Filtering Rock & Roll accompaniment with Gabor transform

Crystal Yang

ABSTRACT

Using Gabor Transform, one can process the signal for both its time and frequency information which is a great finding in signal processing field. We will be going to examine this on two famous RNR songs and extract the instrument sound of the two songs using this technique.

1 INTRODUCTION

In this assignment, I am going to examine the Gabor transform by applying it with two music clips that I have: *Sweet Child O' Mine* by Guns N' Roses and *Comfortably Numb* by Pink Floyd, and I am going to filtered out the instrument sound bass and guitar from these two clips. This is done by applying the Gabor transform and Fast Fourier Transform to the sound signal, and I am going to process the signal in MATLAB.

The use of Fourier Transform allows one to transform the time information into frequency information, this is a great finding however we will lose time information when doing the Fourier Transform, and it is best working with detecting something during the time range. We can know the existence of signal producer by Fourier Transform and it works best for stationary and periodic signal. In most of the time, when we are processing a signal, we want to analyze its frequency information and at the same time learning about their time information. "When does this frequency happen?" becomes the question we want to find an answer for. Thus, the idea of applying a filter to sub-time domain and looking at each time's information arose. This becomes the basic of Gabor Transform, and this is very useful in our task to analyze two music clips seconds by seconds to study its frequency. Gabor Transform allows one to focus on small portion of the signal, and analyze the small portion, then put them together to get both time and frequency information. It is like sliding a 'window' across the signal, the window corresponds to the filter we apply, and this 'sliding' process will be shown using a spectrogram.

2 BACKGROUND THEORY – GABOR TRANSFORM

Gabor Transform still depends on Fast Fourier Transform to transform time function into frequency function, however, now we want to split the time domain into lots of subdomain, each subdomain the center is describe as τ, and the width of the subdomain is described as α. So, we can portion the time information into various subdomain and in each domain, we apply the FFT to the time function to get the information of frequency.

Given a filter function $g(t)$, we apply this function shifted by τ to the original time function $f(t)$ to get the filtered function $f(t) * g(t-\tau)$ with every time, and we can thus, we can get the Gabor Transform of this function as:

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

This Gabor Transform gives frequency information around time τ, and the result is dependent on the choice of filter g(t). Note that this is used in a continuous dataset, in this case, we are having a discrete data points consist of the frequency information in the two songs, thus, we need a discrete version of the Gabor Transform, consider the discrete set of the frequencies:

$$k = m\omega_0$$

$$\tau = nt_0$$

Where m and n are integers and ω_0 and t_0 are frequency information, the discrete Gabor Transform is given by:

$$\tilde{f}_g(m, n) = \int_{-\infty}^{\infty} f(t)g(t-nt_0)e^{2\pi im\omega_0 t}dt$$

Thus, the above Gabor Transform allows us to analyze the frequency information during specified time domain around τ. The filter we are using in this article will be Gaussian filter, which is also the filter used when Gabor did the test. We will decide the width of the Gaussian filter we want to apply to the data consider the situation of the dataset.

Gabor Transform gives us the ability to analyze frequency information time by time, we still need a way to present our result. We will represent this using a spectrogram. A spectrogram is a visualization of spectrum of a frequency with respect to different time. In a spectrogram, we can capture the music note that being played in the song and we can compare it with the music sheets of these two songs. The part that has higher frequency will be highlighted in the spectrogram so we will be able to see that the spectrogram is acting like a music note.

3 ALGORITHM IMPLEMENTATION & DEVELOPMENT

Sweet Child O' Mine by Guns N' Roses

We are going to study the music score of guitars played in this song, and we will use MATLAB to do the Gabor Transform on the dataset. After loading the signal data into the MATLAB, we get the input for our algorithm. The signal is stored in an array of size 659122, this array represents the signal information of the 13.73s clip. We want to look and perform FFT for each 0.05 seconds, that is, using a for loop, for every 0.05 seconds, we multiply the signal by a Gaussian filter centered at that time and perform the Fast Fourier Transform using `fft()` in MATLAB to generate the frequency information.

I am applying a really small window here with a = 100, this large a means we are looking at a very small portion of the data each time. Since we are having a lot of data points, and there are many zeroes in the frequency field, I will sort out those data points whose value is zero just to save space using the `sort()` command in MATLAB, which will sort the given input in descending order. I am keeping the first 100,000 data points and eliminant those that are too small. The output will be a matrix containing

both the frequency information and the time information, the size is 100,000 rows by 275 columns. The spectrogram built from this matrix is included in the next section, we can recognize the music note that being played looking at the spectrogram.

***Comfortably Numb* by Pink Floyd**

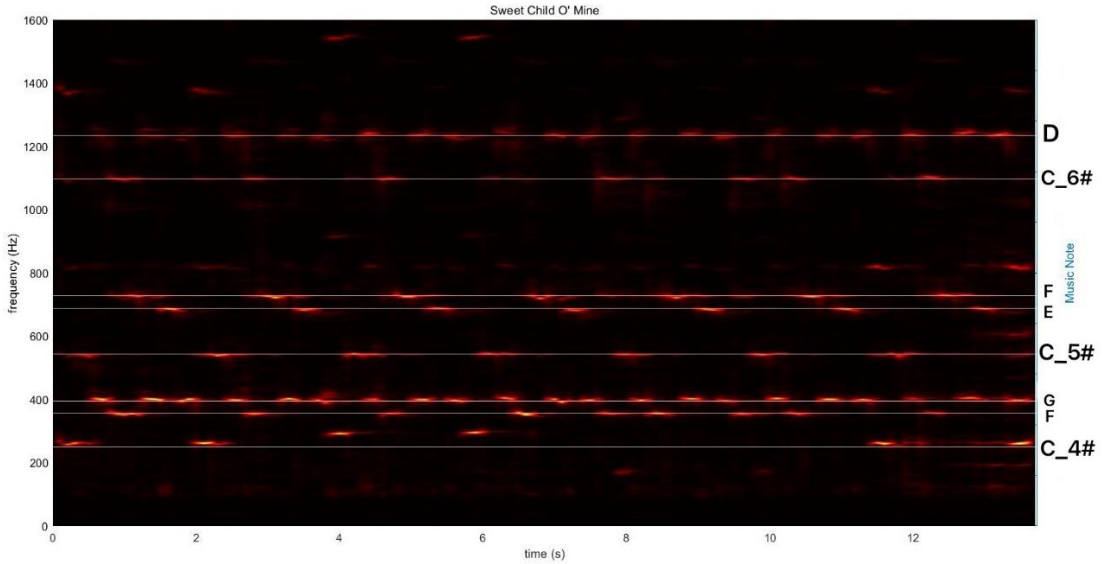
To produce the spectrogram representing the music notes, I will use the same procedure to perform Gabor Transform as above to get the spectrogram. The input for this algorithm is a vector that representing the 59s music, and its size is 2635921 rows, and I portioned it into 4 pieces each last for 14.94s. We want to isolate the sound of bass and look at its music score for each portion of the signal, this requires us to first apply the filter to the frequency of the signal (the FFT of the signal), this will filter out the high frequency. The bass sound is in the range from 60 Hz to 250 Hz, so I set those frequency that are over 200 Hz to be zero to just remain the data points within 200 Hz. Then I performed the Gabor Transform on the filtered signal to build a spectrogram.

We are also interested in finding the music score played by guitars in *Comfortably Numb*. The guitar sound in this clip cannot be well recognized since there are instruments that are playing at the same time and their frequencies range have a lot of overlap. So the spectrogram is not perfect due to this reason. The filter that I used to filter out the low frequency sound is this `highpass()` filter in MATLAB, and by setting a passband filter to 150 Hz, it only keep the signal that are higher than this value.

4 COMPUTATIONAL RESULTS

***Sweet Child O' Mine* by Guns N' Roses**

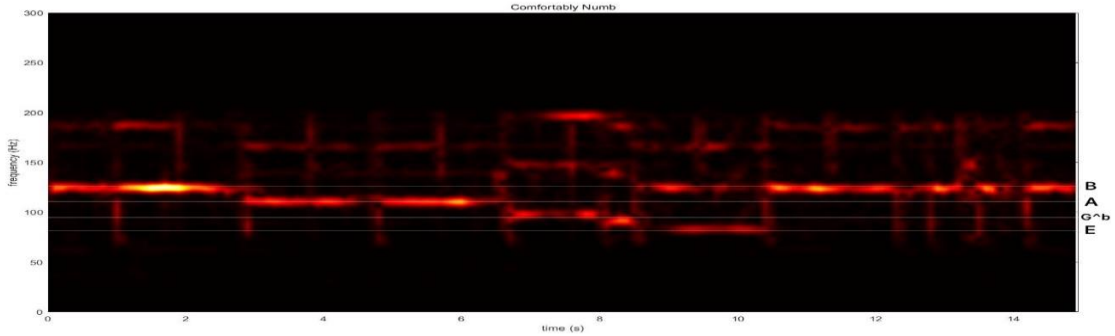
With the implementation above, we can draw the spectrogram, and we can compare this spectrogram with the music score. We can see that the spectrogram lines up with the music score being played, and the melodic contour of the music notes.



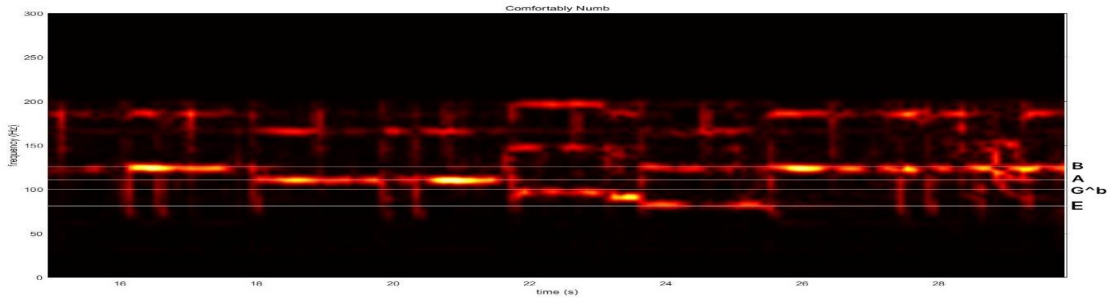
Spectrogram of GNR.m4a

***Comfortably Numb* by Pink Floyd**

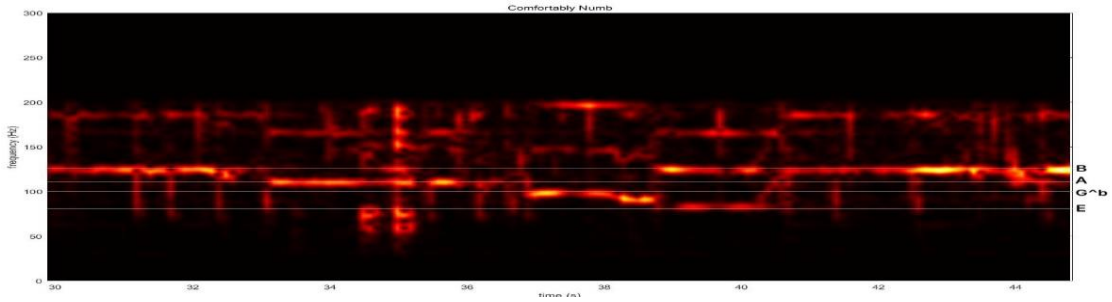
The spectrogram of Floyd.m4a looks like this after we run the algorithm with the signal information, the lower part represents the bass sound:



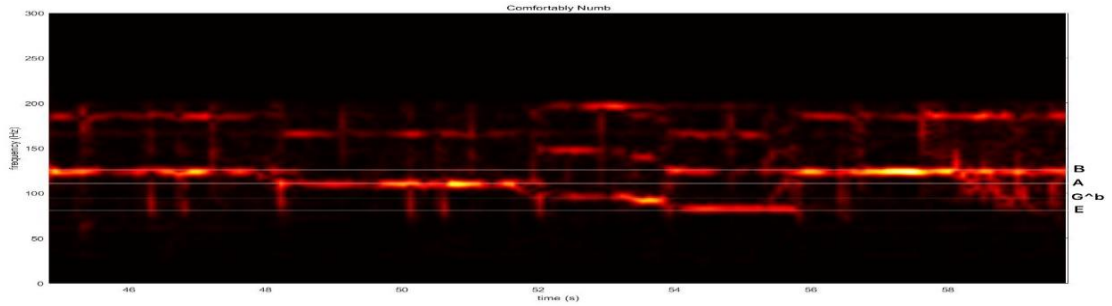
Bass sound from 0 - 14s



Bass sound from 14-29s

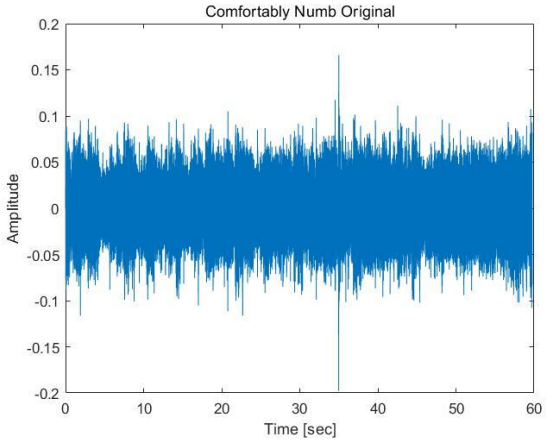


Bass sound from 29-45s

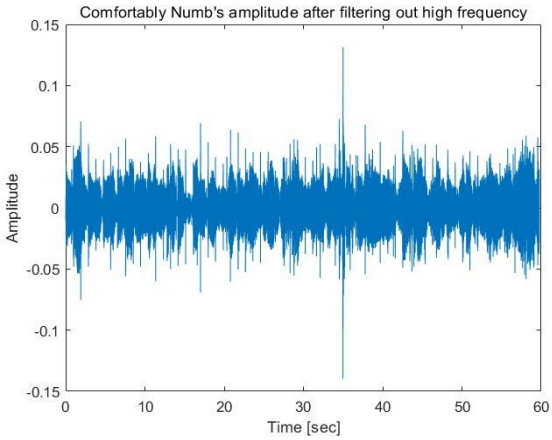


Bass sound from 45-59s

By comparing the frequency value with the music note's different value, we can get the possible music notes that are being played in *Comfortably Numb*. Now let us look at the amplitude graph showing the signal before and after applying the filter to the frequency domain to keep the sound of bass:

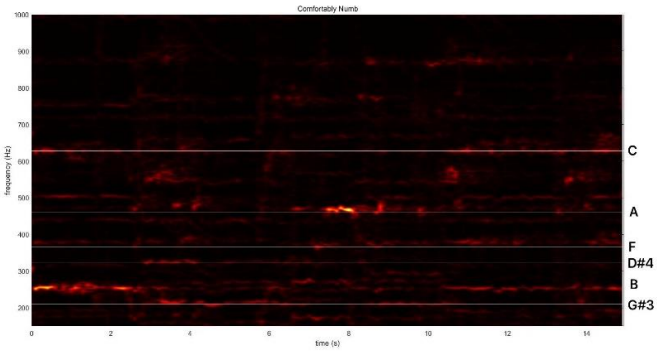


Original Signal

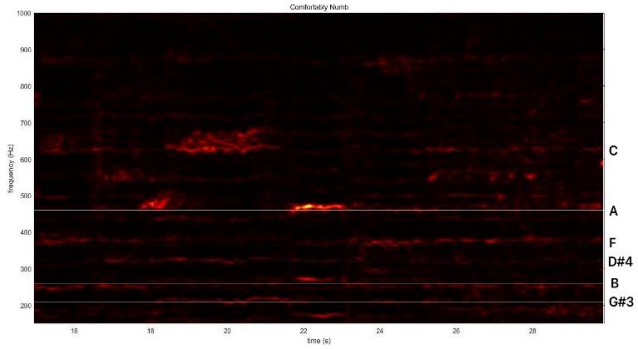


Filtered Signal

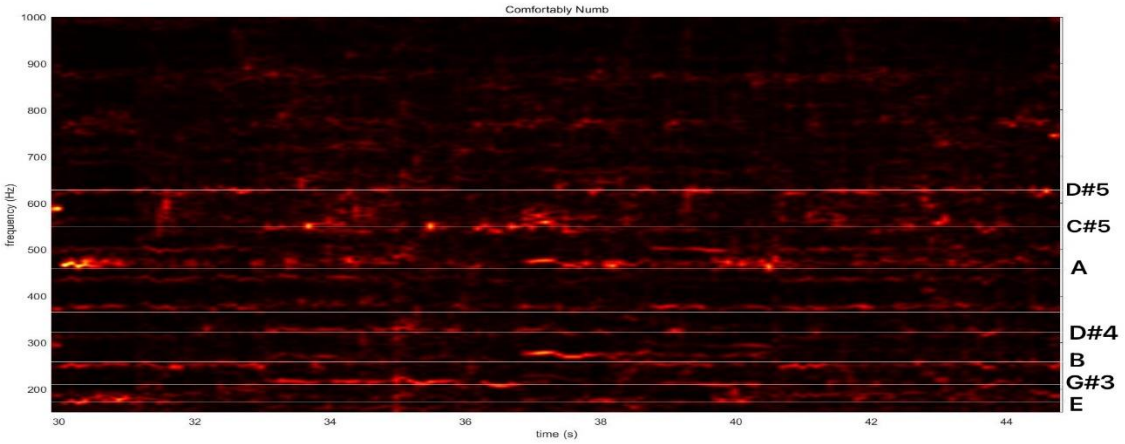
The guitar notes being played in the *Comfortably Numb:*



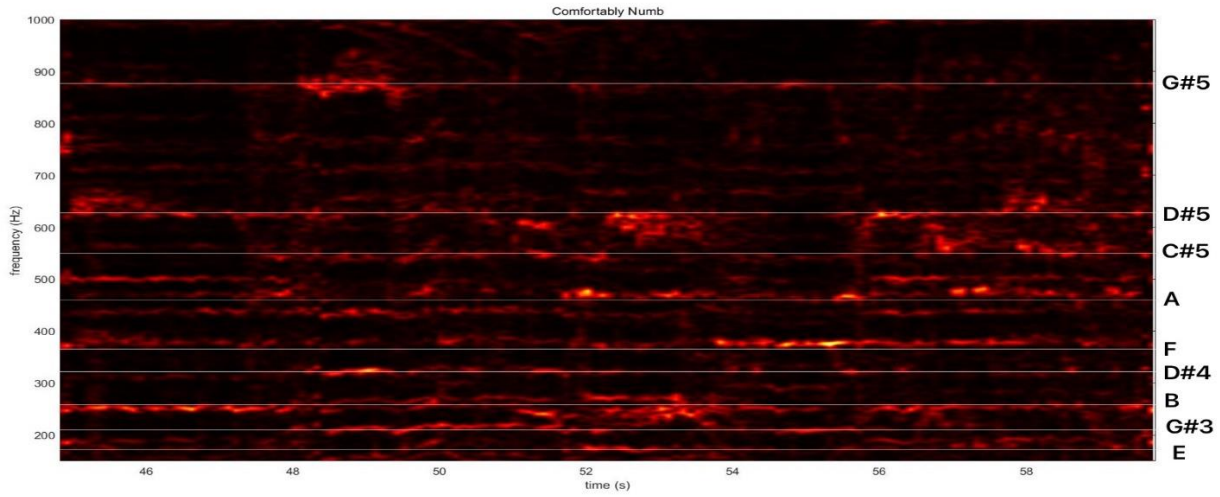
Guitar Spectrogram (0-14s)



Guitar Spectrogram(15-29s)



Guitar Spectrogram (29-45s)



Guitar Spectrogram (45-29s)

5 SUMMARY & CONCLUSIONS

Using Gabor Transform, we are able to generate both time and frequency information of a signal and recover the music sheet by identifying the frequency of that specific notes. By using the passband filter to specify a certain cutoff frequency, we can look at certain range of the frequency in the signal to isolate sound played by certain instruments.

However, we notice the drawback of using a simple passband filter when we are trying to isolate the guitar sound in the *Comfortably Numb,* we are not able to extract the guitar sound when there are other instruments that are playing at the same time. The possible next step if we want to get a better result of the guitar music note, is to design a more sophisticated filter such that the cutoff frequency is not like a straight line, but a curve, so that the cutoff frequency is changing when there are other sound in the signal.

APPENDIX A. MATLAB functions used and brief implementation explanation

Highpass():y = highpass(x,fpass,fs), returns a filtered signal with specifying passband frequency fpass in Hz, an original signal x, and the sample rate of x at fs.

fft(): this function calculate the discreate Fourier Transform of the input vector in 1 dimensions;

ifft():calculate the inverse discrete Fourier Transform of the input, turn the function of frequency back to function of time

APPENDIX B. MATLAB Codes

```
%% GNR.m4a  BASS
clear all; close all; clc

[y, Fs] = audioread('GNR.m4a');
tr_gnr = length(y)/Fs; % record time in seconds
%%
t = transpose((1:length(y))/Fs);
a = 100;
tau = 0:0.1:tr_gnr;
for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2); % Gausian filter
    fl_filtered = g.*y;
    fl_t = fft(fl_filtered);

    fl_t_sort = find(abs(fl_t)>0.01);
    fl_t = fl_t(fl_t_sort);
    fl_t = fl_t(1:50000);

    Sgt_spec(:,j) = fftshift(abs(fl_t)); % We don't want to scale it
end
%%
L = tr_gnr;
n = length(Sgt_spec);
k = (1/L)*[0:n/2-1 -n/2:-1]; % frequency domain
ks = fftshift(k);

pcolor(tau,ks,Sgt_spec);
axis([0 tr_gnr 0 1600]); % change the xlim to fit the portion of the data
shading interp
colormap(hot)
yline(250,'w','Linewidth',0.01);
% text(0.3,250,'C','w');
yline(357,'w')
% text(1.5,357,'G^b','w');
yline(395,'w');
% text(2.6,395,'G_4','w');
yline(544, 'w');
% text(2.3,544,'C_5','w');
yline(688,'w');
% text(5.4,688,'E','w');
yline(728,'w');
% text(5,728,'F_5','w');
yline(1097,'w');
% text(4.7,1097,'C_6','w');
yline(1235,'w');
% text(4.4,1235,'D_6','w');
yyaxis right
ylabel('Music Note');
yyaxis left

title("Sweet Child O' Mine");
xlabel('time (s)'), ylabel('frequency (Hz)')
```

```
%% Floyd.m4a
clear all; close all; clc

[y, Fs] = audioread('Floyd.m4a');
tr_fld = length(y)/Fs; % record time in seconds
y = y(3*(length(y)/4):4*(length(y)/4)); % change the portion to look at
tr_fld = length(y)/Fs;
%% SPECTROGRAM
t = transpose((1:length(y))/Fs);
a = 200;
tau = 0:0.1:tr_fld;
for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2); % Gausian filter
    fl_filtered = g.*y;
    fl_t = fft(fl_filtered);

    fl_t_sort = find(abs(fl_t)>0.01);
    fl_t = fl_t(fl_t_sort);
    fl_t = fl_t(1:100000);

    Sgt_spec(:,j) = fftshift(abs(fl_t)); % We don't want to scale it
end
%% BASS 1
L = tr_fld;
n = length(y);
k = (1/L)*[0:n/2-1 -n/2:-1]; % frequency domain

y_t = fft(y); % got the frequency information
% fc = 250; % low pass filter with 250 Hz
y_t(abs(k) > 200) = 0; % excludes those frequency are higher than 250 Hz
y_filtered = ifft(y_t);
plot((1:length(y))/Fs,y_filtered);
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title("Comfortably Numb Original");



%% BASS 2
a = 200;
t = transpose((1:length(y))/Fs);
tau = 0:0.1:tr_fld;
for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2);
    fl_filtered = g.*y_filtered;
    fl_t = fft(fl_filtered);

    fl_t = fl_t(1:100000); % add filter to filter out frequency that out of tone.

    bass_freq(:,j) = fftshift(abs(fl_t)); % We don't want to scale it
end
```

```
%% GUITAR 1
L = tr_fld;
n = length(y);
k = (1/L)*[0:n/2-1 -n/2:-1]; % frequency domain
guitar_filtered = highpass(y, 150, Fs);

%% GUITAR 2
a = 200;
t = transpose((1:length(y))/Fs);
tau = 0:0.1:tr_fld;
for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2);
    fl_filtered = g.*guitar_filtered;
    fl_t = fft(fl_filtered);

    fl_t_sort = find(abs(fl_t)>0.01); % want to excludes those value that are zero
    fl_t = fl_t(fl_t_sort);
    fl_t = fl_t(1:100000); % add filter to filter out frequency that out of tone.

    guitar_freq(:,j) = fftshift(abs(fl_t)); % We don't want to scale it
end

%% PLOT
L = tr_fld;
n = length(guitar_freq)+1;
k = (1/L)*[0:n/2-1 -n/2:-1]; % frequency domain
ks = fftshift(k);
ks = ks(1:length(guitar_freq));
tau = 3*tr_fld:0.1:4*tr_fld;

pcolor(tau,ks,guitar_freq);
axis([3*tr_fld 4*tr_fld 150 1000]); % change the xlim to fit the portion of the data
shading interp
colormap(hot)
title("Comfortably Numb");
xlabel('time (s)'), ylabel('frequency (Hz)')
% guitar
yline(460,'w');
yline(258,'w');
yline(210,'w');
yline(365,'w');
yline(322,'w');
yline(628,'w');
yline(172,'w');
yline(550,'w');
yline(877,'w');
% bass
% yline(126,'w');
% yline(111,'w');
% yline(95,'w');
% yline(81,'w');
```