ROBUST REALTIME POLYPHONIC PITCH DETECTION

by

John M. Thomas
A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Master of Science
Electrical Engineering

Robust Realtime Polyphonic Pitch Detection

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

John M. Thomas
Bachelor of Science
LeTourneau University, 2006

Director: Dr. Jill Nelson, Professor
Department of Electrical and Computer Engineering

Spring Semester 2012
George Mason University
Fairfax, VA

Dedication

This paper is dedicated to my dear, lovely wife, without whom I would be lost. Thank you for your endless support and understanding during the completion of this project.

Acknowledgments

I would like to acknowledge the brilliant authors – particularly T. Tolonen, M. Karjalainen, A. Klapuri, and Dessein et. al. – whose prior work provided the foundation for my own pitch detection algorithms. I would especially like to thank Klapuri for his wonderful book that provided so much useful background information.

I would also like to thank my advisor, Dr. Jill Nelson, for her support and direction throughout this process.

Table of Contents

List of Tables

List of Figures

Abstract

ROBUST REALTIME POLYPHONIC PITCH DETECTION

John M. Thomas, MS

George Mason University, 2012

Thesis Director: Dr. Jill Nelson

Pitch detection is a subset of automatic music transcription, which is the application of various signal processing algorithms with the specific intent of automatically gathering musical information from audio signals. This field, in various forms, has been the subject of much research over the years, as it has virtually endless possibilities for application. Much work has been done on monophonic signals, however, much less has been done to tackle the problem of polyphonic music.

One major issue for polyphonic pitch detection systems is efficiency. Most existing algorithms sacrifice efficiency for accuracy and robustness, while some others take the opposite tradeoff. The purpose of this paper is to work toward new systems that are both robust *and* efficient enough to run in realtime.

First, the relevant background information necessary to explore this topic is presented. Musical terminology and concepts are explained, and some common analytical tools and algorithms used in existing systems are described.

Three relatively efficient reference systems are then presented. The first is a multiple fundamental frequency (F_0) estimator based on the Auto-Correlation Function (ACF) that utilizes a unique enhancement algorithm to easily identify individual pitch components.

The second is a multiple F_0 estimator based on the Fast Fourier Transform (FFT) that exploits the harmonic nature of musical sounds.

The third reference system outputs directly to pitch numbers by using a modified form of an unsupervised learning algorithm called Non-Negative Matrix Factorization (NMF).

It is clear from investigation of the first two reference systems that the two opposing camps on fundamental frequency estimation (FFT vs. ACF) are actually quite complementary. Therefore, the remainder of the paper explores the inherent high-frequency versus low-frequency accuracy tradeoffs and proposes potential solutions. A novel analysis tool called the Combined ACF/FFT Representation (CAFR) is developed and three new pitch detection algorithms are devised from it. These algorithms are then evaluated for both robustness and efficiency and compared against results for the three reference systems.

Chapter 1: Background

1.1 Introduction

Pitch detection is a task from the field of automatic music transcription, which has been the subject of much research over the years, as it has virtually endless possibilities for application. Klapuri and Davy [1] have compiled an excellent overview of the topic, gathering past research and demonstrating the state of the art as of 2006. It is evident from this volume, that in the realm of pitched music transcription, much work has been done on monophonic signals. However, much less has been done to tackle the problem of polyphonic music. While there have been some more recent advances in the area of polyphonic pitch detection, there is still plenty of room for improvement, particularly in the realm of realtime detection systems. This chapter presents the relevant background information necessary to explore this topic and expand on previous research.

1.2 Musical Terminology and Concepts

Before delving into the problem at hand, it will be beneficial to define the relevant musical terminology and concepts. Music is, first and foremost, a phenomenon of human perception and can therefore be quite subjective in nature. However, there are certain over-arching principles commonly recognized by human listeners.

1.2.1 Low-Level Perception

There are two major categories of musical sounds – pitched sounds (notes) and un-pitched (or percussive) sounds. In the context of this paper, the discussion will be limited to pitched sounds, such as those created by typical musical instruments (piano, violin, flute, guitar, etc.).[1] Sounds created by inherently un-pitched instruments (drums, cymbals, etc.) will not be considered.

A *note* is a single, atomic sound event primarily described by four important perceptual qualities: pitch, loudness, duration, and timbre [2].

Pitch is analogous to frequency, perceived on a logarithmic scale, and provides for a natural ordering from “low” to “high.” Physically, pitch is determined by the fundamental frequency (F_0) of a periodic (or nearly periodic) sound [3].

The perceived *loudness* of a sound is related to its physical energy. Although there are more complicated psychoacoustic[2] factors involved, it is generally represented computationally as the mean-square power on a logarithmic (decibel) scale [3].

A note’s perceived *duration* is almost perfectly related to the physical duration of the sound. There may be some ambiguity in the case of slowly decaying sounds with no abrupt ending, but it can be assumed that a note is considered “live” as long as its loudness is above some threshold.

Timbre is the most subjective and least definable feature of a sound. It is often described as “color” and aids greatly in the recognition of different musical instruments. For instance, even if a piano and trumpet were to play with exactly the same frequency, loudness, and duration, a human listener could still easily determine the source instrument of each note. Timbre can also vary between instances of the same type of instrument or even between playing styles on a single instrument. Mathematically, timbre is largely related to spectral energy distribution and is multi-dimensional in nature [3].

1.2.2 High-Level Concepts

Beyond these low-level qualities, music also carries high-level information. This information is encoded in the *relationships* between individual sounds, which carry far more weight than

[1]These instruments also contain percussive elements – the piano is even grouped with the percussion section of an orchestra – but they are distinguished from regular percussion instruments by their production of a clear, sustained fundamental frequency (pitch).

[2]Psychoacoustics is the science of sound perception. It involves experimental study of how various physical stimuli relate to subjective human sensations [2].

the exact nature of the individual sounds themselves. Chords and melodies are comprised of pitch relationships, and timing relationships – particularly inter-onset intervals (IOIs)[3] – define rhythm. While sound durations may also convey rhythmic information, the IOIs play a much larger role. Duration serves more to control *articulation*, which varies between two extremes: "legato" notes, which seemingly flow from one to the next with no gap, and "staccato" notes, which are abruptly cut short. Loudness and timbre relationships also combine to help define musical structure and mood [3, 4].

A *chord* is a combination of simultaneously sounding notes (though not necessarily with simultaneous onsets) that are perceived as a single musical entity. Depending on the relationships between the individual notes comprising a chord, it may be subjectively described as either "harmonious" (pleasing) or "dissonant" (displeasing). A *melody* is comprised of the pitch and IOI relationships within a succession of individual notes and is generally the most defining and recognizable part of a musical composition [3].

In western music, notes are quantized on a logarithmic scale. There are 12 notes in each *octave*, with the frequency range of each successive octave being a factor of 2 greater than the previous octave. Conventionally, the notes within each octave are denoted as C, C#, D, D#, E, F, F#, G, G#, A, A#, and B.[4] The octave is then indicated with a number, such as A4 or F#5 [3].

To allow multiple instruments to play together (or "in concert"), it is necessary to define a reference pitch to which all instruments are tuned. This standard *concert pitch* is defined such that A4 (the A above "middle C") has a frequency of 440 Hz. Then the "equal-tempered" tuning of any note can be calculated as

$$f = 440\,\mathrm{Hz} \times 2^{\frac{p-69}{12}}, \tag{1.1}$$

where p is the "pitch number," which generally ranges from 0 to 127. The constant 69

[3]An inter-onset interval is the amount of time between the start of one sound event and another.

[4]Alternatively, the "sharp" (#) of the previous note in the series may be replaced by the "flat" (b) of the next note in the series: C, Db, D, E, F, Gb, G, Ab, A, Bb, B. These two notations are often used in combination according to certain patterns that depend on the "key" of a composition.

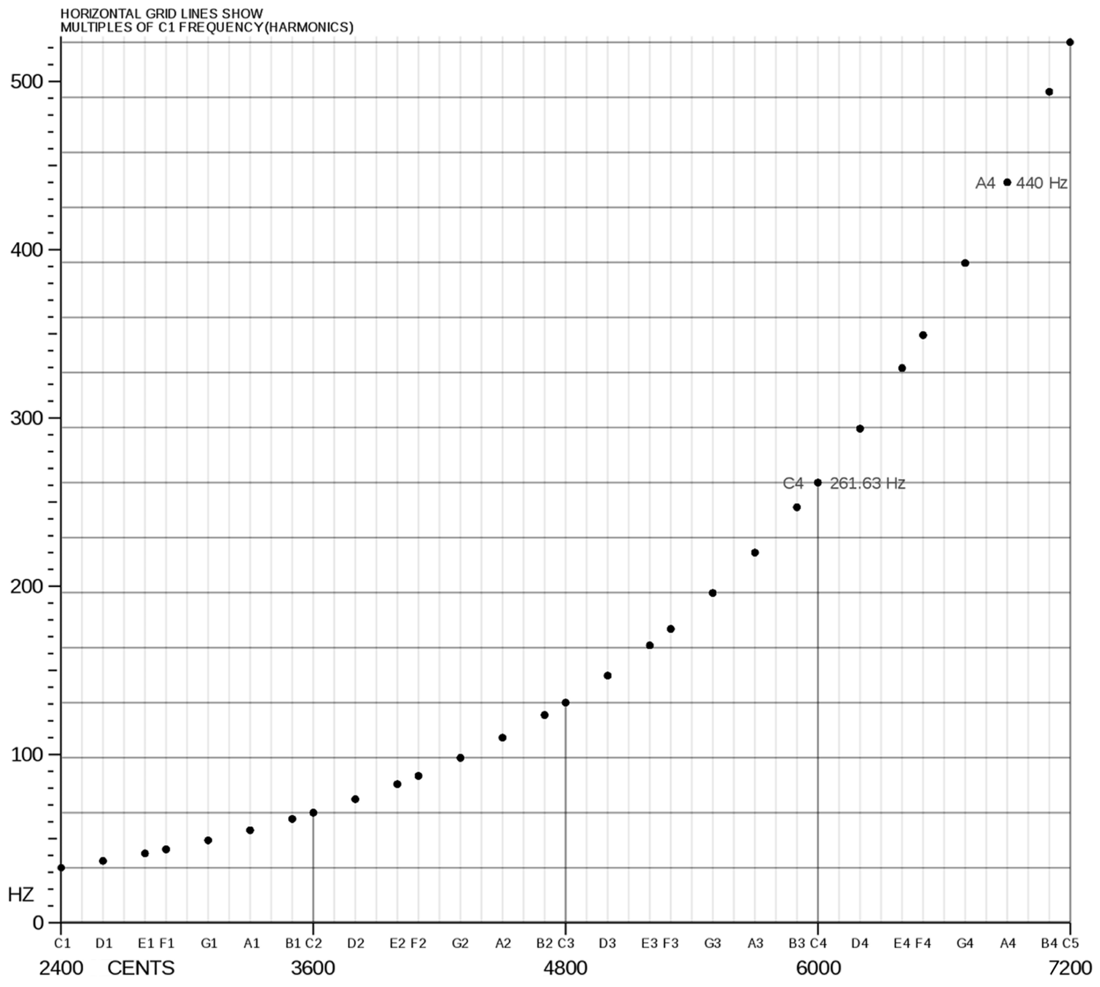

Figure 1.1: Frequency-to-Pitch Mapping. Derived from [6] (Public Domain).

is the pitch number of A4 in the MIDI[5] standard [5]. Likewise, a given frequency can be converted to its corresponding pitch by

$$p = 69 + 12 \log_2 \frac{f}{440\,\mathrm{Hz}}. \tag{1.2}$$

This simple mapping provides a convenient way of working with pitch algorithmically. Pitch numbers may either be quantized to integer values (which is generally done for final

[5]MIDI, the Musical Instrument Digital Interface, is a standard encoding for the exchange of musical performance data for electronic instruments and other digital applications [5].

output) or treated as continuous variables (as is often the case for intermediate representations). It is also common practice to round pitch values to the nearest hundredth and refer to the fractional part as “cents.” For instance, if a singer is slightly off pitch, singing at 444.1 Hz (instead of 440 Hz), one might say she is “16 cents sharp” of A4.[6] The full pitch scale may also be represented in cents, such that pitch 69 (A4) would be “6900 cents,” or half way between C3 and C#3 (48 and 49) would be “4850 cents.”

Humans perceive corresponding notes from different octaves as all belonging to the same “class.” While they are not perfectly interchangeable, the notes E3, E4, and E5, for example, serve much the same purpose. So all notes can be distilled into a set of 12 *pitch classes*, represented by note letter independent of octave number. In any given composition, only a certain subset of pitch classes – defined by the *musical key* – are used to create the melody, harmony, and other parts.[7] For instance, the key of “C major” would usually employ C, D, E, F, G, A, and B. Meanwhile, “B major” tends to use B, C#, D#, E, F#, G#, and A# [3].

1.2.3 Sound Structure

From a more technical perspective, it is important to understand the general structure of musical sounds. Notes have a clear structure in both the time and frequency domains.

The temporal structure of a note is defined by its *amplitude envelope*. Most notes begin with an *attack* period, optionally followed by a section of *sustain*, and are concluded with a *release* period [7]. As can be seen at the top of Figure 1.2, attack is the relatively short initial rise in amplitude, sustain is a period of roughly even near-maximum amplitude, and release is the final falloff to silence. For instruments that produce a clear sustain section (violin, trumpet, flute, etc.), that sustain provides the “meat” of the note. For non-sustained instruments (piano, guitar, harp, etc.), it is instead the release that defines the majority of the note.

[6]The term “sharp” is applied when the frequency is *higher* than some reference. The term “flat” refers to *lower* frequencies. If the singer had performed A4 as 436.7 Hz, she would be “13 cents flat.”

[7]This is obviously not a hard requirement, but the use of pitch classes that are *not* within the key is relatively infrequent.

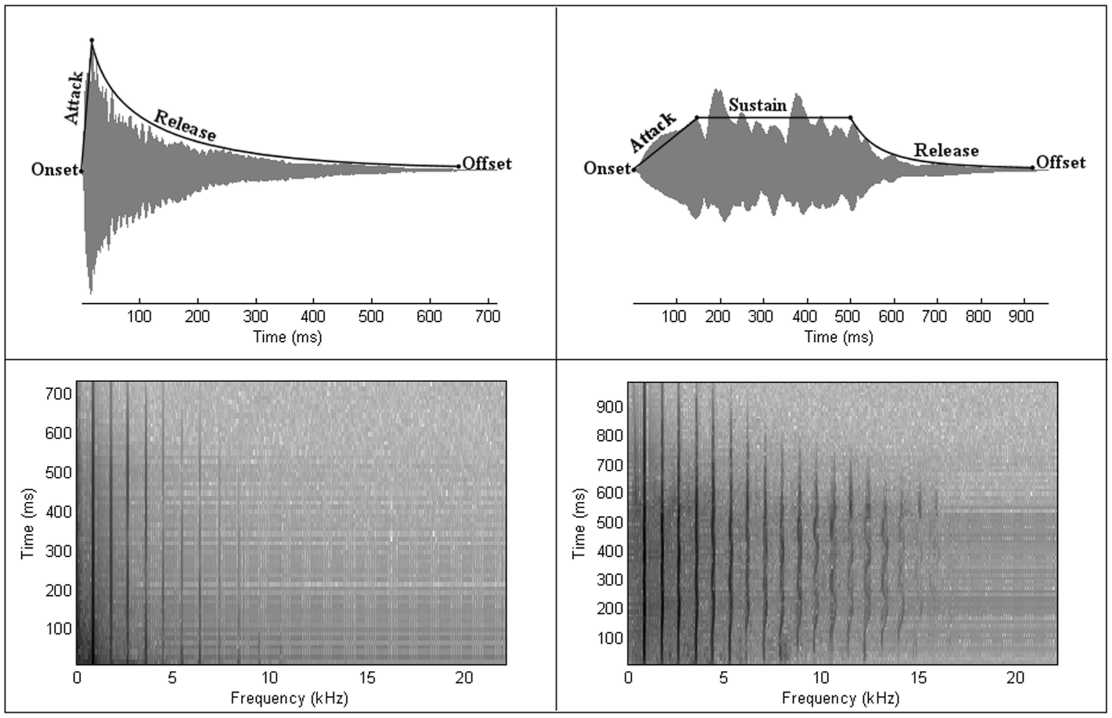

Figure 1.2: Temporal and Harmonic Structures. A single note of A5 played by a piano (left) and violin (right). Waveforms with simplified amplitude envelopes are shown in the top two panels, while the bottom two panels show spectrograms with clear harmonic structures.

In the frequency domain, notes have a clear *harmonic structure*. Most instruments produce sounds with strong frequency components at the note's fundamental frequency (F_0) as well as integer multiples of F_0. These harmonic frequency components generally have lower amplitudes than the F_0 component and gradually decrease for higher multiples. This structure can be seen clearly in the bottom two panels of Figure 1.2, where F_0 is 880 Hz. Some instruments, such as the piano, exhibit a certain degree of *inharmonicity*, in which the harmonic components are not exact integer multiples of F_0, but these variations are generally quite small and may often be ignored computationally [7–9].

1.3 Technical Concepts

Beyond the purely musical concepts described above, there are some technical concepts and mathematical techniques that are particularly applicable to the task of music transcription. Many of the tools used are common, such as the Fast Fourier Transform (FFT), Short-Time Fourier Transform (STFT), and Auto-Correlation Function (ACF), and it is assumed that the reader has a basic knowledge of these. However, some are less common and will be introduced here.

1.3.1 Cepstrum

A *cepstrum*[8] is defined as the Fourier transform of the logarithm of a power spectrum. While it was originally developed as a way of turning filtering into an additive[9] operation, it is often used as a signal analysis tool in the form of *cepstral coefficients*. Specifically, in the context of music processing, analysis may employ *mel frequency cepstral coefficients* (MFCCs) [10].

To compute the MFCCs of a signal $x(n)$, it is first windowed into frames $s^w_{n_0}(n)$ centered at time n_0. The magnitude spectrum $|S^w_{n_0}(k)|$ of each frame is then processed by a bank of K_{mel} (usually 40) mel frequency filters. Each filter, with center frequency k_n, has a triangular magnitude response extending from k_{n-1} to k_{n+1}. The center frequencies are evenly distributed along the mel frequency scale, which relates to the standard frequency scale as [10]

$$k_{mel} = 2595 \log_{10} \left[\frac{k}{700} + 1 \right]. \tag{1.3}$$

For each filter, the frequency components are weighted by the filter response, squared, and summed to produce $\chi_{n_0}(k_{mel})$, the full set of which are placed into a vector of length

[8]Pronounced as "kepstrum," this term derives its name from the reversal of the first four letters of "spectrum."

[9]Filtering, which can be performed as convolution in the time domain ($x(n) * h(n)$) or as multiplication in the frequency domain ($X(k)H(k)$), can ultimately become addition when the logarithm is taken ($\log[X(k)] + \log[H(k)]$).

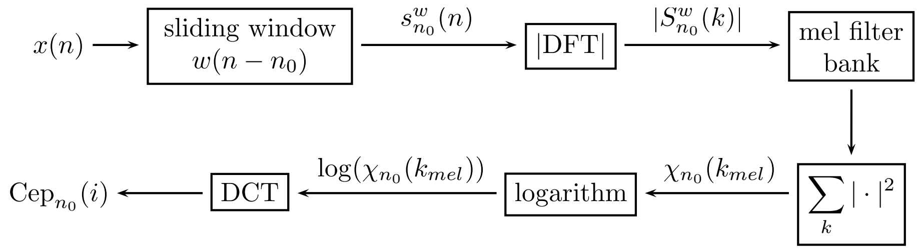

Figure 1.3: MFCC Computation Process

K_{mel}. The logarithm is then taken, and the vector is transformed back into the time lag domain with a discrete cosine transform (DCT), defined for a signal x of length T as [10]

$$\mathrm{DCT}_x(i) = \sum_{n=1}^{T} x(n) \cos\left[\frac{\pi}{T} i \left(n - \frac{1}{2}\right)\right]. \tag{1.4}$$

An overview of the MFCC computation process is shown in Figure 1.3. The final output is a set of K_{mel} individual time-domain signals whose amplitudes are proportional to their corresponding frequency components. In this way, MFCC could be considered a type of principal component analysis [10, 11].

The cepstrum can be a very useful tool for music processing, but it is not very robust against additive noise due to the logarithm, so there is often some form of normalization applied to the output. Some researchers have also proposed changing the basic algorithm to include raising the log-mel-amplitudes to some suitable power, such as 2 or 3, before taking the DCT [12].

1.3.2 Spectral Whitening

One difficulty in music transcription processing is dealing with different sound sources. A way of mitigating this problem is by attempting to suppress the sound's timbre, through a process called *spectral whitening*. Spectral whitening makes a rough estimate of the spectral energy distribution and uses inverse filtering to flatten it, either completely or partially.

There are several ways of achieving this, but one frequency-domain technique [13] that is easy to implement and generally produces good results is presented here.

To begin, the analysis frame is Hanning-windowed and zero-padded to twice its length, and its discrete Fourier transform $X(k)$ is calculated. To estimate the spectral distribution, a simulated bandpass filter bank is applied to the frequency bins. The filters are evenly distributed on the critical-band scale such that each filter has a center frequency,

$$c_b = 229\,\mathrm{Hz} \times (10^{(b+1)/21.4} - 1), \qquad b = 1, ..., 30, \tag{1.5}$$

and the triangular power response $H_b(k)$ of each sub-band extends from c_{b-1} to c_{b+1}. The standard deviation of each sub-band is then calculated as

$$\sigma_b = \sqrt{\frac{1}{K} \sum_k H_b(k) |X(k)|^2}, \tag{1.6}$$

where K is the padded analysis window length. A compression coefficient per sub-band is then calculated as $\gamma_b = \sigma_b^{\nu-1}$, where ν controls the amount of whitening to be applied. Klapuri [13] proposed $\nu = 0.33$ as a good value. These are then linearly interpolated to produce the final compression coefficients $\gamma(k)$ at each frequency bin k. The whitened spectrum $Y(k)$ is finally calculated $Y(k) = \gamma(k)X(k)$.

1.3.3 Non-Negative Matrix Factorization

Non-Negative Matrix Factorization (NMF) is an unsupervised multivariate data analysis model that revolves around the factorization of an $n \times m$ non-negative matrix $\mathbf{V}$ into constituent non-negative matrices $\mathbf{W}$ and $\mathbf{H}$ of rank $n \times r$ and $r \times m$ ($r < \min(n, m)$), respectively, such that

$$\mathbf{V} \approx \mathbf{WH}. \tag{1.7}$$

Each row in the source matrix $\mathbf{V}$ represents a single variable from the multivariate data set, while each column represents an observation. The factorized matrices $\mathbf{W}$ and $\mathbf{H}$ redefine $\mathbf{V}$ in terms of a decomposition into a basis set. The columns of $\mathbf{W}$ define the basis, while the columns of $\mathbf{H}$ are the decompositions for each observation [14].

To apply the NMF model, one must find the factorization that minimizes some given cost function. The standard implementation utilizes the Euclidean distance, requiring minimization of [14]

$$\frac{1}{2}||\mathbf{V} - \mathbf{W}\mathbf{H}||_F^2 = \frac{1}{2}\sum_j ||\mathbf{v}_j - \mathbf{W}\mathbf{h}_j||_2^2. \tag{1.8}$$

A popular method of computation in this case is the multiplicative updates procedure. At each step until convergence, the values of $\mathbf{H}$ and $\mathbf{W}$ are calculated as [14]

$$\mathbf{H} \leftarrow \mathbf{H} \otimes \frac{\mathbf{W}^{\mathrm{T}}\mathbf{V}}{\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{H}} \quad \text{and} \quad \mathbf{W} \leftarrow \mathbf{W} \otimes \frac{\mathbf{V}\mathbf{H}^{\mathrm{T}}}{\mathbf{W}\mathbf{H}\mathbf{H}^{\mathrm{T}}}. \tag{1.9}$$

In the context of music transcription, the data matrix $\mathbf{V}$ is generally some time-frequency representation of the audio signal. The rows represent frequency bins, and the columns represent analysis frames. The basis matrix $\mathbf{W}$ would then hold various spectral templates, while the $\mathbf{H}$ matrix holds decomposition coefficients indicating the relative presence of each template in each frame [14].

By nature, NMF can be quite computationally expensive and has traditionally been applied in off-line systems. However, it has also been used in on-line formulations to support real-time applications [14].

1.3.4 Hidden Markov Models

A Hidden Markov Model (HMM) is a statistical model in which the system is assumed to follow a Markov process with unobserved (hidden) states. In audio processing, HMMs are usually finite, discrete state-space dynamic models. A discrete random variable vector $\boldsymbol{\theta}_n$

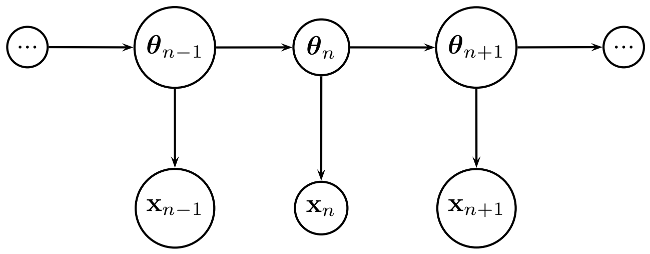

Figure 1.4: Progression of a Hidden Markov Model

lives in a finite state space Θ with E possible values ($\Theta = \{e_1, ..., e_E\}$). The system's time evolution (Fig. 1.4) follows a Markov process, such that the pdf of $\boldsymbol{\theta}_n$ is only dependent on the pdf of the previous state, $\boldsymbol{\theta}_{n-1}$. Only the output $\mathbf{x}_n$ is available at each step, as the state is hidden and cannot be observed directly. The model is then defined by the following properties [10].

1. State transition probabilities: $\mathbf{P}(\boldsymbol{\theta}_n = e_i|\boldsymbol{\theta}_{n-1} = e_j)$ for $(i, j) = 1, ..., E$
2. Observation likelihoods: $\mathbf{p}(\mathbf{x}_n|\boldsymbol{\theta}_n = e_i)$, $i = 1, ..., E$
3. Initial state probabilities: $\mathbf{P}(\boldsymbol{\theta}_1)$

As a classic example [15], consider the following situation. A room contains N large urns, each containing a large number of colored balls, of which there are M distinct colors. A genie in the room chooses an initial urn by some random process. The genie then takes a ball at random from the selected urn, records the color as an observation, and returns the ball to its urn. Then a new urn is selected based on a random process that depends only on the current urn. The genie takes a ball from this new earn, records its color, and returns it to the urn. The process continues in this fashion, producing a finite set of observations such as {red, blue, green, blue, red, red, green}.

This system can be modeled as an HMM in which the urns make up the state space ($\Theta = \{u_1, u_2, ..., u_N\}$), the initial urn choice is $\mathbf{P}(\boldsymbol{\theta}_1)$, the likelihood of a given color being drawn from the current urn is $\mathbf{p}(\mathbf{x}_n|\boldsymbol{\theta}_n = u_i)$, and the process of selecting a new urn is $\mathbf{P}(\boldsymbol{\theta}_n = u_i|\boldsymbol{\theta}_{n-1} = u_j)$.

The HMM is a powerful tool[10] that is often employed in temporal pattern recognition and other machine-learning applications. Transition probabilities and observation likelihoods may be learned from a set of training data, then the completed model is applied to new observations [10].

Once a model has been built, the task is to estimate a sequence of states $\boldsymbol{\theta}_{1:T}$ given a set of observations $\mathbf{x}_{1:T}$. The *Viterbi algorithm* performs a maximum a posteriori estimation to find this sequence such that [10]

$$\hat{\boldsymbol{\theta}}_{1:T} = \underset{\boldsymbol{\theta}_{1:T}}{\operatorname{argmax}}\ \mathbf{P}(\boldsymbol{\theta}_{1:T}|\mathbf{x}_{1:T}), \tag{1.10}$$

where the posterior probability of state sequence $\boldsymbol{\theta}_{1:T}$ is

$$\mathbf{P}(\boldsymbol{\theta}_{1:T}|\mathbf{x}_{1:T}) \propto \mathbf{P}(\boldsymbol{\theta}_{1:T}) \prod_{n=2}^{T} \mathbf{p}(\mathbf{x}_n|\boldsymbol{\theta}_n)\mathbf{P}(\boldsymbol{\theta}_n|\boldsymbol{\theta}_{n-1}). \tag{1.11}$$

The estimation process, which is also called *sequence decoding* in this context, is carried out via exhaustive search along all possible paths, but the Viterbi algorithm is quite efficient, having a complexity of $O(E^2T)$. The complete process is listed below [10].

1. <u>Initialization</u>: for $i = 1, ..., E$, set

$$w_1(e_i) = \mathbf{p}(\mathbf{x}_1|\boldsymbol{\theta}_1 = e_i)\mathbf{P}(\boldsymbol{\theta}_1 = e_i). \tag{1.12}$$

2. <u>Iterations</u>: for $n = 2, ..., T$,

    - For $i = 1, ..., E$, compute

$$w_n(e_i) = \mathbf{p}(\mathbf{x}_n|\boldsymbol{\theta}_n = e_i)\left[\max_{j=1,...,E} w_{n-1}(e_j)\mathbf{P}(\boldsymbol{\theta}_n = e_i|\boldsymbol{\theta}_{n-1} = e_j)\right]. \tag{1.13}$$

[10]The interested reader should refer to [15] for an excellent tutorial on HMMs and example applications in the field of speech recognition.

- For $i = 1, ..., E$, set

$$\psi_n(e_i) = \underset{e_j, j=1,...,E}{\operatorname{argmax}} w_{n-1}(e_j)\mathbf{P}(\boldsymbol{\theta}_n = e_i|\boldsymbol{\theta}_{n-1} = e_j). \tag{1.14}$$

3. <u>Termination</u>: compute

$$\hat{\boldsymbol{\theta}}_T = \max_{e_i, i=1,...,E} w_T(e_i). \tag{1.15}$$

4. <u>State Sequence Backtracking</u>: for $n = T-1, ..., 2, 1$, extract the estimate at time n by

$$\hat{\boldsymbol{\theta}}_n = \psi_n(\hat{\boldsymbol{\theta}}_{n+1}). \tag{1.16}$$

1.3.5 Auditory Model

The best music transcription system available today is the ears and brain of a trained musician. No artificial system has been able to match the human auditory system in its ability to handle complex acoustic signals and perceive the properties of multiple simultaneous sounds, including their individual pitches. This fact has led many researchers to approach the problem of music transcription from the perspective of an auditory model, particularly toward the development of *pitch perception models* [8].

The human auditory system can be split into two major sections. The first is the *peripheral hearing*, which consists of the outer, middle, and inner ear. The important part for pitch perception is contained in the inner ear in an organ called the *cochlea* [8].

The cochlea is a tapered, spiraling, tubular structure, filled with liquid, that is divided along its entire length by the *basilar membrane.* As sound waves reach the cochlea, the basilar membrane begins to vibrate. These vibrations propagate along the membrane in such a way that higher frequencies resonate near the beginning and lower frequencies resonate toward the end. The basilar membrane is covered in a forest of tiny hairs of varying lengths which aid in its varied frequency tuning. Special cells at the base of each hair, called *inner hair cells* (IHCs), convert the vibrations into neural impulses which are then carried to the brain [8].

Computational models of the cochlea first pass the audio signal through a bank of special bandpass filters called *auditory filters.* This auditory filter bank typically consists of about 100 filters with center frequencies distributed evenly on the critical band scale. This process models the frequency selectivity of different regions along the basilar membrane. The auditory filters belong to a special class of filters called *gammatone* filters, which are defined by their time-domain impulse response as [8]

$$g_c(t) = at^{n-1}e^{-2\pi bt} \times \cos(2\pi f_c t + \theta), \tag{1.17}$$

where $a = (2\pi b)^n/\Gamma(n)$ is a normalization factor ensuring a unity response at the center frequency f_c, $\Gamma(n)$ is the gamma[11] function, $n = 4$ is a parameter value that best matches the power response empirically with real auditory filters, and $b = 0.11 f_c + 27.39\,\text{Hz}$ controls the bandwidth [8].

The output of each band, or *auditory channel*, is then processed to model the conversion into neural impulses. Although more exact physical models have been developed, most practical systems perform the conversion via a three-stage process of compression, half-wave rectification, and lowpass filtering, which captures the major characteristics of the IHCs [8].

The compression step has been implemented in various ways, but a common approach is to scale the sub-band signals $x_c(n)$ by the inverse of their variances. This scaling has either been done to unity [16] or more generally [8] by a factor $\sigma_c^{\nu-1}$ $(0 \leq \nu \leq 1)$, where σ_c is the standard deviation of $x_c(n)$. This process resembles and produces results similar to spectral whitening, which has also been used in place of the compression step [17].

Half-wave rectification (HWR) is a clearly non-linear operation that plays an important

[11]The gamma function is similar in concept to the factorial function for integers, but it has been extended to also support real and complex numbers. For a positive integer, it reduces simply to $\Gamma(n) = (n-1)!$, but it is formally defined by the integral $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}\mathrm{d}t$, which converges for all complex numbers that have a positive real part.

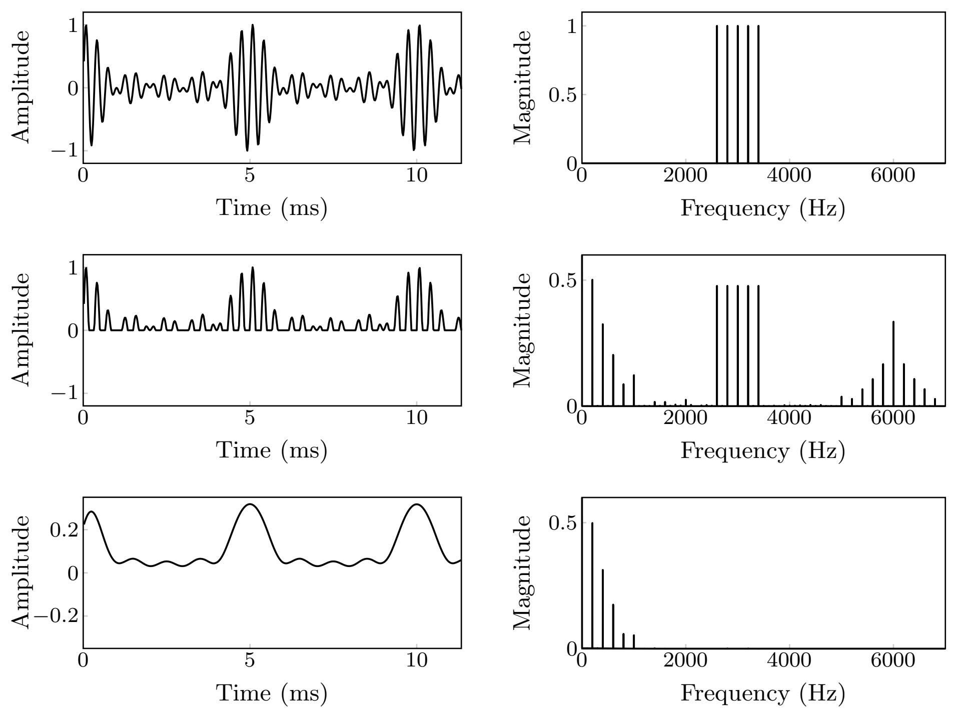

Figure 1.5: Effects of Half-Wave Rectification. The upper panels show a signal created with the harmonic partials 13-17 of a sound with $F_0 = 200\,\text{Hz}$. The middle panels show the result of half-wave rectification, and the lower panels show the rectified signal after lowpass filtering with a 1 kHz cut-off. Adapted from [8].

role in the conversion process. It is defined as

$$\text{HWR}(x) = \begin{cases} x, & x \geq 0, \\ 0, & x < 0. \end{cases} \tag{1.18}$$

This step is vital to pitch perception because it facilitates a synthesis of time-domain and frequency-domain periodicity analysis [8].

Figure 1.5 illustrates the effects of HWR on a narrow-band signal comprised of five harmonic overtone partials. The most important aspect of this process is that the rectification generates new spectral components corresponding to the *intervals* between the original partials. As shown in the lower panels, the components generated below 1 kHz form an amplitude envelope of the input signal. The amplitude envelope of any signal containing multiple tones exhibits periodic fluctuations, called *beating*, due to alternating constructive and destructive interference. The rate of this beating depends on the frequency difference between each pair of components. For harmonic sounds, the interval corresponding to F_0 will dominate. This makes it possible to determine the F_0 even when the input signal has no clear F_0 component [8].

The lowpass filtering stage balances the amplitude envelope versus the original input partials. In practice, most systems use a low-order filter on each channel with a cutoff around 1 kHz. The final output signal of each auditory channel after these three stages is denoted by $z_c(n)$ [8].

The second major section of the human auditory system is the *auditory cortex* of the brain. The exact pitch perception mechanisms of the auditory cortex are difficult to study and are not accurately known. However, experimentation has shown that some form of periodicity analysis is performed on each channel, then this information is combined across channels to provide the final perception [8, 18].

Various techniques of periodicity analysis have been proposed [19–21] such as a bank of comb filters, but most models use the ACF. In one approach [18], the ACF for each auditory channel is computed as

$$r_c(n,\tau) = \sum_{i=0}^{n} z_c(n-i) z_c(n-i-\tau) w(i), \tag{1.19}$$

where $w(i) = (1/\Omega)e^{-i/\Omega}$ is an exponentially decaying window function that emphasizes more recent samples. This information is then integrated across channels by simple summation, creating a summary ACF

$$s(n,\tau) = \sum_{c} r_c(n,\tau). \tag{1.20}$$

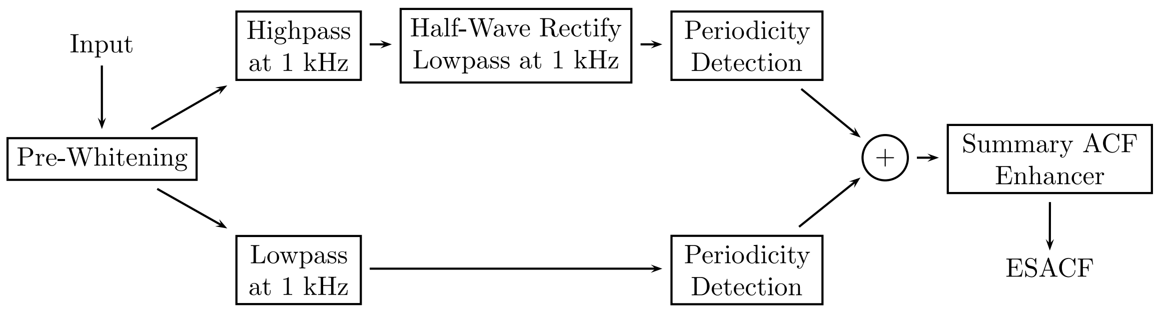

Figure 1.6: Block Diagram of Tolonen and Karjalainen's System

The pitch at time n is then found as the maximum of $s(n, \tau)$ within some range of τ.

1.4 Foundational Research

Polyphonic transcription is an area of active research, and many approaches have been developed. As this paper is primarily focused on realtime transcription, three computationally efficient systems of particular interest are described below. The first two are multiple F_0 estimators, which form the core of many polyphonic transcription systems. These would require a quantization stage to complete the pitch detection process. The third uses NMF to perform pitch detection in a single step. Some other types of approaches, such as the statistical model-driven system of Yeh and Röbel [22], have also shown promise but are not discussed here due to their complexity and computational inefficiency.

1.4.1 System of Tolonen and Karjalainen

Tolonen and Karjalainen proposed in [17] a more computationally efficient variation of the auditory model system introduced in Section 1.3.5. The primary difference between their system (Fig. 1.6) and the traditional auditory model is that it divides the input signal into just two channels – high and low, split at 1 kHz. This greatly reduces the computational load while still retaining many of the important auditory model characteristics.

To make the system more robust in handling different musical instruments, the input signal is subjected to inverse warped-linear-prediction filtering [23] – a form of spectral whitening, or *pre-whitening* in this case – to flatten the overall spectral energy distribution without altering its structural details.

For periodicity analysis, the system employs a *generalized ACF*, which is defined as

$$\hat{r}(\tau) = \text{IDFT}(|\text{DFT}(x)|^{\alpha}), \tag{1.21}$$

where α is a parameter that controls frequency domain compression. This yields the standard ACF for a value of $\alpha = 2$. It should be noted that this bears similarity to the cepstrum, which substitutes a logarithm for the power of α. In practice the ACF, with its power of 2, limits the influence of noise, but may also enhance spectral abberations. The cepstral analysis, with its logarithm, minimizes these abberations, but at the expense of increased noise sensitivity. The purpose of the generalized ACF is to introduce a compromise between the two extremes. For this system, the authors chose $\alpha = 0.67$ to bridge the gap.

Tolonen and Karjalainen achieved multiple F_0 estimation by creating an *enhanced summary ACF* (ESACF). Subharmonics are canceled from the SACF by clipping it to positive values, scaling it by an integral factor m, and subtracting this scaled version from the clipped version. The cancelation is performed for scaling factors $2 \leq m \leq 5$. The final F_0 selection is then accomplished by simple peak selection from the resultant ESACF. The complete enhancement process is detailed below.

1. Initialize the ESACF $\tilde{s}(\tau)$ to the SACF $s(\tau)$ and the scaling factor to $m = 2$.

2. Scale the original SACF by the factor m using linear interpolation.

$$s_m(\tau) = s(d) + \frac{\tau - md}{m}(s(d+1) - s(d)), \qquad d = \left\lfloor \frac{\tau}{m} \right\rfloor \tag{1.22}$$

3. Update the ESACF.

$$\tilde{s}(\tau) \leftarrow \max(0, \tilde{s}(\tau) - \max(0, s_m(\tau))) \tag{1.23}$$

4. Increment m. If $m < 6$, return to Step 2.

This process is relatively simple and elegant, yet it is strikingly effective in removing distracting information to reveal the F_0 peaks. Overall, it is quite accurate for lower-frequency signals, but it does not perform well above about 600 Hz. This derives from the relative inaccuracy of ACF measurements at high frequencies as well as a technical limitation of the ESACF in that components are effectively driven to zero[12] in the range $\tau \in [0, f_s/1000\,\text{Hz}]$, where f_s is the sampling frequency.

1.4.2 System of Klapuri

Klapuri presented a computationally efficient and conceptually tractable multiple F_0 estimation system in [13]. First, timbral information is suppressed via the spectral whitening method presented in Section 1.3.2. Then the *salience*, or strength, of a fundamental period candidate τ is calculated as

$$s(\tau) = \sum_{m=1}^{M} g(\tau, m)|Y(f_{\tau,m})|, \tag{1.24}$$

where $f_{\tau,m} = mf_s/\tau$ is the frequency of the mth harmonic of candidate $F_0 = f_s/\tau$, f_s is the sampling frequency, $g(\tau, m)$ is a weighting function, and $Y(f)$ is the STFT of the whitened input signal.

An optimal weighting function $g(\tau, m)$ was learned from a large set of training data. A parameterized version of the result was found to be of the form

$$g(\tau, m) = \frac{\dfrac{f_s}{\tau} + \alpha}{\dfrac{mf_s}{\tau} + \beta}, \tag{1.25}$$

[12]The ACF's "leading hill" (Fig. 2.2) is to blame here. The iterative scaling and cancelation process of the ESACF causes the leading hill to wipe out high frequency information. This could be corrected by simply removing the leading hill prior to ESACF processing, but the ACF bins being disrupted here are not very accurate anyway (as explored in the next chapter), so it is unlikely that this would result in any significant improvement.

with values optimized to $\alpha = 27\,\text{Hz}$ and $\beta = 320\,\text{Hz}$.

Direct computation of the salience using Equation 1.26 is inefficient due its evaluation of $Y(f)$ for arbitrary frequencies. So Klapuri approximated the calculation by using an FFT of length K and replacing $Y(f)$ with its discrete form $Y(k)$, such that

$$\hat{s}(\tau) = \sum_{m=1}^{M} g(\tau, m) \max_{k \in \kappa_{\tau,m}} |Y(k)|, \tag{1.26}$$

where $\kappa_{\tau,m}$ is a set of frequency bins around the mth harmonic of f_s/τ, defined as

$$\kappa_{\tau,m} = \left[\left\langle \frac{mK}{\tau + \frac{\Delta\tau}{2}} \right\rangle, \ldots, \left\langle \frac{mK}{\tau - \frac{\Delta\tau}{2}} \right\rangle \right], \tag{1.27}$$

where $\langle \cdot \rangle$ denotes rounding to the nearest integer and $\Delta\tau = 0.5$.

Multiple F_0 estimation is then performed by way of an iterative estimation and cancelation process. First, the global maximum $\hat{\tau}$ of $\hat{s}(\tau)$ is found using a fast algorithm (see [13] for details), then the weighted harmonic spectrum of $\hat{\tau}$ is subtracted from $Y(k)$ to form a residual spectrum $Y_R(k)$. The process is then repeated, starting from $Y_R(k)$, until the the estimated polyphony has been reached. This polyphony estimation is judged at each iteration j until the detected fundamental $\hat{\tau}_j$ no longer increases the value of

$$S(j) = \frac{1}{j^{\gamma}} \sum_{i=1}^{j} \hat{s}(\hat{\tau}_i), \tag{1.28}$$

where γ was found empirically to be 0.70. Then the value of j that maximizes Equation 1.28 is the estimated polyphony.

This system was found to perform quite well, though it did suffer from accuracy problems with low-frequency fundamentals. The polyphony estimation accurately detected lower polyphonies, but seemed to struggle for higher numbers of concurrent sounds.

1.4.3 System of Dessein, Cont, and Lemaitre

Presented in [14], the real-time system of Dessein, Cont, and Lemaitre directly estimates the constituent notes of polyphonic music signals via NMF and a β-divergence cost function. Off-line, NMF is first used to learn a set of spectral templates, one for each possible note. Specifically, the templates are learned from simple short-time magnitude spectra, each with a frame size of 50 ms (630 samples) at a sampling rate of 12 600 Hz, zero-padded and computed with a 1024-bin FFT. These templates are then used in an on-line setup to decompose the input signal into this basis set. Finally, note selection is performed by a simple thresholding operation on the decomposition output followed by minimum-duration pruning.

Briefly, the β-divergences are a family of parametric distortion functions defined such that, for any $\beta \in \mathbb{R}$ and any points $x, y \in \mathbb{R}_{++}$ (strictly-positive real numbers),

$$d_\beta(x|y) = \frac{1}{\beta(\beta-1)}(x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1}). \tag{1.29}$$

These functions are all non-negative and become zero if and only if $x = y$. An interesting property within the context of this work is that for any factor $\lambda \in \mathbb{R}_{++}$,

$$d_\beta(\lambda x|\lambda y) = \lambda^\beta d_\beta(x|y). \tag{1.30}$$

This fact led the authors to an intuitive understanding of the effect that different β values have on a musical NMF formulation. When $\beta = 0$, high-energy and low-energy frequency components are given the same importance. As β increases from 0, the higher-energy components are given more importance. The lower-energy components become more important as β decreases from 0. The value $\beta = 0.5$ was chosen as a compromise to reduce common harmonic and octave errors.

A reformulation of the standard NMF problem with β-divergence yields a minimization (subject to non-negativity of $\mathbf{W}$ and $\mathbf{H}$) of the cost function

$$D_\beta(\mathbf{V}|\mathbf{WH}) = \sum_{i,j} d_\beta(v_{ij}|[\mathbf{WH}]_{ij}), \tag{1.31}$$

which is equivalent to the standard formulation (Eq. 1.8) when $\beta = 2$. For the on-line decomposition process, $\mathbf{W}$ is a fixed set of note templates, and the incoming signal $\mathbf{v}$ is a single observation vector. This allows a slight simplification of the problem to

$$D_\beta(\mathbf{v}|\mathbf{Wh}) = \sum_{i} d_\beta(v_i|[\mathbf{Wh}]_i). \tag{1.32}$$

A modified form of the multiplicative updates procedure (Eq. 1.9) is used to perform the calculation. At each iteration until convergence, $\mathbf{h}$ is updated as

$$\mathbf{h} \leftarrow \mathbf{h} \otimes \frac{(\mathbf{W} \otimes (\mathbf{v}\mathbf{e}^{\mathrm{T}}))^{\mathrm{T}}(\mathbf{Wh})^{\cdot(\beta-2)}}{\mathbf{W}^{\mathrm{T}}(\mathbf{Wh})^{\cdot(\beta-1)}}, \tag{1.33}$$

where $\mathbf{e}$ is a vector of ones. Note that the matrix $\mathbf{W} \otimes (\mathbf{v}\mathbf{e}^{\mathrm{T}}))^{\mathrm{T}}$ only needs to be calculated once per analysis frame, allowing for a more efficient implementation.

Under an objective evaluation, this real-time system demonstrated accuracy comparable to that of two state-of-the-art off-line systems. Subjectively, the system tends to spuriously report extra notes at transients. It also often shortens the duration of detected notes. The latter error is likely due to the inherent assumption of a stationary signal. This is obviously incorrect, as the spectral components within a single note instance evolve over time. The authors speculated that this difficulty could potentially be overcome by expanding the spectral template dictionary to include multiple templates per note, each representing a different phase of the note's life-cycle.

1.5 Conclusion

Much progress has been made on the problem of polyphonic music transcription, but it is far from solved. There are still many difficult issues to overcome, especially as the polyphony increases. It appears particularly difficult to devise a robust system that is also capable of realtime performance. From the evaluation of existing work on the subject, it is clear that each approach has its advantages and disadvantages, and there are always compromises to be made.

Of particular interest are the high-frequency versus low-frequency accuracy tradeoffs inherent to the two major categories of periodicity analysis. Time-lag-domain analysis via the ACF has excellent accuracy at lower frequencies but relatively low resolution for higher frequencies. Frequency-domain analysis via the FFT has exactly the opposite problem. This observation leads to the simple conclusion that some marriage of the two should be pursued in attempt to achieve high accuracy in all cases.

The remainder of this paper explores these compromises and tradeoffs and proposes potential solutions to these problems. A novel analysis tool called the Combined ACF/FFT Representation (CAFR) is developed and three new pitch detection algorithms are devised from it. These algorithms are then evaluated for both robustness and efficiency and compared against results for the three reference systems.

Chapter 2: A New Analysis Tool

2.1 Introduction

A great many – perhaps most – music transcription algorithms rely to some extent on picking peaks from the output of an analysis tool such as the FFT or ACF. This process is inherently error prone due to the discrete nature of these tools and their accompanying quantization effects. It is thus useful to study the error incurred by the peak-picking process.

2.2 Error Measurement

For the purposes of pitch detection, the most import metric to consider is the relative difference of an estimated frequency as compared to its true value. For a given frequency estimate, $\hat{f}$, the relative estimation error can be measured as

$$\varepsilon = \frac{|f - \hat{f}|}{f}. \tag{2.1}$$

In order to resolve pitch at the semitone level, the estimate must have at least quarter-tone accuracy. This places an upper bound on the acceptable error. Generalizing Equation 1.1, pitch frequencies are of the form

$$f(p) = f_r \cdot 2^{\frac{p-r}{N}}, \tag{2.2}$$

where f_r is the tuning reference frequency, p is the pitch number, r is the reference pitch number, and N is the number of divisions per octave. So the relative difference between a given pitch and its next lower neighbor can be found by

$$\begin{aligned}
\alpha_- &= \frac{f_r \cdot 2^{\frac{p-r}{N}} - f_r \cdot 2^{\frac{p-1-r}{N}}}{f_r \cdot 2^{\frac{p-r}{N}}} \\
&= \left(2^{\frac{p-r}{N}} - 2^{\frac{p-1-r}{N}}\right) \cdot 2^{\frac{r-p}{N}} \\
&= 2^{\frac{p-r+r-p}{N}} - 2^{\frac{p-1-r+r-p}{N}} \\
&= 1 - 2^{-\frac{1}{N}}.
\end{aligned} \tag{2.3}$$

Similarly, the relative error between a pitch and its next higher neighbor is found by

$$\begin{aligned}
\alpha_+ &= \frac{f_r \cdot 2^{\frac{p+1-r}{N}} - f_r \cdot 2^{\frac{p-r}{N}}}{f_r \cdot 2^{\frac{p-r}{N}}} \\
&= \left(2^{\frac{p+1-r}{N}} - 2^{\frac{p-r}{N}}\right) \cdot 2^{\frac{r-p}{N}} \\
&= 2^{\frac{p+1-r+r-p}{N}} - 2^{\frac{p-r+r-p}{N}} \\
&= 2^{\frac{1}{N}} - 1.
\end{aligned} \tag{2.4}$$

It is easily shown that $\alpha_+ > \alpha_-$ for all N, which is a logical finding considering the exponential nature of pitch frequencies. Substituting $N = 24$ into α_- then yields the upper bound on error for quarter-tone accuracy:

$$\alpha_q = 1 - 2^{-\frac{1}{24}} \approx 0.0285. \tag{2.5}$$

2.3 FFT Analysis

The bins of an FFT are linearly spaced in frequency. For a given frequency, the FFT peak-picked estimate is determined by the nearest bin into which it will fall, which can be expressed as

$$\hat{f} = \left\langle \frac{fK}{f_s} \right\rangle \cdot \frac{f_s}{K}. \tag{2.6}$$

Substituting this into Equation 2.2, it is then possible to calculate the relative error over the full frequency range given the sampling frequency, f_s, and FFT length, K, as

$$\varepsilon_{FFT}(f) = \frac{\left| f - \left\langle \frac{fK}{f_s} \right\rangle \cdot \frac{f_s}{K} \right|}{f}. \tag{2.7}$$

This exact error measurement can be generalized into a bounding curve defined by the maximum difference between the estimated and true frequencies:

$$\tilde{\varepsilon}_{FFT}(f) = \frac{\Delta_{max}}{f}. \tag{2.8}$$

The maximum difference occurs at the half-way point between two successive frequency bins, $\Delta_{max} = \frac{f_s}{2K}$, thus the FFT's error bound is found to be

$$\tilde{\varepsilon}_{FFT}(f) = \frac{f_s}{2Kf}. \tag{2.9}$$

2.4 ACF Analysis

A parallel analysis can be performed on the ACF, whose bins are linearly spaced in time-lag. The relative error in terms of time-lag is defined as

$$\varepsilon_{ACF}(\tau) = \frac{|\tau - \hat{\tau}|}{\tau} \tag{2.10}$$

The ACF estimate for an arbitrary time-lag value is simply $\langle \tau \rangle$. This can then be converted into frequency space by the relation $\tau = \frac{f_s}{f}$, yielding

$$\varepsilon_{ACF}(f) = \frac{f}{f_s} \cdot \left| \frac{f_s}{f} - \left\langle \frac{f_s}{f} \right\rangle \right|. \tag{2.11}$$

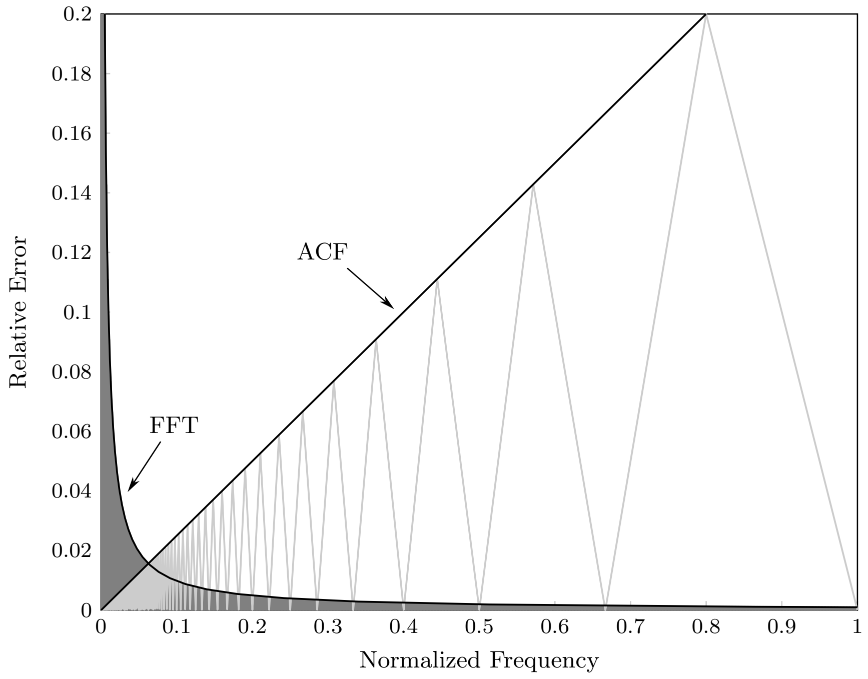

Figure 2.1: Measurement Error Analysis of ACF and FFT. Bounding curves are shown along with exact error for each analysis tool. The ACF's error increases linearly, while the FFT's error is inversely-related to frequency. For this illustration, the analysis frame length is set to $K = 1024$.

To find a bounding curve, note that the maximum error occurs at the half-way point between two time-lag bins, such that $\Delta_{max} = \frac{1}{2}$. So in terms of τ, the error is bound by

$$\tilde{\varepsilon}_{ACF}(\tau) = \frac{1}{2\tau}, \tag{2.12}$$

and substituting back into frequency space, the counterpart to Equation 2.9 is found to be

$$\tilde{\varepsilon}_{ACF}(f) = \frac{f}{2f_s}. \tag{2.13}$$

2.5 The Optimal Blend

Intuitively, it is clear that the FFT should produce relatively accurate estimates for higher frequencies and conversely inaccurate estimates for lower frequencies, while the ACF has the opposite problem. This intuitive understanding is confirmed and further clarified by the analyses above and illustrated in Figure 2.1.

Using the error bounding curves found in Equations 2.9 and 2.13, it is possible to calculate an optimal crossover frequency to minimize the estimation error:

$$\tilde{\varepsilon}_{FFT}(f_c) = \tilde{\varepsilon}_{ACF}(f_c)$$

$$\Rightarrow \frac{f_s}{2Kf_c} = \frac{f_c}{2f_s}$$

$$\Rightarrow f_c = \frac{f_s}{\sqrt{K}}. \tag{2.14}$$

The maximum relative error of this new optimally-blended analysis tool, which will henceforth be referred to as the Combined ACF/FFT Representation (CAFR), is then found as follows:

$$\varepsilon_{CAFR}(f) \leq \tilde{\varepsilon}_{ACF}(f_c) = \frac{f_s}{\sqrt{K}} \cdot \frac{1}{2f_s}$$

$$\Rightarrow \varepsilon_{CAFR}(f) \leq \frac{1}{2\sqrt{K}}. \tag{2.15}$$

An intriguing – and initially surprising – property of this error bound is that it does not depend on sampling frequency, but only FFT length. As such, it is possible to fix the size of the analysis frame, K, at some value that satisfies maximum error requirements while the sampling rate is chosen based on other factors. The lower bound on K is calculated as

$$\varepsilon_{CAFR}(f) \leq \alpha$$

$$\Rightarrow \frac{1}{2\sqrt{K}} \leq \alpha$$

$$\Rightarrow K \geq \frac{1}{4\alpha^2}, \tag{2.16}$$

which for quarter-tone accuracy implies

$$K_q \geq \frac{1}{4(1 - 2^{-\frac{1}{24}})^2} \approx 309. \tag{2.17}$$

The obvious choice for FFT length would then be the next power of 2, i.e. $K = 512$, which should yield more than enough accuracy. For well-crafted FFT implementations, the most efficient choice would instead be $K = 2^2 \times 3^4 = 324$.

2.6 Direct Realization

Now that the guiding principles have been derived, the problem lies in the direct realization and actual implementation of the CAFR.

2.6.1 Computation

Efficient computation of the CAFR can be achieved by exploiting the process of generalized ACF calculation, which produces an FFT magnitude spectrum as an intermediate value. First the magnitude spectrum, $X = |\text{FFT}(x)|$, is computed, followed by the generalized ACF (with power $\alpha = 1$), $R = \text{IFFT}(X)$.

These spectra are then trimmed to their positive-sided versions, i.e., only keeping the unique, non-symmetric values. For an FFT of length K, this yields FFT and ACF spectra of length

$$K_p = \left\langle \frac{K+1}{2} \right\rangle. \tag{2.18}$$

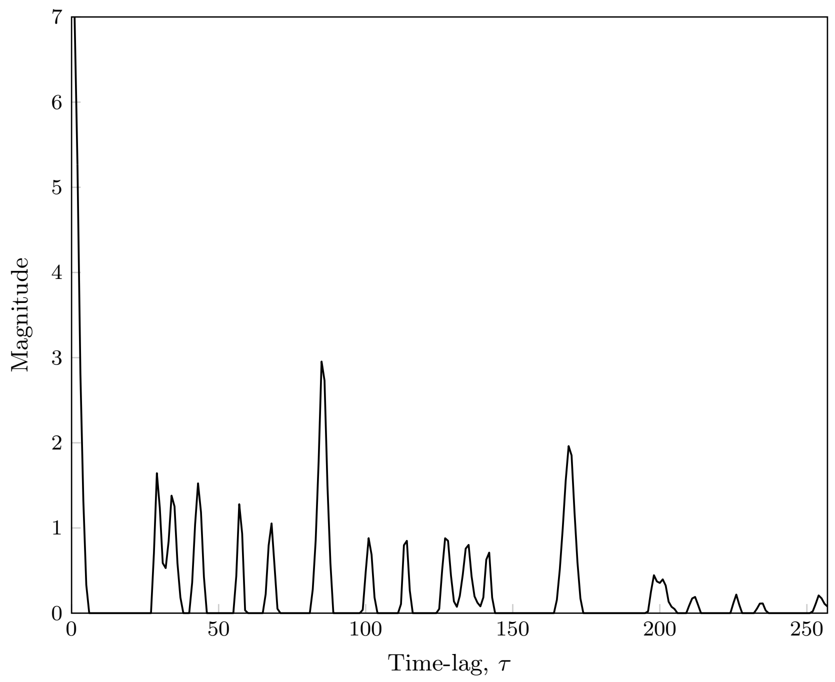

Figure 2.2: ACF "Leading Hill." The set of values on the left are an artifact of the self-similarity measurement. This leading hill artificially dominates the rest of the spectrum and is thus eliminated from processing by the CAFR.

The ACF spectrum is first half-wave rectified, and its leading hill[1] (Fig. 2.2) is zeroed out. From a frequency-analysis perspective, the first few bins of the ACF form an artificial peak, which is a side-effect of the ACF's nature as a self-similarity measure. This region is zeroed out here in order to eliminate its relatively large influence on the normalization process. The remaining ACF spectrum, R_p, is then scaled to match the total energy of the FFT spectrum, X_p, such that

$$R_{p_{(norm)}} = R_p \cdot \frac{\sum X_p}{\sum R_p}. \tag{2.19}$$

The two spectra are then stitched together by applying the optimal crossover frequency

[1]The "leading hill" is the initial set of monotonically decreasing values beginning at $\tau = 0$.

found in Equation 2.14. For the FFT, the optimal crossover bin is

$$\begin{aligned} k_c &= \left\langle \frac{f_c \cdot K}{f_s} \right\rangle \\ &= \left\langle \frac{f_s}{\sqrt{K}} \cdot \frac{K}{f_s} \right\rangle \\ &= \left\langle \frac{K}{\sqrt{K}} \right\rangle \\ &= \left\langle \sqrt{K} \right\rangle . \end{aligned} \tag{2.20}$$

And similarly, the ACF's optimal crossover bin is

$$\begin{aligned} k_c &= \left\langle \frac{f_s}{f_c} \right\rangle \\ &= \left\langle f_s \cdot \frac{\sqrt{K}}{f_s} \right\rangle \\ &= \left\langle \sqrt{K} \right\rangle . \end{aligned} \tag{2.21}$$

To stitch the spectra together, both the FFT and ACF are first trimmed to only include bins in the set $[k_c, K_p - 1]$, i.e., discarding the first k_c bins. The remaining ACF spectrum is then mirrored[2] and prepended to the remaining FFT spectrum.

2.6.2 Interpretation

The CAFR is now realized, but some work is still required to understand its structure and interpret measurements. Specifically, a simple translation from CAFR bin to frequency is

[2]Since the ACF is defined in terms of time-lag, flowing from high to low frequency, it must be mirrored to logically match the FFT's flow from low to high frequency.

$$\begin{tikzcd}
x \arrow[r] & \boxed{|\text{FFT}|} \arrow[rr] \arrow[d] & & \boxed{\text{Crossover}} \arrow[r] & C \\
& \boxed{\text{IFFT}} \arrow[r] & \boxed{\text{HWR}} \arrow[r] & \boxed{\text{Scaling}} \arrow[u] &
\end{tikzcd}$$

Figure 2.3: Direct CAFR Computation Process

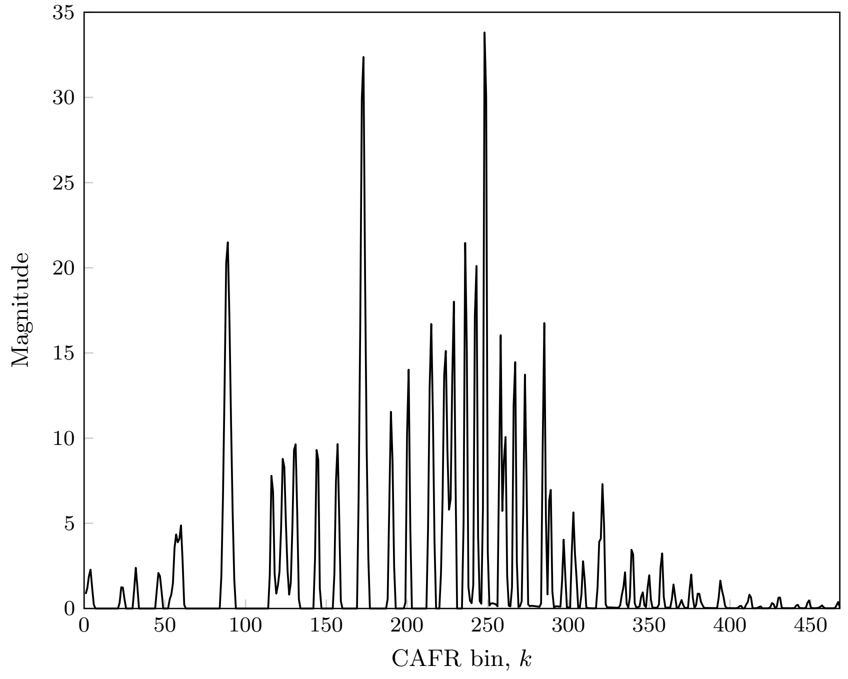

Figure 2.4: Sample CAFR Output. This CAFR was calculated with a frame length $K = 512$, resulting in 468 CAFR bins. The input signal is a hamming-windowed 512-sample frame of a C-major chord in the 5th octave.

needed.

The ACF bins reside in the region $[0, (K_p - 1) - k_c]$, mirrored and truncated. Since the truncation occurs at the end of the region, it does not affect the translation, and a simple de-mirroring operation can be performed. For a given CAFR bin, k, within the region of the ACF, the original time-lag can be recovered as

$$\tau(k) = (K_p - 1) - k. \tag{2.22}$$

And translating into the frequency domain, the mapping becomes

$$f(k) = \frac{f_s}{(K_p - 1) - k}. \tag{2.23}$$

The FFT bins reside in the region $[K_p - k_c, 2(K_p - k_c) - 1]$, truncated at the beginning. In this case, the CAFR bin must first be translated to compensate for the FFT region's starting bin then translated again to account for the truncation. So for a given CAFR bin, k, within the region of the FFT, the frequency can be found by

$$\begin{aligned} f(k) &= (k - (K_p - k_c) + k_c) \cdot \frac{f_s}{K} \\ &= (k - K_p + 2k_c) \cdot \frac{f_s}{K}. \end{aligned} \tag{2.24}$$

Finally, combining these two results produces the general equation to translate from any given CAFR bin to its respective frequency

$$f(k) = \begin{cases} \dfrac{f_s}{(K_p - 1) - k}, & 0 \leq k < K_p - k_c, \\ (k - K_p + 2k_c) \cdot \dfrac{f_s}{K}, & K_p - k_c \leq k < 2(K_p - k_c). \end{cases} \tag{2.25}$$

2.6.3 Application

Direct application of the CAFR is a difficult task. This hybrid analysis tool performs its job well, bounding the measurement error to within a given tolerance across its entire spectrum. However, in spite of its developmental motivations, it does not appear to be well-suited for use in peak-picking algorithms. Specifically, due to the differing natures of the distribution of energy across FFT and ACF bins, global peak-picking across the CAFR spectrum is unreliable. This difficulty is partially mitigated by the scaling applied to the ACF components, but it cannot be removed entirely.

This issue is particularly evident when the fundamental frequencies are close to – but below – the cross over frequency. In such cases, the FFT portion of the CAFR incorrectly dominates the spectrum, leading to erroneous results such as picking the lowest FFT bin, which is generally not even harmonically related to the fundamental in question. Some empirical investigation has been done in an attempt to identify an additional scaling factor to be applied to the ACF values to correct this problem, but the actual frequency distribution was found to affect the relative scaling far more than any other property. In fact, in the case of random white noise, the ACF values dominate. In the end, it was decided that no additional scaling should be applied, as it does not robustly improve the results.

Even though it can't reliably be used for peak-picking algorithms, direct realization of the CAFR still has some great potential in other applications. For instance, consider the NMF-based algorithm presented in Section 1.4.3, which is a learning algorithm that relies more on frame-wise comparisons of each bin than on bin-wise comparisons of each frame. The CAFR could easily be applied to this algorithm by replacing the FFT measurements with CAFR measurements. The same – or quite likely greater – accuracy could be achieved using a smaller number of CAFR bins. This reduction in frame length would increase the system's overall efficiency, while the CAFR's error-bounding properties may also lead to improved pitch detection. Such a system is developed in Section 3.2, and its results are shown in Section 4.5.

2.6.4 Efficiency

An import aspect to consider when choosing an analysis tool is its relative efficiency. The CAFR, with its FFT, IFFT, half-wave rectification, and normalization procedures, requires roughly three times as much computation as a simple FFT magnitude spectrum. So is it really better to use the CAFR?

As seen in the preceding derivations, the CAFR's minimum length is primarily driven by the minimum required accuracy, α. However, there is another important factor to consider, specifically the lowest frequency of interest. The CAFR's lower-bound on frequency is defined by the number of unique ACF bins, K_p, such that the lowest measurable frequency is

$$f_\ell = \frac{f_s}{K_p - 1}. \tag{2.26}$$

For even values of K, Equation 2.18 simplifies to $K_p = K/2 + 1$. So substituting into Equation 2.26, the lower bound on length, K_ℓ, for a given low frequency of interest is found by

$$f_\ell = \frac{f_s}{\frac{K_\ell}{2} + 1 - 1}$$

$$\Rightarrow f_\ell = \frac{2f_s}{K_\ell}$$

$$\Rightarrow K_\ell = \frac{2f_s}{f_\ell}. \tag{2.27}$$

The FFT's lower bound on frequency is always zero, so the driving factor here is simply minimum accuracy. The minimum length of an FFT that is capable of meeting accuracy requirements for a given low frequency of interest is found by

$$\tilde{\varepsilon}_{FFT}(f_\ell) = \alpha$$

$$\Rightarrow \frac{f_s}{2K_\ell f_\ell} = \alpha$$

$$\Rightarrow K_\ell = \frac{f_s}{2\alpha f_\ell}. \tag{2.28}$$

Extending that thought, the lowest possible FFT length needed to attain quarter-tone accuracy at the lowest frequency of interest matching a given CAFR length is

$$\begin{aligned} K_\ell &= \frac{f_s}{2\alpha_q f_\ell} \\ &= \frac{f_s}{2\alpha_q \left(\dfrac{f_s}{K_p - 1} \right)} \\ &= \frac{K_p - 1}{2\alpha_q}. \end{aligned} \tag{2.29}$$

From here, it is possible to perform an efficiency analysis to determine if the shorter-length CAFR can be computed more quickly than its respective longer-length FFT magnitude spectrum.

For example, consider a system that utilizes a CAFR with frame length $K = 512$ ($K_p = 257$) and sampling rate $f_s = 22\,050\,\text{Hz}$. Substituting into Equation 2.29, the shortest possible FFT to match the CAFR is of length $K_\ell \approx 4497$ bins.

A typical response would be to jump to the next power of 2, but that would be overkill at nearly double the required length. Instead, it is best to investigate the performance of shorter lengths as shown in Figure 2.5. For the test system,[3] the best length was found to be $2^2 \cdot 3^2 \cdot 5^3 = 4500$ bins, with a performance of about 1935 frames per second. A similar performance test can then be carried out for the full CAFR calculation of length 512. On the test system, this CAFR could be calculated at a rate of about 6360 frames per second – about 3.3 times faster than the matching FFT magnitude spectrum.

Beyond the initial calculation cost, which is already heavily in favor of the CAFR, one must also consider the cost of additional post-processing – inevitably present in most any algorithm – which may be greatly affected by frame length. In this case, the FFT frame has almost 5 times the number of unique bins as the CAFR frame. Even for very basic post-processing, that is a significant factor.

[3]Speed tests were run on an Intel-based PC clocked at 2.4 GHz with 3.5 GB of RAM.

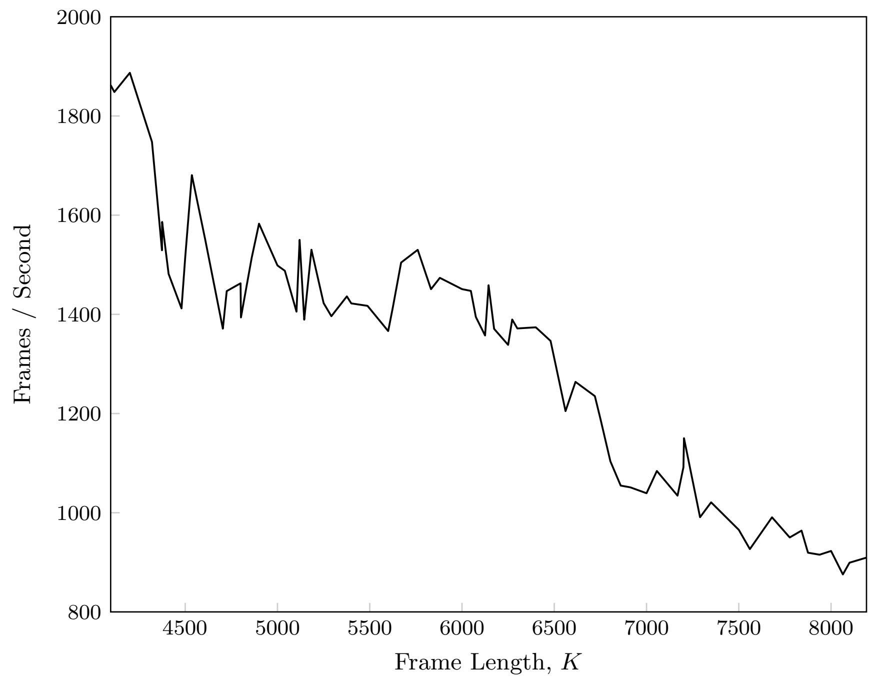

Figure 2.5: FFT Performance vs. Frame Length. This graph shows the performance of FFT magnitude spectrum calculations, as measured on a particular test system, for various frame lengths between $2^{12} = 4096$ and $2^{13} = 8192$. Only the lengths with small prime factors (2, 3, 5, and 7) are considered here, as those are the most likely to perform well.

2.7 Conceptual Application

Aside from directly realizing the CAFR, it may also be useful to apply its guiding principles and underlying concepts in other ways.

2.7.1 Parallel Algorithms

The first such approach is to run two complementary algorithms in parallel, one with an FFT basis, and the other with an ACF basis. The results could then be combined using the optimal crossover frequency calculated above, giving preference to the ACF-based algorithm below the cutoff and to the FFT-based algorithm above the cutoff. For instance, the two reference systems presented in Sections 1.4.1 and 1.4.2 could possibly be combined to produce better overall results. The difficulty for such a hybrid algorithm would lie in the combination process, as normalization issues (similar to those encountered in the direct realization) are likely to come into play.

Upon first glance, the use of two complete algorithms in parallel seems counter-productive from the perspective of efficiency. However, if application of the optimal crossover results in sufficiently reduced frame size, the combination may actually outperform a single algorithm using a frame size large enough to meet the minimum accuracy requirements. This idea is explored further in Section 3.3, where such a system is developed to run Klapuri's and Tolonen's algorithms in parallel and combine their results. The results for this approach are then presented in Section 4.5.

2.7.2 Conditional Refinement

Another more flexible approach to applying these principles is the idea of conditional refinement. The FFT magnitude spectrum is well-suited for peak-picking algorithms but suffers from a lack of accuracy at lower frequencies. To overcome this deficiency, the ACF could be used to refine the FFT's measurements when needed.

The FFT is first used to select peaks across the entire spectrum. Any peaks found below the cutoff frequency are then fed to a refinement process. The two FFT bins on either side of the peak are translated into the time-lag domain and used to define a search range. The ACF's local maximum within this range is then found and its frequency is used as the final measurement. It may also be necessary to choose multiple peaks from within the search range, as the single FFT bin could map to more than one note, especially at the lowest parts of the spectrum.

This approach would be significantly more efficient than running two full algorithms in parallel, since the ACF-based refinement is only necessary in certain cases and is a relatively simple and efficient process. It would also be more likely to easily produce reasonable results, as it would not suffer from the normalization problems found in the parallel algorithms and direct realization approaches. A new algorithm based on this approach is designed in Section 3.4, with results shown in Section 4.5.

Chapter 3: New Algorithms

3.1 Introduction

All of the preceding research and analysis has set the stage for the development of three new polyphonic pitch detection algorithms. These algorithms have been designed specifically to be as robust as possible while still performing efficiently.

The first algorithm is a direct extension of Dessein's NMF-based system (1.4.3), which has been modified to utilize the newly-developed CAFR analysis tool. The second algorithm is a direct hybridization of Tolonen's (1.4.1) and Klapuri's (1.4.2) reference systems. The two algorithms are run in parallel then combined in a novel way via the CAFR's optimal crossover. The final new algorithm is a form of Klapuri's reference system, which has been converted to employ the "conditional refinement" approach described in Section 2.7.2.

3.2 Dessein with CAFR

The first new algorithm is a logical extension of the NMF-based system of Dessein, Cont, and Lemaitre, which was described in Section 1.4.3. The reference system uses 1024-bin FFT magnitude spectra (which can be trimmed to 513 unique bins) as its measurement basis. At its chosen sampling rate of 12 600 Hz, these magnitude spectra are only capable of achieving quarter tone accuracy down to

$$\tilde{\varepsilon}_{FFT}(f_\ell) = \alpha_q$$

$$\Rightarrow \frac{f_s}{2Kf_\ell} = \alpha_q$$

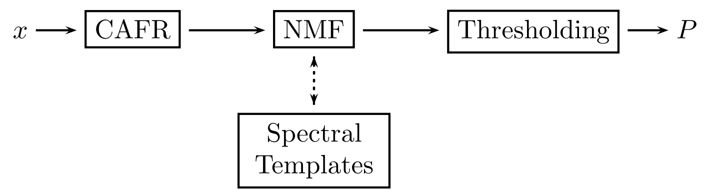

Figure 3.1: Block Diagram of the “Dessein with CAFR” System. The input signal, x, is a hamming-windowed audio frame of length K. The output, P, is the set of detected pitch numbers.

$$\Rightarrow f_\ell = \frac{f_s}{2\alpha_q K}$$

$$\Rightarrow f_\ell = \frac{12\,600\,\text{Hz}}{2 \cdot (1 - 2^{-\frac{1}{24}}) \cdot 1024} \approx 216.1\,\text{Hz},$$

which means the lowest note for which the system can accurately resolve its fundamental frequency is A3.

By applying the CAFR, it is possible to accurately resolve much lower notes while also increasing the system’s overall efficiency. It was shown in Section 2.5 that the minimum frame length required for the CAFR to achieve quarter-tone accuracy is approximately 309 bins. Choosing the next power of 2, a frame length of 512 is used for this new system, which yields a CAFR output with 468 bins. This results in a 9% reduction in the number of unique spectrum bins, which is the driving factor of efficiency of this system.

On top of the gains in efficiency, the new system is also able to achieve quarter-tone accuracy all the way down to

$$f_\ell = \frac{2f_s}{K} = \frac{2 \cdot 12\,600\,\text{Hz}}{512} \approx 49\,\text{Hz}.$$

This allows for accurate resolution of notes down to G#1 – an improvement of more than 2 octaves over the FFT-based reference system.

Happily, the trained NMF system outputs directly to pitch numbers. So a simple thresholding operation is all that is needed to gather the final results.

3.3 Hybrid Tolonen/Klapuri

The second new algorithm is a combination of Tolonen's ACF-based system and Klapuri's FFT-based system, which were described in Sections 1.4.1 and 1.4.2, respectively.

Tolonen's system produces good results within its range of operation, especially considering its conceptual simplicity and ability to run faster than realtime in the testing environment described in Chapter 4. However, this system only performs well in a rather limited range of frequencies comprising the three octaves from C2 to C5. This deficiency stems directly from its sole reliance on ACF measurements.

Klapuri's system, on the other hand, produces good results for a much broader range, consisting of the five octaves from C2 to C7. But this system is much less efficient than Tolonen's and is clearly slower than realtime in the testing environment.

The relative inefficiency of Klapuri's system stems primarily from its search method. Briefly, the salience of a given fundamental period candidate, $\hat{s}(\tau)$, is computed by a weighted sum of the local maxima of FFT values within the few bins surrounding each of the M harmonics of the fundamental frequency in question. This salience measure is then used in a binary search algorithm to find the global maximum over all τ down to a certain tolerance of accuracy.

While it is a sound idea that produces good results, the harmonic summation process can be very taxing, especially for low fundamental frequencies (which have a large number of harmonics to consider). This can be partially mitigated by capping the number of considered harmonics to a relatively small value, but the detection results are significantly degraded. For the reference implementation, the number of harmonics was capped at 20, based on the shape of the weighting function as well as empirical results.

A new hybrid algorithm has thus been developed as a compromise between the two, taking advantage of the larger range of Klapuri's system while overcoming some of its inefficiency by incorporating results from Tolonen's system.

The complete computation process of Tolonen's system is performed exactly as described in Section 1.4.1. Briefly, the input signal is first pre-whitened with inverse warped linear

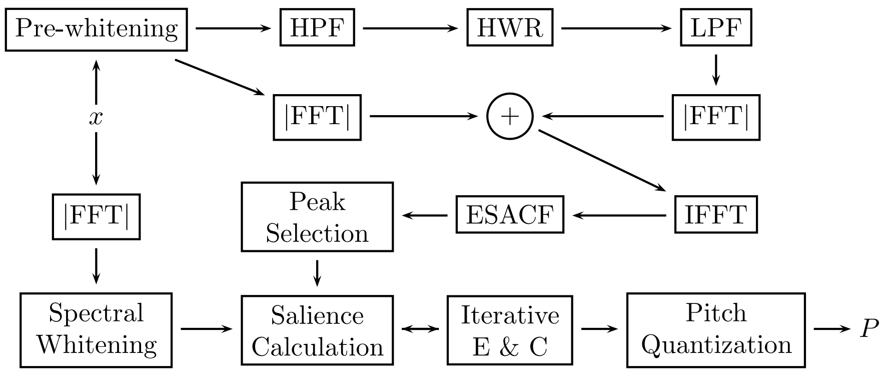

Figure 3.2: Block Diagram of the "Hybrid Tolonen/Klapuri" System. The input signal, x, is a hamming-windowed audio frame of length K. The output, P, is the set of detected pitch numbers.

predictive filtering. It is then fed into a simple two-channel auditory filter bank, where the high channel is half-wave rectified and low-pass filtered to convert high-order harmonics into fundamental beat frequencies. The generalized ACF is then computed from the sum of the two channels' magnitude spectra and fed into Tolonen's summary ACF enhancement process (ESACF).

Separately, Klapuri's system is set up by taking the input signal's magnitude spectrum and whitening it according to the process described in Section 1.3.2. It would be possible to reduce the computation required here by simply reusing the pre-whitened signal from Tolonen's system, but it was found to produce inferior results in this case.

The results of Tolonen's system are then used to inform Klapuri's salience-based iterative estimation and cancelation process. A set of candidate peaks is first selected from the ESACF within the range $\tau \in [k_c, K_p - 1]$ as defined by the CAFR's optimal crossover. Klapuri's salience measure is then computed for each of the candidate periods, and they are sorted in order of descending salience.

Klapuri's iterative estimation and cancelation process is then used. At each step, the standard search process is performed to find the period candidate, $\tau \in [f_s/2100, k_c]$, with maximum salience. Any higher-salience candidates found from Tolonen's system are used first. The result is recorded, weighted harmonic cancelation is performed as described in

Section 1.4.2, then it is removed from the candidate list, and the salience is recomputed for each of the remaining candidates. The process is then continued until the desired polyphony has been reached.

By only comparing results based on their salience measures, this particular method of combination completely side-steps the normalization issues inherent in comparing results from parallel ACF-based and FFT-based systems. Also, by using the results from Tolonen's system, the general search range is greatly reduced, leading to improved efficiency. It should be noted, however, that execution time is heavily dependent on the actual input signal – as is also the case with Klapuri's algorithm. For instance, if all constituent notes have fundamental frequencies above the crossover frequency, much less gain in efficiency will be achieved as compared to Klapuri's original system.

Once the set of fundamental frequencies has been determined, a final pitch quantization stage is required to translate the the results into a set of pitch numbers. This translation is performed quite simply by plugging each measured frequency into Equation 1.2 and rounding to the nearest integer.

3.4 Conditional Klapuri

The final new algorithm is a modified form of Klapuri's system, which was presented in Section 1.4.2. Specifically, it has been modified to take advantage of the "conditional refinement" approach to the CAFR as described in Section 2.7.2.

Klapuri's original system revolves around a salience calculation and associated binary search of the salience space to find the best fundamental frequency candidate. This is done inside an iterative estimation and cancelation process, leading to a large number of expensive salience calculations, particularly for higher polyphonies. The global salience search is quite effective, but the efficiency of the system suffers from its repeated use. To solve this problem, global peak-picking with conditional refinement can be applied to limit the salience search to specific regions of interest.

The new system begins just as Klapuri's by calculating an FFT magnitude spectrum, which is then whitened with the process of Section 1.3.2. Additionally now, the generalized ACF is computed and half-wave rectified.

Peaks are then globally selected from the FFT spectrum to identify potential fundamental frequencies. The selection process uses a simple thresholding to keep all peaks with a magnitude of at least half that of the maximum. Any peaks that fall outside the FFT's range of accuracy are then refined with the ACF using a similar peak selection process limited to the range of ACF bins corresponding to the selected FFT bin.

It was found during development that the peak selection process is slightly more reliable for the FFT (within its range of accuracy) than for the refined ACF measurements. As such, the crossover bin was set to the lowest FFT bin for which the required accuracy can be achieved, found by

$$\begin{aligned} &\tilde{\varepsilon}_{FFT}(f_\ell) < \alpha \\ &\Rightarrow \frac{f_s}{2Kf_\ell} < \alpha \\ &\Rightarrow \frac{f_s}{2K\left(\dfrac{k_\ell f_s}{K}\right)} < \alpha \\ &\Rightarrow \frac{1}{2k_\ell} < \alpha \\ &\Rightarrow k_\ell > \frac{1}{2\alpha} \\ &\Rightarrow k_c = \left\lceil \frac{1}{2\alpha} \right\rceil . \end{aligned} \tag{3.1}$$

It is worth noting that this result depends only on the required accuracy and not on the FFT length or sampling rate. This system requires quarter-tone accuracy, which yields a constant crossover bin of 18 as opposed to the standard CAFR crossover of 32 for an FFT of length 1024.

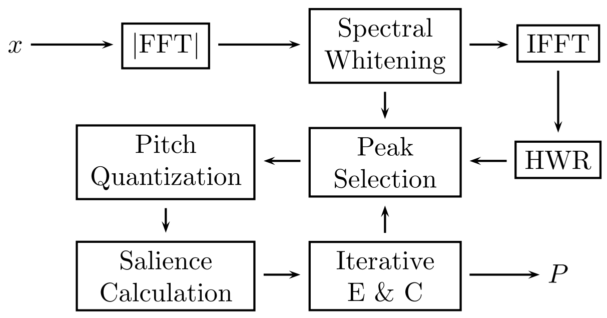

Figure 3.3: Block Diagram of the "Conditional Klapuri" System. The input signal, x, is a hamming-windowed audio frame of length K. The output, P, is the set of detected pitch numbers.

All of the selected FFT and ACF peaks are then converted to frequencies and quantized to a set of unique[1] pitch numbers. The salience of each pitch candidate is then calculated by converting to frequency and time-lag values and applying Equation 1.26. The pitch with the greatest salience is chosen at this point, and Klapuri's iterative estimation and cancelation process is applied until the desired polyphony has been reached.

[1]The pitch quantization is applied at this early stage in order to avoid redundant salience computation for multiple frequencies which would eventually map to the same pitch.

Chapter 4: Experimental Results

4.1 Introduction

Three new pitch detection systems have been developed to prove the concept of the Combined ACF/FFT Representation. This novel analysis tool and these new algorithms appear quite promising from a theoretical standpoint. But now some empirical evidence is needed to back up the theory.

The three new algorithms presented in the previous chapter were put to the test along with the three reference systems described in Sections 1.4.1 - 1.4.3. They were evaluated on objective pitch detection results as well as computational efficiency.

4.2 Testing Environment

Each of the pitch detection systems was been developed in MATLAB (see Appendix **??** for code listings), and all evaluations were performed via a MATLAB test harness. All tests were run on a 4-year-old consumer-grade PC, which had an Intel Core 2 Quad processor clocked at 2.4 GHz with 3.5 GB of RAM. The system was kept in isolation, and special care was taken to ensure the accuracy of efficiency measurements.

4.3 System Parameters

The three reference systems were implemented as described in their respective sections (1.4.1 - 1.4.3) and associated publications ([17], [13], and [14]).

Tolonen's system was run at a sampling rate of 22 050 Hz with 1024-sample (46 ms) hamming-windowed frames, which were zero-padded to 2048 samples for processing. The generalized ACF values were calculated with power $\alpha = 0.67$.

Similarly, Klapuri's system was run at a 22 050 Hz sampling rate with 1024-sample (46 ms) hanning-windowed frames, which were zero-padded to 2048 samples for processing. The harmonic weighting parameters α, β, and d were set to 27 Hz, 320 Hz, and 1.0, respectively. The number of harmonic multiples was capped at $M_{max} = 20$.

The system of Dessein, et. al., was run at a sampling rate of 12 600 Hz with 630-sample (50 ms) hamming-windowed frames, which were zero-padded to 1024 samples for processing. The β-divergence parameter was set to 0.5 for decomposition. Template learning was done with standard NMF using a 25 ms hop size.

The first new system (Dessein with CAFR) was also run at a 12 600 Hz sampling rate, but with 504-sample (40 ms) hamming-windowed frames, which were zero-padded to 512 samples for processing. The CAFR's generalized ACF components were calculated with power $\alpha = 0.67$. All other aspects of the system were exactly the same as the original Dessein reference system.

The second new system (Hybrid Tolonen/Klapuri) was run at a sampling rate of 14 700 Hz with 735-sample (50 ms) hamming-windowed frames, which were zero-padded to 1024 samples for processing. All other parameters were the same as those of the Tolonen and Klapuri reference systems.

The final system (Conditional Klapuri) was also run at a sampling rate of 14 700 Hz with 735-sample (50 ms) hamming-windowed frames, which were zero-padded to 1024 samples for processing. The generalized ACF components were calculated with power $\alpha = 0.67$, and all other system parameters were the same as those of the Klapuri reference system.

4.4 Objective Evaluation

Objective evaluation was performed in a similar fashion to the method of Klapuri in [13]. Individual note samples for MIDI pitch numbers 36 through 96 (the five octaves from C2-C7) were mixed together with equal mean-square levels in random combinations to form composite signals for polyphonies 2, 4, and 6, with 1000 combinations each. The source samples were taken from isolated recordings of four different types of pianos (providing 244 single-note samples in addition to the 3000 composites). The piano was chosen for its large range as well as the unique difficulties it presents, such as hammer noise and heavy octave errors at onset. For the two NMF-based systems, isolated samples from a fifth different type of piano were used for the training set. Another set of random combinations within the range C2-C5 was also generated for use with Tolonen's reference system, since it is not intended to work outside of that range.

The pitch detection systems were given sample frames positioned at the point of onset.[1] They were also given the polyphony of each frame as prior knowledge. While this information would not be available in most practical applications, it was provided here in order to facilitate objective comparisons between the various algorithms independent of polyphony estimation (which is a difficult problem in its own right). A second run was also performed with sample frames positioned 100 ms after onset.

The systems were then evaluated on their error rates for each polyphony, and their performance was measured in terms of processed frames per second (FPS). The three new systems as well as the Dessein reference system already report their findings in terms of pitch numbers. The Tolonen and Klapuri reference systems were given a pitch quantization stage to convert from detected F_0 into pitch numbers.

Two different error rates were measured. The first is total error, which is the proportion of notes that were not correctly identified. The second is non-octave error, in which octave substitutions are considered acceptable. This second (discounted) error rate provides a truer sense of the overall accuracy, as octave substitutions are a very common side effect of transients and can generally be fixed with some simple post-processing.

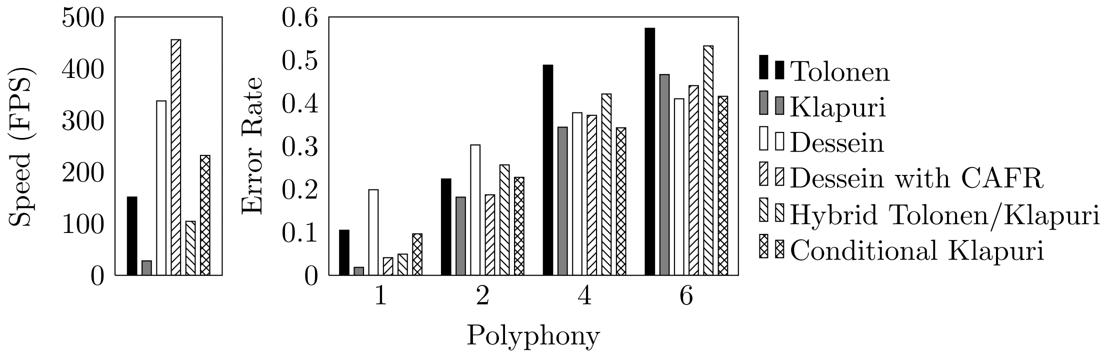

Figure 4.1: Summary of Results. The average speed in frames per second for each algorithm and the average error rates for each algorithm and polyphony.

4.5 Results

The two primary goals of the newly developed pitch detection systems were to improve on the reference systems in both overall accuracy as well as realtime performance. In this context realtime performance is defined as the ability to process at least 100 frames per second, allowing a hop size of 10 ms.

In terms of efficiency, all three developed algorithms performed quite well as compared to their respective reference systems. The Dessein with CAFR system executed about 35% faster on average than the standard Dessein reference implementation, running up to 5 times faster than realtime. The Hybrid Tolonen/Klapuri algorithm was able to process almost 275% faster than the original Klapuri reference implementation, running slightly faster than realtime. And the Conditional Klapuri algorithm processed as much as 750% faster than the standard Klapuri, running almost 2.5 times faster than realtime.

The three new systems also did well in terms of accuracy. The first two algorithms performed extremely well in the monophonic case, having 4-5% total error and less than 1% non-octave error. The third also performed reasonably well in the monophonic case, achieving around 9.5% error and 3.5% non-octave error.

For the sample set taken 100 ms after onset, the Dessein with CAFR system produced

[1]Onset was defined as the point at which the waveform's amplitude reaches one third of its maximum within the first 200 ms [1].

Table 4.1: Error Rates at Onset. The error rates are shown for each algorithm at polyphonies of 1, 2, 4, and 6, along with each algorithm's processing speed in average frames per second (FPS). In this case, sample frames were positioned directly at the point of onset.

Algorithm	ε_1	ε_2	ε_4	ε_6	FPS
Tolonen	0.0878	0.2505	0.5165	0.5913	151
Klapuri	0.0205	0.1890	0.3845	0.4998	26
Dessein	0.2172	0.3330	0.4133	0.4433	326
Dessein with CAFR	0.0410	0.2585	0.4385	0.4895	507
Hybrid Tolonen/Klapuri	0.0451	0.2790	0.4492	0.5592	103
Conditional Klapuri	0.0984	0.2295	0.3550	0.4317	223

Table 4.2: Error Rates at Onset (Discounting Octave Errors). This table shows the error rates corresponding to those of Table 4.5 when octave substitutions are allowed.

Algorithm	ϵ_1	ϵ_2	ϵ_4	ϵ_6
Tolonen	0.0878	0.2035	0.3678	0.3978
Klapuri	0.0082	0.1235	0.2575	0.3133
Dessein	0.0984	0.2320	0.3080	0.3152
Dessein with CAFR	0.0082	0.1980	0.3395	0.3800
Hybrid Tolonen/Klapuri	0.0082	0.2065	0.3180	0.3687
Conditional Klapuri	0.0328	0.1320	0.2120	0.2615

significantly less error than its counterpart for all polyphonies except 6, which had slightly more error. For the sample set positioned directly at onset, it also had slightly more error for polyphony 4. It was also able to outperform the most complex reference system – Klapuri – for all but the monophonic case in the delayed sample set.

It was found during initial testing that error rates at higher polyphonies were largely dependent on the NMF convergence criteria. Strict convergence limits on the NMF decomposition process produced better results for high polyphonies. However, excessively strict limits degraded the results at lower polyphonies, especially the monophonic case. These strict limits also greatly decreased the efficiency of the system. In the end, a compromise was chosen to achieve generally good results while maintaining the system's overall efficiency.

Table 4.3: Error Rates 100 ms after Onset. The error rates are shown for each algorithm at polyphonies of 1, 2, 4, and 6, along with each algorithm's processing speed in average frames per second (FPS). In this case, sample frames were positioned 100 ms after the point of onset.

Algorithm	ε_1	ε_2	ε_4	ε_6	FPS
Tolonen	0.1216	0.1970	0.4595	0.5558	152
Klapuri	0.0164	0.1740	0.3035	0.4325	30
Dessein	0.1803	0.2725	0.3418	0.3767	349
Dessein with CAFR	0.0410	0.1155	0.3045	0.3915	405
Hybrid Tolonen/Klapuri	0.0533	0.2340	0.3930	0.5063	106
Conditional Klapuri	0.0943	0.2255	0.3302	0.3997	241

Table 4.4: Error Rates 100 ms after Onset (Discounting Octave Errors). This table shows the error rates corresponding to those of Table 4.5 when octave substitutions are allowed.

Algorithm	ϵ_1	ϵ_2	ϵ_4	ϵ_6
Tolonen	0.1149	0.1535	0.3317	0.3718
Klapuri	0.0041	0.1300	0.2040	0.2667
Dessein	0.0738	0.1765	0.2453	0.2627
Dessein with CAFR	0.0041	0.0750	0.2325	0.2865
Hybrid Tolonen/Klapuri	0.0041	0.1665	0.2682	0.3288
Conditional Klapuri	0.0369	0.1410	0.1918	0.2343

The Hybrid Tolonen/Klapuri system performed as expected, roughly splitting the difference in error rates between the Tolonen and Klapuri reference systems in most cases. It is truly a compromise between the two, gaining significant efficiency over Klapuri and accuracy over Tolonen. However, the Dessein with CAFR system produced less total error in every case while also running 5 times faster.

The Conditional Klapuri system performed quite respectably at higher polyphonies – even beating the original Klapuri system in several cases – but was less accurate for lower polyphonies. Overall, its results were very good considering the improvement in efficiency.

Chapter 5: Conclusion

The FFT and ACF have complementary error properties. While the FFT is quite accurate for high frequencies, it suffers at lower frequencies. The ACF, on the other hand, is very accurate for low frequencies but quickly loses accuracy further up the spectrum.

5.1 New Development

These complementary properties led to the development of the optimal blend between the two – the Combined ACF/FFT Representation (CAFR). This new analysis tool comes in several flavors.

First, there is the direct realization, which constructs a new data structure representing the optimally blended spectrum. This structure is complex and somewhat intractable, but it is easily applied to certain algorithms, such as Dessein's NMF-based pitch detection system.

The second means of applying the CAFR is to use it as a method of blending results. To different algorithms – one FFT-based and one ACF-based – can be run in parallel, then their results can be blended using the CAFR's optimal crossover.

And the third application of the CAFR is the concept of conditional refinement. The FFT is well-suited to peak-picking and can easily be used to find areas of interest across the complete spectrum. Then, based on the CAFR's optimal crossover, these measurements can conditionally be refined using the ACF.

All of these approaches were developed into new pitch detection systems to test the CAFR's usefulness. The first was a version of Dessein's NMF-based system that replaced the FFT measurements with direct CAFR measurements. The second was a hybridization of Tolonen's ACF-based system with Klapuri's FFT-based system, where both algorithms were run in parallel with results blended together. And the third was a modification of

Klapuri's system to employ conditional refinement as a means of narrowing the search for fundamental frequencies.

5.2 Significance

The three new systems were put to the test along with the three reference systems, and the results clearly demonstrated the usefulness of the CAFR as an analysis tool for pitch detection systems. By applying the CAFR – both directly and conceptually – it was possible to achieve the accuracy needed for pitch detection over a larger range of frequencies while using a smaller number of analysis bins.

The pitch detection results showed that a smaller CAFR was able to match or exceed the performance of a larger FFT or ACF in almost of the tested cases. Meanwhile, the decrease in size resulted in vast improvements in efficiency, ranging from 35% to 750% increases in speed as compared to the original reference systems.

5.3 Future Work

The three new CAFR-based pitch detection systems performed quite well, but there is still plenty of room for future research and development.

The application of the CAFR to Dessein's NMF system is of particular interest. The computational headroom available with this system allows for the addition of much more complexity to further improve its results. For instance, additional spectral templates could be added to represent the attack, sustain, and release components of each note's life-cycle. This may help to reduce errors at onset and offset and in the case of improperly shortened notes. Higher-level post-processing could also be added to take musical expectations into account, allowing for the exclusion or adjustment of detected notes that don't fit within the current musical scene.

Another area of interest is the conditionally-refined version of Klapuri's system. This system achieved the greatest percentage-wise increase in efficiency. It also boasted impressive gains in robustness of pitch detection for high polyphonies. However, it did not perform as well for lower polyphonies, especially the monophonic case. It had particularly high proportions of octave error in monophonic detection, likely stemming from the peak selection process. More work should be done to improve these results.

Finally, while the CAFR has proven its usefulness, it can be difficult to apply due to the inherent differences in scale and energy distribution between the FFT and ACF. It is conceivable that other means of improving both robustness and analytical speed could be developed that would not suffer from such difficulties. One such possibility is multi-rate FFT processing, in which the high frequency information would be captured from analysis at one sampling rate, then the low frequency information would be analyzed using the same FFT length but with a zero-padded decimated signal. It may also be possible to develop a non-linear scaling procedure to match the ACF to the FFT more appropriately.

Appendix A: Code Listings

A.1 General Support Functions

The following are various support and convenience functions, which were defined to simplify common tasks.

```
function [p] = pitch(f)
%PITCH Calculates the pitch number of the given frequency
%   f - frequency of interest
%   p - (out) pitch number

p = 69+12*log2(f/440);

end
```

```
function [f] = pfreq(p)
%PFREQ Calculates the frequency of a given pitch number
%   p - pitch number of interest
%   f - (out) frequency

f = 440*2^((p-69)/12);

end
```

```
function [f] = bin2freq(k,K,Fs)
%BIN2FREQ Calculates the frequency of the given FFT bin
%   k - FFT bin of interest
%   K - FFT length
%   Fs - sampling frequency
%   f - (out) frequency

f = (k-1)*Fs/K;

end
```

```
function [k] = freq2bin(f,Fs,K)
%FREQ2BIN Calculates the FFT bin index of the given frequency
%   f - frequency of interest
%    Fs - sampling frequency
%    K - FFT length
%    k - (out) FFT bin index

k = 1+(f.*(K/Fs));

end
```

```
function [X,f] = pfft(x,Fs)
%PFFT Calculates a positive-sided FFT
%    x - input signal
%    Fs - sampling frequency (if frequency mapping is desired)
%    X - (out) positive-sided FFT
%    f - (out) frequency to bin mapping (if Fs is specified)

% Signal length
K = length(x);

% Number of unique (non-symmetric) FFT values
pK = round((K+1)/2);

% Positive-frequency FFT bins
X = fft(x);
X = X(1:pK);

% Frequency values of bins
if nargin > 1
    f = (0:pK-1)/(K/Fs);
end

end
```

```
function [Y] = ipfft(X)
%IPFFT Calculates the IFFT of a positive-sided FFT
%    X - positive-sided FFT
%    Y - (out) IFFT

% Positive-sided length
pK = length(X);

% Full length
if mod(pK,2) == 0
    K = (2*pK)-1;
else
    K = 2*(pK-1);
end

% Re-generate the symmetric negative frequency values
X(pK+1:K) = conj(X(pK-mod(pK,2):-1:2));

% And perform the IFFT
Y = ifft(X);
```

```
end
```

```
function [pks,locs] = findpeaks1d(x)
%FINDPEAKS1D A simpler, much faster version of findpeaks for 1-D data
%   x - input vector
%   pks - (out) sorted peak values
%   locs - (out) sorted peak locations

if isrow(x)
    x1 = [x(2:length(x)),0];
    x2 = [0,x(1:length(x)-1)];
else
    x1 = [x(2:length(x));0];
    x2 = [0;x(1:length(x)-1)];
end
x(x1 > x | x2 > x) = 0;
[pks, locs] = sort(x,'descend');

end
```

A.2 Tolonen's Reference System

The following code implements Tolonen's reference system. This code makes use of several functions (barkwarp, wlpc, and wfilter) from the freely-available "WarpTB" warped signal processing toolbox [24].

```
function [es] = esacf(s)
%ESACF Enhances a summary ACF to reveal fundamental frequencies
%   s - summary ACF
%   es - (out) enhanced summary ACF

% Initialize the ESACF to positive values of SACF
es = max(0,s);

% Loop through scaling factors
for m=2:5
    % Initialize the scaled SACF to zeros
    sm = zeros(size(s));

    % Scale the SACF by factor m using linear interpolation
    for t=1:length(s)
        d = 1+floor((t-1)/m);
        sm(t) = s(d) + ((t-m*d)/m)*(s(d+1)-s(d));
    end

    % Update the ESACF
    es = max(es-max(sm,0),0);
```

```
end

end
```

```
function [P,V] = tolonen_frame(x,w,K,Kp,Fs,lambda,b1,a1,b2,a2,pmin,pmax,Np)
%TOLONEN_FRAME Tolonen's pitch detection system for a single frame
%   x - input frame
%   w - window function
%   K - FFT length
%   Kp - positive-sided length
%   Fs - sampling frequency
%   lambda - bark scale wlpc filtering constant
%   b1,a1 - low-pass filter coefficients
%   b2,a2 - high-pass filter coefficients
%   pmin - minimum pitch to consider
%   pmax - maximum pitch to consider
%   Np - number of pitches (-1 to auto-estimate)
%
%   P - (out) detected pitch numbers
%   V - (out) associated peak values

% Window and pad to full length
x = x .* w;
x(length(w)+1:K) = 0;

% Apply whitening
y = wfilter(wlpc(x,12,lambda),1,x,lambda);

% Apply the filter bank
low = filter(b1,a1,y);
high = filter(b1,a1,max(0,filter(b2,a2,y)));

% Calculate the ESACF
s = ifft(abs(fft(low)).^0.67 + abs(fft(high)).^0.67);
es = esacf(s(1:Kp));

% Find the peaks and record results
P = []; V = [];
[pks, loc] = findpeaks1d(es);
if Np == -1
    % Borrowing Klapuri's polyphony estimation method
    S = zeros(1,length(pks)+1);
    S(1) = 0;
    st = 0;
    n = 1;
    for j=1:length(pks)
        p = round(pitch(Fs/(loc(j)-1)));
        if p >= pmin && p <= pmax
            st = st + pks(j);
            S(j+1) = st/(j^0.7);
            if S(j+1) <= S(j)
                break;
            end
            P(n) = p;
            V(n) = pks(j);
            n = n+1;
        else
            S(j+1) = S(j);
        end
    end
else
    % Just use the first Np detections
    for j=1:Np
        P(j) = round(pitch(Fs/(loc(j)-1)));
        V(j) = pks(j);
    end
end

end
```

And this listing shows the pre-calculated system parameters.

```
% Decimated sampling frequency
Fs = 22050;

% Window size (46ms)
Kw = 1024;

% Hamming window
w = hamming(Kw)';

% Full padded frame FFT size
K = 2*Kw;

% Positive-sided length
Kp = round((K+1)/2);

% Warping parameter
lambda = barkwarp(Fs);

% Create the filter bank
[b1,a1] = butter(2,[2*70/Fs,2*1000/Fs],'bandpass');
[b2,a2] = butter(2,[2*1000/Fs,2*10000/Fs],'bandpass');

% Pitch range
pmin = 36; % C2
pmax = 72; % C5
```

A.3 Klapuri's Reference System

The following code implements Klapuri's reference system.

```
function [H,c] = swfilters(B,K,Fs)
%SWFILTERS Generates triangular power responses for spectral whitening
```

```
%   B - number of critical bands
%   K - FFT length
%   Fs - sampling frequency in Hz
%   H - (out) set of filter power responses
%   c - (out) critical band center bins

% Calculate the critical band center frequency bins
c(1:B) = 0;
for b=1:B
    c(b) = round(freq2bin(229*(10^((b+1)/21.4)-1),Fs,K));
end

% Generate the triangular power responses
H(1:B,1:round((K+1)/2)) = 0;
for b=1:B
    % Get the lower bin
    if b > 1
        k1 = c(b-1);
    else
        % For first band, calculate the 0th band frequency bin
        k1 = round(freq2bin(229*(10^(1/21.4)-1),Fs,K));
    end

    % The center bin
    kc = c(b);

    % And upper frequency
    if b < B
        k2 = c(b+1);
    else
        % For last band, calculate the (B+1)th band frequency bin
        k2 = round(freq2bin(229*(10^((b+2)/21.4)-1),Fs,K));
    end

    % Line parameters for the rising and falling edges
    m1 = 1/(kc-k1);
    m2 = 1/(kc-k2);
    b1 = -m1*k1;
    b2 = -m2*k2;

    % Use line equations to generate the triangle
    for k=k1:k2
        if k <= kc
            H(b,k) = max(0, m1*k+b1);
        else
            H(b,k) = max(0, m2*k+b2);
        end
    end
end

end
```

```
function [Y,G] = whiten(X,H,c)
%WHITEN Applies the Klapuri spectral whitening algorithm
%   X - positive-sided magnitude spectrum
%   H - filter power responses (from swfilters)
%   c - critical band center bins (from swfilters)
%   Y - (out) whitened positive-sided magnitude spectrum
%   G - (out) bin-wise compression coefficients

% Positive-sided FFT size
Kp = length(X);

% Number of critical bands
B = size(H,1);

% Calculate band-wise compression coefficients
s = sum((H .* repmat(X.^2,B,1)),2)';
g = (s ./ Kp).^(-0.33);

% Interpolate to bin-wise coefficients
G = interp1([1,c,Kp],[g(1),g,g(B)],1:Kp);

% And apply the whitening
Y = G .* X;

end
```

```
function [t,s] = salience(X,K,Fs,tmin,tmax,Mmax)
%SALIENCE Finds the highest salience period over the given range
%   X - positive-sided magnitude spectrum
%   K - FFT length
%   Fs - sampling frequency
%   tmin - minimum period (tau)
%   tmax - maximum period (tau)
%   Mmax - maximum number of harmonics to process
%
%   t - (out) period (tau) where the highest salience was found
%   s - (out) salience value

% Initial number of blocks
Q = 1;

% Initial best block
qbest = 1;

% Initialize the period range
tlow = zeros(1,100);
tup = zeros(1,100);
tlow(1) = tmin;
tup(1) = tmax;

% Initialize max salience
smax = zeros(1,100);

% Perform the search
while tup(qbest)-tlow(qbest) > 0.01*tup(qbest)
    % Split the best block and compute new limits
    Q = Q + 1;
    tlow(Q) = (tlow(qbest) + tup(qbest))/2;
    tup(Q) = tup(qbest);
    tup(qbest) = tlow(Q);

    % Compute new saliences for the two block-halves
    for q = [qbest, Q]
        % Determine the number of harmonics to process
        M = min(floor(tlow(q)/2),Mmax);

        % And process them
        mK = (1:M)*K;
        kmin = 1+round(mK./tup(q));
        kmax = 1+round(mK./tlow(q));
        Xmax = zeros(1,M);
        fup = Fs/tup(q);
        for m=1:M
            Xmax(m) = max(X(kmin(m):kmax(m)))/(m*fup + 320);
        end
        smax(q) = (Fs/tlow(q) + 27) * sum(Xmax);
    end

    % Find the best block so far
    [~,qbest] = max(smax(1:Q));
end

% Return the best period and its salience
t = (tlow(qbest) + tup(qbest))/2;
s = smax(qbest);

end
```

```
function [P,V] = klapuri_frame(x,w,K,Kp,Fs,H,c,tmin,tmax,Mmax,Np)
%KLAPURI_FRAME Klapuri's pitch detection system for a single frame
%   x - input frame
%   w - window function
%   K - FFT length
%   Kp - positive-sided length
%   Fs - sampling frequency
%   H - spectral whitening filters
%   c - spectral whitening center frequencies
%   tmin - minimum tau to search
%   tmax - maximum tau to search
%   Mmax - maximum number of harmonics to consider
%   Np - number of pitches (-1 to auto-estimate)
%
%   P - (out) detected pitch numbers
%   V - (out) associated peak values

% Window and pad to full length
x = x .* w;
x(length(w)+1:K) = 0;

% Calculate the positive-sided FFT magnitude spectrum
X = abs(pfft(x));

% Apply spectral whitening
```

```
Y = whiten(X,H,c);

% Initialize residual and detected spectrums
YR = Y;
YD = zeros(1,Kp);

P = []; V = [];
j = 1;
st = zeros(1,10);
S = zeros(1,10);
while true
    % Find the period with the best salience
    [t,s] = salience(YR,K,Fs,tmin,tmax,Mmax);

    % Remove the period and its partials from the residual
    M = min(floor(t/2),Mmax);
    g3f = Fs/t;
    g1 = (g3f + 27);
    for m=1:M
        g = g1 / (m*g3f + 320);
        kt = 1+round(m*K/t);
        k = max(1,kt-2):min(Kp,kt+2);
        YD(k) = YD(k) + g*YR(k);
        YR(k) = max(0, Y(k)-YD(k));
    end

    % Determine whether to continue (polyphony estimation)
    if Np == -1
        st(j) = s;
        S(j) = sum(st)/j^0.7;
        if (j > 1 && S(j) <= S(j-1))
            break;
        end
    end

    % Record the finding
    P(j) = round(pitch(Fs/t));
    V(j) = s;
    if j == Np
        break;
    end
    j = j+1;
end

end
```

And this listing shows the pre-calculated system parameters.

```
% Decimated sampling frequency
Fs = 22050;

% Window size (46ms)
Kw = 1024;

```

```
% Hanning window
w = hanning(Kw)';

% Full padded frame FFT size
K = 2*Kw;

% Positive-sided FFT size
Kp = round((K+1)/2);

% Hop size (10ms)
hop = floor(Fs/100);

% Period search bounds ~(C2-C7)
tmin = Fs/2100;
tmax = Fs/64;

% Maximum number of harmonics to process
Mmax = 20;

% Number of critical bands
B = 30;

% Generate the whitening filters
[H c] = swfilters(B,K,Fs);
```

A.4 Dessein's Reference System

The following code implements Dessein's reference system.

```
function [w] = learn_template(V)
%LEARN_TEMPLATE Learns an NMF spectral template
%   V - observation matrix for a single note of interest
%   w - (out) the learned spectral template

% Initialize the w and h vectors
w = ones(size(V,1),1);
h = ones(1,size(V,2));

% Perform the multiplicative updates procedure until convergence
i = 0;
d = 0;
dp = 0;
while (dp == 0 || abs(d-dp) > 0.01*dp) && i < 100
    wp = w;
    h = h.*((w'*V)./(w'*w*h));
    w = w.*((V*h')./(w*(h*h')));
    i = i+1;
    if i > 1
        dp = d;
        d = sum(abs(w-wp));
    end
```

```
end

end
```

```
function [h] = nmf_decomp(v,W,b)
%NMF_DECOMP Performs NMF decomposition with beta-divergence
%   v - single-frame observation
%   W - spectral templates
%   b - beta-divergence constant
%   h - (out) template activations

% Pre-calculate constants
Wv = (W.*(v*ones(1,size(W,2))))';
WT = W';
b1 = b-1;
b2 = b-2;

% Initialize the output vector
h = ones(size(W,2),1);

% And perform the multiplicative updates procedure until convergence
i = 0;
d = 0;
dp = 0;
while (dp == 0 || abs(d-dp) > 0.2*dp) && i < 15
    hp = h;
    Wh = W*h;
    h = h .* ((Wv*(Wh.^b2))./(WT*(Wh.^b1)));
    i = i+1;
    if i > 1
        dp = d;
        d = sum(abs(h-hp));
    end
end

end
```

```
function [P,V] = dessein_fft_frame(x,w,K,W,b,pmin,Np)
%DESSEIN_FFT_FRAME Dessein (FFT) for a single frame
%   x - input frame
%   w - window function
%   K - FFT length
%   W - spectral templates
%   b - beta-divergence constant
%   pmin - minimum pitch
%   Np - number of pitches (-1 to auto-estimate)
%
%   P - (out) detected pitch numbers
%   V - (out) associated peak values

% Window and pad to full length
x = x .* w;
x(length(w)+1:K) = 0;
```

```
% Calculate the positive-sided FFT maginitude spectrum
v = abs(pfft(x));

% Run the NMF decomposition
PN = nmf_decomp(v,W,b);

% Record the findings
P = []; V = [];
[pks, locs] = sort(PN,'descend');
if Np == -1
    % Borrowing Klapuri's polyphony estimation method
    S = zeros(1,length(pks)+1);
    S(1) = 0;
    st = 0;
    n = 1;
    for j=1:length(pks)
        if Np == -1
            st = st + pks(j);
            S(j+1) = st/(j^0.7);
            if S(j+1) <= S(j) || n > 6
                break;
            end
        elseif n > Np
            break;
        end
        P(n) = locs(j)+pmin-1;
        V(n) = pks(j);
        n = n+1;
    end
else
    % Just use the first Np detections
    P = locs(1:Np)+pmin-1;
    V = pks(1:Np);
end

end
```

And this listing shows the pre-calculated system parameters.

```
% The beta-divergence constant
b = 0.5;

% Decimated sampling frequency
Fs = 12600;

% Window size (50ms)
Kw = 630;

% Hamming window
w = hamming(Kw);

% FFT length
K = 1024;
```

```
% Minimum pitch
pmin = 36;
```

A.5 CAFR Realization

The following code implements the direct realization of the CAFR and defines a helper function to interpret its results.

```
function [C] = cafr(x)
%CAFR Computes the Combined ACF/FFT Representation
%   x - input signal
%   C - (out) computed CAFR

% Get the FFT length, positive-sided spectrum length, and crossover bin
K = length(x);
Kp = round((K+1)/2);
kc = 1+round(sqrt(K));

% Compute the magnitude spectrum and generalized ACF
X = abs(fft(x));
R = ifft(X.^0.67);

% Trim to possitive-sided spectrums
Xp = X(1:Kp);
Rp = R(1:Kp);

% Half-wave rectify the ACF
Rp = max(Rp,10^(-6));

% Zero out its leading hill
i = 1;
while Rp(i) > Rp(i+1)
    i = i+1;
end
Rp(1:i) = 0;

% Trim based on crossover bin and flip the ACF
Xc = Xp(kc:Kp);
Rc = Rp(Kp:-1:kc);

% Scale the ACF based on the FFT
Rc = Rc .* sum(Xp)/sum(Rp);

% And assemble into the final CAFR
C = [Rc Xc];

end
```

```
function [f] = cafr_bin2freq(k,K,Fs)
%CAFR_BIN2FREQ Calculates the frequency of a CAFR bin
%   k - the CAFR bin of interest
%   K - the FFT length
%   Fs - the sampling frequency
%   f - (out) the calculated frequency

% Calculate the positive-sided spectrum length and crossover bin
Kp = round((K+1)/2);
kc = round(sqrt(K))+1;

% And get the frequency according to the type of bin
if k <= Kp-(kc-1)
    % Mirrored ACF
    f = Fs/(Kp-k);
else
    % Offset FFT
    f = ((k-1)-Kp+2*(kc-1))*(Fs/K);
end

end
```

A.6 Dessein with CAFR

The following code implements the Dessein with CAFR system.

```
function [P,V] = dessein_cafr_frame(x,w,K,W,b,pmin,Np)
%DESSEIN_CAFR_FRAME Dessein (CAFR) for a single frame
%   x - input frame
%   w - window function
%   K - FFT length
%   W - spectral templates
%   b - beta-divergence constant
%   pmin - minimum pitch
%   Np - number of pitches (-1 to auto-estimate)
%
%   P - (out) detected pitch numbers
%   V - (out) associated peak values

% Window and pad to full length
x = x .* w;
x(length(w)+1:K) = 0;

% Calculate the CAFR spectrum
v = cafr(x')';

% Run the NMF decomposition
PN = nmf_decomp(v,W,b);

% Record the findings
P = []; V = [];
```

```
[pks, locs] = sort(PN, 'descend');
if Np == -1
    % Borrowing Klapuri's polyphony esitmation
    S = zeros(1,length(pks)+1);
    S(1) = 0;
    st = 0;
    n = 1;
    for j=1:length(pks)
        st = st + pks(j);
        S(j+1) = st/(j^0.7);
        if S(j+1) <= S(j) || n > 6
            break;
        end
        P(n) = locs(j)+pmin-1;
        V(n) = pks(j);
        n = n+1;
    end
else
    % Just use the first Np detections
    P = locs(1:Np)+pmin-1;
    V = pks(1:Np);
end

end
```

And this listing shows the pre-calculated system parameters.

```
% The beta-divergence constant
b = 0.5;

% Decimated sampling frequency
Fs = 12600;

% Window size (40ms)
Kw = 504;

% Hamming window
w = hamming(Kw);

% FFT length
K = 512;

% Minimum pitch
pmin = 36;
```

A.7 Hybrid Tolonen/Klapuri

The following code implements the Hybrid Tolonen/Klapuri system. This code makes use of several functions (barkwarp, wlpc, and wfilter) from the freely-available "WarpTB" warped signal processing toolbox [24].

```
function [P,V] = hybridtk_frame(x,w,K,Kp,Fs,kc,pmin,lambda,b1,a1,b2,a2,tmin,
    tmax,Mmax,H,c,Np)
%HYBRIDTK_FRAME Hybrid Tolonen/Klapuri pitch detection for a single frame
%   x - input frame
%   w - window function
%   K - FFT length
%   Kp - positive-sided length
%   Fs - sampling frequency
%   kc - crossover bin
%   pmin - minimum pitch
%   lambda - warping parameter
%   b1,a1 - low-pass filter coefficients
%   b2,a2 - high-pass filter coefficients
%   tmin - minimum klapuri search range
%   tmax - maximum klapuri search range
%   Mmax - maximum number of harmonic multiples
%   Np - number of pitches (-1 to auto-estimate)
%
%   P - (out) detected pitch numbers
%   V - (out) associated peak values

% Window and pad to full length
x = x .* w;
x(length(w)+1:K) = 0;

% Calculate the whitened ESACF for Tolonen
y = wfilter(wlpc(x,12,lambda),1,x,lambda);
R = ifft(abs(fft(y)).^0.67 + abs(fft(filter(b1,a1,max(0,filter(b2,a2,y)))))
    .^0.67);
R = esacf(R(1:Kp));

% Calculate the whitened FFT for Klapuri
Y = whiten(abs(pfft(x)),H,c);

% Find the candidates from the ESACF
[~,locs] = findpeaks1d(R(kc:Kp));
N = 6;
rt = zeros(1,N); rs = zeros(1,N);
n = 1;
for i=1:length(locs)
    t = locs(i)+(kc-1)-1;
    if round(pitch(Fs/t)) >= pmin
        % Apply Klapuri's salience calculation
        rt(n) = t;
        f = Fs/t;
        M = min(floor((t-0.5)/2),Mmax);
        mK = (1:M)*K;
        kmin = 1+round(mK./(t+0.5));
        kmax = 1+round(mK./(t-0.5));
        Ymax = zeros(1,M);
        for m=1:M
            Ymax(m) = max(Y(kmin(m):kmax(m)))/(f*m+320);
        end
        rs(n) = (f+27)*sum(Ymax);
        n = n+1;
        if n > N
            break;
        end
    end
end
[rs,ix] = sort(rs,'descend');
rt = rt(ix);

% Initialize residual and detected spectrums
YR = Y;
YD = zeros(1,Kp);

P = []; V = [];
j = 1; n = 1;
st = zeros(1,10);
S = zeros(1,10);
while true
    % Find the period with the best salience
    [t s] = salience(YR,K,Fs,tmin,tmax,Mmax);

    % Check the ESACF candidates
    t2 = t;
    exit = 0;
    while n <= N && rs(n) >= s
        % Determine whether to continue (polyphony estimation)
        if Np == -1
            st(j) = rs(n);
            S(j) = sum(st)/j^0.7;
            if (j > 1 && S(j) <= S(j-1)) || j > 6
                exit = 1;
                break;
            end
        end

        % Record the pitch
        t = rt(n);
        f = Fs/t;
        P(j) = round(pitch(f));
        V(j) = rs(n);
        if j == Np
            exit = 1;
            break;
        end
        j = j+1;
        n = n+1;

        % Remove the period and its partials from the residual
        M = min(floor(t/2),Mmax);
        for m=1:M
            kt = 1+round(m*K/t);
            k = max(1,kt-2):min(Kp,kt+2);
            YD(k) = YD(k) + YR(k)*(f+27)/(f*m+320);
            YR(k) = max(0, Y(k)-YD(k));
        end
        % Recalculate the remaining saliences
        for i=n:N
            t = rt(i);
            f = Fs/t;
            M = min(floor((t-0.5)/2),Mmax);
            mK = (1:M)*K;
            kmin = 1+round(mK./(t+0.5));
            kmax = 1+round(mK./(t-0.5));
            ymax = zeros(1,M);
            for m=1:M
                ymax(m) = max(Y(kmin(m):kmax(m)))/(f*m+320);
            end
            rs(i) = (f+27)*sum(ymax);
        end
        [rs,ix] = sort(rs,'descend');
        rt = rt(ix);
    end
    if exit == 1
        break;
    end
    t = t2;

    % Determine whether to continue (polyphony estimation)
    if Np == -1
        st(j) = s;
        S(j) = sum(st)/j^0.7;
        if (j > 1 && S(j) <= S(j-1)) || j > 6
            break;
        end
    end

    % Record the finding
    P(j) = round(pitch(Fs/t));
    V(j) = s;
    if j == Np
        break;
    end
    j = j+1;

    % Remove the period and its partials from the residual
    M = min(floor(t/2),Mmax);
    for m=1:M
        kt = 1+round(m*K/t);
        k = max(1,kt-2):min(Kp,kt+2);
        YD(k) = YD(k) + YR(k)*(f+27)/(f*m+320);
        YR(k) = max(0, Y(k)-YD(k));
    end
end

end
```

And this listing shows the pre-calculated system parameters.

```
% Decimated sampling frequency
Fs = 14700;

% Window size (50ms)
Kw = 735;

% Hanning window
w = hamming(Kw)';

% FFT length
K = 1024;

% Positive-sided spectrum length
Kp = round((K+1)/2);

% Crossover bin
kc = round(sqrt(K))+1;

% Hop size (10ms)
hop = floor(Fs/100);

% Warping parameter
lambda = barkwarp(Fs);

% Create the filter bank
[b1,a1] = butter(2,[2*70/Fs,2*1000/Fs],'bandpass');
[b2,a2] = butter(2,2*1000/Fs,'high');

% Klapuri search range
tmin = Fs/2100;
tmax = kc;

% Maximum number of harmonic multiples
Mmax = 20;

% Number of critical bands
B = 30;

% Generate the whitening filters
[H c] = swfilters(B,K,Fs);

% Pitch range
pmin = 36;
pmax = 96;
```

A.8 Conditional Klapuri

The following code implements the Conditional Klapuri system.

```
function [p,s] = cond_salience(X,R,K,Kp,Fs,kc,thresh,pmin,pmax,Mmax)
%COND_SALIENCE Finds the best pitch using conditional salience calculation
```

```
%   X - positive-sided magnitude spectrum
%   R - positive-sided generalized ACF spectrum
%   K - FFT length
%   Kp - positive-sided FFT length
%   Fs - sampling frequency
%   kc - crossover bin
%   thresh - peak selection threshold
%   pmin - minimum pitch to consider
%   pmax - maximum pitch to consider
%   Mmax - maximum number of harmonics to process
%
%   p - (out) pitch the highest salience
%   s - (out) salience value

% Get the FFT peaks
[Xpks,Xlocs] = findpeaks1d(X);
Xlocs = Xlocs(Xpks>thresh*Xpks(1));

% Select the candidate pitches
n = 1;
P = zeros(size(Xlocs));
for k=Xlocs
    if k >= kc
        % Use the FFT value directly
        f = bin2freq(k,K,Fs);
        p = round(pitch(f));
        if (p >= pmin) && (p <= pmax) && ~any(P == p)
            P(n) = p;
            n = n+1;
        end
    elseif k > 2
        % Refine using the ACF
        t1 = min(1+round(K/(k-0.5)),Kp);
        t2 = min(1+round(K/(k-1.5)),Kp);
        [Rpks,Rlocs] = findpeaks1d(R(t1:t2));
        Rlocs = Rlocs(Rpks>thresh*Rpks(1));
        for t=Rlocs
            f = Fs/(t+t1-2);
            p = round(pitch(f));
            if (p >= pmin) && (p <= pmax) && ~any(P == p)
                P(n) = p;
                n = n+1;
            end
        end
    end
end
P = P(P>0);

% Calculate the salience of each pitch candidate
if ~isempty(P)
    s = zeros(size(P));
    for n=1:length(P)
        f = pfreq(P(n));
        t = Fs/f;
        M = min(floor((t-0.5)/2),Mmax);
        mK = (1:M)*K;
```

```
            kmin = 1+round(mK./(t+0.5));
            kmax = 1+round(mK./(t-0.5));
            Xmax = zeros(1,M);
            for m=1:M
                Xmax(m) = max(X(kmin(m):kmax(m)))/(f*m+320);
            end
            s(n) = (f+27)*sum(Xmax);
        end
        % And return the pitch with the highest salience
        [s,i] = max(s);
        p = P(i);
else
        p = 0;
        s = 0;
end

end
```

```
function [P,V] = cond_klapuri_frame(x,w,K,Kp,Fs,kc,thresh,pmin,pmax,Mmax,H,c,
    Np)
%COND_KLAPURI_FRAME Conditional Klapuri pitch detection for a single frame
%    x - input frame
%    w - window function
%    K - FFT length
%    Kp - positive-sided length
%    Fs - sampling frequency
%    kc - crossover bin
%    thresh - peak selection threshold
%    pmin - minimum pitch to consider
%    pmax - maximum pitch to consider
%    Mmax - maximum number of harmonics to process
%    H - spectral whitening filters
%    c - spectral whitening center frequencies
%    Np - number of pitches (-1 to auto-estimate)
%
%    P - (out) detected pitch numbers
%    V - (out) associated peak values

% Window and pad to full length
x = x .* w;
x(length(w)+1:K) = 0;

% Calculate the whitened FFT and ACF
Y = whiten(abs(pfft(x)),H,c);
R = max(ipfft(Y.^0.67),0);
R = R(1:Kp);

% Initialize residual and detected spectrums
Yr = Y;
Yd = zeros(1,Kp);

P = []; V = [];
j = 1;
st = zeros(1,6);
S = zeros(1,6);
```

```
while true
    % Find the period with the best salience
    [p,s] = cond_salience(Yr,R,K,Kp,Fs,kc,thresh,pmin,pmax,Mmax);
    if p == 0
        break;
    end

    % Remove the period and its partials from the residual
    f = pfreq(p);
    M = min(floor(2*Fs/f),Mmax);
    g1 = (f + 27);
    for m=1:M
        g = g1 / (m*f + 320);
        km = round(freq2bin(m*f,Fs,K));
        k = max(1,km-2):min(Kp,km+2);
        Yd(k) = Yd(k) + g*Yr(k);
        Yr(k) = max(0, Y(k)-Yd(k));
    end

    % Determine whether to continue (polyphony estimation)
    if Np == -1
        st(j) = s;
        S(j) = sum(st)/j^0.7;
        if (j > 1 && S(j) <= S(j-1)) || j > 5
            break;
        end
    end

    % Record the finding
    P(j) = p;
    V(j) = s;
    if j == Np
        break;
    end
    j = j+1;
end

if length(P) < Np
    P = [P,zeros(1,Np-length(P))];
    V = [V,zeros(1,Np-length(P))];
end

end
```

And this listing shows the pre-calculated system parameters.

```
% Decimated sampling frequency
Fs = 14700;

% Window size (50ms)
Kw = 735;

% Hanning window
w = hamming(Kw)';
```

```
% FFT length
K = 1024;

% Positive-sided spectrum length
Kp = round((K+1)/2);

% Crossover bin
kc = 1+ceil(1/(2*(1-2^(-1/24))));

% Peak-picking threshold
thresh = 1/2;

% Maximum number of harmonic multiples
Mmax = 20;

% Number of critical bands
B = 30;

% Generate the whitening filters
[H c] = swfilters(B,K,Fs);

% Pitch range
pmin = 36;
pmax = 96;
```

Bibliography

Bibliography

[1] A. Klapuri and M. Davy, Eds., *Signal Processing Methods for Music Transcription*, 1st ed. Springer, 2006.

[2] T. D. Rossing, *The Science of Sound*, 2nd ed. Addison Wesley, 1990.

[3] A. Klapuri, *Signal Processing Methods for Music Transcription.* Springer, 2006, ch. Introduction to Music Transcription, pp. 3–20.

[4] D. Deutsch, Ed., *The Psychology of Music*, 2nd ed. Academic Press, 1999.

[5] *The Complete MIDI 1.0 Detailed Specification*, The MIDI Manufacturers Assocation Std., 2001. [Online]. Available: http://www.midi.org

[6] Music frequency diatonic scale. Public Domain. [Online]. Available: http://en.wikipedia.org/wiki/File:Music_frequency_diatonic_scale-3.svg

[7] P. Herrera-Boyer, A. Klapuri, and M. Davy, *Signal Processing Methods for Music Transcription.* Springer, 2006, ch. Automatic Classification of Pitched Musical Instrument Sounds, pp. 163–200.

[8] A. Klapuri, *Signal Processing Methods for Music Transcription.* Springer, 2006, ch. Auditory Model-Based Methods for Multiple F0 Estimation, pp. 229–265.

[9] M. Davy, *Signal Processing Methods for Music Transcription.* Springer, 2006, ch. Multiple Fundamental Frequency Estimation Based on Generative Models, pp. 203–227.

[10] ——, *Signal Processing Methods for Music Transcription.* Springer, 2006, ch. An Introduction to Statistical Signal Processing and Spectrum Estimation, pp. 21–64.

[11] B. Logan and S. Chu, "Music summarization using key phrases," in *IEEE Internation Conference on Acoustics, Speech, and Signal Processing*, vol. 2, 2000, pp. 749–752.

[12] V. Tyagi and C. Wellekens, "On desensitizing the mel-cepstrum to spurious spectral components for robust speech recognition," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 2005, pp. 529–532.

[13] A. Klapuri, "Multiple fundamental frequency estimation by summing harmonic amplitudes," in *International Conference of Music Information Retrieval (ISMIR)*, 2006, pp. 216–221.

[14] A. Dessein, A. Cont, and G. Lemaitre, “Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence,” in *International Society for Music Information Retrieval (ISMIR)*, 2010, pp. 489–494.

[15] L. R. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” in *Proceedings of the IEEE*, vol. 77, no. 2, 1989, pp. 257–286.

[16] D. P. W. Ellis, “Prediction-driven computational auditory scene analysis,” Ph.D. dissertation, Massachusetts Institute of Technology, 1996.

[17] T. Tolonen and M. Karjalainen, “A computationally efficient multipitch analysis model,” *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 6, pp. 708–716, 2000.

[18] R. Meddis and M. J. Hewitt, “Virtual pitch and phase sensistivity of a computer model of the auditory periphery,” *Journal of the Aco*, vol. 89, no. 6, pp. 2866–2894, 1991.

[19] R. D. Patterson, “How complex sounds are represented in the auditory system,” *Journal of the Acoustical Society of Japan (E)*, vol. 21, no. 4, pp. 183–190, 2000.

[20] R. P. Carylon, “Temporal pitch mechanisms in acoustic and electric hearing,” *Journal of the Acoustical Society of America*, vol. 112, no. 2, pp. 621–633, 2002.

[21] P. Cariani, “Recurrent timing nets for auditory scene analysis,” in *International Joint Conference of Neural Networks*, Portland, Oregon, July 2003.

[22] C. Yeh and A. Röbel, “A new score function for joint evaluation of multiple f0 hypotheses,” in *International Conference on Digital Audio Effects*, Naples, Italy, October 2004.

[23] M. K. U.K. Laine and T. Altosaar, “Warped linear prediction (wlp) in speech and audio processing,” in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 1994, pp. III.349–III.352.

[24] A. Härmä and M. Karjalainen. Warptb - matlab toolbox for warped dsp. [Online]. Available: http://www.acoustics.hut.fi/software/warp

Curriculum Vitae

John M. Thomas received his Bachelor of Science in Computer Science and Engineering from LeTourneau University in 2006. He is now a Senior Consultant with Booz Allen Hamilton in McLean, VA, as a software engineer. Prior to joining Booz Allen, Mr. Thomas was a software engineer for Lockheed Martin in Manassas, VA