AMATH 482 HW2 - Gabor Transform

Winnie Shao

2/10/2021

Abstract

In this report, we discuss a method called Gábor Transform to conduct time-frequency analysis. For this problem, we are given two song: Sweet Child O' Mine by Guns N' Roses and Comfortably Numb by Pink Floyd. We are going to reproduce the score of particular instrument of these two songs.

Section I: Introduction and Overview

In the previous report, we stated that Fourier Transform is one of the most powerful method to solve problems related to time-frequency analysis and is widely used in signal processing. However, this method is also acknowledged that the transform translates the signal from time domain to frequency domain and loses the information in time domain. In other words, we fail to obtain information of the changes of the signal as time goes by. Therefore, we need to introduce an approach as a modification of the Fourier transform. In this problem, we are introducing Gábor Transform to solve problems related with both time and frequency analysis.

In this report, we will process two pieces of songs. For the first song, Sweet Child O' Mine by Guns N' Roses, we will need to reproduce the guitar score notes. For the other song, Comfortably Numb by Pink Floyd, we need to isolate and reproduce the bass notes and also the guitar notes.

Section II: Theoretical Background

2.1 Drawback of Fourier Transform

We know that for x $\in [-\infty, \infty]$ the Fourier transform is defined as:

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

The Fourier transform is used to decompose a time defined function to a frequency-amplitude image. In MatLab, we use the fast Fourier transform to perform the Fourier transform. Since Fourier transform converts time domain to frequency domain, it will lose all the information in the time domain. If we use FFT on a song, we cannot tell at which point it is producing high, middle, or low notes. Therefore, we need another method to analyze time and frequency domain at the same time.

2.2 Gábor Transform

The Gábor transform is also known as the short-time Fourier transform. Gábor transform decomposes the time domain into separate time frames so that it can perform time-frequency analysis with various signals changing as the time goes by. Each time frame is called window. The Fourier transform is applied to analyze freqency during that window with remaining frequency zeroed out.

The Gábor transform is defined as the following:

$$G[f](t, \omega) = \tilde{f}_g(t, \omega) = \int_{-\infty}^{\infty} f(\tau) \bar{g}(\tau - t) e^{-i\omega\tau} d\tau = (f, \bar{g}_{t,\omega}) \tag{2}$$

We notice that in the integral e^{-ikx} for the Fourier transform is replaced by the kernel $\bar{g}(\tau - t)e^{-i\omega\tau}$ for Gábor transform. The function $\bar{g}(\tau - t)$ localizes a piece of frequency over a specific time window by filtering out the frequency outside of the window. The window is centered at τ with width a. Therefore, as τ increases, the Gábor filter slides down the entire song and extracts frequency information for each time frame.

However, Gábor transform can only be realized by discretizing the time and frequency domain in practical world. Therefore, we need to consider the discrete version. Then Gábor transform becomes:

$$\tilde{f}(m,n) = \int_{-\infty}^{\infty} f(t)e^{i2\pi m\omega_0 t}g(t-nt_0)dt \tag{3}$$

Where $v = m\omega_0$ and $\tau = nt_0$, m and n are integers and ω_0,$t_0 > 0$ are constants.

2.3 Gábor Filters / Windows

The Gaussian filter is defined as:

$$F(k) = exp(-\tau(k-k_0)^2) \tag{4}$$

It is used to extract the signal pattern within a window in time domain so that we can analyze frequency content at a specific moment. In this problem, besides using Gaussian filter, I also implement some simple manual filter which I will explain in the implementation part.

Section III: Algorithm Implementation & Development

To reproduce music notes, we need to load in two audio files to get the vectors representing two pieces of music (Sweet Child O' Mine and Comfortably Numb) and sample frequency for two songs. Dividing the length of the vector v by the sampling frequency, we gain the length of these two songs which is 14s and 60s. We set the interval of tslide to 0.3 and then apply the Gábor transform on each of the signals. In order to filter out overtones and get a clean score, we look for the max value of the transformed data at each iteration of window, and save them to the notes arrays. Finally, we produce the spectrograms of results and the image of music notes.

Section IV: Computational Results

After loading in the first song: Sweet Child O' Mine, we can clearly recognize that there's only one instrument here which is guitar. Then we apply Gábor transformation and Gaussian filter, we get the spectrogram Figure 1

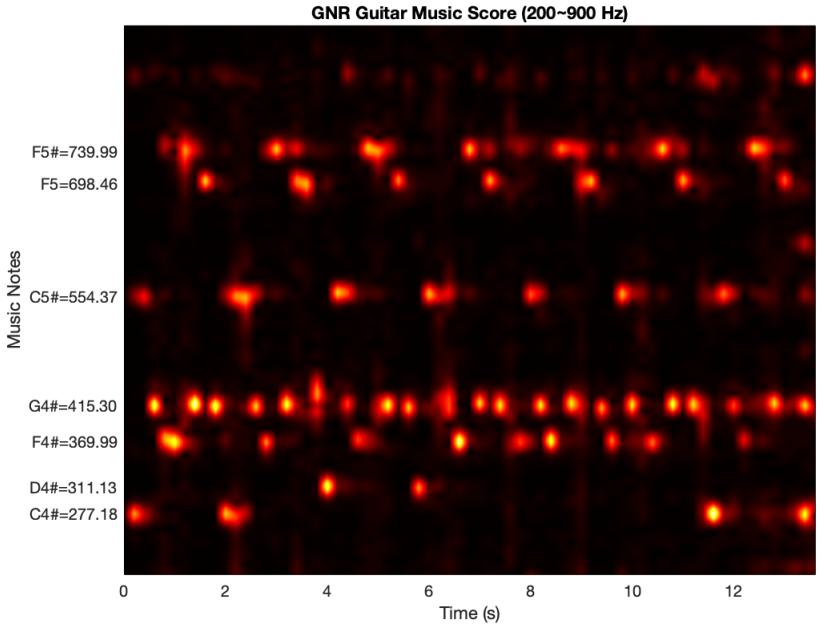

Figure 1: The Guitar Notes for Sweet Child O' Mine

For the second song, first we try to extract bass notes. In this problem, since running a spectrogram for 60s almost shut down my computer every time, I will provide simple notes graph to illustrate. From Figure 2 we can see that the base notes are B2, A2, G2, Gb2, E2 and repeat.

Figure 3 is the guitar notes for the second song and is a little bit confusing. I will discuss more about Figure 3 in the next part.

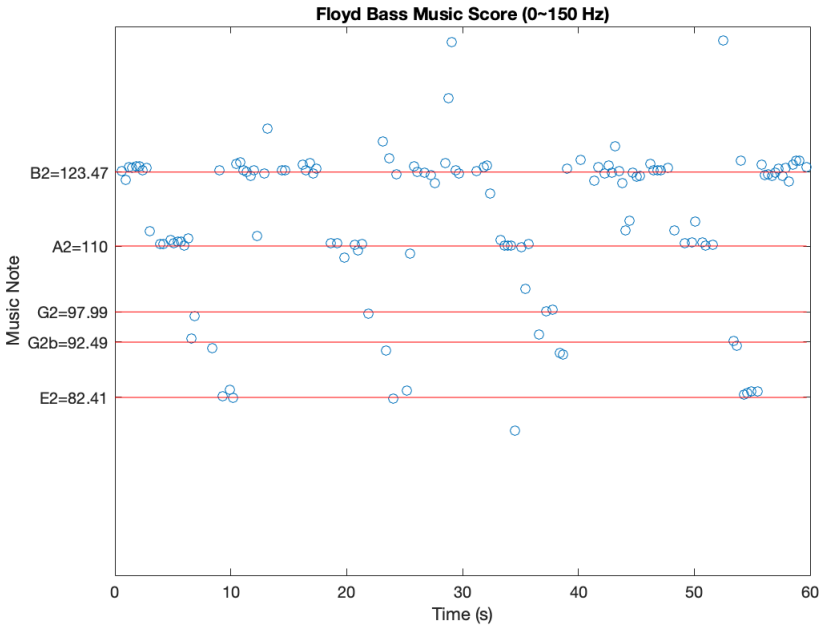

Figure 2: The Bass Notes for Comfortably Numb

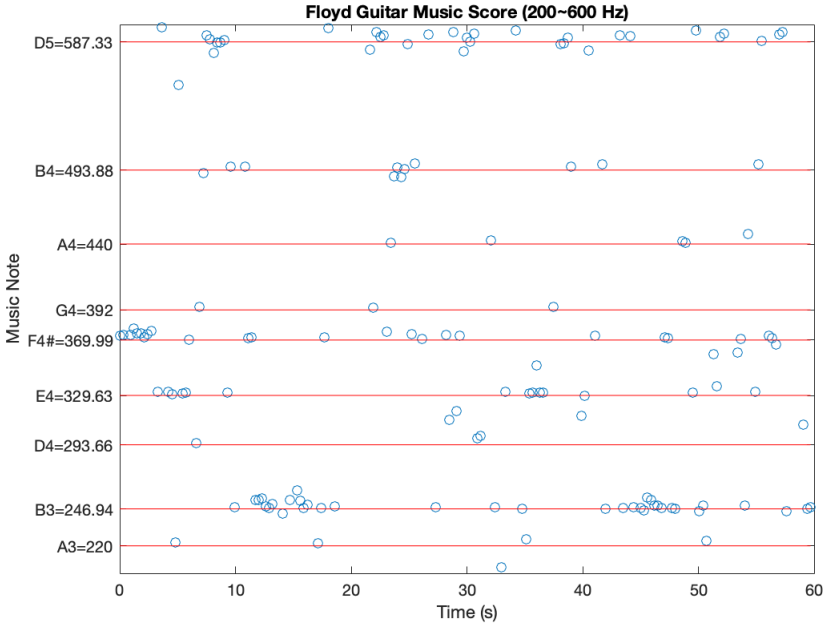

Figure 3: The Guitar Notes for Comfortably Numb

Section V: Summary and Conclusions

By implementing the Gábor transform and using the window, we could obtain the time domain information and do more time-frequency analysis on songs. We can see from Figure

1 and 2 that this technique is promising. However, there's also some disadvantages. When we tried to obtain the guitar notes in the second song, we found that the higher frequency domain is really confusing. This is because that there are several instrument including guitar and different drums in the higher frequency domain. Therefore, extracting the guitar notes did not go successfully. Also, one thing i notice is that the notes we got from the frequency is only the notes we hear by using our ears. I looked through the original music sheet and watched several teaching videos on YouTube. Most of the sound we hear is different from the actual music score written on the paper. For example, when we hear F4, the music score is actually E4. However, due to special technique such as whole step bend and half step bend, we can only hear F4. In general, I think Gábor transform is a promising tool for music recognition.

Appendix A: MATLAB functions used and brief implementation explanation

Function	Syntax	Brief Explanation
fftshift	fftshift(X)	Rearranges a Fourier transform X by shifting the zero frequency component to the center.
fft	fft(x)	apply FFT in 2-D
yticks	yticks(”,”, ”, ...)	set the y-axis tick values
max	[M, I] = max()	returns the max value in the array and also returns the index of that value
audioread	audioread(filename)	read in an audio file and turn it into frequency
lowpass	lowpass(x, fpass, fs)	filter out the frequency above fpass in the array x
highpass	highpass(x, fpass, fs)	filter out the frequency below fpass in the array x
pcolor	pcolor(X, Y, C)	sepcifies x and y coordinates for the vertices. The size of C must match the size of the xy coordinate grid. If x and y define an m by n grid, then C must be an m by n matrix

Appendix B: MATLAB codes

```
close all; clear all; clc;

%% Part 1 (a)
[y, Fs] = audioread('GNR.m4a');
tr_gnr = length(y)/Fs; % record time in seconds
v = y';
```

```
n = length(v);
t2 = linspace(0,tr_gnr,n+1);
t_gnr = t2(1:n);
k_gnr = (2*pi/tr_gnr)*[0:n/2-1 -n/2:-1];
ks_gnr = fftshift(k_gnr);
tslide_gnr = 0:0.2:tr_gnr;
spec_gnr = zeros(length(tslide_gnr), n);
notes_gnr = zeros(1, length(tslide_gnr));

for j = 1:length(tslide_gnr)
    g = exp(-1500*(t_gnr - tslide_gnr(j)).^2);  % Use Gaussian Filter
    vgp = g .* v;
    vgpt = fft(vgp);
    [M, I] = max(vgpt);  % Get the index with strongest frequency (music s

    notes_gnr(1,j) = abs(k_gnr(I))/(2*pi);
    spec_gnr(j,:) = fftshift(abs(vgpt));
end

figure(1);
pcolor(tslide_gnr, (ks_gnr/(2*pi)), spec_gnr.'),
shading interp
colormap(hot);
% plot(tslide_gnr, notes_gnr, 'o');
title ("GNR Guitar Music Score (200~900 Hz)");
xlabel('Time (s)'), ylabel('Music Notes');
ylim ([200 900]);
yticks([277.18, 311.13, 369.99, 415.30, 554.37, 698.46, 739.99]);
yticklabels({'C4#=277.18','D4#=311.13','F4#=369.99','G4#=415.30',...
    'C5#=554.37','F5=698.46','F5#=739.99'});
```

```
%% Part 2
[y, Fs] = audioread('Floyd.m4a');

v = lowpass(y', 150, Fs);
tr_f = length(v)/Fs;
n = length(v);

t2 = linspace(0,tr_f,n+1);
t_f = t2(1:n);
k_f = (2*pi/tr_f)*[0:n/2-1 -n/2:-1];
ks_f = fftshift(k_f);
tslide_f = 0:0.3:tr_f;
spec_f = zeros(length(tslide_f), n);
notes_f = zeros(1, length(tslide_f));

for j = 1:length(tslide_f)
    g = exp(-1500*(t_f - tslide_f(j)).^2);  % Use Gaussian Filter
    vgp = g .* v;
    vgpt = fft(vgp);
    [M, I] = max(vgpt);  % Get the index with strongest frequency (music s
    notes_f(1,j) = abs(k_f(I))/(2*pi);
    spec_f(j,:) = fftshift(abs(vgpt));
end

figure(2);
% pcolor(tslide_f, (ks_f/(2*pi)), spec_f.'),
% shading interp
% colormap(hot);
plot(tslide_f, notes_f,'o');
```

```
title ("Floyd Bass Music Score (0~150 Hz)");
xlabel('Time (s)'), ylabel('Music Note');
ylim([50 150])
yticks([82.41,92.49,97.99,110,123.47])
yticklabels({'E2=82.41','G2b=92.49','G2=97.99','A2=110','B2=123.47'})
hold on
one = ones(1, length(tslide_f));
plot(tslide_f, 82.41*one, 'r')
plot(tslide_f, 92.49*one, 'r')
plot(tslide_f, 97.99*one, 'r')
plot(tslide_f, 110*one, 'r')
plot(tslide_f, 123.47*one, 'r')
%% Part 3
[y, Fs] = audioread('Floyd.m4a');
v = highpass(y', 200, Fs);
tr_f = length(v)/Fs
n = length(v);
t2 = linspace(0,tr_f,n+1);
t_f = t2(1:n);
k_f = (2*pi/tr_f)*[0:n/2-1 -n/2:-1];
ks_f = fftshift(k_f);
tslide_f = 0:0.3:tr_f;
spec_f = zeros(length(tslide_f), n);
notes_f = zeros(1, length(tslide_f));
for j = 1:length(tslide_f)
    g = exp(-1500*(t_f - tslide_f(j)).^2);  % Use Gaussian Filter
    vgp = g .* v;
    vgpt = fft(vgp);
    [M, I] = max(vgpt);  % Get the index with strongest frequency (music s
    notes_f(1,j) = abs(k_f(I))/(2*pi);
```

```
    spec_f(j,:) = fftshift(abs(vgpt));
end
plot(tslide_f, notes_f,'o');
title ("Floyd Guitar Music Score (200~600 Hz)");
xlabel('Time (s)'), ylabel('Music Note');
ylim([200, 600])
yticks([220, 246.94, 293.66, 329.63, 369.99, 392, 440, 493.88, 587.33])
yticklabels({'A3=220','B3=246.94','D4=293.66','E4=329.63','F4#=369.99',...
    'G4=392','A4=440','B4=493.88','D5=587.33'})
hold on
one = ones(1, length(tslide_f));
plot(tslide_f, 220*one, 'r')
plot(tslide_f, 246.94*one, 'r')
plot(tslide_f, 293.63*one, 'r')
plot(tslide_f, 329.63*one, 'r')
plot(tslide_f, 369.99*one, 'r')
plot(tslide_f, 392*one, 'r')
plot(tslide_f, 440*one, 'r')
plot(tslide_f, 493.88*one, 'r')
plot(tslide_f, 587.33*one, 'r')
```