Coding Project 2: Parsing musical frequency signatures

Justin Hexem

Abstract

In this project we analyze a sound clip of a song by looking at the frequencies that are present. Different instruments have different frequencies, and by filtering out some of these frequencies we can isolate each instrument that was played in the song. We will specifically isolate the drumbeats by filtering out all of the higher frequencies, and the guitar by filtering out many of the higher and lower frequencies to isolate the mid-range ones.

1 Introduction

Each musical note has exactly one frequency that corresponds to it. For example, a middle C has a frequency of about 261.63 Hz. This means that for any given song, we can analyze the frequencies of the song and determine which part of the signal came from each instrument that composed it. Instruments like drums will have lower frequencies than instruments like flutes, so we can use this difference in frequency space to isolate only the frequencies that we desire.

In the sound clip that we are analyzing, there are a few different instruments that compose the song. We will be taking a look at the contributions of the drums and the guitar in this analysis, and we will be using a modified version of the Fourier transform to isolate each frequency in the song. This version is called the Gabor transform, and it used a sliding window function to obtain a more precise estimate of the location of the key frequencies that we are searching for. We will then plot an image of the spectrogram for

this song, which is a colored image that colors each present frequency at all time points in the song with a color corresponding to the amplitude of that frequency. These visual estimates will then be used to isolate the sound of just the drumbeats and then we will isolate just the guitar sounds.

2 Theoretical Background

We will now discuss the derivation of the previously mentioned Gabor transform and its discrete counterpart in order to understand how we can isolate certain instruments from the sound clip that we are analyzing.

2.1 The Gabor Transform

Due to the complexity of our signal, the normal Fourier transform will not be very useful here. We must find a way to have more control over the uncertainty principle in order to optimize for as much accuracy as possible in both time and frequency space. We can do this by only looking at a small portion of our signal at a time, so we can apply a filter that we can slide back and forth over our signal to isolate a segment of it. Let us call our original signal $f(t)$ and this new filter function $g(t)$. Applying a Fourier transform to this filtered version of our function gives us more clarity on which signals are present in our song at each given moment in time. We will denote the center of our filter function g by the variable τ. In mathematical notation, we may write this out as:

$$\tilde{f}_g(\tau, k) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-2\pi ikt}dt. \tag{1}$$

This gives us a new transform called the Gabor transform. We see that we have taken our original signal from the time domain into the (τ, k) domain where τ and k are the center of our window and the frequency variable respectively. This transform also has a corresponding inverse transform, which can be written as:

$$f(t) = \frac{1}{2\pi||g||_2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\tilde{f}_g(\tau, k)g(t-\tau)e^{2\pi ikt}dkd\tau. \tag{2}$$

The $||g||_2$ symbol refers to the L-2 norm of our filter function g. This new transform allows us to have much more control over how precise we would

like our measurements to be in either the temporal or the frequency domain. We can choose a more spread out filter function to obtain more certainty in frequency space, or we can choose a much more narrow filter to obtain more certainty in our time domain.

2.2 The Discrete Gabor Transform

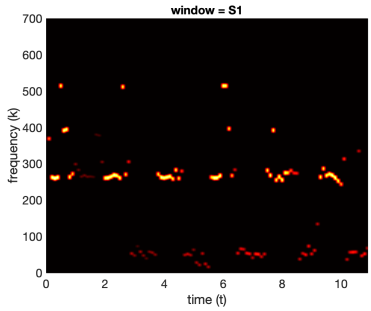
In order to be able to implement this new version of the Fourier transform, we need a way to discretize it. We will consider the following discretization $k = m\omega_0$ and $\tau = nt_0$, where t_0 is the step size and n is the number of steps. The variables ω_0 and m are the associate values in the frequency domain. This gives us the discretized Gabor transform:

$$\tilde{f}_g(\tau, k) \approx \int_{-\infty}^{\infty} f(t)g(t - nt_0)e^{-2\pi i m\omega_0 t}dt. \tag{3}$$

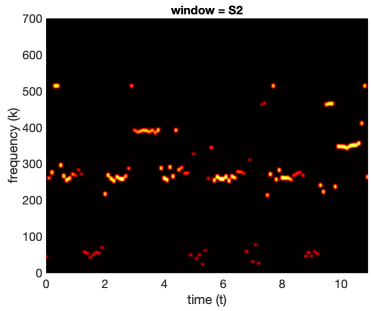
We see that we can now evaluate this integral numerically as opposed to the previous definition of the continuous transform. However in practice, we would make use of the fft algorithm by doing fft to the product of our filter function and our signal.

3 Numerical Methods

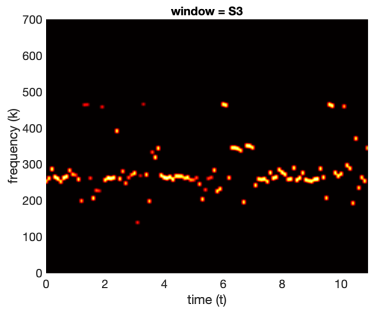
Now we will use MATLAB to isolate the drum and guitar sounds from the original sound clip. Our goal here is to create a spectrogram of our original sound clip so we can see where all of the different frequencies in the clip appear. First we will discretize our time domain, frequency domain, and our tau values that we will be sliding the filter function along. Then we will split our sound clip into four different windows for efficiency purposes. We will then create a spectrogram for each one of the four split clips by using a for loop to loop through each of the windows. For each window, we will define a filter function g that is centered at each point along our discretized τ variable. To perform the Gabor transform, we just have to multiply our filter function to the window of our sound clip and make use of the fft alrogithm. We need to remove some noise from our signal, so we will apply a Gaussian filter around the peak frequency for that value of τ. After filtering, we will apply this new filter function and save the result to a new matrix. This new matrix is our spectrogram that we can then plot to see all of the different



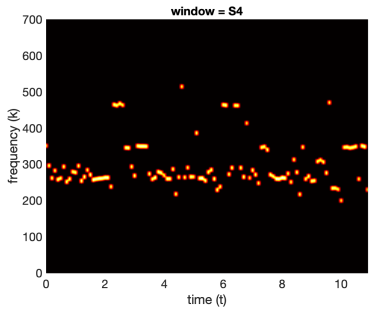
(a) Spectrogram from $t = 0$ to $t = 10$



(b) Spectrogram from $t = 10$ to $t = 20$



(c) Spectrogram from $t = 20$ to $t = 30$



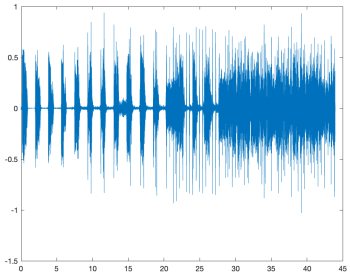
(d) Spectrogram from $t = 30$ to $t = 44$

Figure 1: Spectrograms for each Time Window

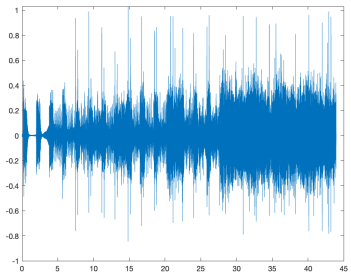
frequencies in the original sound clip. After making the spectrograms for each window, we can then examine the spectrogram to find the drum frequencies. The drums seem to have frequencies between 0 Hz and 200 Hz, so we will set all frequencies that are greater than 200 Hz to zero. To do this we just use the fft algorithm on our original signal, remove the unwanted frequencies, and then transform back to get the sound clip with our drumbeats. We can isolate the guitar frequencies in the same way. We apply the fft algorithm to our original sound clip, set all frequencies to zero that are outside the window of 250 Hz to 1200 Hz, and then transformm back to retrieve our guitar sounds.

4 Results

We may now analyze the spectrogram of our original sound clip to find the frequencies of our drumbeats and guitar sounds. Figure 1 shows each of the



(a) Drums Signal



(b) Guitar Signal

Figure 2: Time vs Amplitude Plots

four spectrograms for each segment of our original sound clip. We see in figure (1a) that there are not very many frequencies playing during this time in the original sound clip, and this becomes much more apparent when comparing it to figure (1d) where there are many more frequencies playing. We observe in figure (1a) that there are faint frequencies below 200 Hz, and we can recognize this as our drumbeat. Most of the frequencies above 200 Hz are different string instruments, which include the guitar sound that we are looking for. This is how we knew exactly which frequencies to target when isolating our drumbeat and guitar sounds.

After we have isolated our drumbeat frequencies, we can make a time vs amplitude plot to examine how our signal has changed (see figure (2a)). When we compare this to a time vs amplitude plot for our guitar sounds (see figure (2b)), we notice that the spikes in the drumbeat plot are much more separated which is what we would expect from the drums. The guitar is much more blended together throughout the signal, which makes sense due to the fact that the guitar melody is much more complex than the drum bassline.

5 Conclusion

By implementing the Gabor transform in MATLAB, we were able to create a spectrogram for our signal that we used to isolate the drumbeat and the guitar sounds. We made use of a filter function in the time domain that

could be used as a sliding window for our original sound clip, and we used a frequency signal to remove some of the noise. We then used the spectrogram to find the frequencies for the drumbeat and the guitar sounds, and removed all other frequencies from those signals.

This project taught me a lot about analyzing music using the tools that we have been discussing in class. I learned how to implement the discrete Gabor transform, make a spectrogram, and apply a filter to select certain frequencies from a given signal. Some possible drawbacks of the methods that were used here could be having to visually examine the spectrogram to select the frequencies that we wanted to isolate. This could have led to possibly selecting frequency ranges that are too narrow or too wide, which would include some undesired frequencies. I have definitely heard the song from somewhere, but I am not sure where. It sounds like it could be pirate themed, so maybe it is in the movie the Pirates of the Caribbean? I really enjoyed working on this project because this is my first experience applying math to anything music related.

Acknowledgment

I appreciated all the help that Professor Rahman and Katherine Johnston provided while I worked through this project. Thank you for taking the time to help me understand certain details of the problem that I had initially overlooked.