AMATH 482 Homework 2

Jonah Wu

February 10, 2020

Abstract

We will analyze a portion of two of the greatest rock and roll songs of all time: Sweet Child O' Mine by Guns N' Roses, and Comfortably Numb by Pink Floyd. Using Gabor filtering, we will reproduce the music score for the guitar for the first clip, and the bass in the second clip.

1 Introduction and Overview

The Fourier transform can be used to turn musical wave signals into frequencies and amplitudes; however, after we transform the waves into frequency domain, we lose information with time. For example, we would know what notes were used in the clip, but we wouldn't know when the notes are played. Thus, we will be using Gabor Transform, which a short-time Fourier transform and can be used to determine the frequencies of local sections of the music waves as we can perform this over time.

2 Theoretical Background

2.1 Fourier Transform

The Fourier Transform breaks down a function or signal of waves into an alternate representation of sines and cosines, showing that we can rewrite any waveform as the sum of sinusoidal functions. Suppose we have a function $f(x)$ with $x \in \mathbb{R}$, the Fourier Transform is defined as:

$$\hat{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ikx} dx \tag{1}$$

In addition, the Fourier Transform has an inverse and is defined as:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(k) e^{ikx} dk \tag{2}$$

In summary, the Fourier Transform transforms a function of space or time, x, into a function of frequencies, k. However, the above Fourier Transform is an integral over infinity, which is not feasible for our computing software such as MATLAB. Instead, we will be using the Fast Fourier Transform for transforming data over a finite interval. In addition, when performing over a large data, Fast Fourier Transform is much faster than the original Fourier Transform. Other differences between the Fast Fourier Transform and the original one are that Fast Fourier Transform centers a frequency of 0 and that it works in a domain of period 2π. As a result, we must re-scale the frequencies by $2\pi/2L$ since the Fast Fourier Transform assumes 2π periodic signals.

2.2 Gabor Transform

The Gabor Transform is a short-time Fourier transform and can be used to determine the frequencies of local sections of the music waves. Gabor Transform is defined as:

$$G_f(\tau, k) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-ikt}dt \tag{3}$$

where $g(t-\tau)$ is a Gaussian:

$$g(t-\tau) = e^{-\alpha(t-\tau)^2} \tag{4}$$

Gabor Transform allows us to pick a value of τ to focus on what time section we want to focus on within the music clip. As you can see from the equation above, the $g(t-\tau)$ function is a Gaussian which centers our choice of τ to focus on. In addition, we can specify how big the window of this localizing we want with the value α. The larger α is, the smaller the window we are going to have because the negative sign in front of α in the power of e.

3 Algorithm Implementation and Development

The algorithm we are going to use is Gabor Transform over the music clips. It allows us to know information about frequencies over time. We perform Gabor Transform on every τ within the time of the clips, and we can save them into matrix of frequencies v.s. time in order to create spectrograms for each clip. After we import the music clips and convert them to vectors representing the music, we re-scale our wave number k by 1/L in order to scale from radians to hertz, which is what we want to use for our frequencies in order to match the scores. Next, we define α and τ for the equation. As I mentioned before, α determines how wide the window should be, so larger α gives smaller window, which is going to be precise; however, we cannot make the window too small or we are not going to be able to observe the frequency, in which I find $\alpha = 50$ to be the best for our vectors. The MATLAB code for this part can be found in **Appendix B**, and **Appendix A** explains the MATLAB functions that are used.

Algorithm 1: Gabor Transform

Import clips from `GNR.m4a` and `Floyd.m4a`
`radioread()` for both clips to get wave vectors
Initialize Gabor Transform space
for each τ **do**
  Initialize Gaussian window $g(t-\tau)$
  Multiply g(t - τ) with the wave vectors
  Perform Fast Fourier Transform on the result using `fft()`
  Store the absolute values of shifted back vectors using `fftshift()`
end for
Plot spectrogram with desired frequencies

After we perform Gabor Transform, we will have a 2-D array that stores the frequencies at given time t, we then can plot the result into a spectrogram. In addition, we can try to isolate the bass by specifying what range of frequencies we want: 50 - 200 for bass, or 200 - 800 for guitar by setting `ylim` to those values.

4 Computational Results

4.1 Guitar in Sweet Child O' Mineby Guns N' Roses clip

Since we want to focus on the guitar part, we set the limits of y-axis to 200 - 800, and the sprectrogram for this song is the following:

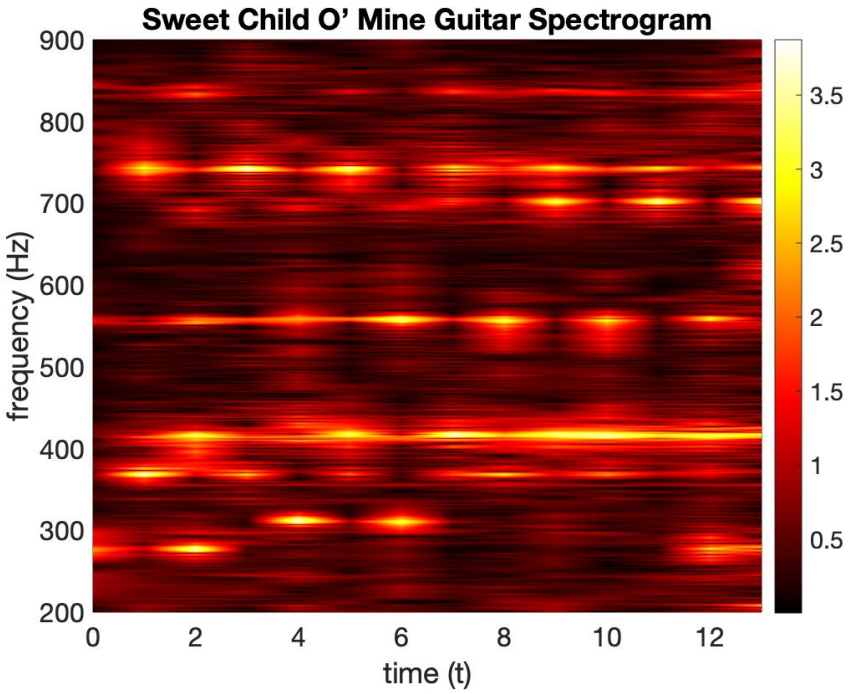

Figure 1: Sweet Child O' Mine Guitar Spectrogram

4.2 Bass in Comfortably Numb by Pink Floyd clip

We use a filter in frequency space to try to isolate the bass by setting the limits of y-axis to 50 - 200 in Comfortably Numb, and the sprectrogram for the bass of this song is the following:

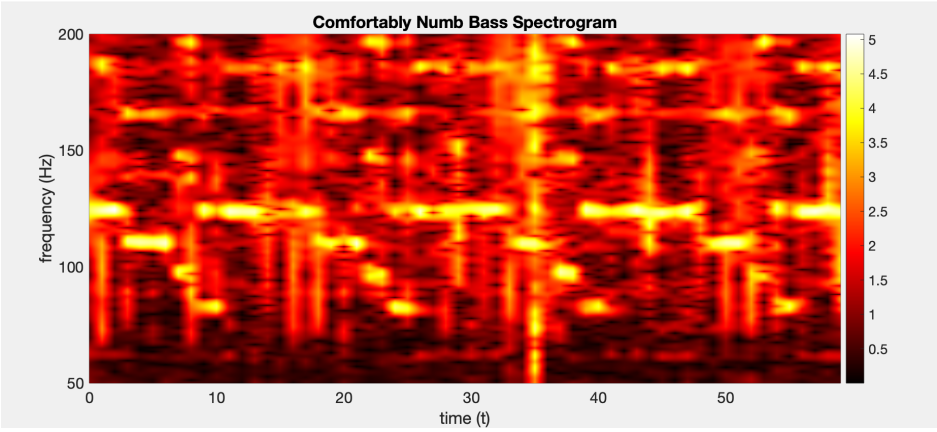

Figure 2: Comfortably Numb Bass Spectrogram

4.3 Guitar in Comfortably Numb by Pink Floyd clip

Since we want to focus on the guitar part, we set the limits of y-axis to 200 - 800, and the sprectrogram for the guitar of this song is the following:

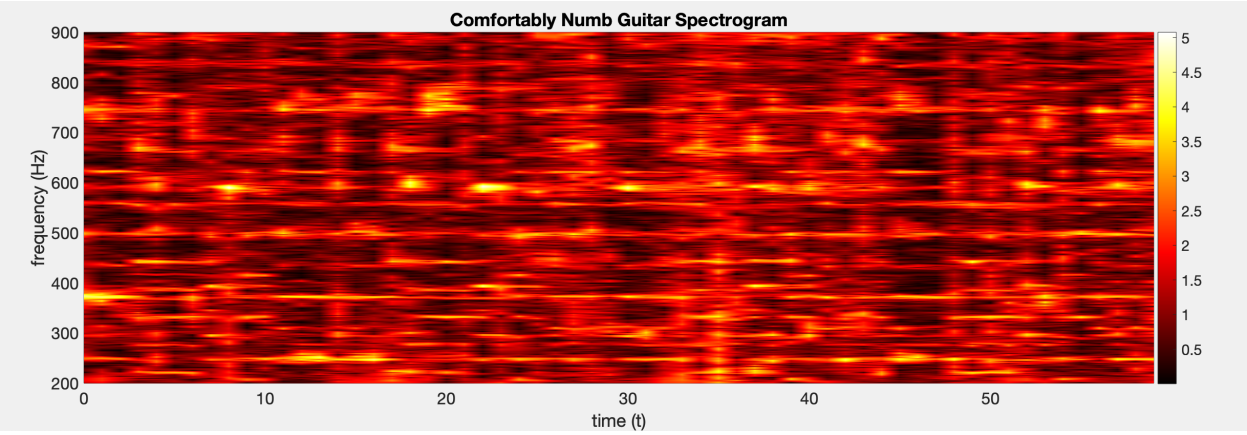

Figure 3: Comfortably Numb Guitar Spectrogram

5 Summary and Conclusions

The Fourier Transform breaks down a function or signal of waves into an alternate representation of sines and cosines, showing that we can rewrite any waveform as the sum of sinusoidal functions. However, we lose information with time with Fourier Transform since it only tells you the frequencies and amplitude. With Gabor Transform, where we can define a small window and iterate through time, we then can specify which frequencies were found at any local time. Thus, through this assignment, we learn how to utilize Gabor Transform when we also need the time information using Fourier Transform.

Appendix A MATLAB Functions

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `L = length(X)` returns the length of the largest array dimension in X.
- `Y = abs(X)`: returns the absolute value of `X`.
- `Y = exp(X)`: returns the exponential e^x for each element in array `X`.
- `M = max(A)` returns the maximum elements of an array.
- `Y = log(X)` returns the natural logarithm ln(x) of each element in array X.
- `Y = fftn(X)`: returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm.
- `Y = fftshift(X)`: rearranges a Fourier transform `X` by shifting the zero-frequency component to the center of the array.
- `[y,Fs] = audioread(filename)`: reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.
- `ylim` sets the y-axis limits for the current axes or chart.
- `pcolor(X,Y,C)` specifies the x- and y-coordinates for the vertices. The size of C must match the size of the x-y coordinate grid. For example, if X and Y define an m-by-n grid, then C must be an m-by-n matrix.
- `colormap(map)` sets the colormap for the current figure to the colormap specified by map.
- `colorbar` displays a vertical colorbar to the right of the current axes or chart.

Appendix B MATLAB Code

```
%% GNR
[y1, Fs] = audioread('GNR.m4a');
tr_gnr = length(y1)/Fs;
n = length(y1);
L = tr_gnr;
k = (1/L)*[0:(n/2-1) -n/2:-1];
ks = fftshift(k);
a = 50;
ts = linspace(0,L,n+1);
t = ts(1:n);
tau1 = 0:1:L;
for j = 1:length(tau1)
    g = exp(-a*(t-tau1(j)).^2);
    G = g.*y1';
    G = fft(G);
    gnr_spec(j,:) = abs(fftshift(G));
end
pcolor(tau1,ks,log(gnr_spec+1)')
shading interp
set(gca,'ylim',[200, 900],'Fontsize',15)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (Hz)')
title("Sweet Child O' Mine Guitar Spectrogram")

%% Floyd
[y2, Fs] = audioread('Floyd.m4a');
tr_floyd = length(y2)/Fs;
n = length(y2);
L = tr_floyd;
k = (1/L)*[0:(n/2-1) -n/2:-1];
ks = fftshift(k);
a = 50;
ts = linspace(0,L,n+1);
t = ts(1:n);
tau2 = 0:1:L;
for j = 1:length(tau2)
    g = exp(-a*(t-tau2(j)).^2);
    yg = g.*y2';
    ygt = fft(yg);
    floyd_spec(j,:) = abs(fftshift(ygt));
end

%% Bass Spectrogram
pcolor(tau2,ks,log(floyd_spec(:,1:end-1)+1)')
shading interp
set(gca,'ylim',[50, 200],'Fontsize',15)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (Hz)')
title("Comfortably Numb Bass Spectrogram")

%% Guitar Spectrogram
pcolor(tau2,ks,log(floyd_spec(:,1:end-1)+1)')
shading interp
set(gca,'ylim',[200, 900],'Fontsize',15)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (Hz)')
title("Comfortably Numb Guitar Spectrogram")
```