# Rock & Roll and the Gabor Transform

# AMATH 482 HW2

**Xuanhui Chen**

## Abstract

This report aims to analyze two famous rock and roll songs fragments: *Sweet Child O' Mine* by Guns N' Roses and *Comfortably Numb* by Pink Floyd. The Gabor Transform (also known as STFT, the short-time Fourier Transform) is used heavily as a fundamental mathematical tool throughout this paper. The Gabor Transform is used to translate the given audio data into time and frequency domain. Then we can easily graph the spectrogram visualizing useful information and thus reproduce the guitar music score and bass music score for these two songs, respectively.

## I. Introduction and Overview

We are given two audio files of two well-know rock and roll songs *Sweet Child O'Mine* by Guns N' Roses and *Comfortably Numb* by Pink Floyd. The GNR riff and Floyd riff ranked #37 and #4 by *Guitar World,* and #8 and #2 by Louder Sound, respectively. Both audio files are read and represented with a vector of sampled data and the sample rate in Hertz. In order to reproduce the music scores of each clip, we need to translate the data in a balanced time and frequency domain, remove the overtones, and find each note.

We will use the Gabor Transform in this report to draw the spectrograms with appropriate $\alpha$ representing the width of the window and $\tau$ as the center of the window. Then we can obtain each note at the corresponding center and reproduce the music score with the provided music scale in Hertz.

## II. Theoretical Background

### Windowed Fourier Transform

We have used the Fourier Transform to analyze the frequency of signal. However, the limitation of the Fourier Transform is that it cannot deal with the signals that are changing over time, meaning it can only analyze stationary or periodic signals. The Fourier Transform can hardly tell the localized information of a signal because it loses information about what is happening in the time domain. It can only tell us the information regarding what frequencies are present instead of details of the signal at a certain time. Therefore, we now introduce the windowed Fourier Transform that splits up the time domain into subdomains, and then take the Fourier Transform on each subdomain, which leads to the new method: the Gabor Transform.

### Gabor Transform

The Gabor Transform (STFT) and its inverse are given by:

$$\widetilde{f_g}(\tau, k) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-ikt}\, dt, \quad (1)$$

$$f(t) = \frac{1}{2\pi\|g\|_2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \widetilde{f_g}(\tau, k)g(t-\tau)e^{ikt}\, dk\, d\tau, \quad (2)$$

respectively, where $\tau$ is the center of the filter, $g(t)$ is the filter function that can be chosen by considering the following assumption:

1. g(t) is real and symmetric.
2. $\|g\|_2 := \left(\int_{-\infty}^{\infty} |g(t)|^2\, dt\right)^{\frac{1}{2}} = 1.$

With the filter center $\tau$, we can apply the Fourier Transform by sweeping $\tau$ over the time domain. When $g(t)$ is specified as a Gaussian function: $g(t-\tau) = e^{-a(t-\tau)^2}$, the new parameter $a > 0$ denotes the width of the window. When a is huge, which means the window is huge, we have no information about the time but frequency. On the contrary, we have no frequency information when a is extremely small. They key point here is that we need to balance the portion of the analysis of time and frequency so that we can obtain information from both domains.

## <u>Spectrogram</u>

When writing a report, spectrogram is a good tool to represent information we get. In a spectrogram, the vertical axis is of value of the frequency, and the horizontal axis is of the window center $\tau$, where we put the Fourier Transforms next to each other. In this way, we can see the changing of Fourier Transforms by sweeping $\tau$ over the time domain. With appropriate a, we can adjust the information represented by the spectrogram.

## III. Algorithm Implementation and Development

For the first GNR clip:

- Loading the given clips and forming the initial conditions:
  - The audio is loaded. Sampling data and sampling rate are obtained. Initial conditions of time domains and frequency domains are set up. Plot of the original audio clip is drawn. (See **Appendix B** Part I of the first GNR file.)
- Use Gabor Transform to repeatedly apply the Fast Fourier Transform:
  - The window size $a$ is chosen. The Gabor Transform is used to apply the Fast Fourier Transform again and again by sweeping the filter center

    - $\tau$. The Gaussian function is used as the window function, and the overtones are filtered out. The note is obtained by looking for the peak of the center. Built-in functions *fft()* and *fftshift()* are used. (See **Appendix A** and **Appendix B** Part II of the GNR files.)
    - The normal or logarithm form of spectrograms are plotted with different window size a. The vertical direction represents the frequency in Hertz, while the horizontal axis denotes time. (See **Appendix B** Part II of the GNR file.)
- Draw the music score:
    - The music score is drawn. The left vertical axis represents the corresponding music notes, while the right hand side represents the frequency in Hertz. The horizontal axis represents time in second. (See **Appendix B** Part III of the GNR file.)

For the second Floyd clip:

- The progress is the same as the first clip, except for only using a quarter of the song because of too large computation (parts are repeated so it is fine). (See **Appendix B** Part I of the first Floyd file.)
- The same as what the first clip does. (See **Appendix B** Part II of the Floyd file.)
- Reset the initial conditions back to the full song and draw music scores: one for bass and one for guitar. Everything else is the same as the first clip but repeated twice. (See **Appendix B** Part III of the Floyd file.)

## IV. Computational Results

### 4.1 The GNR Clip

By choosing $a = 2000$ and $a = 0$, the normal spectrogram and the logarithm of the spectrogram is shown as Figure 1. Notice that $a = 2000$ has better time resolution, but the resolution in frequency space is not clear. For $a = 0$, we just have the Fourier Transform (FFT), which has no localization in time, but the frequency resolution is better.

The music score of the GNR clip is shown as Figure 2, with the left vertical axis representing the corresponding notes, the right vertical axis denoting the frequency, and the horizontal axis as the time step. The frequency is within the range of 250-450 Hertz.

### 4.2 The Floyd Clip

By choosing $a = 200$ and $a = 0.5$, the normal spectrogram and the logarithm of the spectrogram is shown as Figure 3. The logarithm plot provides a more obvious comparison. The result is similar to what we get for the first clip.

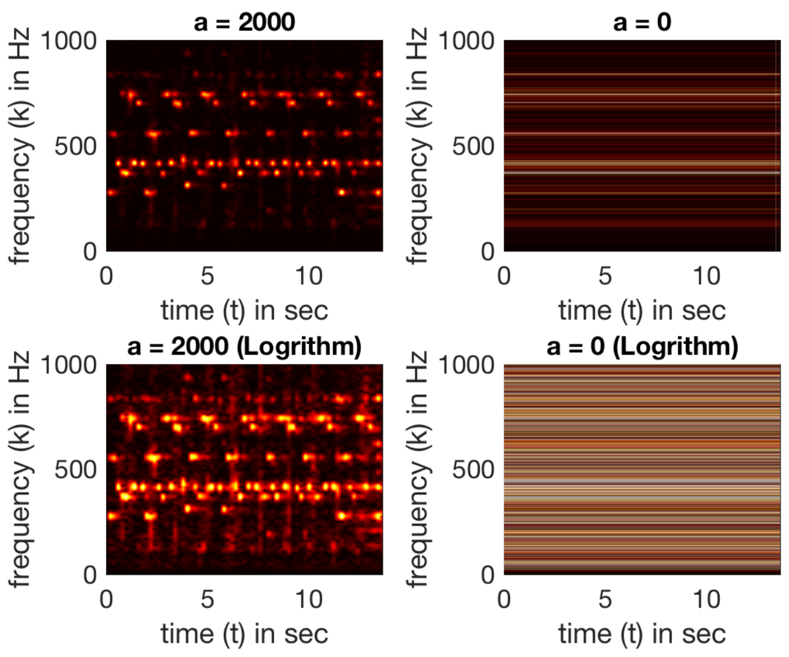

Figure 1: Spectrogram of the GNR clip in both normal form and logarithm form.

The bass and guitar score of the GNR clip is shown as Figure 4, respectively, with the same setting as the one of the first clip. The frequency for the bass score is within the range of 50-250 Hertz, while the one for the guitar score is within the range of 250-700 Hertz.

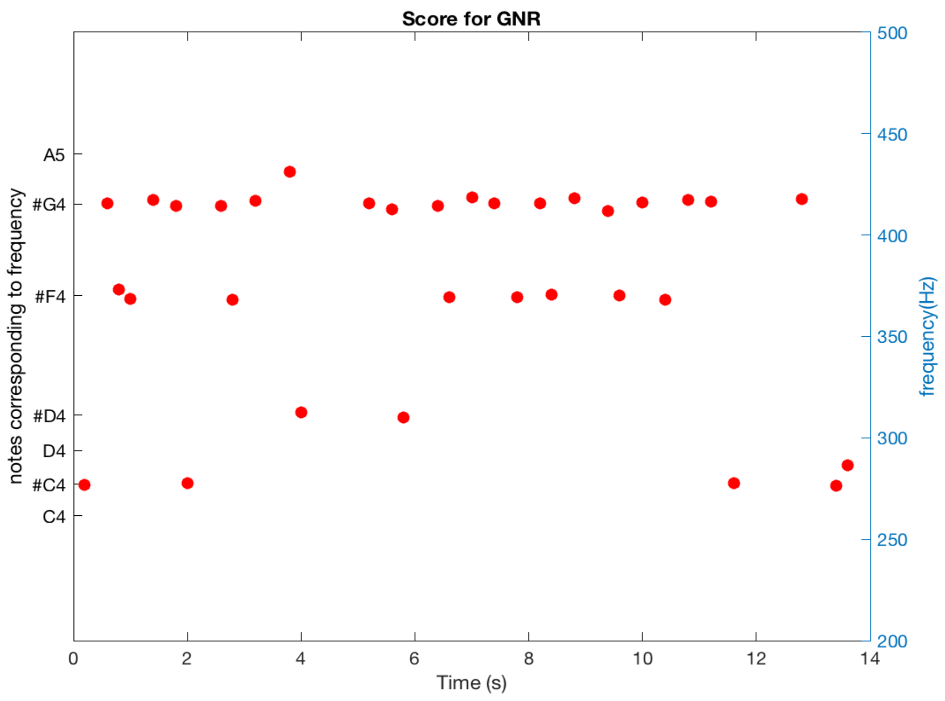

Figure 2: Music score for the GNR clip

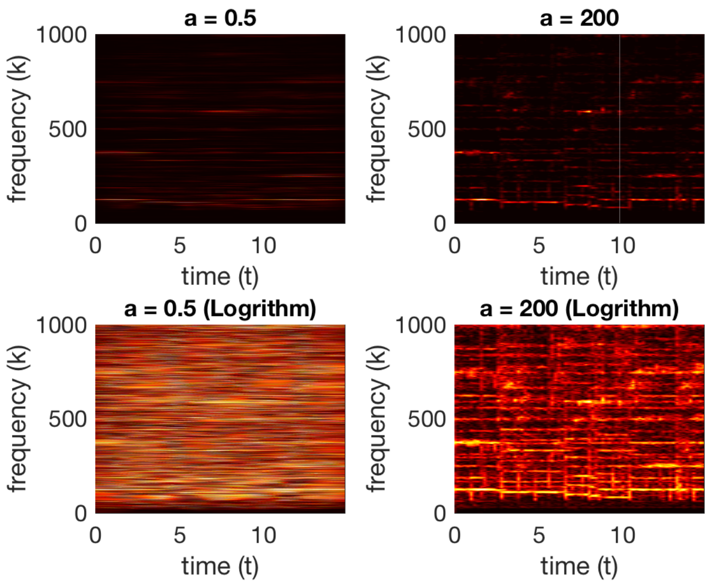

Figure 3: Spectrogram of the Floyd clip in both normal form and logarithm form.

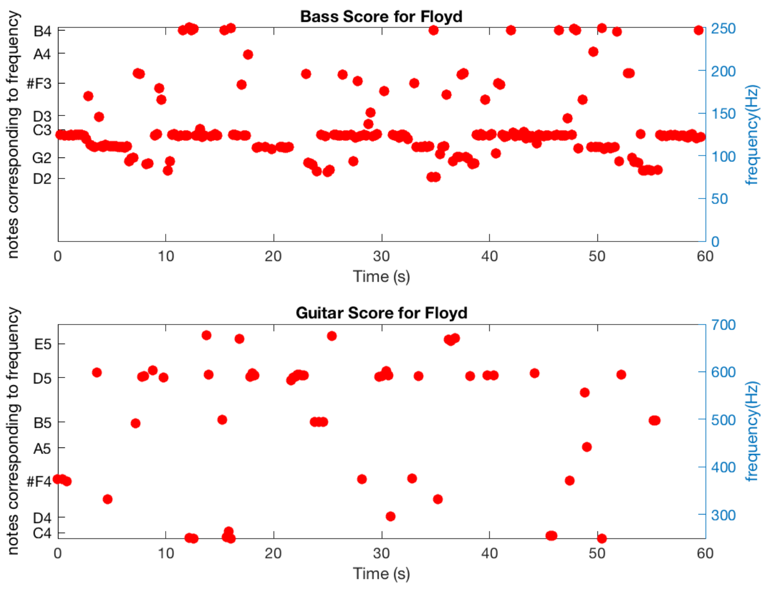

Figure 4: Bass score for the Floyd clip

## V. Summary and Conclusion

To analyze audio clips with signals changing in time, the Gabor Transform is a strong approach other than the original Fast Fourier Transform. By choosing appropriate window size, we are able to balance the information we want from both frequency and time domain. To reproduce the music score, we need to filter out the overtones, and find the peak at the center when applying the FFT with different centers of filter.

# Appendix A: MATLAB functions used and brief implementation explanation

- **linspace(X1, X2, N)**: generate N points between X1 and X2.
- **[Y, FS] = audioread(FILENAME):** reads an audio file specified by the character vector or string scalar FILENAME, returning the sampled data in Y and the sample rate FS, in Hertz.
- **fftshift(X)**: shift zero-frequency component to center of spectrum.
- **pcolor(X,Y,C)**: X and Y are vectors or matrices, makes a pseudocolor plot on the grid defined by X and Y.
- **colormap(MAP)**: sets the current figure's colormap to MAP.
- **B = repmat(A,M,N):** creates a large matrix B consisting of an M-by-N tiling of copies of A.
- **yyaxis RIGHT**: activates the side associated with the right y-axis.
- **fft(X)**: the discrete Fourier transform (DFT) of vector X.  For matrices, the fft operation is applied to each column.

# Appendix A: MATLAB Codes

```
%clear; clc;

% Part I: Initial Condition for GNR
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');
print(gcf,'-dpng','scom.png');
%p8 = audioplayer(y,Fs); playblocking(p8);

n = length(y);
L = n/Fs;
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);

S = y';
St = fft(S);

% Part II: Use Gabor Transform to repeat the FFT and draw the spectrogram
figure(2)
a = 2000;
tau = 0:0.2:tr_gnr;
Sgt_spec = [];
% the center of filter changes everytime
for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2); % Window function
    Sg = g.*S;
    Sgt = fft(Sg);
    [M,I] = max(Sgt);
    notes_gnr(1,j) = abs(k(I));
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end

% Spectrogram for the larger a
subplot(2,2,1)
pcolor(tau,ks,Sgt_spec.')
shading interp
set(gca,'ylim',[0 1000],'Fontsize',16)
colormap(hot) %colorbar
xlabel('time (t) in sec'), ylabel('frequency (k) in Hz')
title(['a = ',num2str(a)],'Fontsize',16)

% Logrithm spectrogram for the larger a
subplot(2,2,3)
pcolor(tau,ks,(log(abs(Sgt_spec)+1)).')
shading interp
set(gca,'ylim',[0 1000],'Fontsize',16)
colormap(hot) %colorbar
xlabel('time (t) in sec'), ylabel('frequency (k) in Hz')
title(['a = ', num2str(a), ' (Logrithm)'],'Fontsize',16)


% Add in the Fourier transform of the whole signal
```

```
%Sgt_spec_l = [];
%Sgt_spec_l = repmat(fftshift(log(abs(St)+1)),length(tau),1);
Sgt_spec = [];
Sgt_spec = repmat(fftshift(abs(St)),length(tau),1);

% Spectrogram for a = 0
subplot(2,2,2)
pcolor(tau,ks,Sgt_spec.'),
shading interp
set(gca,'ylim',[0 1000],'Fontsize',16)
colormap(hot) %colorbar
xlabel('time (t) in sec'), ylabel('frequency (k) in Hz')
title('a = 0','Fontsize',16)

% Logrithm spectrogram for a = 0
subplot(2,2,4)
pcolor(tau,ks,(log(abs(Sgt_spec)+1)).'),
shading interp
set(gca,'ylim',[0 1000],'Fontsize',16)
colormap(hot) %colorbar
xlabel('time (t) in sec'), ylabel('frequency (k) in Hz')
title('a = 0 (Logrithm)','Fontsize',16)

print(gcf,'-dpng','gnr_spec.png');


% Part III: Music score for the GNR
figure(3)
notes_vec = [261.63,277.18,293.66,311.13,369.99,415.3,440];
plot(tau, notes_gnr,'ro','MarkerFaceColor', 'r');
yticks(notes_vec);
yticklabels({'C4','#C4','D4','#D4','#F4','#G4','A5'});
ylim ([200 500])
title("Score for GNR");
xlabel("Time (s)");
ylabel("notes corresponding to frequency");
yyaxis right;
plot(tau, notes_gnr,'ro','MarkerFaceColor', 'r');
ylabel("frequency(Hz)");
ylim ([200 500])
print(gcf,'-dpng','gnr_notes.png');
```

```
clear; clc;

% Floyd
figure(1)
[y, Fs] = audioread('Floyd.m4a');
tr_flo = length(y)/Fs; % record time in seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Comfortably Numb');
print(gcf,'-dpng','cn.png');
%p8 = audioplayer(y,Fs); playblocking(p8);

% Part I: Initial conditions for drawing 1/4 fragment of spectrogram
tr_flo_s = tr_flo/4;
n_s = length(y)/4;
t2 = linspace(0,tr_flo_s,n_s+1);
t_1 = t2(1:n_s);
k = (1/tr_flo_s)*[0:n_s/2-1 -n_s/2:-1];
ks = fftshift(k);

S = y(1:n_s).';
St = fft(S);

% Part II: Use Gabor Transform to repeat the FFT and draw the spectrogram
figure(2)
a = [0.5 200];
tau = 0:0.2:tr_flo_s;
for i = 1:length(a)
    Sgt_spec = []; % Clear at each loop iteration
    for j = 1:length(tau)
        g = exp(-a(i)*(t_1 - tau(j)).^2); % Window function
        Sg = g.*S;
        Sgt = fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end
    subplot(2,2,i)
    pcolor(tau,ks,Sgt_spec.')
    shading interp
    set(gca,'ylim',[0 1000],'Fontsize',16)
    colormap(hot) %colorbar
    xlabel('time (t) in sec'), ylabel('frequency (k) in Hz')
    title(['a = ',num2str(a(i))],'Fontsize',16)

    subplot(2,2,i+2)
    pcolor(tau,ks,(log(abs(Sgt_spec)+1)).')
    shading interp
    set(gca,'ylim',[0 1000],'Fontsize',16)
    colormap(hot) %colorbar
    xlabel('time (t) in sec'), ylabel('frequency (k) in Hz')
    title(['a = ',num2str(a(i)),' (Logrithm)' ],'Fontsize',16)
end


% Add in the Fourier transform of the whole signal
%Sgt_spec = [];
%Sgt_spec = repmat(fftshift(log(abs(Sgt)+1)),length(tau),1);
%subplot(2,1,2)
%pcolor(tau,ks,Sgt_spec.'),
%shading interp
%set(gca,'ylim',[0 1000],'Fontsize',16)
%colormap(hot) %colorbar
%xlabel('time (t)'), ylabel('frequency (k)')
%title('a = 0','Fontsize',16)

print(gcf,'-dpng','floyd_spec.png');

% Part III: music score for floyd

% a. Reset the initial conditions
n = length(y);
L = n/Fs;
t2 = linspace(0,tr_flo,n+1);
t = t2(1:n);
k = (1/tr_flo)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

S = y';
St = fft(S);

% b. Find each note by searching for the peak of the filter center:
a = [0.2 1500];
tau = 0:0.2:tr_flo;

for i = 1:length(a)
    Sgt_spec = []; % Clear at each loop iteration
    for j = 1:length(tau)
        g = exp(-a(i)*(t - tau(j)).^2); % Window function
        Sg = g.*S;
        Sgt = fft(Sg);
        [M,I] = max(Sgt);
        notes_flo(1,j) = abs(k(I));
    end
end

% c. The bass score
figure(3)
subplot(2,1,1)
notes_vec = [73.416,97.999,130.81,146.83,185,220,246.94];
plot(tau, notes_flo,'ro','MarkerFaceColor', 'r');
yticks(notes_vec);
yticklabels({'D2','G2','C3','D3','#F3','A4','B4'});
ylim ([0 250])
title("Bass Score for Floyd");
xlabel("Time (s)");
ylabel("notes corresponding to frequency");
yyaxis right;
plot(tau, notes_flo,'ro','MarkerFaceColor', 'r');
ylabel("frequency(Hz)");
ylim ([0 250])
print(gcf,'-dpng','floyd_bass_notes.png');

% d. The guitar score:
subplot(2,1,2)
notes_vec = [261.63,293.66,369.99,440,493.88,587.33,659.26];
plot(tau, notes_flo,'ro','MarkerFaceColor', 'r');
yticks(notes_vec);
```

```
yticklabels({'C4','D4','#F4','A5','B5','D5','E5'});
ylim ([250 700])
title("Guitar Score for Floyd");
xlabel("Time (s)");
ylabel("notes corresponding to frequency");
yyaxis right;
plot(tau, notes_flo,'ro','MarkerFaceColor', 'r');
ylabel("frequency(Hz)");
ylim ([250 700])
print(gcf,'-dpng','floyd_guitar_notes.png');
```

## Reference

1. Course note.
2. MATLAB help section.