VIRGINIA COMMONWEALTH UNIVERSITY

Virginia Commonwealth University
VCU Scholars Compass

Theses and Dissertations | Graduate School

2011

Real-Time Fundamental Frequency Estimation Algorithm for Disconnected Speech

Thomas Skjei
Virginia Commonwealth University

Follow this and additional works at: http://scholarscompass.vcu.edu/etd

 Part of the Computer Sciences Commons

© The Author

Downloaded from
http://scholarscompass.vcu.edu/etd/191

This Thesis is brought to you for free and open access by the Graduate School at VCU Scholars Compass. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of VCU Scholars Compass. For more information, please contact libcompass@vcu.edu.

Real-time Fundamental Frequency Estimation Algorithm for Disconnected Speech

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science at Virginia Commonwealth University.

By

Thomas Skjei

Director: Kayvan Najarian
ASSOCIATE PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE

Virginia Commonwealth University
Richmond, Virginia
April, 2011

Table of Contents

Abstract

REAL-TIME FUNDAMENTAL FREQUENCY ESTIMATION ALGORITHM FOR DISCONNECTED SPEECH

By Thomas Skjei, M.S.

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2011

Director: Kayvan Najarian
ASSOCIATE PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE

A new algorithm is presented for real-time fundamental frequency estimation of speech signals. This method extends and alters the YIN algorithm, which uses the autocorrelation-based difference function, by adding features to reduce latency, correct predictable errors, and make it structurally appropriate for real-time processing scenarios. The algorithm is shown to reduce the error rate of its predecessor while demonstrating latencies sufficient for real-time processing. The results indicate that the algorithm can be realized as a real-time estimator of spoken pitch and pitch variation, which has applications including diagnosis and biofeedback-based therapy of many speech disorders.

Executive Summary & Contributions:

Pitch detection algorithms (PDA's) have been an active research topic for more than 40 years. A wide variety of methodologies have been employed but their success has been generally domain-specific. This fragmented success underlies not only the complexity of the task, but the breadth of domains that have use for such algorithms. The determination of a fundamental period of some complex signal is one of the most elemental questions posed in the field of signal processing. The research presented here limits the scope of this question to a specific domain: real-time disconnected (i.e. with breaks between words) speech. It employs a novel strategy towards achieving these ends, uses an evaluation methodology consistent with the existing body of PDA research, and produces results which satisfy its stated goals and contributes to the existing literature in the field, particularly for real-time speech processing.

The primary goal of this research is the development of an algorithm that can accurately estimate the fundamental frequency (F0) of a disconnected speech signal in real-time. This requires the algorithm to maximize accuracy while minimizing latency, and to make a voiced/unvoiced decision indicating whether the speaker is currently speaking. There are many PDA's developed primarily for analyzing musical signals and which mention their feasibility for speech domains as well, but their success here has been limited [6, 11, 3]. Additionally, there is very little work targeting speech exclusively. Since the harmonic content of speech often contains more ambiguities and complexities than musical signals, this suggests the need for a more a domain-focused approach.

The development strategy for this algorithm was to start with an existing method that had moderate success in both speech and music domains and then to significantly change and improve it in ways that make it more appropriate to real-time speech processing. The strategy taken in the current work can be summarized into the following steps:

- Selection of a base algorithm: an algorithm that is relatively robust in both music and speech domains is selected. Particular preference is given to an algorithm that "degrades gracefully" (i.e. continues to perform well as the input signal's resolution is decreased).
- Address the problem of latency by developing an algorithm that is capable of processing at a lower sampling rate: Both speech databases used in this research contain speech sampled at 20 kHz, so a target of 10 kHz is used.
- Narrow the assumptions for the range and characteristics of the input signals: Since the algorithm is intended for speech, which has a limited frequency range (compared to music, for example), additional latency and accuracy improvements may be found by narrowing the allowable frequency band.
- Reduce the frequency-matching resolution: Since the base algorithm uses an autocorrelation variant, increasing the distance between lags will decrease frequency resolution, while yielding large latency improvements. This work forms a balanced tradeoff in a way that is appropriate to the given problem.
- Find the most predictable errors as a result of these degradations and implement error correction steps.

To provide a substantive evaluation, the methodology was consistent with the norms established in the PDA literature. This includes the following:

- The use of standard error definitions for assessing accuracy.
- The use of 2 widely-cited speech analysis databases for speech samples and laryngograph-based analyses which define the ground truth frequencies.
- Complete separation of sample files used for parameter tuning versus evaluation.
- Comparison of two main algorithms in the field: Yin (the base algorithm) and WavePitch, a wavelet-based algorithm used in for speech and music signals.

The results of this research may be summarized as follows:

- The new algorithm outperformed both algorithms in terms of accuracy and demonstrated latency sufficient for processing in a real-time environment.
- The new algorithm appears to meet the primary goals stated previously and is therefore a suitable PDA for real-time disconnected speech.

1.0. Background

1.1. Motivation

There are two groups of applications where a real-time PDA for disconnected speech can be extremely useful. The first group involves systems which can recognize prosodic features of language in real-time. Prosody refers to the rhythm, stress, and most importantly, the intonation (i.e. pitch variation) of language. While most current speech recognition systems ignore intonation, it is well known that it carries a great deal of information [6]. In addition to speech recognition systems, hearing impairment devices which can identify and emphasize intonation could provide many benefits.

The second group of applications involves systems which provide immediate pitch or intonation-based feedback for diagnostic or therapeutic purposes. There are a variety of pitch-related speech impairments where this can be useful. One such impairment is 'Dysprosody', which refers to the inability to control some prosodic aspect one's speech. This is common in people afflicted with Parkinson's disease, people with profound hearing loss, and a variety of other rare speech impairments [27, 28]. Another speech impairment which could benefit from real-time pitch feedback software is 'Muscle Tension Dysphonia', which refers to an excess of tension in the muscles around the larynx [29]. This can result in a loss of pitch control in one's speech. Software providing real-time pitch feedback could aid the therapy and diagnosis of both of these conditions and ultimately improve a patient's ability to communicate through speech [30].

1.2 Historical Background

Although the range of methodologies used by PDA's is tremendous, there have been 4 main approaches: time, frequency, and wavelet-domain methods as well as statistical methods.

Time-domain approaches generally fall into 1 of 3 classes:

1) Event-rate detection
2) Phase-space methods
3) Autocorrelation-based approaches

Event-rate detection models count some specific event (e.g. the number of zero-crossings) over time and use that to infer the period. Phase-space methods consider the waveform values vs. the slope over a short-time history and attempt to infer period from any repetitive cycles [6]. Both of these methods have had limited success and academic interest.

The most important time-domain PDA methodology has been autocorrelation [10]. The algorithm presented in this paper belongs to the autocorrelation family, so the mathematical details will be discussed in more detail later (sec. 2.3.4.1). However, the main idea of autocorrelation is to look for the maximum of a signal multiplied by itself at various lags. The lag corresponding to the maximum value indicates the period. This method works well for simple, periodic signals, but in more complicated scenarios such as speech signals, which contain complex harmonic content, it may fail [7]. The most common mistakes occur when correlating with the zero lag [1, 7]. Other known issues include when the maximum corresponds to some partial of the fundamental or when there is variability in the amplitude [7].

Despite the mixed results of classic autocorrelation, there have been many enhancements to help it recognize fundamental frequencies of complex and otherwise problematic signals. One such enhancement is Average Magnitude Difference Function (AMDF) [8] which calculates the difference magnitudes rather than the product of the signal lagged with itself. YIN [1] is another PDA that extends AMDF by adding a series of error correction steps. YIN forms the basis of the new method presented here and will be described in greater detail in section 2.3.

The most prominent frequency-domain PDA is the Cepstral method [12]. The Cepstral method takes the Fourier Transform of the log of the magnitude spectrum. One benefit is that it can be performed efficiently and on smaller windows than autocorrelation-based methods. However, it fails when there are too many high-energy upper partials or when the pitch is sufficiently low. It may also fail if the voiced pitch is sufficiently high, as it will contain less harmonics. In this case, the voiced/unvoiced decision will also likely fail as it is based on a thresholding of its cepstral peaks.

Recently, much attention has been given to wavelet-domain methods. One of the methods evaluated in this paper is WavePitch [5]. WavePitch implements the fast-lifting wavelet transform to decompose a signal into approximations and details. The approximation is then used to estimate the period. As will be shown in this study, it is a very fast algorithm that fits the frequency contour exceedingly well, but has trouble with the voicing decision.

Another school of thought for PDA's is to view the problem as a statistics and/or machine learning problem. As such, maximum likelihood estimators and neural net-based approaches have been used. Both methods have been moderately successful

but their popularity has been limited because of their requirement of training to the speaker's voice and due to the black box model implemented by the NN-based approach.

1.3 Solution

The method presented in this paper constitutes a new algorithm that is a significant extension and alteration of the YIN algorithm. The fundamental strategy in adapting YIN to real-time disconnected speech processing is:

1) Modify existing components to be compatible with the assumptions made regarding the requirements of real-time environments (section 2.2.2.2).
2) Find the least destructive means of reducing the algorithm's resolution, to provide decreased latencies.
3) Narrow the restrictions on the input signal (e.g. assume it must be speech, not music), which constrain the resulting error types and provides shorter search-spaces for decreased latencies.
4) Implement error correction steps for the most predictable set of errors resulting from the previous changes.

YIN was selected as a starting point for a few reasons:

1) Its relatively robust performance and speed with musical and speech signals.

2) It's simple pipeline architecture would be easy to extend and alter for real-time requirements.
3) It takes a measure of aperiodic power at each window, which can be used as a basis for a voiced/unvoiced decision.
4) As a time-domain based algorithm, it is fairly intuitive in design and implementation.

Just as YIN is fundamentally an extension of the AMDF[1, 10], this new algorithm is an extension of YIN. For that reason, the algorithm introduced in this study is referred to as 'YinRT' where 'RT' is an acronym for 'real-time'.

1.4. Summary of Process & Results

Three algorithms were compared for accuracy of pitch estimation based on 2 standard speech frequency-evaluation databases. Each database contains a collection of recorded speech samples along with corresponding captures of activity in the vocal folds as recorded by a laryngograph. Both databases provide a manually-checked fundamental frequency analysis of each laryngograph capture, defining the ground truth fundamental frequencies for each speech sample. The parameters used for the other 2 algorithms were the default parameters suggested by the respective authors, although two parameters- window size and input sampling rate, were made uniform across each algorithm. For the 2 real-time algorithms (YinRT and WavePitch), processing time as a percentage of file length was also reported.

The results showed YinRT to be the most accurate algorithm in every comparison variation. WavePitch was the fastest algorithm, executing about 4% the

length of the file length, however YinRT consistently executed at 15% of the file length size, which is sufficient for real-time processing.

2. Methods

2.1 Data

Two databases containing speech samples and analyses were used in this research: the Fundamental Frequency Determination Algorithm (FDA) Evaluation Database [2], provided by the Centre for Speech Technology Research at the University of Edinburgh and the Keele Pitch Database [9], provided by Keele University and the University of Liverpool School of Psychology. These freely available databases are widely used in PDA research and development and so it provides some common ground for comparison between works.

The FDA database provides 0.12h of clean, disconnected speech recordings with corresponding laryngograph captures and frequency analyses of the raw captures. One male and one female speaker were recorded reciting 50 identical sentences, specially selected to contain a wide phonographic range, while at the same time a laryngograph recorded the excitation energy of the vocal folds. Each recording contained one sentence for a total of 100 audio recordings and 100 laryngograph captures. The laryngograph signals were analyzed using in-house software and produced a counterpart pitch-tracking file. The pitch-tracking files contain a sequence of timestamps and corresponding frequency values, with special identifiers for breaks in speech. The speech recordings were sampled at 20 kHz with a bit-depth of 16 bits and the laryngograph was sampled at 20 kHz with a bit-depth of 12.

The Keele database provides 0.15h of clean, disconnected speech recordings of 5 male and female English speakers reading a phonetically balanced text. Corresponding

laryngograph recordings and pitch analysis files are provided for each audio sample. The pitch analysis files define the ground truth fundamental frequencies and are based on the laryngograph but with manual checking to identify anomalies or noise introduced into the laryngograph. Unlike the FDA Database, the Keele pitch analysis files (ground truth) are sampled at a constant rate- 100ms. The speech and laryngograph recordings were sampled at 20 kHz with a bit-depth of 16 bits.

2.2. Evaluation

The proposed algorithm was evaluated with two separate, but equally important measurements:

1) Accuracy of pitch estimations

2) Processing latency.

Additionally, a method for parameter-tuning was required to select the best parameters prior to comparison with other algorithms.

2.2.1. Accuracy

2.2.1.1. Measurements

To evaluate the accuracy of a pitch tracking algorithm, a class of measurements must first be defined which can quantitatively characterize accuracy. Though there is some variation in the literature of PDA research regarding what measurements should be used and what their particular definition should be. As such, four prominently agreed-upon measurements were used in this study [3, 25, 1].

These 4 measurements can be divided into 2 categories: voiced and unvoiced errors. The former involves an error in a non-zero pitch estimation when the subject is speaking and the latter indicates a disagreement regarding whether a given processing window is voiced or not.

The two types of voiced errors are gross and fine errors. A gross error indicates that a non-zero pitch estimation differs from the actual frequency by +/- 10% when the ground truth is voiced. A fine error occurs when any non-zero pitch estimation differs from the voiced, ground truth frequency by less than +/- 10%.

The two types of unvoiced errors are called ‘type 1’ and ‘type 2’. A type 1 unvoiced error occurs when the actual speech is unvoiced but the PDA’s estimation is non-zero and a type 2 unvoiced error is the converse: the PDA considers the speech region to be unvoiced, when in reality it is not.

2.2.1.2. Defining Ground Truth Fundamental Frequency (F0)

There’s considerable variation in the PDA literature about how to define or acquire the actual F0. The lack of consensus on this point is understandable considering the natural periodic ambiguities and complexities of speech. Some studies use the frequency analysis of the laryngograph signal and others prefer to process the laryngograph using their specific methods [1].

In this study, F0 is defined using the frequency analyses provided by both databases, but this introduces some issues which are resolved as follows.

The first issue is the evaluation rate mismatch between the PDA and the frequency analysis provided by the FDA database. The PDA will produce evaluations at some

fixed, predetermined rate; however the frequency analysis file's evaluations are not strictly periodic. The difference in timestamps for adjacent evaluations varies. To solve this, a timestamp is derived for the PDA's evaluation and then linear interpolation is used for the adjacent ground truth estimates to determine a corresponding actual frequency.

One complication to the linear interpolation strategy occurs when one of the adjacent ground truth values is zero. If not handled as a special case, linear interpolation would find some new ground truth frequency somewhere between the voiced and unvoiced sample, weighted by its proximities to each. This would not represent the actual transition from voiced to unvoiced (or vice versa). The solution is to select the value of whichever adjacent ground truth estimate the evaluation timestamp is closer to. Thus,

$$F0_{cur} = \frac{\left((TS_{cur} - TS_{G1}) * F0_{G2} + (TS_{G2} - TS_{cur}) * F0_{G1}\right)}{(TS_{G2} - TS_{G1})} \qquad | F0_{G1}, F0_{G2} \neq 0,$$

$$= F0_{G1} \qquad |TS_{cur} - TS_{G1} \leq TS_{G2} - TS_{cur},$$

$$= F0_{G2} \qquad |TS_{G2} - TS_{cur} < TS_{cur} - TS_{G1} \qquad (1)$$

where $F0_{cur}$ is the interpolated ground truth frequency to solve for with timestamp, TS_{cur} and $F0_{G1}$ and $F0_{G2}$ are the adjacent ground truth frequencies with corresponding timestamps, TS_{G1} and TS_{G2}.

A final interpretive complexity regarding our definition of ground truth is that the provided frequency analysis is based on the laryngograph signal and it does not account for any time lag due to the acoustic propagation from the vibrating vocal cords through the vocal tract to an audible, recorded utterance. This situation is resolved by

defining a range of lag times and evaluating the PDA's estimations with the ground truth delayed by each lag time. The final accuracy measurements reported correspond to the best performing propagation lag.

2.2.1.3. Comparison to other PDA's

A comparison to other established PDA's provides a context for YinRT's performance to be judged. Two other PDA's are used: Yin [1] and WavePitch [5]. YIN (discussed earlier) is the basis for the current algorithm, so its relative performance should reveal the tradeoff's entailed by constraining the new algorithm to the requirements of real-time processing. WavePitch is a new, wavelet-based PDA which shows promise regarding its speed and relative accuracy.

2.2.2. Latency

2.2.2.1. Defining a Real-time Criteria

A real-time processing algorithm needs to minimize latency below some threshold, but defining that threshold is subject to research. In this study, a simpler but reasonable approach to this threshold is used. The total time processing each batch of raw audio samples is recorded and this time is required to be less than the given audio recording's length multiplied by some fixed percentage. This requirement is elaborated in the parameter-tuning section. This percentage is set to a conservative value to allow for variations in processing time due to platform-specific constraints.

2.2.2.2. Assumptions for Real-time environment

The specific mechanisms for delivering real-time audio samples to some process vary across environments. Data can be delivered at the sample, window, or batch-of-samples level. Consequently, YinRT assumes constraints that make it widely compatible across such environments. Specifically, the algorithm is designed to be compatible in a “batch-dump” environment as well as a “per window” environment.

The “batch-dump” environment is the more permissive of the two and it assumes that the given audio capture device will deliver a batch of samples of arbitrary length. This is common in most architectures because of the I/O inefficiencies of delivering per-sample data to a process. In this model, it is acceptable to look ahead at data in the batch, beyond the boundaries of the given window. It is also acceptable to look backwards in the batch, beyond the window boundary, at both raw and evaluation data. Both of these rules are subject to the real-time latency requirements discussed above.

The “per window” environment is more restrictive and provides some important constraints. First, there are no look-aheads beyond the boundary of the current window. That is because it is assumed only 1 window of data is passed to the algorithm at a time. Secondly, look-behinds are limited to the previous evaluation. This permits some error correction without imposing significant overhead.

Thus YinRT is capable of receiving batches of samples and windowing the data itself, or it can receive each window of data. In both cases, its look-ahead permission is limited to the current window and its look behind permission is limited to the previous evaluation.

2.2.2.3. Measurements

The proposed algorithm records the wall-clock time when it begins windowing data. Prior to that point, the data is still raw and unprocessed. The algorithm records the wall-clock time when all windows have completed processing and uses the difference as the measurement of processing latency for the given recording. Then this value is divided by the length of the sample file and the resulting percentage is reported.

2.2.3. Parameter Tuning

Prior to the accuracy and latency-based comparisons, YinRT goes through a parameter tuning process. The algorithm contains the following 8 parameters:

1) Min F0
2) Max F0
3) Low pass Filter Threshold
4) Difference Function Threshold
5) Unvoiced Threshold
6) Window Overlap
7) Downsample Factor
8) Lag Scaling

This relatively large number of parameters poses a problem for tuning the algorithm. This many parameters would not be problematic provided the parameters were decoupled from each other. Were this the case, a feasible parameter tuning strategy would be to evaluate one parameter at a time, holding all others as constant, then to use the best combination of parameters. But there is a high degree of interdependence

among these parameters and so this complicates the parameter tuning situation. As a result, an exhaustive search of parameters is necessary. However, the number of iterations in a coarse-search of parameters combinations can be in the hundred-thousands.

This problematic overhead for parameter tuning is resolved by the real-time latency constraints. Because this study is not interested in evaluating the accuracy of parameter settings which are not sufficiently fast, if any parameter combination fails to meet the real-time latency constraints for a single file in the corpus, then that combination's evaluation is terminated and the next combination is selected. Any parameter combination whose total processing time exceeded 70% of the actual length of the recording was considered too slow. As a result, hundreds of parameter combinations were able to be evaluated in a couple minutes.

Once the parameter tuning process was complete, the data was inspected and the most accurate combination was selected. If there was a tie, the quickest combination would win.

Since there are four measurements of accuracy, assessing the 'most accurate' combination presented a challenge as to whether the measurements should be weighted equally or not. The conclusion was to find the parameter combination with the best overlap. More specifically, each error measurement was sorted by accuracy for each parameter combination. The parameter combination that had the earliest placement in all groups was selected.

To avoid any overfitting of the data in the parameter tuning process, separate training data was used from each database and these files were not included in the final comparison.

2.3. Proposed Method

YinRT is implemented as a pipeline of signal analysis and error correction steps which build upon one another. The structure is summarized in figure 1.

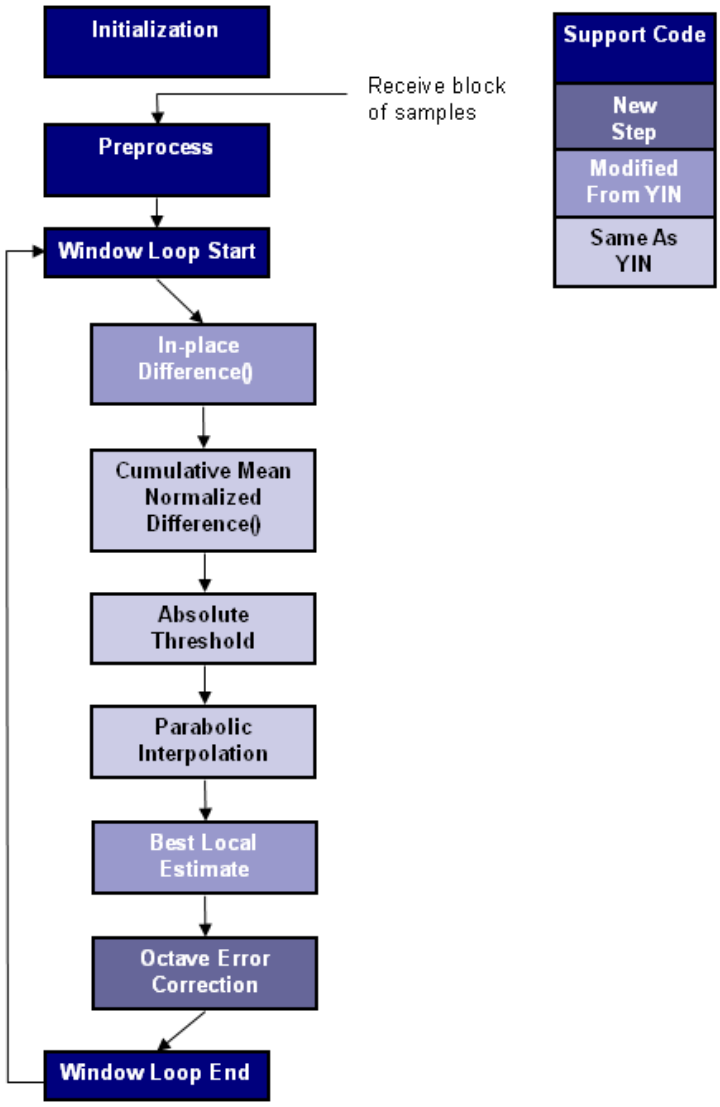

Figure 1. YinRT Block Diagram (Pipeline)

2.3.1. Initialization

During initialization, YinRT uses the input parameters to allocate data structures and set some key values. Many of the parameters are specific to a particular processing step, and so they will be discussed in the context of that step. However, there are 2 parameters which are used throughout the algorithm and should be discussed here:

1) Window Size

2) Window Overlap

Window size is the length, in number of samples, of the processing window. The processing window is simply a contiguous block of samples that is used to create one estimate. The larger the window size, the more information you have to calculate a frequency, however it also increases the risk that the frequency may change within the window, creating an ambiguity for the PDA. The standard window size for autocorrelation-based methods is two times the minimum frequency.

Window overlap specifies the percentage of overlap that processing windows should have. For non-overlapping windows, this value is 0. In a non-overlapping configuration, one window starts on the sample after the previous window's ending sample. Although non-overlapping windows have a computational advantage of less windows, overlapping windows enable some special error correction possibilities, such as the Best Local Estimate. Also, the extra computational disadvantage based on redundant calculation of overlapping regions can be minimized using some special techniques.

The number of samples between windows is known as the evaluation interval or simply the ‘hop’. This number is the closest even number which corresponds to the specified window overlap percentage parameter.

2.3.2. Preprocessing

The preprocessing step begins when a batch of audio samples is provided to the algorithm (presumably from the audio capture device, although in our implementation, this is simulated). There are two main actions during the preprocessing step:

1) Downsample the batch, as specified
2) Low-pass filter the batch, as specified

Downsampling means to keep samples based on a specified interval and to discard the rest. It can greatly reduce the amount of data being processed. Note that the input audio is sampled at 20 kHz. Also note that most phone systems sample at 8 kHz and often humans can still identify pitch while talking on the phone. Thus one could hypothesize that downsampling to 10 kHz (i.e. downsampling by a factor of 2) may provide latency improvements without affecting accuracy too much.

After downsampling, the next preprocessing step is to low-pass filter the block of samples. The threshold for the low-pass filter is specified as one of the input parameters. Removing the jitter from the raw audio greatly simplifies subsequent processing.

Note that downsampling only occurs if specified by an input parameter but that low-pass filtering always occurs, though the specific threshold is specified by another input parameter.

2.3.3. Window Loop Start

The window loop start breaks the sample batch into successive windows separated by the number of samples specified by the ‘hop’ value until the batch is exhausted. Each window gets processed by the following 6 steps (per fig. 1), until control returns to the loop and then next window is forwarded.

2.3.4. In-place Difference Function

The core algorithm of the proposed method is the calculation of the difference function over a specified range of lags. The difference function is based on the autocorrelation function, and so for a proper background, autocorrelation is presented first.

2.3.4.1. Autocorrelation

Autocorrelation is a time-domain based algorithm which measures the “similarity” of a signal with itself at different time lags.

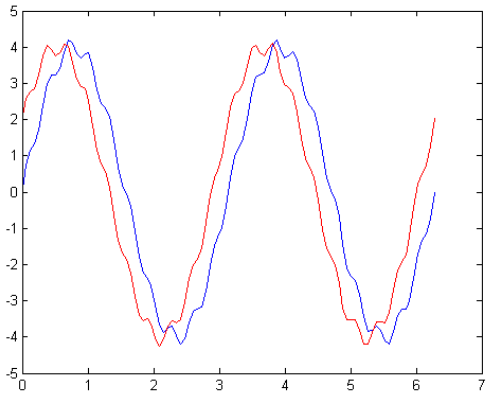

Figure 2: An original signal and a lagged copy of itself.

This similarity can be computed simply as the summation of products of the signal, x_j and its lagged self, $x_{j+\tau}$:

$$r_t(\tau) = \sum_{j=t}^{t+W-1} x_j x_{j+\tau} \quad (2)$$

Where τ indicates the given lag, W equals the window size, and t equals the time at which the autocorrelation is computed.

Under the right conditions, the autocorrelation function can determine the fundamental frequency of a periodic signal. The frequency search range must be defined by a max and min frequency (f_{max} and f_{min}) and this translates into the lag range such that:

$$\tau_{min} = SR / f_{max} \quad (3)$$

$$\tau_{max} = SR / f_{min} \quad (4)$$

Where SR = sampling rate (Hz).

Subject to certain constraints, the specific lag τ, which produces the maximum value of $r(\tau)$ from τ_{min} to τ_{max} will equal the period of the fundamental frequency.

2.3.4.2. The Difference Function

Unfortunately, there are a variety of circumstances which can cause autocorrelation to fail. This includes selecting a zero-lag if the τ_{min} is too small or selecting a higher or lower order peak due to the harmonic content or amplitude variation of the signal.

An alternative which avoids many of these pitfalls is found in the difference function. Rather than summing the products of a signal with it's lagged self as with

autocorrelation, the difference function sums the squared differences of a signal with its lagged self:

$$d_t(\tau) = \sum_{j=t}^{t+W-1} (x_j - x_{j+\tau})^2 \quad (5)$$

Where the difference d at time t and lag τ equals the summation of the difference of signal x with its lagged self over the window t to t+W-1, and W equals the window size.

Rather than searching for the maximum value of *d*, as was the case with autocorrelation, the difference function estimates the fundamental frequency as the lag τ with the minimum value. This corresponds with the commonsense idea that a periodic signal, when subtracted from itself at lags equal to the period, will be zero.

The difference function is obviously closely related to the autocorrelation function. This relationship can be expanded by expressing the difference function in terms of the autocorrelation function:

$$d_t(\tau) = r_t(0) + r_{t+\tau}(0) - 2r_t(\tau) \quad (6)$$

In this case the first two terms are the ACF of the signal at the zero lags, at times separated by τ and then subtracted by the 2 times the autocorrelation at time t lagged by τ.

2.3.4.3 Implementation of the Difference Function

The difference function calculated directly is expensive, especially from the perspective of a real-time application. There are two strategies to calculate it more efficiently. These strategies are:

1) Eq (6) with FFT-based autocorrelation.

2) “In-place” method to exploit overlapping regions.

The first strategy relies on the fact that autocorrelation can be calculated directly using a Fast Fourier Transform. In a general sense, this is based on the similarities between correlation and convolution. A more specific description is as follows:

1) Perform two FFT’s on the signal *x.*

2) Multiply one *X(f)* by the complex conjugate of the other.

3) Perform the IFFT on the result of step 2.

4) Although the result will be complex, the imaginary part will be zero. The values of this result will be the autocorrelation value at different lags.[4]

Using this technique in combination with eq (6), an efficient way to calculate the difference function was implemented. But upon closer inspection, eq (6) is not well suited to our problem as it requires calculation of the middle term $r_{t+\tau}(0)$. This implies that we would calculate the zero-lag autocorrelation, at points $(t + \tau_{min})$ through $(t + \tau_{max})$. However in the proposed windowed setting, the difference function at every $(i * hop)$ where i is an integer > 0 is calculated. Thus the proposed algorithm would multiply the number of autocorrelations one needs to calculate drastically when the window overlap percentage was not extremely high.

The other means to quickly calculate the difference function is to use ‘In-place’ calculation. This method calculates the difference function directly per eq (5), but it divides the process into two steps to take advantage of overlapping regions.

The first step of the ‘In-place’ calculation is to divide the signal into sequential, non-overlapping regions of size *hop* and calculate the difference function across the lag range for each region. Note that the YIN implementation, which uses ‘In-place’ calculation, this is calculated over the full range of the signal at once, whereas YinRT, due to the constraints of the “per window” real-time assumption only processes the current window. Furthermore YIN is free to select any lag range so long as it stays within the signal’s boundaries, while YinRT is again constrained to the current window’s data.

The second step of the ‘In-place’ calculation is to sum the previous per region values of the difference function for each region contained within the current window. The greater the overlap in windows, the greater the advantage because overlapping regions are only calculated once. Conversely, if the algorithm is configured for non-overlapping windows, the ‘In-place’ method is equivalent to a naive implementation of eq (5). But YIN and YinRT are optimized for some amount of overlapping windows as this enables the Best Local Estimate method, which is a valuable error correction step and ensures that estimations are stable within their given period. With non-overlapping windows, the Best Local Estimate method could not be performed.

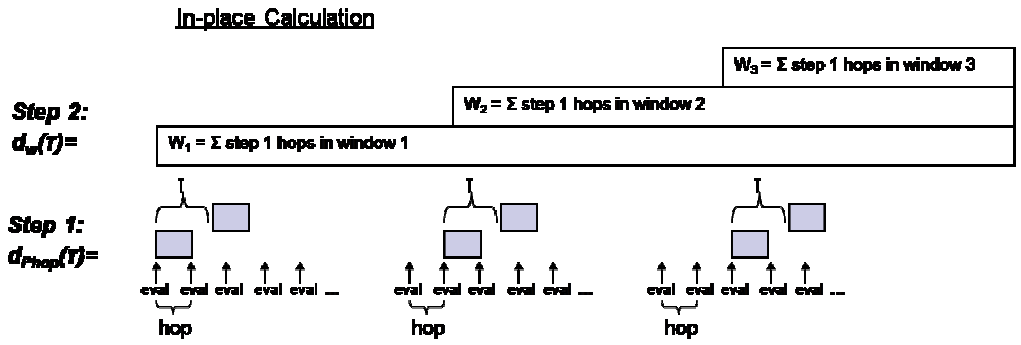

Figure 3: In-place Calculation’s 2 steps

Another interesting aspect of the ‘In-place’ method of calculating the difference function is that the intervals used to calculate step 1 must evenly divide step 2. That is to say that the window size must be divisible by the *hop* value. YIN and YinRT handle this situation differently. YIN lets any *hop* amount be specified and augments the window length by the minimum amount necessary to be evenly divisible by *hop*. YinRT allows any even value of *hop* to be specified and then calculates step 1 using regions with a size equal to the greatest common divisor of window length and *hop*. Worst case, step 1 could be calculated every 2 samples. This decision was made because YIN’s extension of the window length limits its capability for accuracy whereas the YinRT method of using subdivided regions for the initial difference function calculation only affects the memory consumption of the algorithm, but not its capability for accuracy. This downside is probably less pronounced in YIN because of its sophisticated error correction steps, but YinRT is only able to implement a weakened subset of these steps due to its “per window” real-time assumption.

2.3.4.4 Lag Scaling

The proposed method also includes a parameter called ‘Lag Scaling’ which allows the difference function to be calculated using lags at a specified interval. Normally, there will be a minimum and maximum lag value and each value within this range will be calculated. If the lag value is incremented by one sample for every value in this range, then the estimation process has a resolution equal to the sampling rate. This resolution comes at the cost of an increased search range for later processing steps. The lag scaling parameter attempts to trade off some resolution for a smaller search space. Various lag scaling values are experimented with in the parameter tuning section.

2.3.5. Cumulative Mean Normalized Difference

Once the original difference function is calculated, the subsequent processing steps effectively provide error-corrections for circumstances which are known to cause the minimum *d* to not correspond to the fundamental frequency. The first such error correction step is the Cumulative Mean Normalized Difference calculation. This step protects against selecting a too high frequency. There are two common scenarios in which this might occur:

1) Selecting a dip in *d* based on its proximity to the zero-lag rather than the actual fundamental.
2) Selecting a dip in *d* correspond to a highly resonant group of harmonics of the fundamental (i.e. a formant)

Both of these errors are enabled by the fact that the value of *d* might be nonzero at the fundamental frequency (most likely due to complex harmonic content).

The solution to this situation is to first normalize the difference values by dividing them by their cumulative means. The cumulative mean is simply the average difference value for all lag values less than the current lag. Additionally, all lags shorter than τ_{min} are set to 1 to deter a correlation with the 0-lag dip. The cumulative mean normalized difference function is expressed:

$$d'_t(\tau) = 1 \qquad | \tau < \tau_{\min}$$

$$d'_t(\tau) = \frac{d_t(\tau)}{(1/\tau)\sum_{j=1}^{\tau} d_t(j)} \qquad | \tau \geq \tau_{\min} \tag{7}$$

Where $d'_t()$ equals the new cumulative mean normalized difference value, based on the old difference value $d_t()$.

2.3.6. Absolute Threshold

While the cumulative mean normalized difference step protects against too high errors, the absolute threshold protects against a too low error. If one searches for the global min across all lags, due to imperfect or complex periodicities and harmonic content, a lag greater than the fundamental will often be selected and thus the frequency estimate will be too low. The absolute threshold protects against this situation by progressively bounding the upper search range based on the minimum values it sees. It is essentially a 2-pass algorithm that can be described as follows:

1) Find the global min and increment the configured threshold by this value.
2) Start a min search from the beginning again but reduce the upper boundary by a fixed factor every time you hit a cumulative min which is less than the new threshold value.

The output of this step contains two important values: a lag value corresponding to the minimum $d'()$ and the value of $d'()$. The former is the period estimate and the latter can be thought of as analogous to the aperiodic power [1].

One of the parameters of YinRT is an unvoiced threshold, and the voiced/unvoiced determination is based on whether this *d'*() value is greater or less than the threshold. If it is greater, this implies that the cumulative mean normalized difference min was not very small and thus at the selected lag, there were still considerable differences between the signal and it's lagged self. This is considered an indication of aperiodicity and so the given window is considered to be unvoiced.

2.3.7. Parabolic Interpolation

One assumption of the signal processing up until this point is that the actual fundamental frequency is a multiple of the sampling rate. In this case, one can graphically imagine a pitch contour following a step-wise progression up and down. Clearly, this isn't the case and so we use parabolic interpolation to find the best value.

Parabolic interpolation exploits the periodicity of the difference function such that when a frequency estimate is provided, the difference function value at the corresponding lag should be the minimum of a parabola. The difference values on either side must be larger or they would have been selected as the period estimate, and therefore the current lag must represent a dip in the difference function. Then through the well-known technique of parabolic interpolation, a parabola is fit within these 3 values and the new minimum represents a non-integer lag offset which is applied to the original lag value to produce a period estimate that is not a strict multiple of the sampling rate. This is realized in the following equation:

$$pd' = pd - \left(\frac{d_{pd+1} - d_{pd-1}}{2\left(d_{pd-1} + d_{pd+1} - 2d_{pd}\right)} \right) \quad (8)$$

where pd represents the current period estimate with cumulative mean-normalized difference function value d_{pd} and d_{pd-1} and d_{pd+1} represent the cumulative mean-normalized difference functions immediately before and after d_{pd}.

2.3.8. Best Local Estimate

The next error correction step is the Best Local Estimate which looks at the previous estimates from within the timeframe of the current estimate's period. This step sets the current estimate to the estimate with the smallest difference function value in the specified range. This is akin to asking what estimate is the most confident within the period range of the current estimate. The effect is to provide some pitch stability or avoid a fine or gross error.

YIN implements the Best Local Estimate step differently from YinRT as the latter is constrained to only search previous estimates given the no look-aheads assumption.

One useful caveat in the implementation of the Best Local Estimate step is to only search the original cumulative mean normalized difference values, rather than ones which have already been modified by the Best Local Estimate. Otherwise, a very confident estimate can propagate forward until a new highly confident estimate is found.

2.3.9. Octave Error Correction

The octave error correction step is new to YinRT and is used to prevent a frequency estimation from immediately halving from its previous estimate. Since speech does not generally halve frequency in the space of one sample, this can be interpreted as an error by the algorithm and the previous estimate can be repeated.

The necessity of this step became apparent during parameter tuning experiments when limiting the min and max frequency was attempted. The goal was to limit the search range and possibly provide more accuracy with less latency. On the contrary, there was significant octave halving errors which had not existed before. Upon further reflection, these errors were attributed to the fact that by lowering the max lag, the absolute threshold had a shorter search range, resulting in a smaller percentage of upper bounds being reduced; meaning higher order dips were searched and selected.

One possible response to this situation would be to adjust the absolute threshold or its search bounding factor, but given this demonstration of interdependence among the various signal processing steps, it was concluded that there were probably other parameter combinations which could result in a similar errors. Therefore an error correction step would provide more value.

2.3.10. Window Loop End

The final step in YinRT is to join the local results with some global data structure and release any local variables which are no longer needed. Then it returns control to the top of the window loop and the next processing window is passed down.

3. Results

3.1. Parameter Tuning

Parameter tuning for YinRT was conducted as specified in section 2.2.3. Figure 4 illustrates the results of this process. Thousands of parameter combinations were tested and only those with latencies sufficient to meet the specified criteria are plotted below. Selection of the best parameter combination was performed by finding the combination that produced the most accurate results and then as a tie-breaker, to choose the combination with the smallest latency. The final parameters are displayed below in table 1.

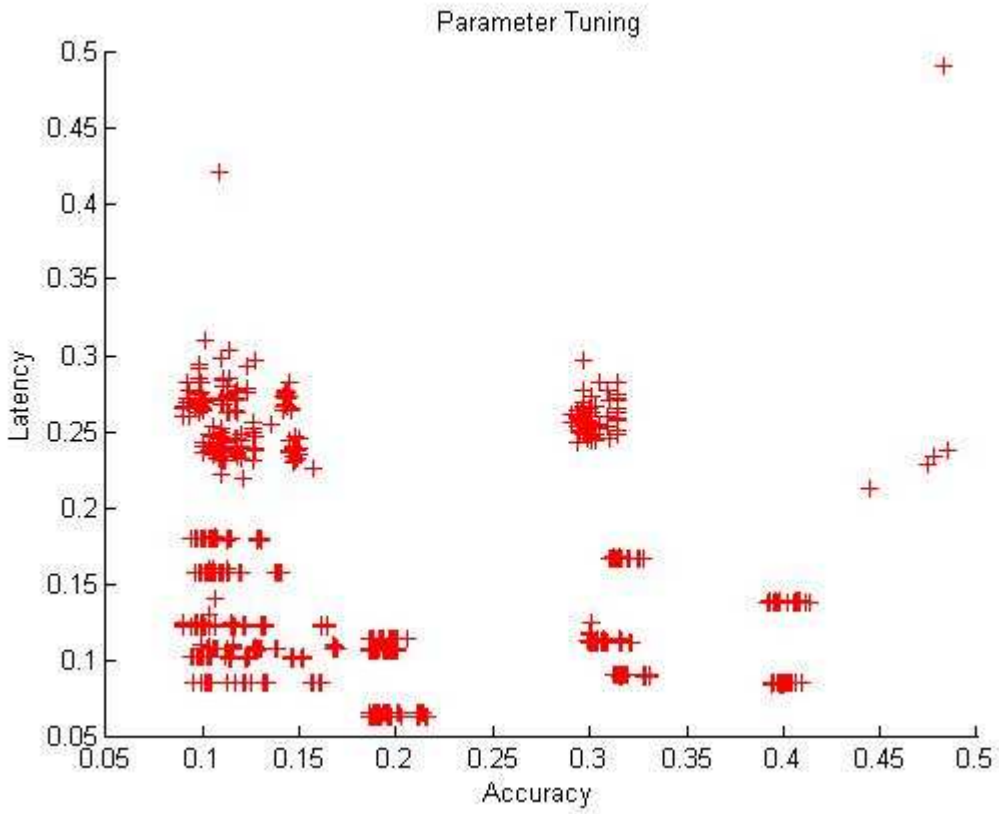

Figure 4: Scatter plot of parameter tuning combinations for male & female recordings. Parameter combinations below real-time latency threshold (where processing time > 70% file length) are not pictured.

Table 1: Final tuned parameters for YinRT.

Parameter	Value
Min F0	60
Max F0	400
Low pass Filter Threshold	2000
Difference Function Threshold	1.2
Unvoiced Threshold	0.4
Window Overlap	70%
Downsample Factor	2 (10 kHz)
Lag Scaling	2

3.2 **Accuracy & Latency Results**

The results from accuracy and performance testing the test set are displayed in tables 2-6. Data was collected for each algorithm on each file in the test set and then aggregated in total (table 2), by database (tables 3-4), and by genders (tables 5-6). Latencies could only be collected for WavePitch and YinRT since YIN is not a real-time algorithm. The columns Gross, Fine, Type 1, and Type 2 correspond to the total count of these respective errors divided by the total number of evaluations. The Proc Time Pct column shows the total processing time divided by the length of the test files. The results below are rounded to the nearest hundredth. The 'total' column represents the summation of all error rates (voiced and unvoiced) and is the primary value used in comparing the algorithms' overall accuracies.

Table 2: All Databases

	Voiced		Unvoiced		Total	
	Gross	Fine	Type 1	Type 2		Proc Time Pct
YIN	0.00	0.00	0.00	0.17	0.18	N/A
YinRT	0.01	0.01	0.03	0.05	0.10	0.16
WavePitch	0.00	0.00	0.00	0.29	0.30	0.06

Table 3: All Genders, FDA Database

	Voiced		Unvoiced		Total	
	Gross	Fine	Type 1	Type 2		Proc Time Pct
YIN	0.00	0.00	0.00	0.22	0.22	N/A
YinRT	0.01	0.01	0.02	0.06	0.10	0.15
WavePitch	0.00	0.00	0.00	0.34	0.35	0.06

Table 4: All Genders, Keele Database

	Voiced		Unvoiced		Total	
	Gross	Fine	Type 1	Type 2		Proc Time Pct
YIN	0.00	0.00	0.01	0.13	0.14	N/A
YinRT	0.01	0.01	0.04	0.03	0.09	0.16
WavePitch	0.00	0.00	0.00	0.25	0.25	0.07

Table 5: Male Samples, All Databases

	Voiced		Unvoiced		Total	
	Gross	Fine	Type 1	Type 2		Proc Time Pct
YIN	0.00	0.00	0.00	0.22	0.22	N/A
YinRT	0.01	0.01	0.03	0.06	0.10	0.15
WavePitch	0.00	0.00	0.00	0.38	0.38	0.06

Table 6: Female Samples, All Databases

	Voiced		Unvoiced		Total	
	Gross	Fine	Type 1	Type 2		Proc Time Pct
YIN	0.00	0.00	0.01	0.13	0.14	N/A
YinRT	0.01	0.01	0.03	0.04	0.09	0.16
WavePitch	0.00	0.00	0.00	0.21	0.22	0.07

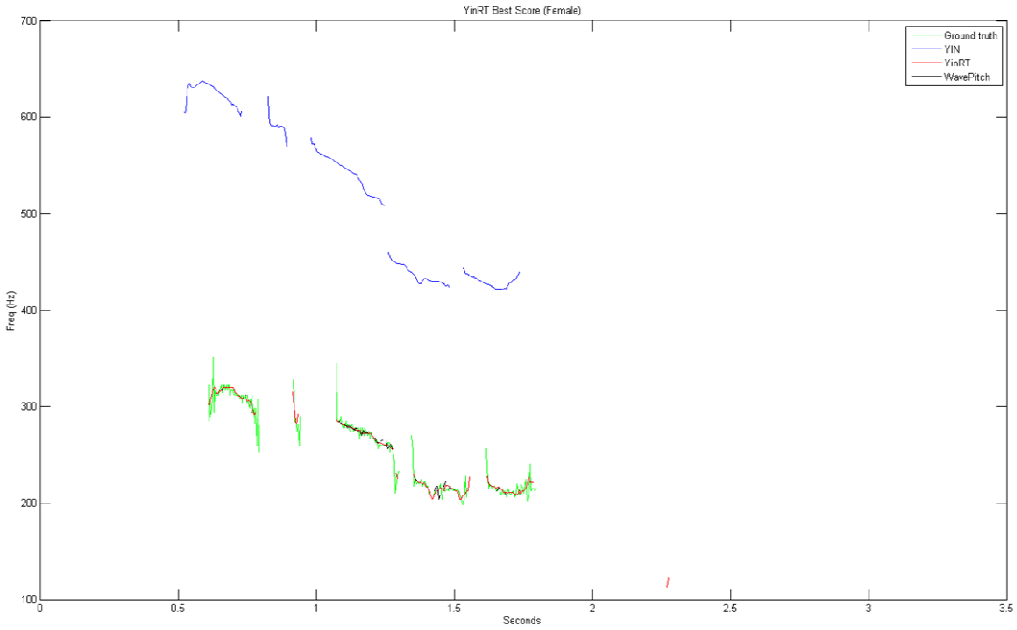

Figure 5: Example of all algorithms compared with the ground truth frequencies. This corresponds to a female speaker saying “Here’s the forwarding address”. Note the YIN (blue) follows the contour accurately, but appears to be doubling the frequencies. WavePitch (black) makes a large number of type 2 unvoiced errors where it decides the window is not voiced when it actually is, although outside of these errors, it seems to be performing quite accurately. YinRT (red) performs the best of all these algorithms.

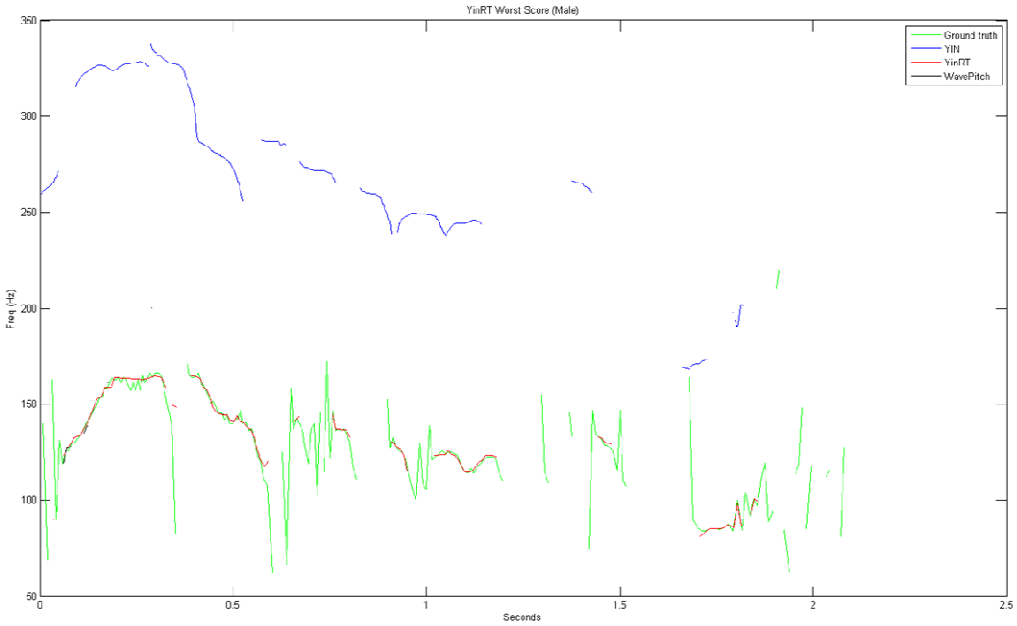

Figure 6: Example of all algorithms compared with the ground truth frequencies. This corresponds to a male speaker saying "Will I have any difficulty with the customers". Note the jagged appearance of the ground truth frequencies (green) which suggest this sample's difficulty for the algorithms. YIN (blue) follows the contour accurately, but again appears to be doubling the frequencies. WavePitch (black) makes almost all type 2 unvoiced errors where it decides the window is not voiced when it actually is. YinRT (red) performs the best of all these algorithms, but even then is missing many whole segments of voiced content.

4. Discussion

The most salient result from each comparison is that YinRT had the smallest cumulative error rate. With all the new constraints placed on YinRT, it is particularly interesting that it reports a greater accuracy than its unconstrained predecessor, Yin. But there are a variety of considerations which should be noted in making sense of this data.

These results invite the scrutiny that a direct comparison was not performed, hence YinRT's better accuracy than Yin. There are two arguments which support this claim. The first suggests that by using the author-recommended parameters rather than tuning Yin on the data as with YinRT, an unfair advantage might have been given to YinRT. The second argument would be that changing the hop parameter for Yin to match YinRT, the rest of the parameters should have been retuned accordingly.

Both of these arguments have some validity, but only with the acknowledgement of the following considerations. The default parameters for Yin were based on its performance against 5 databases, including the Keele and FDA corpuses. Thus to some degree, Yin already had been trained on this same data. However, these samples were not downsampled and the ground-truth used in its evaluation was not the database-supplied F0 files, but rather the raw-laryngograph data as processed by Yin. Additionally, the hop amount may have changed and unbalanced some of the other parameters due to some hidden dependencies in the algorithm, but one would expect this parameter to be the most decoupled to other parameters in the algorithm. In fact, the original Yin paper does report gross error percentages using the same ground-truth data as with YinRT for each database, and the results observed here for Yin are improvements to the gross error percentages reported by the author.

Compared to the other algorithms, the significant advantage of YinRT appears to be its minimization of Type II Unvoiced Errors. For all the algorithms, the greatest source of error was the type II unvoiced error rate. In fact, Yin reports uniformly better accuracy than YinRT for both voiced error types as well as type I unvoiced errors. It is only when Yin's type II unvoiced error is factored in that YinRT overtakes it for the

greatest accuracy. The difference between type I and II unvoiced errors for YinRT was uniformly smaller than either Yin or WavePitch. Type II unvoiced errors represents uncertainty: a signal is sufficiently complex that whatever F0 is estimated has a low confidence associated with it, and so the algorithm decides it must be that the current window is unvoiced. This again begs the question, why YinRT was able to better recognize the true F0 in these situations that its counterparts. One answer is that with its reduced estimation resolution, provided by the lag scaling parameter, more noise was able to be factored out than useful signal. There may be some underlying relationship between sampling rate and the lag rate used to estimate the fundamental frequency.

The latencies reported by Yin are sufficiently fast for real-time processing, involving a processing time about 15% the length of the file, but the latencies reported by WavePitch were more than twice as fast at about 6%. WavePitch is very quick indeed, but this contrasts with it performing uniformly worst in accuracy. Almost all of the error in WavePitch is occurring as type II unvoiced errors, representing a lack of capability for WavePitch to recognize a complex signal.

5. Conclusions and Future Work

The field of pitch detection algorithms remains fragmented as new methods and results generally provide domain-specific improvements. A new algorithm has been presented which attempts to optimize latency (i.e. for real-time applications) and accuracy in the domain of disconnected speech signals. The fundamental strategy was to degrade the signal and estimation resolutions to provide suitably fast performance,

and then to apply various error correction steps to minimize the impact these decreased resolutions. As acknowledged by its name, YinRT, the new method extended and altered the well-known YIN algorithm. YinRT produced improvements in accuracy and processing-time compared to similar domain-oriented methods. These results demonstrate the potential of YinRT as real-time PDA for disconnected speech.

Topics requiring further investigation regarding YinRT include testing on a broader range of speakers and finding ways of addressing noise in the signal. Possible areas for improvement include reducing the number and interdependence of parameters, and finding more efficient means of processing and representing difference function data. In particular, perhaps a new method could be found to exploit the periodicity of the difference function, resulting in a smaller memory footprint and search range. Another possible improvement may be found in using the magnitude of difference function similar to AMDF [8] instead of calculating the squared error. The reduction in multiplication operations may reduce the latency without affecting accuracy.

It should also be noted that none of the test auditory databases supplied samples of dysarthric speech and therefore YinRT's performance in such a setting would be difficult to predict. No such samples were used primarily because there were no freely available databases containing dysarthric speech and their corresponding fundamental frequencies. Evaluating any PDA's performance absent some ground-truth frequency data would be problematic. However, some type of evaluation of YinRT's performance processing dysarthric speech would be an important next step.

6. References

[1] De Cheveigné / Ircam-CNRS, Alain, and Hideki Kawahara / Wakayama University. "YIN, a Fundamental Frequency Estimator for Speech and Music." *Journal of the Acoustic Society of America* 111.4 (2002): 1917-930.

[2] "Evaluating Pitch Determination Algorithms." *Fundamental Frequency Determination Algorithm (FDA) Evaluation Database*. Centre for Speech Technology Research at the University of Edinburgh, 1994. Web. 21 Nov. 2010. <http://www.cstr.ed.ac.uk/research/projects/fda/>.

[3] Rabiner, Lawerence R., Michael J. Cheng, Aaron E. Rosenberg, and Carol A. McGonegal. "A Comparative Performance Study of Several Pitch Detection Algorithms." *IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING* 24.5 (1976): 399-418.

[4] Press, William H. "13.2 Correlation and Autocorrelation Using the FFT." *Numerical Recipes in C: the Art of Scientific Computing*. Cambridge: Cambridge UP, 1992.

[5] Larson, Eric, and Ross Maddox. "Real-Time Time-Domain Pitch Tracking Using Wavelets." Thesis. Departments of Mathematics, Physics, and Philosophy, Kalamazoo College, 2005.

[6] Gerhard, David. *Pitch Extraction and Fundamental Frequency: History and Current Techniques*. Regina: Dept. of Computer Science, University of Regina, 2003.

[7] Lawrence R. Rabiner, "On the Use of Autocorrelation Analysis for Pitch Detection" *IEEE* Trans. Acoust, Speech, Signal Processing, VOL. ASSP-25, NO. 1, 1977**.**

[8] M. J. Ross, H. L. Shaffer, A. Cohen, R. Freudberg, and H.J.Manley, “Average magnitude difference function pitch extractor,” *IEEE Trans. Acoust., Speech, Signal Processing,*vol. ASSP-22, pp. 353-362, Oct. 1974.

[9] Meyer, G. F. "Keele Pitch Database - SCHOOL OF PSYCHOLOGY - University of Liverpool." *The University of Liverpool*. Web. 27 Mar. 2011. http://www.liv.ac.uk/psychology/hmp/projects/pitch.html

[10] Ross, M., H. Shaffer, A. Cohen, R. Freudberg, and H. Manley. "Average Magnitude Difference Function Pitch Extractor." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 22.5 (1974): 353-62. Print.

[11] Rabiner, Lawrence R., and Ronald W. Schafer. *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall, 1978. Print.

[12] Noll, A. Michael. "Cepstrum Pitch Determination." *The Journal of the Acoustical Society of America* 41.2 (1967): 293. Print.

[13] Bagshaw, P. C., Hiller, S. M., and Jack, M. A. (1993). ‘‘Enhanced pitch tracking and the processing of F0 contours for computer and intonation teaching,’’ *Proc. European Conf. on Speech Comm.* (Eurospeech), pp. 1003–1006.

[14] Barnard, E., Cole, R. A., Vea, M. P., and Alleva, F. A. (1991). ‘‘Pitch detection with a neural-net classifier,’’ *IEEE Trans. Signal Process*. 39, 298–307.

[15] Brown, J. C., and Puckette, M. S. (1989). ‘‘Calculation of a ‘narrowed’ autocorrelation function,’’ J. *Acoust. Soc. Am*. 85, 1595–1601.

[16] Brown, J. C., and Zhang, B. (1991). ‘‘Musical frequency tracking using the methods of conventional and ‘narrowed’ autocorrelation,’’ *J. Acoust. Soc. Am.* 89, 2346–2354.

[17] Duifhuis, H., Willems, L. F., and Sluyter, R. J. (1982). ‘‘Measurement of pitch in speech: an implementation of Goldstein’s theory of pitch perception,’’ *J. Acoust. Soc. Am.* 71, 1568–1580.

[18] Hedelin, P., and Huber, D. (1990). ‘‘Pitch period determination of aperiodic speech signals,’’ *Proc. ICASSP*, pp. 361–364.

[19] Hess, W. J. (1992). ‘‘Pitch and voicing determination,’’ in Advances in Speech Signal Processing, edited by S. Furui and M. M. Sohndi , pp. 3–48.

[20] Eric Scheirer and Malcolm Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *International Conference on Acoustics, Speech and Signal Processing, volume II,* pages 1331–1334. IEEE, 1997.

[21] Curtis Roads. The Computer Music Tutorial. MIT Press, Cambridge, 1996.

[22] Lopresti, P., and H. Suri. "A Fast Algorithm for the Estimation of Autocorrelation Functions." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 22.6 (1974): 449-53. Print.

[23] Barnwell, T. "Recursive Windowing for Generating Autocorrelation Coefficients for LPC Analysis." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29.5 (1981): 1062-066. Print.

[24] Carayannis, G. "An Alternative Formulation for the Recursive Solution of the Covariance and Autocorrelation Equations." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 25.6 (1977): 574-77. Print.

[25] M. J. Cheng, “A comparative performance study of several pitch detection algorithms,” M. S. thesis, Mass. Inst. Technol., Cambridge, June 1975.

[26] N. J. Miller, “Pitch detection by data reduction,” *IEEE Trans. Acoust., Speech, Signal Processing (Special Issue on IEEE Symposium on Speech Recognition)*, vol. ASSP-23, pp. 72-79, Feb. 1975.

[26] J. D. Markel, “The SIFT algorithm for fundamental frequency estimation,” *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp.367-377, Dec. 1972

[27] "Parkinson's and the Speech and Language Therapist." *Physiotherapy* 84.8 (1998): 384. Print.

[28] Penner, Nicholas Miller, Ingo Hertr, Heike. "Dysprosody in Parkinson's Disease: an Investigation of Intonation Patterns." *Clinical Linguistics & Phonetics* 15.7 (2001): 551-66. Print.

[29] Slavit, D., and B. Leader. "Functional Voice Disorders/muscle Tension Dysphonia." *Otolaryngology - Head and Neck Surgery* 112.5 (1995): P100. Print.

[30] Ballard, Kirrie, and Robin, Donald. “A Treatment for Dysprosody in Childhood Apraxia of Speech”. *Journal of Speech, Language, and Hearing Research* Vol.53 1227-1245 October 2010.