Solo acapella

A tool to quickly test your music idea

Zheng Wenyi
e0509848@u.nus.edu
Contribution:50%

Shen Zongyi
e0392432@u.nus.edu
Contribution:50%

ABSTRACT

To quickly compose new music, generation of notes and instant prototype will be helpful. Human voice can be viewed as the most creative and convenient instrument. The combination of human voice and composing has unlimited potential. This project implements an application that allows musicians having the opportunity to experiment with various musical instrument sounds without actually having to learn how to play them. Users can hum or sing whatever they want to test and a sound wave will be recorded. The sound wave is received and processed mainly based on YIN algorithm with outputting digital sound frequency to a frequency to MIDI (Musical Instrumental Digital Interface) converter. The MIDI note is then used to produce different instrumental sound.

1 Problem Definition

The task of this application is to generate sound waves of different instrumental music based on human voice. There are many assumptions used here. The main task will be divided into these parts:

Track the pitches. Given a sound wave of human voice, we need to detect its pitch for further generation of notes. As notes are determined by frequency, in this step we will detect the fundamental frequency at each time point. Here, we assume: 1. The sound wave of human voice only has one dominant fundamental frequency, we will call this frequency f_0. 2. The volume will be considered equal throughout. The result should be accurate and this process should not take long time.

Generate notes. Given the frequency of the sound wave, we need to generate notes accordingly. Because our application is meant to help musicians compose, the music we generated should fit in music theory, otherwise it is meaningless. The notes should be as close as possible to the pitch we tracked in the previous step. And they should obey the standards of music theory. Also, this step should not be time-consuming.

Output instrument music. Given the notes we extracted from the sound wave of human voice, we have to convert them into a sound wave of the same notes played by different instruments. In this step, we should define the interface between notes and sound wave.

2 Review of Literature

There are a lot of work on music and human voice. We will first introduce them in three parts described in Problem Definition. After that we will cover some existing systems or applications.

Track the pitches. The subjective pitch of a sound usually depends on its fundamental frequency, but there are exceptions. Some methods for pitch detection use LPC (linear predictive coding) model by detecting peaks from LPC residual signal through adaptive filtering[2][2], or by calculating energy[3]. A more common solution is YIN[4], which uses correlation in time zone. It simply assumes the sound wave will have highest correlation with itself after shifting multiple periods, and this step is called autocorrelation. And this algorithm uses more steps to reduce the error rate including Difference function, Cumulative mean normalized difference function, Absolute threshold Parabolic interpolation and Best local estimate. There are many algorithms based on YIN, and a well-known one is pYIN[5], which can get the probability distribution of fundamental frequency.

Generate notes. Transcribing the singing voice into music notes is challenging due to pitch fluctuations such as portamenti and vibratos. Most time we can just try to model the probabilistic distribution of each note. Multi-layer HMM (Hidden Markov Model) can be used to model the sequence of the notes and thus re-estimate each note[6]. Due to the fact that we do not know the labels(notes) of the sound wave, an EM (Expectation Maximization) algorithm[7] can be applied. The common methods are heuristic and based on simple threshold.

Output instrument music. MIDI (Musical Instrument Digital Interface) was put forward in the early 1980s to solve the communication problem between electroacoustic instruments. Now it has been the most widely used standard musical format in the field of music editing. It records music using a digital control signal of notes. A complete MIDI music is only a few kilobytes large, but can contain dozens of music tracks. There are 3 reasons why we use MIDI. 1.MIDI is a technology that enables music to be recorded on a computer in the form of audible notes, not only on the piano, but also on instruments such as drums, bass and synthesizers. That means we can use MIDI compose whatever we

want. 2.MIDI is a technology that allows multiple instruments to play simultaneously. 3. save time and space. This is the same argument as the original purpose of the project, because the composer does not need to communicate with the musicians on every instrument, saving some time for rehearsals.

There has been some research on the human voice and midi already. MIDI is a popular format to store music, while human voice can be viewed as the most convenient instrument. Their combination will have potential application scenarios. It can be used to score a singing[8]. A method to align singing voice to its corresponding musical score is proposed. MIDI score is first converted to sound wave and then more process will be done to evaluate the similarity between them. And instant feedback will help users learn music undoubtedly, like in [9], real-time feedback is used to help users learn music therapy.

3 Approach

Our application is designed for quick music prototype of human voice. It converts sound wave of human voice to the sound wave of the same notes played by different instruments.

We will still follow the three main parts defined in the section Problem Definition. After the main functions, we will introduce our UI design.

3.1 Track the pitches

In this step we use the YIN algorithm. We will introduce how we implemented it. The basic assumption here is that: if the signal is periodic, and we shift the signal in different lags, the signal will be most similar to the signal shifted with multiple periods. This can be easily illustrated using an autocorrelation function, which is defined as: $r_t(\tau) = \sum_{j=t+1}^{t+W} x_j\, x_{j+\tau}$. This is intuitive and we can just pick the non-zero lag that maximizes the autocorrelation function.

But autocorrelation function has a relatively large error rate, so YIN algorithm uses several steps to optimize it.

Difference function. This function is defined as $d_t(\tau) = \sum_{j=1}^{W}\left(x_j - x_{j+\tau}\right)^2$. It evaluates the similarity of the signals at 0 and τ lag(samples). If the difference function is small, it means the more likely τ will be a period of this signal. In this application we set the windows size and windows-step to be 1024 and 512 by default. The min and max of τ will also depend on predefined parameters. In this application we allow users to set the min and max of fundamental frequency so that the lag will be decided accordingly. But we can see that this function will always be very small if τ is close to 0. So the next step is to avoid the algorithm to choose 0 as the period.

Cumulative mean normalized difference function. This function is defined as $d'_t(\tau) = d_t(\tau) / \left[\frac{1}{\tau}\sum_{j=1}^{\tau} d_t(j)\right]$. This function equals 1 if $\tau = 0$. This function is obtained by dividing each value of the old by its average over shorter-lag values. It differs from d(t) in that it starts at 1 rather than 0, tends to remain large at low lags, and drops below 1 only where d(t) falls below average. But this may lead the algorithm to prefer higher-order dips. The next step will help avoid this.

Absolute threshold. Here we set an absolute threshold and choose the smallest value of τ that gives a minimum of d' deeper than that threshold. If none is found, the global minimum is chosen instead. According to the original paper, it will reduce "too low" errors accompanied by a very slight increase of ''too high'' errors. In this application we set this threshold to be 0.1.

Parabolic interpolation. All the previous steps assume the period is a multiple of the sampling period. And the actual error by up to half the sampling period. This step will interpolate between all local minimum of difference functions and its immediate neighbors. And then the interpolated minimum is used in the dip-selection process. The interpolation of difference function is computationally cheaper than upsampling the signal. Actually in this application we did not implement this step, so here we skip it.

Best local estimate. The last step is called best local estimate. This step iteratively optimizes the estimate. Actually, this step is not implemented in our application, so here we skip it.

3.2 Generate Notes

From the previous results, we get the estimation of fundamental frequency for each frame. With all the frequency data, we will generate a sequence of notes accordingly.

In this part, we have the assumption that if there is a note event, it will remain the same fundamental frequency and last longer than a pre-defined period. In this application, the time threshold of a shortest note is 0.1 by default but it can be changed by users. And here the case is often that the fundamental frequency of a note will fluctuate slightly, so we will decide a note if in this period the fundamental frequency is in a range with a less-than-5Hz range. We will detect in all the possible length exhaustively, and output all the candidates of note events.

Then, we will map all the fundamental frequency to music notes. Here we just use one scale—C major scale. It is a common scale and we will round each candidate to a note in C major scale. The volume of each note will be set as the same. Here we do not consider the volume, we only consider the frequency. The relationship between notes and their frequency is at https://pages.mtu.edu/~suits/notefreqs.html.

3.3 Output instrument music.

Now we've got all the notes that we've been humming before. All we need to do is write each note to the MIDI file.

In this part, we use third-party libraries called Mido (for generating midi file) and PyGame.midi (for playing midi file). These two libraries Provides readable commands to replace the hexadecimal representation of MIDI event instructions.

We defined two interfaces. The first is to generate a MIDI file according to the notes and its duration. The second is to adjust the instrument before playing it.

3.4 UI Design.

To provide more pleasant user experience, we design a user-friendly interface for users. The interface provides all the needed functions including recording you voice, importing existing wav file, see the analysis of the sound wave and hear the generated music.
Besides the main functions, it also allows users to modify the parameters of the algorithm. But we did not do a very detailed and strict input check, so there may happen that sometimes the program will abort if your input parameters are invalid. For example, if we set the min of fundamental frequency as 1 while windows size as 1024, it will abort. This is because frequency with 1 Hz means we need to do a lag as 44100 samples (Assume the sampling rate is 44100Hz). This is larger than the size of a window, so the program will be aborted.

4 Results and Analysis

Due to limit of time and capacity, especially under current situation of COVID-19, we did not do a user test. We will briefly describe our final application and use several input and their output to discuss.
Our user interface is like this:

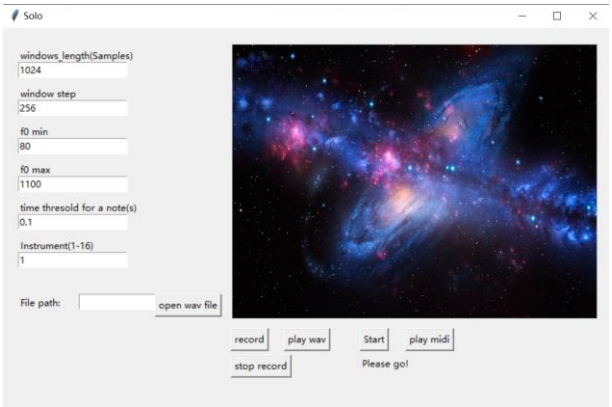


Figure 1. Main menu

In the left panel, there are text entries that you can modify the parameters. In the right, there is an area for a picture which will show some visualization of analysis of input. Below that, there are buttons for record, play or generate midi functions. More details will be provided in the readme file.
Here, we will analyze results of two types of input, standard piano scale sound and human voice sound.

Standard piano scale sound. This input is a standard C major scale sound. We apply the algorithm using the default parameters.

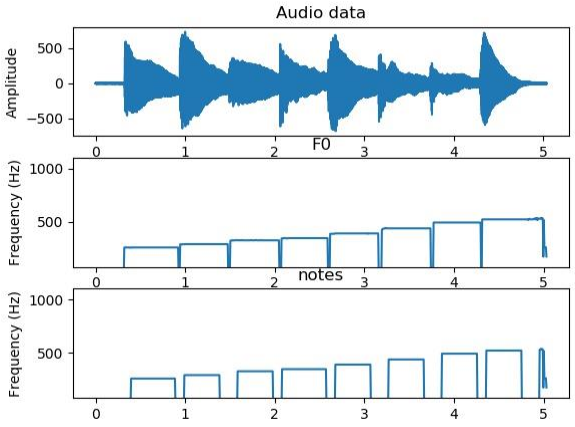
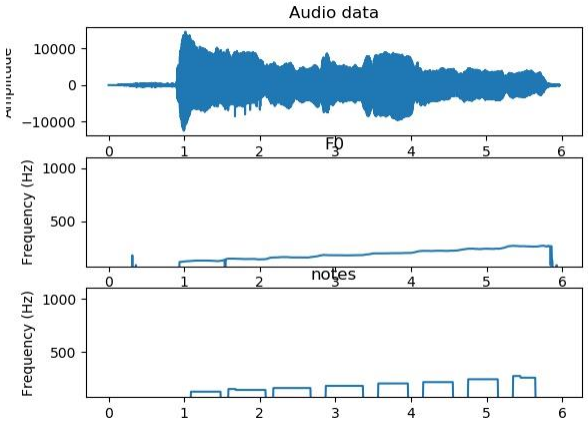


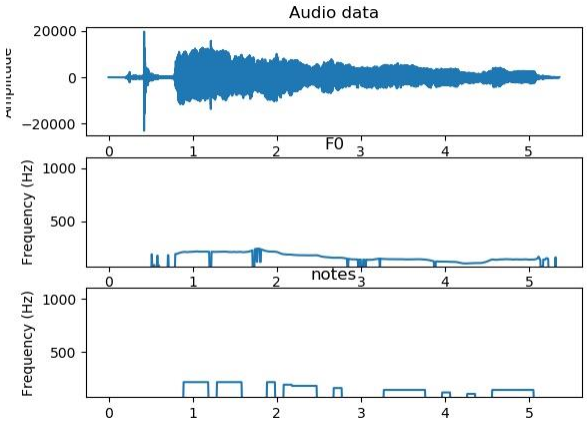
Figure 2 analysis of C major scale

Figure 2 shows our output of the C major scale. We can see that the pitch track is clear and accurate. When converted to notes, the music retains most characteristics and the final result is meaningful. This shows that the application can do very well if the input is standard.
Next, we want to test the result of other instruments and human voice, which will not be so standard and will include pitch fluctuations such as portamenti and vibratos. All the input files can be found in the attachment.
Human voice to mimic a Scale. To compare with the standa rd piano C major scale, the sound of voice tends to be smoot her. It will not encounter a sudden change in frequency. But still, the extracted pitch can show a simple track of scale. The curve shows the basic shape of a continuously increasing sca le.



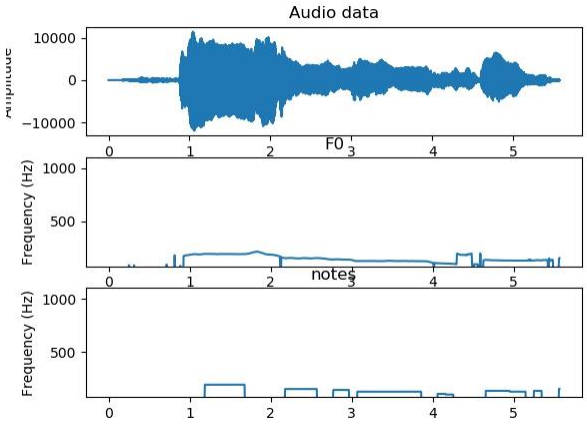
Meanwhile, note that actually in the last part of the extracted note, there are two notes. But the input does not intentionally produce two notes. This is because human voice tends to flu ctuate. As a result, this process will not be as stable as sound from instrument.

Human voice without lyrics. To avoid the effect of lyrics, th is input is the only word “A” continuing. It is a start of the so ng “five rings”.



We can see that the pitch track is very smooth. But when we hear the wav sound, we can distinguish it is the song. The alg orithm will work accordingly but the output may not be so di stinguishable.

Human voice with lyrics. To compare with voice without ly rics, we add this input. It is also the start of the song “five rin gs”. We can see that although the sound is generated by the s ame person, it will have different notes. And this is also affec ted by the property of fluctuation of human voice. Even the s ame notes pronounced by the same person will not always ke ep the same.



To conclude, currently the application works well if the user input has strong patterns like a scale. If the input is regular, t he algorithm here will generate meaningful output and it is di stinguishable for human ears. But if the input is arbitrary, the n the algorithm will possibly fail to generate good notes and t hus making the output not so meaningful.

5 Future Work

Our application is currently a toy for composers. But it will h euristic to developers to implement more ideas. In the future, more characteristics can be added to this application. For exa mple, the feature of volume can be added. Meanwhile, better algorithms to extract notes will be come up with. And in this application we just use a simple C major scale, if possible mo re scales can be considered.

REFERENCES

[1] M. Dong, P. Chan, L. Cen and H. Li, "Aligning singing voice with MIDI melody using synthesized audio signal," 2010 7th International Symposium on Chinese Spoken Language Processing, Tainan, 2010, pp. 95-98.

[2] D.G. Childers, H.T. Hu, "Speech synthesis by glottal excited linear prediction", Journal of the Acoustical Society of America, 1994

[3] P.A. Naylor, A. Kounoudes, J. Gudnason, M. Brookes, "Estimation of glottal closure instants in voiced speech using the DYPSA algorithm", IEEE Transactions on Audio, Speech, and Language Processing, Volume 15, Issue 1, pp.34-43, January 2007

[4] Alain de Cheveigné and Hideki Kawahara. YIN, a fun-damental frequency estimator for speech and music.Journal of the Acoustical Society of America (JASA),111(4):1917–1930, 2002.

[5] M. Mauch and S. Dixon, "PYIN: A fundamental frequency estimator using probabilistic threshold distributions," 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, 2014, pp. 659-663.

[6] L. Yang, A. Maezawa, J. B. L. Smith and E. Chew, "Probabilistic transcription of sung melody using a pitch dynamic model," 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, 2017, pp. 301-305.

[7] T. K. Moon, "The expectation-maximization algorithm," in IEEE Signal Processing Magazine, vol. 13, no. 6, pp. 47-60, Nov. 1996.

[8] M. Dong, P. Chan, L. Cen and H. Li, "Aligning singing voice with MIDI melody using synthesized audio signal," 2010 7th International Symposium on Chinese Spoken Language Processing, Tainan, 2010, pp. 95-98.

[9] H. Kawahara, E. Haneishi and K. Hagiwara, "Realtime feedback of singing voice information for assisting students learning music therapy," 2017 International Conference on Orange Technologies (ICOT), Singapore, 2017, pp. 99-102.