Amath482 Homework2

Kexin Jiao

Abstract:

The homework mainly focuses on the practice of the Fast Fourier Transform, the Gabor Transform, and the Spectrogram to solve the actual problem: analyzing music clips, filtering certain musical instruments, and reproducing music scores.

Introduction and overview:

The homework is about analyzing a portion of two rock and roll songs--Sweet Child O' Mine by Guns N' Roses and Comfortably Numb by Pink Floyd. We are asked to perform the following:
1. Through the use of Gabor filtering, reproduce the music score for the guitar in the GNR clip and the bass in the Floyd clip.
2. Use a filter in frequency space to try to isolate the bass in Comfortably Numb.
3. See how much of the guitar solo you can put together in Comfortably Numb. It may help to look at smaller portions of the clip to guide your reconstruction of the music score.

Theoretical background:

Fourier Transform

Suppose we are given a function $f(x)$ with $x \in R$. The Fourier Transform of f(x),written $f\_hat(k)$, is defined by the formula:

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

Discrete Fourier Transform(DFT)

If we are given a a function at a discrete set of points, we can never know about extremely high frequency between these points.The DFT is similar to the Fourier series but truncated at some maximum frequency to reflect this potential shortcoming. Suppose we are given a sequence (or vector) of N values that are a function sampled at equally-spaced points {x0, x1, x2, . . . , x(N-1)}. The Discrete Fourier Transform is a sequence of numbers:

$$x\_hat\_k = \frac{1}{N} \sum_{n=0}^{N-1} x\_n * e^{\frac{2\pi i k n}{N}}$$

Fast Fourier Transform(FFT)

The FFT does faster than DFT: the total complexity of DFT is O(N^2), while the total complexity of FFT is only O(Nlog(N)). The idea behind FFT is that if we have a sequence of length N, we can split the DFT into two of length N/2 and repeat the process over and over again, referred to as a divide and conquer algorithm. MATLAB has a built-in fft function to perform the process.

The Gabor Transform

The Gabor transform, or STFT, is given precisely by

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

Let us consider g(t) a filter function. Then, shifting by τ and multiplying by a function f(t) represents the filtered function f(t)g(t-τ), with the filter centered at τ. That is, for a fixed τ the function $\tilde{f}_g(\tau, k)$ gives you information about the frequency components near time τ. We should note that your results are dependent on the choice of filter g(t), hence the subscript g on $\tilde{f}$. There are many choices for g and some commonly used assumptions are:

1. The function g is real and symmetric.
2. The L2 norm(total energy of a function) of g is set to unity.

To keep the window function g(t) simple, we will use an Gaussian:

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

$a > 0$ represents the with of the window(a small a means a wide window while a large a means a thin window), τ represents the center of the window. With a wider window we can get better frequency resolution, but we won't be able to localized the signal in time. With a thinner window, we are localizing in time, but then get poor resolution in frequency space.

Discrete Gabor Transform:

Like Fourier Transform, we need a discrete version to actually use the Gabor Transform. A discrete set of frequencies where m and n are integers and w0 and t0 are positive integers:

$$k = mw_0$$
$$t = nt_0$$

The Discrete Gabor Transform is defined by

$$\tilde{f}_g(m, n) = \int_{-\infty}^{\infty} f(t)g(t-nt_0)e^{2\pi imw_0 t}dt$$

If we want to reproduce the signal, we need a large enough window to have some overlap (or a small enough step t0). Similarly, we need a small enough w0 to get good frequency resolution.

Spectrogram

It’s not a great way to convey information through a video to illustrate the sliding window over the domain. Rather, a spectrogram allows us to stack all the Fourier transforms next to each other to make a plot with the horizontal direction being the value of (window center) and the vertical direction the frequency. This will give us a still that represents the changing of the Fourier transform as the window slides over the domain.
(Reference: Amath482 lecture notes from Professor Jason Bramburger)

Algorithm implementation and development:

First, I used audioread function to read the m4a file and return the sampled data and its sample rate. Dividing the data by its rate, I got the time in seconds. I made the frequency component K = (1/L)*[0:(n/2 -1) -n/2:-1]. Specifically, I scaled the frequencies by 1/L rather than 2pi/L since the scale used for the frequencies of musical notes is Hertz. I reorganized it by fftshift function. After I made my window function with Gaussian filter function in the frequency domain, I multipled the signal by the window function, applied FFT by fft function, and rearranged the outputs by fftshift function. Then I slided the window across the domain by t=0.1 each time, calculated the FFT, and reorganized it by fftshift method at each shift by making a loop. I then made my spectrogram by the use of pcolor, corlormap, and colorbar functions and adjusted the tau(determine the window center) and a(determines the window size) according to the spectrogram. To be specific, the larger the a, the thinner the window, while the smaller the a, the wider the window. To get the guitar from the GNR clip and bass from the Floyd clip respectively, I used ylim function to filter out overtones respectively. Finally, I found the frequencies of the music note, remarked on the spectrogram by yticks and yticklabels function, and reproduced the music score for the guitar in the GNR clip and the bass in the Folyd clip(changed the filename and repeated the process). Similarly, I used ylim function to filter out frequencies of guitar and isolate the bass in Comfortably Numb(below 200 Hertz). For the guitar solo, I also applied ylim function to filter out overtones and bass successively.

Computational results:

1. Guitar in the GNR clip:

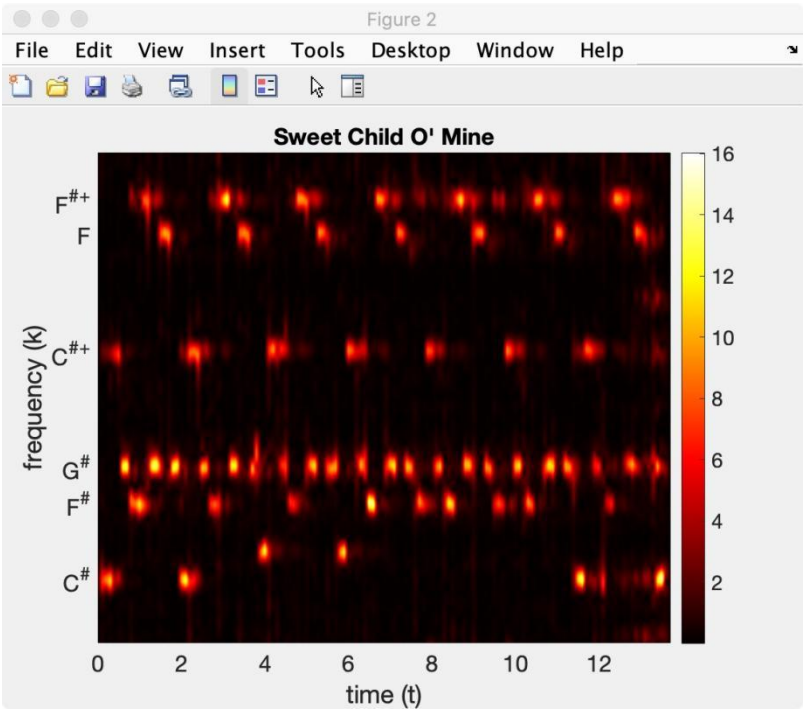

Figure 1: The spectrogram of Sweet Child O' Mine with a = 1500 and tau=0.1 for the window function and frequency(y-axis) ranges from 200 to 800

Music score for the guitar in Sweet Child O' Mine:

{C#, F#, G#, C#(+), F, F#(+)}(low to high)

Bass in the Floyd clip:

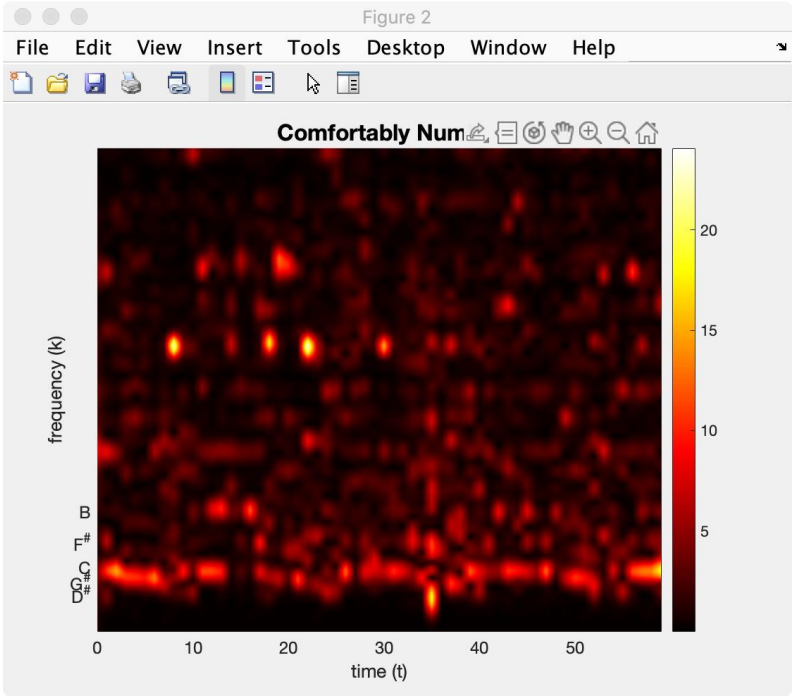

Figure 2: The spectrogram of Comfortably Numb with a = 5500 and tau = 1 for the window function and frequency(y-axis) ranges from 0 to 1000

Music score for the bass in Comfortably Numb:

{D#, G#, C, F#, B}(low to high)

2.

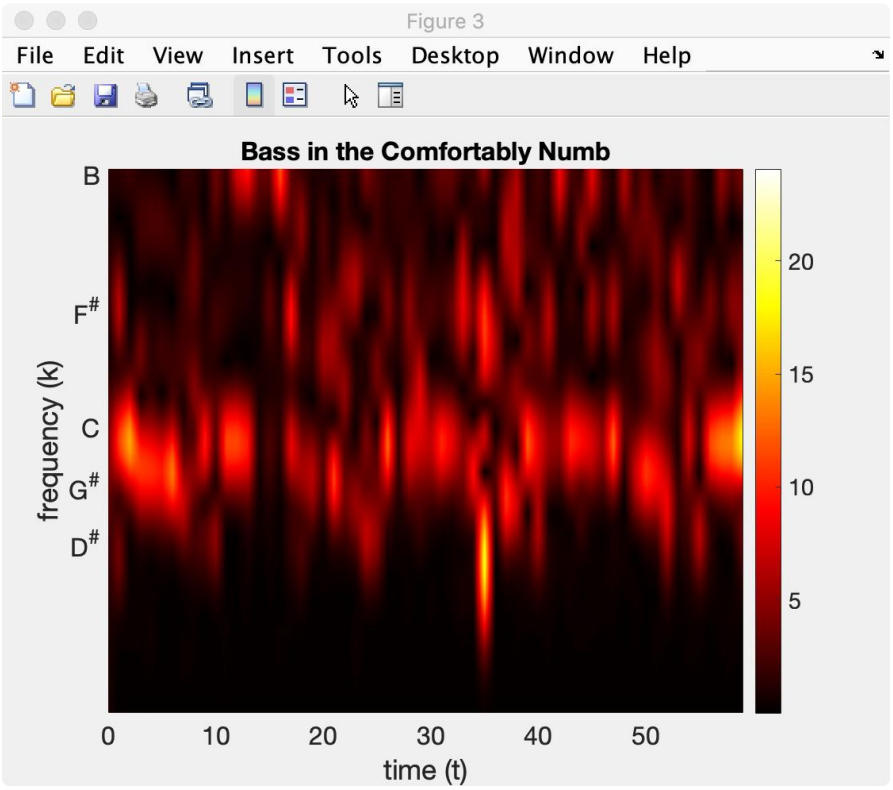

Figure 3: The spectrogram of the bass in the Comfortably Numb with a = 5500 and tau = 1 for the window function and frequency(y-axis) ranges from 0 to 250

3.

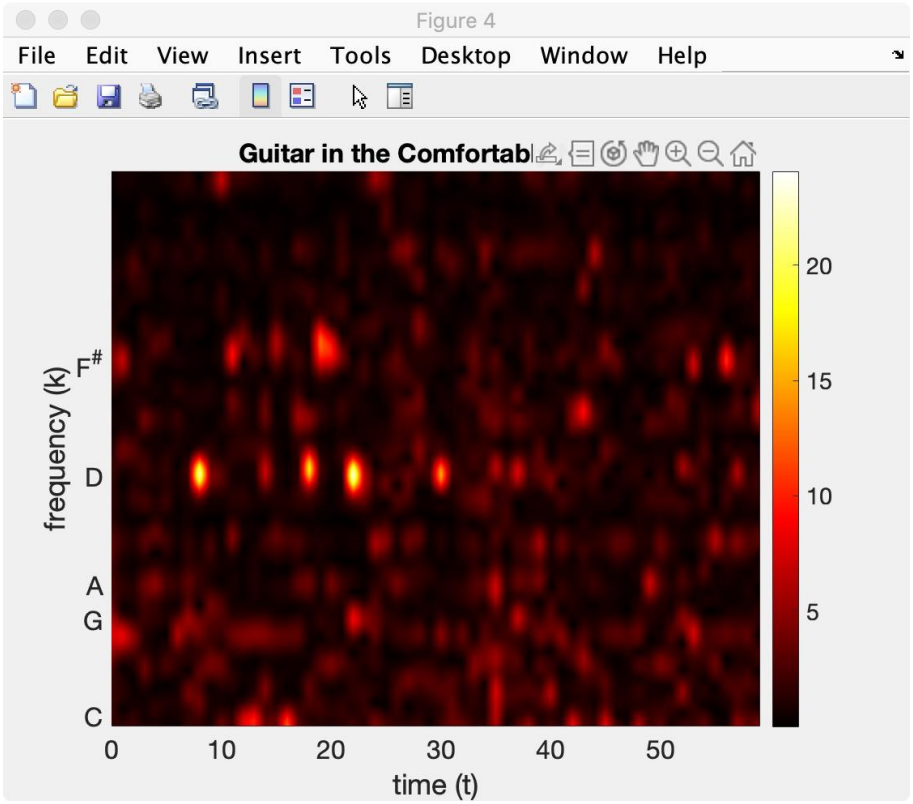

Figure 4: The spectrogram of the guitar in the Comfortably Numb with a = 5500 and tau = 1 for

the window function and frequency(y-axis) ranges from 250 to 1000
Music score: {C,G,A,D,F#}(low to high)

Summary and conclusions:

I used Gaussian filter function as my window function, multipled the signal by the window function, and applied FFT(fft and fftshift functions) at each subdomain within a loop. With the use of pcolor, corlormap, and colorbar functions, I made my spectrogram and adjusted factors tau and a accordingly. Employing ylim function to filter out frequencies of overtones and certain instruments, I got the guitar from Sweet Child O' Mine, the bass and guitar solo from Comfortably Numb, successively. I used yticks and yticklabels function to remark the music note on the spectrogram and found their music scores.

MATLAB function used and brief implementation explanation:

-audioread: read data from audio file and return sampled data and its sample rate
-linespace: generate linearly spaced vector
-fft: compute the DFT using a FFT algorithm
-fftshift: rearrange a Fourier Transform by shifting the zero-frequency component to the center of the array
-pcolor: specify the x- and y-coordinates for the vertices and create a pseudocolor plot using the values in matrix C
-colormap: set the colormap for the current figure to the colormap specified by map
-colorbar: show color scale and display a vertical colorbar to the right of the current axes or chart
-set: set graphics object properties
-ylim: set the y-axis limits for the current axes or chart
-xlabel: label x-axis
-ylabel: label y-axis
-title: add title
-yticks: set the y-axis tick values, which are the locations along the y-axis where the tick marks appear
-yticklabels: set the y-axis tick labels for the current axes

(Reference: MATLAB search helps)

MATLAB codes:

GNR clip:

```
clear all; close all; clc
```

```
figure(1)
[y, Fs] = audioread('GNR.m4a');
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Sweet Child O'' Mine');
p8 = audioplayer(y,Fs); playblocking(p8);

S = y';
trgnr = length(y)/Fs; % record time in seconds
L = trgnr;
n = length(y);
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

a = 1500;
tau = 0:0.1:L;
for i = 1:length(tau)
    g = exp(-a*(t - tau(i)).^2); % Window function
    Sg = g.*S;
    Sgt = fft(Sg);
    Sgt_spec(:,i) = fftshift(abs(Sgt));
end

figure(2)
pcolor(tau,ks,Sgt_spec(1:end,:));
shading interp
set(gca,'ylim',[200 800],'Fontsize',16)
colormap(hot)
colorbar
yticks([277 370 415 554 698 740])
yticklabels({'C^#','F^#','G^#','{C^#}^+','F','{F^#}^+'})
xlabel('time (t)'), ylabel('frequency (k)')
title('Sweet Child O'' Mine','Fontsize',16);

Floyd clip:
clear all; close all; clc

figure(1)
[y, Fs] = audioread('Floyd.m4a');
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Comfortably Numb');
p8 = audioplayer(y,Fs); playblocking(p8);
```

```
S = y';
trgnr = length(y)/Fs; % record time in seconds
L = trgnr;
n = length(y);
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);
Sgt_spec = [];

a = 5500;
tau = 0:1:L;
for i = 1:length(tau)
    g = exp(-a*(t - tau(i)).^2); % Window function
    Sg = g.*S;
    Sgt = fft(Sg);
    Sgt_spec(:,i) = fftshift(abs(Sgt));
end

figure(2)
pcolor(tau,ks,Sgt_spec(1:end-1,:));
shading interp
set(gca,'ylim',[0 1000],'Fontsize',12)
colormap(hot)
colorbar
yticks([78 104 131 185 247]);
yticklabels({'D^#','G^#','C','F^#','B'});
xlabel('time (t)'), ylabel('frequency (k)')
title('Comfortably Numb','Fontsize',16)

figure(3)
pcolor(tau,ks,Sgt_spec(1:end-1,:));
shading interp
set(gca,'ylim',[0 250],'Fontsize',16)
colormap(hot)
colorbar
yticks([78 104 131 185 247]);
yticklabels({'D^#','G^#','C','F^#','B'});
xlabel('time (t)'), ylabel('frequency (k)')
title('Bass in the Comfortably Numb','Fontsize',16)

figure(4)
pcolor(tau,ks,Sgt_spec(1:end-1,:));
shading interp
```

```
set(gca,'ylim',[250 1000],'Fontsize',16)
colormap(hot)
colorbar
% yticks([78 98 110 117]);
% yticklabels({'Em', 'G', 'A', 'Bm'});
yticks([262 392 440 587 740]);
yticklabels({'C','G','A','D','F^#'});
xlabel('time (t)'), ylabel('frequency (k)')
title('Guitar in the Comfortably Numb','Fontsize',16)
```