Rock & Roll and the Gabor Transform

Paul Abboud

***Abstract*—We are tasked with analyzing a portion of two rock & rolls songs. The music clips come from Sweet Child O' Mine by Guns N' Roses and Comfortably Numb by Pink Floyd. We will employ a Gabor transform and spectrogram to determine the score of the guitar in Sweet Child O' Mine and the score of the bass in Comfortably Numb. Then we will isolate the bass in Comfortably Numb through a Gaussian filter and attempt to reconstruct the guitar solo.**

I. Introduction

In this application of signal processing, we read in two audio files which contain the amplitude of signals over time. To determine the target frequency at each time interval, or the note being played, we utilize the Gabor transform to filter the signal in windows where we can isolate the dominant signal. We then apply a Fourier transform to the filtered signal. Each Fourier transform gives us insight about the frequency produced during that specific window of time. The collection of Fourier transforms are stored in a matrix which is then used to produce a spectrogram and determine the score of the guitar in Sweet Child O' Mine and the bass in Comfortably Numb. We can then apply another filter in the frequency space to isolate the bass frequencies in Comfortably Numb and then reproduce the guitar score.

II. Theoretical Background

To understand the issue at hand, the following will discuss the key ideas behind implementing the Fourier transform, the Gabor transform, Gaussian Filtering and reading spectrograms.

A. Fourier Transform

The first essential aspect of processing the provided signal data is applying the Fourier transform. The Fourier transform is denoted by $\hat{f}(k)$, taking in the function $f(x)$.

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

The Fourier transform in Eq.(1) inputs a function of space or time, x, and transforms it into a function of frequencies, k. To revert the transform from a frequency domain to a space or time domain, we can apply the inverse Fourier transform defined as $f(x)$.

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

In this application, the scale for frequencies of musical notes is Hertz, so we scale the frequencies by $\frac{1}{L}$ where L is the length of computational domain. We will compute the Fast Fourier transform of the signal at each note iteration after applying the Gabor transform.

B. Gabor Transform

Since our data is sampled at discrete time intervals, we will utilize the discrete Gabor transform described by:

$$\tilde{f}_g(m,n) = \int_{-\infty}^{\infty} f(t) g(t - nt_0) e^{2\pi i m \omega_0 t} dt \tag{3}$$

Where nt_0 is the time interval shift. The filter $g(t - nt_0)$ isolates our signal, $f(t)$, around the specified time interval so we can focus on the frequency occurring during that window of time. After breaking our signal into a number of windows by applying the Gabor transform, we can determine all the relevant frequencies by iterating through every interval.

C. Gaussian Filtering

After computing the target frequencies we can filter noise around them by applying a Gaussian Filter. The Gaussian is centered around the frequency k_0 and minimizes values around it by multiplying the filter and frequency data

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

where τ determines the width of the filter. Applying the Gaussian Filter will isolate the desired frequency in each time interval resulting in a more accurate reproduction of the frequencies as time progresses.

D. Spectrograms

Spectrograms help us decipher frequency data as it progresses with time. The strength of a frequency (amplitude) determines the color of the "imprint" on the spectrogram. The frequency data is recorded for each time interval and then reproduced on the spectrogram making it a useful tool to understand how amplitude, frequency and time relate.

III. Algorithm Implementation

Firstly, we need to establish the problem space by loading in the signal data for the music files and computing the computational domain.

```
[y, Fs] = audioread('GNR.m4a');
v = y';
p8 = audioplayer(y,Fs);

L = length(v)/Fs;
n = length(v);
t2 = linspace(0,L,n+1);
t = t2(1:n);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);
```

We can now implement the Gabor transform by multiplying the filter by our signal data then Fourier transforming the product to the frequency domain. We use a time interval of 0.2 seconds to capture all the notes being played for the guitar in Sweet Child O' Mine and a time interval of 1 second for the bass in Comfortably Numb since notes aren't played as frequently. We can clean up the data by only extracting the maximum frequency at each interval. These maximum values represent the notes being played. Finally, we store the complete Fourier transformed data into a matrix to later be processed into a spectrogram.

```
tau = 0:0.2:L;
a = 100;
Vgt_spec=[];

for j = 1:length(tau)
    g = exp(-a * (t-tau(j)).^2);
    Vg = g.*v;
    Vgt = fft(Vg);
    [M, I] = max(fftshift(abs(Vgt)));
    notes(j) = abs(ks(I));
    Vgt_spec(:,j)= abs(fftshift(Vgt));
end
```

To isolate the bass in Comfortably Numb, we can filter around the notes we found above. By multiplying the Fourier transformed signal space by the Gaussian filter at each note, we minimize noise around the bass notes.

```
for j = 1:length(tau)
    g = exp(-a * (t-tau(j)).^2);
    Vg = g.*v;
    Vgt = fft(Vg);
    [M, I] = max(fftshift(abs(Vgt)));
    notes(j) = abs(ks(I));
    g2 = exp(-a2 * (ks-notes(j)).^2);
    VgtF = Vgt.*fftshift(g2);
    Vgt_spec(:,j)= abs(fftshift(VgtF))
        ;
end
```

We can implement the same technique as above to determine the guitar score of Comfortably Numb after removing bass notes and then applying a filter around the adjusted maximum frequencies found.

IV. Computational Results

Plotting the spectrogram of Sweet Child O' Mine, in Fig. 1, notice the prevalence of distinct marks which represent the frequencies of the notes being played by the guitar. The spectrogram in Fig. 1 can be transcribed into a score which is illustrated in Fig. 2. Then we applied a Gabor transform to Comfortably Numb and the resulting spectrogram in Fig. 3 was produced. Again, this can be transcribed into a score for the bass in Comfortably Numb shown in Fig. 4. Isolating the bass after filtering around the relevant frequencies produced a cleaner and more succinct spectrogram illustrated in Fig. 5.

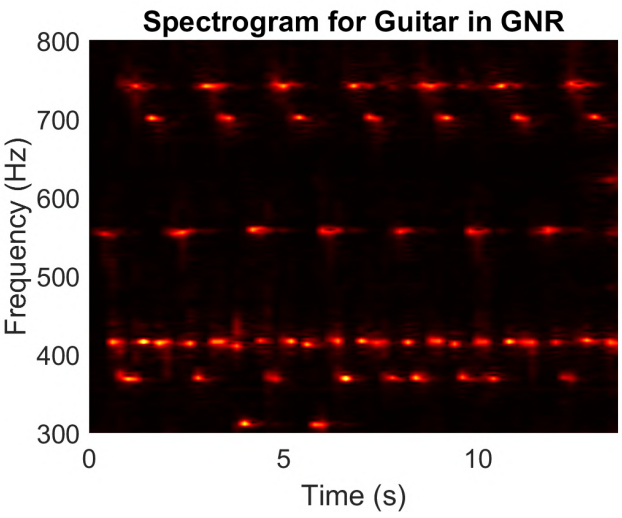

Fig. 1. Spectrogram of Sweet Child O' Mine Guitar

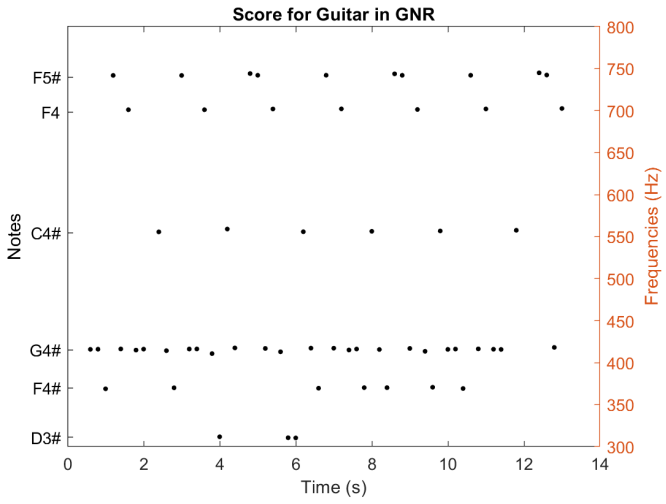

Fig. 2. Score of Sweet Child O' Mine Guitar

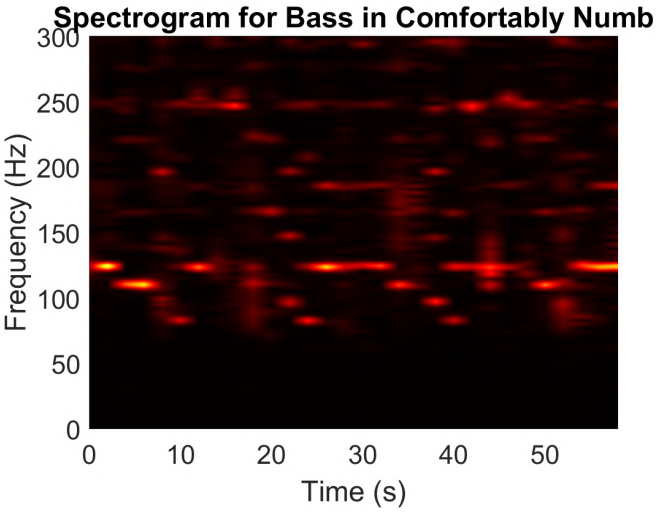

Fig. 3. Spectrogram of Comfortably Numb Bass

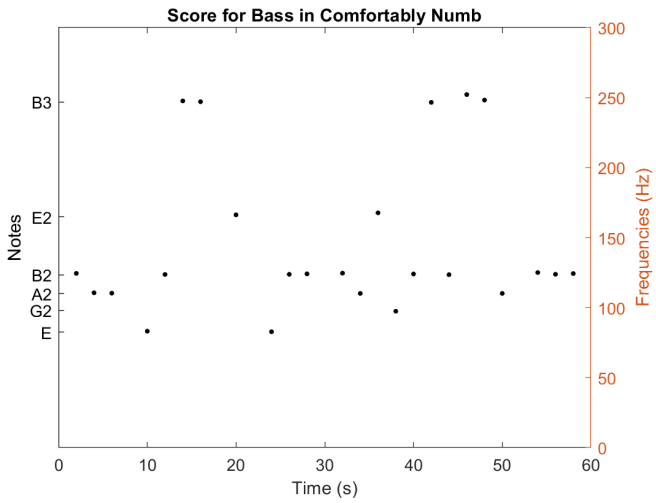

Fig. 4. Score of Comfortably Numb Bass

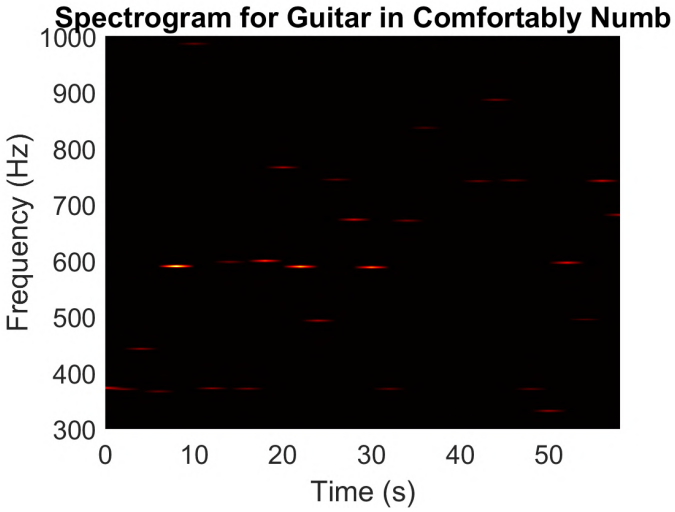

Fig. 6. Filtered Spectrogram of Comfortably Numb Guitar

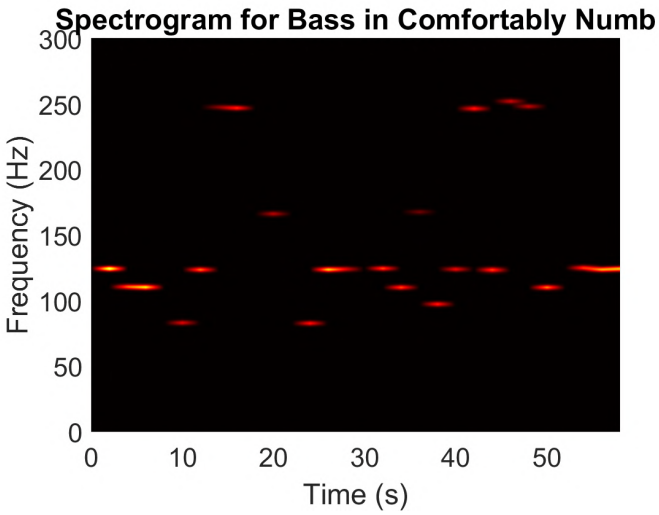

Fig. 5. Filtered Spectrogram of Comfortably Numb Bass

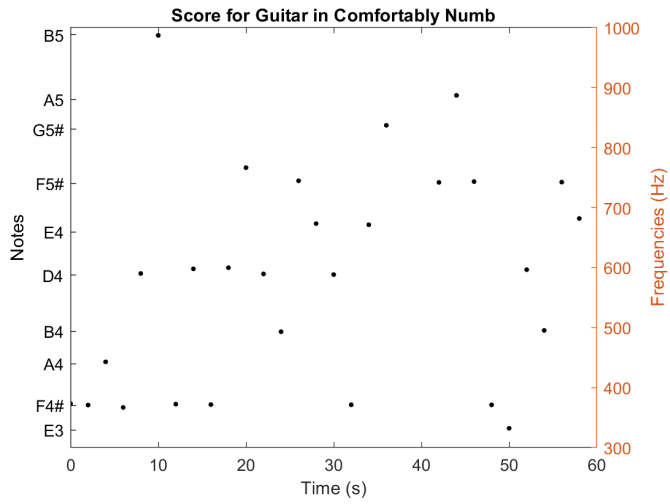

Fig. 7. Score of Comfortably Numb Guitar

Finally, we removed the bass signals from the audio and then filtered around the relevant frequencies to isolate the guitar in Comfortably Numb, Fig. 6 shows the resulting spectrogram. Fig. 7 shows the score produced from the spectrogram in Fig. 6.

V. Summary of Results

In this assignment we were tasked with reproducing the scores for Sweet Child O' Mine and Comfortably Numb. We successfully implemented the Gabor transform to develop spectrograms for both recordings. Isolating the signals in time intervals then recording the Fourier transformed frequency data allowed us to then plot the scores of both audio files. Using a Gaussian filter in the frequency domain, we improved the spectrogram for the bass in Comfortably Numb. Applying the same technique after removing bass signals produced the spectrogram in Fig. 6, which can be interpreted as a score for the guitar in Fig. 7.

Appendix A

MATLAB Functions

fft: Using the Fast Fourier Transform algorithm, returns the Fourier transformed function.

fftshift: Rearranges the Fourier transformed data by shifting the zero frequency to the center.

linspace: Creates a discrete interval from a specified interval and number of points.

ifftshift: Reverses the shift in values produced by fftshift, reverting it to matlabs default order.

ifft: Reverses a Fourier transformation, sending the function back to a space domain.

audioread: Reads an audio file a returns the sampled data points and the sampling rate of those data points.

max: Returns the maximum value and the index that maximum value was found at.

pcolor: Plots x and y coordinates and takes in a matrix which specifies the coloring of the coordinates.

Appendix B

MATLAB Code

```
%% GNR Guitar
close all; clear all; clc;
[y, Fs] = audioread('GNR.m4a');
v = y';

L = length(v)/Fs;
n = length(v);
t2 = linspace(0,L,n+1);
t = t2(1:n);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

tau = 0:0.2:L;
a = 100;
Vgt_spec=[];

for j = 1:length(tau)
    g = exp(-a * (t-tau(j)).^2);
    Vg = g.*v;
    Vgt = fft(Vg);
    [M, I] = max(fftshift(abs(Vgt)));
    notes(j) = abs(ks(I));
    Vgt_spec(:,j)= abs(fftshift(Vgt));
end

figure(1);
pcolor(tau, ks, Vgt_spec), shading
    interp
set(gca,'ylim',[300 800],'Fontsize'
    ,16)
xlabel(['Time (s)'])
ylabel(['Frequency (Hz)'])
title('Spectrogram for Guitar in GNR')
    ;
colormap(hot)
figure(2);
plot(tau, notes, 'k.','Markersize',10)
    ;
yticks([311.1, 370, 415.3,
    554.4,698.5, 740]);
yticklabels({'D3#','F4#','G4#','C4#','
    F4','F5#'});
ylim([300 800])
ylabel('Notes')
yyaxis right
ylabel('Frequencies (Hz)')
ylim([300 800])
title('Score for Guitar in GNR');
xlabel('Time (s)');

%% Floyd Bass Unfiltered
close all; clear all; clc;
[y, Fs] = audioread('Floyd.m4a');
v = y';

L = length(v)/Fs;
n = length(v);
t2 = linspace(0,L,n+1);
t = t2(1:n);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

t = t(1:n-1);
v = v(1:n-1);

tau = 0:2:L;
a = 100;
a2 = 0.2;
Vgt_spec=[];

for j = 1:length(tau)
    g = exp(-a * (t-tau(j)).^2);
    Vg = g.*v;
    Vgt = fft(Vg);
    [M, I] = max(fftshift(abs(Vgt)));
    notes(j) = abs(ks(I));
    Vgt_spec(:,j)= abs(fftshift(Vgt));
end

figure(3);
pcolor(tau, ks, Vgt_spec), shading
    interp
set(gca,'ylim',[0 300],'Fontsize',16)
xlabel(['Time (s)'])
ylabel(['Frequency (Hz)'])
title("Spectrogram for Bass in
    Comfortably Numb");
colormap(hot)
figure(4);
plot(tau, notes, 'k.','Markersize',10)
    ;
yticks([82.41, 98, 110, 123.5, 164.8,
    246.9]);
yticklabels({'E','G2','A2','B2','E2','
    B3'});
ylabel('Notes')
```

```
ylim([0 300])
yyaxis right
ylabel('Frequencies (Hz)')
ylim([0 300])
title('Score for Bass in Comfortably Numb');
xlabel('Time (s)');

%% Floyd Bass Filtered
close all; clear all; clc;
[y, Fs] = audioread('Floyd.m4a');
v = y';

L = length(v)/Fs;
n = length(v);
t2 = linspace(0,L,n+1);
t = t2(1:n);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

tau = 0:2:L;
a = 100;
a2 = 0.2;
Vgt_spec=[];

t = t(1:n-1);
v = v(1:n-1);

for j = 1:length(tau)
    g = exp(-a * (t-tau(j)).^2);
    Vg = g.*v;
    Vgt = fft(Vg);
    [M, I] = max(fftshift(abs(Vgt)));
    notes(j) = abs(ks(I));
    g2 = exp(-a2 * (ks-notes(j)).^2);
    VgtF = Vgt.*fftshift(g2);
    Vgt_spec(:,j)= abs(fftshift(VgtF));
end

figure(5);
pcolor(tau, ks, Vgt_spec), shading interp
set(gca,'ylim',[0 300],'Fontsize',16)
xlabel(['Time (s)'])
ylabel(['Frequency (Hz)'])
title("Spectrogram for Bass in Comfortably Numb");
colormap(hot)
figure(6);
plot(tau, notes, 'k.','Markersize',10);
yticks([82.41, 98, 110, 123.5, 164.8, 246.9]);
yticklabels({'E','G2','A2','B2','E2','B3'});
ylabel('Notes')
ylim([0 300])
yyaxis right
ylabel('Frequencies (Hz)')
ylim([0 300])
title('Score for Bass in Comfortably Numb');
xlabel('Time (s)');

%% Floyd Guitar Filtered

close all; clear all; clc;
[y, Fs] = audioread('Floyd.m4a');
v = y';

L = length(v)/Fs;
n = length(v);
t2 = linspace(0,L,n+1);
t = t2(1:n);
k = (1/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

vG = fftshift(fft(v));
for j = 1:length(ks)
    if abs(ks(j)) < 260
        vG(j) = 0;
    end
end
v = ifft(ifftshift(vG));

t = t(1:n-1);
v = v(1:n-1);

tau = 0:2:L;
a = 100;
a2 = 0.2;
Vgt_spec=[];

for j = 1:length(tau)
    g = exp(-a * (t-tau(j)).^2);
    Vg = g.*v;
    Vgt = fft(Vg);
    [M, I] = max(fftshift(abs(Vgt)));
    notes(j) = abs(ks(I));
    g2 = exp(-a2 * (ks-notes(j)).^2);
    VgtF = Vgt.*fftshift(g2);
    Vgt_spec(:,j)= abs(fftshift(VgtF));
end

figure(7);
pcolor(tau, ks, Vgt_spec), shading interp
set(gca,'ylim',[300 1000],'Fontsize',16)
xlabel(['Time (s)'])
```

```
187 ylabel(['Frequency (Hz)'])
188 title('Spectrogram for Guitar in
        Comfortably Numb');
189 colormap(hot)
190 figure(8);
191 plot(tau, notes, 'k.','Markersize',10)
        ;
192 yticks([293.7, 329.6, 370, 440, 493.9,
         587.3, 659.3, 740, 830.6, 880,
        987.8]);
193 yticklabels({'D3','E3','F4#','A4','B4'
        ,'D4','E4','F5#','G5#','A5','B5'});
194 ylim([300 1000])
195 ylabel('Notes')
196 yyaxis right
197 ylabel('Frequencies (Hz)')
198 ylim([300 1000])
199 title('Score for Guitar in Comfortably
         Numb');
200 xlabel('Time (s)');
```