AMATH482 Homework 2

Vandy Zhang
February 8, 2021

Abstract

This assignment is to use the Gabor Transform to filter the two music clips provided. Then we have to use spectrogram to visualize their frequencies and isolate the frequencies of certain musical instruments. By doing that and matching music notes to the frequencies showed on the spectrograms, we can have the music scores of certain music instruments in the two songs reproduced.

1 Introduction and Overview

For this homework, I was asked to filter two music clips, isolate the sound of some certain musical instruments such as guitar and bass, and reproduce their music scores. The two clips I was offered were GNR.m4a and Floyd.m4a, which play clips of the songs *Sweet Child O' Mine* by Guns N' Roses and *Comfortably Numb* by Pink Floyd, respectively. Those are two pieces of music ranked high in the music professional rankings. To filter the two clips, I applied the Gabor Transform, which was to use Gaussian filter functions to denoise the signals, and the Fourier Transform to convert the signal into the frequency space. While plotting the figure to visualize the frequencies, I created my own spectrogram using color hot maps. To isolate certain musical instruments, I modified my spectrogram by changing the range of its y-axis so that a specified scope of the frequencies can be isolated then. To reproduce the music scores, I searched for the frequencies of music notes, matched them to my graph, and wrote scores according to the beats showed on my spectrogram.

2 Theoretical Background

To filter the data of music clips, I applied the Gabor Transform, which is theoretically given by [3]:

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

τ represents the center I am going to filter around. $\mathbf{g(t)}$ represents the filter function. Therefore, $f(t)g(t-\tau)$ represents the filtered function with the filter centered at τ. There are many choices for g and some commonly used assumptions are listed below:

1. The function g should be real and symmetric.
2. The L2-norm of g should be set to unity. (The L2-norm is commonly understood to represent the total energy of a function.

The filter function g(t) I used in this project was the Gaussian filter function, which is given by:

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

After applying the filter to denoise the data, I converted the signals to frequency space. Here I used Fourier Transform [2]:

Suppose we are given a function $f(x)$ with x ∈ R. We define the Fourier Transform of $f(x)$, written as

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

In this case, we have our data over frequency domain. Theoretically, we are making projections of the frequencies over time to one time point.

The Fast Fourier Transform is a function that does the Discrete Fourier Transform faster. It helps split the DFT into two DFT sequences and use a divide and conquer algorithm [2]. The Discrete Fourier Transform has a complexity of $\mathcal{O}(\mathrm{N}^2)$, while the Fast Fourier Transform only has a complexity of $\mathcal{O}(\mathrm{N}\log(\mathrm{N}))$.

To plot the data, I created my own spectrogram, which is the best way to show the Fourier transforms over a time span. That is, to stack all Fourier transforms next to each other with the horizontal direction being the value of τ, the window center, and the vertical direction being the frequency [4].

3 Algorithm Implementation and Development

To import the data from the .m4a files, I implemented the method audioread in MATLAB, which is to read audio files and get the data of the signal and the sample rate. By dividing the length of the signal by the sample rate, I got the total time of the clip. Based on the total time, I had my time domain and frequency domain set up. When setting k up, I

rescaled the frequencies by 1/total time instead of 2π/total time since though FFT assumes 2π periodic signals, the scale used for the frequencies of the musical notes is Hertz.

After choosing a proper time step for the values of the two clips, I could loop to filter my data within my window scales. I used the Gabor Gaussian filter functions, transformed the data over frequency domain after being filtered by using the built-in Fast Fourier Transform **fft** method in MATLAB, and then **fftshift**ed my data so that it could be ordered correctly since **fft** made it lose time information. In the Gaussian filter function, the factor a, which is a determining factor of the window size that is going to be filtered in, could be properly selected by visualizing the data in spectrogram.

I created my own spectrogram instead of using the built-in MATLAB function **spectrogram** because it could probably generate some bias such as only plotting positive frequencies, which were already specified by the programmers of MATLAB. To create my own, I used **pcolor** to take in the x, y coordinates and the final signal we got from the previous steps and added a hot **colormap** as well as a **colorbar**. If the graph I got was too blur, then the value of a was too large. If the graph contains mostly thin lines, then I had to increase the value of a. By adjusting the a value according to the spectrogram I produced, I filtered my data properly.

Then to reproduce music scores, I had to find the frequencies of music notes and match the values to my spectrogram. After getting the music notes I needed for the music scores, I could write the scores according to the spectrogram I created.

To filter out the overtones and isolate the bass in the second song Comfortably Numb, I used **ylim**, which could act as a square filter so that all the frequencies that are not in the scope of y limits could be cut. That is to say, those frequencies could be set to zero. Similarly, to get the guitar solo in the second song Comfortably Numb, I applied **ylim** as well to filter out the overtones and the bass at the same time.

4 Computational Results

4.1 GNR Clip

Music Score: $\mathrm{C}^{\#}\ C^{\#^{+}}\ G^{\#}\ F^{\#} \quad F^{\#^{+}}\ G^{\#}\ F\ G^{\#} \quad \mathrm{C}^{\#}\ C^{\#^{+}}\ G^{\#}\ F^{\#} \quad F^{\#^{+}}\ G^{\#}\ F\ G^{\#}$

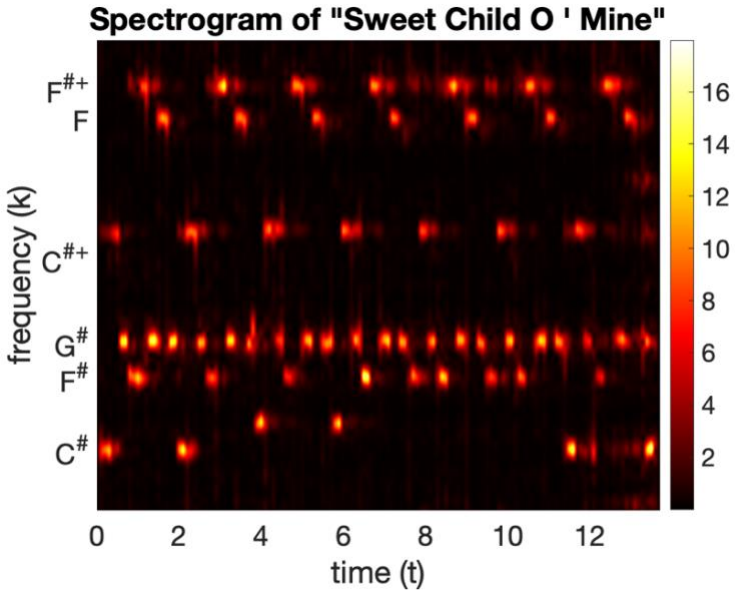

Figure 1

The spectrogram of the song "Sweet Child O' Mine" created by applying the Gabor Gaussian filter function with a = 1200, timestep for τ = 0.1, and limits for the y-axis is [200, 800]. Overtones have been filtered out.

4.2 Floyd Clip

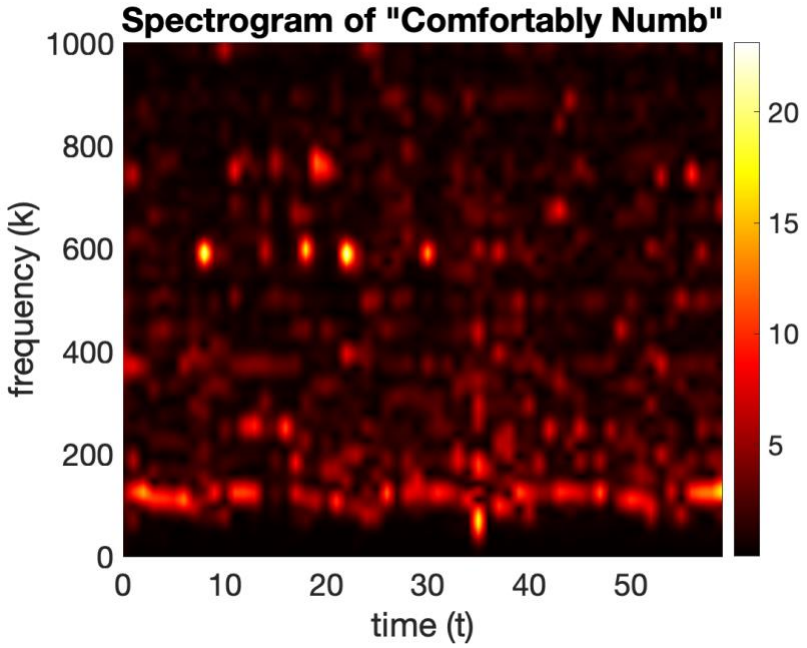

Figure 2

The spectrogram of the song "Comfortably Numb" created by applying the Gabor Gaussian filter function with a = 6000, timestep for τ = 1, and limits for the y-axis is [0,1000]. Overtones have been filtered out.

Bass Music Score: B A G E B B A G E B B A G E B B A G E B

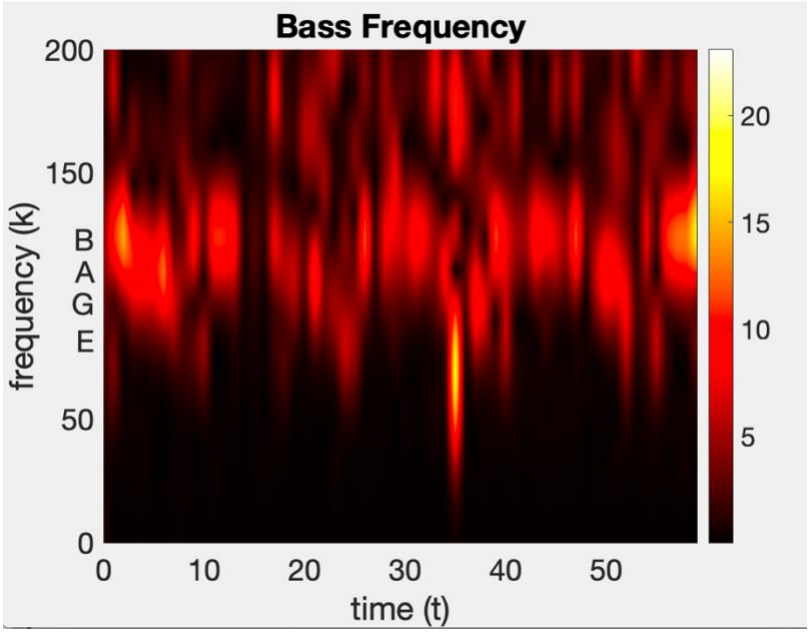

Figure 3

The spectrogram of the bass part of the song "Comfortably Numb" created by applying the Gabor Gaussian filter function with a = 6000, timestep for τ = 1, and limits for the y-axis is [0,200]. Overtones and other instruments have been filtered out.

Guitar Music Score: $F^{\#}\ A\ G\ D\ E$ $F^{\#}\ A\ G\ D\ E$ $F^{\#}\ A\ G\ D\ E$ $F^{\#}\ A\ G\ D\ E$

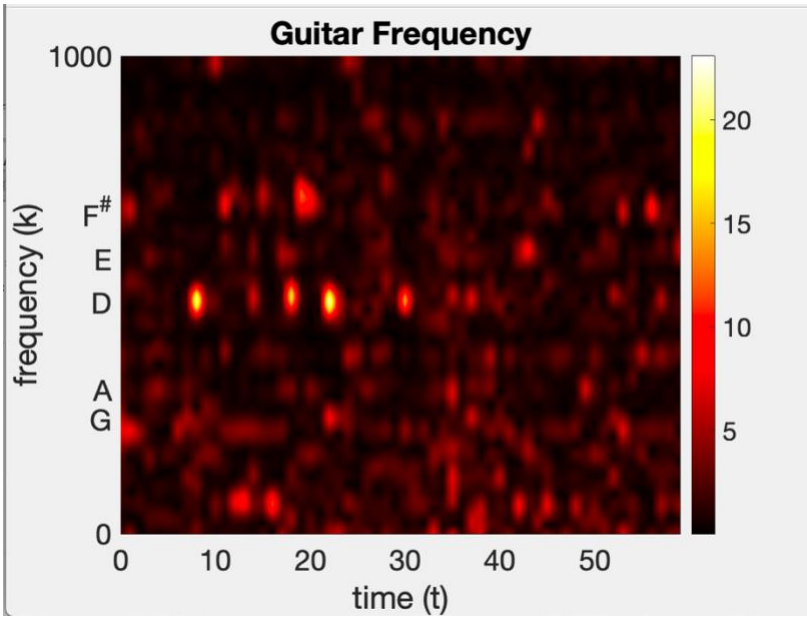

Figure 4

The spectrogram of the guitar solo of the song "Comfortably Numb" created by applying the Gabor Gaussian filter function with a = 6000, timestep for τ = 1, and limits for the y-axis is [200, 1000]. Overtones and bass have been filtered out.

5 Summary and Conclusions

Filtering the signal of the music clips, I used the Fast Fourier Transform and the Gabor Transform, especially the Gabor Gaussian filter functions. To find a good value for the variables in the filter function, I visualized the signal data by creating a spectrogram. Adjusting the values of the variables and the graphs, I had a good spectrogram displayed. Then I limited the range of y-axis to filter out the overtones and certain frequencies of some musical instruments so that I could get the range I want. To reproduce the music scores, I then searched the frequencies of music notes and matched them to my spectrogram so that I could recognize music values from my plot.

References

[1] Jose Nathan Kutz. Data-driven modeling & scientific computation: methods for complex systems & bigdata. Oxford University Press, 2013.

[2] University of Washington. AMATH482 Lecture 1 Notes. URL: https://canvas.uw.edu/courses/1433289/files/72729571?wrap=1

[3] University of Washington. AMATH482 Lecture 4 Notes. URL: https://canvas.uw.edu/courses/1433289/files/71735353?wrap=1

[4] University of Washington. AMATH482 Lecture 7 Notes. URL: https://canvas.uw.edu/courses/1433289/files/71735382?wrap=1

Appendix A. MATLAB Functions

audioread Read audio files and return the sampled data and the sample rate of the data

linspace (x1, x2, n) Generate linearly spaced vector between X1 and X2 with a length of n.

fftn (X) Apply the 2D Fast Fourier Transform to X

fftshift (X) Shift zero-frequency component to the center of the spectrum

pcolor(X, Y, C) Create a pseudocolor surface with specified x, y coordinates and plot the data C

colormap(map) Set the colormap for the current figure to the colormap specified by map.

colorbar Add a bar showing the color scale

set Set properties for the graph

yticks Set locations along the y-axis where the tick marks appear

yticklabel Set tick labels

Appendix B. MATLAB Code

```
% GNR Part

clear; clc; close all;
figure(1)
[y, Fs] = audioread('GNR.m4a');
tr_gnr = 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);

y = y.';

a = 1200;
tau = 0:0.1:tr_gnr;

n = length(y);
t2 = linspace(0,tr_gnr,n+1); t = t2(1:n);
k = (1/tr_gnr)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

for j = 1:length(tau)
   g = exp(-a*(t - tau(j)).^2); % Window function
   yg = g.*y;
   ygt = fft(yg);
   ygt_spec(:,j) = fftshift(abs(ygt)); % We don't want to scale it
end

pcolor(tau,ks,ygt_spec)
shading interp
set(gca,'ylim',[200 800],'Fontsize',20)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title('Spectrogram of "Sweet Child O '' Mine"');
yticks([277 370 415 523 698 740]);
yticklabels({'C^#','F^#','G^#','{C^#}^+','F','{F^#}^+'});
print('HW2GNR.png', '-dpng');


% Floyd Part

clear; clc; close all;
figure(1)
[y, Fs] = audioread('Floyd.m4a');
tr_gnr = 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);

y = y.';
```

```
a = 6000;
tau = 0:1:tr_gnr;

n = length(y);
t2 = linspace(0,tr_gnr,n+1); t = t2(1:n);
k = (1/tr_gnr)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

for j = 1:length(tau)
   g = exp(-a*(t - tau(j)).^2); % Window function
   yg = g.*y;
   ygt = fft(yg);
   ygt_spec(:,j) = fftshift(abs(ygt)); % We don't want to scale it
end

pcolor(tau,ks,ygt_spec(1:end-1,:))
shading interp
set(gca,'ylim',[0 1000],'Fontsize',20)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title('Spectrogram of "Comfortably Numb"');
print('HW2Floyd.png', '-dpng');

% problem 2

figure(2)
pcolor(tau,ks,ygt_spec(1:end-1,:))
shading interp
set(gca,'ylim',[0 200],'Fontsize',20)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title('Bass Frequency');
yticks([0 50 82 97 110 123 150 200]);
yticklabels({0, 50, 'E', 'G', 'A', 'B', 150, 200});
print('HW2FloydBass.png', '-dpng');

% problem 3

figure(3)
pcolor(tau,ks,ygt_spec(1:end-1,:))
shading interp
set(gca,'ylim',[200 1000],'Fontsize',20)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title('Guitar Frequency');
yticks([200 391 440 587 659 740 1000]);
yticklabels({0,'G','A','D','E','F^#',1000});
print('HW2FloydGuitar.png', '-dpng');
```