Notation Processing 451

Olivia Palmer
University of Michigan, Department of Biomedical Engineering

***Abstract*—A process for converting a .mp3 file of a violin melody to notated sheet music using digital signal processing is described in this work. With frequency domain analysis, the input signal was filtered with a Butterworth low pass filter to minimize background noise and higher frequencies. A rectangular window was implemented to determine the maximum frequency at each time point in the short time Fourier Transform. The frequencies were then notated using a classification algorithm which matched each frequency to a specific location on the music staff. The program has limitations in detecting note duration, fundamental frequency, and an issue of dominance of open string resonance.**

***Keywords*—STFT, DFT, FFT, filtering, music, mp3.**

I. Introduction

MUSIC analysis presents complex problems in signal processing. While identification of a single note is relatively simple, analyzing a string of notes with various pitches and rhythms becomes a difficult technical feat. Most song books of popular songs or old favorites are the result of musicians carefully listening to a recording many times over and transcribing everything by ear. As a musician who attended a performing arts high school, I often grappled with the painstakingly time-consuming task of transcribing a song for a group of musicians. The prospect of automated musical notation presents great opportunity for saving time and improving sharing methods between musicians.

The Notation Processing software attempts to reach this goal within certain limitations. For the purposes of this project, I will focus on interpreting signals from a single violin melody played within the middle range of the instrument, or approximately within the D4 octave. Recordings were made with a Sony Digital Recorder, with recordings under 20 seconds long. The program can handle longer clips, but the entire clip is notated to a single line which can become overcrowded with longer recordings.

II. Methods

A. Initial Filtering

Upon initial recording of the audio signal there is inevitably some background noise that should be minimized prior to frequency classification. This is best done through visualization of the frequency spectrum of a signal. The fast Fourier transform, or FFT, rapidly computes the factorized discrete Fourier transform. The discrete Fourier transform of signal x_n is given by Eq. 1. Figure 1 shows the signal from a sound clip of a violin playing an arpeggio and its associated Fourier transform in the frequency domain.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \tag{1}$$

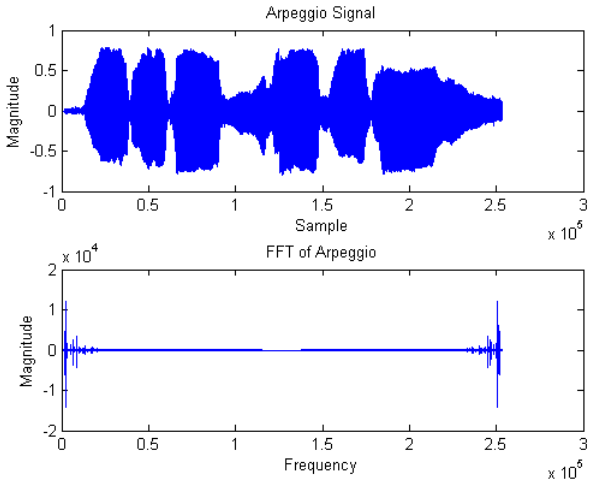

Fig. 1. Signal of arpeggio (top and FFT of signal (bottom).

When we zoom in and look at the absolute value of the frequency spectrum for the region of interest (low frequencies), the background noise becomes visible in the signal (Fig. 2, top). Since we are only interested in the frequencies played for the middle range of a violin, the signal must be filtered to elinimate higher frequencies associated with background noise. This was done using a Butterworth low pass filter. The Butterworth filter has a maximally flat frequency response in the passband, at the expense of a wide transition band. Its frequency response, $H(\omega)$ is given by Eq. 2, where ϵ is the maximum passband gain and n is the filter order.

$$H(j\omega) = \frac{1}{\sqrt{1+\epsilon^2(\frac{\omega}{\omega_p})^{2n}}} \tag{2}$$

As shown in Fig. 2, the Butterworth low pass filter has eliminated higher frequencies due to background noise. This will minimize errors when detecting the frequencies of the notes that were played in the recording.

B. Identification Filtering

Once the signal has been filtered to minimize background noise, the actual notes played must be identified. This was

O. Palmer is with the Department of Biomedical Engineering, University of Michigan, Ann Arbor, MI, 48109 USA e-mail: opalmer@umich.edu.

Received Deceber 9, 2014.

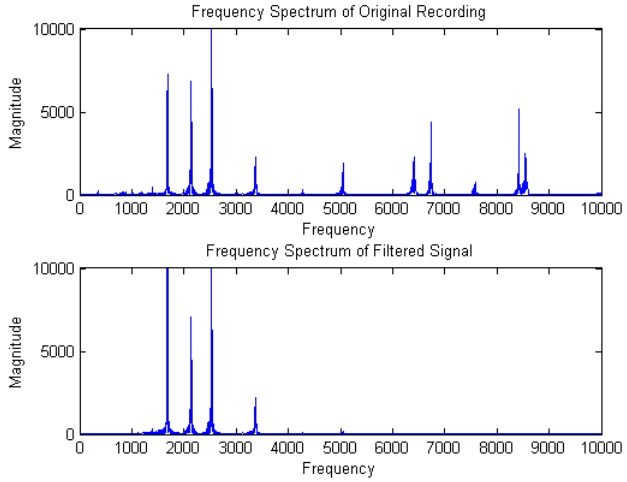

Fig. 2. Zoomed-in frequency spectrum of original signal (top) and filtered signal (bottom).

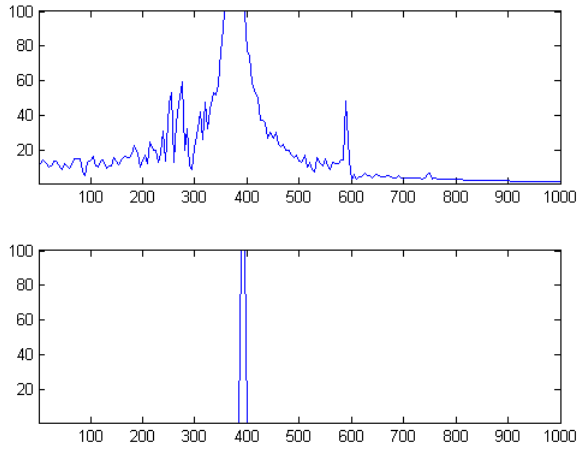

Fig. 3. Frequency spectrum of a time clip (top), and the same frequency spectrum with a rect window applied (bottom).

done by dividing the recording into smaller time clips of 0.2 seconds in length. This length was chosen because it is short enough to not skip over any notes played within an eighth note range, and long enough to avoid moments of total silence or space between notes. Next, a rectangular window is defined for each of the frequencies of interest (in this case, the notes in the middle range of a violin). The rect window is given a width of 6 Hz. This width was chosen to be small enough to avoid overlapping with the frequency of adjacent notes while remaining wide enough to detect notes that are not perfectly in tune. The filtered frequency spectrum of each time clip is multiplied by the rect filter at each note. The amplitude of the rectangular filtered spectrum is maximum when the rect window is centered at the frequency of the note being played (as shown in Fig. 3) and has a smaller amplitude when the note is not played during the time clip (see Fig. 4).

The set of rectangular windows could be expanded beyond the middle range of the violin for a broader range of note detection. Each note has several associated harmonics, however, that in some cases have a higher power density than the fundamental frequency. When expanding to a broader range, it would be beneficial to find the maximum amplitudes within each octave set. If the note detected was D5, for example, and there was a relative maximum in the D4 octave as well, it is more likely that the note being played is a D4 and the D5 that was detected was a harmonic with higher power.

C. Classification

The process of writing the sheet music for the audio recording requires a method of note classification. For this project, I defined a music staff with lines spaced 1 unit apart on a plot and used a series of lines to draw a treble cleff. This could be expanded to a grand staff for the use of reading piano or choir music, or bass cleff for lower register instruments. To classify each note played, the maximum amplitude of the rect-filtered frequency clip is found for each time clip. The frequency that

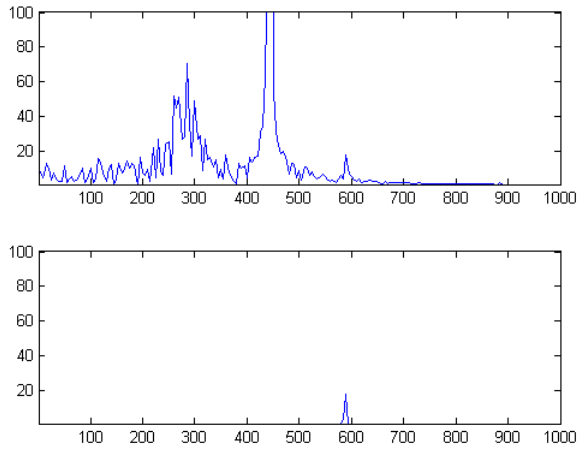

Fig. 4. When the rect window is centered at a frequency that is not being played in the current time clip, the amplitude will be low (bottom).

the rect window defines at the maximum amplitude is taken to be the frequency of the note played over that 0.2 second interval. Each note is matched to a specific location on the treble cleff, and plotted across x as a function of the clip number, spaced at equal distances apart.

As shown in Fig. 5, each note was played over an average of 3 or 4 time clips. For a cleaner representation of which notes were played, only the changing notes are plotted in Fig. 6. Plotting the changing notes prevents any visualization of note duration, but makes for a cleaner and more understandable score. Additionally, the number of times a single note is repeated on the plot (Fig. 5) is not accurately representative of note duration due to the arbitrary selection of time clip durations and resonance dominance of open strings.

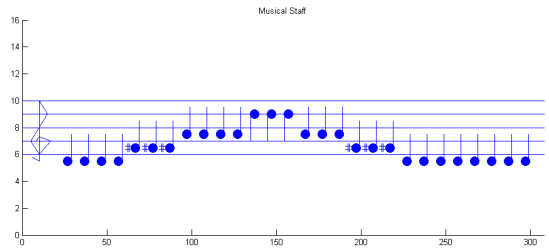

Fig. 5. Frequency plotted for every time point of arpeggio recording.

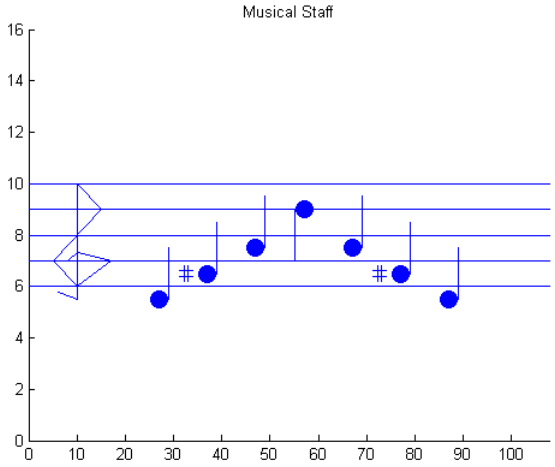

Fig. 6. Notation output of an arpeggio recording.

D. Spectrogram

The spectrogram proves to be a useful tool in finding sources of errors in the notation classification. The spectrogram provides a visual representation of the short time Fourier transform, given by Eq. 3 where w is the window. Each time window is plotted on the x-axis, with frequency plotted on the y-axis. The amplitude of each frequency at each timepoint is represented by the intensity of the color of each window.

$$X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{j\omega n} \tag{3}$$

Looking at the spectrogram shown in Fig. 7, it is clear that the fundamental frequency does not always have the highest power. Before multiplying each time clip by a series of rect windows, I had tried to extract the frequency played by analyzing spectrogram data. This becomes difficult because of the dominance of harmonics and the unpredictable nature of the power density function. Some notes have their highest power in the 1st or 2nd harmonic, while for other notes it occurs in the 3rd or 4th. I initially attempted to find the maximum power at each time window and set a threshold to be 70 percent of that maximum power, as shown in Fig. 8, but that method was not reliable with the large variance in harmonics.

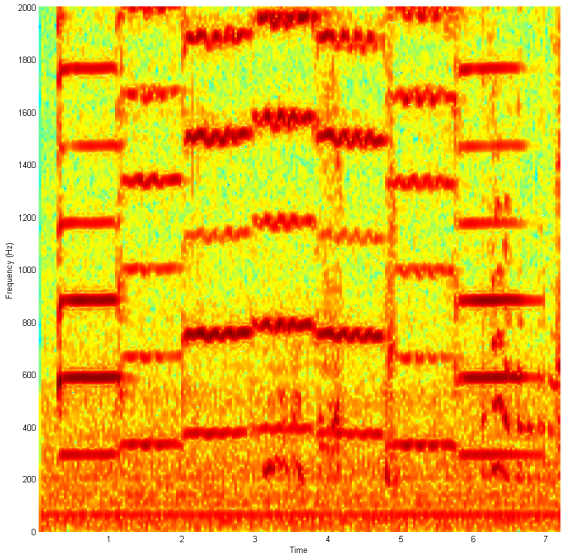

Fig. 7. Spectrogram of brief scale sound clip.

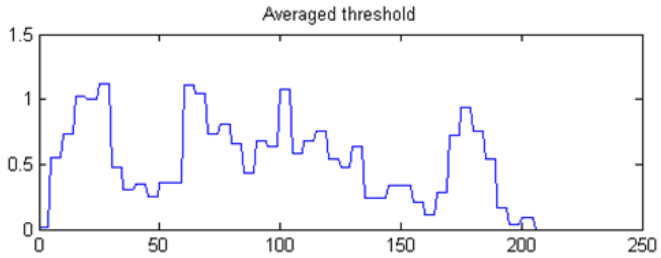

Fig. 8. Threshold values from spectrogram.

Another issue made apparent by the spectrogram is the dominance of open string resonance on the violin. The violin's open strings are G, D, A, and E. For the middle range, I saw this most with the open D and A strings. Figure 9 shows the spectrogram of a short melody. The circled note, a B natural, is misread as a D since the open string D was played immediately before this note and is still resonating at a higher power than the note played during that time interval. The notation output from this recording, shown in Fig. 10, shows the comparison between the actual notes played (bottom) and the output of the program, with the location of the missing B natural circled.

Attempting to correct this issue, I set the program to look for the next largest amplitude in the output of rectangular windowed time clips after an open D or A string were detected. This had more luck in detecting other notes including the missing one, but was unreliable in correct notation output because it detected notes that were really just noise in the signal. Looking at Fig. 3, the next highest amplitude would put the frequency at around 294 Hz, when the frequency being played is 392 Hz.

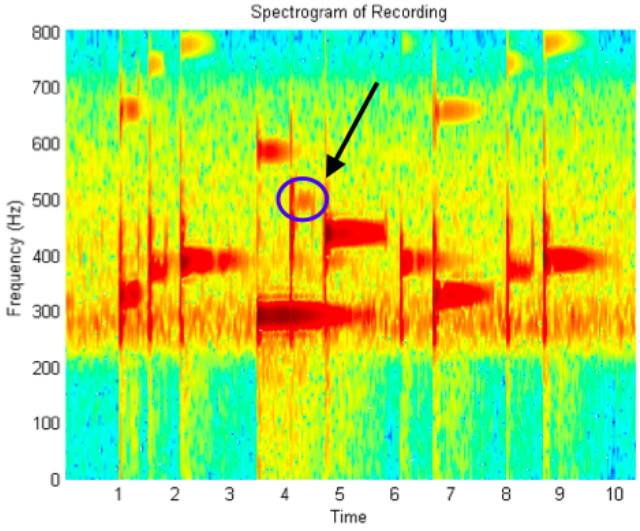

Fig. 9. Spectrogram of the "Let It Go" melody from Frozen. The open D string has a dominant resonance, causing some notes to be misread as D such as the B (circled).

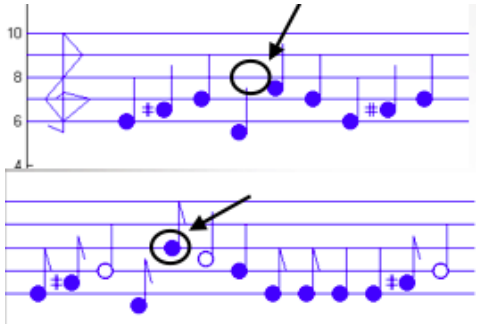

Fig. 10. Comparison of program output (top) to the actual notes played with approximate rhythms (bottom, not made with this program).

III. Other Possible Methods

Another method that could be implemented is cross correlation. Cross correlation measures the similarity between two waveforms. The cross correlation of two signals x and y is given by Eq. 4. I did not implement cross correlation for this project because the rectangular window proved to be more reliable. For notes that were not completely in tune, the cross correlation for a specific frequency did not equal 1 at any point. In instances where more than one frequency are apparent in a time clip, cross correlation is not as useful for determining which note is actually sounding. This could be improved by analyzing what notes are played in previous time clips and future time clips to see which note is more likely being played at that time point. Another possibility would be to throw out clips that have more than one frequency sounding at once, but this runs the risk of missing notes and losing data. This method could be improved by adding beat detection and segmenting the song based on fractions of a beat rather than arbitrary time clips.

$$x[n] * y[n] = \sum_{m=-\infty}^{\infty} x^*[m]y[m+n] \tag{4}$$

The notation software could be improved by incorporating note duration. Duration detection becomes complicated when the same note is sounded more than once. In this instance there would have to be brief moment of silence between repeated notes to distinguish that the same note had been played more than once. The previous note played, however, will still be resonating somewhat during the brief clip between notes. Repeated notes may be more distinguishable on a piano or when played with exaggerated staccato bowing. As is, the notes do not have distinguishable edges visible in the spectrogram.

Duration could also be accomplished using beat detection. Most humans are easily able to tap their foot to the beat of a song, but making the process automated is much more difficult, especially in instances of accelerando, ritardando, or a grand pause. One solution could be to have the user input a temp, and begin dividing the song into segments determined by temp starting at the detection of the first note. This method would not account for changes in tempo, however. Beat detection may be more accessible in songs with a drum set tracking the beat, which would be apparent across the frequency spectrum in the spectrogram.

Without beat detection, duration could be assumed by looking at the number of time clips over which a specific frequency is played. This would be a rough estimate because, as seen in Fig. 5, although every note but the final note was played for about the same amount of time, the number of timeclips representing each frequency varied from 3 to 4, with the open strings tending to be represented over more time clips due to resonance dominance. Duration would also be dependent on the tempo of the song and the duration of each time clip.

IV. Discussion and Conclusions

The complexity of audio recording analysis shows how intricate and accurate the human ability to interpret audio truly is. This program is able to identify the notes played for a monophonic recording of a stringed instrument. It would be interesting to test the differences if a recording of a wind instrument was used, in which sound is produced through vibrating air columns rather than strings. Music signal analysis with digital signal processing is a rapidly growing field of research, and there is a lot of room for improvement especially in polyphonic analysis. The complexity of music explains why this research has lagged behind developments in speech recognition software. Overall, this project was able to build a notation software using frequency domain analysis, various filters, spectrogram analysis, and a classification algorithm.

References

[1] Oppenheim and Schafer, *Discrete Time Signal Processing*, 3rd ed. Upper Saddle River, NJ: Pearson, 2010.

[2] M. Muller, D. Ellis, A. Klapuri, and G. Richard, *Signal Processing for Music Analysis*. J. Selected Topics in Signal Processing:2011.

[3] Z. Settel and C. Lippe, *Real-time Frequency-Domain Digital Signal Processing on the Desktop*. Buffalo, NY, USA:Hiller.