Transactions on Engineering and Computing Sciences - Vol. 12, No. 3
Publication Date: June 25, 2024
DOI:10.14738/tecs.123.16927.
Michael, C. (2024). Singing Voice Melody Detection. Transactions on Engineering and Computing Sciences, 12(3). 09-17.
Services for Science and Education

Singing Voice Melody Detection

Chourdakis Michael
University of Athens

ABSTRACT

This is an enhancement on our time domain pitch detection algorithm (Anonymous 2007), optimized for voice singing. Various time and frequency domain methods have been researched to detect the pitch of the human voice. This manuscript describes our methods to convert human singing to music notation. Over the past years, significant progress is made in speech recognition, but methods used in that process recognize textural patterns, not melody. We use a combination of time and frequency domain algorithms to achieve our aim using C++. The results of our algorithms are satisfactory and accurate enough, not only to represent human voice singing in the European notation, but also to represent it when higher resolution script is available - for example, when using the Byzantine Script which has a minimum note distance of about 17 Cents (as opposed to the European notation with a minimum note distance of 100 Cents).

OTHER ALGORITHMS

In this section we will describe our results with known pitch detection algorithms. There are two major categories, "time domain" in which the algorithm works mostly in a function of time and "frequency domain" in which the algorithm works mostly in a function of frequency. We will write a quick review on these algorithms and later we will compare their results with our own.

The Fundamental Problem of a Window

Most pitch detection algorithms suffer by a fundamental problem, the choosing of a Window (J. Smith 2010). A Window is an area of a signal in which we will work at. If the area is too small, then we cannot be sure if the frequency contained there is completed as a function of time. If the area is too large, then we cannot know if there is only one frequency inside or perhaps more.

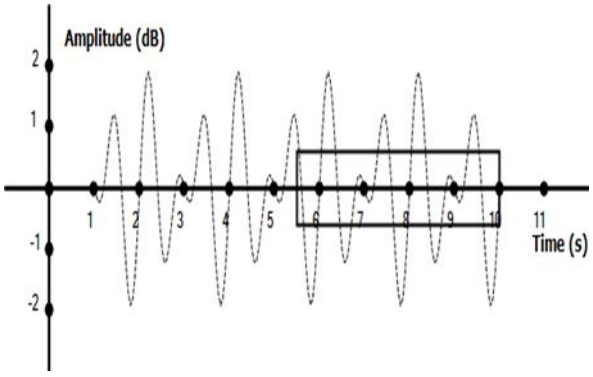


Fig 1: Improper picking of a section to analyze. The signal time is greater than the windows' time.

Time Domain

These algorithms work mainly in the time domain, that is, in a function like $S(t) = A_0 \sin(2\pi f_0 t) + A_1 \sin(2\pi f_1 t) + \ldots + A_v \sin(2\pi f_v t)$ **(1).**

Direct Attribute Count Methods:

These algorithms count the basic properties of the signal (zero crossings, peaks, or phase) directly. They work only on very simple signals which is not the case in most vocal signals. A plain Zero Crossing approach (Anonymous 2006) counts the crossings of the signal with the X-axis and divides by period to find the frequency. Obviously, this will only work in very simple signals like the plain sine wave. For something more complex like the signal below, we can see that if the maximum time is 10 seconds, then the frequency of the signal is 1Hz, but the zero crossings are 20.

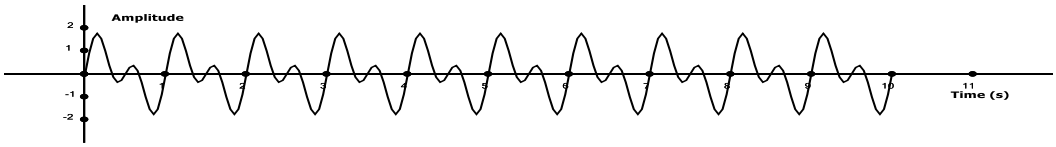


Fig. 2: $f(t) = \sin(2\pi\tau) + \sin(4\pi\tau)$. Zero crossings cannot return reliable results.

The "Peak Detection" method, closely related to Zero Crossings (Kedem 1986) counts the peaks instead of the zero crossings. Theoretically this can be more accurate, but in practice we very often have signals of the following type:

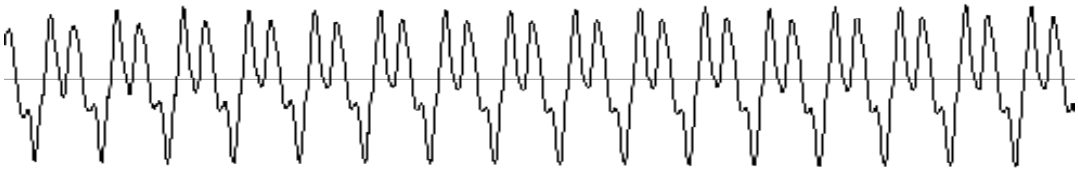


Fig. 3: Adult male voice.

In the above pattern, we don't know if we are about to count the positive or the negative peaks. The plain Peak Detection algorithm only plays slightly better in results when compared to the plain Zero crossings. The Phase Detection algorithm (Gibiat 1988) relies on counting phase differences. For each type $f(t) = Asin2\pi ft$ signal there is a phase movement from 0 to 2π (a circle) for each period T = 1/f and therefore, to calculate the frequency. This again works on signals that have a pretty much constant frequency to estimate.

Related Signal Methods:

A correlation of two signals is the similarity check of a function by itself, when time is shifted by τ, defined as follows: $(f \bowtie g)(t) = \int_{-\infty}^{+\infty} f^*(t)g(\tau + t)$ **(2).** Autocorrelation (Tan and Karnjanadecha 2003) occurs when the signal is compared to itself (f = g). If the function has a period T, then for $t \in \left[0, \frac{T}{2}\right]$ the autocorrelation function decreases and for $t \in \left[\frac{T}{2}, T\right]$ it increases. By measuring the points that this increases and decreases we can calculate the frequency. This works only in "clean" signals that do not vary significantly in the recording and

therefore, it's not recommended for a human voice. For example, in the signal of a 70ms at 164Hz recorded below:

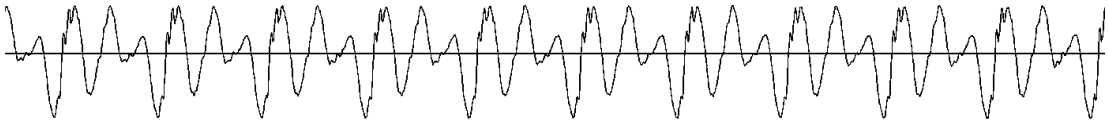


Fig. 4: Voice recorded with a tone of 164 Hz.

The correlation function returns this signal:



Fig. 5: Autocorrelation function of the signal in Fig. 4.

Which returns an incorrect frequency result as the above signal peak count would result in 122 Hz. YIN, a specialization of the autocorrelation (Cheveigné and Kawahara 2002) function tries to resolve various autocorrelation errors in high frequencies but still suffers from the general autocorrelation issues as we saw above. pYIN, a new version of YIN still suffers from the autocorrelation issues but enhances the "melody aspects" of YIN (Mauch and Dixon 2014).

Frequency Domain

These algorithms work mainly in the frequency domain, that is, in the Fourier transformations of the time signal: $F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}dt$ (**3**) and $F(\omega) = \sum_{i=-\infty}^{+\infty} f(t)e^{-i\omega t}$ (**4**).

Fast Fourier Transform:

The generic Fast Fourier Transform function suffers from a "cut" in the signal to be represented known as the Gibbs phenomenon (Gibbs 1899) that occurs when the frequency to be analyzed contains frequencies that are not exact multiplications of the fundamental frequency (which is the case of the human voice). Also, the digital (discrete) Fourier transform can only return us results for specific frequencies (for these that returned from the following formula when j is integer: $f_j = \frac{jSR}{N}, j \in N, j \in [0, \frac{M}{2} - 1]$ **(5)**) (Marchand 2001). All algorithms based on the frequency domain suffer from these fundamental inaccuracies.

Cepstrum:

The Cepstrum approach (Noll 1964) apples the Fourier transform to the logarithm of the signal: $|F(\log(\left|F\left(f(t)\right)\right|^2)|^2$ (**6**)**,** on the theory that a linear evaluation of the frequency is more easily analyzed than a logarithmic one. This method works better in speech recognition but not in human signing melody recognition.

Maximum Likelihood:

The max likelihood (Doval and Rodet 1993) is a statistical hypothesis of the fact that, on a given signal, there is more probability that it was originally composed from a series of exact

multiplications than otherwise. For example, a signal that contains 110Hz, 220Hz and 312Hz is more probable to be composed by 110 Hz (since 220 is an exact multiply). This method applies better to signals that indeed contain exact (or close) multiplications of the fundamental f0, but in human voice this is most cases is not valid.

Wavelets:

The Wavelet approach (Chan and Shen 2005) tries to resolve Window errors by replacing Fourier Transform with a variable-size Window. At high frequencies, a smaller window is applied (so there is a better time resolution) where, at low frequencies, a larger window is applied (so there is a better frequency resolution). When analyzing human voice, we cannot know beforehand if the voice is high or low, so the wavelet functions are featuring worse, comparing to our algorithm.

OUR ALGORITHM

After a quick examination of pitch detection algorithms and their features, we can now present our methods.

Noise Reduction

We need a signal as clean as possible from environmental noise. The better the hardware and the environment used to record, the better results we can achieve.

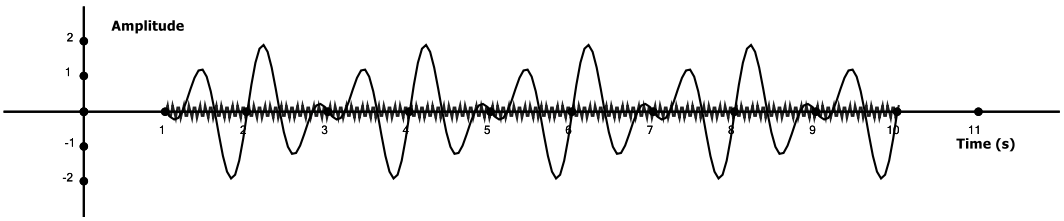


Fig. 6: Noisy signal.

To clean the signal, we first define what "noise" is (by recording air and taking the Fourier Transform of the signal) then we apply a "noise gate" to lower the frequencies of the noise, which are usually different than human signing (Davis and Jones 1988). The following table shows our results for various levels of signal cleanliness:

Signal clean %	Algorithm results in next step
95%	95,4%
75%	78.2%
45%	36.6%
25%	23.3%
15%	2.6%

Filtering

By applying a low pass filter (either Chebyshev or a Windowed-SINC (Hamming 1997)) we can cut out higher unneeded frequencies.

Our algorithm can be hinted on what is recording (for example, Bass, Tenor, Alto or Soprano) to cut higher or lower as per voice.

Signal Simplification

A simple silence detection is the first step in melody segmentation:

1. Apply a noise gate to segment the melody (step 1 segmentation):
2. Count Zero Crossings and find maximum possible frequency
3. Removal of redundant peaks. At the function f(t) = Asin($2\pi ft$), for each Zero Crossing we have a maximum peak and a minimum peak. In case there are more, they are combined to one (to the maximum of them) and the signal is simplified.

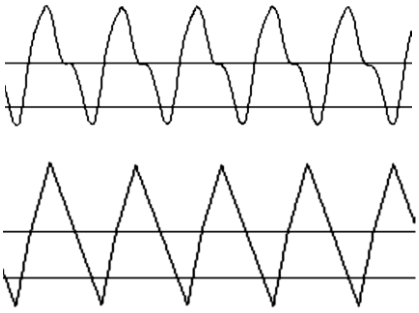


Fig. 7: Before and after signal simplification

In creating a C++ loop of the above formula, we input:

1. A zero-crossing count per loop, which we have found in the human frequency area that a value of 24 ZCs per loop is efficient,
2. An accepted-stability parameter (0 to 1), the greatest variation of the peaks between them (in which a female voice parameter is 0.5 a male voice parameter is 0.3)

Attack Detection

By the sound assumption that the human voice has an attack and a release value, we apply an attack detection algorithm (a statistical count of the peak increase) to assume that a note starts at that point with a fade in (Step 2 segmentation). This is aided by a reverse release-detection algorithm (a statistical count of the peak decrease) to assume that the 'note' ends at a position with a fade out.

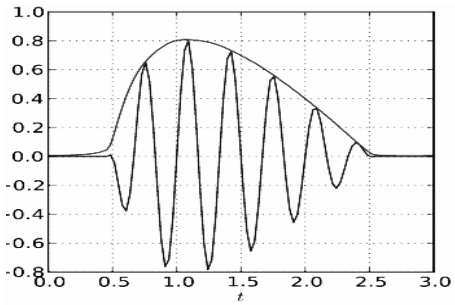


Fig. 8: Attack and release detected

Main Algorithm Loop

1. Assume a signal with the 24 zero crossings as noted before and with the specific stability parameter,
2. Find zero crossings and peaks,
3. Is there a major silence in this area? If yes, assume this segment belongs to the note previously detected, else continue,
4. Peak detection using the function below:
    - Normalize peaks
    - Find average of their amplification and the standard deviation of their peak differences. If all peaks have the same amplitude, standard deviation = 0 (while excluding fade-in and fade-out as described in the previous paragraph).
    - Find variation of their positions and compare with the autocorrelation function to get an idea of how much the signal is actual 'periodic'

The above formula results to this:

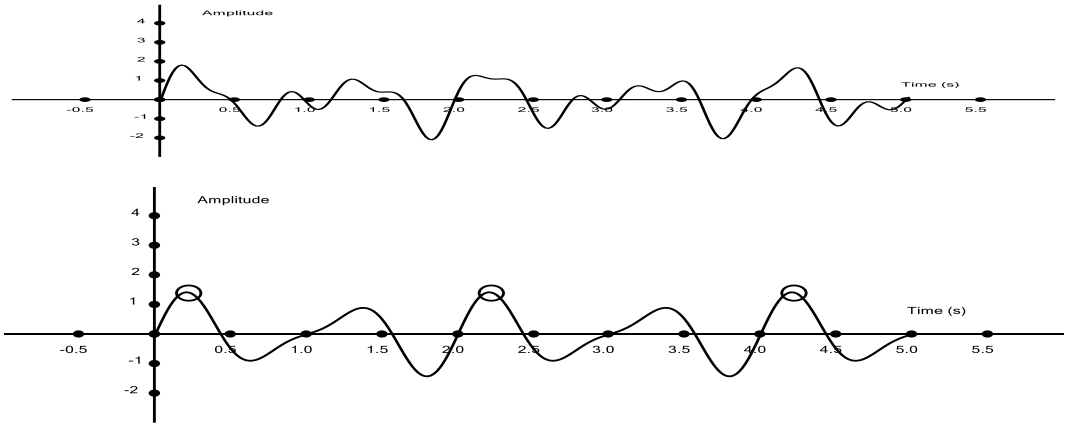


Fig. 9: Chosen peaks

In which we pick the result that gives the smaller standard deviation between the peaks. If the result decides that there are more peaks than how many were counted, then the signal has more frequencies inside which were detected and eliminated.

1. Pick positive or negative peaks. If positive peaks have a greater standard deviation, pick the negative peaks (and vice versa). If more frequencies were detected in the signal after application of the FFT, then the peaks that are closed to the fundamental frequency are chosen.
2. If the above method still does not suffice, we analyze the whole recording and pick the class of peaks that occur more frequently or have more amplitude.
3. If the peak collection has a standard deviation greater than a maximum threshold (defined for male voice higher than a female or child voice since the male voice has lower frequencies) then the algorithm starts cutting peaks until this threshold is reached (Anonymous 2007).
4. If the above still gives us unsatisfactory results, then we take the attack and release sections to determine which peak collection we can work with.

5. Segment frequency occurs from the function $F = (num_peaks - 1) * (SR/TS)$. SR is the sampling frequency and TS is the number of samples between the first and the last picked peak.
6. If the FFT transform returns a frequency greater than the double of ours, then it's discarded (detected as non f0).
7. If the FFT transform returns a frequency lower than ours, then we divide by 2. For example, if we have calculated F = 259Hz and FFT returns 125Hz, then we assume 259/2 = 129.5Hz. If this phenomenon does not repeat in the next or previous segments, then an average is taken (since it's unlikely that the human voice jumps octaves up and down).
8. Fade analysis now occurs:
    - Two close segments are checked, if the one's amplitude is greater than the others (by a specified threshold), then a fade-in or fade-out is placed.
    - We check for vibrato (Driedger, et al. 2016), that is, the standard deviation of segments in close frequency is below a threshold.
    - We check for glissando, that is, a frequency increase and the amplitudes are very close (no attack or release).
9. We join similar – frequency segments by a threshold – a resolution depending on what sort of analysis we are performing. For example, in standard European major/minor scale, there is a resolution of 100 cents (1 semitone). In Greek Byzantine Music (Savas 1965) the resolution is 17 cents.
10. We re-check for observable jumps due to false peak detection. For example, when we have many segments of 220Hz and one "jump" to 440Hz we discard it, and we join it with the 220Hz segments.
11. We clamp frequencies if there's a hint of the voice analyzed (the frequency range of, for example, the Bass voice cannot go more than 400 Hz).
12. We check for vibrato (Yang 2016), glissando (Panda, Malheiro and Paiva 2018) and fades on the joined segments.
13. We quantize the segments based on any time resolution hint (for example, if it's known that the melody contains only quarter notes it's not likely that a note will have a 100ms duration).

RESULTS & COMPARISONS

The following table gives the results of our methods compared to other pitch detection algorithms:

Method	Adult Male	Adult Female	Child
Plain Zero Crossings	12,4%	35.2%	47.5%
Plain Peak Detection	8.2%	36.1%	38.5%
Phase Detection	6.2%	31.5%	31.2%
Autocorrelation	3.5%	12.4%	16.2%
Plain FFT	10.2%	15.5%	18.6%
Cepstrum	9.5%	12.5%	17.2%
Maximum Likelihood	17.5%	39.5%	43.5%

Wavelets	11.5%	32.5%	55.5%
YIN	17.2%	42.2%	66.2%
pYIN	21.5%	48.5%	68.3%
Our method	**57.3%**	**76.5%**	**88.6%**

The results above are generated by a total of 1500 human voice experiments in various environments, hardware, melody types and note resolution (Major, Minor or Byzantine). Our algorithm is capable of very precise frequency detection (since it's not suffering from the FFT-based methods, and it combines segments based on fade and statistic patterns rather than a fixed value Window). Errors in our algorithm occur from:

1. Unusual high amplitude of harmonics when compared to the fundamental f0,
2. Unstable human voice with lots of vibrato and/or glissando,
3. Too fast melody (which reduces the statistical data to work with), for example melody with a 1/16-time notes in 120bpm,
4. Low amplitude sound, which reduces the quality of the noise-removal process,
5. Very low frequency (a deep-bass voice in the lower notes), which increases the likelihood that the recording picks many higher frequencies and decreases the quality results of the filtering.
6. Low recording quality as described earlier.

Error Type	**Male Voice**	**Female Voice**	**Child Voice**
High harmonics	70% worse	30% worse	20% worse
Unstable voice	70% worse	40% worse	22% worse
Low amplitude	80% worse	60% worse	30% worse
Low frequency	90% worse	10% worse	5% worse
Fast Melody	70% worse	50% worse	45% worse
Low recording quality	80% worse	60% worse	45% worse

References

(Author). 2007. "Time domain pitch recognition." *Music & Informatics 4o SMC.* Lefkada.

(Author), and Charalambos Spyridis. 2006. «Zero Crossings Tune.» *11o EEF Physics*, 30 - 31 March.

Chan, T.F., and J. Shen. 2005. *Image Processing and Analysis - Variational, PDE, Wavelet, and Stochastic Methods.* Society of Applied Mathematics.

Cheveigné, A., and H. Kawahara. 2002. "Yin, a fundamental frequency estimator for speech and music." *Journal of the Acoustical Society of America (Vol. 111, Issue 4)*, April: 1917 - 1930.

Davis, Gary, and Ralph Jones. 1988. *The Sound Reinforcement Handbook.* Yamaha International Corporation.

Doval, B., and X. Rodet. 1993. "Fundamental frequency estimation and tracking using maximum likelihood harmonic matching and HMMs." *International Conference on Acoustics, Speech and Signal Processing (Vol. 5)*, April 14 - 17: 3657 - 3660.

Driedger, Jonathan , Stefan Balke, Sebastian Ewert, and Meinard Müller. 2016. "Template-Based Vibrato Analysis of Music Signals." *International Conference on Music Information Retrieval.* Ismir.

Gibbs, J.W. 1899. "Fourier's Series." *Nature (606).*

Gibiat, V. 1988. "Phase space representation of acoustical musical signals." *Journal of Sound and Vibration (Vol. 123)*, 537 - 572.

Hamming, Richard W. 1997. *Digital Filters.* New York: Dover.

Kedem, B. 1986. "Spectral analysis and discrimination by zero-crossings." *Proceedings of the IEEE (Vol. 74. Issue 11)*, November: 1477 - 1493.

Marchand, S. 2001. "An efficient Pitch-tracking Algorithm Using a Combination of Fourier Transforms." *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-01)*, December: 170 - 174.

Noll, M.A. 1964. "Short-Time Spectrum and Cepstrum Techniques for Vocal-Pitch Detection." *Journal of the Acoustical Society of America (Vol. 36, No. 2)*, February: 296 - 302.

Panda, Renato, Ricardo Malheiro, and Rui Pedro Paiva. 2018. "Novel Audio Features for Music Emotion Recognition." *IEEE Transactions on Affective Computing.* 614 - 626.

Rabiner, R.L. 1977. "On the use of Autocorrelation Analysis for Pitch Detection." *IEEE Signal Processing (Vol. ASSP-25, Issue 1)*, February: 24 - 33.

Savas, Savas I. 1965. *Byzantine Music in Theory and in Practice.* Boston: Hercules Press.
Smith, Julius O. 2010. *Physical Audio Signal Processing.* Stanford: W3K Publishing.

Smith, Steven W. 1997. *The DSP Guide.* Poway: California Technical Publishing.

Tan, L., and M. Karnjanadecha. 2003. "Autocorrelation method and AMDF." *Proceedings of the 3rd International Symposium on Communications and Information Technology*, September: 551 - 556.

Wang, Changhong , Vincent Lostanlen, and Emmanouil Benet. 2019. *Playing Technique Recognition by Joint Time–Frequency Scattering.*

Yang, Luwei. 2016. *Computational Modelling and Analysis of Vibrato and Portamento in Expressive Music Performance.*