Music Transcription for Polyphonic Melodies

Siobhan Plouffe, Aine Ryhn, Jake Fox, Luke Nash

Department of Electrical and Computer Engineering
University of Rochester

ABSTRACT

This paper describes a method for polyphonic transcription for a two line piece of the same instrument. The frequencies were detected using the constant Q transform and cross correlation methods. The rhythms were determined by the RMS energy of the signal, which showed each note's start time and approximate end time. The note duration was then set to be the difference between those two values. A matrix was then created to be converted to a MIDI file using code from Ken Schutte. Then, this MIDI file was easily imported into external music software to create sheet music.

1. INTRODUCTION

As musicians, it is vital to be able to dictate music being played into writing. Checking dictations solely through one's ear can often be tedious, which is why this MatLab program was created. This program takes a .wav file as an input and exports a MIDI file which can then be converted to sheet music through third party software applications like MuseScore. Our code sets the foundations for full polyphonic midi transcription software, which would have enormous benefits for musicians. Dictations would be much easier to complete, and so much time and effort would be saved if transcriptions were done through a program that could create a piece of sheet music in a few minutes.

While trying to attack this problem, multiple goals were set. These goals were to be able to detect the different frequencies being played in the audio recordings, be able to detect the rhythms of the notes in the audio file, and be able to create and export a MIDI file for a monophonic line. We then wanted to get as close as possible to to achieving the same effect for a polyphonic audio file of two lines of the same instrument.

This paper will mainly detail our methods for finding the notes and rhythms of a melody played on piano. The first part of this paper will detail how the pitches and rhythms were detected while the second part will detail the ways that monophonic lines as well as polyphonic lines were found.

2. PITCH DETECTION

2.1 The CQT

Pitch detection was arguably the most vital part of this project as notes are the most fundamental part of music transcription.

Quantity	CQT	DFT
Center frequencies	exponential in k $f_k = f_0 \left(\sqrt[b]{2}\right)^k$	linear in k $f_k = k \cdot \Delta f$
Window length	variable = $N(k)$	constant = N
Filter bandwidth	variable = f_k/Q	constant = f_s/N
Resolution	constant = Q	variable = k

Table 3.1: Comparison of CQT and DFT.

Detection was done using the constant Q transform, or CQT, which is similar to the discrete Fourier transform, however the bin size varies for the different range of frequencies.

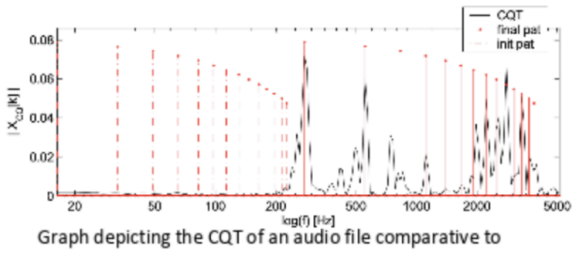
It is larger for the lower frequencies, which aids in the accuracy of the pitch detection because the discrete Fourier transform creates wide peaks for lower frequencies that are difficult to discern. In other words, the Q factor is constant throughout all frequencies with the CQT. This allows for more accurate estimation of pitches, as semitones can be equally spaced in the constant Q domain based on the bin size. For

$$B_x(\eta_1,\eta_2) = X(\eta_1)X(\eta_2)X^*(\eta_1+\eta_2) = \\ = \left(\sum_{k=1}^{4}\delta(\eta_1 \pm f_k)\right)\left(\sum_{l=1}^{4}\delta(\eta_2 \pm f_l)\right) \\ \left(\sum_{m=1}^{4}\delta(\eta_1+\eta_2 \pm f_m)\right).$$

Equation used to implement polyphonic transcription

our purposes, 12 bins were used to allow each center frequency in the constant Q domain to match up with the semitones of the equal temperament musical scale. Using this idea, we created a hypothesis of the harmonic pattern to use as a basis for pitch recognition.

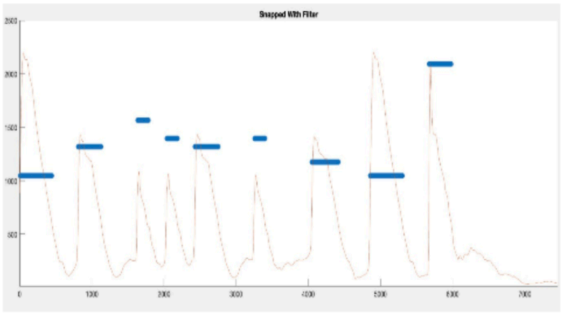
2.2 The Hypothesis Method



Graph depicting the CQT of an audio file comparative to patterns.

A typical harmonic pattern of a note played on a piano follows the overtone series. In mathematical terms, a fundamental frequency will have a harmonic at certain semitone distances away. This follows that the first harmonic will be 12 semitones above the fundamental, the second will be 7 semitones above that and so on. By then weighting these harmonics by a factor of 1/k based on the kth harmonic, we were able to create a hypothesis of weighted impulses representing where each harmonic would lie in the constant Q domain, where each semitone is represented by the 12 center frequencies.

In theory, when this hypothesis signal is convolved or cross-correlated with an audio frame, the impulses of the hypothesis should align almost exactly with the peaks in the audio frame. Thus, the largest crosscorrelation value will correspond exactly to the fundamental frequency value in monophonic audio. To bridge the gap to polyphonic transcription, subtractive analysis is used. By going back into the original CQT of the frame and subtracting

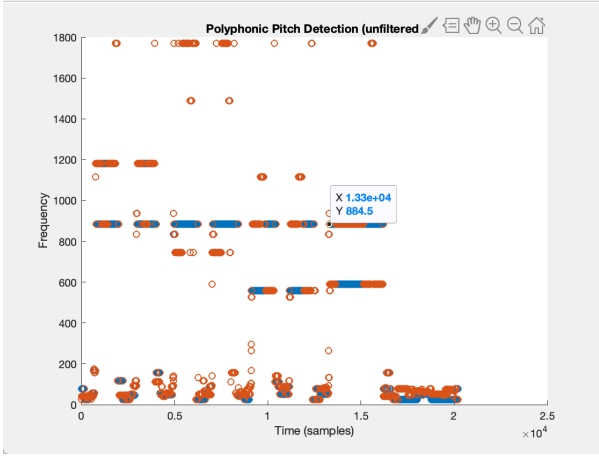


Start time of notes matched well to RMS peak values.

the energy of the coefficient by the energy of the coefficient weighted with the factor of 1/k, the harmonic context of the first fundamental note is removed. This leaves any remaining notes in tact. Thus, by re-evaluating the crosscorrelation, we are able to find the next fundamental note. In theory, we can implement this multiple times until the energy of the CQT is low enough that there aren't any notes left to be recognized.

2.3 Outliers

One consistent issue with the pitch detection was the presence of what we call outliers. Outliers are incorrect frames of



Example of Polyphonic Pitch Detection Output

detected pitch, which occur usually at the beginning of each note (where the attack of the piano is more percussive). Examples of these can be seen above. We filtered outliers through 2 methods. The first method involved making a histogram of all of the present midi frequencies in a section of audio. Anything below the average of these numbers of the counts of frequencies were removed, which was effective at removing the random one-off frequencies. The second method of filtering involved removing quiet frames entirely, as these often contained useless frequency data. The threshold for what makes a frame "quiet" is if its peak amplitude is smaller than the RMS of every peak amplitude across the entire audio clip. We found that, particularly with monophonic files, these two methods combined were very effective in removing all of the outliers.

3. RHYTHM DETECTION

Rhythm was detected using RMS peaks of the total signal spectrum. Each note's start time was determined by where significant peaks began. This is most accurately seen in the graph below, where the RMS plot is overlaid on top of the "piano roll" view of pitch detection. Originally, the note duration was determined by looking at the times when frames went from an acceptable frequency to 0. However, this required errorless pitch detection, which cannot be 100% guaranteed. The solution was to use RMS data to determine the note length, where the note duration was the difference between the peak and the point at which the slope of the RMS plot turned positive for a consistent period of time. This method ended up being more forgiving to occasional gaps in the frequency detection.

One error that occurred in terms of rhythm detection was that the rhythms of the notes were too accurate. Because of slight human error, the transcriptions would often insert 64th or even 128th note rests, even when the melody was simple eighth and quarter notes. A solution to this problem would be a sort of rhythm quantization, which would round note lengths to a predetermined value (most likely 16th).

4. EXPORTING TO A MIDI FILE

Ken Schutte's MatLab Midi code was used to write the midi file itself to be read by MuseScore. The inputs required by the code are Track Number, Channel, Velocity, Midi Note Number, Start Time (seconds), and Note Duration (seconds). For monophonic transcription, the track number and channel were both 1, and the velocity was set to 100 for every note. It was briefly considered to use the RMS values as an indication of how hard the note was played (and hence, velocity), but it was found that the RMS peaks would often change in size at certain pitches,

even when played with the same intensity, hinting that some frequencies may simply have more RMS energy than others depending on the specific piano or room that was recorded in.

5. CONCLUSION

In the end, monophonic transcription was able to export completely to midi, and polyphonic transcription was accurate enough to visualize a piano roll, although with plenty of outliers. Even with the rhythmic errors cited at the end of Section 3, the midi files played back sound just like the way they were originally played. In addition, the two-part transcription is easily scalable into 3 and 4 part transcription. This seems to justify that this code sets legitimate groundwork for more complex audio transcription, and has potential to be developed into a fully fledged audio transcription software, albeit with some serious improvements

6. REFERENCES

[1] Schutte, Ken. "Matlab and MIDI." *Ken Schutte,* 2012, kenschetste.com/midi.

[2] Wolfe, Joe. "Note Names, MIDI Numbers, and Frequencies." *Note Names, MIDI Numbers and Frequencies,* 2005, newt.phys.unsw.edu.au/jw/notes.html.

[3] "Note Input." *MuseScore.org,* 2018, muscore.org/en/handbook/note-input#enter-pitch.

[4] Vass, Jirí. "Automatic Transcription of Audio Signals." *Czech Technical University in Prague,* Czech Technical University in Prague, 2004, pp. 1-66.

[5] Brown, Judith. "Musical Fundamental Frequency Tracking Using a Pattern Recognition Method." *Musical Fundamental Frequency Tracking Using a Pattern Recognition Method,* academics.wellesley.edu/Physics/brown/pubs/cqptrv92P1394-P1402.pdf.