Pitch Detection Using YIN Autocorrelation Method

Todd Nguyen
University of Illinois at Urbana Champaign, ECE 397
May 5, 2016

1 Motivation

This goal of this project is to find a fast and robust method for pitch detection to replace the Two-Way Mismatch method of pitch detection currently implemented in the SNDAN Audio Processing suite. I chose to do an autocorrelative method because of its high accuracy without smoothing. Autocorrelation methods also have relatively good accuracy with pitch detection, however the autocorrelation method has relatively low accuracy for voiced/unvoiced pitches.[3]. The YIN method [1] is based on the autocorrelation method with six add on features in order to improve accuracy. According to the paper by de Cheveigné, the YIN method's six add-ons improve the error rate from 17% to 0.5%. In the next section, I will explain the YIN method's improvements over the base autocorrelation method.

2 Method

This section will describe the YIN method, taking some information and formulas from the paper as written by de Cheveigné.[1]

The first step of the YIN method is the autocorrelation function, and this is defined as:

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

where τ is the lag value (1 unit represents one sampling interval), t is the time index, and W is the integration window size. Another version commonly used is:

$$r'_t(\tau) = \sum_{j=t+1}^{t+W} x_j x_{j+\tau} \tag{2}$$

This version of the equation shrinks the autocorrelation as the lag τ increases, which shortens the number of computations as well as tapers the autocorrelation function. For my use in this project, most of the wavelengths will not be very long relative to the window size of the function, the taper effect will not adversely affect most frequencies. In this project, I have chosen a window length of 6ms, which means the lowest frequency I can detect is 166 Hz. This window length can be changed easily through a define macro in my program, and a longer frame size will mean that we can detect a longer wavelength. However, the normal autocorrelation function is strongly subject to "octave" errors, or errors in which upper harmonics with large magnitude change the output of the autocorrelation function.

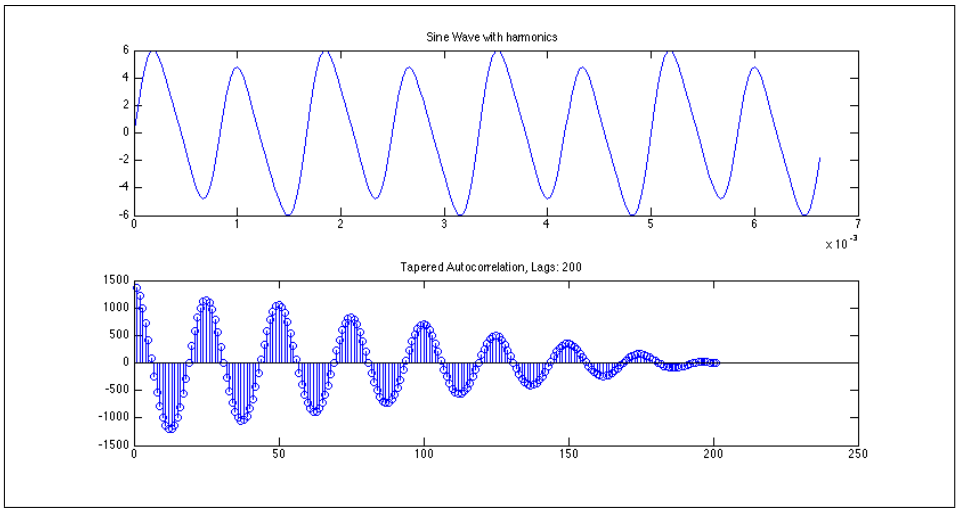

Figure 1: Example of an octave error. Wave and Autocorrelation Function using equation (2). Note that the peak at lag 50 actually corresponds to the fundamental frequency, while the peak at lag 25 is the highest peak.

In order to prevent octave errors, the YIN method replaces autocorrelation function with the difference function. This is the second step of the YIN method. The difference function is defined as follows:

$$d_t(\tau) = \sum_{j=1}^{W}(x_j - x_{j+\tau})^2 \tag{3}$$

Now instead of looking for peaks, we will be looking for zero values, which will give us the wavelength for the fundamental frequency. The difference equation can also be generated from the autocorrelation function in the equation below (4), however in my program I skip the autocorrelation step and instead just calculate the difference function using equation (3).

$$d_t(\tau) = r_t(0) + r_{t+\tau}(0) - 2r_t(\tau) \tag{4}$$

The figure below shows the difference function in comparison to the autocorrelation function.

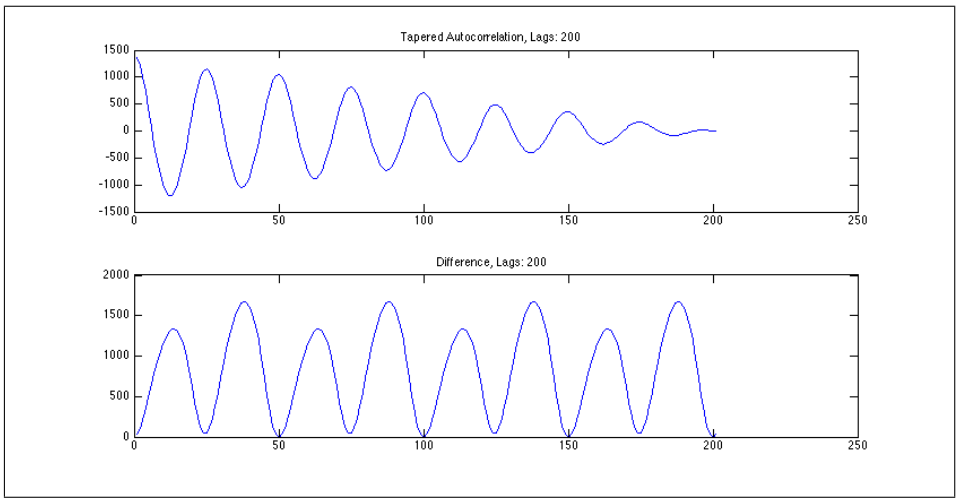

Figure 2: Autocorrelation Function above and difference function below. Note that the difference function only goes to 0 at lag 50 and not at lag 25, where we would have previously mistook as the fundamental frequency using the Autocorrelation function only.

However, this difference function fails if the recorded save is not perfectly periodic. Likewise, very strong upper harmonics may also cause secondary dips in the function as well. Steps 3 and 4 of the YIN method are used to combat this. In step 3, we will instead use a "cumulative mean normalized difference function" to replace the existing difference function. This is shown below:

$$d'_t(\tau) = \begin{cases} 1 & \text{if } \tau = 0, \\ \frac{d_t(\tau)}{(1/\tau)\sum_{j=1}^{\tau} d_t(j)} & \text{otherwise} \end{cases} \tag{5}$$

The cumulative mean normalized difference function divides each value of the difference function by the average of the previous values, which adds multiple benefits in addition to preparing the function for step 4. These benefits include having the difference function start at 1 rather than at 0, so searching through the beginning of the difference function is easier, which theoretically removes upper limit bound for wavelength.

After we have normalized the difference function, the YIN method calls for step 4, setting an absolute threshold. Since it is unlikely that we will have exact periodicity within the frame, we will instead look for the first local minima value that passes below the threshold. The YIN paper [1] states that they used a value of 0.1, and I used a value of 0.1 in my implementation as well.

I did not implement step 5 of the YIN pitch detection method. Step 5 is parabolic interpolation, which interpolates the result of the results of the difference function in order to further improve pitch accuracy. However, in the usage of this pitch detector program, we are not looking at fine pitch, so I chose to omit this step in order to save computation time.

Step 6 of the YIN method is Best Local Estimate, which looks at small windows, then looks at a much larger window around the original frame to further refine the pitch accuracy. In the paper it was stated that this provided marginal improvement in accuracy (from 0.77% to 0.5%), though this would have increased computation time significantly due to almost double the amount of autocorrelations, as well as autocorrelations on a larger window size. Because of the negligible accuracy increase and the longer computation time, I chose to omit this step as well.

3 Results

First, here is the output of the current SNDAN Two-Way Mismatch pitch detection. Note that between times 2s and 8s, there are many incorrect pitch readings in the upper octaves.

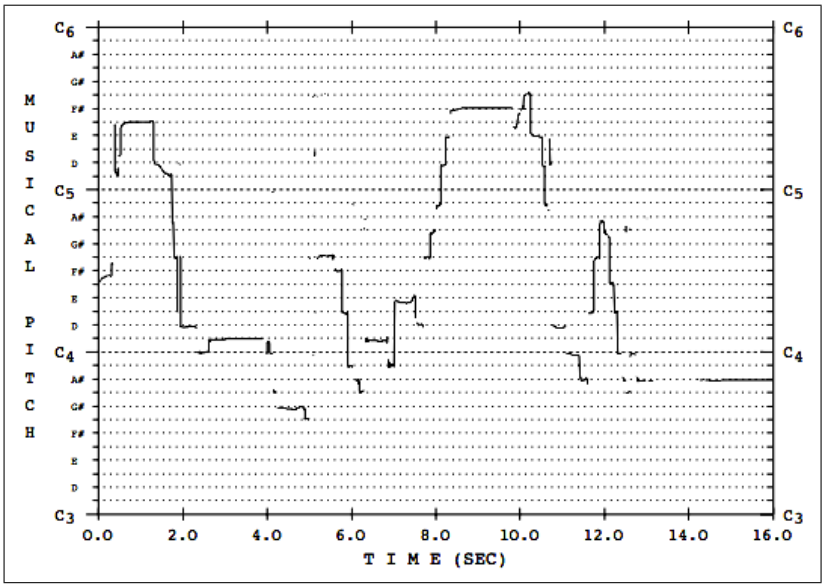

Figure 3: Pitch output of the leonard.16 sound file on the Two-way Mismatch pitch detector.

Now, the output of my YIN pitch detector at frame size 6ms. Notice that in this, there are very few connecting lines, between the notes - this is because on pitch changes the YIN method would often not often read a note - raising the threshold for detected notes would probably fix this, though it would lead to increased octave errors.

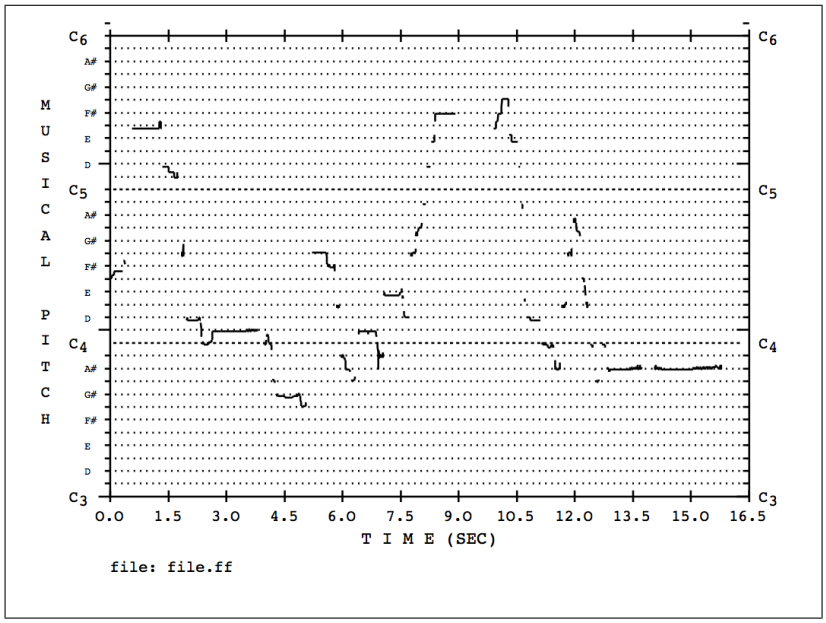

Figure 4: Pitch output of the leonard.16 sound file on the modified YIN pitch detector at 6ms frame size

As you can from the above drawings, the YIN pitch detection method is much more accurate than the Two-Way Mismatch method, especially with unvoiced notes and silence. One way in which the YIN method falls short, however, is at note transitions.

One way to help fix this disconnection is to increase the frame size - I have increased the autocorrelation's frame size to 30ms. This gives significant improvement, though it does increase computation time, as each frame now has to process many more samples. Though there are less frames to process, frame processing time runs at $O(n^2)$ time.

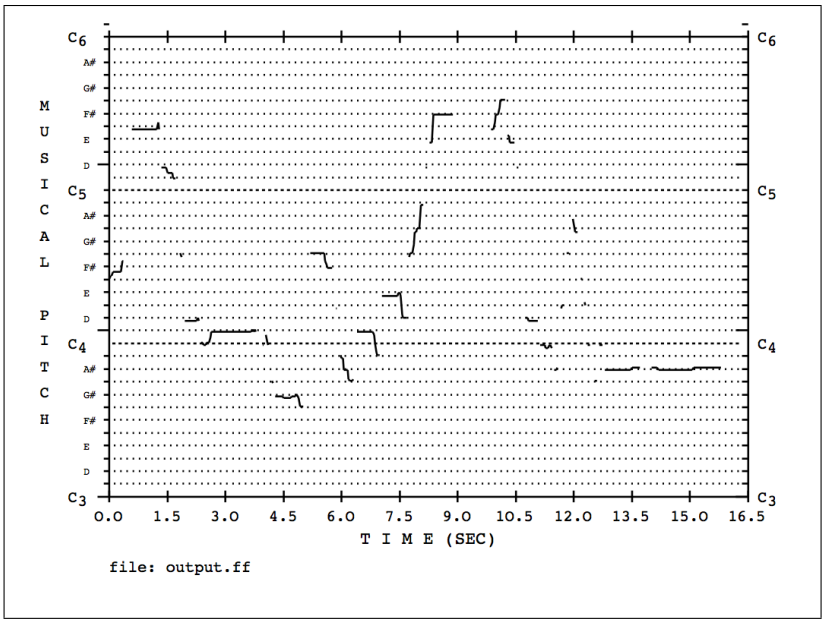

Figure 5: Pitch output of the leonard.16 sound file on the modified YIN pitch detector at 30ms frame size

However, pitches that have very fast time-varying pitches (i.e. recordings with a lot of vibrato) will cause the pitch detector to fail at large window sizes, because the difference function will not be able to fall below the threshold value. Here is a tenor vocal recording with a lot of vibrato, analyzed with a 30ms frame size.

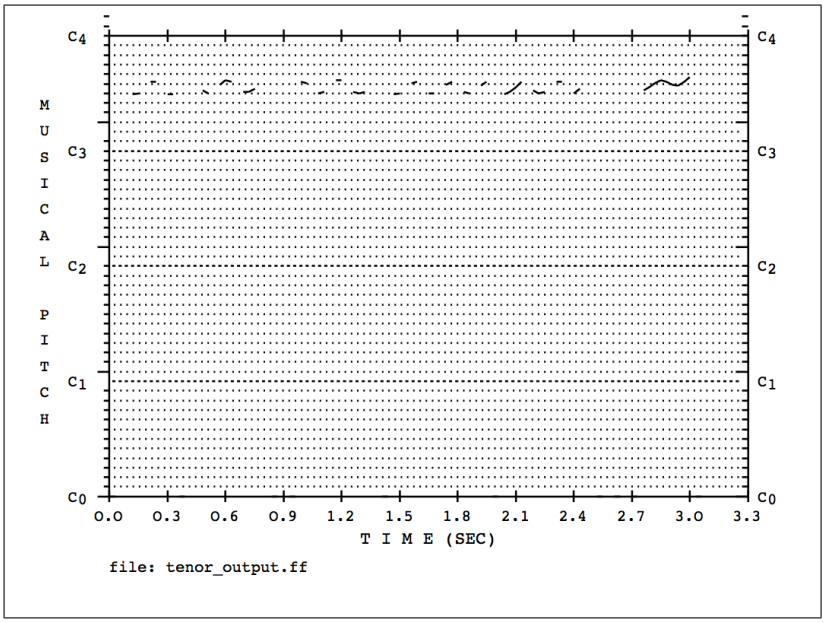

Figure 6: Pitch output of the tenor sound file on the modified YIN pitch detector at 30ms frame size

From the pitch detection otuput it would appear the voice is singing different notes. However, when we reduce the frame size to 15ms, we can see that this is not the case.

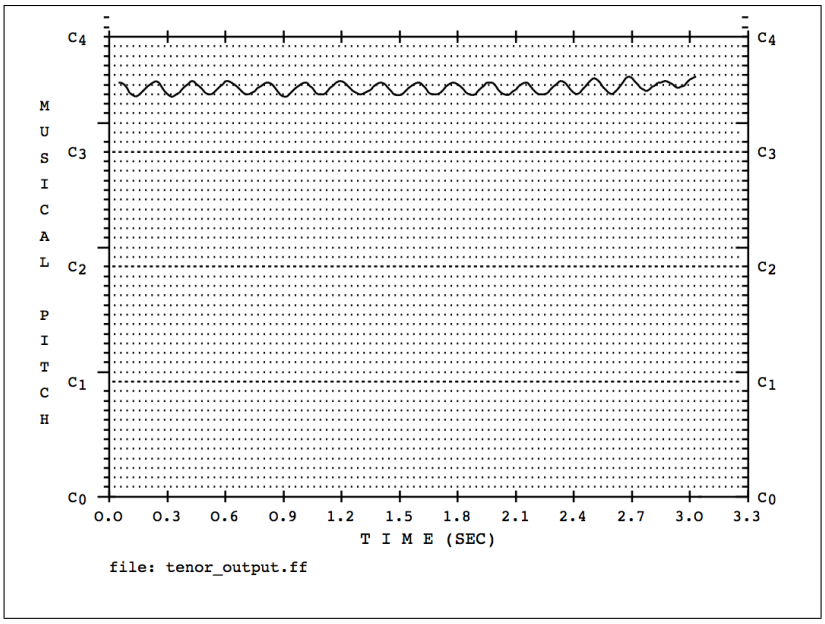

Figure 7: Pitch output of the tenor sound file on the modified YIN pitch detector at 15ms frame size

I have set the default frame size in my program to 15ms, however it can be changed by editing the define macro FRAMESIZE in the program. Having a shorter frame size results in greater precision and shorter computation time, though often fails at note transitions. Longer frame size worked better on recordings like leonard.16 with many notes, though failed on recordings with a vibrato, and had much longer computation time. Overall, the YIN pitch detection method shows much greater accuracy over the Two-Way Mismatch method, and corrects many of the shortcomings of a normal autocorrelative pitch detector, such as octave errors.

References

[1] de Cheveigné, Alan, et al. *YIN, A Fundamental Frequency Estimator For Speech and Music.* 2002

[2] Maher, Robert C, et al. *Fundamental Frequency Estimation of Musical Signals Using a Two-Way Mismatch Procedure.* 1993

[3] Rabiner, Lawrence, et al. *A Comparative Performance Study of Several Pitch Detection Algorithms* IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-24, No. 5, October 1976.