PITCH IDENTIFICATION AND MODIFICATION

James McCarthy and Matthew Kyong

Audio and Music Engineering Department, University of Rochester

ABSTRACT

In the field of music, the need to work on tuning and intonation on an instrument is an essential and ever present part for all musicians. To be able to work on this crucial aspect of music making, most instrumentalists and vocalists turn to a tuner for guidance on how sharp or flat they are. The ultimate problem with this method is the lack of aural feedback, which is undoubtedly an extremely useful learning tool for musicians, and the amount of trial and error a musician will have to run through to finally achieve the right pitch. Therefore, a system that both identifies pitch and plays back corrected audio for the musician was implemented.

1. INTRODUCTION

Tuning and intonation are two of the most important facets of music. For many musicians, practicing intonation and perfect pitch allows for improved musicality and performance. Being able to recognize when you are singing/playing out of tune will greatly improve your overall skills as a musician. In order to assist musicians in their intonation training, this pitch identification and modification tool was created.

The project was designed with two main goals in mind; identify the pitch that a user plays and play back a pitch corrected audio file so the user can hear themselves correctly intonated. With these two goals in mind, the project utilizes two different algorithms to achieve both pitch identification and modification. The pitch identification part of the project relies on the equal temperament tuning system (with A4 = 440 Hz) to identify the correct pitch from a frequency. This tuning system also determines the frequencies that the pitch modification algorithm will tune to.

The program works by taking in a five second recording of the user singing or playing a note. Then, the program will playback what the user has recorded and display the average pitch of the note played and the closest note based on equal temperament tuning. After this, the program will warp the recording, tune the note to the true frequency of the note, and play it back for the user to hear. The idea is for the user to be able to improve their intonation by being able to hear themselves sing perfectly in tune.

2. PITCH IDENTIFICATION

The first part of this project is concerned with readily storing user data and identifying the pitch from this recording. In order to modify the recording to a frequency within equal temperament, it is imperative that the original frequency of the recording is found first. To achieve this, the user is prompted to play or sing a tone within a window of time. Once this window of time is completed, the tool then uses the audiorecorder object from the audio toolbox in MATLAB to create an object of the recording. From this object, the raw audio data can be extracted from the object's built in function getaudiodata.

From here, the idea of background noise and unwanted artifacts comes into question. To try to combat against this noise, a simple limiter can be implemented. This limiter takes in the audio signal and only lets samples with an amplitude over a specified threshold pass through. The only drawback of this method is that most of the noise being taken out is ambient noise. However, by fully getting rid of this ambient noise, the accuracy of the following pitch function vastly improves.

To find the fundamental frequency of the audio recording, a built in function from the audio toolbox was employed. pitch() is a function that, through its Normalized Correlation Function and use of a Pitch Estimation Filter, can estimate the fundamental frequency over time in an audio file. However, it's important to note that this function takes the fundamental frequencies across the entire signal, including any room noise. To correct this, a limiter is employed to smoothen out the response of the pitch function and zero out all non essential fundamental frequencies. This sets clear boundaries for the pitch that needs to be identified. Using these boundaries, each sample is processed and the average frequency within this set is calculated.

Undoubtedly, a frequency table must be created in order to match the average frequency to its closest frequency in equal temperament where A = 440 Hz. So, this identification tool uses an array with the frequencies between C1 and C7. To find the frequency (and therefore the pitch name) that is closest to the average pitch of the signal, a process is used that calculates the difference between the average pitch and each frequency in the table. The index that gives the lowest difference is outputted and given to the user as a note name.

3. PITCH MODIFICATION

The second part of this project deals with pitch modification. A major goal of the project is for the user to be able to hear themself singing/playing a note perfectly in tune. In order for the user to benefit from the pitch corrected recording of themself, the modified recording must be correctly tuned while maintaining enough sonic characteristics of the original input recording. To achieve this, a phase vocoder algorithm was used to ensure that the signal could be pitch shifted and time warped independently.

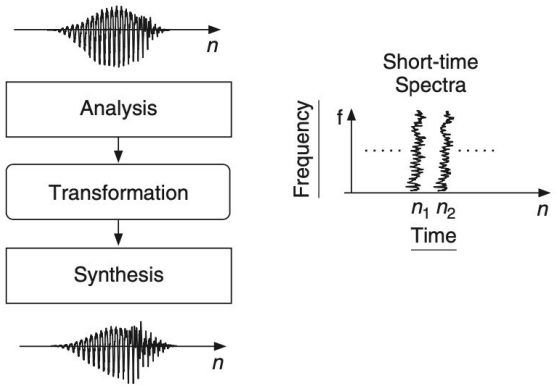

Figure 3.1 Basic time-frequency processing overview

3.1 Basic Phase Vocoder Algorithm

The phase vocoder algorithm relies on a short-time Fourier analysis/synthesis structure. A signal gets analyzed using short-time Fourier analysis. With both the time domain and frequency domain analyses of the signal, the two-dimensional representation can be modified in multiple ways. For the purposes of this project, the two ways the signal will be modified are pitch shifting and time warping. Then, a new signal is synthesized using the modified representation. The overview for this time-frequency processing algorithm is shown in the figure below:

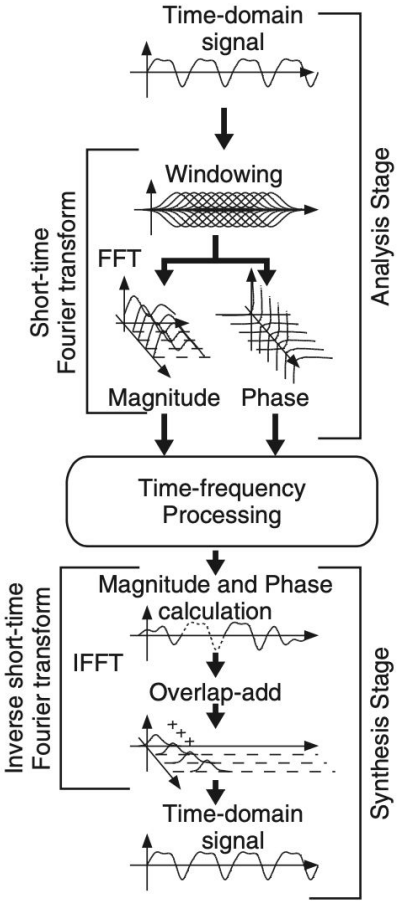

Figure 3.2 Phase vocoder algorithm overview

For the original phase vocoder implementation for this project, there were three main steps to achieve the desired time-frequency processing.

First, the original signal was resampled to achieve the desired pitch change. This step was simplified due to the fact that the average pitch of the signal has already been calculated in the pitch identification part of the project. Given the calculated average pitch and the frequency value for the closest pitch, a pitch-change factor β can be calculated (calculated average frequency / desired pitch frequency). The original signal can then be resampled at 1/β times the original sampling rate to achieve the desired pitch change. However, this resampling will also affect the timing of the signal. In order to maintain the timing of the original signal, frequency domain processing must be done.

The second step in this phase vocoder algorithm deals with analyzing the signal to be processed in the frequency domain. Using a hamming window with 2048 sample length, 50% overlap, and hop size of WindowLen/4, the

original signal is windowed and the FFT is taken frame by frame.

The final step deals with interpolating the resampled spectrogram to achieve the desired time change. The idea here is that pitch warped frequency content can be linearly interpolated over a different time vector in order to maintain the pitch changes while changing the timing. Just like the pitch change factor β, the speed change factor α is already known. The final pitch modified signal should maintain the timing of the original signal, so α should be equal to 1/β in order to get back to the original spectrogram length. The frames of the resampled spectrogram are linearly interpolated to fit the length of the original signal's spectrogram. The phase advance is also calculated to ensure the frequency content gets time warped correctly. The IFFT is taken for this interpolated spectrogram to synthesize the new time domain signal for the pitch modified recording. Based on the pitch and speed change factors β and α, the modified pitch recording is tuned to the correct pitch and has the same length in samples as the original signal.

There were some major limitations associated with this basic phase vocoder algorithm. Mainly, the presence of artifacts and distortions in the synthesized signal. Due to the fact that the frequency content of a single frame is simply linearly interpolated into a new frame means that there may be discontinuities created between frames. This issue manifests itself into some very audible artifacts that distort the signal. When testing this algorithm, the distortions were so present that the pitch modified output could not be considered a good way to train intonation, as the artifacts made the pitch modified signal too different from the original recorded input. Furthermore, the artifacts weakened the precision of our pitch identification algorithm. The artifacts created artificial harmonics in the signal that would skew the identified pitch. For example, when warping an A3 note from 217.38 Hz to 220 Hz, the program identified the pitch modified output to have an average pitch of 232.67 Hz.

3.2 Improved phase vocoder algorithm

To improve upon the basic phase vocoder algorithm, a few different methods were tried. Initially, high-pass and denoise filters were applied after the synthesis stage to try to remove the noise/artifacts created during the previous stage. However, this method proved ineffective as not all of the noise was able to be removed. So in order to remove the artifacts and distortions, the algorithm was revisited altogether.

Rather than linearly interpolate each frame the resampled FFT, the analysis and synthesis hop size lengths were scaled. To time stretch the signal, the improved algorithm uses a different hop size length in the analysis section than the synthesis section. The ratio between the hop sizes is based on the pitch change factor β (β = SynthesisLen / AnalysisLen). Rather than do linear interpolation on each frame of the FFT, the number of frames are determined by the hop sizes when taking the STFT and ISTFT. The phase increments are also scaled by the ratio between the hop sizes (phaseAdvance = (phaseAdvance+phaseData) * Hopratio). This results in a signal with virtually zero audible artifacts or distortions; giving a near perfect pitch corrected signal.

There are drawbacks to this improved phase vocoder method. The major limitation is that it is impossible to perfectly time stretch the resampled signal to the original length. This is due to the fact that the time warping is done with hop size scaling. When using the linear interpolation method, each frame is interpolated to achieve the exact time stretch to fit the desired number of frames. However with the hop size scaling, there is no exact frame number calculation.

The ratio of analysis and synthesis hop size lengths are determined by the pitch change factor β. There are other constraints for these hop size lengths. First, they must be integer values in order to be used in the STFT calculations. They also must be a fraction of the value of the window length to ensure an accurate synthesis of the signal. These constraints are especially important when β is close to 1. This is often the case when tuning a note only a few hertz sharp or flat. When the hop size lengths are rounded off and scaled down (to fit the constraints) some of the time stretching precision is lost and the exact number of samples can not be resynthesized. For example with β = 1.0099 the rational integer values for the hop size lengths should be 1637 and 1621. To accomodate for the window length of 1024, these values must be scaled down and rounded off. The scaled values become 128 and 127. Clearly, a large amount of precision is lost, and the synthesized signal has length 219961 samples while the original signal has length 220500.

Even with the inability to time stretch exactly to the original signal, this new algorithm works extremely well for the purposes of this project. The amount of precision that is lost can be considered negligible due to the length of the overall signal. There is only a difference of a few hundred samples between the 5 second long signals, and the pitch modified signal has no audible artifacts.

4. CONCLUSION

Utilizing the pitch identification tools and improved phase vocoder algorithm yields a high precision intonation tool. The accuracy of the tool provides a lot of benefit to musicians, and the ability to hear themself perfectly tuned will hopefully provide a unique way to practice their intonation.

There are still numerous ways to improve upon the current tool. Currently, the limiter used in the pitch identification algorithm is rather basic and does not remove noise in any spectral way. This allows for background noise above the limiter threshold to incorrectly skew the pitch identification tool and in some cases not return a note at all. A potential solution to this noise issue involves improving the original pitch detection algorithm. Due to the fact that most background noise does not have a consistent pitch center, the algorithm could only look for sections with more sustained pitch centers between frames. This would allow the tool to identify notes in the presence of background noise.

A major next step for the project would involve identification and modification of multiple notes. This would allow the user to sing/play an entire scale or melody into the tool and have each note identified and correctly pitched. For this pitch identification, the program would have to identify when the average pitch deviates from the previously identified pitch for long enough to constitute a new note. Once each note within the signal is marked, the time vector from the pitch data can be linearly interpolated to fit the original time domain signal. This will allow the tool to know which sections of samples are identified as different notes, and subsequently pitch correct each of these sections. This project has a strong, working foundation and hopefully has the potential to benefit a variety of musicians.

5. REFERENCES

[1] Mathworks.com. 2020. *Object For Recording Audio - MATLAB*.[online]Availableat:<https://www.mathworks.com/help/matlab/ref/audiorecorder.html> [Accessed 7 May 2020].

[2] Mathworks.com. 2020. *Estimate the fundamental frequency of audio signal*. <https://www.mathworks.com/help/audio/ref/pitch.html> [Accessed May 7 2020].

[2] Udo Zolzer, *DAFX: Digital Audio Effects Second Edition*, John Wiley & Sons, 2011

[3] Mathworks.com. 2020. *Pitch Shifting and Time Dilation Using a Phase Vocoder in MATLAB*. <https://www.mathworks.com/help/audio/examples/pitch-shifting-and-time-dilation-using-a-phase-vocoder-in-matlab.html> [Accessed May 7 2020].