AMATH 482 Homework 2

Jeremy Lu

February 10, 2021

Abstract

We are analyzing 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. Given the m4a files for these songs, we want to reconstruct the music score for the guitar and the bass solo in *Sweet Child O' Mine* and *Comfortably Numb*, respectively. Using MATLAB in this project, we implement the Gabor Transform, which reveals to us the frequency and time information of both songs, allowing us to reconstruct their music scores.

1 Introduction and Overview

The frequency signature of a signal is often analyzed through the use of Fourier Transforms. However, the drawback of this technique is that time information about the signal is lost. This would not be a problem if the signal being analyzed were constant and uniform; but in the application of analyzing the notes in a song, the Fourier Transform will reveal the most prominent notes, not when they are being played.
In order to analyze frequency signature without losing time information, we can employ the Gabor transform, which is series of Fourier Transforms over small time windows. The result of this is frequency information over windows of time, allowing us to know both which frequencies are active, and when they are active. We also use a Shannon filter to remove overtones, and focus only on the guitar and bass frequencies.
This sort of analysis, often called "time-frequency analysis", is best visualized through a graph known as the spectrogram. The spectrogram plots the active frequencies throughout time, using color to denote the strength of each signal. Using these graphs, we are able to see which notes are being played, and reconstruct a music score in our two chosen songs.

2 Theoretical Background

2.1 Fourier Transform

The Fourier Transform is an important tool that allows any function $f(x)$ to be written as the sum of sines and cosines. In the domain $x \in [-\infty, \infty]$, the this can be written as:

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

Additionally, the Inverse Fourier Transform is defined as:

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

Essentially, the Fourier Transform is able to decompose a signal in the time domain into its frequency components. However, this comes at a cost—a narrow signal in the time domain will have a wide spread in the Fourier domain, and vice versa.
Note that the original Fourier Transform is an integral over infinity, which takes nontrivial processing power to compute using software such as MATLAB. Instead, the Fast Fourier Transform (FFT) is employed as it operates under $O(n \log n)$ time, which is ideal for large data. The FFT is able to run so quickly by discretizing

$x \in [-L, L]$ with 2^n points. Two key characteristics of the FFT are that it centers on zero frequency and that it works in a domain of period 2π. Because FFT shifts the data to fit this zero frequency requirement, it is important to shift it back in order to gain meaningful frequency data. Additionally, it is necessary to rescale the frequency data by multiplying it with $\frac{2\pi}{L}$ in order to fit this requirement.

2.2 Gabor Transform

As mentioned before, the Fourier Transform is unable to keep both time and frequency information. The Gabor Transform addresses this by performing Fourier Transforms over small windows. The equation for the Gabor Transform using a Gaussian window is described as,

$$g(t-\tau) = e^{-a(t-\tau)^2} \tag{3}$$

Where a describes the width of the window, and τ represents the center of the window. Note that we have a trade off with the a we select. If a is smaller, we will have less frequency data, but if a is larger, we have less accurate information about when these frequency signals occur. To balance time and frequency information, it is best to compare multiple spectrograms with different bandwidths.

2.3 Shannon Filter

The Shannon Filter, or "top hat filter" is used to focus only on certain frequencies. It's equation is defined as:

$$s(x) = \begin{cases} 1 & x \in (i, j) \\ 0 & x \notin (i, j) \end{cases} \tag{4}$$

where (i, j) are the bounds which we wish to focus on. This is also represented in graphical form as:

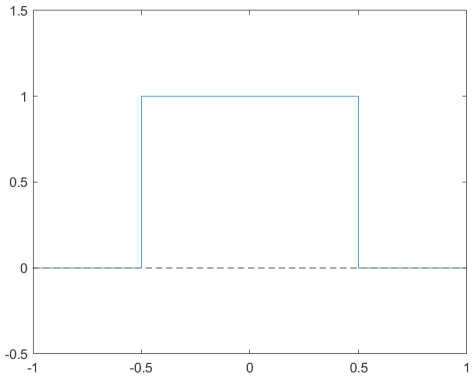

Figure 1: Shannon filter, with bounds (-0.5, 0.5)

In this project, instead of implementing traditionally, we used the `ylim` command in MATLAB, which limits the visible y axis, effectively removing frequencies which we do not want.

3 Algorithm Implementation and Development

The algorithm used in this project is the Gabor Transform, which reveals to us which frequencies are active and when they are active. Plotting the transformed data then allows us to determine visually the notes being played. The MATLAB code for this is available in **Appendix B**.

Before performing these algorithms, there were several steps to prepare the signal data read from the .m4a files. First, we retrieved the length of the audio, to create a vector of time. We also rescale our wave number k with $1/L$ as we want to use hertz instead of radians for our measures of frequency. Then, we make sure

to define values for a and τ, which will determine the width and the centers of our windows for the Gabor Transform.
With these steps complete, we can then perform the Gabor Transform on the signal and use the output to plot spectrograms.

Algorithm 1: Gabor Transform Algorithm

for $j = 1 : 49$ **do**
  Define Gaussian Window function
  Multiply window function by the signal
  Perform Fast Fourier Transform on section of signal using `fft`
  Use `fftshift` to shift back and take the absolute value to store for spectrogram
end for

3.1 *Sweet Child O' Mine* Spectrograms

For *Sweet Child O' Mine*, we used a τ value of 0.5, and multiple a values, to get 4 spectrograms:

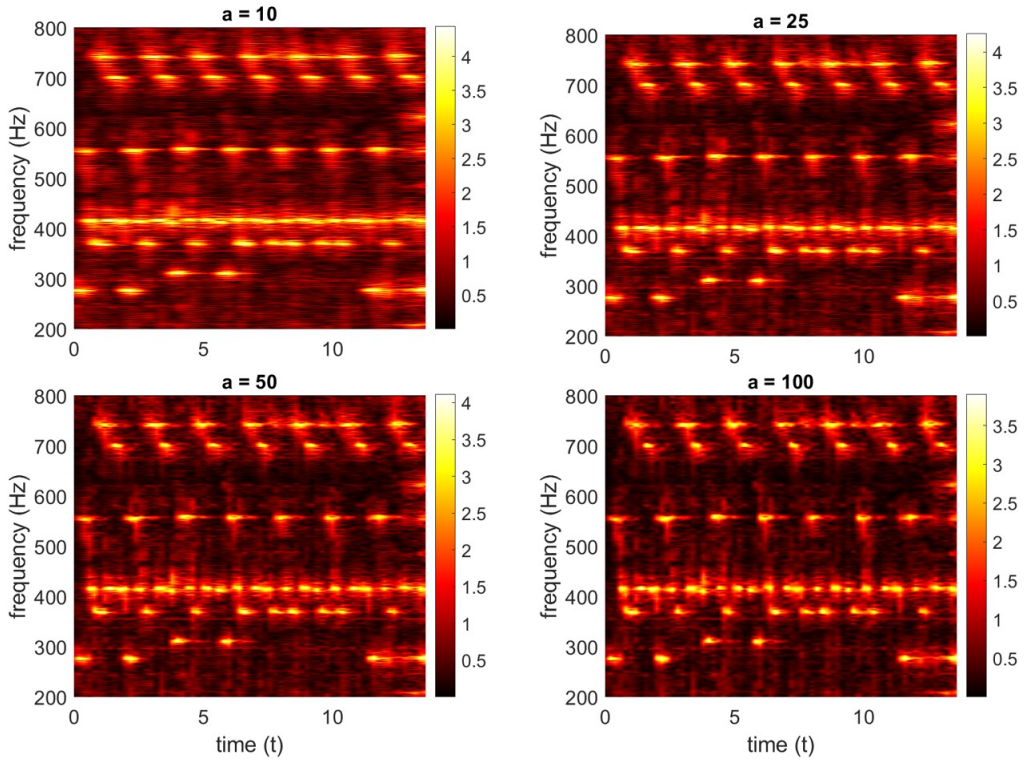

Figure 2: Guitar spectrogram with $\tau = 0.5$ and multiple a, of *Sweet Child O' Mine* by Guns N' Roses

We focused on frequencies from 200 to 800, as these are guitar frequencies, with the higher ones being overtones. From these graphs, we can clearly see 6 distinct notes being played, and as a increases, we gain more time information allowing us to see the order which they are played.

3.2 *Comfortably Numb* Spectrograms

Since *Comfortably Numb* was a much longer song, it was broken into fifths to more efficiently analyze with the Gabor Transform. For the bass, a τ of 0.1 was used, with a bandwidth of 20, since this produced the cleanest graph. The resulting spectrogram is pictured below:

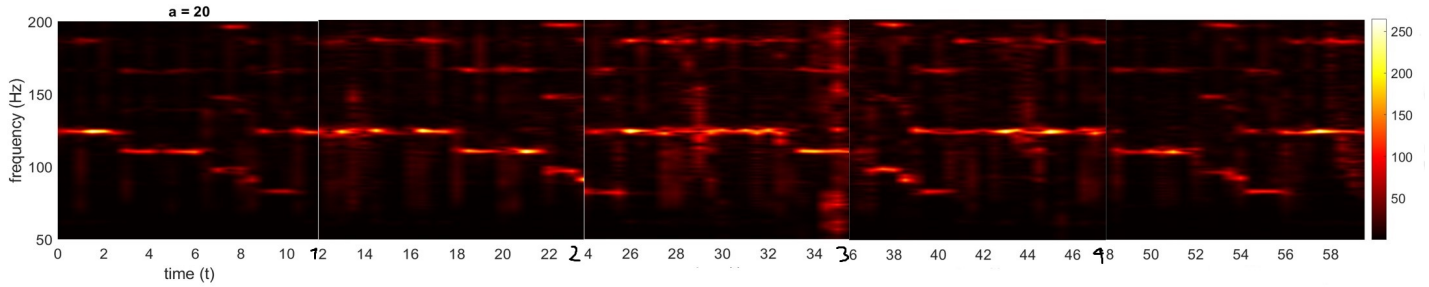

Figure 3: Bass spectrogram with $\tau = 0.1$ and $a = 20$, of bass line of *Comfortably Numb* by Pink Floyd

Filtering on frequencies from 50-200, which are standard for a bass guitar, we see several notes being played. To get a better idea of the strength of these signals, we can make a spectrogram of $(\log |s| + 1)$, which results in:

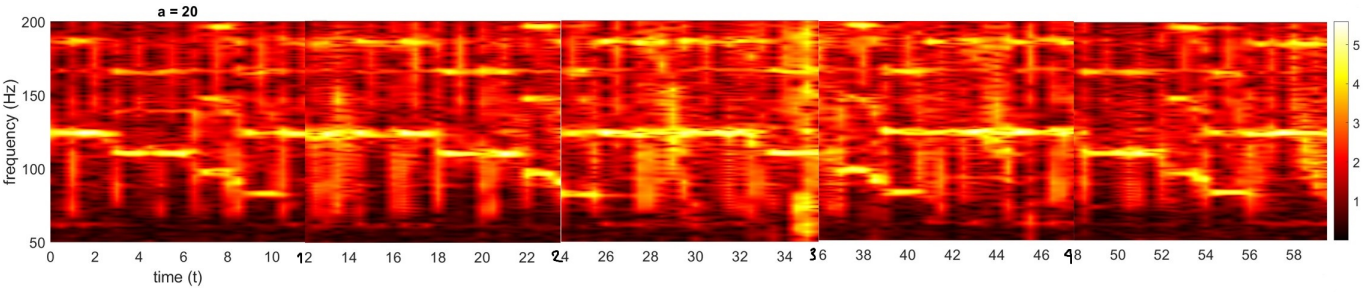

Figure 4: Bass spectrogram of $(\log |s| + 1)$ with $\tau = 0.1$ and $a = 20$, of *Comfortably Numb* by Pink Floyd

To analyze the guitar in *Comfortably Numb*, we look at frequencies from 200-900 Hertz with $a = 20$ and $\tau = 0.1$, which results in the following figure:

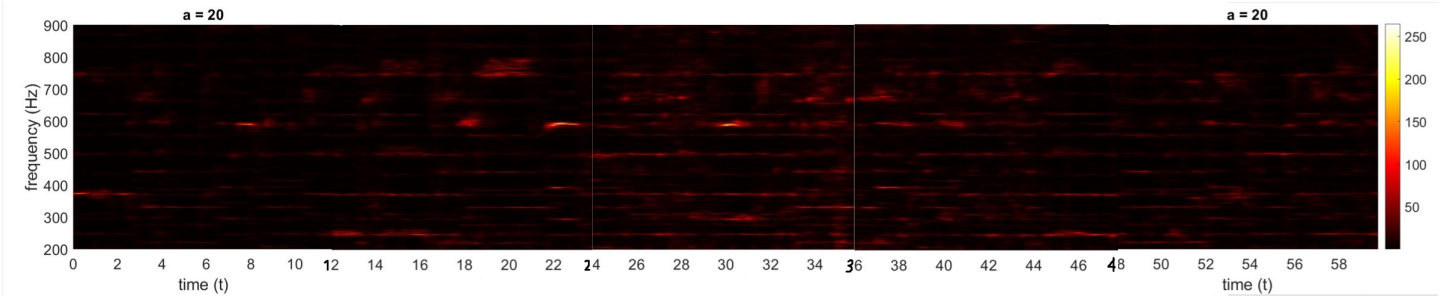

Figure 5: Guitar spectrogram with $\tau = 0.1$ and $a = 20$, of *Comfortaly Numb* by Pink Floyd

4 Computational Results

4.1 *Sweet Child O' Mine* Guitar Score

Using the spectrograms in Figure 2, we match the frequencies to notes, and can construct the score based on this information. For reference, the first measure is D4-D5-A4-G4-G5-A4-F5sharp-A4.

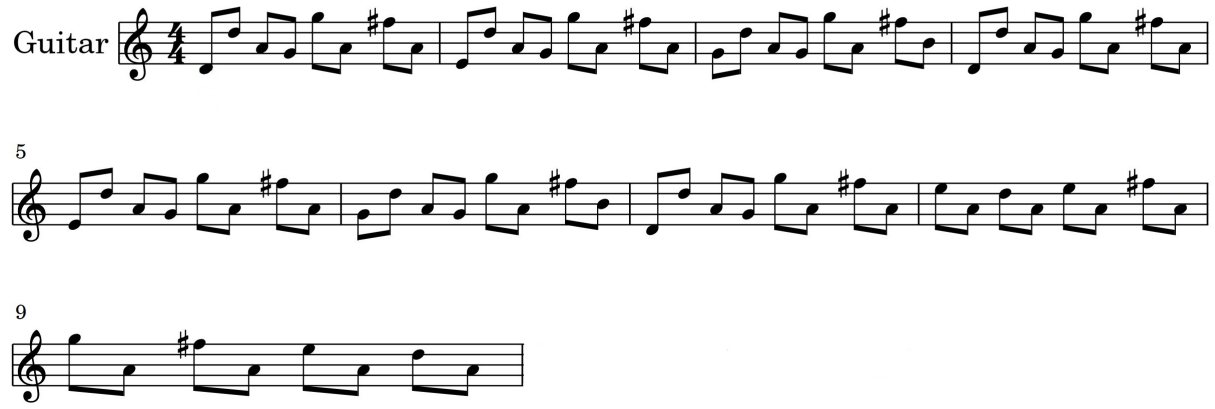

Figure 6: Score for guitar riff in *Sweet Child O' Mine* by Guns N' Roses.

4.2 *Comfortably Numb* Bass Score

Using the combined spectrogram in Figure, we can use the same technique of matching frequencies to notes in order to get the following score for bass:

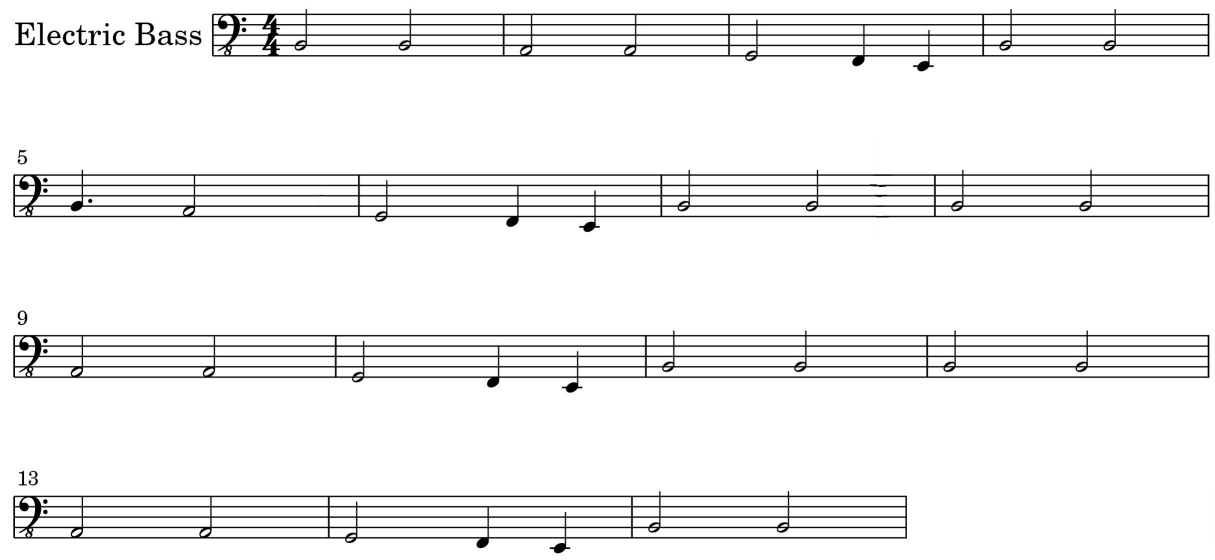

Figure 7: Score for bass during solo in *Comfortably Numb* by Pink Floyd

4.3 *Comfortably Numb* Guitar Score

Based on Figure 5, it is difficult to tell where guitar notes are present. Unlike the spectrograms for the guitar in *Sweet Child O' Mine* and the bass in *Comfortably Numb*, there are no frequencies which clearly stand out as notes being played. It appears that there are overtones present, which are not filtered out by the Shannon Filter. Thus using only the Gabor Transform, we are unable to focus solely on the guitar and reconstruct a score.

5 Summary and Conclusions

Through this project, we can see the power of the Fourier and Gabor Transforms, as they give us both frequency and time data, allowing the reconstruction of music scores. Additionally, we utilized the Shannon filter, which helps remove overtones and focus on the frequencies that we desire, such as the bass or guitar ranges.
For the *Sweet Child O' Mine* audio file, it was relatively simple to find the notes being played in the guitar solo via the spectrogram. However, for the *Comfortably Numb* guitar and bass solos, this job was much harder, given the other sounds (such as drums) present. This shows that the Gabor Transform has limitations, as it is insufficient in reconstructing the scores of more complicated pieces with confidence.

Appendix A MATLAB Functions

- `abs(X)`: Returns the absolute value of `X`.
- `audioplayer(Y, Fs)`: Plays the audio with a signal of `Y` and sampling rate of `Fs`.
- `audioread(file)`: Given the input `file`, will read data and return both the signal and its sampling rate.
- `exp(X)`: Returns the exponential function e^X for the inputted `X`.
- `fft(U)`: Performs the Fast Fourier Transform on an input `X`.
- `fftshift(U)`: Returns a rearranged Fourier Transform by shifting the zero frequency component to the center of the array, given an input `U`.
- `length(X)`: Returns the indices of the matrix which correspond to the indices defined by `ind` in a square matrix defined with size `sz`.
- `linspace(x,y,n)`: Returns n evenly spaced points between x and y.
- `pcolor(X, Y, C)`: Plots a pseudocolor plot with `X` and `Y` as the coordinates, and `C` to determine color intensity.
- `playblocking(X)`: Plays the audio associated with an audioplayer object `X`.
- `ylim(X)`: Restricts the output of a graph to the lower and upper bounds defined by `X`.

Appendix B MATLAB Code

```
clear
%%% Plot filter

figure(1)
x = [-1 -0.5 -0.5 0 0.5 0.5 1];
y = [0 0 1 1 1 0 0];
xL = xlim;
yL = ylim;
plot(x,y)
line([-1 1], [0 0],'Color','black','LineStyle','--');
ylim([-0.5 1.5])

saveas(gcf,'tophat.png')

clear
```

```
%% Loading Songs

%Playing Guns n Roses
figure(2)
[y, Fs] = audioread('GNR.m4a'); %y=data, Fs = rate
trgnr = length(y)/Fs; % record time in seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Sweet Child O'' Mine');
p8 = audioplayer(y,Fs);
playblocking(p8);
close;

%Playing Comfortably Numb

figure(2)
[y, Fs] = audioread('Floyd.m4a');
fifth = (length(y)-1)/3;
y=y(1:fifth); % For the n-th fifth, use (n-1)*fifth+1:n*fifth
trfloyd = length(y)/Fs; % record time in seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Comfortably Numb');
p8 = audioplayer(y,Fs);
playblocking(p8);

n = length(y);
L = trfloyd;
k = (1/L)*[0:(n/2-1) -n/2:-1]; % use hertz instead of radians
ks = fftshift(k);
a=20;
ts=linspace(0,L,n+1);
t=ts(1:n);
tau=0:1:L;

%% Gabor Transform

for j = 1:length(tau)
    g=exp(-a*(t-tau(j)).^2);
    yg = g.*y';
    ygt = fft(yg);
    ygtspec(j,:) = abs(fftshift(ygt));
end

%% Plot Spectrogram

figure(3)
pcolor(tau,ks,log(ygtspec+1)') % plots with logarithm
%pcolor(tau,ks,ygtspec')  plots without logarithm
shading interp
set(gca,'ylim',[200, 900],'Fontsize',16)
```

```
colormap(hot)
colorbar
xticks([0 2 4 6 8 10])
%xticklabels({'48','50','52','54','56','58'}) Change accordingly
xlabel('time (t)'), ylabel('frequency (Hz)')
title(['a = ', num2str(a)],'FontSize', 16)
```