error. Table 4.1 shows the baseline performance error rate for each algorithm tested on the University of Iowa database. The baseline error rates are critical for evaluating the robustness of the algorithms for the other three categories of sounds. The Iowa database consisted of long, sustained notes played in an anechoic chamber. All notes within each instrument's (brass, woodwind, and string instruments were tested) range (altogether B_1 through B_7) were evaluated. A second set of sounds were sounds with many transients, including pizzicato (plucked strings) examples from the University of Iowa database (violin, viola, cello, and bass recordings were used), as well as dry recordings of virtuosic examples (short excerpts with many rapidly performed notes) from the UIUC CMP collection. A third set of test sounds used consisted of the baseline recordings with their SNR lowered by adding Gaussian white noise. The final set of sounds tested consisted of virtuosic recordings recorded in concert halls with natural reverberation.

Table 4.1 Baseline error rates

Algorithm	Baseline Error Rate %
Autocorrelation Function	4.08%
Average Magnitude Difference Function	3.62%
YIN	2.94%
Maximum Likelihood	5.16%
Two-Way Mismatch	6.04%

4.2 Testing of Nontrivial Signals

4.2.1 Transient signals

It can be seen in Table 4.2 that the error rates for the virtuosic examples do not increase much above the baseline error rates.

Table 4.2 Error rates for sounds with many transients

Algorithm	Virtuosic Examples Error Rate % (change in % from baseline)	Pizzicato Examples Error Rate % (change in % from baseline)
ACF	4.53% (+0.45%)	10.47% (+6.39%)
AMDF	3.64% (+0.02%)	9.19% (+5.62%)
YIN	3.02% (+0.08%)	8.51% (+5.57%)
ML	5.24% (+0.08%)	6.19% (+1.02%)
TWM	5.81% (-0.23%)	8.70% (+2.66%)

However, as shown in Fig. 4.1 a marked difference in performance between the ML and the ACF can be easily seen for a portion of a dry recording of the Bach Preludio from Partita No. 3 for solo violin. For this case, the ACF fails to detect any period during note transitions, while the ML algorithm is able to accurately

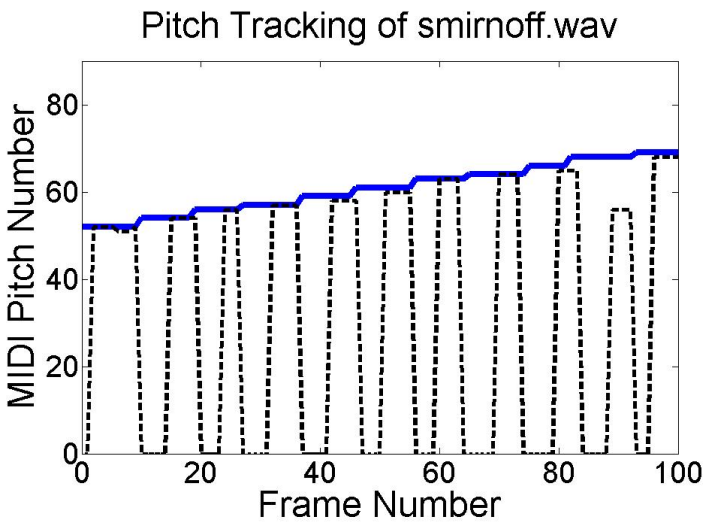

Figure 4.1 Comparison of ML (solid line) and ACF (dashed line) pitch tracks.

estimate the correct F0 even during transitions. The AMDF and YIN also exhibited difficulties with transitions, while the TWM gave results similar to the ML. The difficulty that periodicity analyzers have in dealing with transitions is due to their reliance on estimating F0 as the first satisfactory peak or minima. If the fundamental period of the signal changes partway through a frame, as is the case during any note transition, a shifted version of the signal will rarely correlate well with the original signal. In order for these techniques to work properly it is necessary that the signal be approximately wide-sense stationary relative to the window size. A similar difficulty of detecting the onset and offsets of pizzicato notes was observed for the periodicity analyzers, but unlike the virtuousic examples, a significant increase in error rates was also observed.

4.2.2 Noisy signals

In order to test how robust each of the algorithms are against noise, all sample files from the University of Iowa database with at least 50 dB of SNR were reevaluated. Gaussian white noise was added to each of the signals to lower the SNR to 50 dB,

44 dB, 35 dB, 20 dB, 0 dB and -20 dB SNR. While the pitch of a signal can be accurately perceived by humans for signals with SNRs as low as -20 dB SNR, none of the algorithms exhibited error rates below 99% for 0 dB or -20 dB SNR, so Table 4.3 only contains error rate data for 50 dB, 44 dB, 35 dB, and 20 dB SNR.

Table 4.3 Error rates for varying SNR levels

Algorithm	50 dB SNR	44 dB SNR	35 dB SNR	20 dB SNR %
ACF	3.84%	4.18%	24.33%	74.12%
AMDF	3.32%	3.53%	25.51%	72.95%
YIN	2.69%	2.89%	23.60%	71.02%
ML	4.12%	4.65%	24.78%	68.42%
TWM	5.01%	5.74%	30.01%	69.59%

While there is not a significant difference in robustness between the periodicity analyzers and the harmonic pattern matchers, it is worth mentioning that the accuracy of the ML and TWM methods can be improved in the presence of noisy signals by using a larger window size. If the window size of the STFT is larger, then the peaks of the spectrum become more pronounced and rise further above the background noise spectrum. Figure 4.2 demonstrates this phenomenon. The left two graphs show FFTs of 2048 sample (46.4 ms) hanning windowed frames of a bassoon. For the right two graphs the windows have been enlarged to 20480 samples (464 ms). The top two graphs are FFTs of the signals with 50 dB SNR, while the bottom two graphs are FFTs of the same signal with the SNR lowered by the addition of Gaussian white noise to -20 dB SNR. The partials of the sound can be seen for both window sizes at 50 dB SNR. However, when the SNR is lowered to -20 dB the partials of the signal completely disappear for the 2048 sample window, but the larger window size causes the first two partials to be visible again. Having just these two partials makes it possi-

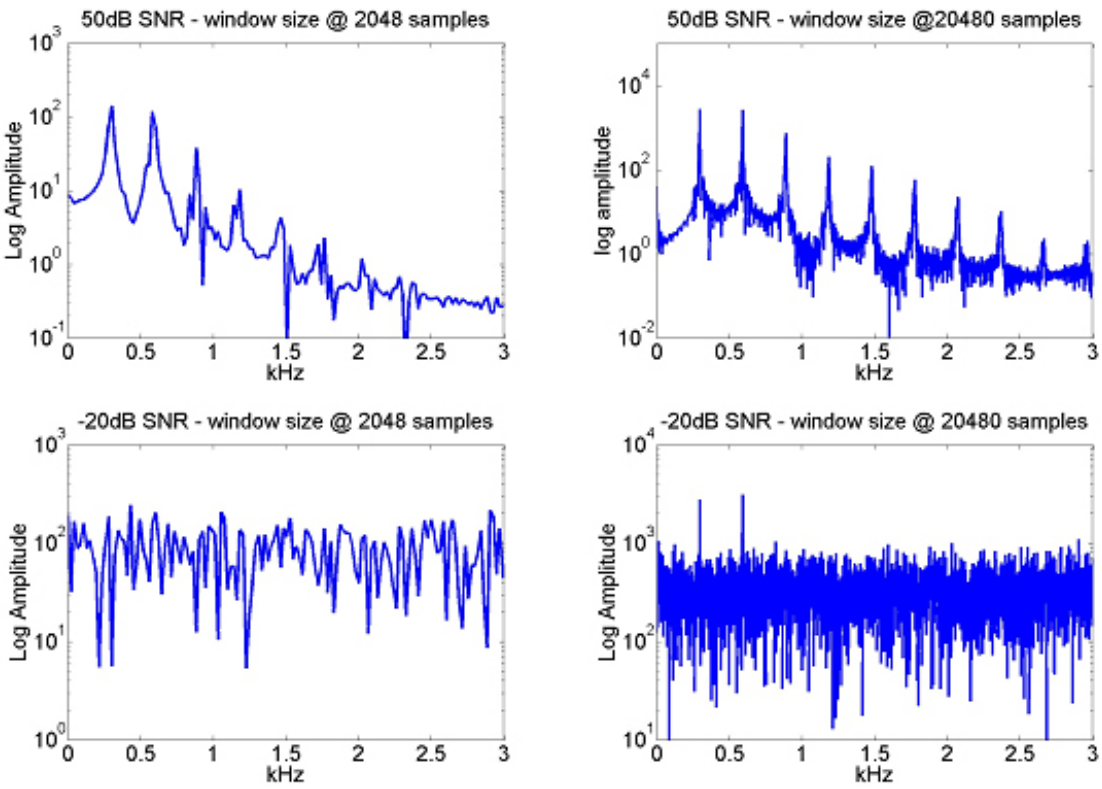

Figure 4.2 FFT of a sound at 50 dB and -20 dB SNR and window sizes of 2048 samples and 20480 samples.

ble for the ML and TWM to accurately estimate F0. However, this technique should only be used sparingly for fairly stationary signals due to the attendant decrease in time resolution.

4.2.3 Reverberant signals

Only the ML algorithm performed well for recordings with natural concert hall reverberation as shown in Table 4.4. The periodicity analyzers performed poorly, because reverberation makes a sound aperiodic. The TWM performed poorly, perhaps because reverberation adds more partials to each frame of the sound. These added partials from reverberation are unrelated to each candidate F0's harmonic series, and therefore every candidate F0 has a large amount of error. The increase in error for all F0 candidates effectively obscures the choice of the correct F0. The ML does not

Table 4.4 Reverberation error rates

Algorithm	Reverberation Error Rate %
Autocorrelation Function	99.7%
Average Magnitude Difference Function	100.0%
YIN	99.1%
Maximum Likelihood	11.3%
Two-Way Mismatch	99.3%

experience these penalties, because any partial which is unrelated to the harmonic series of a candidate F0 is zeroed out by a template spectrum. Therefore these unrelated partials do not affect the salience of the correct F0. Any partials coming from reverberation typically have smaller amplitude than the partials coming from the source, and therefore any candidate F0 that includes these spurious partials will only experience a slight increase in salience. The increase in salience of the false F0 candidates can shift away from the true F0 and induce an error, but this shift only occurs occasionally. Nevertheless, it must be conceded that the error rates given in Table 4.4 are for the author's implementations of the algorithm and that the original designer's implementations may give better results.

4.3 Summary

Overall, the ML algorithm appeared to be the most robust against the predicted added difficulties of multiple-F0 estimation. In addition to the ML's robustness, it was decided to use the ML as the basis for a multiple-F0 estimator because there are several intuitive methods to extend the ML to polyphonic signals that have been documented in the literature. These methods will be described in full in Chapter 5.

CHAPTER 5

MULTIPLE F0 ESTIMATION ALGORITHMS

5.1 Introduction

Tracking the pitches present in a polyphonic signal is significantly more difficult than the monophonic case explored in the previous chapters. Creating a system that is robust enough to track all pitches present in most polyphonic recordings requires determining the number of pitches (number of voices) present as well as determining the pitch of those voices as a function of time. Prior research [13] indicates that both multiple-F0 estimation tasks required are equally complex, and discussion of the computation required for both techniques goes beyond the scope of this paper. The research described herein focuses only on the task of estimating the F0s present in a sound given a priori knowledge of the number of voices.

Using the Maximum Likelihood (ML) method as a basis for a multiple-F0 estimator gives a straightforward approach for estimating all F0s present in a sound. If P is the assumed number of voices present in the signal, the basic ML algorithm chooses the P largest local maxima in the salience function to be the P estimated F0s, $\hat{F}$0s. This simple technique, called the *direct approach*, works surprisingly well, but it can fail for certain complex tone combinations. For example, the direct approach fails when one voice creates many strong salience peaks, when overlapping harmonics result in beating, and when multiple voices induce *chimeric sounds* (to be defined in Sec. 5.2.3). These cases will be examined in the following sections.

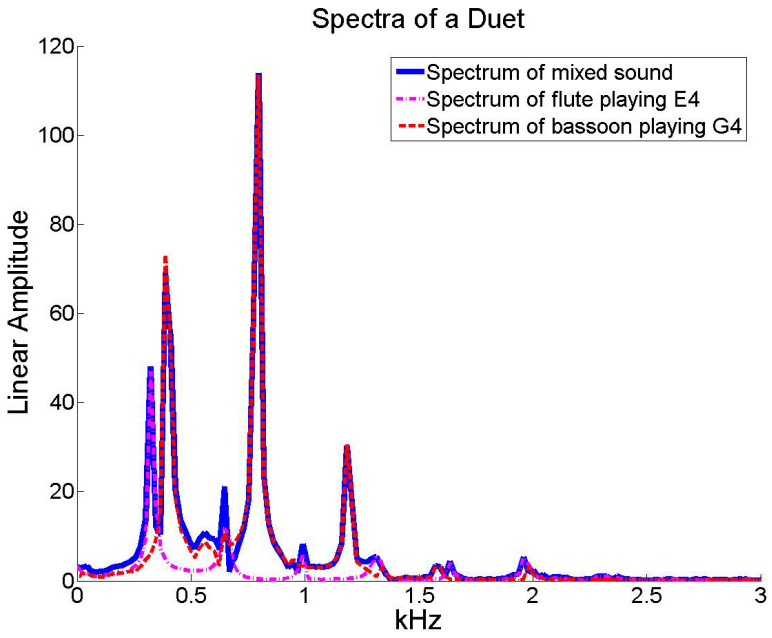

Figure 5.1 Spectra of a flute playing E_4, a bassoon playing G_4, and the spectrum of their combination.

5.2 Difficulties Introduced by Polyphony

5.2.1 Redundancy errors

One of the common mistakes encountered with the direct approach occurs when a single voice results in the estimation of multiple $\hat{F}0$s. This type of redundant error commonly occurs when the pitch of one voice is estimated to be the predominant pitch at its correct octave and the second most predominant pitch is estimated to be another pitch an octave higher or lower. This problem of redundant detection stems from the ML algorithm's intrinsic tendency to exhibit salience function peaks at F0s in an octave relationship with a real F0. If one voice is stronger than the others or if one of the voice's higher harmonics (f_2, f_3, etc.) has greater amplitude than than that of its f_1, there is a significant chance that a redundant error will occur. This type of problem is demonstrated by Fig. 5.1 where the individual spectra of a flute playing E_4 (F0≈330 Hz) and a bassoon playing G_4 (F0≈390 Hz) are superimposed on the spectrum of the

two instruments mixed together. Note that the flute's rms amplitude is about half that of the bassoon. This amplitude difference causes the largest salience peak to correspond to G_4 and the second largest salience peak to correspond to G_5 (F0≈780 Hz). Even without the amplitude difference, it is likely that the bassoon's high f_2 (≈780 Hz) amplitude would cause the second largest salience peak to correspond to G_5.

Redundancy errors are further demonstrated in Fig. 5.2 where P=3 and the F0s that are present in the sound are 78 Hz, 294 Hz, and 1696 Hz. From this salience function, it can be seen that the three most salient local maxima in the salience function are at 294 Hz, $s(i) = 97.71$; 588(=2×294) Hz, $s(i) = 79.29$; and 1696 Hz, $s(i) = 97.29$. The middle $\hat{F}0$, 558 Hz, is an octave above 294 Hz and would result in a redundancy error in the direct approach.

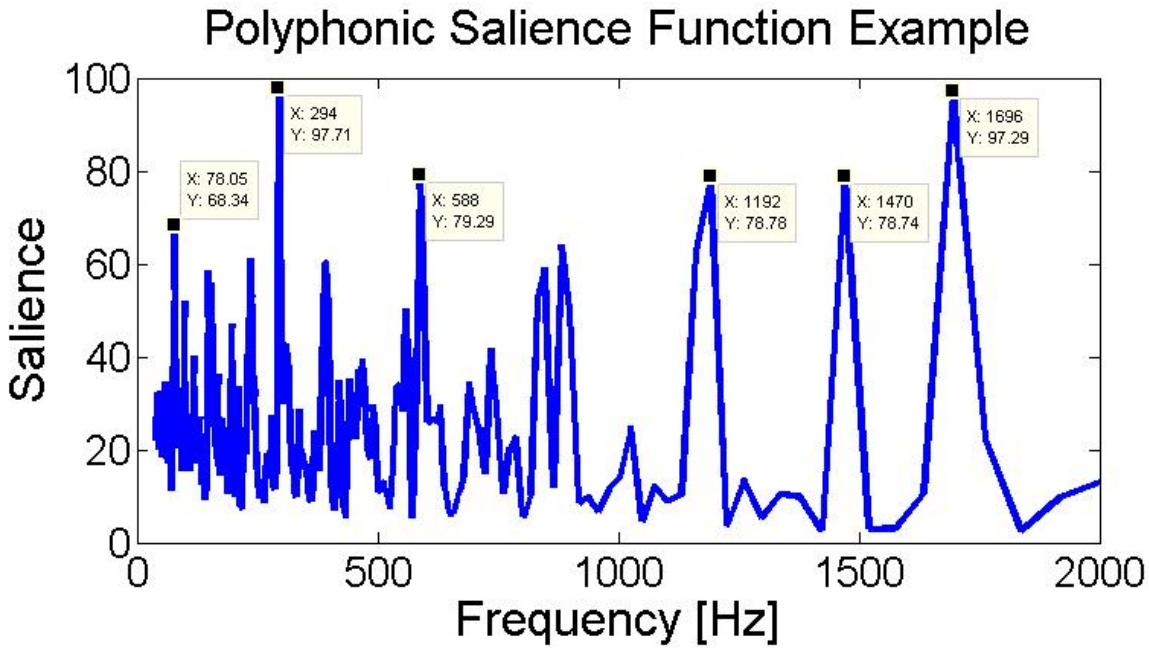

Figure 5.2 Polyphonic salience function example, where the real F0s are at 78, 294, and 1696 Hz corresponding to $E_2^\flat$, D_4, and $A_6^\flat$.

An expedient solution to this problem would be to disallow having 2 $\hat{F}0$s in an octave relationship. However, such a requirement is overly restrictive for most musical

examples, as sounds in octave relationships are common in music.

Another common problem is that occasionally two voices will produce the same pitch (unisons). In the case of unisons, the ML salience function will only have one peak associated with the unison. Therefore, the direct approach will be forced to select an erroneous salience peak in order to find P $\hat{F}0$s. Disallowing unisons in the test sounds can be used and has been used, but ultimately such restrictions minimize the usefulness of the algorithm and its commercial viability. A successful, ideal multiple F0 estimator needs to decrease the likelihood of redundancy errors and be able to properly analyze unisons while not restricting the types of sounds that can be analyzed. A common, successful technique (known as *iterative subtraction*) for dealing with redundancy errors (to be discussed in Sec. 5.3) involves iteratively repeating two steps: (1) estimating the predominant F0 of a remainder spectrum using the ML and (2) subtracting this harmonic spectrum from the remainder spectrum, until P $\hat{F}0$s have been detected. Such a technique is helpful for eliminating redundancy errors, but new errors can be introduced by corrupting partials that belong to more than one harmonic series.

5.2.2 Overlapping harmonics

Whenever two or more complex tones are present in a signal, their harmonic series may overlap (e.g., the partials in Fig. 5.1 occurring at approx. 2 kHz). Overlapping harmonics (a.k.a., "collisions") increase the difficulty of multiple F0 estimation both because of beating patterns that "corrupt" the mixed spectrum and because straightforward implementations of the iterative subtraction method previously mentioned are more likely to fail in this case. When tones are tuned according to the equal-tempered scale, overlapping partials will have small differences in frequency and therefore beats result. Beating, which was discussed in Chapter 2, can induce salience function peaks to change significantly from frame to frame, thus causing $\hat{F}0$s

to change erratically away from true F0s to other candidate F0s, commonly manifesting themselves as quasi-periodic errors in pitch-versus-time tracks. For example, Fig. 5.3 shows the pitch tracks for the same three note chord of Fig 5.2, where overlapping partials cause rapid periodic pitch shifts in the lower two voices (pitch_1=39 ($E_2^\flat$) and pitch_2=62 (D_4), but the highest pitch (pitch_3=92 ($A_6^\flat$) is unaffected.

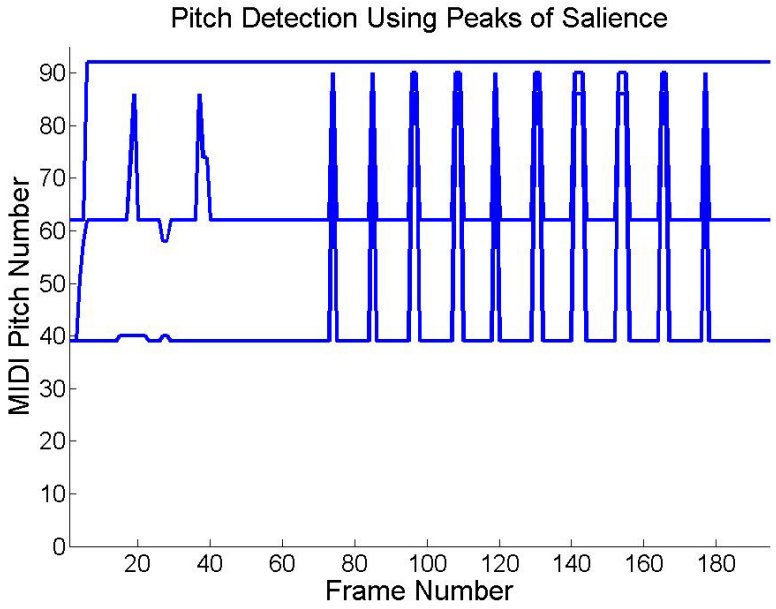

Figure 5.3 MIDI pitch tracks for a three-note chord demonstrating quasi-periodic errors caused by partial beating. Note that the top pitch (92) is uncorrupted whereas the bottom two (39 and 62) have several errors. One frame unit = 11.6 ms.

A general iterative subtraction method can be used to compensate for overlapping partials. If the predominant F0's contribution to each colliding partial can be accurately estimated, the iterative subtraction method can remove the $\hat{F}0$'s harmonic series without corrupting the spectra of the remaining F0s. Methods for estimating an $\hat{F}0$'s contribution to the spectra will also be discussed in Sec. 5.3 in more detail.

5.2.3 Chimeric sounds

Depending on the way the harmonics of different sounds collide, two voices are capable of alternatively masking the presence of another voice as just discussed or they can

artificially create the illusionary presence of an imaginary voice (*chimeric sound*). Because chimeric sounds are pleasing to listen to, they are common in recorded music. A common chord/chimeric sound in music is the major chord consisting of three pitches (e.g., C_4, E_4, and G_4) whose corresponding F0s share an integer relationship with the lowest note (the *root*) of the chord. The middle note's (the *third*) F0 (E_4) is in a 5:4 relationship with the root's F0 (C_4) and the highest note's (the *fifth*) F0 (G_4) is in a 3:2 relationship with the root's F0 (C_4). Because of the structure of harmonic sounds, many of the three voices' harmonics overlap. Sixty percent of the fifth's (G_4) harmonics collide with the harmonics of each of the root's and the third's harmonics (C_4 and E_4), and 47% of the root's (C_4) harmonics collide with the third's and the fifth's (E_4 and G_4) harmonics. If only the root and the third are played simultaneously, the chimeric relationship of the two tones may cause the chord to have a strong salience (and perception) at the fifth. In fact, the fifth's salience may be stronger than the salience of the two tones which created its chimeric presence. If it is specified that there are only two tones present in the signal, a multiple F0 estimator may choose the chimeric F0 and ignore one of the real F0s [26]. Another example of a chimeric sound can be seen in Fig. 5.2, where the F0s 294 Hz and 1696 Hz combine chimerically to increase the salience of the candidate F0s of 1192 Hz and 1470 Hz above the true F0 of 78 Hz. Therefore, one of the jobs of an advanced F0 estimator is to avoid wrongful F0 estimations caused by chimeric sounds.

5.3 Iterative Subtraction Methods

As mentioned earlier, an alternate approach to the direct approach for multiple F0 estimation is an Iterative Subtraction Estimator (*ISE*). Algorithm 1 details the basic ISE approach. In Algorithm 1, $Y[k]$ is the whitened observed spectrum, $Y_E[k]$ is the estimated combined spectrum due to the $\hat{F}$0s, and $Y_R[k]$ is the residual spectrum

obtained by subtracting $Y_E[k]$ from $Y[k]$. $Y[k]$ is whitened according to an algorithm developed by Klapuri [27].

Algorithm 1: Iterative Subtraction Estimator (ISE)

1. Initialize $Y_R[k] = Y[k]$, $Y_E[k] = 0$, and $\hat{P} = 0$.
2. Estimate the predominant F0 in $Y_R[k]$.
3. Add estimated spectrum to $Y_E[k]$.
4. Subtract $Y_E[k]$ from $Y[k]$ using $Y_R[k] = \max(0, Y[k] - Y_E[k])$, increment $\hat{P}$.
5. If $\hat{P}<P$, then repeat steps 2-4. [27]

If $Y_E[k]$ can be estimated well and can be successfully removed from $Y[k]$, redundancy errors will be eliminated and the salience of chimeric sounds will be lowered with each iteration. This algorithm is attractive because it reduces the chance of redundancy errors occurring, it can reduce the chance of errors induced by chimeric sounds, and it has low computation cost.

The major difficulty of implementing an ISE is estimating $Y_E[k]$. A number of different tactics can be used to estimate $Y_E[k]$ [27],[28], but a simple and effective method is to estimate $Y_E[k]$ as follows:

$$Y_E[k] = dY[k] \sum_{p=1}^{\hat{P}} z_p[k], \tag{5.1}$$

where p is the iteration number, $z_p[k]$ is $\hat{F}0_p$'s ML template spectra as described in Eq. 3.10, $\hat{P}$ is the iteration number, and d is a fixed tunable parameter between zero and one. If d is close to one, most spectral data of $\hat{F}0_p$'s harmonic series will be removed from $Y[k]$. In this case, if there are any partial collisions, the salience of

the F0s containing the colliding partials will be lowered and their chances of being chosen as the next predominant $\hat{F}0$ will be reduced. If it is desired to avoid this type of spectral corruption, a lower value of d can be chosen. However, if d is close to zero, little spectral data of the $\hat{F}0_1$'s harmonic series will be removed from $Y[k]$, and $\hat{F}0_1$ will remain as the most predominant F0 for every iteration. Therefore, it is best to select d to be around 0.5, unless it is known that there are no or few collisions.

Although an ISE is useful for reducing redundancy errors or some chimeric sound errors, it introduces a new type of error through spectrum corruption in the presence of collisions. This problem is magnified if the ISE makes a mistake during an early iteration. Any mistake in early iterations will cause many partials to be corrupted that otherwise should not have been changed, and therefore the first error will propagate to later iterations.

Another major problem for ISE is that it is extremely difficult to tell whether an error has occurred during estimation. If all F0s are correctly estimated in a wide-sense-stationary portion of a signal, when the $\hat{F}0$s' harmonic spectra are removed from $Y[k]$, $Y_R[k]$ should be fairly flat and its average RMS amplitude should be close to zero. If there is an error in estimating a F0, it is likely that there will be several high amplitude partials remaining in $Y_R[k]$. The presence of these high amplitude partials in $Y_R[k]$ can be used as one method to indicate that an error has occurred. However, this type of error is not the only source of high amplitude partials in $Y_R[k]$. If there is reverberation or if the signal is not wide-sense-stationary relative to the window size (which is often the case in polyphonic sounds as discussed in Chapter 3) or if there are inharmonicities in the sound, the predicted spectra will not be able to flatten $Y_R[k]$, and high amplitude partials will remain. Distinguishing between those partials remaining in $Y_R[k]$ due to F0 misestimation and those due to inharmonicities and nonstationary signals is difficult at best. Even if errors can be detected reliably, it is impossible to correct the errors efficiently. Error correction requires determining

which iteration was in error and then repeating the algorithm until a satisfactory final $Y_R[k]$ is found. Running multiple trials of ISE negates the computational advantages of ISE making it less attractive.

5.4 Joint Subtraction Method

In light of the weaknesses of the direct and iterative subtraction approaches, another possible algorithm is Joint Estimation (JE). JE attempts to estimate all voices by simultaneously finding the best combination of F0s rather than finding the P most salient F0s one at a time. This method requires creating a template spectrum for every combination of F0s, and then calculating the ML salience function using all the template combination spectra. The most salient combination of F0s is chosen to be the set of $\hat{F}$0s. This approach is attractive, because it is simple to understand and visualize and offers the possibility of computationally efficient error correction. JE also does not suffer as much from redundancy errors or chimeric sounds. A practical JE error correction algorithm is discussed in Sec. 5.6.2.

Unfortunately, a straightforward implementation of a universal JE algorithm is computationally untenable. To demonstrate why this method is impractical, consider even a well-defined and well-bounded set of F0 combinations. If 100 F0s (a convenient number) are considered as candidates and if the polyphony P is three, there will be one million (100^3) different combinations of F0s to evaluate. Computing the salience of so many combinations of F0s is already untenable, but even these initial combinations are not enough for a successful algorithm. Straightforward combinations of ML template spectra work well for mixes where every voice has similar average RMS amplitude (loudness), but they may fail if the average amplitudes are significantly different. Therefore, for this case more than one million combinations would be needed to account for the range of average rms amplitudes. For example,

quantizing the average RMS amplitudes to just 10 different levels would result in one billion ($(100\times10)^3$) possible candidate spectra to be examined.

The computational costs of JE have kept many true JE algorithms from being created, but simplifying approximations can be made. In order to get a feel for how to optimize the algorithm, a rough model of the computational costs of the algorithm was created to determine how the algorithm can be optimized. Using big O notation, JE is approximately a $O((LM)^P)$ algorithm (this use of big O notation may not be completely correct, but illustrates the author's reasoning), where L is the number of quantized average RMS amplitudes, M is the number of F0 candidates, and P is the number of pitches assumed to be in the signal. Because P is a fixed number determined by the signal, the only way to optimize the algorithm is to reduce L and M. If possible, it would also be desirable to convert the exponential cost of P to a multiplicative cost.

In order to reduce M, L can be ignored for the moment without too much loss of generality. When the ML's salience function is calculated, it is readily apparent that many F0 candidates do not appear in a given spectrum. Every reasonable F0 candidate should create a peak in the test spectrum's salience function. Therefore, all F0 candidates that do not result in a peak in the salience function can be ignored, and only the salience function's J ($J \ll M$) most prominent peaks need to be used to make combination spectra. From experience, retaining $J = 50$ F0 candidates is normally sufficient to capture all prominent peaks, even peaks from sources with low average RMS amplitude. Using more than 50 F0 candidates does not really improve performance, and good results can even be obtained with fewer candidates.

The computational expense incurred by L can be reduced by making a compromise between using a JE and an ISE. When using an ISE, the computational costs of the algorithm do not depend on L, because amplitude information is gathered directly from the signal. A JE can similarly gather amplitude information directly from the

signal if it is made to be iterative. This compromise algorithm is called a *Joint Subtraction Estimator* (*JSE*).

The goal of the JSE method is to iteratively build the best combination of $\hat{F}0$s. Algorithm 2 outlines the basics of how a JSE is calculated and Fig. 5.4 contains a simplified example of a JSE calculation. The general principle of JSE is to sum up the saliences of the $\hat{F}0$s chosen to compose a set, and then penalize or "inhibit" the selection of the set based on how much its $\hat{F}0$ template spectra overlap.

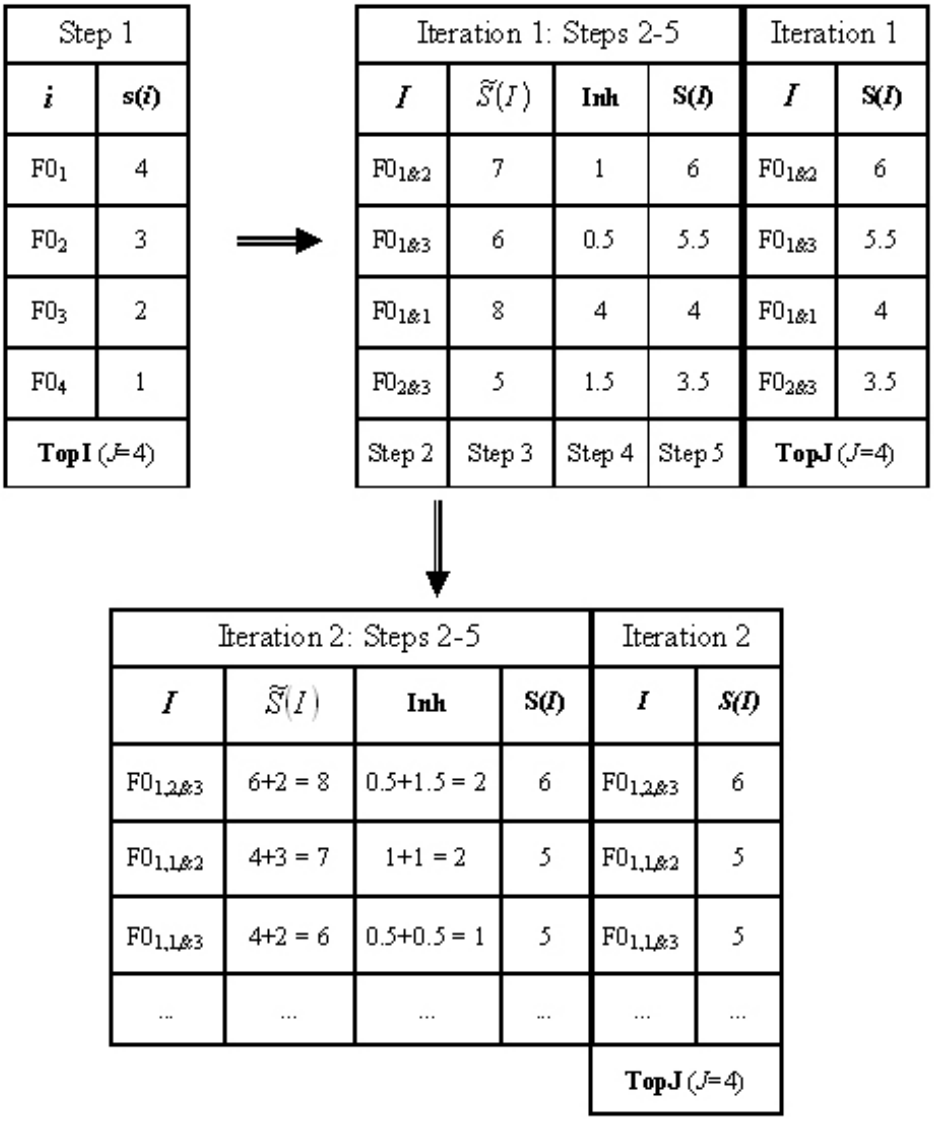

Figure 5.4 Demonstration of algorithm 2 through two iterations.

More specifically, the JSE algorithm can be explained as follows: Let I be a set of $\hat{P}$+1 F0 candidates, I_c be a set of $\hat{P}$ F0 candidates, and i be a single F0 candidate. In step 1 the saliences of all M F0 candidates are calculated by the ML algorithm, and the J most salient salience function local maxima are retained. J is a tunable parameter as previously described, and appropriate choices of J will be discussed in Sec. 5.6.5. **TopI** stores the F0 of each retained F0 candidate i and the salience of each candidate $s(i)$. **TopJ** stores the F0 of each candidate as a set I_c and the salience of each set $S(I_c) = s(i)$. In Fig. 5.4 the table labeled Step 1 shows an example of what may be stored in **TopI** with $J = 4$ after step 1 is completed. After step 1, **TopJ** is equivalent to **TopI**.

For step 2, all I_c from **TopJ** and i from **TopI** are combined to form J^2 sets. The initial salience of each new set I, $\tilde{S}(I)$, is the summation of the salience of set I_c and the salience of i (Eq. (5.2)).

$$\tilde{S}(I) = \tilde{S}(I_c \bigcup i) = S(I_c) + s(i) \tag{5.2}$$

Table *Iteration 1: Steps 2-5* in Fig. 5.4 shows the $\tilde{S}(I)$ for the first iteration. As an example, $\tilde{S}(F0_{1\&2})$ was calculated according to Eq. (5.2) as follows:

$$\tilde{S}(F0_{1\&2}) = \tilde{S}(F0_1 \bigcup F0_2) = S(F0_1) + s(F0_2) = 4 + 3 = 7. \tag{5.3}$$

The inhibition of step 4 is a measurement of how much the F0s in I overlap. Inhibition is essentially a penalty applied to the salience of a set I. The penalty mimics the corruption and loss of salience of candidate F0s that occurs during the ISE. Therefore, the inhibition or penalty against selection should increase as the overlap between the harmonic series of the F0s in I increases. This increasing inhibition decreases the chance of I's selection. One possible definition for the inhibition between two F0s is

$$Inh(i,j) = d\sum_{k} z_i[k]|Y[k]|z_j[k], \tag{5.4}$$

where d is a tunable parameter (see Eq. (5.1)), $z_\tau[k]$ are the template spectrum described in Chapter 3, and $Y[k]$ is the whitened observed spectrum.

A separate inhibition score should be calculated between each F0 in I_c and the F0 in i. The separate inhibition scores are added together for a total inhibition score which is subtracted from $\tilde{S}(I)$ to find the final salience:

$$S(I) = S(I_c) + s(i) - \sum_{j \in I_c} Inh(i,j) \tag{5.5}$$

In table *Iteration 1: Steps 2-5*, the values of Inh were calculated according to Eq. (5.4) and the final saliences $S(I)$ were calculated according to Eq. (5.5). In the first row

$$S(F0_{1\&2}) = \tilde{S}(F0_{1\&2}) - Inh(F0_2, F0_1) = 7 - 1 = 6. \tag{5.6}$$

A new **TopJ** is composed as shown in Fig. 5.4, and a new iteration is started.

Table *Iteration 2: Steps 2-5* in Fig. 5.4 shows the calculation of each $\tilde{S}(I)$ for the sets obtained during the second iteration. The first row of $\tilde{S}(I)$ was again calculated according to Eq. (5.2) as

$$\tilde{S}(F0_{1,2,\&3}) = \tilde{S}(F0_{1\&2} \bigcup F0_3) = S(F0_{1\&2}) + s(F0_3) = 6 + 2 = 8. \tag{5.7}$$

Similarly to iteration 1, the $S(I)$ in the first row in table *Iteration 2: Steps 2-5* was calculated as

$$S(F0_{1,2,\&3}) = \tilde{S}(F0_{1,2,\&3}) - [Inh(F0_3, F0_1) + Inh(F0_3, F0_2)] = 8 - (0.5 + 1.5) = 6. \tag{5.8}$$

The inhibitions between $F0_3/F0_1$ and $F0_3/F0_2$ are combined, because the harmonic

series of $F0_3$ can overlap with the harmonic series of both $F0_1$ and $F0_2$, and the salience should be penalized for both sets of overlaps. Because the inhibition between $F0_1$ and $F0_2$ was already accounted for in $S(F0_{1\&2})$, their inhibition does not need to be subtracted from the final salience of the set again. As a final step, a new **TopJ** is composed and the most salient set $F0_{1,2,\&3}$ is selected as the correct set of $\hat{F}0$s.

Algorithm 2: Joint Subtraction Estimator (JSE)

1. Initialize $\hat{P}$=1. If P >1, compute saliences for M F0 candidates, select the J largest local maxima of $s(i)$ as the J F0 candidates, and store the candidate F0s and their saliences in two arrays: **TopI** and **TopJ**. **TopI** stores the selected J F0 candidates and **TopJ** stores the J most salient combinations of F0s found. If P=1, then simply return the most salient F0 as $\hat{F}0$.
2. Generate J^2 new combinations of $\hat{P}$+1 F0 candidates using **TopI** and **TopJ**.
3. Calculate the salience of the J^2 combinations of F0s (Eq. 5.2).
4. Calculate the inhibition of each combination (Eq. 5.4).
5. Subtract this inhibition from each combination's salience to find the final salience of the combination. (Eq. 5.5).
6. Retain the J most salient unique $\hat{F}0$ combinations as **TopJ**.
7. If $\hat{P} \leq P$, then increment $\hat{P}$ and repeat steps 2-5. Otherwise, the most salient combination of $\hat{F}0$s should be selected as the correct combination of $\hat{F}0$s.

Calculting a JSE as just described not only removes the calculation time's dependence on L, but it converts the exponential dependence on P to a multiplicative dependence because the algorithm is only run through $P-1$ iterations. The JSE's

final computational cost is approximately $O(PJ^2)$ which is a great improvement over the joint detection's computational cost of $O((LM)^P)$.

From the computational analysis above it can be seen that the computation time required for JSE should increase quadratically with respect to J. Although less obvious, the accuracy of the JSE also has a dependence on J. If J is very small, then the accuracy of the JSE should decrease significantly. For example, if $J = 1$, then only one F0 candidate will be in **TopI**, and therefore only one combination of F0s can be formed (i.e., all P voices will be assigned the most salient F0). As J increases more F0 candidates will be kept, and more diverse combinations of F0s will be created. If J is large enough, the true combination of F0s should be found in **TopJ**. From experience it was found that (1) the salience of pairs drops fairly quickly after the 50 most salient sets and (2) a permutation of the true F0s present in the sound typically appears in the 50 most salient sets. The number of sets kept per iteration can be altered if desired, but in general J≈50 should be sufficient. Optimal choices of J are discussed later in Sec. 5.6.5.

It was decided to focus on and design a JSE for the following reasons:

1. JSE should theoretically be more accurate than ISE or the direct approach.
2. JSE has reduced computation costs when compared with JE.
3. Trade-offs between computation time and accuracy can be investigated.
4. Simple error correction techniques are available.

5.5 Implementation Details

The direct approach, ISE algorithm, and JSE algorithm investigated were all based on the the ML algorithm described in Secs. 3.2.2 and 3.3. However, the basic ML algorithm was modified by applying a spectral whitening algorithm to the observed

spectrum prior to calculating the salience function as mentioned in Sec. 5.3 [27]. The STFT uses a 4096 sample (93 ms) window and a 256 sample (5.8 ms) step size. The template spectra, $z_\tau[k]$, are created according to Eq. (3.10) using $\Delta\tau$=0.5 in Eq. (3.9).

The ISE algorithm is calculated according to Eq. (5.1) with $d = 1$.

The inhibition of the JSE algorithm is calculated according to Eq. (5.4) with d=0.5, and J was initially set to 50.

5.6 Modifications to the Basic JSE Algorithm

5.6.1 Introduction

Initial tests of the basic JSE algorithm using J=50 on random mixes of the Iowa Theremin [25] sounds yielded unsatisfactory performance both in terms of speed and accuracy. A number of different algorithms were investigated to improve the accuracy of this algorithm: an error-correction algorithm called antisalience, median filtering to smooth glitches in the pitch tracks, and a beating compensator. The value of J was also varied to explore the exact tradeoffs between speed and accuracy.

5.6.2 Antisalience

In order to analyze why the basic JSE algorithm failed, all of the J most salient sets of $\hat{F}0$s were analyzed for every frame. It was found that the correct set of F0s for any given frame frequently corresponded to one of the five most salient sets. It was also found that the most salient erroneous $\hat{F}0$ set frequently would align with the largest spectral peaks, but many spectral peaks would not be covered by the $\hat{F}0$ set. For example, in Fig. 5.5 the most salient $\hat{F}0$ set's spectra align with the largest peaks at f≈1192 Hz, 1470 Hz, and 1753 Hz, but all the peaks below 1192 Hz are ignored by the $\hat{F}0$ spectra. In Fig. 5.6, it can be seen that the correct $\hat{F}0$ set's spectra align well

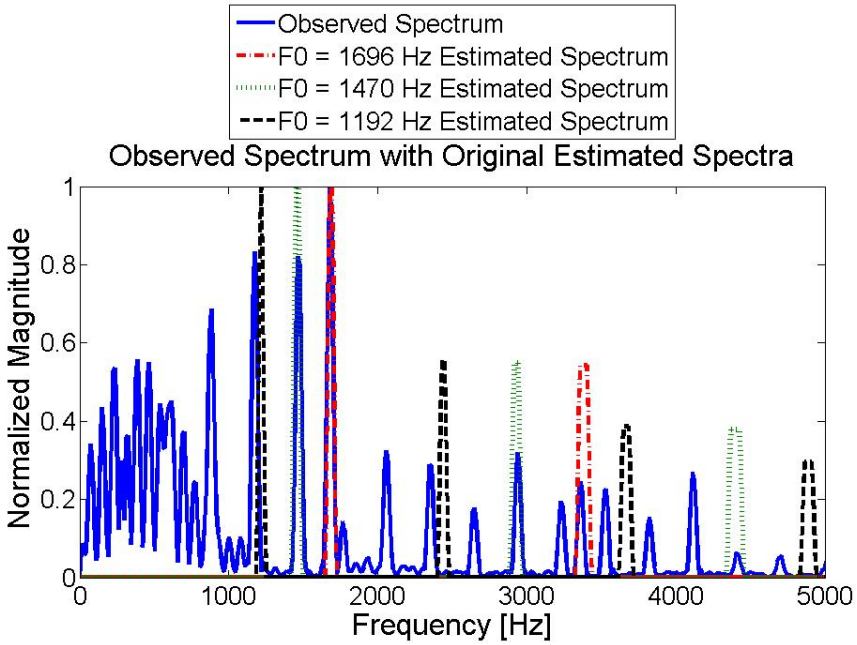

Figure 5.5 Superposition of the estimated set of spectra over the actual spectrum shows that many of the peaks (particularly those below 1192 Hz) are not accounted for by the estimated spectra.

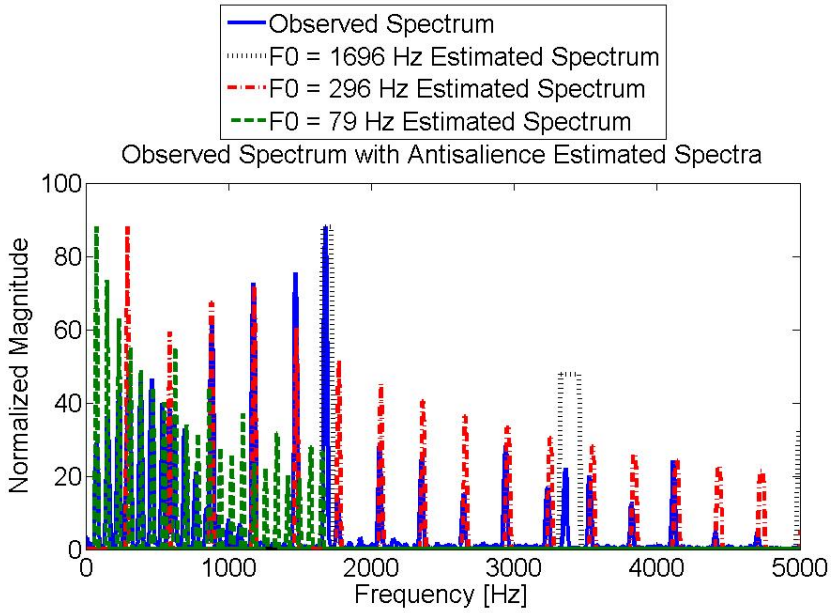

Figure 5.6 Superposition of the estimated set of spectra over the actual spectrum after correction using antisalience. Few peaks are not accounted for by these spectra. (See Fig. 5.2 for the original salience function.)

with all peaks in the observed spectrum. It was decided that $\hat{F}0$ sets which do not align with a number of spectral peaks should be penalized. This penalty is applied by reducing the saliency of an $\hat{F}0$ set in proportion to how much the spectral data were left unexplained by the $\hat{F}0$ set. The reduction of salience is why this error correction algorithm is called *antisalience*.

Algorithm 3 demonstrates how to calculate the antisalience of an $\hat{F}0$ set. First, a residual spectrum is calculated in a manner similar to step 4 from Algorithm 1 ($Y_R[k]$ =max(0,$Y[k] - Y_E[k]$)). For antisalience calculations, we want to remove all spectral data associated with the $\hat{F}0$ set, so d in Eq. (5.1) should be one. It was also found empirically that better results were gained by using boxcars with wider bandwidth for the template spectra. The antisalience template spectra, $Z_\tau[k]$ for distinction from the original $z_\tau[k]$, use $\Delta\tau = 1$ in Eq. 3.9, use boxcars with constant amplitude of one, and are not smeared. The residual spectrum, $Y_R[k]$, should completely remove any spectral data aligned with the $\hat{F}0$ set.

$$Y_R[k] = max\left(0, Y[k] - \sum_{i \in I} Z_i[k]Y[k]\right) \tag{5.9}$$

The residual spectra of the two $\hat{F}0$ sets in Fig. 5.5 and Fig. 5.6 are shown in Fig. 5.7. $Y_{R1}[k]$ is the remainder spectrum of the incorrect $\hat{F}0$ set and $Y_{R2}[k]$ is the remainder spectrum of the correct $\hat{F}0$ set. The large number of peaks remaining in $Y_{R1}[k]$ creates a severe penalty against the salience of the incorrect set, while $Y_{R2}[k]$ has few peaks and therefore only a small penalty is applied to the salience of the correct set.

The second step for calculating antisalience requires summing all of the residual spectrum's frequency bin amplitudes ($\sum_k Y_R[k]$) and then scaling the sum by the factor a_s. This summation, $a(I)$,

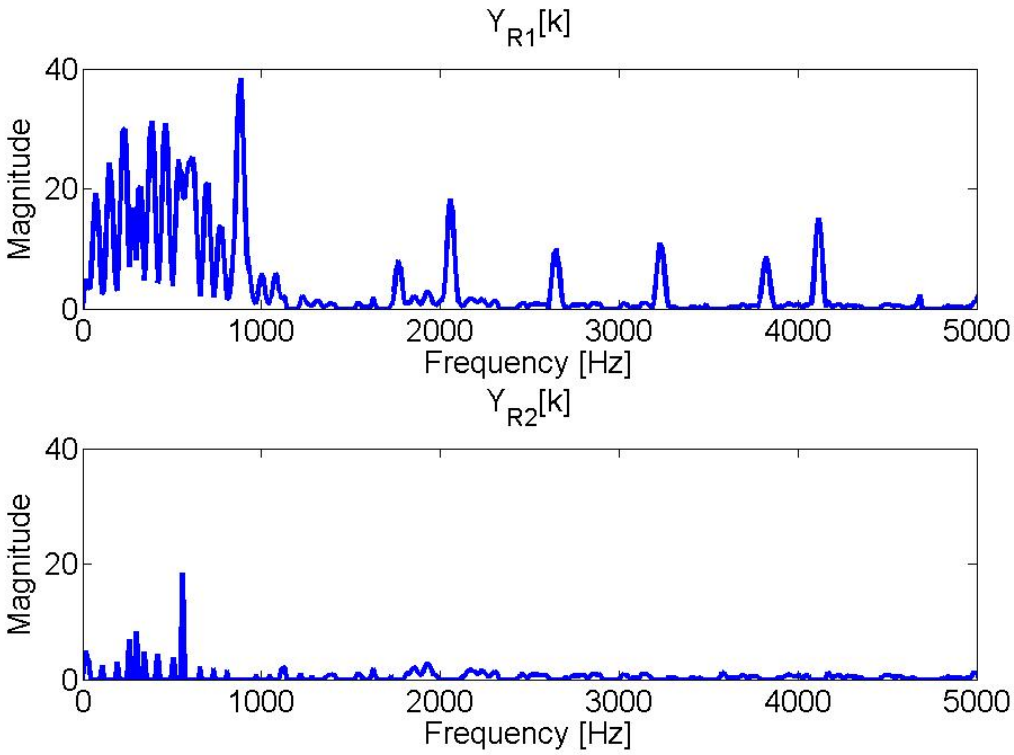

Figure 5.7 $Y_{R1}[k]$ is the remainder spectrum of an incorrect $\hat{F}0$ set and $Y_{R2}[k]$ is the remainder spectrum of a correct $\hat{F}0$ set.

$$a(I) = a_s \sum_k Y_R[k] \tag{5.10}$$

tells us how much of the observed spectrum was left unexplained by the $\hat{F}0$ set and is the magnitude of the antisalience penalty applied. It was found empirically that setting the scaling constant a_s=1/16 was optimal.

For step 3, a final salience is calculated for set I.

$$S_{as}(I) = S(I) - a(I) \tag{5.11}$$

The set with the largest $S_{as}(I)$ is then selected as the correct $\hat{F}0$ set.

Algorithm 3: Antisalience Calculation

1. Subtract all estimated spectra of set I from the observed spectrum (Eq. (5.9)).
2. Scale and sum the amplitudes of all frequency bins of the residual spectrum (Eq. (5.10)).
3. Subtract summation from set I's salience for a final set salience (Eq. (5.11)).
4. Repeat steps 1-3 for all J sets.
5. Select the set with the largest final set salience to be the correct set.

5.6.3 Median filtering

During sustained chords, the chosen $\hat{F}0$ set occasionally changes dramatically for a few frames and then returns to the original $\hat{F}0$ set. Dramatic shifts in pitch that last for only a few frames are rare in music. These glitches can be attributed to analysis error and are easily corrected using median filtering. Different filter lengths are optimal for different musical sounds. Long median filters can correct longer glitches and work well for sustained notes, such as the mixes of Iowa Theremin [25] sounds, but long median filters are too restrictive for rapidly changing notes. Imposing these restrictions can over-smooth a pitch track which requires many pitch changes. Shorter median filters correct very few glitches, and therefore offer only minimal improvements to error rates. It was found empirically that a median filter length of 15 frames (80 ms) was a good compromise between performance enhancement for sustained notes and not adversely affecting performance in passages of rapidly changing notes.

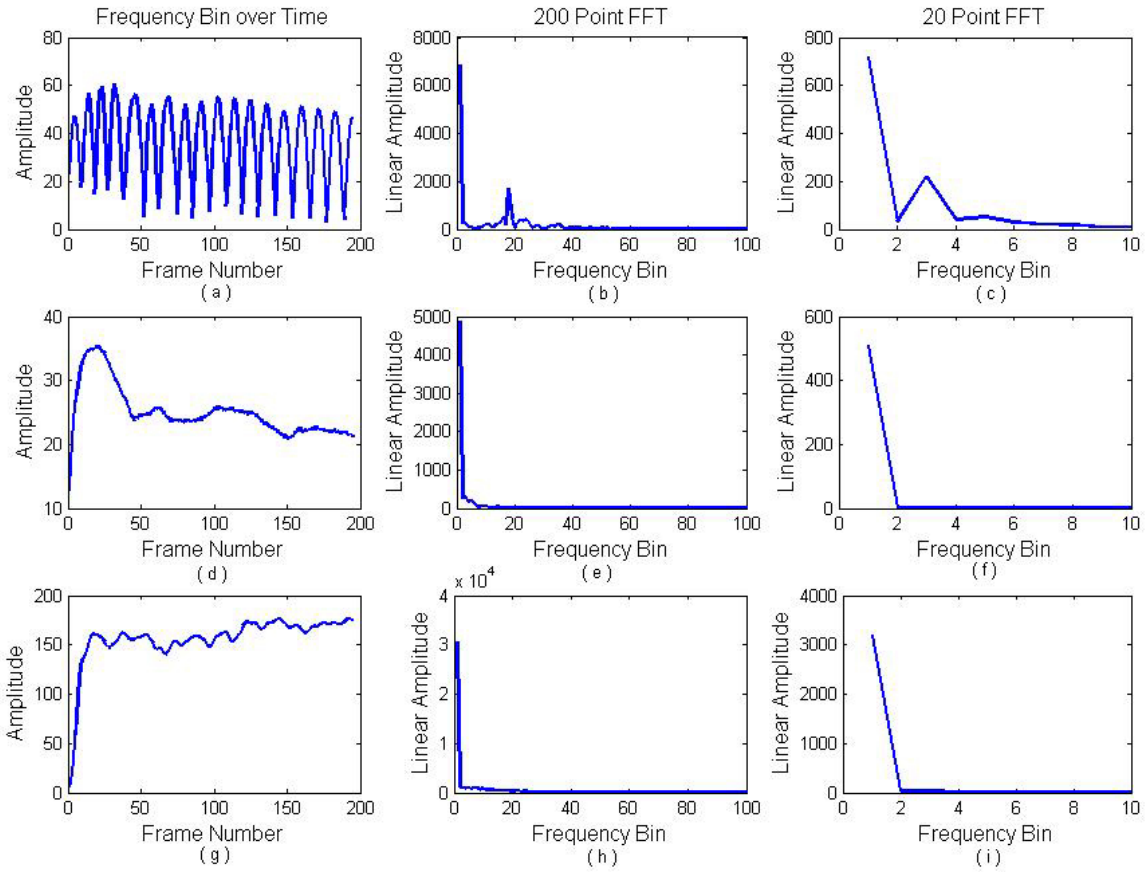

Figure 5.8 (a) Amplitude of a beating partial plotted over time. (d) Amplitude of an attack and decay partial. (g) Amplitude of a partial with random amplitude fluctuations. (b), (e) and (h) are 200 point FFTs of (a), (d), and (g) respectively. (c), (f) and (i) are 20 point FFTs of (a), (d) and (g), respectively

5.6.4 Beating compensation

Pitch tracks such as Fig. 5.3 motivated the idea to create an algorithm that could improve performance in the presence of beating. In Fig. 5.8 (a), (d), and (g) the evolution of three frequency bins is tracked over 200 frames. Fig. 5.8 (a) shows the amplitude vs. time envelope of a frequency bin exhibiting strong beating; (d) and (g) show envelopes for frequency bins exhibiting normal amplitude variation (i.e., no beating); (b), (e), and (h) show 200 point FFTs of (a), (d), and (g) respectively; and (c), (f), and (i) show 20 point FFTs of the same envelopes. According to Eq. (2.2), frequency bins affected by beating should have a peak in frequency content away from

the lowest frequency bins. The peaks in the FFT can be used to flag beating.

In order to capture this frequency peak, every frequency bin's amplitude envelope is analyzed using a STFT. The STFT uses 16 frame windows and a step size of 4 frames. Every frame of the new STFT is evaluated for a sufficiently large peak indicating a beating partial and is flagged if a peak is found. Each frame of the original STFT can be flagged by 4 frames of the new STFT. If 3 or more new frames flag an original frame, then a beating compensation algorithm should be applied to the pitch analysis of the original frame.

Beating compensation is performed by smoothing the beating partial's amplitude to remove the valleys from the beating patterns. The amplitude of the beating bin is smoothed by assigning every frame of the beating partial to have amplitude equal to the maximum amplitude for the bin within a 17 frame (100 ms) window centered on the bin being adjusted. The maximum amplitude is chosen, because the maximum amplitude reflects that two tones are contributing to the partial. Figure 5.9 demonstrates the smoothing operation. The dotted line is the beating partial from Fig. 5.8 a) and the solid line shows the adjusted amplitude of the frequency bin.

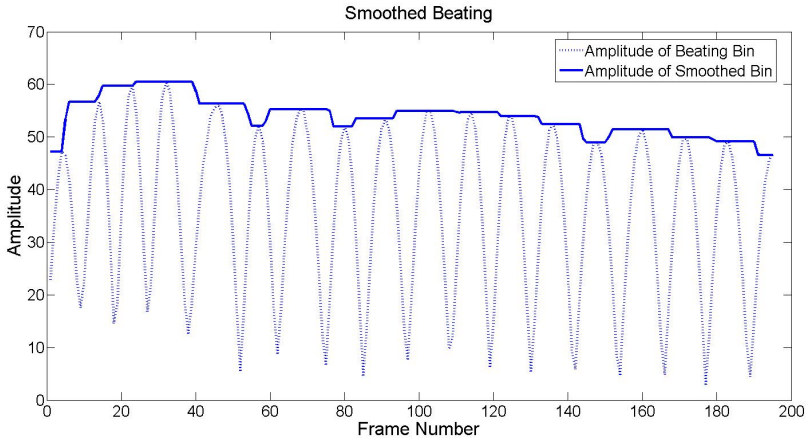

Figure 5.9 Dotted line is the amplitude of a beating partial before smoothing and the solid line is the amplitude of the beating partial after smoothing.

5.6.5 Time-versus-accuracy scaling

Finally, the tradeoffs between accuracy and speed of the JSE algorithm were analyzed. The most potent parameter for tradeoffs between accuracy and speed is the choice of J. When J is low, the algorithm has fewer combinations of F0s to examine and is faster. However, if J is too low a correct combination of F0s may not exist in an early iteration and therefore cannot exist in the final iteration. It is important to note that the number of highly salient F0 candidates increases as P increases, so the ideal choice of J scales with P.

To find the ideal choice of J, the computation time of the JSE and accuracy of the JSE were evaluated at $J = P$, $J = 2P$, $J = 5P$, and $J = 10P$. These measurements were compared with the computation time and accuracy of the direct approach. The direct approach's metrics serve as a baseline expectation for the performance of the JSE. Detailed results are presented in Chapter 6.

CHAPTER 6

RESULTS AND ANALYSIS

6.1 Introduction

The performance of several polyphonic F0 estimators discussed in Chapter 5 were evaluated using error rates as defined in Sec. 4.1. The algorithms were again tested using sounds taken from the Iowa Theremin database [25] and from the University of Illinois Computer Music project database. Random mixes of sounds from the Iowa Theremin database [25] were used to create part of the test set, and random mixes of sounds from the Computer Music Project database made up the rest of the test set. The Iowa Theremin recordings [25] were combined into mixes of 1, 2, 3, 4, and 6 voices for easier comparison with previous work. The Computer Music Project recordings were mostly gathered from a woodwind quintet, so random mixes of these recordings were used to test only 1, 2, 3, and 4 voices (5 voices was not tested, because that number of voices had not been evaluated with the Iowa recordings). The author's JSE algorithm used J=50 (J = number of F0 sets retained per iteration of the JSE) for all cases, except when analyzing the trade-offs between accuracy and computation time.

6.2 Original Algorithms

Performance of the author's algorithms is compared to the methods proposed by Klapuri [27] using his reported error rates. Approximate error-rate data for Kla-

puri's direct approach estimator, an ISE, and a JSE are shown in Table 6.1. It can be seen in Table 6.1, that the author's direct approach algorithm and Klapuri's direct approach algorithm had comparable performance, but Klapuri's ISE and JSE algorithms performed better than the author's comparable algorithms. It is believed that the disparity between the two ISE performances was largely caused by different methods of subtracting away $\hat{F}0$ spectra. The disparity in performance between JSE algorithms was similarly caused by different inhibition function designs. The comparatively poor error rates of the basic algorithm motivated the use of antisalience. The use of antisalience resulted in significant accuracy gains for the JSE.

Table 6.1 Error rate percentages for the Herman and Klapuri direct approach, ISE, and JSE algorithms

# of voices	Herman Direct Approach	Klapuri Direct Approach	Herman ISE	Klapuri ISE	Herman JSE	Klapuri JSE
1	3.5%	3.5%	3.5%	3.5%	3.5%	3.5%
2	16.0%	20.0%	11.7%	5.0%	9.8%	5.0%
3	22.9%	-	20.2%	-	17.0%	-
4	29.7%	30.0%	27.8%	12.0%	20.8%	12.0%
6	35.9%	35.0%	30.7%	21.0%	28.1%	21.0%

6.3 Antisalience

Table 6.2 shows the error rates for the author's direct approach, JSE without antisalience, and JSE with antisalience algorithms as well as Klapuri's JSE algorithm.

Using antisalience reduced the JSE's error rates by an average of four percentage points and the error rates are now comparable to Klapuri's error rates. Use of

Table 6.2 JSE error rates with and without antisalience

# of voices	Herman Direct Approach	Herman JSE w/out Anti-salience	Herman JSE with Anti-salience	% decrease in error rate %	Klapuri JSE
1	3.5%	3.5%	1.2%	65.7%	3.5%
2	16.0%	9.8%	5.3%	45.6%	5.0%
3	22.9%	17.0%	11.1%	34.7%	-
4	29.7%	20.8%	15.5%	25.5%	12.0%
6	35.9%	28.1%	25.7%	8.5%	21.0%

antisalience even proved useful for monophonic F0 estimation where the lowest error rates for any monophonic F0 estimator tested so far were obtained.

However, error rate improvements obtained by using antisalience diminished as the number of voices increased. This decrease is disappointing but is not too surprising. The strength of antisalience is that it tends to stop large blocks of partials from being excluded from F0 estimation. If an octave error is made when performing monophonic F0 estimation, over half of the partials of the correct F0 will be missed by the predicted spectrum thereby creating a large antisalience penalty. However, if six harmonic spectra are being used to explain a spectrum, it is much less likely that all six spectra will miss large blocks of partials, as six spectra should provide a fairly diverse coverage of all frequencies. On the other hand, inharmonicities, which may occur in some individual instrument spectra, only aggravate the problem by causing some partials to move away from their predicted locations. Since the number of inharmonicities would increase with the addition of more voices, even the correct set of F0s may have a stiff penalty against its selection. When the number of voices is high, antisalience is no longer sufficient to distinguish between partials in the remainder spectrum due to errors and inharmonic partials in the remainder spectrum.

6.4 Median Filtering

The use of median filters to smooth F0 estimation outliers in pitch tracks can also provide significant error rate reductions, as shown in Table 6.3. The modest decreases in error rates compared to those offered by antisalience is because median filters only fix errors caused by rapidly changing elements of a sound, such as the beating spikes shown in Fig. 5.3. These types of errors occur in most sounds, but only a handful of frames are affected by these errors. Median filtering only fixes these rare errors, while antisalience can fix some of these errors as well as octave errors which are much more common.

Table 6.3 JSE error rates with and without median filtering

# of voices	Herman JSE w/out Median Filters	Herman JSE with Median Filters	% decrease in error rate %
1	1.2%	1.2%	0.0%
2	5.3%	4.8%	9.4%
3	11.1%	9.8%	11.7%
4	15.5%	14.6%	5.8%
6	25.7%	24.2%	5.8%

6.5 Beating Compensation

Beating compensation applied to the basic JSE algorithm with and without antisalience applied provided modest performance improvements compared to the basic algorithm but had almost no effect when antisalience was used (see Table 6.4). From examination of corrections made by beating compensation and corrections made by antisalience, it was found that antisalience corrected most of the same mistakes that

beating compensation corrected. Because of this observation and the attendant high computational costs, beating compensation was not used in the final algorithm.

Table 6.4 JSE error rates using beating compensation

# of voices	Basic JSE Algorithm	JSE w/out Beating Comp. w/ Antisalience	JSE w/ Beating Comp. w/out Antisalience	% JSE w/ Beating Comp. w/ Antisalience
1	3.5%	1.2%	3.5%	1.2%
2	9.8%	5.3%	8.7%	5.2%
3	17.0%	11.1%	15.3%	11.1%
4	20.8%	15.5%	18.1%	15.1%
6	28.1%	25.7%	28.1%	25.6%

6.6 Time Versus Accuracy Scaling

Normalized computation time and error rate percentages for the direct approach were used as a baseline for the evaluation of the JSE algorithm. The JSE algorithm was tested with $J = P$, $2P$, $5P$, and $10P$, where P = number of voices.

In Fig. 6.1 the computation times for the direct approach and for each value of J tested are normalized to the computation time required for the direct approach. An approximately quadratic growth in computation time can be seen as J increases.

Figure 6.2 presents error-rate percentages for the direct approach as well as for each value of J tested. Decreases in error rate percentage are most dramatic when increasing from $J = P$ to $J = 2P$. Less dramatic improvements in error rate percentage are gained when increasing from $J = 2P$ to $J = 5P$, and negligible improvements are gained between $J = 5P$ and $J = 10P$ (except for 6 voices).

If it is desired to keep computational costs to a minimum while maintaining ac-

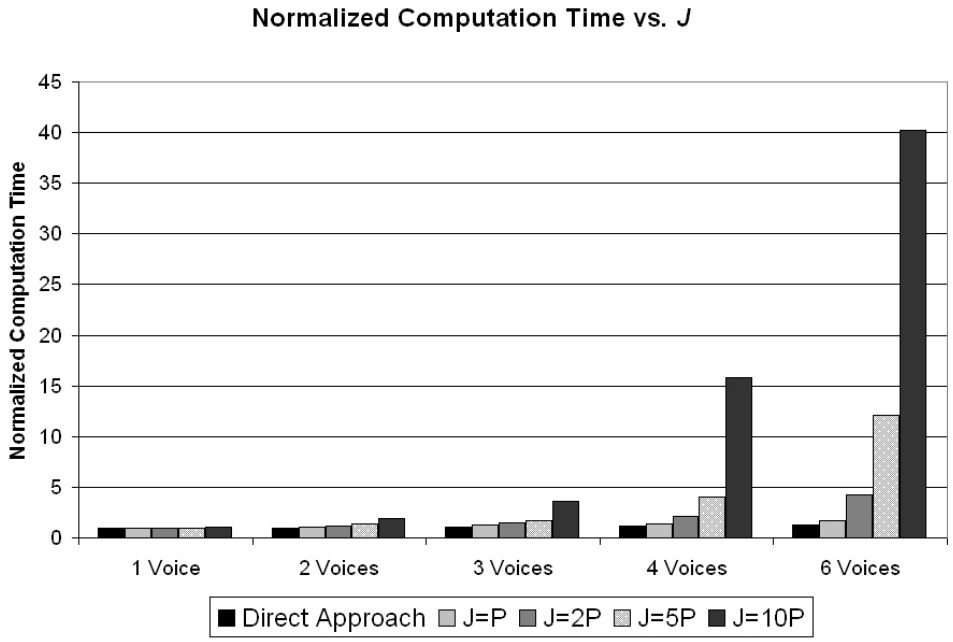

Figure 6.1 Normalized computation time vs. J.

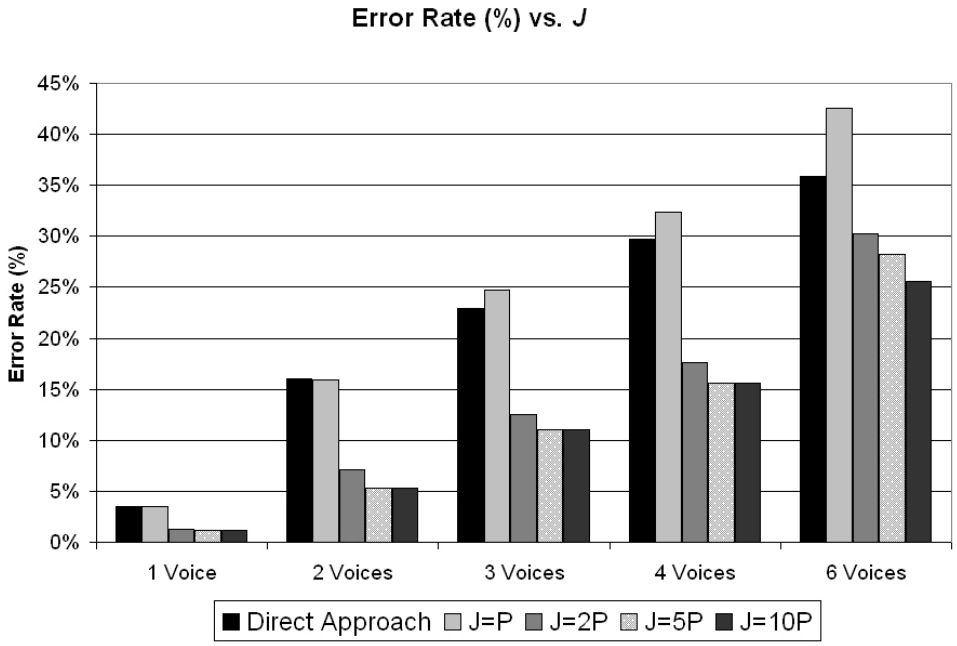

Figure 6.2 Error rate (%) vs. J.

ceptable accuracy, it is best to use $J = 2P$. If accuracy is more important than computational costs, $J = 5P$ should be sufficient for most applications, unless P is large (i.e., $P \geq 6$).

The large increase in accuracy using $J = 2P$ instead of $J = P$ is because moving from $J = P$ to $J = 2P$ allows each true candidate F0 and one spurious candidate F0 to be present in the initial selection pool of F0s. For the most part these spurious candidates are F0s that are in octave relationships with the true F0s or peaks that correspond to the largest partials in the spectrum. The modest increase in accuracy between $J = 2P$ and $J = 5P$ seems to be a result of the times when more than one spurious F0 candidate is present per true F0. Moving to $J = 10P$ does not provide much improvement, if at all, which indicates that adding more initial candidates beyond $J = 5P$ is not productive, and the remainder of the errors can only be eliminated by improving the JSE algorithm.

6.7 Rapidly Changing Note Mixtures

To test the robustness of the JSE algorithm against transients, error rate data was collected from recordings with sustained chords and with rapidly changing notes separately. Random mixes of the Iowa Theremin database [25] sounds were used for the sustained sound examples, and mixes of the University of Illinois Computer Music Project sounds were used for the rapidly changing note examples.

In Table 6.5 it can be seen that the error rates for rapidly changing notes are not significantly greater than the error rates for sustained notes. This error rate stability indicates that the joint subtraction algorithm is robust enough to handle transients well.

JSE-estimated pitch tracks of a trio consisting of an alto saxophone, a trombone, and a clarinet are shown in Fig. 6.3. The ground truths for the sounds are displayed

Table 6.5 JSE error rates when analyzing stationary notes and rapidly changing notes

# of voices	Stationary Note Error Rate %	Rapidly Changing Note Error Rate %
1	1.1%	1.2%
2	4.6%	5.1%
3	9.7%	10.9%
4	14.4%	15.1%

using thin lines and each frame's $\hat{F}0$s (in MIDI pitch units) are displayed as thick lines. While no specific mistakes were observed to be more problematic with changing notes than with sustained notes, it was observed that the JSE was able to handle the temporary withdrawal of a voice better than expected. This observation can be seen in Fig. 6.3 between frames 250 and 400 where one voice in the mix drops out. It was found that although P still equaled three during these frames (and therefore we should expect to see two correct $\hat{F}0$s and one spurious $\hat{F}0$) only two unique $\hat{F}0$s were identified in most of these frames. In these cases the JSE normally selects one correct $\hat{F}0$ twice, rather than select a spurious $\hat{F}0$. The duplicate selection of a correct $\hat{F}0$ when a voice is missing is fortunate, because the pitch tracks still accurately reflect the ground truth even when the estimated number of voices is wrong. It also indicates that the JSE is robust against overestimations of P.

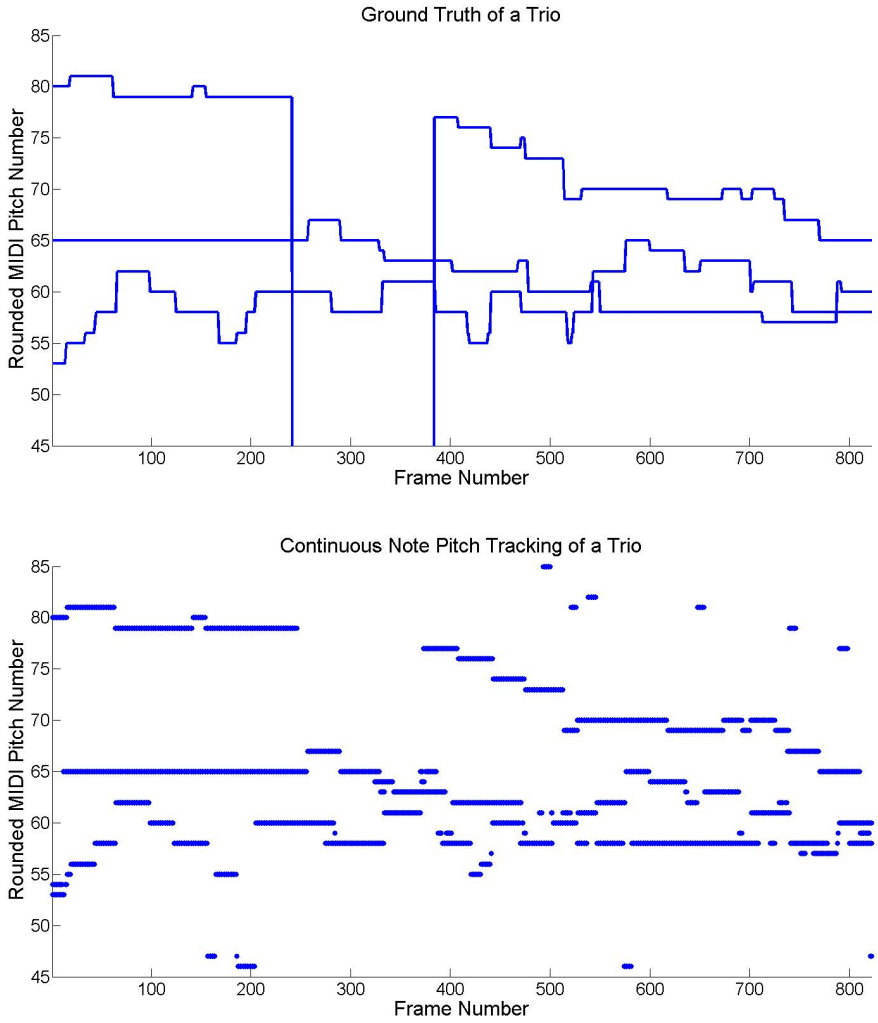

Figure 6.3 Continuous note pitch tracking of a trio mix where the tracks are (1) upper: alto saxophone; (2) middle (beginning): trombone; (3) lower (beginning): clarinet; (4) middle (after frame 550): clarinet; (5) lower (after frame 550): trombone. Ground truth, based on separate pitch tracking of the individual voices, is plotted with a thin line (upper graph) and the estimated pitch tracks are plotted with thick lines (lower graph). Pitches are rounded to the nearest semitone (1 MIDI pitch unit).

CHAPTER 7

CONCLUSIONS

A strong case for the use of Joint-Subtraction-based F0 estimators can be made through this research. All of the author's original hypotheses concerning JSE in Sec. 5.4 were found to be true:

1. Basic JSE is more accurate than either ISE or the direct approach.

2. Computationally efficient methods of JSE can be implemented even with large values of the J parameter.

3. Trade-offs between computation time and accuracy can be made by varying the J parameter.

4. A simple yet powerful error correction technique called antisalience provides a substantial error-rate improvement.

When compared with median filtering and beating compensation, antisalience is the most powerful error correction technique available to JSE. Antisalience is capable of correcting most of the mistakes that can be corrected by either beating compensation or median filtering, yet is still computationally inexpensive. The power of antisalience can be seen by comparing Fig. 5.3 and Fig. 7.1. By simply adding antisalience to the JSE, all but one error in the pitch tracks is corrected. Antisalience is also adaptable and can be applied to any algorithm that calculates the saliences of sets of $\hat{F}0$s.

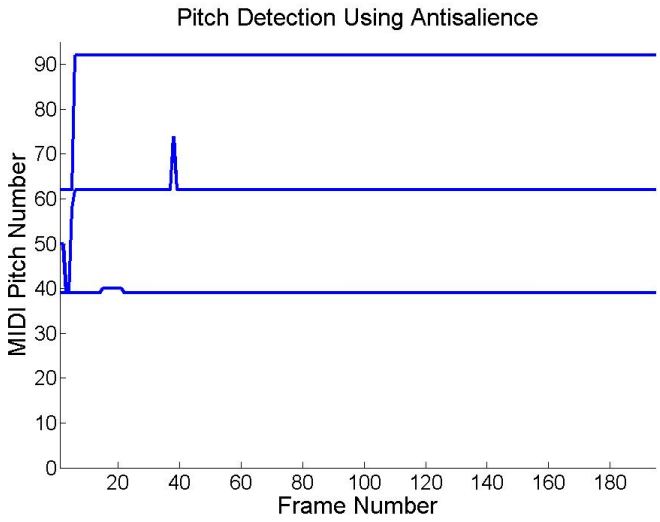

Figure 7.1 Pitch tracks for three-note chord corrected by antisalience.

REFERENCES

[1] D. Sasha, "NYU query by humming," January 2005. [Online]. Available: http://querybyhum.cs.nyu.edu/.

[2] A. deCheveigné and H. Kawahara, *Comparative Evaluation of F0 Estimation Algorithms.* Copenhagen, Denmark: Eurospeech, 2001.

[3] H. Fletcher, "Auditory patterns," *Reviews of Modern Physics*, vol. 12, pp. 47–65, 1940.

[4] ANSI, *Psychoacoustical Terminology.* New York, NY: American National Standards Institute, 1973.

[5] W. M. Hartmann, "Pitch, periodicity, and auditory organization," *J. Acoust. Soc. Am.*, vol. 100, no. 6, pp. 3491–3502, 1996.

[6] A. J. Ellis and A. J. Hipkins, "Tonometrical observations on some existing non-harmonic musical scales," in *Proceedings of the Royal Society of London*, vol. 37, 1884, pp. 368–385.

[7] S. S. Stevens, *Psychophysics.* New York, NY: John Wiley & Sons, 1975.

[8] R. C. Maher and J. W. Beauchamp, "Fundamental frequency estimation of musical signals using a two-way mismatch procedure," *J. Acoust. Soc. Am.*, vol. 95, no. 4, pp. 2254–2263, 1994.

[9] J. G. Roederer, *Introduction to the Physics and Psychophysics of Music*, second edition ed. New York, NY: Springer-Verlag, 1975.

[10] L. R. Rabiner, M. J. Cheng, A. E. Rosenberg, and C. A. McGonegal, "A comparative performance study of several pitch detection algorithms," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-24, no. 5, pp. 399–418, 1976.

[11] W. J. Hess, "Pitch and voicing determination," in *Advances in Speech Signal Processing*, S. Furui and M. M. Sondhi, Eds. New York, NY: Marcel Dekker, 1991.

[12] D. Gerhard, *Pitch Extraction and Fundamental Frequency: History and Current Techniques.* Regina, Sask.: University of Regina, November 2003.

[13] A. P. Klapuri, "Signal processing methods for the automatic transcription of music," Ph.D. dissertation, Tampere University of Technology, Tampere, Finland, 2004.

[14] M. Lahat, R. J. Niederjohn, and D. A. Krubsack, "A spectral autocorrelation method for measurement of the fundamental frequency of noise-corrupted speech," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-35, no. 6, pp. 741–750, 1987.

[15] N. Kunieda, T. Shimamura, and J. Suzuki, "Robust method of measurement of fundamental frequency by ACLOS - autocorrelation of log spectrum," in *Proc. IEEE International Conf. on Acoust., Speech, and Signal Processing*, 1996, pp. 232–235.

[16] T. Tolonen and M. Karjalainen, "A computationally efficient multipitch analysis model," *IEEE Trans. Speech and Audio Processing*, vol. 8, no. 6, pp. 708–716, 2000.

[17] R. Meddis and L. O'Mard, "A unitary model of pitch perception," *J. Acoust. Soc. Am.*, vol. 102, no. 3, pp. 1811–1820, 1997.

[18] R. Meddis and M. J. Hewitt, "Virtual pitch and phase sensitivity of a computer model of the auditory periphery. I: Pitch identification," *J. Acoust. Soc. Am.*, vol. 89, no. 6, pp. 866–882, 1991.

[19] D. Talkin, "A robust algorithm for ptch tracking," in *Speech Coding and Synthesis*, W. B. Kleijn and K. K. Paliwal, Eds. Elseview Science B. V.

[20] J. C. Brown and B. Zhang, "Musical frequency tracking using the methods of conventional and "narrowed" autocorrelation," *J. Acoust. Soc. Am.*, vol. 89, no. 5, pp. 2346–2354, 1991.

[21] A. M. Noll, "Short-time spectrum and 'cepstrum' technique for vocal-pitch detection," *J. Acoust. Soc. Am.*, vol. 36, pp. 296–302, 1964.

[22] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *J. Acoust. Soc. Amer.*, vol. 111, no. 4, pp. 1917–1930, April 2002.

[23] X. Rodet and B. Doval, "Estimation of fundamental frequency of musical sound signals," in *Proc. IEEE International Conf. on Acoust., Speech, and Signal Processing*, 1991.

[24] B. Doval and X. Rodet, "Fundamental frequency estimation and tracking using maximum likelihood harmonic matching and hmm's," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, 1993, pp. 221–224.

[25] L. Fritts, "The University of Iowa musical instrument samples," January 2005. [Online]. Available: http://theremin.music.uiowa.edu/MIS.html.

[26] A. P. Klapuri, "Automatic transcription of music," Master's thesis, Tampere University of Technology, Tampere, Finland, 1997.

[27] A. P. Klapuri, "Multiple fundamental frequency estimation by summing harmonic amplitudes," in *International Conference on Music Information Retrieval*, 2006.

[28] A. P. Klapuri, "Automatic transcription of music," *IEEE Trans. Speech and Audio Processing*, vol. 11, no. 6, pp. 804–816, November 2003.