AN APPLICATION FOR VOCAL TRAINING AND EVALUATION USING REAL-TIME MONOPHONIC PITCH TRACKING

Kumar Ashis Pati

Georgia Tech Center for Music Technology
840, McMillan Street,
Atlanta, Georgia,
ashis.pati@gatech.edu

ABSTRACT

This paper presents a cross-platform application for vocal training and performance evaluation using monophonic pitch tracking. The system is designed to take real-time voice input using any standard microphone, extract the pitch of the signal using an auto-correlation based pitch tracker and evaluate the vocal performance by measuring the accuracy in pitch against a reference song or lesson. The system also provides the user real-time visual feedback with regards to the pitch being sung and the pitch supposed to be sung.

Index Terms— Vocal Training, Pitch Tracking, Performance Evaluation

1. INTRODUCTION

The singing voice is a musical instrument which allows precise control of pitch and expression. Human ears are extremely sensitive to pitch and can detect minor changes in pitch with reasonable accuracy. It takes years of intense practice to develop good singing skills. Most of the vocal performances rely heavily on pitch accuracy and hence, pitch is one of the most important features which is stressed upon in almost all forms of vocal training.

Traditionally, vocal training involves a lot of listening and uses auditory perception as the primary source of feedback. For beginners in particular, who are still training their ears, making an objective evaluation of their performance is a challenging task. Thus, the presence of a teacher who can listen to the performance , evaluate it and guide the student towards improvement is of paramount importance. Without undermining the need of a teacher or that of improving the auditory pitch perception through rigorous ear training, an automatic system which would assist the beginners to practice basic lessons and evaluate the performances would indeed be of great help to many. The use of real-time visual feedback to show the students how accurately the song is being performed might also augment the learning process.

The fundamental research question here is can a computer or machine be able to listen to a vocal performance, analyze and evaluate it. A related question which follows is to what extent will such a system be able to help a student in his endeavors to improve his vocal capabilities. Although both questions are equally important, this paper looks to touch on the first and leaves the second for future work. It presents an architecture for a basic system which can evaluate a vocal performance against a reference lesson based on pitch and timing information. The intent is to design it in such a way that it can be scaled later so as to include more complex evaluation measures such as amplitude modulation and note transitions.

The paper is structured in the following manner. I shall briefly discuss about the related work in this area in the Section 2 before describing the system architecture and the algorithm in detail in Section 3. Subsequently Section 4 will discuss the evaluation of the system. Section 5 will conclude the paper and discuss scope for future work.

2. RELATED WORK

Pitch detection is a fundamental and highly researched topic in the area of audio or music signal analysis. There have been several approaches to solve the problem of fundamental pitch detection and the approaches differ based on whether the signal is monophonic or polyphonic in nature. For the purposes of this paper, only monophonic signals are considered. An introduction to several approaches for monophonic pitch tracking can be found at [1]. A more detailed comparative analysis of various pitch detection algorithms is given in [2]. Some of the popular methods for pitch detection employ variants of Auto-Correlation Function (ACF), the Average Magnitude Difference Function (AMDF) [3], the Harmonic Product Spectrum (HPS) [4] etc.

Although, pitch tracking and related tasks such as transcription of singing voice [5] and score alignment [6] have been given attention earlier, off late a lot of research has also gone into the automatic evaluation of singing voice. Nakano et. al. [7] present a system for automatic binary classification of a vocal performance as “good” or “bad” based on vibrato analysis and pitch steadiness features. Pitch steadiness

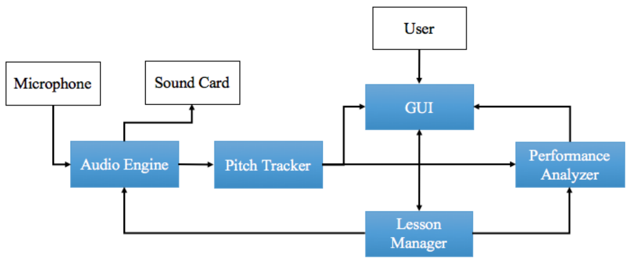


Fig. 1. System Overview

is measured by measuring deviation from an equi-tempered scale. The authors consider usage of vibrato as an "important" singing technique and define a vibrato likeliness measure which is used as a feature for classification. Mayor et. al. [8, 9] present systems for evaluating and scoring vocal performances of reference songs. They use both note and intra-note level analysis to rate the performance based on pitch accuracy, volume accuracy, timing accuracy and expression categorization in terms of note attacks, sustains and vibratos.

It is also worth noting that off-late a lot of products have been made available commercially in the form of mobile apps [10] and games [11, 12] which perform a similar task but use them for entertainment purposes.

3. SYSTEM OVERVIEW

The system architecture is shown in Figure 1. There are 5 primary modules of the system namely:

1. **Audio Engine**: This module carries out all the audio and buffer handling for the system. It reads the input audio buffers returned by the device microphone and creates overlapping windows from them based on a specified block size and hop size. It also writes data to the output buffers which are sent to the sound card for playback. The audio engine also implemented a basic metronome which the user can toggle to get a better sense of timing.

2. **Pitch Tracker**: The pitch tracker module receives the input buffers from the audio engine and calculates the fundamental pitch as a floating point MIDI note number. This detected pitch is used by the subsequent modules. The pitch tracking algorithm is described in detail in Section 3.1.

3. **Lesson Manager**: This module reads vocal training lessons which are in the form of single track MIDI files and converts them into reference pitch values which are used in the evaluation module. The user can select from a list of available lessons for practise. The lessons currently are pre-loaded into the system and mostly comprise of fundamental vocal exercises. The lessons are in the form of monophonic single-track General MIDI standard files containing a single melody. The system is capable of reading any MIDI file which satisfies this criteria.

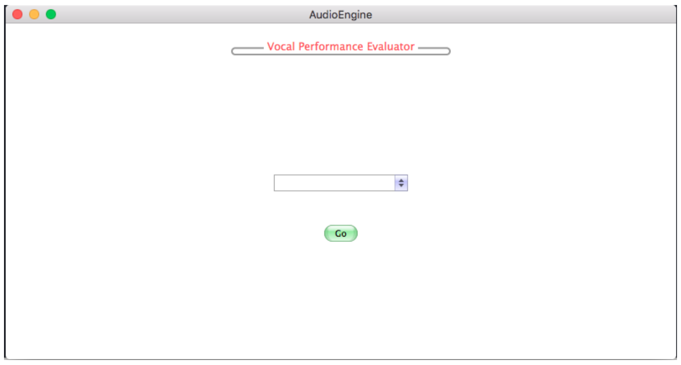


Fig. 2. GUI Screenshot for Lesson Selection

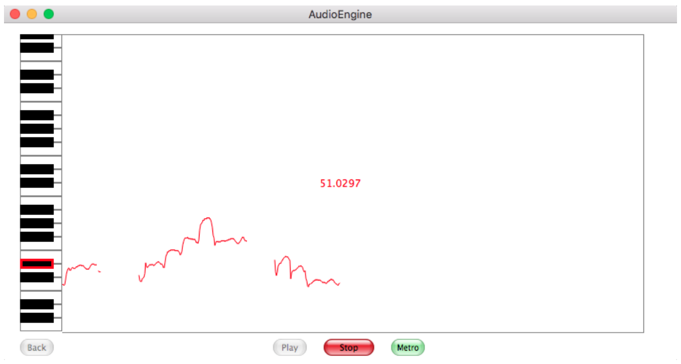


Fig. 3. Visual Feedback by displaying Pitch Contour

4. **Performance Analyzer**: This module takes the detected pitch from the pitch tracker and the reference pitch and timing information from the Lesson Manager and performs the objective evaluation of the performance. The algorithm is described in Section 3.2.

5. **GUI**: The GUI (graphical user interface) interfaces between the different modules and allows the user to select lessons for practice, playback the selected lesson and view the real-time singing performance in the form of a pitch contour and on a graphical piano keyboard (see Figure 3). A few more snapshots of the GUI are shown in Figure 2 and 3 .

3.1. Pitch Tracking Algorithm

The pitch tracking algorithm uses an ACF based method to estimate the fundamental frequency. The different steps of the algorithm are enumerated below:

1. First, a simple energy based threshold is used to implement voicing detection. Only blocks which have an RMS energy above a certain threshold E are considered for pitch extraction. The detected pitch for such blocks is set to zero.

2. The ACF of the input audio signal x_t is calculated for each block n of size K as per the following equation:

$$r_t(\eta, n) = \sum_{i=i_s(n)}^{i_e(n)-\eta} x(i).x(i+\eta) \tag{1}$$

   where $i_s(n)$ and $i_e(n)$ are the indices corresponding to the start and end samples of the block and η is the lag ranging from 0 to $K-1$

3. The ACF for each block is then normalized by dividing the ACF at zero lag i.e. $r_t(0, n)$.

4. The normalized ACF function is then searched for local maxima within a fixed search range determined by the maximum and minimum expected frequency of the input audio signal. This is done to avoid octave errors and reduce the computational complexity.

5. Another additional criteria is imposed to improve the accuracy which uses the previous detected pitch to further narrow down the search region by looking at an interval within ± 1 octave of the previous detected pitch. If the detected pitch for the previous block is zero, then this condition is not imposed.

3.2. Performance Evaluation

The performance evaluation section uses a rudimentary pitch comparison algorithm to classify each sung note as correct or incorrect based on whether the note is within a certain threshold T (0.50 cents) when compared to the reference pitch at that particular time instant. The evaluation currently only happens for regions where the reference pitch has a non-zero value. Further a small window is ignored at either ends of reference pitch note to ensure that deviations in pitch during note transitions are not penalized.

4. EVALUATION

4.1. Pitch Tracker

A total of 3 pitch trackers were initially considered to be used. These included the ACF based algorithm described in the previous section, the YIN algorithm [3] (a MATLAB implementation made available online by the author at [13] was used) and a wavelet based algorithm [14]. For ease of implementation and testing, all the 3 algorithms were implemented in MATLAB and were tested using the TONAS dataset ([15],

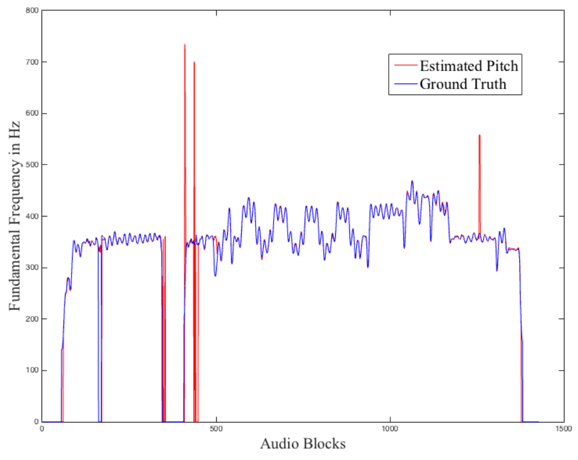
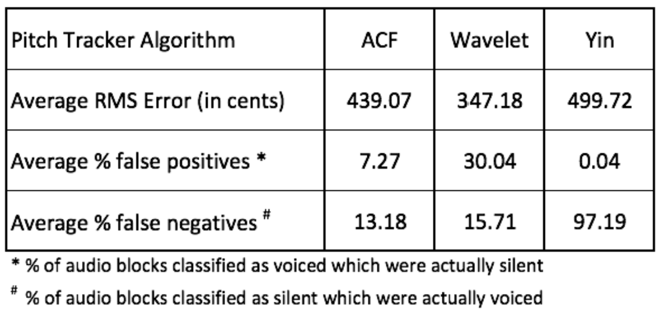


Fig. 4. Estimated Pitch v/s Ground Truth for a sample audio in the Tonas Dataset using ACF Algorithm



Pitch Tracker Algorithm	ACF	Wavelet	Yin
Average RMS Error (in cents)	439.07	347.18	499.72
Average % false positives *	7.27	30.04	0.04
Average % false negatives #	13.18	15.71	97.19

* % of audio blocks classified as voiced which were actually silent

# % of audio blocks classified as silent which were actually voiced

Fig. 5. Performance of different Pitch Tracking Algorithms on the TONAS Dataset

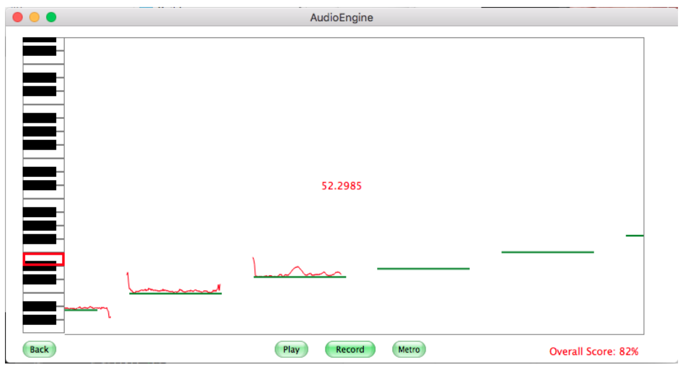


Fig. 6. Sung Pitch v/s Reference Pitch (Sung Pitch is in red color, Reference Pitch is in green color)

[16]). The dataset contains 72 excerpts of monophonic traditional flamenco singing within annotations covering the fundamental pitch for each time block. The results of the evaluation are tabulated in Figure 5. The following parameters were used for evaluating the pitch trackers:

1. Average RMS Error in Cents: This metric was computed for audio blocks which are both classified as voiced (or pitched) in the estimate as well as the ground truth.

2. Average % of false positives: The average % of blocks which were classified by the pitch tracker as pitched but were unvoiced in the annotations.

3. Average % of false negatives: The average % of the blocks which were classified as silent but where actually pitched in the annotations.

A sample plot of estimated pitch v/s ground truth is shown in Figure 4. Although the average error was the least for the wavelet based algorithm, it had higher % of false positives and false negatives. The yin algorithm on the other hand had the lowest % of false positives but had a very high % of false negatives. The ACF algorithm had acceptable performance on all the 3 metrics.

4.2. Overall System

The real-time system was developed using the JUCE framework [17] which is C++ based cross platform framework for developing audio, interactive, embedded and graphical applications. The current version of the system has been tested on a MacBook Pro (running a 2.7Gz processor with 8GB RAM). The block size was chosen as 1024 and the hop size was chosen as 512. The sampling rate of the device was 44100Hz. Different modules of the program were tested individually to verify their performance.

1. The audio engine was tested to ensure that it is able to handle all possible block size and hop size combinations. This was done by passing a ramp signal through the system and checking the blocks returned by the audio engine.

2. The pitch tracker was tested using sine waves for known frequency as input. As as additional check to compare the implementation, the pitch contour returned by the system for a test signal was compared with the MATLAB implementation to check if the C++ implementation was accurate.

A sample plot of the sung pitch v/s the reference pitch is shown in Figure 6. The system is able to accurately determine the sung pitch and is able to evaluate the performance against the reference pitch as per the evaluation algorithm. The extent of the accuracy of the evaluation is not verified currently. This would require the system to be used by several individuals in a controlled environment and the evaluation results would have to be compared with those of human judges who rate the performances. A strong correlation between the two would confirm the effectiveness of the system.

5. CONCLUSION & FUTURE WORK

This paper presents a system which can be used for automatic real-time evaluation of vocal performance. Considering the simplicity of the performance evaluation algorithm, the current evaluation is an extremely basic estimation of the performance of the user and scores only on whether the right pitch was sung at the right time. It doesn't take into account the modulations or note transitions as in some of the earlier works. However, the system can be scaled to include more complex performance descriptors for getting more insights into the performance and that is the major area over which the future work would be directed. As mentioned earlier, the aim would be to objectively evaluate the performance based on note accuracy, timing accuracy, modulations and transitions. Attempts to include articulations and expression can also be made.

A few other areas for improvement would be the following:

1. ACF is a time-based pitch tracker. A frequency based pitch-tracker such as HPS could be used in conjunction with ACF to further increase the accuracuy of pitch detection. The ACF can also be computed using the FFT (Fast-Fourier Transform) to reduce computational complexity.

2. The current GUI design ensures functionality but is pretty basic. More work is needed on the GUI to improve the overall user experience which can later be developed into a full-fledged application.

Further work would include packaging the system in the form of easy-to-use tool which can be used by students and teachers. The structure of the lessons should allow the users as well as the teachers to author their own lessons. These lessons can be shared in a "cloud based" community learning environment where all users should be able to access them and use them to improve their vocal skills.

6. REFERENCES

[1] Alexander Lerch, *An introduction to audio content analysis: Applications in signal processing and music informatics*, John Wiley & Sons, 2012.

[2] Onur Babacan, Thomas Drugman, Nicolas d'Alessandro, Nathalie Henrich, and Thierry Dutoit, "A comparative study of pitch extraction algorithms on a large variety of singing sounds," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7815–7819.

[3] Alain De Cheveigné and Hideki Kawahara, "Yin, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[4] Manfred R Schroeder, "Period histogram and product spectrum: New methods for fundamental-frequency measurement," *The Journal of the Acoustical Society of America*, vol. 43, no. 4, pp. 829–834, 1968.

[5] Timo Viitaniemi, Anssi Klapuri, and Antti Eronen, "A probabilistic model for the transcription of single-voice melodies," in *Proceedings of the 2003 Finnish Signal Processing Symposium, FINSIG03*, 2003, pp. 59–63.

[6] Pedro Cano, Alex Loscos, and Jordi Bonada, "Score-performance matching using hmms," in *Proceedings of the international computer music conference, Beijing, China*, 1999.

[7] Tomoyasu Nakano, Masataka Goto, and Yuzuru Hiraga, "An automatic singing skill evaluation method for unknown melodies using pitch interval accuracy and vibrato features," *Rn*, vol. 12, pp. 1, 2006.

[8] Oscar Mayor, Jordi Bonada, and Alex Loscos, "Performance analysis and scoring of the singing voice," in *Proc. 35th AES Intl. Conf., London, UK*, 2009, pp. 1–7.

[9] Oscar Mayor, Jordi Bonada, and Alex Loscos, "The singing tutor: Expression categorization and segmentation of the singing voice," in *Proceedings of the AES 121st Convention*, 2006.

[10] "Smule karaoke: An mobile application available on android and ios which provides users with karaoke tracks of popular songs which the user can sing. the app rates the performance of the user and also ranks them against peers.," `http://www.smule.com/apps`, Accessed on November 29, 2015.

[11] "Rock band 4: a series of music video games developed by harmonix music systems and mtv games that allows for up to four players to virtually perform rock music songs on lead guitar, bass guitar, drums, and vocals.," `http://www.rockband4.com`, Accessed on November 29, 2015.

[12] "Lips: karaoke game for the xbox 360 developed by inis and published by microsoft game studios. the game will feature the use of a motion sensitive microphone, and supports the use of songs already owned by the user," .

[13] "Matlab implementation of the yin pitch estimator," `http://audition.ens.fr/adc/yin.zip`, Accessed on November 29, 2015.

[14] Eric Larson and Ross Maddox, "Real-time time-domain pitch tracking using wavelets," *Proceedings of the University of Illinois at Urbana Champaign Research Experience for Undergraduates Program*, 2005.

[15] Emilia Gómez and Jordi Bonada, "Towards computer-assisted flamenco transcription: An experimental comparison of automatic transcription algorithms as applied to a cappella singing," *Computer Music Journal*, vol. 37, no. 2, pp. 73–90, 2013.

[16] J Mora, F Gómez, E Gómez, FJ Escobar-Borrego, and JM Diaz-Bañez, "Melodic characterization and similarity in a cappella flamenco cantes," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2010, pp. 351–356.

[17] "Juce: a c++ based cross platform framework for developing audio, interactive, embedded and graphical applications.," `http://www.juce.com`, Accessed on November 29, 2015.