Rock & Roll and the Gabor Transform

Luotong Kang

Abstract

Using the Gabor Transform, we are able to perform time-frequency analysis. In this report, we are going to implement Gabor Transform using MATLAB to reproduce the music score for a portion of two of the greatest rock and roll song.

1 Introduction

Gabor Transform, also called Short Time Fourier Transforms, can tell information in both time and frequency space. By doing so, we can analyze music score of the songs *Sweet Child O' Mine* by Gums N' Roses and *Comfortably Numb* by Pink Floyd. To start, we'll reproduce the guitar scores in *Sweet Child O' Mine* in Part I by directly creating a spectrogram. Then, we'll look at the bass score in *Comfortably Numb* in Part II. Since this portion is an ensemble of several different instruments, we'll need to isolate bass either by directly setting the graph window size or using a filter in frequency space beforehand. In the last part, we'll try to put together the guitar solo in *Comfortably Numb*.

2 Theoretical Background

In last report, we've looked at basics of Fourier Transform, and I'll assume the readers is familiar with it. In Fourier transform, we loses information about when certain frequencies occur or how the frequencies change over time. Different from Fast Fourier Transform, we are trading of processing speed to capture time information. The Gabor transform improves to work for non-stationary signals by creating time windows. To avoid sharp transitions, the Gabor Transform uses a time filter to pick out windows. With those modification, the Gabor Transform is defined as:

$$\tilde{f}_g(\tau, k) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-ikt}dt, \text{ where } g(t-\tau) \text{ is the time filter.}$$

We assume the filter to be real and symmetric, and it's L_2-norm is often 1. For the rest of this report, I'll just choose the Gaussian as my time filter:

$$g(t-\tau) = e^{-a(t-\tau)^2}, \text{ where } a \text{ represents the window width.}$$

The smaller a we choose, the wider a window will be.

In this report, we'll also need some knowledge of music scale. The notes that I'll be using and their corresponding frequency is listed here:

E2	F2	F#2	G2	G#2	A2	A#2	B2	C3	C#3
82.407	87.307	92.499	97.999	103.83	110	116.54	123.47	130.81	138.59
D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3
146.83	155.56	164.81	174.61	185	196	207.65	220	233.08	246.94
C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4
261.63	277.18	293.66	311.13	329.63	349.23	369.99	392	415.3	440
A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5	G5
466.16	493.88	523.55	554.37	587.37	622.25	659.26	698.46	739.99	783.99

Table 1: Music Notes in Hz

3 Algorithm Implementation and Development

3.1 Part I (Lines 1-66)

After importing the portion from *Sweet Child O' Mine* into MATLAB, we first initiate time line using the sampling rate `Fs`. It is crucial to reorder the frequency array beforehand as in the previous report. In addition, we need to scale it by `1/L` to have a result in Hertz. We choose a 0.1 second time step to avoid losing information, even though a small time step will lower the processing speed.

If the time window is huge, we cannot get accurate information about time. Vice versa. This is not a trade-off about processing time anymore. In Lines 12-30, we create spectrograms (Figure 1) with different window width to find the most appropriate window size. We can choose a wide enough frequency range to draw at this point as a exploration of true frequency range of guitar.

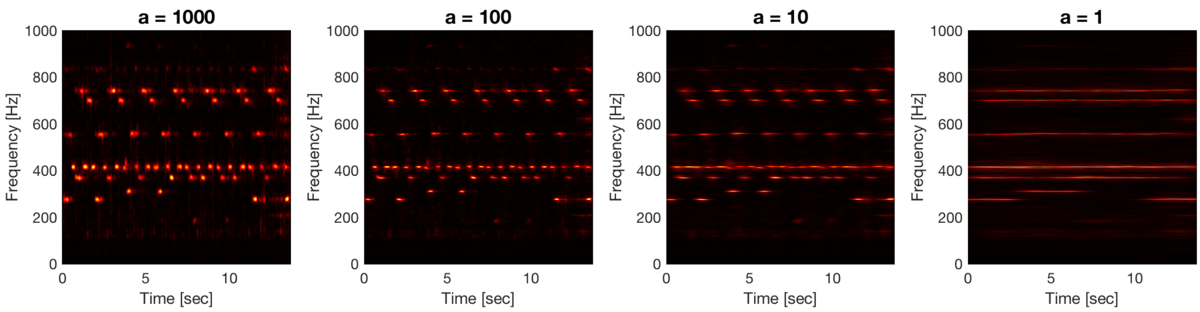

Figure 1: Spectrograms for *Sweet Child O' Mine* with different window width.

From the figure above, we can see that a=100 is clear enough win both time domain and frequency domain. We'll choose a=100 window width to continue to derive our computational results. We can also narrow down the choice of frequency bounds to 200 to 800 because we detect most of our signal in that range.

Lines 31-65 make a side-by-side comparison between the selected spectrogram and the reproduced score with notes name listed in frequency axis. The reproduced score is created using `scatter()`, in which the most outstanding frequency in each window is plotted. This strategy depends on the fact that guitar in this clip is clearly identifiable and we have a small enough time step.

3.2 Part II (Lines 67-194)

In Lines 67-84, we initiate arrays in a similar way as in Part I. However, because the array y extracted from 60s music is too large, the computer refuses to run the code when we set the time step to divide L into 600 windows. Thus we want to separate the data into 4 equal intervals (each about 15s) along time. The last element of the array is ignored (1/44100s) for the length to be divided with no remainder.

Just like in Part I, we have to make a clever choice for window width. Lines 85-103 create spectrograms for first 15s clip with different window width below:

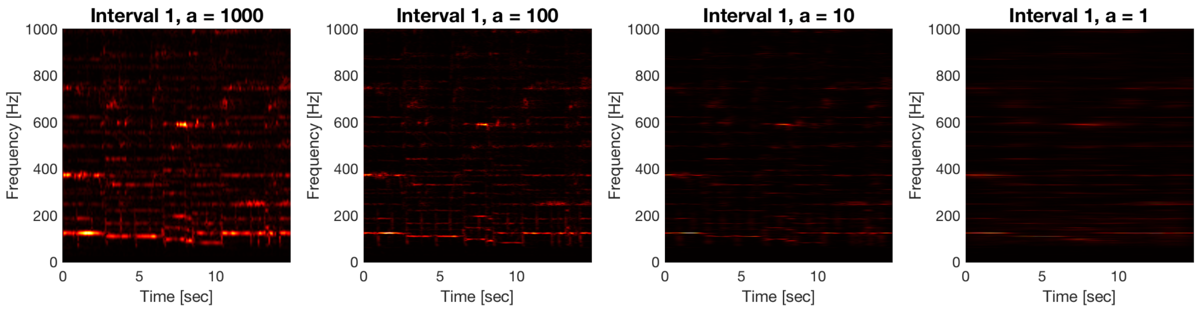

Figure 2: Spectrograms for *Comfortably Numb* with different window width.

From the figure above, we can see that a=100 is clear enough win both time domain and frequency domain. We'll choose a=100 window width to continue to derive our computational results. This figure also gives us some information about frequency range of bass using the fact that bass should be clearly identifiable.

This report will develop two methods for isolating bass sound from the ensemble.

3.2.1 Method 1

Lines 104-144 directly narrow down the plotting window using the frequency range we've searched online and the appearance of spectrograms in Figure 2. Based on that, we plot segmented spectrograms for total 60s with frequency bounds set to 0 to 300 Hz.

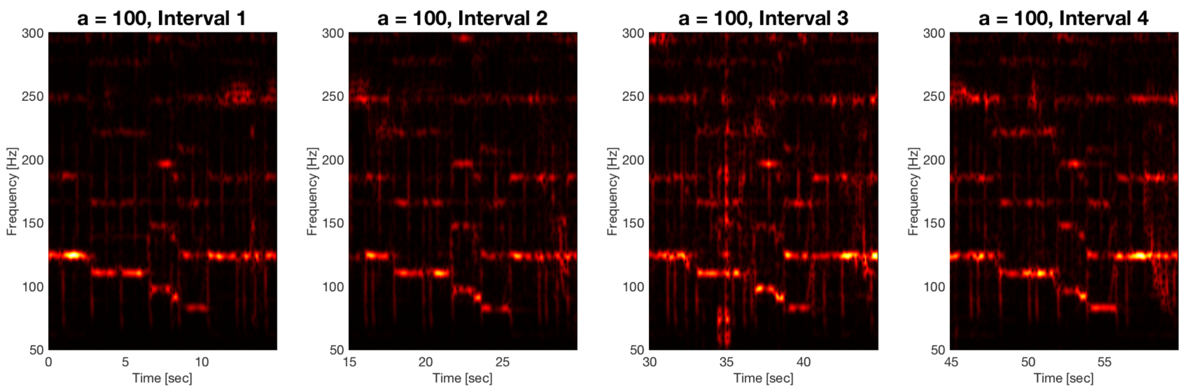

Figure 3: Segmented Sepctrograms for *Comfortably Numb*. The four 15 seconds intervals are numbered chronologically.

From the spectrogram above, we can further adjust our frequency range for bass sound and draw the bass score using `max()` and `scatter()` commands with note names on frequency axis in next section of this report.

3.2.2 Method 2

Lines 145-194 multiply a filter to the function on frequency space before plotting. Remember to use `fliplr()` to compensate FFT mechanism. We may do explorations about frequency range of bass. For convenience, we can also just look at the exploration result in Method 1. It appears that bass frequency in this clip is below 130 Hz for the most of time. Therefore, we set a rectangular filter to make frequencies higher than 130 to return zero. Accordingly, we plot the filtered signal in frequency space and draw spectrogram using filtered data in the same figure.

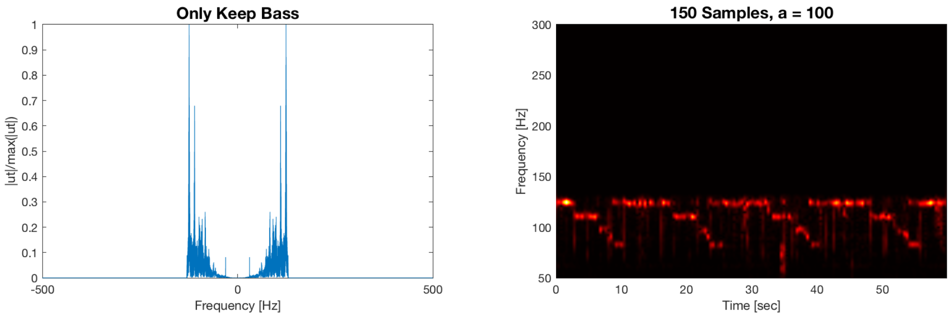

Figure 4: The filtered data in frequency space is on the left. The right portion shows spectrogram plotted using 150 windows.

Different from how we plot Figure 3, we used a bigger time step for Figure 4 to allow MATLAB draw all 60s in one spectrogram. The total number of windows is divided by 4 but we can still see a clear score, showing the efficiency of our filter. We also draw a scatter diagram shown in next section.

3.3 Part III (Lines 196-251)

Lines 196-206 create filters to isolate guitar frequency. Besides eliminating bass, we also set a upper-bound for frequency at 800 Hz because the main portion of guitar solo in *Comfortably Numb* doesn't sound to have a higher pitch than in *Sweet Child O' Mine*.

It is not efficient to take 150 windows for this Part because we do not know whether the guitar or the drum set is more identifiable. Thus, we go back to use 600 windows, meaning that we again have to separate this 60s into 2 interval. Lines 207-224 create filtered segmented spectrogram as Figure 5. The process is a redo of Part II Method 1.

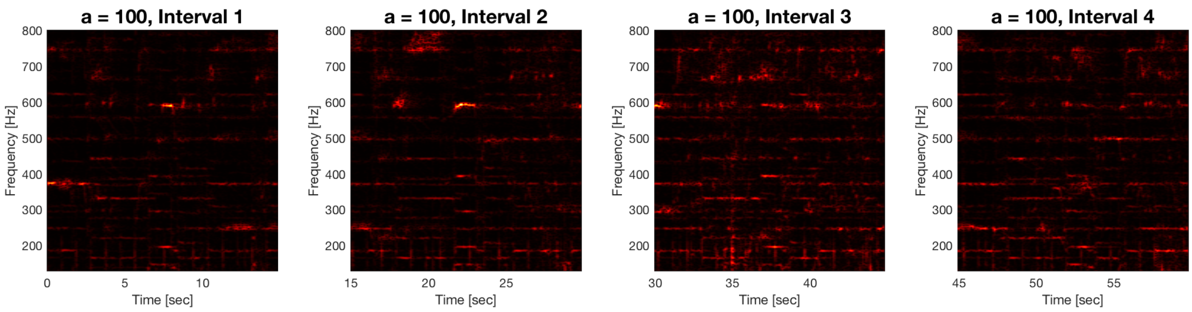

Figure 5: This is the spectrogram of *Comfortably Numb* after isolating 130Hz to 800Hz

From the graph above, we can already recognize some notes. Lines 225-251 create segmented scatter diagram to show the guitar score. Although we don't meet oversize array problem here, it is better to split 60s in 4 pieces to make the score more readable.

4 Computational Results

4.1 Part I

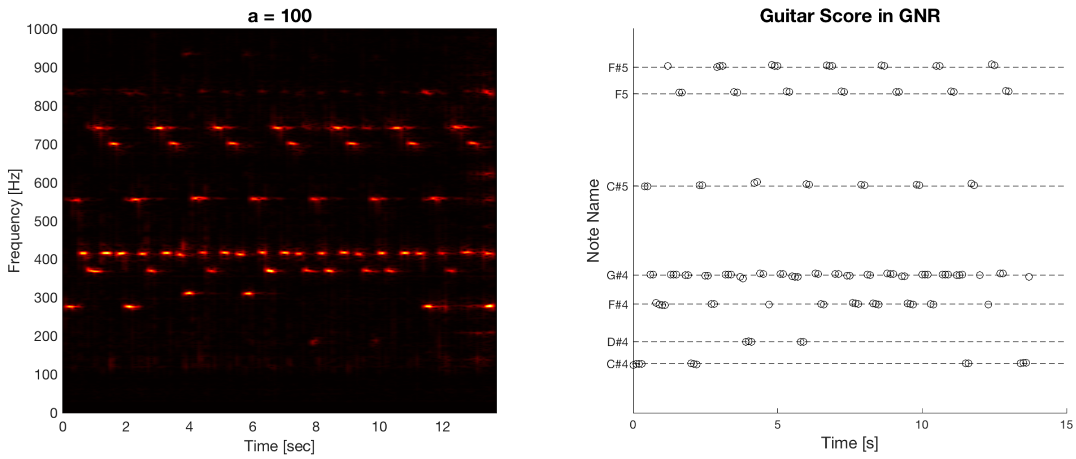

Figure 6: *Sweet Child O' Mine* Spectrogram (left) with appropriated window size and the music score as scatter diagram (right).

The notes played are C#4, C#5, G#4, F#4, F#5, F5, and D#4. See the diagram for order.

4.2 Part II

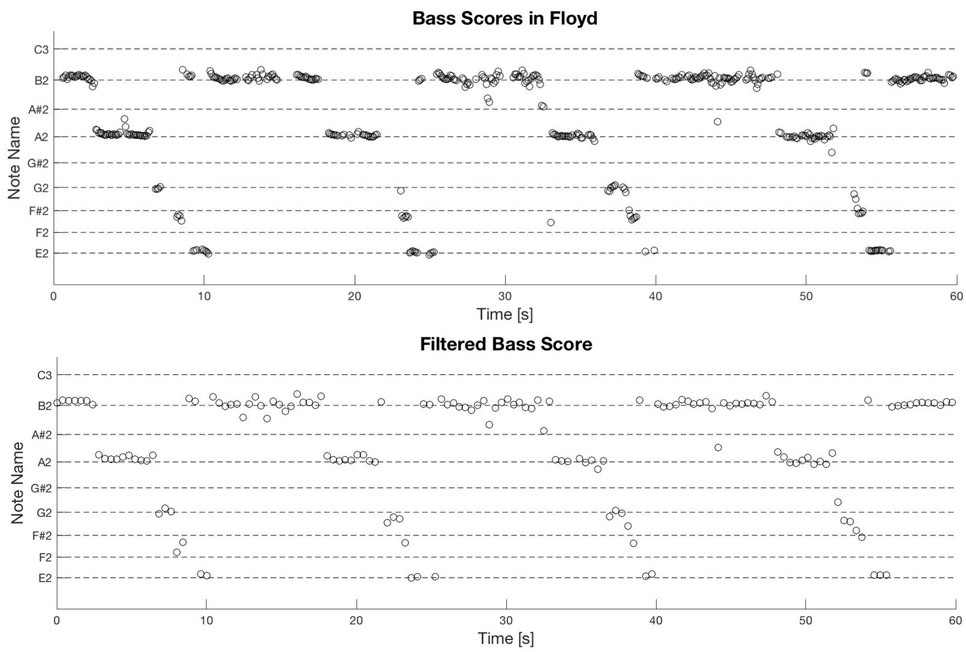

Figure 7: Music score of *Comfortably Numb*.
Upper one uses Method 1 and the lower one uses Method 2

The notes played are B2, A2, G2, F#2 and E2. See the diagram for order.

4.3 Part III

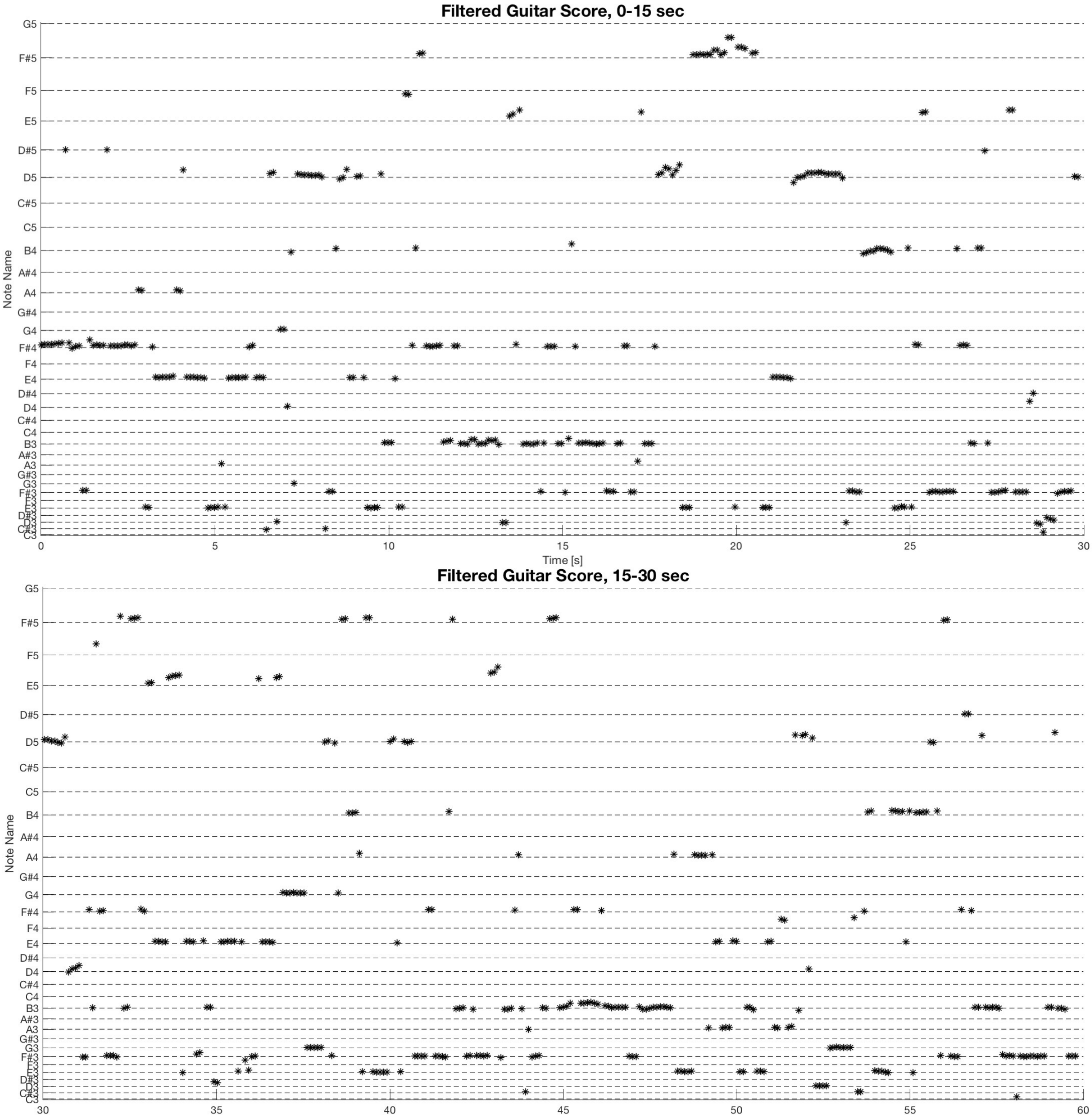

Figure 8: Filtered guitar score for *Comfortably Numb*.

It is hard to tell all the notes in the solo, but we can recognize some of it like F#4, E4, B3, E3, F#3. We can also recognize the "descending" pitchs in 20-25s marked by points on F5, D5, B4, F#4. More details can be found on the graph above

5 Summary and Conclusions

This report is an example applications of time-frequency analysis tools. We use Short Time Fourier Transform to reproduce scores for two songs. Making reasonable decision for window size and getting enough samples is crucial. The ensemble of different instrument increases the difficulties. To deal with it, we isolate the instrument in frequency space.

6 Appendices

6.1 MATLAB functions used

The commands are NOT listed chronologically.

`[y,Fs] = audioread(filename)`: reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.

`linspace(a,b,n)`: creates a vector with n elements linearly spaced between a and b

`fft(A)`: computes the DFT of A using a fast Fourier transform algorithm.

`ifft(A)`: undoes FFT.

`fftshift(A)`: shifts the zero-frequency component to the center of spectrum.

`ifftshift(A)`: undoes fftshift.

`fliplr(A)`: returns A with its columns flipped in the left-right direction.

`subplot(m,n,p)`: divides the current figure into an m-by-n grid and creates axes in the position specified by p.

`pcolor(X,Y,C)`: creates a pseudocolor plot using the values in matrix C with x- and y-coordinates for the vertices specified.

`set(gca,)`, `set(gcf,)`, `title()`, `xlabel()`, `ylabel()`, `yticks()`, `yticklabels()`: edit component labels/axes of the graph.

`saveas(gcf,'filename')`: saves the graphing output as a file.

`reshape(A,[sz])`: reshapes A into a matrix of the size specified by [sz]

`[M,I]=max(A)`: returns the index into the operating dimension that corresponds to the maximum value of A for any of the previous syntaxes.

`abs(A)`: takes the absolute value of every element in A, also works for complex numbers

`exp(n)`: the exponential of the elements of n, e to the n

`scatter(x,y)`: creates a scatter plot with defult circles at the locations specified by the vectors x and y.

6.2 MATLAB codes

```
%% GNR initiate
clc;clear;close all;
[y, Fs] = audioread('GNR.m4a');
n = length(y);
L = n/Fs; % time in seconds
t = linspace(0,L,n+1);
t = t(1:n);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);
S = y;
tau = 0:0.1:L;
%% Explore window width
a=[1000,100,10,1];
for jj = 1:length(a)
    Sgt_spec = zeros(n,length(tau));% Clear at each loop iteration
    for j = 1:length(tau)
        g = exp(-a(jj)*(t - tau(j)).^2); % Window function
        Sg = g.*S';
```

```
        Sgt = fft(Sg);
        Sgt_spec(:,j) = fftshift(abs(Sgt));
    end
    subplot(1,4,jj)
    pcolor(tau,ks,Sgt_spec),shading interp
    set(gca,'Ylim',[0 1000],'Fontsize',12)
    colormap(hot)
    xlabel('Time [sec]'), ylabel('Frequency [Hz]')
    title(['a = ',num2str(a(jj))],'Fontsize',16)
end
set(gcf,'Position',[0,0,1200,250])
saveas(gcf,'GNR_explore_window_width.png')
%% Spectrogram and Scores
a = 100;
Sgt_spec = zeros(n,length(tau));
notes = zeros(1,length(tau));
for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2); % Window function
    Sg = g.*S';
    Sgt = fft(Sg);
    Sgt_spec(:,j) = fftshift(abs(Sgt));
    [M,I] = max(fftshift(abs(Sgt)));
    notes(j) = abs(ks(I));
end
subplot(1,2,1)
pcolor(tau,ks,Sgt_spec),shading interp
set(gca,'Ylim',[0 1000],'Fontsize',12)
colormap(hot)
xlabel('Time [sec]')
ylabel('Frequency [Hz]')
title(['a = ',num2str(a)],'Fontsize',16)
subplot(1,2,2)
for j=1:length(notes)
    scatter(tau(j),notes(j),'k')
    hold on
end
note_names = [277.18,311.13,369.99,415.3,554.37,698.46,739.99];
for j=1:length(note_names)
    plot([0 15],[note_names(j) note_names(j)],'k--')
end
yticks(note_names)
yticklabels({'C#4','D#4','F#4','G#4','C#5','F5','F#5'})
title('Guitar Score in GNR','Fontsize',16)
xlabel('Time [s]','Fontsize',14), ylabel('Note Name','Fontsize',14)
set(gca,'Ylim',[200 800])
set(gcf,'Position',[0,0,1100,400])
saveas(gcf,'GNR_spec_and_score.png')

%% Floyd Initiate
clc;clear;close all;
[y, Fs] = audioread('Floyd.m4a');
S = y(1:(length(y)-1)); %ignore the last term
n = length(S);
L = n/Fs; % time in seconds
t = linspace(0,L,n+1);
```

```
t = t(1:n);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);
tau = linspace(0,L,150*4);
n_int = 658980; %number of data in one interval
L_int = n_int/Fs; %interval in sec
k_int = (1/(L/4))*[0:n_int/2-1 -n_int/2:-1];
ks_int = fftshift(k_int);
tauu = (reshape(tau,[150,4]))';
tt = (reshape(t,[n_int,4]))';
SS = (reshape(S,[n_int,4]))';
%% Explore best window width
a=[1000,100,10,1];
for k = 1:length(a)
    Sgt_spec = zeros(n_int,150);
    for j = 1:150
        g = exp(-a(k)*(tt(1,:) - tauu(1,j)).^2); % Window function
        Sg = g.*SS(1,:);
        Sgt = fft(Sg);
        Sgt_spec(:,j) = fftshift(abs(Sgt));
    end
    subplot(1,4,k)
    pcolor(tauu(1,:),ks_int,Sgt_spec),shading interp
    set(gca,'Ylim',[0 1000],'Fontsize',12)
    colormap(hot)
    xlabel('Time [sec]'), ylabel('Frequency [Hz]')
    title(['Interval 1, a = ',num2str(a(k))],'Fontsize',16)
end
set(gcf,'Position',[0,0,1200,250])
saveas(gcf,'floyd_bass_explore_window_width.png')
%% Method 1: adjust the window based on spectrogram and experience
a=100;
notes = zeros(1,length(tau));
for k =1:4
    Sgt_spec = zeros(n_int,150);
    for j = 1:150
        g = exp(-a*(tt(k,:) - tauu(k,j)).^2); % Window function
        Sg = g.*SS(k,:);
        Sgt = fft(Sg);
        Sgt_spec(:,j) = fftshift(abs(Sgt));
    end
    subplot(2,4,k)
    pcolor(tauu(k,:),ks_int',Sgt_spec),shading interp
    set(gca,'Ylim',[50 300],'Fontsize',9)
    colormap(hot)
    xlabel('Time [sec]','Fontsize',9), ylabel('Frequency ...
        [Hz]','Fontsize',9)
    title(['a = 100, Interval ',num2str(k)],'Fontsize',14)
end
for j=1:length(tau)
    g=exp(-a*(t-tau(j)).^2);
    Sg=g.*S';
    Sgt=(fft(Sg));
    [M,I] = max(fftshift(abs(Sgt)));
    notes(j) = abs(ks(I));
```

```
end
subplot(2,4,[5,6,7,8])
for j=1:length(notes)
    scatter(tau(j),notes(j),'k')
    hold on
end
note_names = [82.407, 87.307,92.499, 97.999, 103.83,110, 116.54, ...
    123.47, 130.81];
for j=1:length(note_names)
    plot([0 60],[note_names(j) note_names(j)],'k--')
end
yticks(note_names)
yticklabels({'E2','F2','F#2','G2','G#2','A2','A#2','B2','C3'})
title('Bass Scores in Floyd','Fontsize',16)
xlabel('Time [s]','Fontsize',14),ylabel('Note Name','Fontsize',14)
set(gca,'Ylim',[75 135])
set(gcf,'Position',[0,0,1000,800])
saveas(gcf,'part2method1.png')
%% Method 2: use a filter in frequency space before drawing ...
    spectrogram/scores
[Jl,pos_ind] = max(-abs(ks-130));% find the boundary of rectangular ...
    filter
[Jh,neg_ind] = max(-abs(ks+130));
filter1 = ...
    [zeros(1,neg_ind-1),ones(1,pos_ind-neg_ind+1),zeros(1,n-pos_ind)];
St = fft(S);
Sf1 = St'.*ifftshift(filter1); %only keep bass
Sft1  = fliplr(ifft(Sf1));
% prep
a = 100;
tau = linspace(0,L,150);
Sgt_spec = zeros(n,length(tau));
notes = zeros(1,length(tau));
for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2); % Window function
    Sg = g.*Sft1;
    Sgt = fft(Sg);
    Sgt_spec(:,j) = fftshift(abs(Sgt));
    [M,I] = max(fftshift(abs(Sgt)));
    notes(j) = abs(ks(I));
end
% plot
subplot(2,2,1)
plot(ks,fftshift(abs(Sf1)/max(abs(Sf1))))
set(gca,'Xlim',[-500 500])
ylabel('|ut|/max(|ut|)')
xlabel('Frequency [Hz]')
title('Only Keep Bass','Fontsize',14)
subplot(2,2,2)
pcolor(tau,ks,Sgt_spec),shading interp
set(gca,'Ylim',[50 300])
colormap(hot)
xlabel('Time [sec]')
ylabel('Frequency [Hz]')
title('150 Samples, a = 100','Fontsize',14)
```

```
subplot(2,2,[3,4])
for j=1:length(notes)
    scatter(tau(j),notes(j),'k')
    hold on
end
note_names = [82.407, 87.307,92.499, 97.999, 103.83,110, 116.54, ...
    123.47, 130.81];
for j=1:length(note_names)
    plot([0 60],[note_names(j) note_names(j)],'k--')
end
yticks(note_names)
yticklabels({'E2','F2','F#2','G2','G#2','A2','A#2','B2','C3'})
title('Filtered Bass Score','Fontsize',14)
xlabel('Time [s]','Fontsize',16), ylabel('Note Name','Fontsize',14)
set(gca,'Ylim',[75 135])
set(gcf,'Position',[0,0,1000,800])
saveas(gcf,'part2method2.png')

%% Add filters to isolate guitar
a = 100;
tau = linspace(0,L,150*4);
notes = zeros(1,length(tau));
[Kl,pos_ind1] = max(-abs(ks-800));% find the boundary of rectangular ...
    filter
[Kh,neg_ind1] = max(-abs(ks+800));
filter2 = ...
    [ones(1,neg_ind-1),zeros(1,pos_ind-neg_ind+1),ones(1,n-pos_ind)];
filter3 = ...
    [zeros(1,neg_ind1-1),ones(1,pos_ind1-neg_ind1+1),zeros(1,n-pos_ind1)];
St = fft(S);
Sf2 = St'.*ifftshift(filter2).*ifftshift(filter3);% eliminate bass
Sft2  = fliplr(ifft(Sf2));
%% Plot Segmented Spectrogram
for k =1:4
    Sgt_spec = zeros(n_int,150);
    for j = 1:150
        g = exp(-a*(tt(k,:) - tauu(k,j)).^2); % Window function
        Sg = g.*SS(k,:);
        Sgt = fft(Sg);
        Sgt_spec(:,j) = fftshift(abs(Sgt));
    end
    subplot(1,4,k)
    pcolor(tauu(k,:),ks_int',Sgt_spec),shading interp
    set(gca,'Ylim',[130 800])
    colormap(hot)
    xlabel('Time [sec]'), ylabel('Frequency [Hz]')
    title(['a = 100, Interval ',num2str(k)],'Fontsize',16)
end
set(gcf,'Position',[0,0,1200,250])
saveas(gcf,'filtered_guitar_spec.png')
%% Plot Score
for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2); % Window function
    Sg = g.*Sft2;
    Sgt = fft(Sg);
```

```
    [M,I] = max(fftshift(abs(Sgt)));
    notes(j) = abs(ks(I));
end
for k=1:2
    for j=1:length(notes)/2
        scatter(tau((k-1)*length(notes)/2 ...
            +j),notes((k-1)*length(notes)/2+j),'k*','LineWidth',0.8)
        hold on
    end
    note_names = [130.81, 138.59, 146.83, 155.56, 164.81, 174.61, ...
        185, 196, 207.65, 220, 233.08, 246.94, 261.63, 277.18, ...
        293.66, 311.13, 329.63, 349.23, 369.99, 392, 415.3, 440, ...
        466.16, 493.88, 523.55, 554.37, 587.37, 622.25, 659.26, ...
        698.46, 739.99, 783.99];
    for j=1:length(note_names)
        plot([(k-1)*30 30+(k-1)*30],[note_names(j) note_names(j)],'k--')
    end
    yticks(note_names)
    yticklabels({'C3','C#3','D3','D#3','E3','F3','F#3','G3','G#3','A3', ...
        'A#3','B3','C4','C#4','D4','D#4','E4','F4','F#4','G4','G#4','A4',
    'A#4','B4','C5','C#5','D5','D#5','E5','F5','F#5','G5'})
    title(['Filtered Guitar Score, ...
        ',num2str(15*(k-1)),'-',num2str(15*k),' sec'],'Fontsize',16)
    xlabel('Time [s]'), ylabel('Note Name')
    set(gca,'Ylim',[130 785])
    set(gca,'Xlim',[(k-1)*30 30+(k-1)*30])
    set(gcf,'Position',[0,0,1700,600])
    saveas(gcf,['filtered_floyd_guitar_score_',num2str(k),'.png'])
    close all
end
```