Pitch Detection for Music in Noisy Environments

Performance evaluation of BaNa, a hyrbrid approach for a Noise Resilient Pitch Detection

Myron Vasilik, Logan Stillings, and Carmen Cortazar,
Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY, USA
Email: mvasili2@u.Rochester.edu, lstillin@u.Rochester.edu, ccortaza@ur.rochester.edu

***Abstract*—A noise resilient pitch detection algorithm, BaNa, is evaluated to test its accuracy and effectiveness in achieving the lowest Gross Pitch Error (GPE) rate in noisy environments. Pitch detection in noisy environments remains at the forefront of current research like music signal analysis with emerging applications such as speech perception and recognition and music notation programs. An analysis of BaNa is provided and will mimic the results obtained in the BaNa paper. Furthermore, additional iterations will be performed changing various parameters such as, timestep and frequency bounds of the signal. Lastly, the results obtained in the experiment were compared to those presented in the BaNa paper. The results show that BaNa provides lower GPE rates than other pitch detection algorithms in music evaluation of noise resilient pitch detection. BaNa achieves the lowest GPE rate with the most consistency and is responsive to parameter changes.**

Keywords—fundamental frequency; cepstrum; pitch detection algorithm; signal-to-noise ratio; gross pitch error rate.

I. INTRODUCTION

The idea for the project stems from research done by a former PhD student in the Wireless Communications and Networking Group (WCNG) in the Electrical and Computer Engineering department. Pitch detection is an important facet of signal processing, and although it provides many advantages there are several challenges still faced in research today. One of those challenges is pitch detection is noisy environments. The BaNa algorithm is specifically designed as a noise resilient pitch detection algorithm; therefore will be modified and tested for use in this project.

Fundamental Frequency (Fo) in itself is a well-researched topic in audio signal processing; however, F0 in noisy environments, introduced by audible background noise and the recording devices being used, still remains an issue for most audio applications. Fundamental frequency is an objective estimation of pitch and when evaluating the pitch of an audio signal, whether it is music or voice there are key features that can be extracted from these signals. Pitch is only defined for periodic, quasi harmonic sounds with period 1/Fo [5]. Unlike the Fo, the pitch of a signal is extremely subjective and is the relative highness or lowness of a tone as it is perceived by the ear [6].

In this project, the researchers propose an evaluation of the BaNa algorithm by first mimicking the results of the paper, "BaNa: A Noise Resilient Fundamental Frequency Detection Algorithm for Speech and Music". Next, three sets of sound samples are used: (1) prerecorded samples from BaNa project archives (2) sound samples recorded from the same instruments used in [1] (violin) and (3) sounds samples from new instruments (marimba and guitar). The goal is to evaluate BaNa's ability to accurately detect the Fo of a music signal in noisy environments. For this project, 'effectiveness' is defined by the lowest GPE rate.

A. Experimental Method

All recordings of music signals took place in a studio at the Eastman School of Music and included a guitar, violin and a marimba. Any other audio files used in the project were used from the BaNa project archives which can be found on the WCNG website [4]. This includes all noisy files used as well. In addition, all processing and analyzing of the music signals was done using Matlab.

(1) Sound samples

- Instruments used in BaNa project: pre-recorded sound samples from BaNa project archives
- Instruments not used in BaNa project: record sound samples using of the guitar and marimba.

(2) Algorithm

The following files in the original BaNa project were modified to fit our project:

- Bana_music.m – This file implements the BaNa music algorithm for Fo detection in music. One of the parameter is a music marker.
- Bana_music_auto.m – This files was implemented and used for most of the simulations in the project. This file was set up with default parameters set for the user. This allowed us to test its effectiveness by changing other parameters. This particular code was not used in the BaNa paper
- add_noise.m – This files adds each one of the 8 types of noises with different levels of SNR (0 dB to 20 dB) to clean music files.
- Evaluate_music.m – This file reads noisy music files and applies Fo detection algorithms BaNa, YIN, HPS, Praat and Cepstrum to the files.

In addition to the aforementioned codes there were additional associated functions and protocols from the BaNa project that were used.

(3) Experimental Setup

- Run simulations in [1] and [2].
    - Functions used are the ones mentioned in Part I-A-2.
- BaNa uses the lowest GPE rate in determining its accuracy.
- For this experiment, we will implement an evaluation technique based on variations in the timestep and the frequency bounds of the music signal.
- Results are compared against several well-known, sophisticated algorithms (i.e. YIN, PRATT, HPS and Cepstrum).

II. The BaNa Algorithm

A. *The Bana Process*

The BaNa algorithm, first developed for speech analysis [2], was expanded upon and modified for music analysis [1].

The following is the process of the BaNa algorithm.

- Preprocessing to include a bandpass filter prior to extraction of the pitch values. This allowed for the frequency bounds to be determined.
- Determination of the pitch candidates: This process was done in two steps. The first was a search for the harmonic peaks in the signal (frames) being evaluated using a Hanning window. The next step was a calculation of the pitch candidates. Included in the pitch candidates is the peak with the smallest frequency. Here the cepstrum method was also used to find a pitch candidate in order to justify the ones chosen by the harmonic ratio analysis.
- Selection of the pitch from the candidates chosen in the previous step. A cost function was developed which used the pitch differences between adjacent frames [2]. The Viterbi algorithm was then used in order to find the minimum cost for the optimal path.
- Modified BaNa for music signals in noisy environments: for music Fo detection the peaks with the *highest* amplitudes in the perspective frequency range were chosen.

B. *Comparitive Analysis to other Algorithms*

In both [1] and [2] the researchers provided an effective evaluation of the BaNa algorithm in comparison to some of the most sophisticated algorithms to date (Cepstrum, HPS, YIN, Praat). BaNa and the other algorithms were evaluated by rating their performance under 8 different types of noise and ranging SNR values. According to [1] and [2] the algorithm was able to obtain the lowest GPE rate consistently. Furthermore, to test the noise resiliency of the pitch detection algorithm, the researcher mixed 8 types of noises into the original signal with varying SNR ranges. Lastly, in [1] and [2] the BaNa algorithm is tested by averaging all of the pitch detection ratios calculated using each of the algorithms presented earlier. These are labeled as ground truth samples. All calculations performed were done as a function of SNR. These results show that BaNa indeed achieves the lowest GPE rate between all of the algorithms tested. Evaluation of BaNa Fo detection for music.

III. Project Experiment

A continuation of the work performed in the WCNG lab, the goal of this project was to evaluate the accuracy of BaNa in detecting the Fo of a music signal in noisy environments. Unlike in [1] and [2] instead of utilizing the synthetic noise files, we recorded sound samples, using Protools, which contained natural noise (i.e. noise from the room, microphone, etc.). Using these sound samples our approach was to first mimic the results in [1] and [2] and then after recording the necessary sound samples (guitar, violin and marimba) we used them to evaluate the performance of BaNa.

A. *Experiment – Part A – Mimic the results in [2]*

To mimic the results in [1] the function BaNa _auto.m was used and for the results in [2] the function BaNa_music_auto.m was used.

The plot figures below show the GPE rates of the different algorithms for the Violin, Trumpet, Clarinet and Piano when the BaNa_music.m was simulated. In addition the bar graphs below show how each of the four instruments performed in different noisy environments at various SNR rates. The following figures below show the GPE rates of the Piano, Trumpet, Clarinet and the Violin for following algorithms: BaNa, HPS, YIN, Praat and Cepstrum. It can be seen that BaNa's performance exceeds that of the others in most cases by 20%.

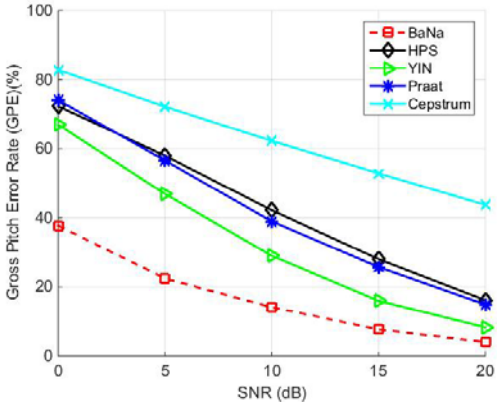

Fig. 1. GPE rates of various algorithms for Piano

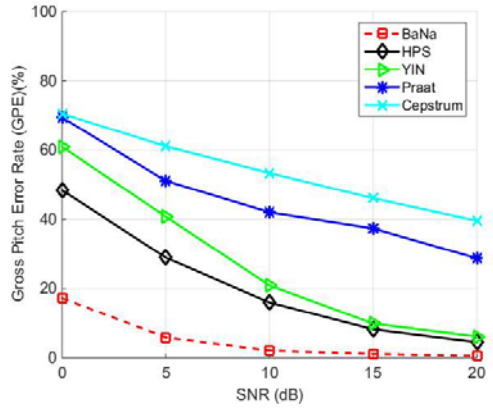

Fig. 2. GPE rates of various algorithms for Trumpet

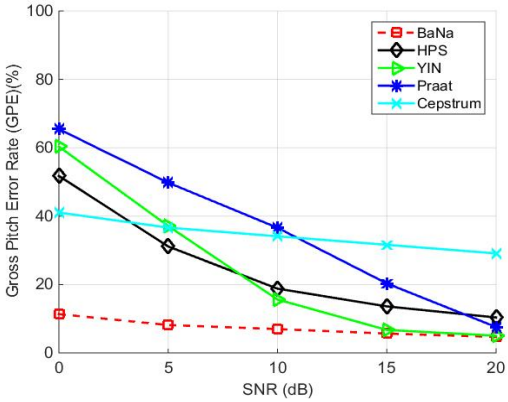

Fig. 3. GPE rates of various algorithms for Clarinet

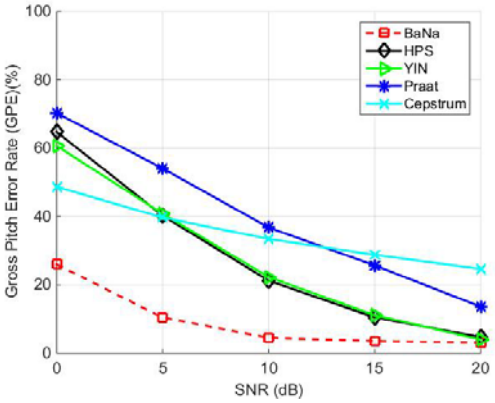

Fig. 4. GPE rates of various algorithms for Violin

The researchers were able to successfully mimic the results in [2] and obtain plots and graphs of the simulations.

B. *Experiment – Part B - Evaluation of BaNa – using sound samples recorded for this project*

The processing of the sound samples proved to be a bit more tedious then we thought initially. When we recorded the instruments and processed them in Matlab we recognized that the way in which the musical instruments were recorded produced a high amount of noise. In the following section, the approach taken to handle the excess noise in the recording samples and the results for the simulations will be discussed.

1) Evaluation of BaNa – using synthetic and recorded sound samples

Noise is simply the introduction of unwanted frequency information into a signal. The Bana_music_auto.m Fo detection algorithm is meant to find fundamental frequencies in each frame, and the final product offers a "snapshot" of the waveform. For example, running the file 'clarinet.wav' yields the results displayed below in Fig. 5.

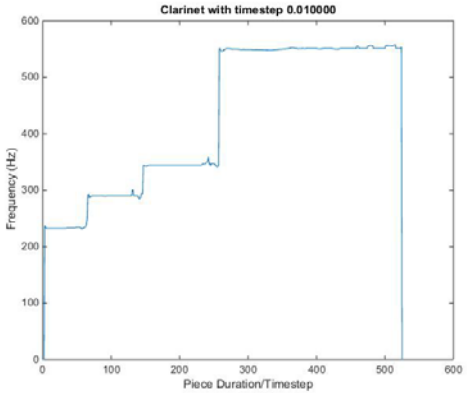

Fig. 5. For provided bounds [2]: Fomin = 220Hz, Fomax = 660Hz

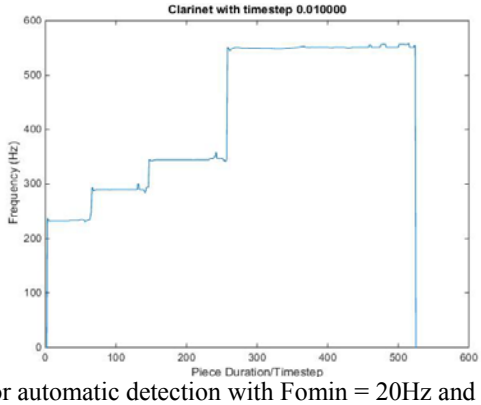

Fig. 6. For automatic detection with Fomin = 20Hz and Fomax = 4200 Hz:

The two figures above show the plots when the frequency maximum and minimum are provided and when they are detected. If one listens to the 'clarinet.wav' file, one could easily see how well-matched the frequencies are to their respective parts in the recording even with different frequency bounds. For all intents and purposes, we can consider these two results (auto-detected f0 bounds and provided f0 bounds) identical. It is assumed by the authors in [2], and should soon become apparent to the reader of this paper, that the origin of the 'clarinet.wav' recording is in fact a midi file and not a live recording of an instrument. Running other files tested by the

BaNa algorithm (i.e. trumpet.wav, piano.wav, and violin.wav) all yield similar results. However, when the BaNa algorithm is presented with a live recording of an instrument, the Fo detection does an interesting thing. Observe the results of the following live recording of a marimba:

As one can clearly see in Fig. 7, the algorithm "detects" noise that isn't meant to be considered as part of the instrument's performance. Listening to the actual content of 'marimba.wav', it does not present us with any obvious artifacts in the recording, but the algorithm picks up on high-frequency recording noise not apparent to the listener and assumes that this is part of the Fo content. Now, this would ordinarily be seen as a fault in the recording, however, in this case our noisy marimba recording is very useful.

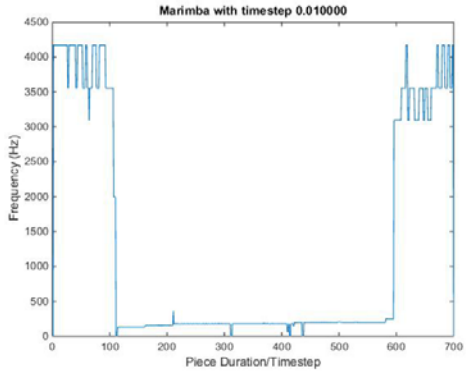

Fig. 7. Automatic settings at Fomin = 20Hz, Fomax = 4200 Hz:

Since we are researching methods to detect Fo features in a noisy environment, and noise is simply unwanted harmonic content in a given signal, the room hum and recording noise in our marimba recording become microcosms of the various types of noise we will be testing for later. In other words, if we can successfully find a way to accurately detect the Fo information of the marimba recording, we should also be able to detect the Fo information of the marimba recording with various types of noise added.

How does one achieve such a thing?

The Fo detection algorithm in BaNa defaults to values of 50Hz and 4000Hz if the Fomin and Fomax values are not specified. For our project, we chose to default to 20Hz and 4200Hz as the Fomin and Fomax, respectively, simply because these values encompass the full harmonic range of a grand piano and most instruments used in western tonal music. We have already seen the effects of using the default (Fomin = 20Hz, Fomax = 4200Hz) settings on the marimba Fo evaluation (Fig. 7). Supposing we were able to figure out the upper bound (Fomax) to improve our results:

Below, in Fig. 8, there is an obvious improvement in the Fo detection, achieved simply by removing unnecessary frequencies which, while they were only noise in reality, would have been interpreted by the algorithm as some likely Fo content. Similarly, by providing a lower (Fomin) bound, as seen in Fig. 9 below.

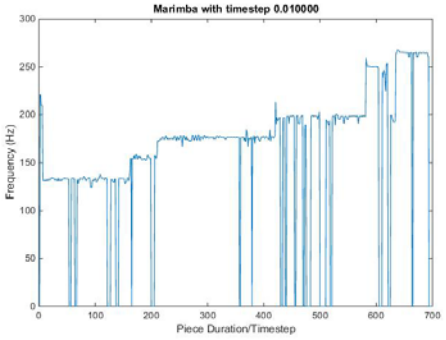

Fig. 8. Fomin = 20Hz, Fomax = 270Hz

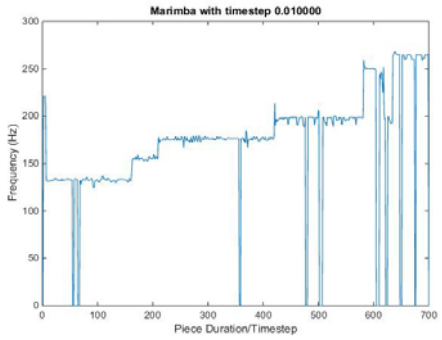

Fig. 9. Fomin = 120 Hz, Fomax = 270 Hz

A close comparison with Fig. 8 shows fewer irregularities in the waveform, providing a more accurate representation of the marimba's Fo content. It now goes without saying that providing Fo bounds for our signal will likely improve our results considerably. However, suppose a user may not be able to identify the Fomin and Fomax of their sound file? How can we implement an algorithm that would do this on its own?

For all of the above examples, the selected timestep, the time offset of the detected Fo, was .01s. This value will vary by instrument, and is likely to provide useful results just by being adjusted multiple times for the same instrument. For example, in our marimba file, using the default Fomin = 20Hz and Fomax = 4200Hz, and changing only the timestep, we achieved the results exhibited in Fig. 10.

A larger timestep allows for bigger computational frames. The result is that our signal is mostly noise. Using the same frequency bounds, Fig. 11 shows that while still having a wide and very general frequency boundary, our Fo detection is actually quite close to the input signal.

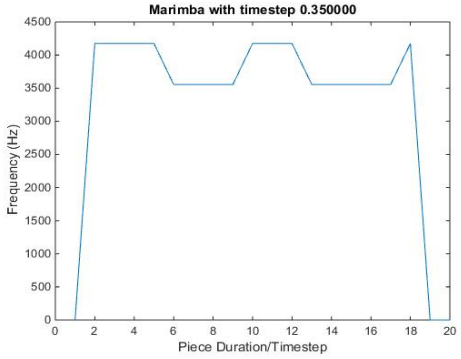

Fig. 10. f0min = 20Hz, f0max = 4200Hz, timestep = .35s

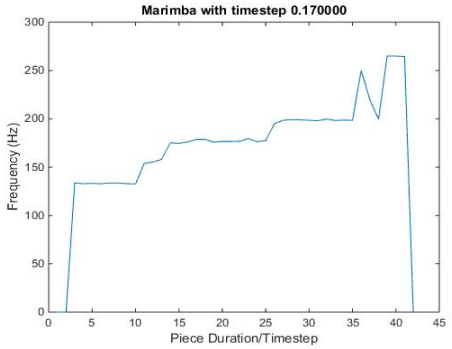

Fig. 11. Default f0 bounds, timestep = .17s

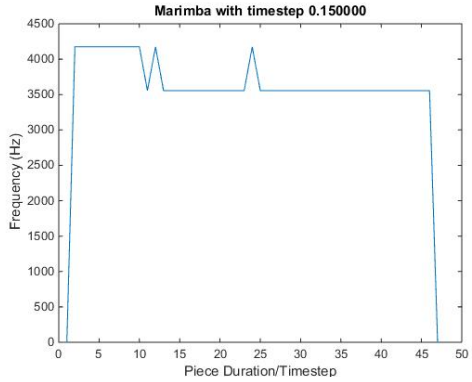

Fig. 12. Default f0 bounds, timestep = .15s

This figure shows just how unpredictable timestep can be. A difference of .02s changes the algorithm's perception of the signal entirely. However, if we were to narrow the frequency bounds down to an acceptable range, we would see this issue less and less.

Herein lies the solution to our problem. We begin by evaluating every signal at the default frequency bounds, while varying timesteps. We obtain the detected maximum and minimum Fo bounds of the signal that don't deviate too far from what we would expect (discard anomalous values). We then use these detected maximum and minimum values as our new bounds and repeat as necessary until we are confident we have an accurate representation of the signal. More noise will likely require more iterations of the algorithm.

IV. CONCLUSION

Despite adding noise, the BaNa algorithm is very accurate for both music and speech. In speech the only problem to arise was the necessary distinction between frames that are voiced and unvoiced. Without letting the code know, it is impossible for the algorithm to recognize high frequency consonants as not part of the voiced speech. The only problem to arise in music is found in the gap between notes when only the noise is audible. There also seems to be an effect in instruments with longer release times where the volume of the actual Fo decreases to the point where the algorithm likely mistakes the noise as the louder fundamental. Instruments with longer release times seem more susceptible to errors in Fo predictions using the BaNa algorithm.

REFERENCES

[1] N. Yang:, H. Ba, W. Cai, W. Heinzelman and I. Demirkol. "BaNa: A Noise Resilient Fundamental Frequency Detection Algorithm for Speech and Music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing 22.*

[2] H. Ba, N. Yang, I. Demirkol and W. Heinzelman, "BaNa: A Hybrid Approach for Noise Resilient Pitch Detection," *Proceedings of the 2012 IEEE Statistical Signal Processing Workshop (SSP '12), Aug. 2012.*

[3] J.P Bello, G. Monti, and M. Sandler, "Techniques for automatic music transcription," in *Intl Symposium on Music Information Retrieval, 2000, pp. 23-25*J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[4] University of Rochester Electrical and Computer Engineering Dpeartment, *Wireless Communications and Networking Group*, 2011, http://www.ece.rochester.edu/projects/wcng/, (March-April 2015).

[5] Z. Duan, ECE 472 Audio Signal Processing, "Pitch Analysis", Univeristy of Rochester, http://www.ece.rochester.edu/~zduan/teaching/ece472/lectures/Lecture_07.pdf

[6] N. Yang and H. Ba, "Single Pitch Detection", a guest lecture for ECE 492 Computer Audition.