 DOI: 10.1109/TASLP.2014.2352453

BaNa: A Noise Resilient Fundamental Frequency Detection Algorithm for Speech and Music

Na Yang, *Student Member, IEEE,* He Ba, *Student Member, IEEE,* Weiyang Cai, Ilker Demirkol, *Member, IEEE,* Wendi Heinzelman, *Senior Member, IEEE*

***Abstract*—Fundamental frequency (F_0) is one of the essential features in many acoustic related applications. Although numerous F_0 detection algorithms have been developed, the detection accuracy in noisy environments still needs improvement. We present a hybrid noise resilient F_0 detection algorithm named BaNa that combines the approaches of harmonic ratios and Cepstrum analysis. A Viterbi algorithm with a cost function is used to identify the F_0 value among several F_0 candidates. Speech and music databases with eight different types of additive noise are used to evaluate the performance of the BaNa algorithm and several classic and state-of-the-art F_0 detection algorithms. Results show that for almost all types of noise and signal-to-noise ratio (SNR) values investigated, BaNa achieves the lowest Gross Pitch Error (GPE) rate among all the algorithms. Moreover, for the 0 dB SNR scenarios, the BaNa algorithm is shown to achieve 20% to 35% GPE rate for speech and 12% to 39% GPE rate for music. We also describe implementation issues that must be addressed to run the BaNa algorithm as a real-time application on a smartphone platform.**

***Index Terms*—Fundamental frequency detection, noise resilience, harmonics, Cepstrum, Viterbi algorithm.**

EDICS Categories: SPE-ANLS and AUD-MSP

I. Introduction

FOR human speech, pitch is defined by the relative highness or lowness of a tone as perceived by the human ear, and is caused by vibrations of the vocal cords. Since pitch is a subjective term, in this paper we use the objective term fundamental frequency (F_0), which is an estimate of pitch. If there were perfectly periodic speech signals, F_0 would be the inverse of the period of voiced speech. However, the interference of formant structure for speech signals, or the interference of spectral envelope structure for music signals, makes the accurate detection of F_0 difficult. Also, due to the aperiodicity of the glottal vibration itself and the movement of the vocal tract that filters the source signal, human speech is not perfectly periodic [1]. Additionally, accurate F_0 detection is difficult when the speech signal is corrupted with noise. Therefore, F_0 detection has always been a challenge in speech signal analysis.

A variety of speech-based applications can benefit from a more precise and robust F_0 detection algorithm. For example, F_0 detection is essential in automatic speech recognition, where pitch-accent patterns can be used to improve recognition performance [2], or homophones can be differentiated by recognizing tones [3]. For synthesized speech to be natural and intelligible, it is crucial to have a proper F_0 contour that is compatible with linguistic information such as lexical accent (or stress) and phrasing in the input text. Therefore, F_0 modeling can be used for speech synthesis [4][5]. F_0 and azimuth cues can be jointly used for speech localization and segregation in reverberant environments [6]. Moreover, in emotion detection or other affective measurement, it has been found that prosodic variations in speech are closely related to one's emotional state, and the F_0 information is important for the identification of this state [7]. A warning system has been developed in [8] to detect if a driver exhibits anger or aggressive emotions while driving, using statistics of the F_0 value and other metrics. Some health care providers and researchers implemented F_0 detectors and other behavior sensing technologies on mobile devices, such as smartphones, for behavioral studies [9] [10] or patient monitoring, such as the clinical trials conducted by the University of Pittsburgh for detecting depression in patients [11]. However, for these types of applications, the vehicle noise captured by the detector or the ambient noise captured by mobile devices may strongly influence the F_0 detection performance.

F_0 detection also plays a very important role in music signal analysis and music information retrieval, and has a broad range of applications. Music notation programs use F_0 detection to automatically transcribe real performances into scores [12]. Reliable F_0 extraction from humming is critical for query-by-humming music retrieval systems to work well [13]. Music fingerprinting technology also uses F_0 information for music identification among millions of music tracks [14]. F_0 detection in noisy music is also challenging. Music may be recorded in noisy environments such as in a bar or on the street. Noise may also be introduced by the recording device itself. One challenge is that the F_0 generated from tonal musical instruments spans a large range, normally from 50 Hz to 4,000 Hz [15]. For musical signals with high F_0, the wide range of possible F_0 candidates increases the likelihood of finding a wrong F_0 value. The other challenge is that, unlike for human speech, the sound for musical signals can last for several seconds, thus the overlapped musical tones can also be

Na Yang, He Ba, Weiyang Cai and Wendi Heinzelman are with the Department of Electrical and Computer Engineering, University of Rochester. Rochester, NY, 14627 USA. Phone: +1-(585)2758078, fax: +1-(585)2734919, e-mail: {nayang, he.ba, w.cai, wendi.heinzelman}@rochester.edu.

Ilker Demirkol is with the Department of Telematics Engineering, Universitat Politecnica de Catalunya. C/ Jordi Girona 1-3, Modul C3 Barcelona, Catalunya, Spain, 08034. Phone: +34-934011055, fax: +34-934011058, e-mail: ilker.demirkol@entel.upc.edu.

[1]This research was supported by the Eunice Kennedy Shriver National Institute of Child Health and Human Development Grant R01HD060789.

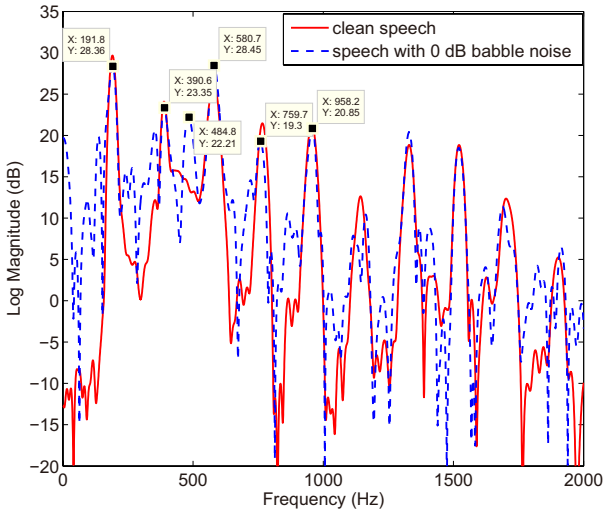


Fig. 1: Spectrum of one frame of clean speech and speech with babble noise at 0 dB SNR.

considered as noise. Due to these reasons, when performing F_0 detection in real scenarios, the quality of the input signal may be greatly degraded. Therefore, F_0 detection of musical signals in noisy environments is necessary.

Adding noise may introduce spectral peaks in the spectrum of the speech signal or distort the shape of the speech peaks, depending on the type and level of the noise. The key to detecting F_0 from noisy speech or music is to differentiate speech or music spectral peaks from noise peaks. In Fig. 1, we plot the spectrum of one frame from a clean speech file and the same frame with babble noise at 0 dB SNR. By examining this frame, we can see that F_0 is located at 192 Hz. By comparing the spectrum of the clean speech and the noisy speech, we can see that the added noise peaks distort the shape of the speech peaks, causing the highest point of the peak to be shifted. For the noise at 0 dB SNR, the amplitudes of the noise peaks can even be comparable with the amplitudes of the speech peaks. However, their locations in the frequency domain are not periodic, and the distribution of the noise peaks in the frequency range varies for different types of noise. Thus, the locations of the spectral peaks are affected less by the additive noise than the amplitudes of the peaks. Therefore, the ratios of harmonic frequencies are essential to find F_0 from a noisy signal.

Also, as seen in the spectrum of the noisy speech shown in Fig. 1, the first four harmonics are located at 391 Hz, 581 Hz, 760 Hz, and 958 Hz. The spectral peak located at 485 Hz is from the noise signal. We can see that the harmonics are not exactly spaced at integer multiples of the fundamental frequency F_0 in the frequency domain, and the higher order harmonics have larger drift than the lower order harmonics. Therefore, we need to set a tolerance range to account for the drifts when calculating the ratios of harmonic frequencies.

As existing F_0 detectors, such as Cepstrum [16], HPS [17], and Praat [18], do not perform well when the input data is noisy, we are motivated to design a noise resilient F_0 detection algorithm that is better suited for practical uses. This paper is based on our previous work [19], which proposed the BaNa algorithm for F_0 detection in speech signals. The BaNa algorithm is a hybrid F_0 detection algorithm that combines the idea of using the ratios of harmonic frequencies with tolerance ranges and the Cepstrum approach to find F_0 from a noisy signal. In this paper, we discuss F_0 detection for both speech and music signals, and we describe the simple modifications of BaNa required for music F_0 detection. We show that using the ratios of harmonic frequencies with pre-tuned tolerance ranges for F_0 detection enables the algorithm to be resilient to additive noise. We also show that incorporating Cepstrum and post-processing techniques can improve the F_0 detection performance.

In addition, we extend the work in [19] by evaluating the BaNa algorithm on a range of speech databases and by comparing it with seven classic and state-of-the-art F_0 detection algorithms. We test the proposed BaNa algorithm on real human speech and music samples corrupted by various types and levels of realistic noise. Evaluations show the high noise resilience of BaNa compared to the classic and state-of-the-art F_0 detection algorithms. For noisy speech at 0 dB SNR, BaNa achieve 20% to 35% Gross Pitch Error (GPE) rate for speech and 12% to 39% GPE rate for music. Also, we discuss issues with implementing BaNa on a smartphone platform. Test results on a real device show that our implementation of BaNa can process recorded speech files with a reasonable speed, opening the door for real-time F_0 detection on mobile platforms.

The rest of the paper is organized as follows. Section II provides a brief survey of well-known F_0 detection algorithms and also some of the most recent F_0 detection algorithms. Section III describes the BaNa algorithm for F_0 detection in noisy speech. Experimental settings and extensive experimental results comparing the BaNa algorithm with several classic and state-of-the-art F_0 detection algorithms using different speech databases are presented in Section IV and Section V, respectively. Section VI presents the slight modifications of the BaNa algorithm to improve its performance for F_0 detection in noisy music. We describe an implementation of the BaNa F_0 detection algorithm in Section VII. Finally, Section VIII concludes the paper.

II. Related Work

Among the well-known classic F_0 detection algorithms, autocorrelation function (ACF) and cross correlation are among the most widely used time domain methods. A number of algorithms have been developed based on these two approaches. Average Magnitude Difference Function (AMDF) [20] is a variation of ACF, which calculates a formed difference signal between the delayed signal and the original one. Since the AMDF algorithm does not require any multiplications, it is desirable for real-time applications. Praat [18] considers the maxima of the autocorrelation of a short segment of the sound as F_0 candidates, and chooses the best F_0 candidate for each segment by finding the least cost path through all the segments using the Viterbi algorithm. YIN [21] uses a novel difference function similar to autocorrelation to search

for the F_0 period. It further refines the F_0 detection result using some post-processing methods. Two types of modified difference function used in YIN are proposed in [22]. The RAPT F_0 tracker [23], on the other hand, is a variation of cross correlation, which computes the F_0 by extracting the local maxima of the normalized cross correlation function.

In the frequency domain, F_0 is found by searching for harmonic peaks in the power spectrum. The Cepstrum method [24] [16] is among the most popular algorithms. Cepstrum is found by computing the inverse Fourier transform of the log-magnitude Fourier spectrum, which captures the period in the speech harmonics, and thus shows a peak corresponding to the period in frequency.

Schroeder's frequency histogram [17] enters all integer submultiples of all the peaks in the spectrum in a histogram. Since F_0 is the integer submultiple of all the harmonics, in an ideal case, the entry with the highest weight in Schroeder's frequency histogram is the correct F_0. As pointed out in [25], Schroeder's frequency histogram is susceptible to octave errors, as F_0 and $F_0/2$ will have the same weight in Schroeder's frequency histogram. In cases where noise peaks are selected, Schroeder's histogram will make mistakes by selecting the greatest common divisor of both the harmonics and the noise peaks.

The Harmonic Product Spectrum algorithm (HPS) [17] multiplies the original signal with downsampled signals, thus in the frequency domain, the spectra of all the downsampled signals line up the peaks at the F_0 value for isolation. Another harmonic summation method is proposed in [26], which modifies the HPS method for multiple F_0 estimation in polyphonic music. The harmonic components' energy distribution is used, and F_0 candidates are selected using a competition mechanism. The algorithm is tested on three different instruments. However, for these harmonic summation methods, noise peaks with high amplitudes can be easily mistaken for harmonic peaks at low SNR scenarios. Since our proposed BaNa algorithm only relies on the locations of the harmonic peaks to calculate the frequency ratios of those spectral peaks, with the peak's amplitude information only being considered for peak selection, we show in Section IV-C and Section VI-E that the BaNa algorithm is more robust than the HPS algorithm for noisy speech and noisy music.

The PEFAC (Pitch Estimation Filter with Amplitude Compression) algorithm proposed in [27] is another frequency-domain F_0 detection algorithm for noisy speech. This approach estimates F_0 by convolving its power spectral density in the log-frequency domain with a filter that sums the energy of the F_0 harmonics while rejecting additive noise that has a smoothly varying power spectrum. Also, amplitude compression is applied before filtering to attenuate narrow-band noise components.

Some F_0 estimators operate in the time-frequency domain by applying time-domain analysis on the output of a filter bank. In the algorithm proposed by Jin and Wang [28], a new frequency channel selection method is proposed to select less corrupted channels for periodicity feature extraction. F_0 scores for each F_0 state are derived given the periodicity features and are given to a hidden Markov model for F_0 state tracking.

Recently, an increasing number of F_0 detection algorithms have been designed using a data-driven statistical approach to combat noise, such as the algorithms described in TAPS [29], Wu [30], and SAFE [31]. In [29], Huang et al. propose an F_0 estimation method that uses Temporally Accumulated Peaks Spectrum (TAPS). Since the harmonic structure of voiced speech changes more slowly than the noise spectrum over time, spectral peaks related to F_0 harmonics would stand out after temporal accumulations. Clean and noisy speech data is required to train the peak spectrum exemplar set and the Gaussian mixture model.

The Wu algorithm [30] is also a statistical approach, which integrates a new method for extracting periodicity information across different channels, and a Hidden Markov Model for forming continuous F_0 tracks. The modeling process incorporates the statistics extracted from a corpus of natural sound sources. The SAFE algorithm [31] also uses a data-driven approach to model the noise effects on the amplitudes and locations of the peaks in the spectra of clean voiced speech.

However, these data-driven approaches may not always be practical. Since these algorithms are trained with known noise types and specific noise levels, the noise information of the test sample is also required as input to the model. However, this information is not always available, since the user often does not know the type of noise, and it is even harder to measure the noise level. The proposed BaNa algorithm, on the other hand, does not require any prior information about the noise.

Though most F_0 detection algorithms were developed for F_0 detection in speech, a number of the aforementioned algorithms have also been used in music. The YIN and Praat algorithms are evaluated in [32] for synthetic signal and real-time guitar signal F_0 tracking. In [33], F_0 detection performance of the HPS algorithm and its variation called Cepstrum-Biased HPS are compared for interactive music. Clean cello and flute pieces are used in the experiments. However, robust F_0 detection in noisy music is still a topic that needs to be explored.

In this paper, we perform an extensive quantitative comparison analysis to show the performance, in terms of Gross Pitch Error (GPE) rate, for our proposed F_0 detection algorithm, BaNa, and several of the aforementioned algorithms (YIN, HPS, Praat, Cepstrum, PEFAC, SAFE, and Wu) for noisy speech and music signals.

III. BaNa F_0 Detection Algorithm for Speech

In this section, we describe the BaNa hybrid F_0 detection algorithm for speech.

A. Preprocessing

Given a digital speech signal, preprocessing is performed before the extraction of the F_0 values. In the BaNa algorithm, we filter the speech signal with a bandpass filter. Let F_0^{min} and F_0^{max} denote the lower limit and upper limit for F_0 values of human speech, respectively. The lower bound of the bandpass filter is set to F_0^{min}. The upper bound is set to $p \cdot F_0^{max}$, where p is the number of spectral peaks captured that will later be used for F_0 detection.

B. Determination of the F_0 candidates

Since harmonics are regularly spaced at approximately integer multiples of F_0 in the frequency domain, we use this characteristic of speech in the proposed BaNa algorithm to achieve noise resilience. If we know the frequency of a harmonic and its ratio to F_0, then F_0 can be easily obtained. However, even if a harmonic is discovered, its ratio to F_0 is unknown. Therefore, we propose an F_0 detection algorithm that looks for the ratios of potential harmonics and finds the F_0 based on them by applying the following steps.

Step 1: Search for harmonic peaks

Spectral peaks with high amplitudes and low frequencies are preferred to be considered for F_0 candidates, since peaks with high amplitudes are less likely to be caused by noise, and peaks with low frequencies are easier to be identified to be harmonics by calculating the ratios. Peaks with high frequencies may be high order harmonics, which cannot be used to calculate harmonic ratios, since we only consider the ratios of the first p harmonics. Therefore, we consider the p peaks with amplitudes higher than a certain threshold and with the lowest frequencies to derive F_0 candidates. We use the peak detection algorithm provided in [34] to search for the peaks in the spectrum. In [34], spectral peaks with small amplitudes are filtered out by setting a peak amplitude threshold, and peaks located very close to dominant peaks are smoothed by setting a threshold of the window width for smoothing. Settings that we use for the number of selected peaks p, the peak amplitude threshold parameter, and the window width parameter for the smoothing function in the peak detection function are presented in Table II.

Let $\hat{F}_i$ and $\left|\hat{H}_i\right|$ represent the frequencies and the magnitudes of the p spectral peaks with the lowest frequencies whose magnitudes are above a certain threshold, where $i = 0, \cdots, p-1$. We place the p peaks in ascending order of frequency to obtain the set of $\hat{F}_i$, denoted as $\hat{F}$. For most human speech, energy concentrates in the low frequency part, thus some or all of the p peaks are likely to be at the first p harmonics, which are at $m \times F_0$, $m = 1, \cdots, p$. For each frame, F_0 candidates are derived from the ratios of the frequencies of $\hat{F}$ using the following algorithm.

Step 2: Calculate F_0 candidates

We calculate the ratios $R_{ij} = \hat{F}_j / \hat{F}_i$ for all $\hat{F}_i, \hat{F}_j$, where $i < j$, and $i, j = 0, \cdots, p-1$. Take the number of selected spectral peaks $p = 5$ for example. If a calculated ratio R_{ij} falls into any tolerance range of the harmonic ratios shown in the left table in Fig. 2, we are able to find to which harmonics $\hat{F}_i$ and $\hat{F}_j$ correspond. For harmonic ratios with small numbers, we set adjacent tolerance ranges to be bounded with each other, i.e., the upper bound of the tolerance range of the ratio of $\hat{F}_4$ and $\hat{F}_3$ is the same as the lower bound of the tolerance range of the ratio of $\hat{F}_3$ and $\hat{F}_2$, which is $(5/4 + 4/3)/2 = 1.29$, as shown in Fig. 2. For harmonic ratios with large numbers, the width of the tolerance range is set to 0.2 or 0.4, depending on the ratio number. We show in Section IV-C that these tolerance range numbers are tuned to achieve the best F_0 detection performance.

A potential F_0 candidate can be obtained by dividing the harmonic by its ratio to F_0: $\tilde{F} = \hat{F}_i / m$, where $m = 1, \cdots, p$. Note that due to the imperfect periodicity of human speech, the harmonics may not be exactly on integer multiples of F_0, and we observed that higher order harmonics have even larger drift than lower order harmonics in practice. Therefore, we set a smaller ratio tolerance range for lower order harmonics, and we set a larger ratio tolerance range for higher order harmonics. In total, C_2^p ratio values are calculated between every pair of $\hat{F}$. Since both ratios of F_1/F_0 and F_3/F_1 are equal to 2, it is not trivial to differentiate to which harmonics this ratio belongs. In our algorithm, we assume it belongs to F_1/F_0 and calculate the F_0 candidate based on that.

In order to combat these octave errors, the proposed BaNa algorithm adds two other F_0 candidates. One added candidate is the spectral peak with the smallest frequency value, since we have found that in some cases only the F_0 peak is high enough to be detected. The other added F_0 candidate is the F_0 value found by the Cepstrum method. The reason is that the p spectral peaks we choose mainly belong to low frequency values. For some rare cases, the higher order harmonics (e.g., 5th to 10th harmonics) are found to yield higher spectral peak values compared to the low order harmonics. In that case, the spectral peaks at low frequencies are more vulnerable to noise. However, since the Cepstrum method depicts the global periodicity of the spectrum, and considers all spectral peaks, it can help to detect the F_0 in those rare cases. In Section V-B, we show the benefit of including the spectral peak with the smallest frequency value and the Cepstrum F_0 as additional candidates.

The number of F_0 candidates derived from the ratio analysis and the two added candidates, K, is then less than or equal to $C_2^p + 2$. Among these K F_0 candidates, the ones that are out of the F_0^{min} to F_0^{max} human speech range are discarded, and the number of candidates is reduced from K to K'. If no candidate is derived from the ratios, we set the F_0 value to 0 Hz. We then order the K' candidates in ascending order of frequency. F_0 candidates that are within ξ Hz of each other are considered to be "close" candidates. For each of these K' candidates $\tilde{F}_k$, where $k = 1, ..., K'$, we count the number of "close" candidates U_k, and select the one with the largest number of "close" candidates to be a "distinctive" candidate $\check{F}_d$, where $d = 1, ..., D$, and D is the number of "distinctive" candidates. The "distinctive" candidate and its "close" candidates are deleted from the candidate list. If there is more than one candidate that has the same largest number of "close" candidates, we select the one with the smallest frequency to be the "distinctive" candidate. We continue the same procedure for the remainder of the list until the list is empty. We set the number of "close" candidates, including the chosen candidate itself, to be the confidence score V_d for the corresponding "distinctive" candidate $\check{F}_d$. Among the D "distinctive" candidates, where $D \leq K'$, the ones with higher confidence scores are more likely to be F_0.

In Fig. 2, we use the frame shown in Fig. 1 to illustrate the process of calculating F_0 candidates. In Fig. 1, the dotted line represents the spectrum of one frame of speech with 0 dB babble noise. The five selected spectral peaks that have the lowest frequencies are located at 192 Hz, 391 Hz,

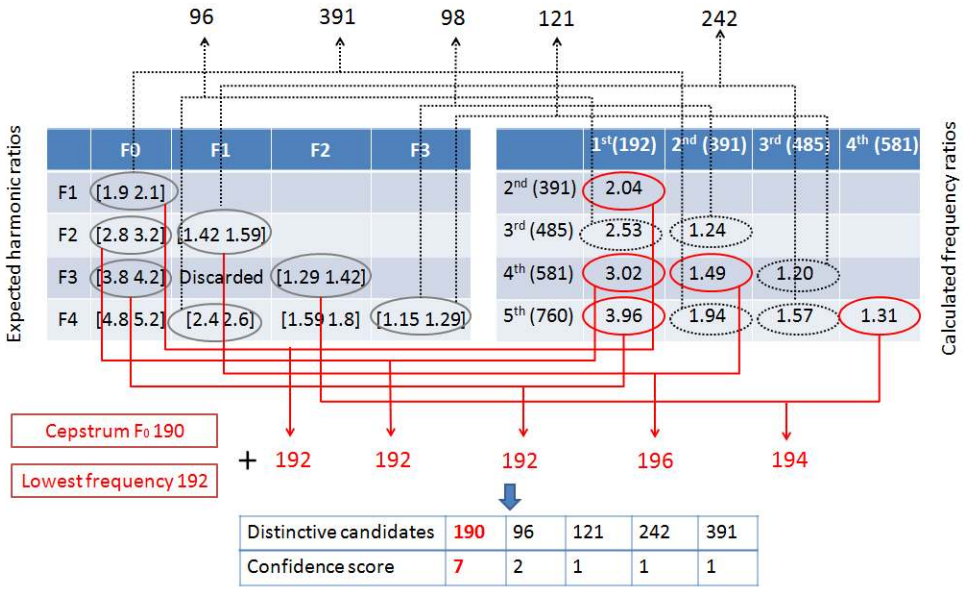


Fig. 2: Tolerance ranges for harmonic ratios when the number p of selected spectral peaks is set to 5, and an example to illustrate the procedure for determining the F_0 candidates.

485 Hz, 581 Hz, and 760 Hz. The 485 Hz peak is caused by the noise signal, and the remaining four peaks are from the speech signal. We map each calculated frequency ratio in the right table in Fig. 2 to one expected harmonic ratio in the left table in Fig. 2. For example, the ratio of the 5^{th} and 4^{th} spectral peaks is $\hat{F}_5/\hat{F}_4$ = 760/581 = 1.31, which maps to the [1.29 1.42] frequency ratio tolerance range for the expected frequency ratio of the 3^{rd} and 2^{nd} harmonics. Therefore, the F_0 candidate is derived as 581/3=194 Hz. In this example, all calculated frequency ratios are mapped to one expected harmonic ratio in the left table, which results in 10 F_0 candidates. The Cepstrum candidate and the peak with the lowest frequency are added as two additional F_0 candidates, which are 190 Hz and 192 Hz, respectively.

If we use the parameters shown in IV-C, all the 12 candidates are within the F_0^{min} = 50 Hz to F_0^{max} = 600 Hz range for F_0. Candidates that are within ξ = 10 Hz of each other are considered to be "close" candidates. In Fig. 2, the bottom table shows the "distinctive" candidates and their confidence scores. The 190 Hz candidate has the highest confidence score, which is very close to the ground truth F_0, i.e., 191 Hz calculated from the corresponding clean speech signal. In Fig. 2, correct F_0 candidates are listed on the bottom and are marked by solid red lines. Incorrect F_0 candidates are listed on the top and are marked by dotted black lines. We can see that the candidates calculated from the 485 Hz noise peak are all incorrect candidates.

C. Selection of F_0 from the candidates

In Section III-B, the "distinctive" candidates of individual frames are obtained independently. However, the F_0 values of neighboring frames may correlate, since the F_0 values of human speech exhibit a slow time variation, and hence, large F_0 jumps among subsequent frames are rare. Therefore, we use the Viterbi algorithm [35] for post-processing to go through all the candidates in order to correct F_0 detection errors, similar to the post-processing used in the Praat algorithm [18]. We aim to find a path that minimizes the total cost, which consists of two parts: the frequency jumps between the candidates of two consecutive frames, and the inverse of the confidence scores of each "distinctive" candidate.

Let $\check{F}_i^n$ denote the i^{th} "distinctive" F_0 candidate of frame n and N_{frame} denote the number of frames in the given speech segment. Moreover, let p_n denote the index of the chosen F_0 candidate for the n^{th} frame. Thus, $\{p_n | 1 \leq n \leq N_{frame}\}$ defines a path through the candidates. For each path, the path cost is defined to be

$$PathCost\left(\{p_n\}\right) = \sum_{n=1}^{N_{frame}-1} Cost\left(\check{F}_i^n, \check{F}_j^{n+1}\right), \quad (1)$$

where $p_n = i$ and $p_{n+1} = j$. The $Cost$ is used to calculate the cost of adjacent frames. We define the function $Cost$ by using the F_0 differences between the adjacent frames and the confidence score of the candidates. The F_0 difference is modeled similarly with the $transition\ cost$ defined in the Praat algorithm [18]. The larger the F_0 difference, the higher the $Cost$ should be. We present the F_0 difference in the Mel scale, which is a perceptual scale of F_0 judged by listeners. The perceived F_0 in the Mel scale has a logarithm relation with the F_0 in frequency, as shown in (2):

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right). \quad (2)$$

Therefore, in the cost function, the F_0 difference in frequency is modeled as the logarithm of the F_0 division in the Mel scale. The other part of the cost function is modeled using the confidence score. We assign a lower cost to those F_0 candidates with higher confidence scores, thus we use the inverse of the confidence score in the expression of the cost function. A weight w is introduced to balance the two parts. The setting for this value is shown in Table II. Then, $Cost$ is

defined mathematically as

$$Cost\left(\check{F}_i^n, \check{F}_j^{n+1}\right) = \left|\log_2 \frac{\check{F}_i^n}{\check{F}_j^{n+1}}\right| + w \times \frac{1}{V_i^n}, \qquad (3)$$

where V_i^n is the confidence score of the i^{th} "distinctive" F_0 candidate of the n^{th} frame.

We use the Viterbi algorithm to find the minimum cost path, i.e., the path that reduces the F_0 jumps the most, while giving priority to the F_0 candidates with higher confidence scores. The optimal path is found for each voiced part in the speech. The Praat algorithm also uses the Viterbi algorithm to choose F_0 from several F_0 candidates for each frame. However, the F_0 candidates of Praat are local maxima of the autocorrelation of each frame, which have the same confidence score to be selected as F_0. On the other hand, the F_0 candidates in BaNa have different confidence scores, and thus F_0 candidates derived from noise spectral peaks are less likely to be selected as F_0. Therefore, the cost function of BaNa's Viterbi algorithm shown in (3) is different from that in the Praat algorithm. The complete BaNa algorithm that describes the selection of the peaks and the calculation and selection of the F_0 candidates is given in Algorithm 1.

For each frame, the time complexity to calculate K F_0 candidates by calculating frequency ratios of p selected peaks is $\mathrm{O}\left(p^2\right)$. The time complexity to calculate D 'distinctive' candidates from K' remaining candidates is $\mathrm{O}\left(K'^3\right)$, which is the most complex process. The time complexity to use the Viterbi algorithm to choose the final F_0 from 'distinctive' candidates is $\mathrm{O}\left(D^2\right)$.

Algorithm 1 The BaNa F_0 Detection Algorithm

// For frame n***:***
// Select harmonic peaks
select $\hat{F}^n$: the p peaks with lowest frequencies
// Calculate F_0 ***candidates***
number of candidates $K \leftarrow 0$
for i =1 to p, $j = i+1$ to p **do**
  ratio $R_{ij} = \hat{F}_j^n / \hat{F}_i^n$
  for m =1 to p, $m' = m+1$ to p **do**
    if R_{ij} falls in the left table of Fig. 2 and close to $\frac{m'}{m}$ **then**
      $K \leftarrow K+1$, $\tilde{F}_K^n \leftarrow \hat{F}_i^n / m$
    end if
  end for
end for
$K \leftarrow K+1$, add spectral peak with the lowest frequency $\hat{F}_1^n$: $\tilde{F}_K^n \leftarrow \hat{F}_1^n$
$K \leftarrow K+1$, add Cepstrum F_0: $\tilde{F}_K^n \leftarrow Cepstrum\ F_0$
discard $\tilde{F}^n$ that are out of the F_0^{min} to F_0^{max} range
$K' \leftarrow$ number of remaining candidates $\tilde{F}^n$
number of "distinctive" candidates $D^n \leftarrow 0$
while $\exists \tilde{F}^n \neq null$ **do**
  for k =1 to K' **do**
    if $\tilde{F}_k^n \neq null$ **then**
      num. of "close" candidates of $\tilde{F}_k^n$: $U_k \leftarrow 0$
      for $l = 1$ to K' **do**
        if $\left|\tilde{F}_l^n - \tilde{F}_k^n\right| \leq \xi$ Hz **then**
          $U_k \leftarrow U_k + 1$
        end if
      end for
    end if
  end for
  $D^n \leftarrow D^n + 1$, $V_D^n \leftarrow \max U_k$
  "distinctive" candidate $\check{F}_{D^n}^n \leftarrow \tilde{F}^n$ with max U_k
  $\tilde{F}^n$ with max $U_k \leftarrow null$
  all "close" candidates of $\tilde{F}^n$ with max $U_k \leftarrow null$
end while
// For all frames within a voiced segment:
// Choose F_0 ***from "distinctive" candidates***
for n =1 to number of frames N_{frame} **do**
  for i, j =1 to D^n **do**
    $Cost\left(\check{F}_i^n, \check{F}_j^{n+1}\right) = \left|\log_2 \frac{\check{F}_i^n}{\check{F}_j^{n+1}}\right| + w \times \frac{1}{V_i^n}$
  end for
end for
return $\{p_n\}$ of $\min\{PathCost\} \leftarrow Viterbi\,(Cost)$, where path $\{p_n\}$ denotes F_0 for all frames

IV. Experimental Settings for BaNa F_0 Detection For Speech

In this section, we present the speech and noise databases we use for F_0 detection performance evaluation, the error measurement metric, and parameter tuning of the proposed algorithm.

A. Speech and noise databases

Noisy speech samples can be generated by adding noise recorded in noisy environments to clean speech samples. Using this approach, the ground-truth F_0 values can be obtained from the clean speech. An alternative approach is to use speech samples directly recorded in real noisy environments, such as the SPEECON database [36], where additive noise, reverberations, and channel distortions are present. The ground-truth F_0 values in the SPEECON database are derived by manually F_0-marked recordings from a close speaking microphone with relatively little noise (clean speech). Several F_0 detection algorithms use the SPEECON database to evaluate their performance [37] [38] [39].

In this work, we use noisy speech samples generated from clean speech and different types of additive noise.

The clean speech samples we use are taken from four English speech databases: LDC [40], Arctic [41], CSTR [42], and KEELE [43]. Since female speakers normally have higher F_0 values than male speakers, approximately an equal number

TABLE I: Evaluated speech databases and their features. Parameters are tuned using samples from the Arctic database.

Speech databases	**Emotion**	**# of speakers**	**# of selected samples**	**% of voiced frames**	**Has F_0 ground truth?**
Arctic [41]	neutral	4	10	54.2%	No
LDC [40]	various	7	20	50.4%	No
CSTR [42]	neutral	2	100	50.3%	Yes
KEELE [43]	neutral	10	10	50.4%	Yes

of speech samples from male and female speakers are chosen from these databases. Also, since the frequency characteristics in speech differ from person to person, we select speech samples from all the available speakers within these databases. Table I presents the specifications of these speech databases.

The LDC database is the Emotional Prosody Speech and Transcripts Database from Linguistic Data Consortium. It is chosen because it includes speech samples with strong emotions such as hot anger and elation, for which the F_0 values may change dramatically even within a short utterance. In the BaNa algorithm, the difference of F_0 values for neighboring frames is taken into consideration by the Viterbi algorithm. Therefore, the LDC database helps to investigate whether this discontinuity in F_0 values may influence the performance. The Arctic, CSTR and KEELE databases all contain speech samples with neutral emotion. All the speech samples used for the evaluation are included in the BaNa toolkit [44].

To test the noise resilience of the investigated algorithms, eight types of noises are added to the original signals with different SNR levels. The noise database we use is the NOISEX-92 noise database [45], in which we choose six different types of real life background noise: speech babble (labeled as $babble$ in the figures for performance comparison), destroyer engine room noise ($engine$), destroyer operations room noise ($operation$), factory floor noise ($factory$), vehicle interior noise ($vehicle$), high frequency radio channel noise ($highfreq$), as well as two common types of noise: white noise ($white$) and pink noise ($pink$). To generate noisy speech with a certain SNR value, the signal energy is calculated only on the voiced part, and the noise is amplified or attenuated to a certain level to meet the target SNR value.

B. Error measurement metric

For the noisy speech data, if the detected F_0 deviates more than 10% from the ground truth value, it is counted as a gross pitch error. Otherwise, it is counted as a fine pitch error. The Praat algorithm also uses the 10% deviation range in their error measurement in [18]. Gross Pitch Error (GPE) rate is the percentage of incorrectly detected F_0 values in voiced speech segments. GPE rate has been widely used as the error measurement metric for F_0 detection [29] [31] [46]. Mean and standard deviation of Fine Pitch Errors (FPE) are also used in this study. FPE is calculated by the relative deviation of the detected F_0 from the ground truth F_0, with the unit in percent, for any pitch that does not represent a Gross Pitch Error [47] [48].

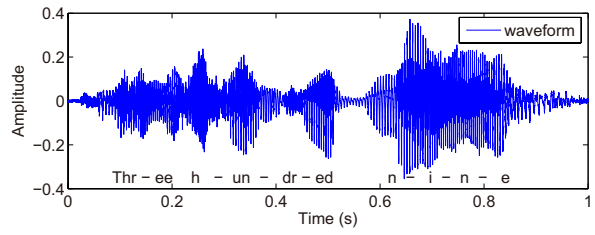
The F_0 ground truth values for the CSTR and KEELE databases are provided, which are obtained from the simultaneously recorded laryngograph signals. We downloaded the speech data and the ground truth values for the CSTR and KEELE databases from the SAFE toolkit [49], and then shifted the ground truth values in time as needed to line up with the F_0 detected by all the algorithms tested. For the LDC and Arctic databases with no F_0 ground truth provided, we use auto-labeled F_0 values of the original clean speech as the ground truth F_0 values and the voiced/unvoiced delineation, since the original speech samples are clean and with very little background noise. To best estimate the ground truth F_0 values, we calculate the detected F_0 values of three algorithms: PEFAC, YIN and Praat, which all perform well in F_0 detection for clean speech. For one frame, if the detected F_0 values from all three algorithms are within 10%, we assume that this frame is voiced, and the auto-labeled ground truth is determined by averaging the three detected F_0 values. Otherwise, we assume that the frame is unvoiced and do not detect F_0 for that frame.

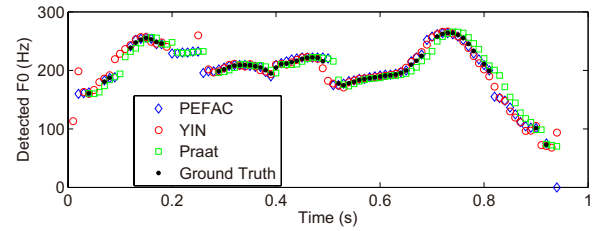
Fig. 3(a) shows an example of a clean speech recording of the utterance 'three hundred (and) nine' along with the auto-labeled F_0 values as the ground truth. The word 'and' in the middle is skipped and is not spoken. We can see that for most of this clean utterance, the detected F_0 values from the three algorithms are very close. We use black solid dots to represent the ground truth F_0 values, which are calculated by averaging the detected F_0 values from PEFAC, YIN and Praat. We also note that the detected F_0 values from these three algorithms differ at frames corresponding to unvoiced stop consonants, i.e., 'th' in 'three' and 'h' in 'hundred', and discontinuities, i.e., the spaces between two words. Those frames are regarded as unvoiced and are ignored. For some frames, no F_0 value is shown on the plot for Praat, since Praat has its own voiced/unvoiced frame detection, and those frames are considered as unvoiced by Praat. The corresponding spectrogram is shown in Fig. 3(b), in which the lowest dark red curve verifies the calculated F_0 ground truth in Fig. 3a. The frame length used to compute the spectrogram is 60 ms.

The MATLAB code for the BaNa algorithm is available on the University of Rochester Wireless Communications and Networking Group's website [44]. Although the voiced/unvoiced speech detection is not within the scope of this paper, we provide one version of the MATLAB implementation of the BaNa algorithm with an automatic voice marker [44]. The voiced/unvoiced speech detector used in this version of the BaNa code is the one implemented in [24] as the voiced/unvoiced speech detector for the Cepstrum F_0 detection algorithm. Frames with a dominant cepstrum peak, with an amplitude higher than the amplitude of the second highest peak by a certain threshold, are considered as voiced frames. However, we have not evaluated the performance of this voiced/unvoiced speech detector on noisy speech. Other voiced/unvoiced speech detectors are also available in the literature [31][37].

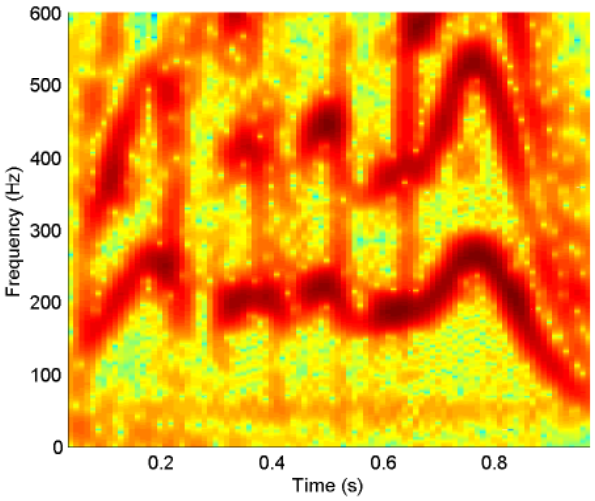
C. Parameter tuning

The frame shift is set to 10 ms in order to obtain smooth F_0 detection results. The absolute value of the Fourier transform of the Hann windowed speech signal is calculated, with the FFT size set to 2^{16} = 65,536 to provide good frequency





(a)



(b)

Fig. 3: For one clean speech utterance: a) speech waveform and the auto-labeled ground truth F_0 derived from three algorithms: PEFAC, YIN, and Praat, and b) the spectrogram. The frame length used to compute the spectrogram is 60 ms.

resolution. Candidates that are within ξ = 10 Hz of each other are considered to be "close" candidates. Since the F_0 of human speech is normally higher than 50 Hz and can be as high as 600 Hz for children or female voices [50], we set the lower limit and the upper limit for F_0 of human speech to be F_0^{min} = 50 Hz and F_0^{max} = 600 Hz, respectively.

There are several parameters in the BaNa algorithm that can be pre-tuned to achieve a more accurate estimate of F_0. The Arctic samples are used for the tuning of these parameters, and the set of parameters that provides the lowest GPE rate averaged over all levels of noise and all types of the NOISEX-92 noise [45] is chosen as the parameter set used in this paper.

The parameter settings are shown in Table II. To obtain a stable estimate of F_0, the frame length is chosen to be at least three times the F_0 period. Since the minimum F_0 we consider for both speech and music is 50 Hz, the frame length is thus $1/50 \times 3$ = 0.06 s, i.e., 60 ms. We also list in Table II other

TABLE II: Optimal values of tuned parameters, and other values of the parameters for which BaNa algorithm is tested.

Parameters	**Optimal value**	**Other values tested**
Frame length	60 ms	20 ms, 90 ms
Number of chosen spectral peaks p	5	3, 4, 6, 7
Spectral peak amplitude threshold in peak selection	1/15 of the highest peak	1/25, 1/20, 1/10 of the highest peak
Window width for smoothing in the frequency domain in peak selection	50 Hz	40 Hz, 60 Hz, 70 Hz, 80 Hz
Tolerance range for harmonic ratios	Numbers shown in Table I	Narrowed range, extended range
Weight w in the cost function in (3)	0.4	0.05, 0.1, 0.2, 0.3, 0.5, 0.6, 0.7, 0.8, 0.9

frame length values that we have tested. Using the 20 ms frame length, which is one F_0 period at 50 Hz, results in a higher GPE rate. Although using the 90 ms frame length can slightly reduce the GPE rate, the temporal resolution is sacrificed.

Parameters in the spectral peak selection process are also tuned, including the number of spectral peaks p chosen to calculate the F_0 candidates, the spectral peak amplitude threshold and the threshold of the window width for smoothing, which is the width of the smoothing function applied before spectral peak detection. With these parameters being properly set, spectral peaks with low amplitudes and small widths are not chosen. We tested the performance of BaNa by choosing more or fewer spectral peaks, which means possibly more or fewer harmonics, but we found that choosing 5 peaks provides good F_0 detection performance. Also, choosing more spectral peaks increases the complexity in calculating the F_0 candidates.

Other parameters that are tuned are the tolerance range for the harmonic ratios used in the left table of Fig. 2, and the weight parameter used in the cost function in (3). Note that these parameters represent the optimal set across all noise types and SNR values for the Arctic speech database; they may not be optimal for a given noise type or SNR value or samples from other databases. A user could, of course, optimize the parameters for specific noise conditions, but we will show in Section V that using these tuned parameters provides good performance without the need for tuning for specific noise environments. Note that for all the other F_0 detection algorithms, we choose their default parameters in the evaluation.

To evaluate the parameter sensitivity of the BaNa algorithm on new types of noise, we use another widely-used noise database [51] with eight types of common ambient noise, including airport, babble, car, exhibition, restaurant, street, subway, and train noise. This noise database was used to construct the AURORA noisy speech database [52] for speech recognition. Note that the AURORA noise database is only used for this parameter sensitivity test. All the remaining performance evaluations in this paper are performed on noisy speech and noisy music generated using noise samples from the NOISEX-92 noise database [45].

We compare the performance of BaNa on the LDC database [37] by using 1) the set of parameters provided in the paper, that are tuned on the Arctic database and the NOISEX-

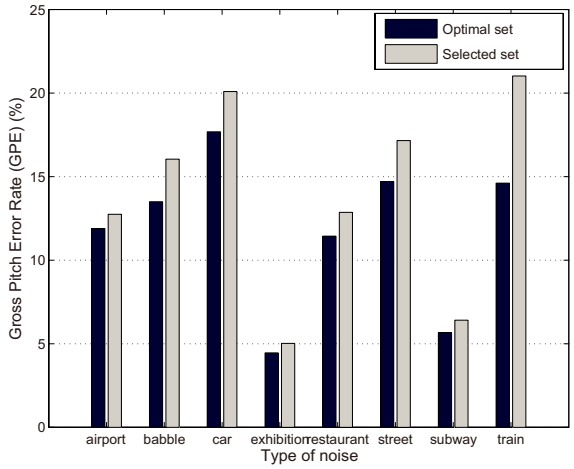


Fig. 4: GPE rates of BaNa for the LDC database [40] with eight types of AURORA noise [51] averaged over all SNR values, using individually optimized parameter sets that provide the lowest GPE rates for a specific type of AURORA noise, and using the tuned parameter set selected in the paper. Detected F_0 deviating more than 10% from ground truth are errors.

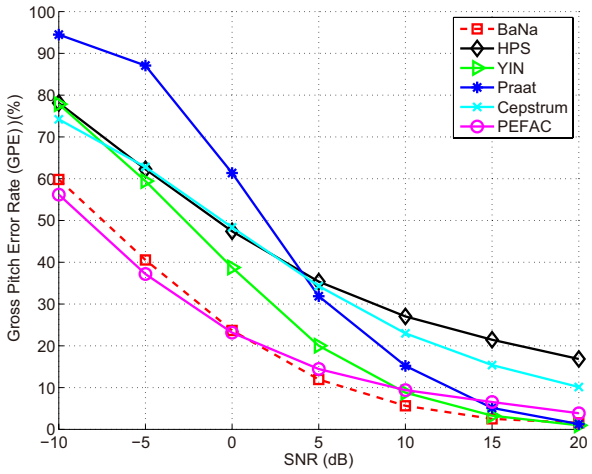


Fig. 5: GPE rate of the different algorithms for the LDC database [40], averaged over all eight types of noise. Detected F_0 deviating more than 10% from ground truth are errors.

92 noise database [45], and 2) the parameter sets that are individually optimized on a specific type of noise from the the AURORA noise database [51] that yields the lowest GPE rates for the LDC database, averaged over 0 dB, 5 dB, 10 dB, 15 dB, and 20 dB SNR values.

As shown in Fig. 4, the difference in the performance when using the individually optimized parameter sets and when using the parameter set selected in the paper is relatively small for most noise types. These results show that the performance of BaNa is not very sensitive to the specific parameters chosen. Thus, we can trade a slight drop in the GPE performance of BaNa for the benefit of not needing to optimize the parameters for a specific type of noise environment.

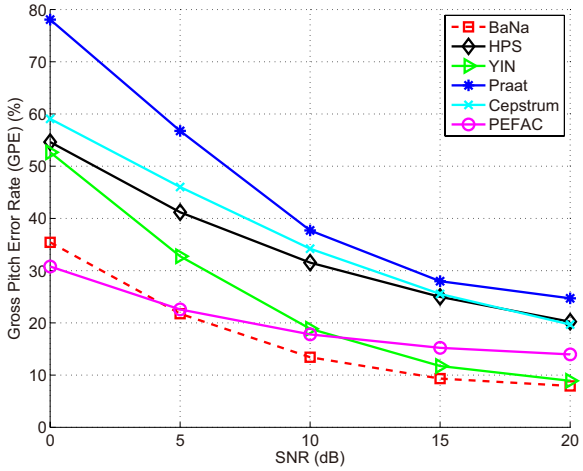


Fig. 6: GPE rate of the different algorithms for the CSTR database [42], averaged over all eight types of noise. Detected F_0 deviating more than 10% from ground truth are errors.

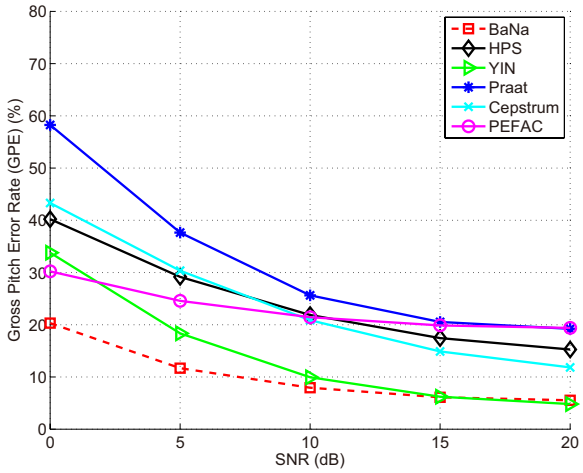


Fig. 7: GPE rate of the different algorithms for the KEELE database [43], averaged over all eight types of noise. Detected F_0 deviating more than 10% from ground truth are errors.

V. F_0 Detection Performance For Speech Signals

In this section, we compare the F_0 detection performance of the proposed BaNa algorithm with that of several classic and state-of-the-art algorithms on speech signals in various noisy environments and for a wide range of SNR values. Seven algorithms are considered due to their popularity or good performance: YIN, HPS, Praat, Cepstrum, PEFAC, SAFE, and Wu. These algorithms have been described in Section II. The source code for YIN, Praat, Cepstrum, PEFAC, SAFE, and Wu are from [53], [54], [24], [55], [49], and [56], respectively. We implement the HPS algorithm based on the algorithm described in [17]. F_0 detection in eight different types of noise environments are evaluated, where noisy speech samples are generated by adding background noise to clean real speech samples with different noise power levels to achieve different SNR values.

Note that in our study, we only detect F_0 when only one speaker is speaking or only one instrument is played.

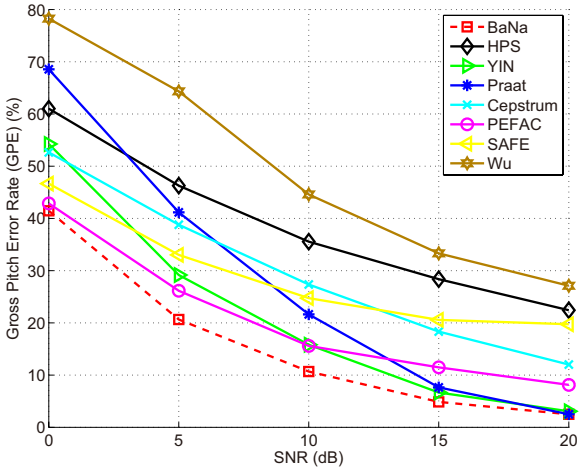


Fig. 8: GPE rate of the different algorithms for the LDC database [40] for speech with babble noise. Detected F_0 deviating more than 10% from ground truth are errors.

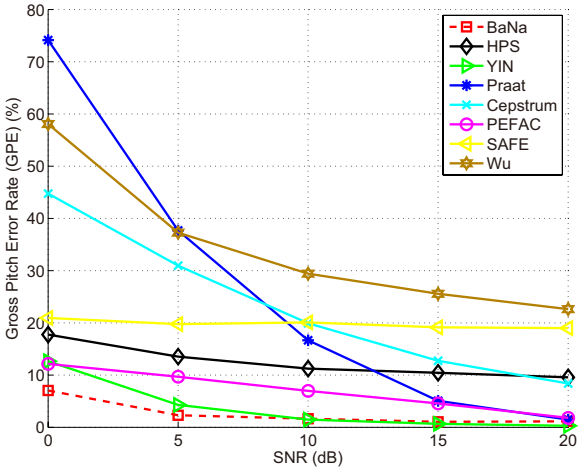


Fig. 9: GPE rate of the different algorithms for the LDC database [40] for speech with white noise. Detected F_0 deviating more than 10% from ground truth are errors.

If multiple people are speaking or multiple instruments are played at the same time, multiple F_0 values coexist. Multiple F_0 detection, as studied in work such as [57] [58] [59] [60], is not within the research scope of this work.

A. F_0 detection performance for speech

We test all the F_0 detection algorithms on each one of the speech databases mentioned in Section IV-A, except the Arctic database, which was used for tuning the BaNa parameters. The GPE rate is evaluated as a function of SNR value, where the GPE rate is averaged over all types of noise for each SNR value.

For the LDC database with emotional utterances, Fig. 5 depicts the results, which shows that the BaNa algorithm achieves the best F_0 detection accuracy, i.e., the lowest GPE rate, among all of the algorithms for 0 dB SNR and above 0 dB SNR. PEFAC performs slightly better than BaNa at -5 dB SNR and -10 dB SNR. BaNa achieves the lowest GPE rate of 20.6%, which is obtained by averaging over -10 dB, -5 dB, 0 dB, 5 dB, 10 dB, 15 dB, and 20 dB SNR levels. Similar to the BaNa algorithm, the HPS algorithm is also based on the ratios of the potential harmonics. However, in real speech, the harmonics are not integer multiples of F_0, which may greatly affect the F_0 detection performance. We can also see that the BaNa algorithm has a very high resilience to severe noise, as it only wrongly detects 23.7% of F_0 values with noise at 0 dB SNR.

For a more stringent evaluation, we have also tested all algorithms on the LDC database using the GPE rate with a 5% deviation range. BaNa performs slightly better than PEFAC for above 5 dB SNR, while PEFAC performs slightly better than BaNa for below 5 dB SNR. The GPE rate for BaNa with a 5% deviation range is 30% at 0 dB, averaged over all 8 types of noise. The mean and standard deviation of Fine Pitch Errors (FPE) are also evaluated, using a 10% deviation range. The mean and standard deviation of FPE for BaNa are both 1.9% at 0 dB, which are only about 0.5% higher than the mean and standard deviation of FPE for PEFAC and HPS.

Since the SAFE algorithm is only trained to detect F_0 for speech with babble noise and white noise, we show its performance for these two types of noise at the end of this section, where we also present Wu's results, since it is unclear how to run Wu's code on long speech samples. Therefore, we only test the Wu algorithm for the LDC database. Since the -10 dB SNR and -5 dB SNR scenarios are very severe noisy environments, we present the rest of the F_0 detection results for noise conditions with SNR greater than or equal to 0 dB.

The GPE rates for speech with neutral emotion are shown in Figs. 6 and 7 for the CSTR and KEELE databases, respectively. Similar results are obtained for the proposed BaNa algorithm and the five other algorithms, with the main difference being that PEFAC at 0 dB SNR performs slightly better than BaNa for the CSTR database. With noise at 0 dB SNR, the GPE rate of BaNa is 35.4% for the CSTR database, and 20.3% for the KEELE database. However, since the ground truth F_0 values for the CSTR and KEELE databases are based on the laryngograph signals, we checked the ground truth values for a few speech samples and found that there are many spikes and discontinuities in the ground truth F_0 values found by using the laryngograph, especially on the boundaries of voiced and unvoiced frames. We can see from Figs. 6 and 7 that even at 20 dB SNR, the lowest GPE rate for all algorithms is still greater than 5%. While the ground truth for these databases may include several unvoiced frames and less reliable data, we present these results for the CSTR and KEELE databases in Figs. 6 and 7 in order to facilitate comparison with other F_0 detection algorithms that use these databases.

Babble noise and white noise are the most common types of noise in speech processing. Since the SAFE algorithm is only trained on babble noise and white noise, we only compare the results of SAFE for these two types of noisy speech. The KEELE database is used for training of SAFE, as in [31], and the LDC database is used for testing. We also show the performance of the Wu algorithm proposed

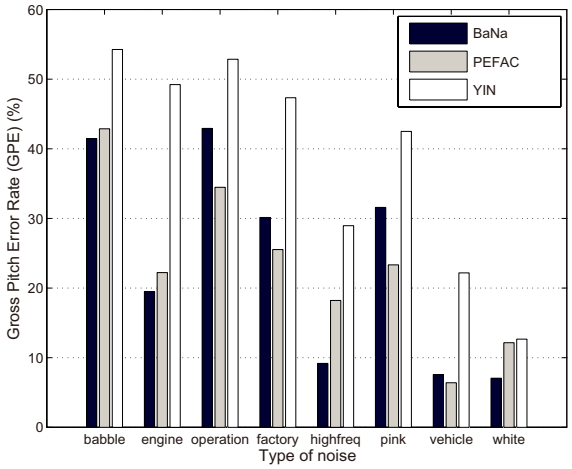


Fig. 10: GPE rate of BaNa, PEFAC and YIN for the LDC database [40] with eight types of noise at 0 dB SNR. Detected F_0 deviating more than 10% from ground truth are errors.

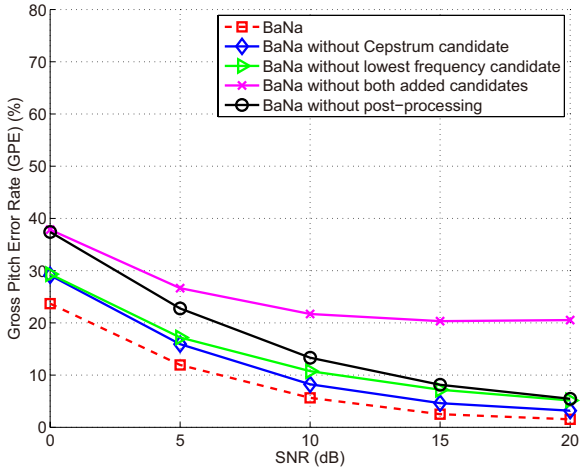


Fig. 11: GPE rate of BaNa, BaNa without the Cepstrum candidate, BaNa without the lowest frequency candidate, BaNa without both added candidates, and BaNa without post-processing for the LDC database, averaged over all eight types of noise. Detected F_0 deviating more than 10% from ground truth are errors.

in [30]. The detected F_0 value is considered to be an error if it deviates more than 10% from the ground truth value, and again we use GPE rate as the error measurement metric. Figs. 8 and 9 present the GPE rate of the different algorithms for the LDC database for speech with babble noise and white noise, respectively. We can see that the F_0 detection for speech with babble noise is more difficult than F_0 detection for speech with white noise. Results show that BaNa, YIN, and PEFAC provide the lowest GPE rate for F_0 detection for speech with babble and white noise.

Speech with noise at 0 dB SNR is a challenging scenario for F_0 detection. For a head to head comparison, we present the performances of the BaNa algorithm and the closest competing algorithms, PEFAC and YIN, using the LDC database for eight different types of noise at 0 dB SNR in Fig. 10. We can see that BaNa has the lowest GPE rate for four out of eight types of noise. For the babble noise, which is a very common type of noise in real life scenarios, the BaNa algorithm achieves a 41.5% GPE rate compared with PEFAC's 42.9% and YIN's 54.3%, even when the speech is only slightly audible by the human ear. We can also see from Fig. 10 that the babble noise and the destroyer operations noise cause the worst degradation in the F_0 detection performance. By investigating the spectrum of several noisy speech samples, we found that the high spectral peaks of these two types of noise concentrate in the same frequency range as the spectral peaks of speech. On the other hand, the high spectral peaks of high frequency noise, vehicle noise and white noise are distributed in the frequency range, which is quite different from the spectrum of human speech, making it easier to differentiate speech spectral peaks from noise spectral peaks. Therefore, the GPE rate for speech with these types of noise remains at a relatively low level even at 0 dB SNR.

B. Breakdown analysis of the BaNa algorithm

As we can see from the above F_0 detection performance for speech, the proposed BaNa algorithm has the most advantage at 0 dB SNR across almost all speech databases. To provide additional insights to understand the core design of this noise-resilient algorithm, as well as the differences between BaNa and other algorithms, we provide a breakdown analysis of BaNa here:

- BaNa only considers the frequency ratios among the lower-order harmonics, and also Cepstrum is included as one of the F_0 candidates, thus BaNa is less affected by octave errors than Schroeder's frequency histogram.
- Harmonic summation methods use the amplitudes of spectral peaks to weight the frequency histogram, which is not a noise-resilient approach, since noise peaks with high amplitudes are likely to be chosen as F_0 after the harmonic summation. The BaNa algorithm, on the other hand, only uses the peak amplitude information to choose the spectral peaks, but the F_0 candidates calculation is solely based on the frequency ratios of the chosen peaks. No peak amplitude information is used at this point, as it may be severely corrupted by noise.
- By providing a tolerance range for these frequency ratios, our algorithm is able to combat the frequency drift of harmonics and shape distortions of harmonic peaks caused by the noise.
- Post-processing using the Viterbi algorithm in BaNa considers the F_0 continuity, which helps to choose the F_0 candidates more accurately.
- Since the F_0 candidates calculated from peak frequency ratios are only based on lower-order harmonics, adding the Cepstrum as an additional candidate helps to capture the general period information for all spectral peaks.

To show the effectiveness of using the Cepstrum candidate and the spectral peak with the lowest frequency as two additional F_0 candidates, and using the Viterbi post-processing, in Fig. 11 we plot the GPE rates for the BaNa algorithm, BaNa

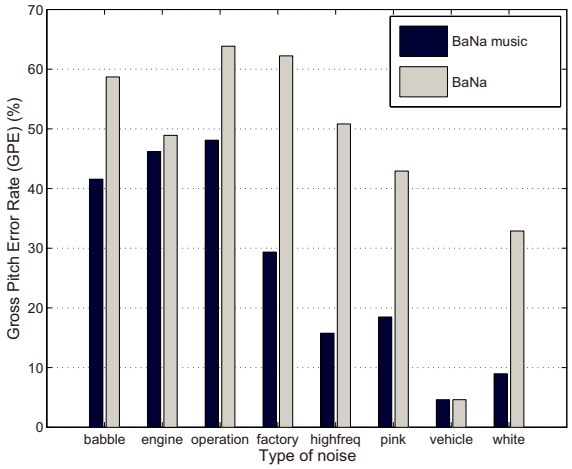


Fig. 12: GPE rate of BaNa and BaNa music for a piece of violin music with eight types of noise at 0 dB SNR. Detected F_0 deviating more than 3% from ground truth are errors.

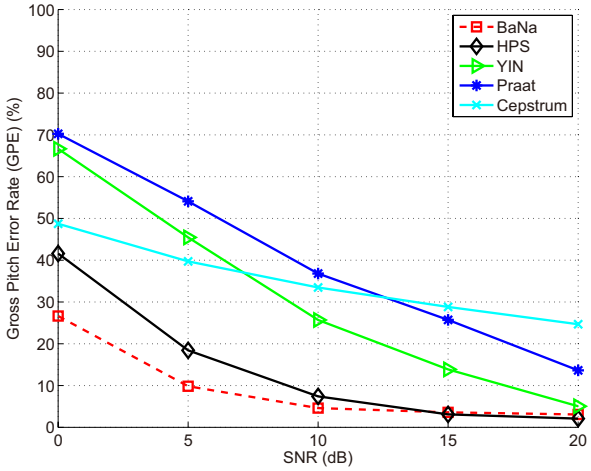


Fig. 13: GPE rate of the different algorithms for a piece of violin music with eight types of noise. Detected F_0 deviating more than 3% from ground truth are errors.

without the Cepstrum candidate, BaNa without the lowest frequency candidate, BaNa without both added candidates, and BaNa without post-processing for the LDC database. BaNa without post-processing means that we choose the F_0 candidate with the highest confidence score to be F_0 for each frame. We can see that using the two added candidates and using post-processing are effective to reduce the GPE rate. We can see that the GPE rate is as high as 20% when SNR is 20 dB without using both added candidates. This is because for some frames, only the F_0 peak's amplitude is high enough to be detected. Therefore, no F_0 candidates are derived from calculating frequency ratios.

By comparing the results for BaNa without post-processing with the results in Fig. 11 for the two algorithms that have no post-processing, HPS and Cepstrum, with the results in Fig. 5, we can see that BaNa without post-processing still achieves a lower GPE rate. Thus, from the breakdown analysis we conclude that the post-processing is helpful, but it is not the most critical step in determining the performance of BaNa.

VI. BaNa F_0 Detection Algorithm for Music

In this section, we extend the BaNa algorithm to enable F_0 detection of music signals in noisy environments.

A. Modifications on BaNa for F_0 detection for music

Since speech and music have different frequency characteristics, the BaNa algorithm needs to be slightly modified for F_0 detection in music. In Section III-B, when detecting F_0 for speech, the p peaks with the lowest frequencies are selected. However, music signals can have high F_0 values, thus the low frequency region can be dominated by noise peaks. Thus, if we still choose the p peaks with the lowest frequencies, noise peaks are chosen incorrectly. Therefore, for music F_0 detection, we select the p peaks with the highest amplitudes in the frequency range considered. We show the benefit of this change in Section VI-D.

B. Experimental settings for F_0 detection for music

Due to the variety of spectrum characteristics for different musical instruments, to show the performance of the F_0 detection algorithms for musical instruments, samples from four instruments are used: violin, trumpet, clarinet and piano. These music pieces are selected and downloaded from [61], which were all recorded in a quiet environment. These music pieces include a piece of 3.7 s long violin with 9 notes, a piece of 12.9 s long trumpet with 12 notes, a piece of 5.3 s long clarinet with 4 notes, and a piece of 7.8 s long piano with 8 notes. All the music samples used are also included in the BaNa toolkit [44]. The additive noise is from the same noise database as in Section IV-A.

For F_0 detection in music, we use hand-labeled ground truth F_0 values, which are determined by manually inspecting the spectrum and locating the F_0 peaks for each frame. Due to the large F_0 range in music, we use a more stringent F_0 deviation criteria for error measurement. The difference between two neighboring key frequencies is $2^{\frac{1}{12}}$, which is approximately 6%. Thus, we use half of this number, i.e., 3%, as the F_0 deviation criteria, which is also called the musical quarter tone [62]. Thus, detected F_0 values that deviate more than 3% from the ground truth values are counted as errors. This error measurement metric is also used by other studies [62].

C. Parameter tuning

According to the music F_0 range specified in [15], the lower and the upper limit for F_0 of music are set to F_0^{min} = 50 Hz and F_0^{max} = 4,000 Hz, respectively. It is set to 50-4,000 Hz for these competing algorithms as well for a fair comparison. The other parameters are the same as those in Table II, and are not further optimized using music signals.

D. BaNa vs. BaNa music

To show the effectiveness of the changes made to the BaNa algorithm to be suitable for F_0 detection in music, we plot the

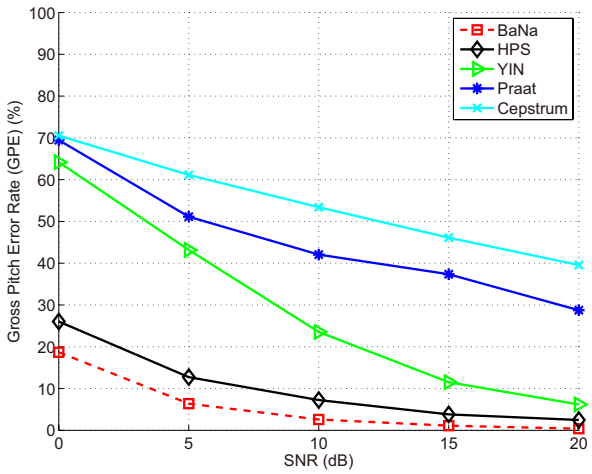


Fig. 14: GPE rate of the different algorithms for a piece of trumpet music with eight types of noise. Detected F_0 deviating more than 3% from ground truth are errors.

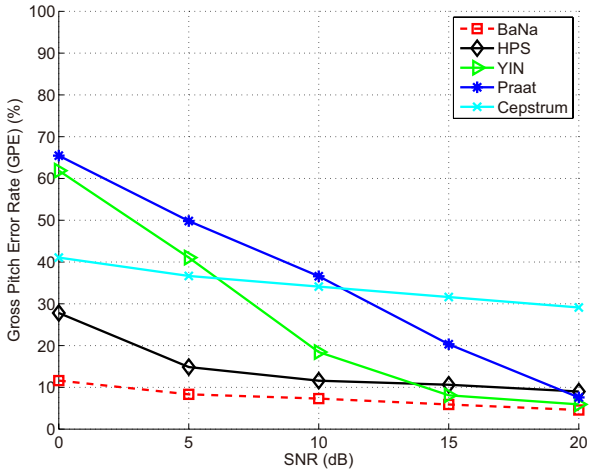


Fig. 15: GPE rate of the different algorithms for a piece of clarinet music with eight types of noise. Detected F_0 deviating more than 3% from ground truth are errors.

GPE rate in Fig. 12 for a piece of violin music using both the original BaNa algorithm and the customized BaNa music algorithm with eight different types of noise at 0 dB SNR. The F_0 detection range is set to be the same for the original BaNa algorithm and the customized BaNa music algorithm, i.e., F_0^{min} = 50 Hz and F_0^{max} = 4,000 Hz. We can see that the modifications in the BaNa algorithm for music F_0 detection are necessary, and can greatly reduce the GPE rate for almost all types of noisy music. Note that throughout this section, we just use 'BaNa' to represent the BaNa music algorithm.

E. F_0 detection performance for music signals

In this set of experiments, we compare the BaNa algorithm with other algorithms for music F_0 detection. Within the evaluations of the SAFE algorithm in [31], there are no detection results for music. Therefore, we are not able to run the SAFE algorithm here due to the lack of noisy music training data. Also, according to the authors of PEFAC [27],

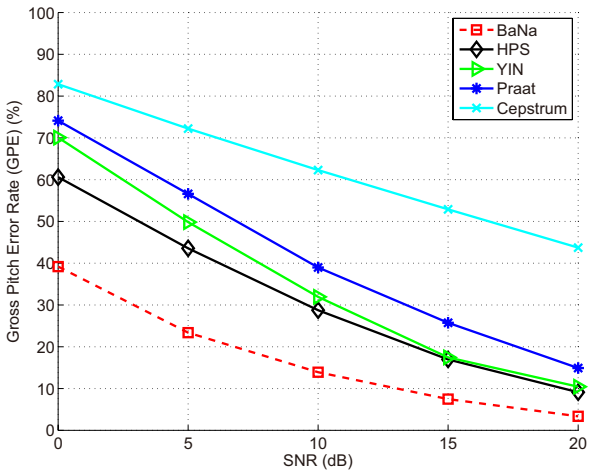


Fig. 16: GPE rate of the different algorithms for a piece of piano music with eight types of noise. Detected F_0 deviating more than 3% from ground truth are errors.

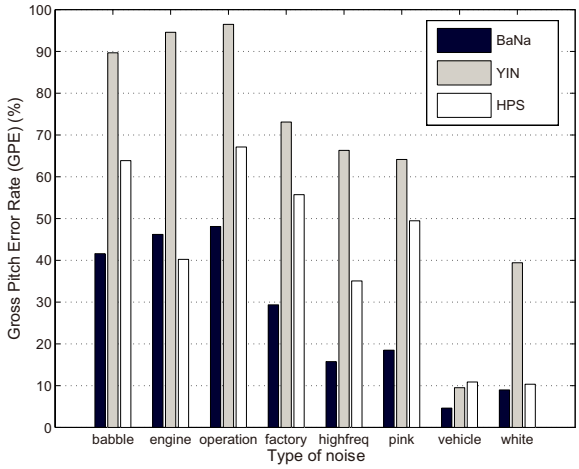


Fig. 17: GPE rate of BaNa, YIN and HPS for a piece of violin music with eight types of noise at 0 dB SNR. Detected F_0 deviating more than 3% from ground truth are errors.

PEFAC is not suitable for F_0 detection in music, hence we do not include that here. Also, it is unclear how to use the code for the Wu algorithm [30] to process long audio samples. Therefore, we only compare the proposed BaNa algorithm with YIN, HPS, Praat, and Cepstrum. Figs. 13-16 show the GPE rates of the different algorithms for violin, trumpet, clarinet, and piano, respectively, averaged over the eight types of noise. Results on all these four instruments show that the BaNa algorithm achieves the lowest GPE rate among all the algorithms. At 0 dB SNR, BaNa achieves the lowest GPE rates, which are 36.1%, 28.1%, 58.3%, and 35.3% lower than the closest performing algorithm, HPS, for violin, trumpet, clarinet, and piano, respectively.

From the above results, we can see that BaNa, HPS, and YIN provide the overall best F_0 detection performance in noisy music. Praat and Cepstrum do not provide consistent or satisfying results. Therefore, we choose BaNa, YIN, and HPS for detailed comparison using the violin piece with eight

TABLE III: Elapsed time (in seconds) for F_0 detection using the BaNa algorithm implemented on an Android platform with a different number of threads and FFT sizes. The speech file is 1.3 s long.

Number of threads	FFT size			
	2^{16}	2^{15}	2^{14}	2^{13}
1	11.05	5.16	2.52	1.42
2	6.85	3.15	1.49	0.85
3	5.93	2.67	1.28	0.92
4	5.89	2.67	1.25	0.80

different types of noise at 0 dB SNR. In Fig. 17 we can see that BaNa has the lowest GPE rate for seven out of eight types of noise, especially for the speech babble noise.

VII. Implementation Issues

With an increasing number of speech-related smartphone apps emerging in the market, and due to the fact that speech captured by smartphones are usually affected by different types of noise, it is important to discuss the challenges in implementing the BaNa F_0 detection algorithm on a mobile platform. To explore these issues, we implemented BaNa as an app on an Android platform[1]. Since the F_0 candidates and their confidence scores can be calculated separately for each frame, as explained in Section III-B, we can take advantage of multithreading to speed up the implementation. Single-core and multi-core devices can both benefit from multithreading through an increased utilization of the processor(s). When all threads finish the calculation of F_0 candidates for their own assigned frames, the Viterbi post-processing can go through all the frames to determine F_0 for each frame.

To test the speed of the BaNa F_0 detection implementation, we ran tests with different parameter settings and speech sample lengths on a Google Nexus 7. The specs of the device are: Nvidia Tegra 3 quad-core processor clocked at 1.2GHz, 1GB of RAM. Of course, the speed of the algorithm highly depends on the capabilities of the mobile device. Table III shows the elapsed time to process a 1.3 s long speech sample with sampling rate of 22,050 Hz. All the parameters for the BaNa algorithm are set to be the same as those in Table II. For a more reliable measurement, the elapsed time for each test is averaged over 10 trials. We can see that the BaNa F_0 detection algorithm runs roughly 8 times faster by using the 2^{13} FFT size than using the 2^{16} FFT size, though using the 2^{13} FFT size still provides a reasonable frequency resolution of $22,050/2^{13} = 2.7$ Hz per sample. Also, we can see that multithreading helps to further reduce the elapsed time.

We show in Table IV the elapsed time for F_0 detection for speech samples with different lengths. For this test, we choose the setting that provides the fastest speed, i.e., the number of threads is set to 4, and the FFT size is set to 2^{13}. These results show the possibility to turn the BaNa algorithm into a real-time F_0 detector even on mobile devices.

VIII. Conclusions and Future Work

In this paper, we presented BaNa, a noise resilient hybrid F_0 detection algorithm for speech and music. BaNa was designed to detect F_0 in noisy environments, for example on a smartphone. This would enable the wide deployment of speech-based applications, such as the ones that use emotion detection. Evaluations show that BaNa achieves the lowest GPE rate for most cases among the algorithms investigated from the literature including YIN, HPS, Praat, Cepstrum, PEFAC, SAFE and Wu for different types of background noise, and under different SNR levels from -10 dB to 20 dB. Even for the very noisy scenario of 0 dB SNR, the GPE rate of BaNa averaged over all types of noise is only about 20% to 35% for speech for the different databases evaluated. The GPE rate for music at 0 dB SNR is 12% to 39% for different instrument pieces. Additionally, we implemented the BaNa algorithm on an Android platform, and implementation issues such as delay and multithreading are discussed. Tests on a real device show that the implementation is fast enough to provide for real-time F_0 detection applications in the future.

[1]Code for the Android implementation of BaNa is available at [44].

TABLE IV: Elapsed time (in seconds) for F_0 detection using the BaNa algorithm implemented on an Android platform for speech samples with different lengths.

Number of threads	FFT size	Length of speech sample (s)				
		2	4	6	8	10
4	2^{13}	0.91	1.61	2.39	3.05	3.82

Acknowledgment

The authors would like to thank Dr. Zhiyao Duan for his comments on this paper, and Thomás Horta for his work implementing BaNa on the Android platform.

References

[1] B. Cardozo and R. Ritsma, "On the perception of imperfect periodicity," *Audio and Electroacoustics, IEEE Trans. on*, vol. 16, no. 2, pp. 159 – 164, 1968.

[2] J. H. Jeon, W. Wang, and Y. Liu, "N-best rescoring based on pitch-accent patterns," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, 2011, pp. 732–741.

[3] C. Wang, "Prosodic modeling for improved speech recognition and understanding," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[4] Z.-H. Ling, Z.-G. Wang, and L.-R. Dai, "Statistical modeling of syllable-level F0 features for hmm-based unit selection speech synthesis," in *Proceedings of International Symposium on Chinese Spoken Language Processing*, 2010, pp. 144–147.

[5] S. Sakai and J. Glass, "Fundamental frequency modeling for corpus-based speech synthesis based on a statistical learning technique," in *Proc. of IEEE ASRU*, 2003, pp. 712–717.

[6] J. Woodruff and D. L. Wang, "Binaural detection, localization, and segregation in reverberant environments based on joint pitch and azimuth cues," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 21, pp. 806–815, 2013.

[7] O. W. Kwon, K. Chan, J. Hao, and T. W. Lee, "Emotion recognition by speech signals," in *Eighth European Conference on Speech Communication and Technology*, 2003.

[8] F. Al Machot, A. H. Mosa, K. Dabbour, A. Fasih, C. Schwarzlmuller, M. Ali, and K. Kyamakya, "A novel real-time emotion detection system from audio streams based on bayesian quadratic discriminate classifier for adas," in *Nonlinear Dynamics and Synchronization 16th Intl Symposium on Theoretical Electrical Engineering*, 2011.

[9] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas, "EmotionSense: a mobile phones based adaptive platform for experimental social psychology research," in *Proceedings of the 12th int. conference on Ubiquitous computing*, 2010, pp. 281–290.

[10] K. Chang, D. Fisher, and J. Canny, "AMMON: A Speech Analysis Library for Analyzing Affect, Stress, and Mental Health on Mobile Phones," in *2nd Intl Workshop on Sensing Applications on Mobile Phones*, 2011.

[11] Y. Yang, C. Fairbairn, and J. F. Cohn, "Detecting depression severity from vocal prosody," *Affective Computing, IEEE Trans. on*, vol. 4, no. 2, pp. 142–150, 2013.

[12] J. P. Bello, G. Monti, and M. Sandler, "Techniques for automatic music transcription," in *Intl Symposium on Music Information Retrieval*, 2000, pp. 23–25.

[13] M. Antonelli, A. Rizzi, and G. Del Vescovo, "A query by humming system for music information retrieval," in *Intelligent Systems Design and Applications (ISDA), 10th Intl Conference on*, 2010, pp. 586 – 591.

[14] S. Kim, E. Unal, and S. Narayanan, "Music fingerprint extraction for classical music cover song identification," in *Multimedia and Expo, 2008 IEEE Intl. Conference on*, 2008, pp. 1261 –1264.

[15] P. Cariani, "Neural Representation of Musical Pitch - MIT OpenCourseWare," http://ocw.mit.edu/courses/health-sciences-and-technology/hst-725-music-perception-and-cognition-spring-2009/lecture-notes/MITHST_725S09_lec04_pitch.pdf, 2009.

[16] A. M. Noll, "Cepstrum pitch determination," *Journal of the Acoustical Society of America*, vol. 41, pp. 293–309, 1967.

[17] M. R. Schroeder, "Period histogram and product spectrum: New methods for fundamental frequency measurement," *Journal of the Acoustical Society of America*, vol. 43, pp. 829–834, 1968.

[18] P. Boersma, "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound," in *Proceedings of the Institute of Phonetic Sciences 17*, 1993, pp. 97–110.

[19] H. Ba, N. Yang, I. Demirkol, and W. Heinzelman, "BaNa: A hybrid approach for noise resilient pitch detection," in *IEEE Workshop on Statistical Signal Processing*, 2012, pp. 369 –372.

[20] M. J. Ross, H. L. Shaffer, A. Cohen, R. Freudberg, and H. J. Manley, "Average magnitude difference function pitch extractor," *Audio, Speech, and Language Processing, IEEE Trans. on*, pp. 353 –362, 1974.

[21] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *Journal of the Acoustical Society of America*, vol. 111, pp. 1917–1930, 2002.

[22] J. Liu, T. F. Zheng, J. Deng, and W. Wu, "Real-time pitch tracking based on combined SMDSF," in *Proc. of Interspeech*, 2005, pp. 301–304.

[23] D. Talkin, "A robust algorithm for pitch tracking (RAPT)," *Speech Coding and Synthesis*, 1995.

[24] L. R. Rabiner and R. W. Schafer, *Theory and Application of Digital Speech Processing*. Pearson, 2011.

[25] T. W. Parsons, "Separation of speech from interfering speech by means of harmonic selection," *Journal of the Acoustical Society of America*, vol. 60, pp. 911–918, 1976.

[26] X. Chen and R. Liu, "Multiple pitch estimation based on modified harmonic product spectrum," in *Proceedings of Intl Conference on Information Technology and Software Engineering*, 2013, pp. 271–279.

[27] S. Gonzalez and M. Brookes, "A pitch estimation filter robust to high levels of noise (PEFAC)," in *Proc. European Signal Processing Conf., Barcelona, Spain*, 2011.

[28] Z. Jin and D. Wang, "HMM-based multipitch tracking for noisy and reverberant speech," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 19, no. 5, pp. 1091–1102, 2011.

[29] F. Huang and T. Lee, "Pitch estimation in noisy speech using accumulated peak spectrum and sparse estimation technique," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 21, no. 1, pp. 99 –109, 2013.

[30] M. Wu, D. Wang, and G. J. Brown, "A multipitch tracking algorithm for noisy speech," *Speech and Audio Processing, IEEE Trans. on*, vol. 11, no. 3, pp. 229–241, 2003.

[31] W. Chu and A. Alwan, "SAFE: A statistical approach to F0 estimation under clean and noisy conditions," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 20, no. 3, pp. 933 –944, 2012.

[32] A. von dem Knesebeck and U. Zölzer, "Comparison of pitch trackers for real-time guitar effects," in *Proc. of the 13th Int. Conference on Digital Audio Effects*, 2010.

[33] P. De La Cuadra, A. Master, and C. Sapp, "Efficient pitch detection techniques for interactive music," in *Proceedings of the Int. Computer Music Conference, La Habana*, 2001.

[34] Thomas O'Haver, Command-line findpeaks MATLAB function, http://terpconnect.umd.edu/~toh/spectrum.

[35] P. van Alphen and D. van Bergem, "Markov models and their application in speech recognition," in *Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam*, 1989, pp. 1–26.

[36] D. Iskra, B. Grosskopf, K. Marasek, H. van den Huevel, F. Diehl, and A. Kiessling, "SPEECON - speech databases for consumer devices: Database specification and validation," in *Proceedings of International Conference on Language Resources and Evaluation (LREC)*, 2002, pp. 329–333.

[37] B. Kotnik, H. Höge, and Z. Kacic, "Evaluation of pitch detection algorithms in adverse conditions," in *Proceedings of the Third International Conference on Speech Prosody*, 2006, pp. 149–152.

[38] I. Luengo, I. Saratxaga, E. Navas, I. Hernáez, J. Sanchez, and I. naki Sainz, "Evaluation of pitch detection algorithms under real conditions," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2007, pp. 1057–1060.

[39] G. Seshadri and B. Yegnanarayana, "Performance of an event-based instantaneous fundamental frequency estimator for distant speech signals," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 19, pp. 1853–1864, 2011.

[40] "Emotional prosody speech and transcripts database from Linguistic Data Consortium (LDC)," http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2002S28.

[41] "CMU Arctic Database," http://www.festvox.org/cmu_arctic/.

[42] P. C. Bagshaw, S. M. Hiller, and M. A. Jack, "Enhanced pitch tracking and the processing of F0 contours for computer aided intonation teaching," in *Proceedings of Eurospeech*, 1993, pp. 1003–1006.

[43] F. Plante, G. Meyer, and W. A. Ainsworth, "A pitch extraction reference database," in *Proceedings of Eurospeech*, 1995, pp. 837–840.

[44] "Generated noisy speech data and BaNa source code, WCNG website," http://www.ece.rochester.edu/projects/wcng/project_bridge.html.

[45] A. P. Varga, H. J. M. Steeneken, M. Tomlinson, and D. Jones, "NOISEX-92 study on the effect of additive noise on automatic speech recognition," http://spib.ece.rice.edu/spib/data/signals/noise/, 1992.

[46] L. R. Rabiner, M. J. Cheng, A. E. Osenberg, and C. A. McGonegal, "A comparative performance study of several pitch detection algorithms," *Acoustics, Speech, and Signal Processing, IEEE Trans. on*, vol. 24, no. 5, pp. 399 – 418, October 1976.

[47] M. Wohlmayr, M. Stark, and F. Pernkopf, "A probabilistic interaction model for multipitch tracking with factorial hidden Markov models," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 19, no. 4, pp. 799–810, 2011.

[48] O. Babacan, T. Drugman, N. d'Alessandro, N. Henrich, and T. Dutoit, "A comparative study of pitch extraction algorithms on a large variety of singing sounds," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013, pp. 7815–7819.

[49] "Download page for the SAFE toolkit," http://www.ee.ucla.edu/~weichu/safe/.

[50] X. Huang, A. Acero, and H.-W. Hon, *Spoken language processing*. Prentice Hall PTR New Jersey, 2001, vol. 15.

[51] "Background noise samples used to construct the AURORA noisy speech database," http://www.ee.columbia.edu/~dpwe/sounds/noise/.

[52] D. Pearce, H.-G. Hirsch, and E. E. D. Gmbh, "The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *ISCA ITRW ASR2000*, 2000, pp. 29–32.

[53] "Source code for the YIN algorithm," http://audition.ens.fr/adc/.

[54] "Source code for the Praat algorithm," http://www.fon.hum.uva.nl/praat/.

[55] "Source code for the PEFAC algorithm included in the VOICEBOX toolkit," http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html.

[56] "Source code for the Wu algorithm," http://www.cse.ohio-state.edu/pnl/software.html.

[57] M. G. Christensen and A. Jakobsson, *Multi-Pitch Estimation*. Morgan & Claypool Publishers, 2009.

[58] S. A. Raczynski, E. Vincent, and S. Sagayama, "Separation of speech from interfering speech by means of harmonic selection," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 21, pp. 1830–1840, 2013.

[59] M. Wu, D. L. Wang, and G. J. Brown, "A multipitch tracking algorithm for noisy speech," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 11, pp. 229–241, 2003.

[60] J. Wu, E. Vincent, S. A. Raczynski, T. Nishimoto, N. Ono, and S. Sagayama, "Polyphonic pitch estimation and instrument identification by joint modeling of sustained and attack sounds," *IEEE Journal of Selected Topics in Signal Process.*, vol. 5, pp. 1124–1132, 2011.

[61] "Freesound website for short pieces of music download," http://www.freesound.org/.

[62] Z. Duan, B. Pardo, and C. Zhang, "Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 18, no. 8, pp. 2121–2133, 2010.

Na Yang Na Yang is a Ph.D. student in the Department of Electrical and Computer Engineering at the University of Rochester. She received her B.S. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2009, and her M.S. degree from the University of Rochester, Rochester, NY, in 2011. Her research interests lie primarily in the interdisciplinary area of signal processing and machine learning towards affective computing applications and the broad human-computer interaction area. She is currently interested in noise-resilient designs for voice-based applications on mobile platforms.

He Ba He Ba is a Ph.D. student in Electrical and Computer Engineering at the University of Rochester. He received his B.S. degree from the Department of Electrical Engineering at Beijing Institute of Technology in 2008 and his M.S. degree from the Electrical and Computer Engineering department at the University of Rochester in 2011. His research interests lie in the areas of wireless communications, mobile computing and digital signal processing.

Weiyang Cai Weiyang Cai earned his M.S. degree in Electrical Engineering from the University of Rochester in 2013. His thesis focused on noise-resilient fundamental frequency detection algorithms for speech and music. He earned his B.S. degree in Electronic Engineering from the Hong Kong University of Science and Technology in 2011, where he was keen on developing automatic music transcription tools. From 2012 to 2013, he was a research assistant in the Wireless Communication and Networking Group, University of Rochester. He is currently a software engineer at A10 Networks, San Jose, CA.

Ilker Demirkol Ilker Demirkol is a Senior Researcher in Telematics Engineering at Universitat Politecnica de Catalunya, where he works on wireless networks, including wireless mesh, ad hoc and sensor networks, along with different computer engineering related topics. His recent research targets communication protocol development for aforementioned networks, along with performance evaluation and optimization of such systems.

Demirkol received his B.Sc., M.Sc., and Ph.D. degrees in Computer Engineering from the Bogazici University, Istanbul, Turkey. Over the years, he has worked in a number of research laboratories and corporations holding positions such as Network Engineer, System and Database Consultant, and Lecturer.

Wendi B. Heinzelman Wendi B. Heinzelman is a Professor in the Department of Electrical and Computer Engineering at the University of Rochester, and she holds a secondary appointment in the Computer Science Department at Rochester. She also currently serves as the Dean of Graduate Studies for Arts, Sciences & Engineering. Dr. Heinzelman received a B.S. degree in Electrical Engineering from Cornell University in 1995 and M.S. and Ph.D. degrees in Electrical Engineering and Computer Science from MIT in 1997 and 2000, respectively. Her current research interests lie in the area of wireless communications and networking, mobile computing, and multimedia communication. She is a member of Networking Networking Women (N^2 Women) and the Society of Women Engineers (SWE), a Distinguished Scientist of ACM Sigmobile, and a Senior Member of the IEEE Communications Society and the IEEE Signal Processing Society.