AMATH 482 Homework 2

Sandy Yang

February 11, 2020

Abstract

This assignment deals with real world application of time-frequency analysis. We are given two recordings of the hit song: Sweet Child O' Mine by Guns N' Roses and Comfortably Numb by Pink Floyd, respectively. Through the use of the Gabor filtering, we will reproduce the music score for the guitar in the GNR clip, and the bass in the Floyd clip. The process and results of generating a spectrogram from audio signals using a windowed Fourier transform(Gabor transform) and how to combine this with the filter in frequency space is described in this report.

1 Introduction and Overview

Time frequency analysis comprises of techniques that simultaneously study a signal in time domain and frequency domain. This is useful for analyzing audio signals, since they include several frequencies (and their overtones) played at different moments in time. In this work, two audios are analyzed using time frequency analysis and their spectrograms are created. The audios are Sweet Child O' Mine by Guns N' Roses and Comfortably Numb by Pink Floyd. Spectrograms of the audio signals are created by implementing the Gabor transform, using different functions as the sliding window. With the help of frequency domain filter, we can further isolate the note and make the spectrograms more clear and obvious.

2 Theoretical Background

2.1 Gabor transforms

The Gabor transform, also known as the short term Fourier transform (STFT), is defined as

$$f_g(\tau, k) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-ikt}dt \tag{1}$$

The function $g(t-\tau)$ acts as a time filter for localizing the signal over a specific time window. The integration over the parameter τ slides the time-filtering window down the entire signal in order to pick out the frequency information at each instant of time. τ is the time at which the function is centered, and is the width of the function.

One thing to note is that there is a tradeoff which is in some sense inherent in using Gabor transforms. We are able to retain some temporal information when moving into the frequency domain, but the range of frequencies which we can resolve decreases. This is because using a window which does not encompass the entire temporal domain means that very long frequencies present in the signal will not be captured within the window and therefore will not appear in the spectrogram. Taking a window that is large enough to resolve these frequencies means sacrificing temporal precision and aiming for extreme temporal resolution by taking a very small window results in the loss of the ability to detect all but the shortest frequencies.

2.2 Spectrograms

A spectrogram is a visual representation of a spectrum of frequencies in a signal as they vary with time. In this work, spectrograms are constructed using frequency data gathered using the Gabor transform on three audio signals. Spectrograms are useful tools because they provide information about the frequencies present at a moment in time, as opposed to the Fourier transform, which gives only frequency information. The time-frequency analysis can can be used to produce speech recognition algorithms given the characteristic signatures in the time-frequency domains of sounds. Thus spectrograms are kind of like the fingerprint of sound.

3 Algorithm Implementation and Development

In this section we describe the numerical algorithms developed to explore various parameters associated with Gabor transforms and to determine the scores of given audios by ploting the Spectrograms. The procedures are given below along with a few comments and pertinent details.

3.1 Part 1

- **Import audio:** Load audio file into MATLAB. Calculate its length and the sampling rate(number of points).
- **Define grid vectors:** The scale we used for the frequencies of the musical notes is Hertz, so we should scale the k vector by $1/L$ and not $2\pi/L$. These grid vectors also need to be shifted using the fftshift function, since the FFT algorithm shifts the data such that $x \in [-L, 0] \rightarrow [0, L]$ and $x \in [0, L] \rightarrow [-L, 0]$.
- Define a time-slide vector for the sliding window (the time by which the window is translated).
- Start loop for translating window.
- Inside the loop, define the function for the translating window f_g.
- Take the absolute and fftshifted value of the resulting resulting vector and add it as a new row to **Sgt-spec**, a matrix from which the spectrogram will be constructed.
- Plot the spectrogram using the pcolor function resulting from the Sgt spec matrix with time on the x-axis and frequencies on the y-axis. We used the log(Sgt-spec)+1 to make the visual more obvious.

3.2 Part 2

- **P**lot out the Comfortably Numb's signal in both time and frequency domain. In the frequency domain, we can identify the range we want for filtering. In this case, the frequency range for bass is around 0 150Hz.

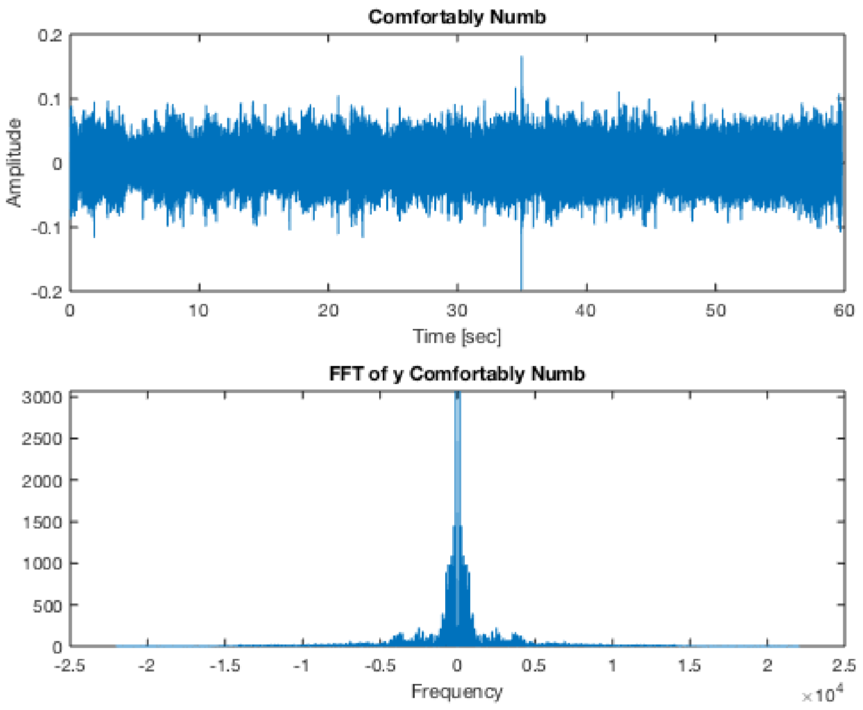


Figure 1: Audio signal and its Fourier transform

- Define the filter to see if the frequency is in between the require range, which is depended on the instrument of interest. We let the frequency of interest = 1 and set other to 0 so that when we time the filter and the fft(y) we can get rid of the unwanted frequency. For the bass we did the low-pass filter to keep the bass part. (Note: The filter in Figure 2 looks like 2 vertical line, but it's actually a really-high-hat-shape filter with maximum value of 1.)

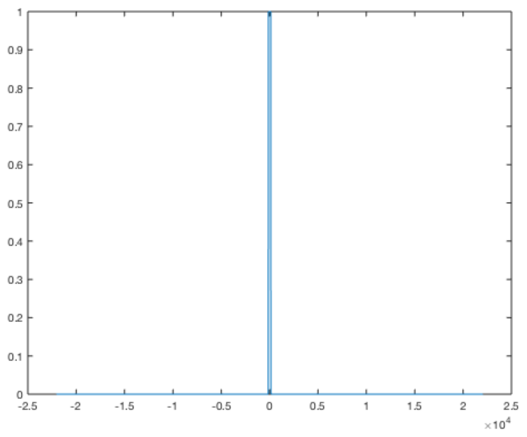


Figure 2: Filter for bass

- fftshift on the filter so the zero-frequency component can be in the center of spectrum.
- Apply the filter to the Fourier transform of y and do the inverse Fourier transform by doing `fy = fft(y).*filter`.

- Inverse Fourier transform of fy.
- Plot the result and use audioplayer/playblocking to see if we have successfully isolate the instrument.

3.3 Part 3

Using the technique in Part 2, we know we can filter out some certain frequency for guitar. Yet, due to the overtone, we can't inverse our filter to get rid of bass. The alternative strategies go like this:

- Since we already have the spectrogram for the bass part of the Comfortably Numb, we have a rough idea of which note is being played at which time, for example, the bass in playing B in the first 2 second (See Figure 4).
- Since we are analyzing only the first 10 second of the song, filter out the bass in the certain time period. For instance, in the first 2 second, rule out the frequency around 123Hz.
- There are about 5 notes in the first 10 second of the song. So we will repeat the process for 5 time. Or, create the list with the desires time frame and build up the if statement that filter out the unwanted based on the time in the list.
- With the new signal, form the spectrogram by step describe in Part 1.

4 Computational Results

4.1 Part 1

In this section we first outline the findings of our numerical experiments from testing various combinations of parameters in the algorithm in Section 3.

Using the methods outlined in Section 3.1 we produce the spectrograms of given audios shown in the Figure 1 below. Since the plot is colored using a logarithmic scale, the brighter spots actually correspond to components of much much larger magnitude in Figure 1. This is the result of the timbre of the instrument. With the chart provided in the problem set, we can essentially translate the frequency into notes (in figure 2 with note on the right-hand y-axis). Note: The translation is based on the chart provided in `https://pages.mtu.edu/~suits/notefreqs.html`

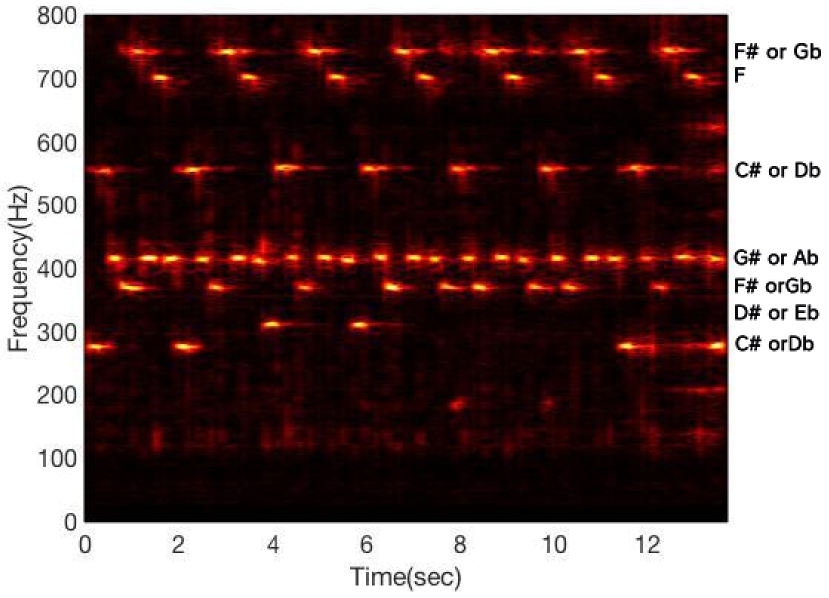


Figure 3: Spectrogram with score

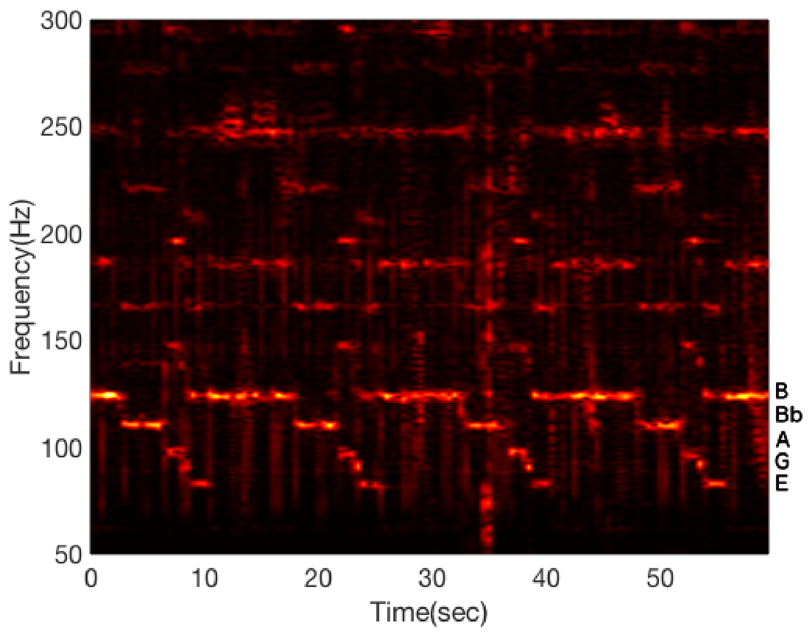
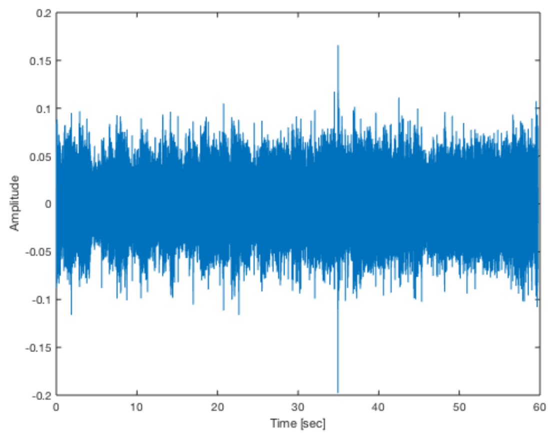


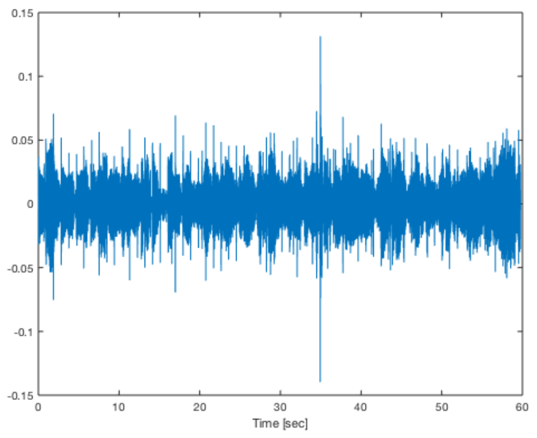
Figure 4: Spectrogram with score

4.2 Part 2

The result for part 2 is in the audio form. Yet, by doing the ifft to the filtered signal, we can still see the comparison before and after we applied the filter. In Figure 5, we can see some of the sound with the unwanted frequency has be dropped off, and only bass was in the signal.



(a) Original signal



(b) Signal after filtering

Figure 5: Signal before and after filtered

5 Summary and Conclusions

Both of the audio signals were analyzed using time frequency analysis. The Gabor transform was successfully used to create spectrograms of the two audio signals and enable us to find its score. With the use of filtering in frequency space, we are also being able to isolate the instrument we want. The condition may not be ideal since guitar and bass has some of the overlap but the outcome still enable us to isolate sound in certain range of frequency.

Appendix A MATLAB Functions

Here we outline the nonstandard MATLAB functions used to complete this assignment.

- `audioread('filename')`: Reads the sound file specified by filename and returns two outputs: a vector of sampled sound data and the sample rate for said data.
- `audioplayer(y,Fs)`: Creates an audioplayer object which can play the sound stored in the vector y with sampling rate Fs. It is typically paired with the playblocking function which actually plays the sound.
- `playblocking(AP)`: Takes an audioplayer object, AP as an input and plays the associated sound clip.
- `fft(X)`: : Returns the 1dimensional discrete Fourier transform of given input.
- `fftshift(v)`: Shifts the vector/matrix output of fft so that the 0 frequency lies at the center of the vector/matrix. To plot the frequency components given by fft one should first apply the fftshift command as fft returns a shifted version of the frequency components.
- `pcolor(X,Y,vals)`: Plot a pseudocolor plot. The elements of C are linearly mapped to an index into the current colormap. The mapping from C to the current colormap is defined by colormap.

Appendix B MATLAB Code

Add your MATLAB code here. This section will not be included in your page limit of six pages.

```
clear all; close all; clc

%% P1 for GNR -- Define space and time domain

[y, Fs] = audioread('GNR.m4a');
% p8 = audioplayer(y,Fs); playblocking(p8);

v = y.';
n = length(v);
L = n/Fs; % record time in seconds
a = 500;
k=(1/(L))*[0:n/2-1 -n/2:-1];
ks=fftshift(k);

tfinal =n/Fs;
t = (1:n)/Fs;
tslide = 0:.1:tfinal;

%%
Sgt_spec = zeros(length(tslide), length(t));
%Max=  zeros(length(tslide), length(t));  % to

for n = 1:length(tslide)
     g = exp(-a*(t-tslide(n)).^2);
     Sg = g.*v;
     Sgt = fft(Sg);
     Sgt_spec(n,:) =  fftshift(abs(Sgt));

     %[fmax, ind] = max(Sgt);
     %Max(n,:)= k(ind);
end

%%
%figure(1)
%plot(tslide, abs(Max),'o')
%xlabel('Time(sec)'); ylabel('Frequency(Hz)');
%set(gca,'Ylim',[0 800],'Fontsize',16)

%%

figure(2)
pcolor(tslide,ks,log(Sgt_spec.'+1))
shading interp
colormap('hot')
set(gca,'Ylim',[0 800],'Fontsize',16)
xlabel('Time(sec)'); ylabel('Frequency(Hz)');
```

Listing 1: Part 1 for GNR

```
clear all; close all; clc

%% P1 for Floyd -- Define space and time domain

[y, Fs] = audioread('Floyd.m4a');
% p8 = audioplayer(y,Fs); playblocking(p8);

y = y(1:end-1);

v = y.';
n = length(v);
L = n/Fs;% record time in seconds
a = 20;
k=(1/(L))*[0:n/2-1 -n/2:-1];
ks=fftshift(k);
index = find(ks > 0 & ks < 800);

tfinal1 =n/Fs;
t = (1:n)/Fs;
tslide = 0:0.1:tfinal1;

%% Gabor

Sgt_spec = zeros(length(tslide), length(index));

for n = 1:length(tslide)
    %Gaussian window
    g = exp(-a*(t-tslide(n)).^2);

    Sg = g.*v;
    Sgt = fft(Sg);
    fftshtsgt = abs(fftshift(Sgt));
    Sgt_spec(n,:) = fftshtsgt(index);
end
Sgt_spec = Sgt_spec(:, index); % trim the matrix so ploting can be faster

%%
figure()

pcolor(tslide,ks(index),log(Sgt_spec.'+1))
shading interp
colormap('hot')
set(gca,'Ylim',[20 200],'Fontsize',16)
xlabel('Time(sec)'); ylabel('Frequency(Hz)');
```

Listing 2: Part 1 for Floyd

```
%% Code for part 2
clear all; close all; clc

%%
[y, Fs] = audioread('Floyd.m4a');
% p8 = audioplayer(y,Fs); playblocking(p8);

y = y(1:end-1);
yt = fft(y);
n = length(y);
L = n/Fs; % record time in seconds

k=(1/(L))*[0:n/2-1 -n/2:-1];
ks=fftshift(k);


%%

filter=zeros(size(yt));

for i = 1:length(ks)
     if(abs(ks(i))> 150)
         filter(i) =0;
     else
         filter(i) = 1;
     end
end

fftg = fftshift(filter);
ys = yt.*fftg;
s = (ifft(ys));

%% Plot

figure(1)
subplot(2,1,1)
plot((1:n)/Fs, y);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Comfortably Numb')

subplot(2,1,2)
plot(ks, abs(fftshift(yt)));
xlabel('Frequency');
title('FFT of y Comfortably Numb')

figure(2)
plot((1:n)/Fs,(s));
xlabel('Time [sec]');

%%
%p8 = audioplayer((s),Fs); playblocking(p8)
```

Listing 3: Part 2 code