An investigation into several pitch detection algorithms for singing phrases analysis

Behnam Faghih
Dept of Computer Science,
Maynooth University,
Maynooth, Co. Kildare, Ireland
Behnam.Faghih@mu.ie

Joseph Timoney
Dept of Computer Science,
Maynooth University,
Maynooth, Co. Kildare, Ireland
Joseph.Timoney@mu.ie

***Abstract*— This article provides an investigation into four different pitch detection algorithms: pYin, Praat, Phase Lock Loops (PLL)-based, and an Extended Complex Kalman Filter approach. The first two algorithms compute the pitch on a frame-by-frame basis while the latter two work on a sample-by-sample basis. Only in recent years has there been a noticeable increase in the number of papers applying pitch detection techniques to sung phrases. This investigation is done on a dataset contained 76 files of singers. To create a ground truth from the data an alternative approach using the *Spear* analysis program is applied. The algorithms are compared using a new *Singing Data Analyser tool*. It was observed that the pYin and Praat are the most reliable algorithms while the PLL and Kalman filter algorithms are very dependent on the user-selected parameters.**

Keywords—Pitch detection, singing analysis, PLL, Kalman filter

I. INTRODUCTION

The estimation of the fundamental frequency (F0) of a waveform is known in the literature as the problem of pitch detection. This has been a long-standing task in signal processing and many different algorithms have been proposed over the years. Up until about 20 years ago, the problem of monophonic pitch detection only was considered but since then the much more difficult task of polyphonic pitch detection has been tackled. Although some sample-by-sample detection methods have been proposed, most algorithms first separate the audio signal in short frames, generally of the order of 15-35ms in length, within which it is assumed that the frequency information is stationary. The pitch is then computed for each frame. The analysis is done either in the time domain, using an algorithm that relies on computing the autocorrelation function or a variant, or in the frequency domain, using an algorithm that applies a type of Fourier transform. The initial algorithm outputs are assumed to be raw estimates that require post-processing. It is this later stage that can really differentiate the effectiveness of an approach.

The idea behind using the autocorrelation function for a time-domain algorithm is that when applying this function to a waveform it should produce a representation that shows significant peaks at positions related to the period of the waveform, with the largest peak occurring first. A well-known variant is the Average Magnitude Different function, which was introduced as a computationally efficient alternative to the autocorrelation function. In more recent times, the Yin algorithm [1] has become very popular and it uses a related cumulative mean normalized difference function. To enhance the accuracy of the estimate around the detected peak in the computed time-domain function some form of interpolation is required, for example, parabolic interpolation.

For frequency-based methods, the frame is transformed into the frequency domain, often using the Fourier transform. An early algorithm implemented a further transformation of the Fourier spectrum into what was termed the 'Cepstrum' [2], essentially dividing the spectrum into a fast-varying (because of pitch harmonics) and slowly-varying components (because of the spectral envelope). Isolating the fast-varying component facilitated a pitch estimate. Another technique was the Harmonic Product Spectrum [3]. This attempts to emphasize the harmonic peak of the fundamental component in the spectrum by a succession of decimations of the original spectral representation and then adding them together. More recent algorithms use a template approach where the spectral representation is compared with a template of the known fundamental frequency. The one with the best match signifies the pitch.

For both these techniques, they benefit from a tracking stage that follows. All algorithms can produce incorrect pitch estimates, particularly if the second harmonic or a subharmonic is too strong, leading to the problem of octave-doubling or octave-halving, or if the harmonicity of the signal is weak leading to erroneous values. Thus, the pitch estimates need to be tracked and refined to remove any unexpected jumps from a 'smooth' contour. Tracking can be done in a forward manner, that is, as the estimates are produced and determining how well they fit with previous values. It can also be done in a backward manner, using an algorithm such as dynamic programming, where estimates are obtained from the start to the end of the signal, and then the best possible contour is traced out from the end to the beginning. A good example of this is the pitch detector in the Pratt software package [4]. Another recent approach is pYin [5], which has included a Hidden Markov Model (HMM) with the Yin algorithm[1]. The HMM uses the Viterbi algorithm which is a dynamic programming technique.

More recently sample-by-sample methods for pitch detection have appeared. This obviates the need for explicit tracking following the estimation as it is built in to the algorithm. These use techniques from other areas of signal processing, that is the Phase-locked loop (PLL) [6], a communications tool, and the Extended Kalman filter [7], more familiar in statistical signal detection. These take as input the audio signal and provide a value for the pitch at every sample. The initial PLL method was augmented to have a set of PLLs to track the pitch and the most likely pitch value selected [6]. The Extended Kalman Filter can produce good

978-1-7281-2800-9/19/$31.00 (c) 2019 IEEE

results according to [7] but care is required when setting the parameters of the signal model. Another recent work is the Harmonic locking Loop [8], which extends the tracking idea to all harmonics to produce an improved estimate. The difficulty again is that parameter values need to be set.

Once the pitch contour is found the next stage is to convert this into a melodic representation. In the case of singing, it has to be recognized that singers use many techniques such as portamento and vibrato in their style so a true description needs to retain these qualities [9]–[11].

It is noteworthy that PLL and Kalman Filter have been evaluated only with one instrument and no a human's voice, PLL with cello [6] and Kalman Filter with guitar [7]. Therefore, this study evaluates their performance for humans' voice

The next section will introduce the dataset to be analyzed and explain the difficulties associated with pitch detection of these files. It will also detail the software framework for the analysis. Lastly, it will describe and justify the evaluation criteria. This is followed by a section that will explain the pitch detection algorithms and their implementations, including the importance of the parameters associated with some of these. The section after this will provide the results and explain the performance of each the algorithms. Graphs will be given to illustrate. The final section will be the conclusion and will discuss some aspects of interest for future investigation.

II. Methodology

A. Dataset

We used the VocalSet dataset [12] which is a singing voice dataset consisting of more than 10 hours of monophonic recorded audio of professional singers demonstrating both standard and extended vocal techniques on all 5 vowels. VocalSet contains recordings from 20 different singers (11 males and 9 females) with a range of voice types. VocalSet not only has the full set of vowels, but also a diverse set of voices on many different vocal techniques, sung in contexts of scales, arpeggios, long tones, and excerpts. For this study, we selected the C scale and arpeggios performance of 10 males and 9 females in both fast-forte and slow-forte; in other words, the musical material is the same and is of a loud volume (forte) but in one case it is sung at a quick tempo (fast) which in the second case the tempo is much slower (slow). Therefore, the total number of our files used was 76.

B. Tools

With the natural singing files, there is no accompanying file containing the exact musical pitches that are being sung and the times at which they are sung at. To be able to assess the accuracy of the pitch detection algorithms therefore, such an extension to the dataset must be made. It was considered that one possible way to achieve this is to use another signal decomposition tool that will facilitate the isolation of the fundamental component only from which an accurate pitch track can be obtained. This tool must be operated manually and the fundamental identified visually. The *Spear* tool [13] was discovered as being suitable for generating this ground truth. The Spear tool performs a frame-by-frame sinusoidal analysis [14], identifies all the important peaks in each frame, and connects together peaks that exhibit a trajectory. Additionally, the Spear tool provides both a visualization of all the important frequency components in an audio file, and a means by which they can be edited and removed. Thus, all unwanted components except the fundamental can be deleted, and a ground truth can be achieved. In Fig. 1, two screenshots of the visualization of Spear are given. In the above part of this figure, the highlighted line, red line, is the base pitch and any other lines in black and grey are the harmonics. The strength of a component is indicated by the colour, varying from grey to black illustrating weak to strong. By manually finding the fundamental all other components can be deleted, and a frequency-varying sinusoid with respect to the fundamental frequency can be resynthesized, In the lower panel in Fig. 1 the isolated fundamental is shown. On all occasions, it was found to be straightforward to select the base pitch among the components.

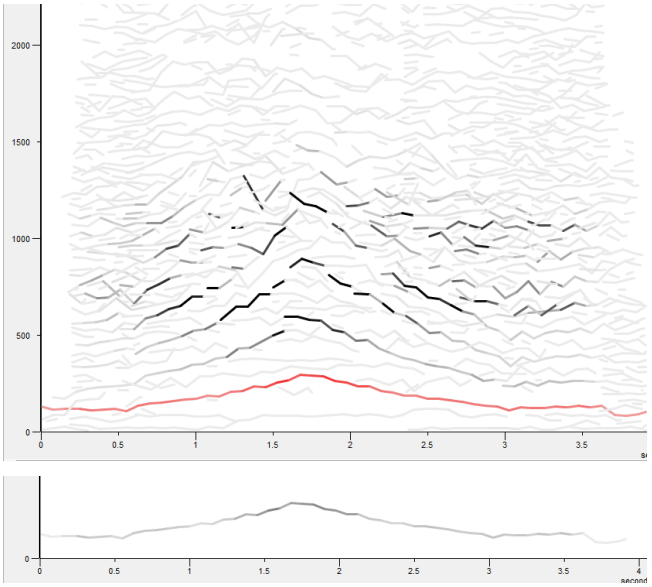

Fig. 1. Example of the interface for Spear with the fundamental highlighted in red. In general, strong components are black and weak ones are grey.

III. Pitch Detection Algorithms

Four well-known pitch detection algorithms (pYin, Praat, PLL-Based, and Kalman filter) were selected for this study. Different tools were employed to implement these algorithms.

The *Tony tool* [15] was used to make an analysis as it contains an implementation of the pYin algorithm [5]. One of the useful features of this tool is that once the pitches have been detected, all the estimated frequencies can be saved into a text file. The pYin algorithm is based on YIN algorithm and its approach is to consider multiple candidates for the pitch based on a probabilistic interpretation of the earlier YIN [1] pitch detection algorithm . A HMM is used to produce the final pitch track from the estimates [5].

The *Praat tool* [4] is employed to analyse the dataset based on the Boersma algorithm [16]. This uses an autocorrelation approach followed by dynamic programming to find the best path among a set of pitch candidates. All the estimated frequencies can be saved into a text file.

Matlab is used to implement Extended Complex Kalman Filter (ECKF) [7] and the PLL-Based pitch detection [6] algorithms.

The ECKF algorithm operates on a sample-by-sample basis. This algorithm is based on Kalman filter which is a well-known approach to tracking parameters in noise.

However, in this case of the ECKF there is a nonlinear relationship between the changing state (pitch value) and the observation (the output waveform). The iterative nature of the algorithm means that at the beginning of a sound it has difficulty to estimate the pitch, but after a while the parameter values adapt to give better estimates.

Similarly, the PLL-based pitch detection provides a sample-by-sample instantaneous pitch estimation. Its basic operation is to lock its internal oscillator to the input signal in such a way so as to minimize the error in phase between the two [6]. PLLs are a common tool in communication applications and only recently have found application to audio pitch detection.

One of the difficulties in implementing the PLL and Kalman algorithms is finding the best values for their parameters. In this study, in order to find the best value for their parameters, after selecting a wave file from the dataset, testing loops with small step sizes were used to create values for the parameters across their possible ranges. Many files were generated to assess the different values for parameters. Then, by plotting the data, the values for the parameters the can work well with the selected sound file can be ascertained. Consequently, those parameters were then applied for the whole dataset and the estimated pitch frequencies were saved

A *Singing Data Analyser tool*, which was written in C#. was created by the authors of this paper, was used to manage the testing of these four pitch detection algorithms. The inputs of this software tool are the text files generated by the Tony, Praat, and Matlab software. After preparing the format of the input text files this tool facilitated plotting of the results.

IV. RESULTS

Praat and pYin worked well with all items of the dataset, as highlighted from tables I and II, without any incorrect pitch estimation. In these tables, oct means octave-doubling problem and inc means any other incorrect estimation other than octave-doubling. On the other hand, the performance of the PLL and Kalman were not promising because from all the 76 files, only the pitch contours from 16 files and 48 files respectively were found correctly by the PLL and Kalman algorithms. In addition, it can be observed that the Kalman algorithm worked better for female voices rather than male voices, with 70 correct estimations for females as compared to 54 for the male performances. In order to find the reason for this issue, one of the men's voices was selected to find the best algorithmic parameters for that file. After that, these new algorithmic parameters were applied to the files with all the men' voices, but then this implementation resulted in more inaccurate pitch values as compared to previously. The sensitivity of the ECKF algorithm was clear.

Moreover, in both PLL and Kalman implementations there were more problems observed than just octave doubling: it was often trebling or more, either due to an octave overestimation or higher harmonic tracking. An example of octave doubling is shown in Fig. 2(a) and other problems in Fig. 2(b). However, the detected contours for the pYin and Praat algorithms are overlaid with the ground truth contour and are thus hidden by it because they are the same except in a few instances that can be seen in Fig. 2 (b) for the pYin around time 3.75 sec and 5.5 sec.

TABLE I. THE NUMBER OF INCORRECT INSTANCES OF F0 DETERMINATION IN THE PITCH DETECTION ALGORITHMS FOR THE FAST-FORTE DATA

	Female				*Male*			
	Scale		*Arpeggios*		*Scale*		*Arpeggios*	
	oct[a]	*inc*[b]	*oct*	*inc*	*Oct*	*inc*	*Oct*	*inc*
pYin	0	0	0	0	0	0	0	0
PLL	0	4	0	5	4	6	2	8
Kalman	2	0	2	1	4	3	5	2
Praat	0	0	0	0	0	0	0	0

[a] oct = Octave-doubling
[b] inc = incorrect

TABLE II. THE NUMBER OF INCORRECT INSTANCES OF F0 DETERMINATION IN THE PITCH DETECTION ALGORITHMS FOR THE SLOW-FORTE DATA

	Female				*Male*			
	Scale		*Arpeggios*		*Scale*		*Arpeggios*	
	Oct[a]	*inc*[b]	*oct*	*inc*	*Oct*	*inc*	*Oct*	*inc*
pYin	0	0	0	0	0	0	0	0
PLL	3	3	2	4	3	6	4	6
Kalman	0	0	0	1	2	1	2	3
Praat	0	0	0	0	0	0	0	0

[a] oct = Octave-doubling
[b] inc = incorrect

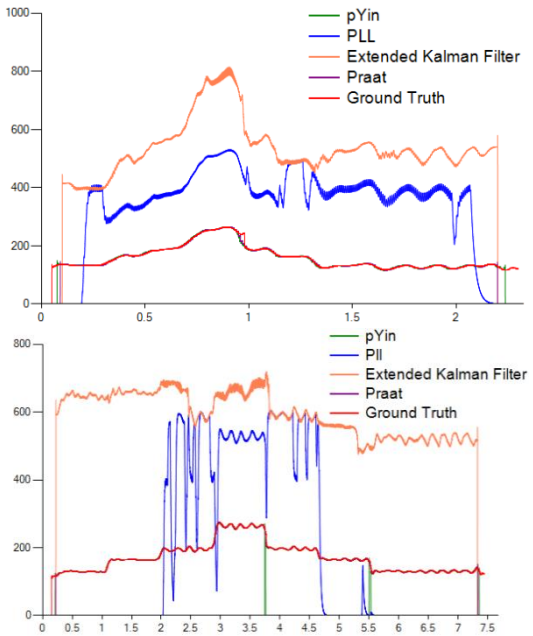

Fig. 2. Example outputs of the various pitch detection algorithms

Other observation was that since Kalman filter and the PLL are real-time algorithms that detect the pitch on a sample-by-sample basis, they will be a settling time before they start to track correctly, as can be seen in Fig. 3. However, the pYin and Praat algorithms have some pre-processing to get everything into line so it will force errant values to confirm to the contour it has detected.

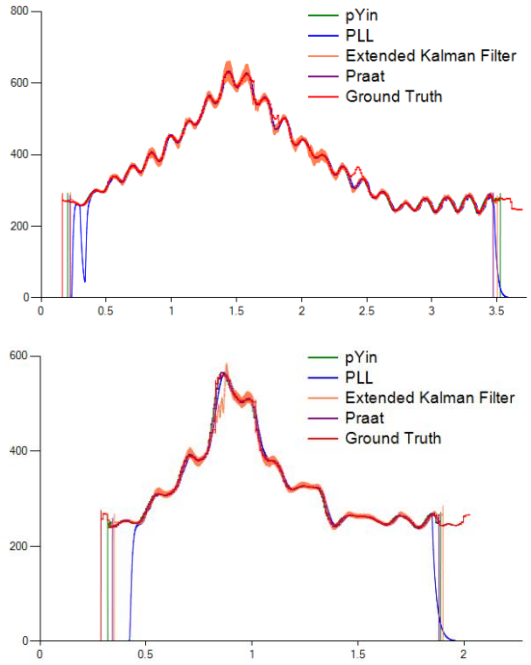

Fig. 3. This picture depicts the PLL algorithm problem at the beginning of sounds

Another comparison that was carried out is to ascertain the standard deviation of the differences between each algorithm with the ground truth. It conducted by subtracting on a sample by sample basis the estimated pitch of each algorithm from its corresponding pitch in the ground truth for all the 76 recorded files. Finally, the average of the standard deviations of each algorithm was calculated and are presented in Table III. This measurement can provide an overview among the frequencies detected as compared to ground truth. If the value is zero means that the algorithm estimated the pitches without any variance, and if the result is closer to 0, it is better than when it is far from zero. As can be seen in this table, the best performance was recorded for pYin followed by Praat and Kalman, and the worst one is for the PLL algorithm.

TABLE III. DIFF BETWEEN THE STANDARD DEVIATION OF EACH ALGORITHM WITH GROUND TRUTH

	pYin	**Praat**	**Kalman**	**PLL**
Diff Std with ground truth's Std	54.13	64.15	83.99	99.89

V. CONCLUSION

This paper has analyzed the performance of four pitch detection algorithms. The PLL and ECKF operate on a sample-by-sample basis and so they seem suitable for real-time implementations. From the experimental results, it was observed that pYin had the best performance followed by Praat. However, the performance of the ECKF and PLL algorithms was not good and, in many cases, exhibited inaccuracies that were manifested as the pitch appearing in the incorrect octave and displaying spurious large deviations. One problem recognized with the ECKF and PLL algorithms is that they are very sensitive to the set of algorithm parameters and are highly dependent on the input data. This means that for different input data it is necessary to find the most appropriate values in advance of applying the algorithm. This is obviously a disincentive to applying these algorithms as it means a significant pre-processing step is required.

With regards to future work, the PLL and ECKF algorithms should be modified to find the best implementation arrangement so that they could work well with human voices. In addition, an effective method should be found that could adjust the algorithm parameters dynamically in scenarios where input signals have properties that vary rapidly.

REFERENCES

[1] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *J. Acoust. Soc. Am.*, vol. 111, no. 4, pp. 1917–1930, 2002.

[2] A. M. Noll, "Cepstrum Pitch Determination," *J. Acoust. Soc. Am.*, vol. 41, no. 2, pp. 293–309, Feb. 1967.

[3] M. R. Schroeder, "Period Histogram and Product Spectrum: New Methods for Fundamental-Frequency Measurement," *J. Acoust. Soc. Am.*, vol. 43, no. 4, pp. 829–834, Apr. 1968.

[4] P. Boersma and V. van Heuven, "PRAAT, a system for doing phonetics by computer," *Glot Int.*, vol. 5, no. 9–10, pp. 341–347, 2001.

[5] M. Mauch and S. Dixon, "PYIN: A fundamental frequency estimator using probabilistic threshold distributions," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, no. 1, pp. 659–663, 2014.

[6] U. Zolzer, S. V. Sankarababu, and S. Moller, "PLL-based Pitch Detection and Tracking for Audio Signals," in *2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2012, pp. 428–431.

[7] O. Das, J. O. Smith, and C. Chafe, "REAL-TIME PITCH TRACKING IN AUDIO SIGNALS WITH THE EXTENDED COMPLEX KALMAN FILTER," in *International Conference on Digital Audio Effects (DAFx-17)*, 2017, pp. 118–124.

[8] R. M. Bittner, A. Wang, and J. P. Bello, "Pitch contour tracking in music using Harmonic Locked Loops," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 191–195.

[9] J. Sundberg, *Perception of Singing*, Third Edit., no. 1960. Elsevier Inc., 2013.

[10] R. M. Van Besouw, J. S. Brereton, and D. M. Howard, "Range of Tuning for Tones With and Without Vibrato," *Music Percept. An Interdiscip. J.*, vol. 26, no. 2, pp. 145–155, 2008.

[11] S. Anand, J. M. Wingate, B. Smith, and R. Shrivastav, "Acoustic parameters critical for an appropriate vibrato," *J. Voice*, vol. 26, no. 6, p. 820.e19-820.e25, 2012.

[12] J. Wilkins, P. Seetharaman, A. Wahl, and B. Pardo, "VocalSet: A Singing Voice Dataset," Mar. 2018.

[13] M. Klingbeil, "Software for Spectral Analysis, Editing, and Synthesis," *Int. Comput. Music Conf.*,

no. 1, 2005.

[14] X. Serra and J. Smith, “Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition,” *Comput. Music J.*, vol. 14, no. 4, p. 12, 1990.

[15] M. Mauch *et al.*, “Computer-aided Melody Note Transcription Using the Tony Software : Accuracy and Efficiency,” *Proc. First Int. Conf. Technol. Music Not. Represent. (TENOR 2015)*, p. 8, 2015.

[16] 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*, 1993, vol. 17, pp. 97–110.