applied sciences

MDPI

Article

OneBitPitch (OBP): Ultra-High-Speed Pitch Detection Algorithm Based on One-Bit Quantization and Modified Autocorrelation

Davide Coccoluto , Valerio Cesarini and Giovanni Costantini *

Department of Electronic Engineering, University of Rome Tor Vergata, 00133 Rome, Italy; davide.coccoluto@gmail.com (D.C.); valerio.cesarini@uniroma2.it (V.C.)
* Correspondence: costantini@uniroma2.it

Featured Application: Fast pitch detection algorithm for the real-time estimation of the fundamental frequency, optimized for hardware implementation.

Abstract: This paper presents a novel, high-speed, and low-complexity algorithm for pitch (F0) detection, along with a new dataset for testing and a comparison of some of the most effective existing techniques. The algorithm, called OneBitPitch (OBP), is based on a modified autocorrelation function applied to a single-bit signal for fast computation. The focus is explicitly on speed for real-time pitch detection applications in pitch detection. A testing procedure is proposed using a proprietary synthetic dataset (SYNTHPITCH) against three of the most widely used algorithms: YIN, SWIPE (Sawtooth Inspired Pitch Estimator) and NLS (Nonlinear-Least Squares-based). The results show how OBP is 9 times faster than the fastest of its alternatives, and 50 times faster than a gold standard like SWIPE, with a mean elapsed time of 4.6 ms, or 0.046 × realtime. OBP is slightly less accurate for high-precision landmarks and noisy signals, but its performance in terms of acceptable error (<2%) is comparable to YIN and SWIPE. NLS emerges as the most accurate, but it is not flexible, being dependent on the input and requiring prior setup. OBP shows to be robust to octave errors while providing acceptable accuracies at ultra-high speeds, with a building nature suited for FPGA (Field-Programmable Gate Array) implementations.

Keywords: pitch detection; F0; algorithm; auto-tune; audio signal processing

check for updates

Citation: Coccoluto, D.; Cesarini, V.; Costantini, G. OneBitPitch (OBP): Ultra-High-Speed Pitch Detection Algorithm Based on One-Bit Quantization and Modified Autocorrelation. *Appl. Sci.* **2023**, *13*, 8191. https://doi.org/10.3390/app13148191

Academic Editor: Douglas O'Shaughnessy

Received: 4 June 2023
Revised: 10 July 2023
Accepted: 11 July 2023
Published: 14 July 2023

1. Introduction

A signal is defined as periodic when the same sequence of values re-occurs after a fixed amount of time, defined as a period, whose inverse is the frequency. By Fourier's principles, a real-world signal can be described as a sum of sinusoids (or "pure tones") that only carry one frequency [1]. The fundamental frequency, or F0, is defined as the lowest frequency describing a periodic component of a signal. In the audio domain, F0 is defined as the pitch, which in music is translated to a specific note.

Detecting the F0 of a signal is a crucial application in many fields, with audio, music, and speech processing being heavily reliant on pitch-detection-based technologies, as well as other fields such as fault detection of moving parts, which are related to their resonance frequency [2], or sonar systems for target detection, classification and localization [3]. Moreover, pitch detection can also be employed in the characterization of accurate sinusoidal voltages, as described by Krajewski et al. [4].

The problem of pitch detection is crucial in all the applications that rely on knowing the fundamental frequency in order to perform periodicity-related computations, such as acoustic feature extraction relying on prosodic metrics such as HNR, jitter or shimmer that evaluate "cycle-to-cycle" variations [5]. Moreover, professional audio relies on pitch detection to build tuners for real instruments, or for real-time pitch re-adjusting applications especially directed toward vocal tuning. Real-time detectors are thus necessary to enable performers to monitor pitch accuracy and trigger events in real time, especially related to MIDI applications [6]. Additionally, fast, real-time pitch detection is valuable in interactive audio applications, such as games and virtual reality, where it enables dynamic audio synthesis and effects as well as responsive processing.

The two main characteristics of pitch detection algorithms are speed and accuracy, which often imply a trade-off depending on the specific application—for example, posterior analyses do not need real-time F0 estimation, as opposed to live music [6].

With the diffusion of technologies such as "Autotune" for real-time pitch correction in singers [7], and with the widespread use of MIDI instruments and/or live MIDI transformers to digitalize acoustic instruments, real-time pitch detection sees a crucial application in the music industry. Especially for MIDI purposes where a sound needs to be translated into a discrete note, speed is favored over accuracy due to the need for low-latency live solutions, and thanks to the fact that the detected frequency is discretized into a note of the tempered system allowing for a certain range for errors. The problem of detecting the fundamental frequency in real time is crucial whenever live performances are involved, as even minuscule latencies of a few milliseconds can be perceived by the musician or operator.

In speech analysis, F0 carries a crucial role as a biometric feature for characterizing voice impairment, up to singlehandedly being used for pre-diagnostic purposes, where F0-related features and their variations are used for the detection of respiratory [8], phonatory [9,10] or neurodegenerative diseases [11–13].

Given the multifaceted applications and different industry needs, the state of the art of pitch detection depends on the application.

From the mathematical point of view, although an ever-growing plethora of algorithms are being developed, the vast majority can be generally categorized into three approaches: time-based, frequency or Cepstrum-based, full heuristic. Time domain approaches are generally based on the mathematical principles behind autocorrelation, which inherently has peaks every time a signal repeats (maximum correlation with itself): the problem of pitch detection is thus translated into the problem of finding the maximum of the autocorrelation function. Frequency domain algorithms are based on Fourier domain or cepstral analysis, with the Harmonic Product Spectrum (HPS) [14] as a notable example: it computes the product of the power spectra of a signal and its downsampled versions to emphasize harmonic components. The fundamental frequency is then estimated by identifying peaks in the resulting spectrum. In recent years, the research trend was predominantly based on Deep Learning methods based on Convolutional Neural Networks (CNN) [15], which offer the advantages of being able to control the size of the computational net and also to specifically train on suitable data, since the problem of pitch detection is data-dependent [16]. However, CNN-based methods are not easily generalizable, and require data and especially time for training time before they are usable in their "inference" form—which usually brings high accuracies but slightly less optimal speed [17].

The prior definition of the latency/complexity of a pitch detection algorithm is hard to determine, since each algorithm and performance inherently depends on the acoustic and digital nature of the data—such as the number of bits for quantization.

Theoretically, HPS-based methodologies yield a complexity of O(NlogN), and straightforward autocorrelation is at O(N^2), whereas FFT-based autocorrelation is O(NlogN) yet again, being comparable to HPS. The Fast Fourier Transform (FFT) algorithm is used to speed up the computation of autocorrelation. By leveraging the symmetry properties in the autocorrelation function, the complexity of FFT-based autocorrelation is reduced. The process involves padding the input signal to the nearest power of 2, computing the FFT of the padded signal, squaring the magnitude of each frequency bin to obtain the power spectrum, computing the inverse FFT of the power spectrum and normalizing the result by dividing it by the length of the input signal.

After a rough estimate of F0, many algorithms have to rely on some corrective heuristics to fine-tune the result and/or avoid octave errors. Common signal processing solutions may be employed for this purpose as well, with notable results such as the work by Khadem-hosseini et al. [18] employing HPS and Euclidean summation. However, although this results in improved accuracy, it is an inherently computationally expensive mean, and real-time pitch detectors might choose to avoid relying on correctors—this is also the approach that we chose in the present paper.

Due to the inherent harmonic nature of most real-world sound signals, especially when dealing with speech or music analysis, octave errors are a common criticality among pitch detection algorithms, being triggered by the eventual presence of strong first- or second-order harmonics and, partially, by aliasing.

Two of the most widely used algorithms, SWIPE [19] and YIN [20], which will be detailed later in Section 2, are based on autocorrelation. More algorithms are listed in the works by Camacho and Harris [19] and Ruslan et al. [1].

Attempts at fast pitch detectors are based on the simplification of the transformation procedures, such as the work by Grinewitschus et al. [21], which leverages the constant-Q Gabor transform for a threshold-based approach within a four-dimensional logarithmic harmonic spectrum shift. A work by Mnasri et al. [22] aims to avoid short-time analysis and thus the underlying approximations about local stationarity by employing the Hilbert transform to derive "instantaneous" frequency components to contour F0: the performances might be comparable to YIN or SWIPE, but no indication on speed is given.

In general, the problem of real-time, high-speed pitch detection has to be faced with the development of a computationally light algorithm that still retains a relative error suitable for the required application (mainly professional audio and live performances).

The scope of this paper is to propose a novel, high-speed implementation of a pitch detection algorithm based on a modified version of the autocorrelation, and to assess the performances of the most highly regarded algorithms in terms of speed and accuracy, on a suitable dataset purposefully built as a test bench.

For the purposes of testing pitch detection algorithms on sheer speed or recognition capabilities, a custom dataset named SYNTHPITCH was built by producing synthetic signals so that the original pitch/F0 is objective and priorly known.

Other algorithms such as YAAPT [23], SHRP [24] or the CREPE [25] CNN approach have been experimented with, but their preliminary results were not notable with respect to the others considered, especially for the speed vs. accuracy tradeoff. With speed being the main characteristic to search for, non-notable algorithms that provide high accuracies but poor speed have not been included in the present assessment although experimentations were made on them in order to rule them out, and synthetic signals are employed to evaluate the sheer computational complexity, while not forgoing the ability to infer pitch.

The main contributions of this paper lie in the presentation of a novel pitch detection algorithm, based on a partially unexplored approach focused on high-speed and low bit depth, very suitable for hardware implementations. All of these characteristics make it a good candidate for live performance applications or MIDI instruments, which rely on real-time note detection. The mathematical and signal processing theories behind our novel algorithm explore the characteristics of the autocorrelation function, its maximization and its approximations, as well as the effect of quantization on the fundamental frequency of a signal.

Along with the new algorithm, a testing paradigm for evaluating the speed and computational complexity of pitch detection algorithms is proposed, and a custom, synthetic dataset is produced and made available to the public. State-of-the-art, pre-existing pitch detection algorithms, especially those focusing on speed, are thoroughly explored and tested. The article is organized as follows. Section 2 presents the OneBitPitch algorithm along with its mathematical discussion, theoretical derivation and implementation, as well as three other algorithms (YIN, NLS, SWIPE) used for comparison and the custom SYNTHPITCH dataset produced and used in this paper. Section 3 presents the numerical results obtained from the simulations, as well as a statistical analysis. Then, Section 4 provides an in-depth discussion of the obtained results, especially focusing on real-time implementation and runtime speed, which are the main focuses of the analyses. The strengths and weaknesses of every algorithm are analyzed along with practical situations where each algorithm is best suited. Limitations and future works, especially regarding hardware solutions, are presented at the end of the Discussion and before the Conclusions.

2. Materials and Methods

This paper proposes a novel pitch detection algorithm called OneBitPitch, in short, OBP, based on a modified autocorrelation function applied to a one-bit version of the original signal, for maximum speed and hardware implementation capabilities. In order to evaluate the performance of the OneBitPitch algorithm, a custom synthetic dataset (SYNTHPITCH) was built and 4 different algorithms were compared on it.

The choice of the algorithms was based on well-known, state-of-the-art pitch estimators especially directed toward high speed or high accuracy. The main focus is the sheer algorithmic performance, although it is well known that the effectiveness of any pitch detection model is inherently dependent on the dataset and purpose (i.e., pure signals, voice data, etc.).

This section first presents the public dataset built for the purpose of this study, then details the OBP algorithm along with its mathematical basis and briefly presents the three algorithms used for comparison: YIN, SWIPE and NLS.

2.1. SYNTHPITCH Dataset

The SYNTHPITCH dataset was purposefully built for the scope of the analyses presented in this work, i.e., for testing pitch detection algorithms on arbitrarily complex signals in terms of fundamental frequency intelligibility, and to evaluate their computational complexity/speed.

All signals were sampled at 96 ksps and represented in floating points. A typical audio setup was reproduced, so a 20 kHz low-pass filter was applied. Twelve different categories are present, with each one encompassing 99 signals with increasing fundamental frequency, starting from 100 Hz up to 5000 Hz with a 50 Hz step size.

The categories were built with increasing pitch complexity, the simplest one being pure sine waves, to which multiple artifacts have been applied to generate sounds of increasing complexity. There are also two categories encompassing square waves; the amplitude of each starting wave is normalized to have a peak of 1. Table 1 details the characteristics of each category and its name, as well as the number of artifacts applied, with the following macroscopic characteristics:

- Harmonics: Addition of a number of harmonic frequencies, i.e., integer multiples of F0. The amplitude of each harmonic is a random number between 0 and 1, sampled from the Gaussian distribution, and eventually re-scaled if more/less amplitude is needed;
- Partials: Addition of non-integer, random multiples of F0, with random starting amplitude linearly scaled according to the order, and random phase. The following formula explains the construction of an i-th order partial, with A_i, f_i and φ_i being random amplitude (scaled according to the order), random frequency obtained by multiplying F0 by a random number between 0 and 1 (scaled according to the order) and random phase. All random quantities are obtained by sampling a Gaussian distribution with max = 1, and the formula is as follows:

$$Partial_i(t) = \frac{A_i}{\mathrm{i}} \cdot \sin(2\pi i f_i t + \varphi_i); \tag{1}$$

- White Gaussian Noise: Addition of white Gaussian noise of a given SNR, after measuring the power of the signal with added harmonics/partials;
- Reverb: Addition of a reverberated copy of the signal with amplitude equal to 0.1 of the measured amplitude of the starting signal.

Table 1. Name and description of the signal categories making up the SYNTHPITCH dataset. The order is alphabetical.

Name	Description
2harm	Pure sine waves plus the first two harmonics with random amplitude between 0 and 1
2harm_wgn15	Pure sine waves plus 2 harmonics (random amplitude between 0 and 1) plus white Gaussian noise with SNR = 15
4harm	Pure sine waves plus 4 harmonics with random amplitude between 0 and 1
4harm_4part_wgn15	Pure sine waves plus 4 harmonics (random amplitude between 0 and 1) and 4 partials (linearly decreasing amplitude)
4harm_high	Pure sine waves plus 4 harmonics with random amplitude between 0 and 3
4harm_wgn15	Pure sine waves plus 4 harmonics with random amplitude between 0 and 1 plus white Gaussian noise with SNR = 15
full1	Pure sine waves plus 10 harmonics (random amplitude between 0 and 2) and 10 partials (maximum amplitude = 2), plus white Gaussian noise with SNR = 1 and reverb (0.1 RMS)
full2	Pure sine waves plus 10 harmonics (random amplitude between 0 and 2) and 10 partials (maximum amplitude = 2), plus white Gaussian noise with SNR = 10 and reverb (0.1 RMS)
pure	Pure sine waves
pure_wgn0P3	Pure sine waves plus white Gaussian noise with SNR = 0.3
square_pure	Square waves
square_wgn10	Square waves plus white Gaussian noise with SNR = 10

Figure 1 details some examples of signals found on the SYNTHPITCH dataset; notice how, with complex/dirty signals such as those present in the "full1" category, pitch and sinusoidal behavior become very hard to infer. The dataset is free to use for the public.

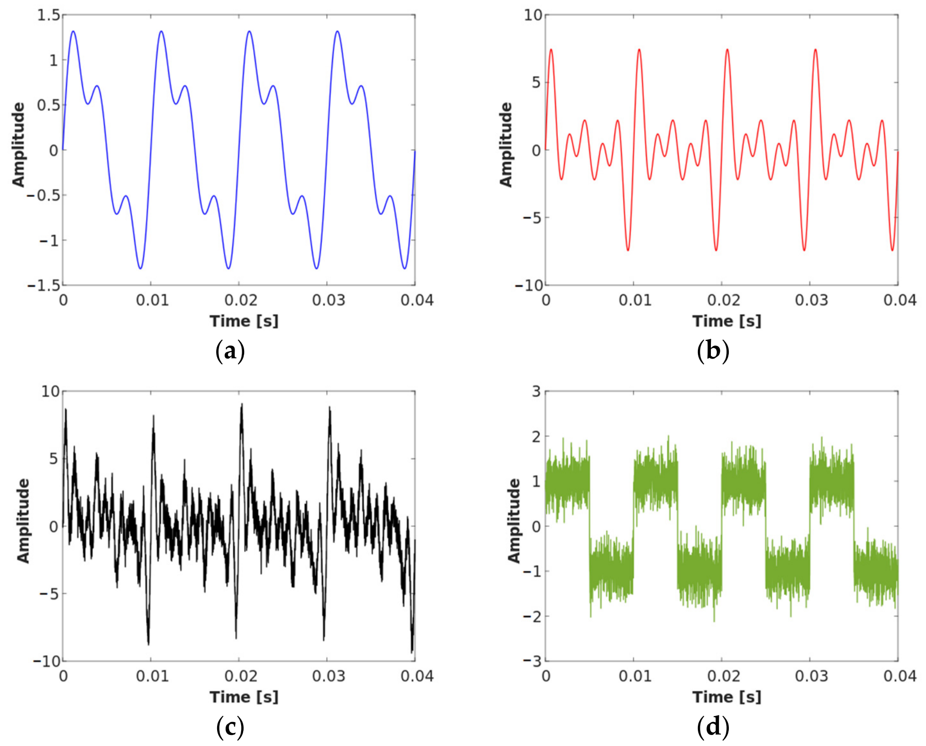


Figure 1. Sample signals from the SYNTHPITCH dataset (F0 = 100 Hz): (**a**) "2harm" sample (blue); (**b**) "4harm_high" sample (red); (**c**) "full1" sample (black); (**d**) "square_wgn10" sample (green).

2.2. *OneBitPitch Algorithm*

The algorithm proposed in this paper is aimed at ultra-fast pitch detection, for real-time usage, and was developed as a starting point for future hardware implementation and for heavy-duty, latency-free live use.

With these premises, our proposed algorithm aims to reduce the computational complexity to its bare minimum, sacrificing accuracy while still staying in acceptable territories, to provide the highest possible speed performances.

The OneBitPitch (OBP for short) algorithm exploits a modified version of the autocorrelation in time approach highly optimized for execution time and computational complexity.

The basic idea is that reducing the resolution of a signal, i.e., the number of quantization bits, worsens the signal but retains its periodicity. Taking this idea to its limit, we can state the following:

Proposition 1. *Let x be a digital signal modeled as a zero-average periodic sequence quantized with N bits and with F0 being its fundamental frequency. Re-quantizing x with M < N bits, the original F0 is preserved in the re-quantized signal.*

This can be easily proven by considering that, for a periodic signal in which F0 is the reciprocal of a period T0, the time duration of such a period does not change with truncation (re-quantization), although the exact timeframe can be anticipated/delayed according to rounding conventions [26–28]. With the assumption of no aliasing (anti-aliasing has been performed priorly), this holds true for any amount of bits, up to the minimum limit of 1-bit quantization, which basically results in the sign function. Although discretizing the amplitudes might indeed insert artifacts that generate new periodicities, the original frequency that acted as a fundamental is ultimately preserved. No matter how coarsely quantized, a sequence will still be repeating itself regardless of the discretization steps. Figure 2 explains this by showing an infinite precision signal, a down-quantized version and a "sign" signal quantized with 1 bit. With the period starting at 0, it can be shown that all signals re-start at the exact same moment, despite the quantization error being increasingly high.

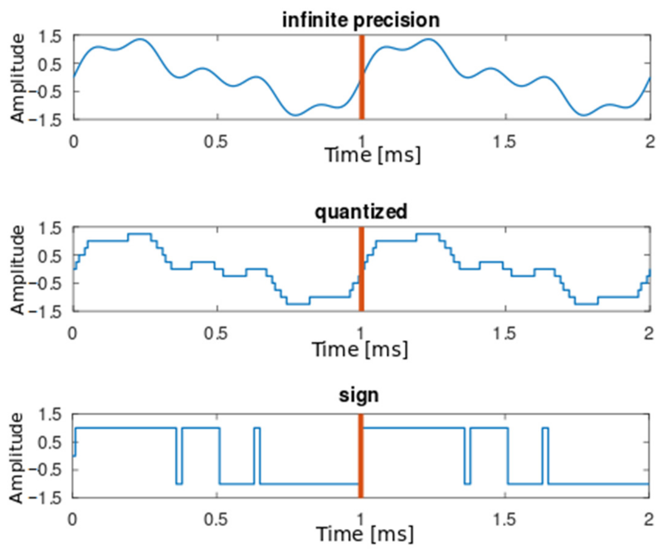


Figure 2. Period of a periodic signal and its quantized versions. The time at which the signal is measured to be repeating (period) is in orange. The "quantized" version is at N = 4 bits (16 values) and the "sign" version is quantized with 1 bit.

The mathematical principles for pitch detection are based on the intuition that the maximum of the autocorrelation [29] of a signal is when such a signal repeats itself, i.e., at its fundamental frequency.

The common definition for the autocorrelation of a signal x is [30]:

$$r_{xx}(l) = \frac{1}{\text{N}} \cdot \sum_{k=1}^{N} x(k)x(k+l) \tag{2}$$

With l being the "lag" (progressive shift of a signal to have it slide on the other), N is the length of the signal in terms of the number of samples, and k is the discrete time (number of samples).

Implementing a full autocorrelation function has some drawbacks: it is relatively computationally expensive due to the need for reiterated multiplications, leading to a complexity of O(N^2), and further normalization is required because the output is heavily dependent on the magnitude of the input signal and its variations, which skew the autocorrelation [20].

However, for a periodic signal, which can be defined as $x(k)$, which repeats after a period of T samples, so that $x(k) = x(k + T)$, a "difference function" can be defined so that its minimum corresponds to the period. The formula, expressed in terms of the lag l as the independent variable (in a digital sequence simply refers to the sample number), is the following:

$$d_{xx}(l) = \sum_{k=1}^{N} |x(k) - x(k+l)| \tag{3}$$

with l being the lag (sample), and N is the length of the signal x in samples. This formula will be referred to, for simplicity, as "modified autocorrelation", with the basic idea that instead of searching for the maximum of the product like pitch detection algorithms employing "usual" autocorrelation, we can search for the minimum of the difference. Moreover, this function is inherently independent of the input amplitude and its variations [31].

Figure 3 displays an example comparing the autocorrelation and the difference function ("modified autocorrelation"), showing how the maxima of the first roughly correspond to the minima of the latter.

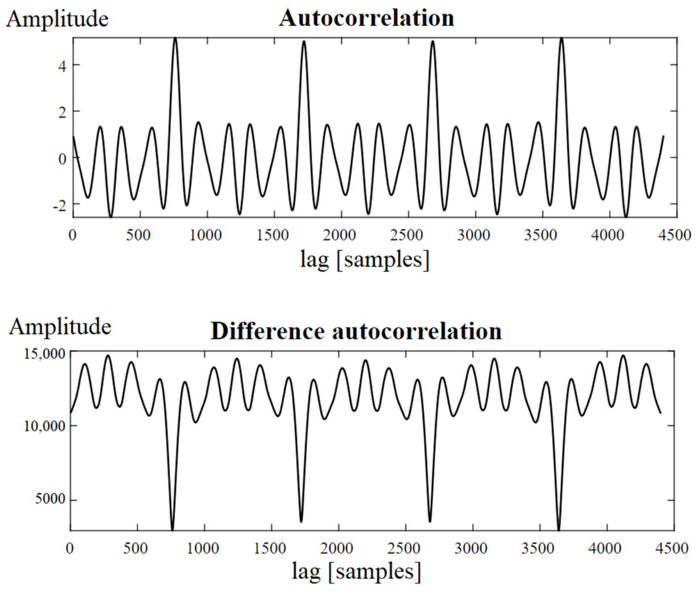


Figure 3. Comparison of autocorrelation versus the modified difference function version, computed on a signal from the "full2" category (F0 = 100 Hz). Notice how a maximum of the autocorrelation corresponds to a minimum in the difference function.

The operation to make the sum independent from negative values can be the square, as implemented by de Cheveignè and Kawahara [20], or the absolute, as we chose. In our specific case, this actually becomes irrelevant due to the 1-bit quantization, which essentially renders this formula into an XOR operation, which is inherently optimized for hardware implementations and parallelization.

However, for more than 1-bit quantization, the absolute is still more efficient, especially for FPGA implementations, because it can essentially be realized with a multiplexer for the MSB and a conditioned re-assign of the sign.

From the O(N^2) computational complexity of the normal autocorrelation function, a single-bit XOR implementing the difference autocorrelation yields a theoretical constant complexity of O(1), being essentially a binary addiction without carry [32,33].

The idea behind the OBP algorithm is thus to apply an optimized autocorrelation-based pitch detection to a 1-bit version of the original signal, which drastically decreases computation complexity, especially considering that common bit depths employ 16 to 64 bits. Within this picture, Kawecka and Podahjecki explored the probabilistic properties of quantizers [34]. The operation of using 1 bit is logically equal to the "sign" operation. In the most common digital representations (two's complement, floating-point IEEE-754 [35], sign/magnitude [36]), the MSB (most significant bit) is used as the sign and thus a simple truncation of the original sequence is required—this could also be performed in the hardware domain for maximum speed.

The building blocks of the OBP algorithm are represented in Figure 4 and described as follows:

- Input: The input signal is assumed to be limited in band, with no DC components, which can be obtained before ever entering the digital realm by analog filtering; this also guarantees the zero average.
- Sign: From the original signal, only the sign is extracted; in any common digital environment, this is obtained by just retaining the MSB. For different kinds of representation, a comparator would be required.
- Buffer: It is needed to store a fixed amount of the previous samples of the signal and its length is related to the minimum frequency that needs to be detected (*Fmin*). With *Fs* being the sampling rate, a buffer length of *Fs/Fmin* samples is required.
- Modified autocorrelation: The difference autocorrelation previously defined is applied to the sign signal within the buffer. Instead of performing a circular correlation, like other algorithms (namely, YIN), a linear correlation is applied on a total frame length of one buffer and a half to avoid phase jumps. For periodic signals, autocorrelation tends to rise and fall from/to the minimum symmetrically: for this reason, the center of the minimum is to be considered to evaluate the period.
- Threshold: The search for the minimum of the difference autocorrelation function is simply performed by thresholding the signal with a fixed threshold. This can be visualized with a "Thresholding" logical signal that is 1 only when the autocorrelation is under the threshold. Although more sophisticated methods can be implemented, such a naïve implementation is computationally efficient and has been empirically observed to be a good trade-off between accuracy and speed.
- Output: As previously stated, the very minimum point is the center of the sections below the thresholds, which is equal to a logical 1 in the "Thresholding" signal. The frequency result is simply obtained by counting the number of samples between two minima.

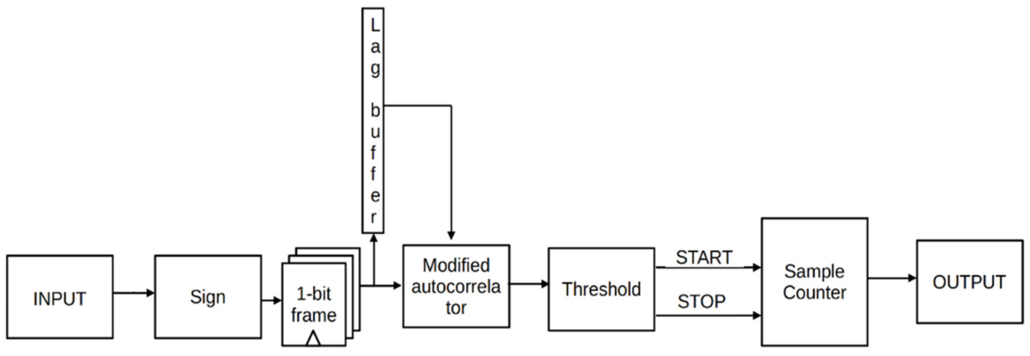


Figure 4. Block diagram of the OneBitPitch (OBP) pitch detection algorithm.

Figure 5 displays the progression of the algorithm by showing each internal signal.

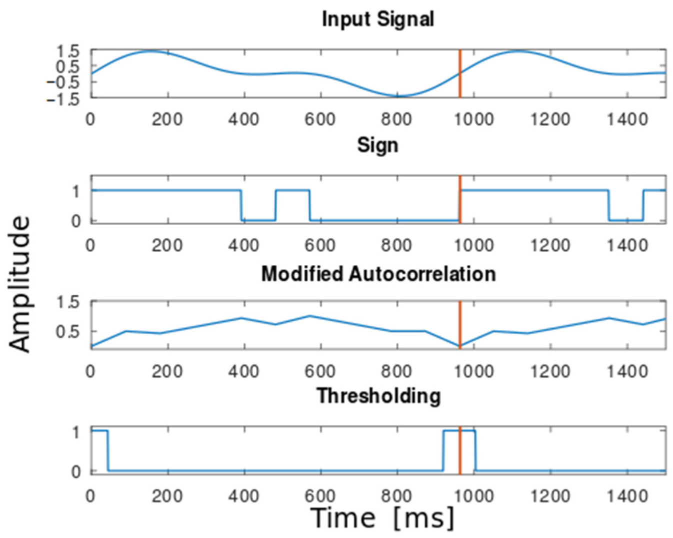


Figure 5. Processing steps within the OBP pipeline. The orange line shows the period, which is correctly detected as the center of the section below the threshold. The "Thresholding" signal is 1 when the "Modified Autocorrelation" signal goes below a fixed threshold (minimum search).

2.3. YIN Algorithm

The YIN algorithm [20] is an autocorrelation-based pitch estimator allegedly bringing high speed and good accuracy with few parameters to tune. It utilizes a modified autocorrelation difference function called Cumulative Mean Normalized Difference Function, used iteratively to avoid zero lag and to normalize the difference function with respect to large lags. Parabolic interpolation is then employed to obtain sub-sampling resolutions. In order to reduce octave errors and optimize the result, the search for the pitch candidate is aided by a heuristic based on range reduction. As for many algorithms, additional tuning possibilities are forecasted for unclean signals (i.e., presence of additive noise or additive frequencies/harmonics), such as comb filtering.

2.4. SWIPE (Sawtooth-Inspired Pitch Estimator) Algorithm

SWIPE was developed by Camacho and Harris [19]. It consists of measuring the average distance between valleys and peaks on the spectrum, at harmonics of the pitch. After the first estimation, SWIPE tries to refine the output by exploiting a variable window size and evaluating the best pitch candidate.

The pitch is estimated by comparing the spectrum of the signal to the sawtooth waveform whose spectrum is most similar. This is achieved by calculating a normalized inner product between the signal spectrum and a modified cosine. The analysis window size is adjusted to align the main lobes of the spectrum with the positive lobes of the cosine and parabolic interpolation is employed for added accuracy. SWIPE′ (or SWIPE prime or SWIPEP) is a variant of SWIPE built to minimize subharmonic errors, which the original algorithm was prone to, by only employing the first harmonic and the prime ones. It is the most widespread version, present in advanced libraries such as Tsanas' Voice Analysis Toolbox [37], and will thus be the one used in this paper.

2.5. NLS (Nonlinear Least Squares) Algorithm

The Nonlinear Least Squares (NLS) principle is based on a statistical Maximum Likelihood (ML) candidate research [38,39]. This class of algorithms can theoretically achieve the highest degree of accuracy, especially on discretized pitches, at the cost of greater computational complexity.

The algorithm proceeds by iteratively minimizing the cost function for the estimation error.

The algorithm starts by selecting a range of pitch candidates that cover the expected range of the fundamental frequency: this operation requires a prior setup of the algorithm, which adds latency before being usable. For each candidate, a synthesized signal is generated with harmonics or sinusoids at that pitch frequency. The objective is to find the pitch candidate that best matches the observed signal by minimizing the sum of squared differences between the observed and synthesized signals. Optimization techniques like gradient descent are used to update the pitch candidates iteratively until convergence. Finally, the pitch candidate with the minimum objective function value is selected as the estimated fundamental frequency or pitch of the signal.

The NLS algorithm is known for its ability to handle complex harmonic structures, variable pitch signals, and noisy environments. It is particularly effective when the signal contains multiple overlapping or interfering harmonic components.

For the purpose of this study, a fast implementation of an NLS-based algorithm by Wang et al. [40] is employed, shown to reduce the complexity by solving two Toeplitz-plus-Hankel systems of equations and using the recursive-in-order matrix structures.

2.6. Test Conditions

All the algorithms were tested on the same set of signals from the SYNTHPITCH dataset. The sampling rate was 96 KHz, the considered duration of the signals was 100 ms (9600 samples) and the frequency limits for algorithms that require it (e.g., YIN needs to set the minimum detectable frequency) were between 50 and 5000 Hz, in order to also observe eventual subharmonic errors on even the lowest of the frequencies in the dataset (which is 100 Hz). This resulted in buffer lengths of $Fs/Fmin$ = 1920 samples. The following hyperparameters and setup were employed for each algorithm under test:

- OBP: Our algorithm is used with a fixed threshold at 400. Tuning the threshold, predictably, allows us to adapt to different input classes or characteristics.
- YIN: The resolution for the parabolic interpolation is 1 cent, the fixed threshold is at 0.1.
- SWIPE: The resolution for the parabolic interpolation is 1 cent, the harmonics considered are only the first and the prime ones (making it SWIPEP), the timeframe is at 10 ms and the final result is the mean of the tracked frequencies. Using a bigger or smaller timeframe does not sensibly change the elapsed time due to the inherent nature of SWIPE and it being reliant on generating sawtooth waves.
- NLS: An NLS-based model is generated from a synth sample having a number of harmonics equal to 4. Increasing the number of harmonics, in the case of the present study, leads to sensibly higher elapsed times while not improving accuracy. The time needed to generate the model, i.e., the unavoidable latency at the beginning, is on average around 40 ms. For simplicity and uniformity, this latency will not be considered when discussing elapsed times, with the assumption that in a real-world scenario, the model is pre-made. However, this is a small disadvantage.

The test involves running all of the algorithms on each signal (duration = 100 ms) in each category of the SYNTHPITCH dataset. The main metrics are time elapsed in seconds (TE) and relative absolute error (RAE) computed by comparing the estimated frequency with the known F0 of each signal according to the following formula:

$$RAE = \left| \frac{y_E - y}{y} \right| \tag{4}$$

with y_E being the estimated value and y being the real/target one.

Both the RAE and the TE are averaged for each category. Due to the real-world applications of pitch detection algorithms, and also taking into account the fact that real-world signals might not present a discrete F0, the accuracy within a certain range is also presented. Specifically, the ranges 1%, 2% and 10% of the true F0 are considered, producing the metrics ACC-1, ACC-2 and ACC-10. These accuracies are presented as averages over categories as well.

The ranges were chosen empirically, with 2% being a truly "acceptable" error in most music/MIDI-related applications, and with 10% ruling out octave errors, which inherently produce 100% or above errors. Acceptable RAE values can be approximately below 0.025, because most musical applications use discretized pitches that do not result in note errors if within a range < 2.5% around the starting pitch. We chose 2% as a safety measure: this range is well represented by the "ACC-2" metric, which is the percentage of instances in which the algorithm brings an error equal to or lower than 2%. Section 4 will further detail this.

In order to assess the statistical validity of the results, a Mann–Whitney U-test was performed on each table reporting RAE, TE and ACC metrics within the results [41]. With OBP as the main algorithm under test, U, Pearson's p (or "rho") and the z-score were derived for each one-vs.-one comparison, with $p < 0.05$ used as the significant threshold [42].

3. Results

Tables 2 and 3 show the performances of each algorithm in terms of time elapsed (TE) and relative absolute error (RAE). Tables 4–6 detail the accuracy with acceptance rates of 1% (ACC-1), 2% (ACC-2) and 10% (ACC-10). Table 7 presents the results of the statistical analysis.

Table 2. RAE (relative absolute error) of each algorithm averaged over each category of the SYNT-PITCH dataset, along with the mean for each algorithm over the whole dataset.

RAE (Average)				
Dataset Category	**Algorithm**			
	YIN	**SWIPE**	**NLS**	**OBP**
2harm	0.003157	0.002998	0.000000	0.013210
2harm_wgn15	0.003071	0.002738	0.000000	0.014091
4harm	0.001324	0.002568	0.000000	0.052182
4harm_4part_wgn15	0.420658	0.304243	0.000000	0.154955
4harm_high	0.005729	0.004558	0.069865	0.985475
4harm_wgn15	0.001185	0.002136	0.000000	0.014095
full1	0.396471	0.252746	0.473064	0.201402
full2	0.254010	0.210246	0.483956	0.369483
pure	0.007338	0.004879	0.000000	0.013462
pure_wgn0P3	0.008713	0.019694	0.000000	0.012983
square_pure	0.334296	0.011628	0.000000	0.013462
square_wgn10	0.262356	0.013746	0.000000	0.013683
MEAN	0.141526	0.069348	0.085574	0.154873

Table 3. TE (time elapsed) in seconds for each algorithm averaged over each category of the SYNT-PITCH dataset, along with the mean for each algorithm over the whole dataset.

Time Elapsed (TE, Average)				
Dataset Category	**Algorithm**			
	YIN	**SWIPE**	**NLS**	**OBP**
2harm	0.040090	0.266015	0.025799	0.004731
2harm_wgn15	0.033833	0.248275	0.025342	0.005095
4harm	0.033381	0.251229	0.026266	0.004853
4harm_4part_wgn15	0.033516	0.253206	0.025506	0.004677
4harm_high	0.035921	0.245899	0.025181	0.004831
4harm_wgn15	0.035036	0.271767	0.025841	0.005500
full1	0.036627	0.267752	0.031347	0.004833
full2	0.035664	0.254639	0.024700	0.004661
pure	0.035881	0.247622	0.025245	0.005238
pure_wgn0P3	0.035488	0.248970	0.025257	0.004766
square_pure	0.035509	0.246896	0.024810	0.004741
square_wgn10	0.035650	0.246456	0.028136	0.004685
MEAN	0.035550	0.254061	0.026119	0.004884

Table 4. ACC-1: percentage of the times each algorithm has provided an estimated frequency that brings RAE < 0.01. Averaged over each category of the SYNTHPITCH dataset, along with the mean for each algorithm over the whole dataset.

ACC-1				
Dataset Category	**Algorithm**			
	YIN	**SWIPE**	**NLS**	**OBP**
2harm	0.949495	0.989899	1.000000	0.474747
2harm_wgn15	0.969697	0.979798	1.000000	0.393939
4harm	0.989899	0.989899	1.000000	0.434343
4harm_4part_wgn15	0.414141	0.242424	1.000000	0.303030
4harm_high	0.989899	0.989899	0.939394	0.303030
4harm_wgn15	1.000000	0.989899	1.000000	0.424242
full1	0.434343	0.191919	0.595960	0.414141
full2	0.575758	0.262626	0.646465	0.474747
pure	0.777778	0.868687	1.000000	0.444444
pure_wgn0P3	0.979798	0.606061	1.000000	0.484848
square_pure	0.626263	0.939394	1.000000	0.444444
square_wgn10	0.606061	0.858586	1.000000	0.424242
MEAN	0.776094	0.742424	0.931818	0.418350

Table 5. ACC-2: percentage of the times each algorithm has provided an estimated frequency that brings RAE < 0.02. Averaged over each category of the SYNTHPITCH dataset, along with the mean for each algorithm over the whole dataset.

ACC-2				
Dataset Category	**Algorithm**			
	YIN	**SWIPE**	**NLS**	**OBP**
2harm	1.000000	1.000000	1.000000	0.818182
2harm_wgn15	1.000000	1.000000	1.000000	0.777778
4harm	1.000000	1.000000	1.000000	0.737374
4harm_4part_wgn15	0.414141	0.333333	1.000000	0.545455
4harm_high	0.989899	1.000000	0.939394	0.464646
4harm_wgn15	1.000000	1.000000	1.000000	0.757576
full1	0.434343	0.191919	0.595960	0.575758
full2	0.575758	0.272727	0.646465	0.717172
pure	1.000000	1.000000	1.000000	0.818182
pure_wgn0P3	0.989899	0.797980	1.000000	0.818182
square_pure	0.676768	0.949495	1.000000	0.818182
square_wgn10	0.606061	0.858586	1.000000	0.797980
MEAN	0.807239	0.783670	0.931818	0.720539

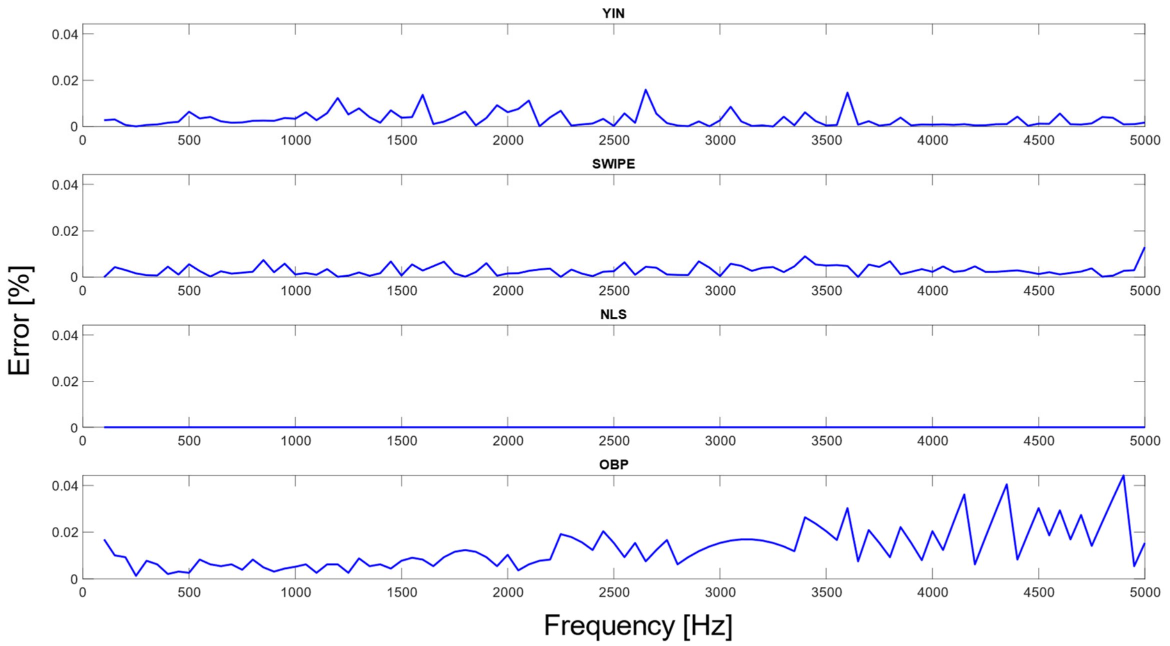
Table 6. ACC-10: percentage of the times each algorithm has provided an estimated frequency that brings RAE < 0.10. Averaged over each category of the SYNTHPITCH dataset, along with the mean for each algorithm over the whole dataset.

ACC-10				
Dataset Category	**Algorithm**			
	YIN	**SWIPE**	**NLS**	**OBP**
2harm	1.000000	1.000000	1.000000	1.000000
2harm_wgn15	1.000000	1.000000	1.000000	1.000000
4harm	1.000000	1.000000	1.000000	0.979798
4harm_4part_wgn15	0.414141	0.484848	1.000000	0.757576
4harm_high	0.989899	1.000000	0.939394	0.585859
4harm_wgn15	1.000000	1.000000	1.000000	1.000000
full1	0.434343	0.383838	0.595960	0.696970
full2	0.575758	0.444444	0.646465	0.828283
pure	1.000000	1.000000	1.000000	1.000000
pure_wgn0P3	0.989899	0.989899	1.000000	1.000000
square_pure	0.727273	0.979798	1.000000	1.000000
square_wgn10	0.606061	0.979798	1.000000	1.000000
MEAN	0.811448	0.855219	0.931818	0.904040

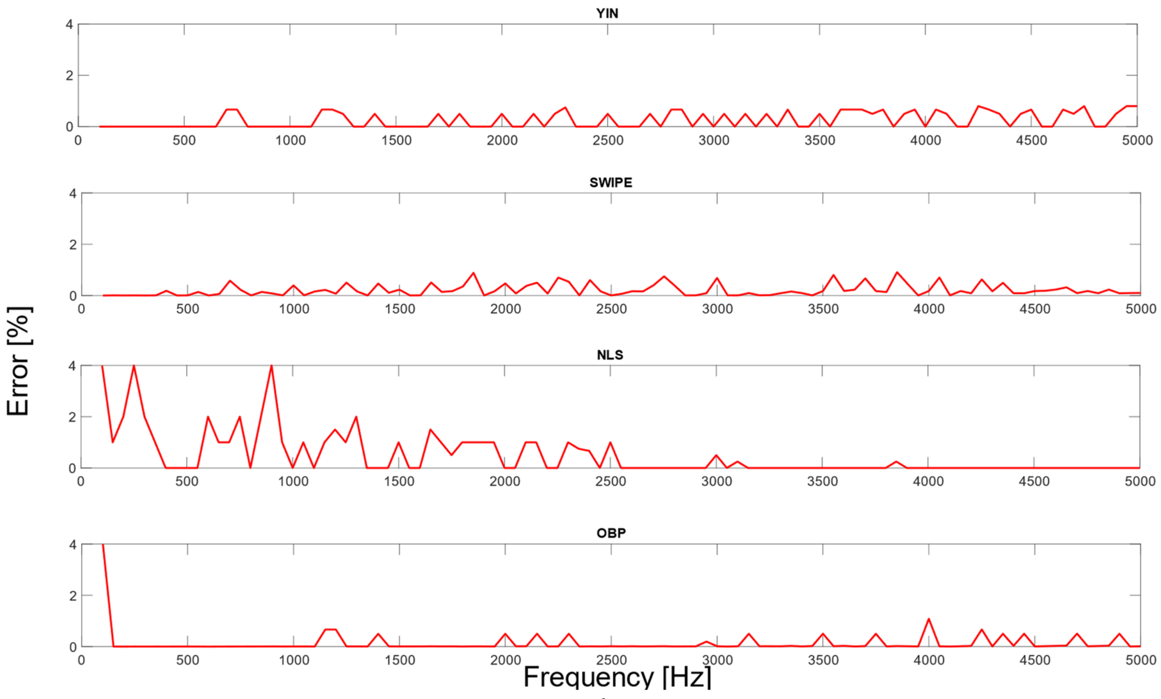
Table 7. Results of the Mann–Whitney statistical test for the relevance of each metric for each couple of algorithms. Comparisons involving OBP are in bold. A z-score < −4 associated with $p < 0.000001$ is related to a confidence higher than 99.997%.

Comparison	**Statistical Measures**		
	U	**Pearson *p***	**z-Score**
RAE (OBP vs. SWIPE)	42	0.08914	1.70318
RAE (OBP vs. NLS)	30	0.0164	2.396
RAE (OBP vs. YIN)	52	0.25848	1.12583
RAE (YIN vs. SWIPE)	60	0.50926	0.66395
RAE (YIN vs. NLS)	31	0.01928	2.33827
RAE (SWIPE vs. NLS)	33	0.02642	2.2228
TE (OBP vs. SWIPE)	0	<0.000001	<−4
TE (OBP vs. NLS)	0	<0.000001	<−4
TE (OBP vs. YIN)	0	<0.000001	<−4
TE (YIN vs. SWIPE)	0	<0.000001	<−4
TE (YIN vs. NLS)	0	<0.000001	<−4
TE (SWIPE vs. NLS)	0	<0.000001	<−4
ACC-1 (OBP vs. SWIPE)	36	0.04036	−2.04959
ACC-1 (OBP vs. NLS)	0	<0.000001	<−4
ACC-1 (OBP vs. YIN)	14	0.0009	−3.31976
ACC-1 (YIN vs. SWIPE)	70	0.92828	0.0866
ACC-1 (YIN vs. NLS)	26.5	0.00932	−2.59808
ACC-1 (SWIPE vs. NLS)	22.5	0. 00466	−2.82902
ACC-2 (OBP vs. SWIPE)	40.5	0.07346	−1.78979
ACC-2 (OBP vs. NLS)	18	0.002	−3.08882
ACC-2 (OBP vs. YIN)	51.5	0.25014	−1.1547
ACC-2 (YIN vs. SWIPE)	72	0.97606	0.02887
ACC-2 (YIN vs. NLS)	46.5	0.14986	−1.44338
ACC-2 (SWIPE vs. NLS)	52	0.25848	−1.12583
ACC-10 (OBP vs. SWIPE)	65	0.70394	0.37528
ACC-10 (OBP vs. NLS)	61.5	0.56192	−0.57735
ACC-10 (OBP vs. YIN)	55.5	0.35758	0.92376
ACC-10 (YIN vs. SWIPE)	67	0.79486	−0.25981
ACC-10 (YIN vs. NLS)	46.5	0.14986	−1.44338
ACC-10 (SWIPE vs. NLS)	54	0.3125	−1.01036

Figure 6 shows a sample of the RAE plotted for each signal within two categories of the dataset, as an RAE vs. frequency plot, where it can be appreciated how OBP performs in a solid way throughout all the categories, to the point where, for complex signals like those included in “full2”, it actually brings lower errors than most of the other algorithms; it can also be observed how most of the errors in NLS are octave errors (integer RAE). The exemplified categories are “2harm” and “full2” in order to show the behavior of the four algorithms on clean vs. unclean signals.



(**a**)



(**b**)

Figure 6. Example of RAE (percentage error) vs. frequency plots for two categories (a very clean one vs. a very complex one) of the SYNTHPITCH dataset. Frequencies span from 100 Hz to 5000 Hz with a step size of 50 Hz. (**a**) "2harm" category (blue); (**b**) "full2" category (red).

All the analyses, data creation, tests, simulations and algorithm implementations were performed using MATLAB® R2023a (by Mathworks Inc., Natick, MA, USA [43]) on a Dell Latitude E5550 computer, with an Intel Core i5 5200U processor and a 16 GB Dual-Channel DDR3 RAM.

Figure 6 details an example of the percentage error (RAE) with respect to the frequency, i.e., the different signals within a category of the SYNTPITCH dataset.

The time elapsed (TE) was empirically shown to be independent of the frequency of the input; plots are thus not shown as they are erratic and only depend on the randomness within the signals, given for example by the white Gaussian noise. TE was also shown to be independent of the dataset category, i.e., from the pitch complexity of the input signal.

4. Discussion

The premises of this study, besides the presentation of the custom-made SYNTHPITCH dataset, were to implement an ultra-fast pitch detection algorithm for real-time applications and ease of hardware implementation. Our proposed algorithm, OneBitPitch (OBP), despite running on a software environment, confirms the premises by providing by far the fastest results in terms of time elapsed for pitch detection. On the other hand, as is expected due to its nature, SWIPE is the heaviest algorithm and takes an average of 256 ms to be executed, despite the "large" window frame selected and the SWIPEP variant.

YIN, considered one of the fastest pitch detection algorithms, being based on a modified autocorrelation, was about nine times faster than SWIPE, with only a 38 ms average. NLS needs a special mention, as its accuracy values are outstanding, and, on synthetic signals, its time performances are too, with only a 27 ms average.

However, OBP shows its real strength with an average elaboration time of only 4.6 ms. It is 50 times faster than SWIPE and even 9 times faster than YIN, which is considered a fast algorithm. Figure 7 displays an alternative visualization of the performance of the four algorithms in terms of speed, plotting the elapsed time, which immediately shows the differences in speed.

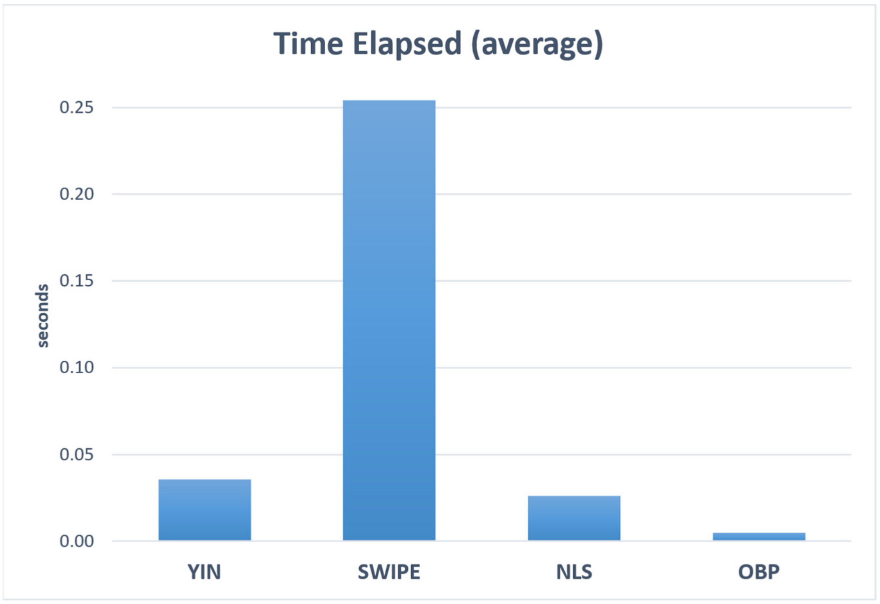


Figure 7. Average elapsed time (in seconds) for each algorithm. See Table 3 for numeric data.

Other works such as that by Grinewitschus et al. [21] report speed as a multiplier of the "realtime", which is the duration of the audio segment to be evaluated. Their approach, based on leveraging the constant-Q Gabor Transform followed by harmonic shift algorithms and corrective heuristics, reaches a declared 0.29 × realtime speed on a more powerful machine than what has been used in the present work; nonetheless, OBP reaches 0.046 × realtime.

The choice of the right algorithm for a specific application is mainly based on the latency and accuracy tradeoff required, and the computational power available, so it is safe to say that each of the proposed algorithms has a specific field and reason to be applied.

Looking at the RAE and accuracies, the NLS appears as the best-performing algorithm, always providing less than 1% RAE on average, followed by SWIPE. It is, however, worth noting that the NLS was one of the algorithms suffering the most from octave errors, especially when processing noisy signals or square waves. Moreover, this test confirms the noise-robust nature of SWIPE, performing well even with harsh signals like square waves and/or noise.

Looking at the accuracy tables for ACC-1, ACC-2 and ACC-10, NLS confirms its performances by always providing the highest accuracy, at 93%. However, this value stays the same for all three ranges, which indicates that all the errors reported by NLS within these tests were greater than 10% in RAE, which is a good indicator of a certain proneness to octave errors.

Although OBP underperforms when it comes to 1% accuracy, that is to be expected: only 42% of the time was the error smaller than 1%. On the other hand, its performances become comparable to YIN and SWIP for ACC-2, which represents the acceptability range. The discrete nature of musical notes within the Western tempered system is so that a 2.5% error still leads to a discretized pitch being correct.

For larger-scale errors, OBP actually brings better ACC-10 values than YIN and SWIPE, with more than 90% accuracy.

TE can be considered as the most crucial metric for the present paper, due to the improvement that OBP aims to bring, which is specifically in terms of speed.

Within this context, the differences between all of the algorithms considered, especially OBP, have been proven to be statistically significant with a Mann–Whitney test yielding a U close to 0, which brings $p < 0.00001$, which is much lower than the desired threshold of 0.05. A z-score < -4 points to a confidence higher than 99.997%.

RAE and ACC-1 also generally present statistical significance in the comparisons, especially when OBP is involved. On the other hand, results regarding ACC-10 are not statistically significant (all of the algorithms perform similarly).

In general, differences between YIN and SWIPE within this picture appear to bear less statistical significance, being only significant for the TE—which is the most crucial metric for the scope of this study.

These results in terms of time vs. accuracy are completely in line with the premises, since OBP was designed to be minimalistic, forgoing sheer, pinpointed accuracy (hence the lower ACC-1 values) but still staying within an acceptable range for most applications (hence the higher ACC-2 values), with an inherent observed robustness with respect to signal variations and to octave errors.

It is worth noting that despite the average accuracy, the average RAE of OBP is comparable to the YIN algorithm.

Moreover, all the presented algorithms have been used in one fixed setup with a fixed threshold. However, in real-world applications, hyperparameters can be tuned to better adapt to the nature and variation of the input: in fact, manually changing the threshold of the OBP algorithm provides better performances on certain datasets, especially the noisiest.

The noisiest, most unclean dataset categories are full1, full2, 4harm_4part_wgn15 and 4harm_high: on these sets, most algorithms provide poor performances. However, the average RAEs that would be obtained on all of the other sets are as follows:

- YIN: 0.077680, i.e., 7.7% average error;
- SWIPE: 0.007548, i.e., 0.7% average error;
- NLS: 0, i.e., no errors. Due to its heuristic-powered nature, NLS is able to pinpoint discrete frequencies on synthetic datasets;
- OBP: 0.018396, i.e., 1.8% average error.

On cleaner signals, such as those produced by many musical instruments that do not have added noise, OBP actually performs better in terms of error than YIN, while providing more than acceptable results overall.

The minimalistic, stripped-down nature of OBP does not allow it to reach almost-perfect accuracy levels; however, it is critical to assess its "acceptability" ranges, given that it is by far the fastest [44].

YIN, on the other hand, is a fast and simple algorithm. It suffered from octave errors 12% of the time, but when the estimation was right, the precision was satisfying, with an 87% chance of obtaining a result with less than 1% error.

The OBP algorithm presents a peculiar behavior because it has only a 4% chance of suffering from octave errors.

An advantage of the mathematical model behind OBP over other similar autocorrelation-based methods is that, being based on the product of signals, the output heavily depends on the magnitude of the input and its variations. Because of that, an additional normalization process is usually needed, increasing the computational complexity, requiring knowledge of the energy of the signals and adding more multipliers.

On the other hand, OBP only processes sign signals, inherently independent from the original magnitude, removing the need for explicit normalization procedures.

Despite the performances reported in these tests, different environments and data see the algorithms behave differently, at least in terms of accuracy. In fact, SWIPE is one of the gold standards for feature extraction for medical or Machine Learning purposes, due to it being well suited for noisy environments and due to the applications not needing real-time elaboration. On the other hand, NLS is not robust with respect to the nature of the input and is more prone to suffer from octave errors, making it sometimes an inconsistent choice despite the high performances on the proposed synthetic dataset.

This study employed the SYNTHPITCH dataset because the large-scale speed was the main indicator to be evaluated; however, future steps will evolve around the experimentation on real-world, validated data for pitch detection, which also leads to the application of pitch-tracking, to follow in real-time the varying pitch of a real sound/speech signal [45].

Future Works

We are currently working on expanding the present experimentation with other test datasets, focusing especially on real-world scenarios such as sounds from real instruments (as those employed for MIDI conversion) or vocal signals.

Technically speaking, the most likely future implementations of OBP will definitely focus on hardware implementation, due to its bitwise nature and maximum speed. Its simple structure and the low usage of memory and computations make it perfectly suitable for a hardware-only implementation—for example, with a DSP—to exploit its speed capabilities and inherent characteristics suitable for digital electronics. A future FPGA implementation is foreseen, aimed to produce a stand-alone IpCore that, using just a small amount of logic, can provide ultra-low-latency pitch tracking [46,47]. With the increasing trend and the low cost/low performance of System on a Chip (SoC) technologies, the OBP algorithm can be a perfect candidate for wearable electronic, embedded music processors for Autotune or real-time MIDI conversion [48], as well as IoT applications, surveillance or vocal recognition. We can thus summarize some of the future directions of OBP as follows:

- Test on real-world datasets, especially within the professional audio/musical department;
- Full-hardware FPGA implementation;
- Addition of optional features such as posterior (heuristic) correction for added accuracy at the cost of speed, selectable N-bit expansion, etc.

5. Conclusions

In this paper, a novel algorithm was proposed for ultra-fast pitch detection for real-time applications, based on a modified autocorrelation implemented on a single-bit signal. The OneBitPitch (OBP) algorithm was compared with the most widely used models for high-speed F0 detection, namely, YIN, SWIPE and an NLS-based implementation. Additionally, a custom dataset made of synthetic waves has been proposed and made available to the public: the SYNTHPITCH dataset encompasses sinusoidal and square waves with added artifacts for an increasing pitch complexity, realized through the addition of harmonics, partials, white noise and reverb.

OBP is shown to be an ultra-fast, reliable pitch detection algorithm for real-time applications: it has been built with a minimalist approach that aims to reduce all computationally expensive steps in pitch detection, which are mainly represented by a multi-bit elaboration, the transformation/metric production and the eventual presence of corrective heuristics.

The focus of the OBP algorithm is solely on speed, provided that acceptable accuracy levels are reached, and its characteristics make it exceptionally suitable for an easy and lightweight hardware implementation on FPGA or SoC, for ever-higher customization possibilities as well as lower latencies.

The comparison of different state-of-the-art algorithms shows that OBP is 9 times faster than the peak speed of the other algorithms (namely, YIN) and 50 times faster than SWIPE. OBP is shown to be the fastest pitch detection algorithm within the presented test, with only 4.6 ms of mean elapsed time on each data instance, or 0.046 $\times$ realtime runtime, which is the lowest reported to date.

On the other hand, although relative error might be increased for certain datasets, OBP is less prone to octave errors, and in general demonstrates the ability to stay within the acceptability range on most of the tested signals. The main compromise was between speed and accuracy, and OBP stays within acceptable ranges of 2% accuracy, with a 72% average that peaks at almost 82% for less noisy signals, which is enough for most discrete-note musical applications. The NLS-based algorithm is the most accurate one, although it is less robust to noise or input variations and requires a prior building of a model; in fact, SWIPE is one of the most employed algorithms in real-world applications not centered on speed. Additional tests are needed on real-world signals, such as validated voices of known F0, and a hardware implementation, with more parameters and selectable options is foreseen for the OBP algorithm.

Author Contributions: Conceptualization, D.C.; methodology, D.C. and V.C.; software, D.C. and V.C.; validation, V.C. and G.C.; formal analysis, V.C.; investigation, D.C. and V.C.; resources, V.C.; data curation, V.C.; writing—original draft preparation, D.C.; writing—review and editing, V.C.; visualization, D.C. and V.C.; supervision, G.C.; project administration, G.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The proposed dataset SYNTHPITCH of synthetic signals for testing pitch detection algorithms is available at the following link: https://drive.google.com/drive/folders/1YP15ULjyyeI27k_2wPzWFxuig7fbSveb?usp=sharing (accessed on 10 July 2023).

Acknowledgments: The authors would like to thank Voicewise S.r.l. for the support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ruslan, N.; Mamat, M.; Porle, R.; Parimon, N. A Comparative Study of Pitch Detection Algorithms for Microcontroller Based Voice Pitch Detector. *Adv. Sci. Lett.* **2017**, *23*, 11521–11524. [CrossRef]
2. Qurthobi, A.; Maskeliūnas, R.; Damaševičius, R. Detection of Mechanical Failures in Industrial Machines Using Overlapping Acoustic Anomalies: A Systematic Literature Review. *Sensors* **2022**, *22*, 10. [CrossRef] [PubMed]
3. Kim, J.; Kim, J.; Nguyen, L.T.; Shim, B.; Hong, W. Tonal signal detection in passive sonar systems using atomic norm minimization. *EURASIP J. Adv. Signal Process.* **2019**, *2019*, 43. [CrossRef]
4. Krajewski, M.; Sienkowski, S.; Kawecka, E. Properties of selected frequency estimation algorithms in accurate sinusoidal voltage measurements. *Prz. Elektrotechniczny* **2018**, *94*, 52–55.
5. Teixeira, J.P.; Oliveira, C.; Lopes, C. Vocal Acoustic Analysis—Jitter, Shimmer and HNR Parameters. *Procedia Technol.* **2013**, *9*, 1112–1122. [CrossRef]
6. Bharathi, V.; Abraham, A.; Ramya, R. Vocal pitch detection for musical transcription. In Proceedings of the 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies, Thuckalay, India, 21–22 July 2011. [CrossRef]
7. Hildebrand, H.A. Pitch Detection and Intonation Correction Apparatus and Method. U.S. Patent No. 5,973,252, 26 October 1999.

8. Costantini, G.; Cesarini, V.; Robotti, C.; Benazzo, M.; Pietrantonio, F.; Di Girolamo, S.; Pisani, A.; Canzi, P.; Mauramati, S.; Bertino, G.; et al. Deep learning and machine learning-based voice analysis for the detection of COVID-19: A proposal and comparison of architectures. *Knowl. Based Syst.* **2022**, *253*, 109539. [CrossRef] [PubMed]
9. Cesarini, V.; Robotti, C.; Piromalli, Y.; Mozzanica, F.; Schindler, A.; Saggio, G.; Costantini, G. Machine Learning-based Study of Dysphonic Voices for the Identification and Differentiation of Vocal Cord Paralysis and Vocal Nodules. In Proceedings of the 15th International Conference on Bio-inspired Systems and Signal Processing, Online, 9–11 February 2022; pp. 265–272. [CrossRef]
10. Costantini, G.; Parada-Cabaleiro, E.; Casali, D.; Cesarini, V. The Emotion Probe: On the Universality of Cross-Linguistic and Cross-Gender Speech Emotion Recognition via Machine Learning. *Sensors* **2022**, *22*, 7. [CrossRef]
11. Costantini, G.; Cesarini, V.; Di Leo, P.; Amato, F.; Suppa, A.; Asci, F.; Pisani, A.; Calculli, A.; Saggio, G. Artificial Intelligence-Based Voice Assessment of Patients with Parkinson's Disease Off and On Treatment: Machine vs. Deep-Learning Comparison. *Sensors* **2023**, *23*, 4. [CrossRef]
12. Amato, F.; Saggio, G.; Cesarini, V.; Olmo, G.; Costantini, G. Machine learning- and statistical-based voice analysis of Parkinson's disease patients: A survey. *Expert Syst. Appl.* **2023**, *219*, 119651. [CrossRef]
13. Fant, G. *Acoustic Theory of Speech Production*; Walter de Gruyter: Berlin, Germany, 1970.
14. Hermes, D. Measurement of pitch by subharmonic summation. *J. Acoust. Soc. Am.* **1988**, *83*, 257–264. [CrossRef]
15. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2012; Available online: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html (accessed on 28 February 2023).
16. Illner, V.; Sovka, P.; Rusz, J. Validation of freely-available pitch detection algorithms across various noise levels in assessing speech captured by smartphone in Parkinson's disease. *Biomed. Signal Process. Control.* **2020**, *58*, 101831. [CrossRef]
17. Su, H.; Zhang, H.; Zhang, X.; Gao, G. Convolutional neural network for robust pitch determination. In Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 20–25 March 2016. [CrossRef]
18. Khadem-hosseini, M.; Ghaemmaghami, S.; Abtahi, A.; Gazor, S.; Marvasti, F. Error Correction in Pitch Detection Using a Deep Learning Based Classification. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2020**, *28*, 990–999. [CrossRef]
19. Camacho, A.; Harris, J.G. A sawtooth waveform inspired pitch estimator for speech and music. *J. Acoust. Soc. Am.* **2008**, *124*, 1638–1652. [CrossRef]
20. De Cheveigne, A.; Kawahara, H. YIN, a fundamental frequency estimator for speech and music. *J. Acoust. Soc. Am.* **2002**, *111*, 4. [CrossRef]
21. Grinewitschus, L.; Jung, P. The Harmonic Shift Algorithm for Efficient Multi-Pitch Detection. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2022**, *30*, 548–561. [CrossRef]
22. Mnasri, Z.; Rovetta, S.; Masulli, F. A Novel Pitch Detection Algorithm Based on Instantaneous Frequency. In Proceedings of the 2021 29th European Signal Processing Conference (EUSIPCO), Dublin, Ireland, 23–27 August 2021; pp. 16–20. [CrossRef]
23. Zahorian, S.; Dikshit, P.; Hu, H. A spectral-temporal method for pitch tracking. In Proceedings of the Ninth International Conference on Spoken Language Processing, Pittsburgh, PN, USA, 17–21 September 2006. [CrossRef]
24. Staudacher, M.; Steixner, V.; Griessner, A.; Zierhofer, C. Fast fundamental frequency determination via adaptive autocorrelation. *EURASIP J. Audio Speech Music. Process.* **2016**, *2016*, 17. [CrossRef]
25. Kim, J.; Salamon, J.; Li, P.; Bello, J. CREPE: A Convolutional Representation for Pitch Estimation. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018.
26. Kay, S.M. *Fundamentals of Statistical Signal Processing*; Prentice Hall Signal Processing Series; Prentice-Hall PTR: Englewood Cliffs, NJ, USA, 1993.
27. Kehtarnavaz, N. Analog-to-Digital Signal Conversion. In *Digital Signal Processing System Design*; Elsevier: Amsterdam, The Netherlands, 2008; pp. 57–91. [CrossRef]
28. Host-Madsen, A.; Handel, P. Effects of sampling and quantization on single-tone frequency estimation. *IEEE Trans. Signal Process.* **2000**, *48*, 650–662. [CrossRef]
29. Apicella, B.; Bruno, A.; Wang, X.; Spinelli, N. Fast Fourier Transform and autocorrelation function for the analysis of complex mass spectra. *Int. J. Mass Spectrom.* **2013**, *338*, 30–38. [CrossRef]
30. Ortigueira, M. On the estimation of the autocrrelation function. *Discuss. Mathematicae. Probab. Stat.* **2010**, *30*, 103–115. [CrossRef]
31. Hess, W. *Pitch Determination of Speech Signals*; Springer Series in Information Sciences; Springer: Berlin/Heidelberg, Germany, 1983; Volume 3. [CrossRef]
32. Granlund, T. Instruction Latencies and Throughput for AMD and Intel x86 Processors 2019. Online x86-Timing.pdf. Available online: https://gmplib.org/ (accessed on 20 June 2023).
33. Dodmane, R.; Aithal, G.; Shetty, S. Construction of vector space and its application to facilitate bitwise XOR—Free operation to minimize the time complexity. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 9836–9843. [CrossRef]
34. Kawecka, E.; Podhajecki, J. Probabilistic Properties of Deterministic and Randomized Quantizers. *Procedia Comput. Sci.* **2022**, *207*, 754–768. [CrossRef]
35. *IEEE Std 754-2019 (Revision of IEEE 754-2008)*; IEEE Standard for Floating-Point Arithmetic. IEEE: New York, NY, USA, 2019; pp. 1–84. [CrossRef]
36. Samavi, S. *Representing Signed Numbers*; McMaster University: Hamilton, ON, Canada, 2014.

37. Tsanas, A.; Little, M.; Mcsharry, P.; Ramig, L. New nonlinear markers and insights into speech signal degradation for effective tracking of Parkinson's disease symptom severity. In Proceedings of the International Symposium on Nonlinear Theory and Its Applications (NOLTA), Krakow, Poland, 5–8 September 2010; pp. 457–460.
38. Teunissen, P. Nonlinear least-squares. *Manuscripta Geod.* **1990**, *15*, 137–150.
39. Marquardt, D.W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *J. Soc. Ind. Appl. Math.* **1963**, *11*, 431–441. [CrossRef]
40. Wang, D.; Wei, Y.; Wang, Y.; Wang, J. A Robust and Low Computational Cost Pitch Estimation Method. *Sensors* **2022**, *22*, 16. [CrossRef] [PubMed]
41. Nachar, N. The Mann-Whitney U: A Test for Assessing Whether Two Independent Samples Come from the Same Distribution. *Tutor. Quant. Methods Psychol.* **2008**, *4*, 13–20. [CrossRef]
42. Kirch, W. (Ed.) Pearson's Correlation Coefficient. In *Encyclopedia of Public Health*; Springer: Dordrecht, The Netherlands, 2008; pp. 1090–1091. [CrossRef]
43. The MathWorks Inc. *MATLAB Version: 9.13.0 (R2022b)*; The MathWorks Inc.: Natick, MA, USA, 2022; Available online: https://www.mathworks.com (accessed on 10 June 2023).
44. Hess, W. Pitch and Voicing Determination of Speech with an Extension Toward Music Signals. In *Springer Handbook of Speech Processing*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 181–212. [CrossRef]
45. Host-Madsen, A.; Händel, P. The effect of sampling and quantization on frequency estimation. In Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181), Seattle, WA, USA, 15–15 May 1998; Volume 4, p. 2220. [CrossRef]
46. Temple, A.R. Real-Time FPGA Implementation of a Neuromorphic Pitch Detection System. Ph.D. Thesis, Loughborough University, Loughborough, UK, 1999. Available online: https://hdl.handle.net/2134/13610 (accessed on 20 May 2023).
47. A Simplified Speaker Recognition System Based on FPGA Platform | IEEE Journals & Magazine | IEEE Xplore. Available online: https://ieeexplore.ieee.org/document/8897096 (accessed on 28 February 2023).
48. Monti, G.; Sandler, M. Monophonic transcription with autocorrelation. In Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00), Verona, Italy, 7–9 December 2023; pp. 257–260.