Rock & Roll and the Gabor transform

Yumin Yin

February 10, 2021

Abstract

We will analyze two audio signals, *Sweet Child O' Mine* and *Comfortably Numb*, by applying the Gabor transform. Through the use of the Gabor filtering, we are able to reproduce the music score for the guitar in *Sweet Child O' Mine*. Using a filter in frequency space, we first isolate the bass in Comfortably Numb, and then we are able to put the guitar solo together in *Comfortably Numb*.

1 Introduction and Overview

We will analyze two of the greatest rock and roll songs of all time, *Sweet Child O' Mine* and *Comfortably Numb*. There is only guitar played in *Sweet Child O' Mine*, so we can reproduce the music score for the guitar in it through the use of the Gabor filtering. It is more complicated for *Comfortably Numb*: both guitar and bass are played in it. We first use a band-pass filter in frequency domain to only examine the frequencies of music notes that are inside the bass frequency range, 60-250 Hertz, so that we can isolate the bass. Then we focus on the frequencies of music notes that are inside the guitar frequency range, 200-500, and filter out the bass notes and its overtones. We are now able to put much of the guitar solo together in *Comfortably Numb*.

2 Theoretical Background

2.1 Heisenberg Uncertainty Principle

The drawback of applying the Fourier transform to analyze the frequency of a signal is that the shift invariance of the absolute value of the Fourier transform loses information about what is happening in the time domain. Particularly, we lose *when* certain frequencies occur or how the frequencies change over time. This is the Heisenberg Uncertainty Principle in action: we cannot have all the information about both time and frequency. Our task is to strike a balance so that we can have some information about frequency and some information about time.

2.2 The Gabor Transform

The *Gabor transform* is given precisely by equation (1)

$$\tilde{f}_g(\tau, k) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-ikt}dt. \tag{1}$$

For a fixed τ, the function $\tilde{f}_g(\tau, k)$ gives the information about the frequency components near time τ. It is noted that the results are dependent on the choice of filter $g(t)$. When Gabor was developing this method he used a Gaussian, so a Gaussian will be our default choice as well.

2.3 The Inverse of The Gabor Transform

The inverse of the Gabor transform is given by equation (2)

$$f(t) = \frac{1}{2\pi||g||_2} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{f}_g(\tau, k)g(t-\tau)e^{ikt}dt. \tag{2}$$

2.4 Width of the Gabor Window

After specify $g(t)$ to be a Gaussian function, we still have another parameter: the width of the window. If the window is huge, then we just recover the Fourier transform over the signal, which has all the frequency information, but no information about time. If the window is extremely small, then we are almost look at individual times along the signal, so there would be no frequency information. Our task is to find a window, which is not too big or too small, so that the Gabor transform returns some information about time and some information about frequency.

2.5 Discrete Gabor Transform

As pointed out with Fourier transform, if we actually want to use the Gabor transform on data, we need a discrete version. We also have to consider a discrete set of frequencies: $k = mw_0$ and $\tau = nt_0$, where

m and n are integers and w_0 and t_0 are positive constants (the frequency resolutions). The discrete Gabor transform is by equation (3)

$$\tilde{f}_g(m,n) = \int_{-\infty}^{\infty} f(t)g(t-t_0)e^{2\pi i m w_0 t}dt. \tag{3}$$

2.6 Hertz Versus Angular Frequency

MATLAB scales out the 2π in the periods of oscillation to make the periods integers. That is Eq.(4),

$$e^{ikt}, k \in \mathbb{Z}. \tag{4}$$

has frequency (inverse of the period) $|k|/2\pi$, while Eq.(5)

$$e^{i2\pi ft}, f \in \mathbb{Z}. \tag{5}$$

has frequency $|f|$. MATLAB is using the latter, while we have been using the former. They both represent frequencies, but they have different units. With the notation above, f is measured in Hertz, while k is sometimes called the angular frequency.

2.7 Parameters

We will have two parameters in the Gabor transform implementation, $a > 0$ and τ, representing the width of the window and the center of the window, respectively. Notice that a small a means a wide window and large a means a thin window.

3 Algorithm Implementation and Development

3.1 the Guitar in *Sweet Child O' Mine*

The frequency domain and time domain are properly constructed based on the given audio signal, *Sweet Child O' Mine*. The scale used for the frequencies of the musical notes is Hertz, so we scaled k vector by $1/L$. At each center of the window, τ, we multiplied the signal by a (discrete) window function and apply the Fourier transform (FFT). We then slide the window over the time domain and repeat the process. In this case, we set $\tau = 0.1$ and use the default filter function, the Gaussian function with $a = 1000$ so that the resulting spectrum tells us both information about time and frequency. We only need to look at frequencies below 1000 Hertz because frequencies higher than this limit are just overtones.

3.2 the Bass in *Comfortably Numb*

The frequency domain and time domain are properly constructed based on the given audio signal, *Comfortably Numb*. The scale used for the frequencies of the musical notes is Hertz, so we scaled k vector by $1/L$. We first use a band-pass filter in frequency domain to only examine the frequencies of music notes that are inside the bass frequency range, 60-250 Hertz. Then, at each center of the window, τ, we multiplied the signal by a (discrete) window function and apply the Fourier transform (FFT). After that, we slide the window over the time domain and repeat the process. In this case, we set $\tau = 2.5$ due to the computational constraints and use the default filter function, the Gaussian function with $a = 10$ so that the resulting spectrum tells us both information about time and frequency. By looking at the spectrum of entire song, we notice that the music notes are repeated about every ten seconds. Therefore, we can build a spectrum of the first ten seconds in *Comfortably Numb*, applying the same Gabor transform implementation but with $\tau = 0.25$ to get a better idea of which music notes are being played.

3.3 the Guitar in *Comfortably Numb*

The frequency domain and time domain are properly constructed based on the given audio signal, *Comfortably Numb*. The scale used for the frequencies of the musical notes is Hertz, so we scaled k vector by $1/L$. We

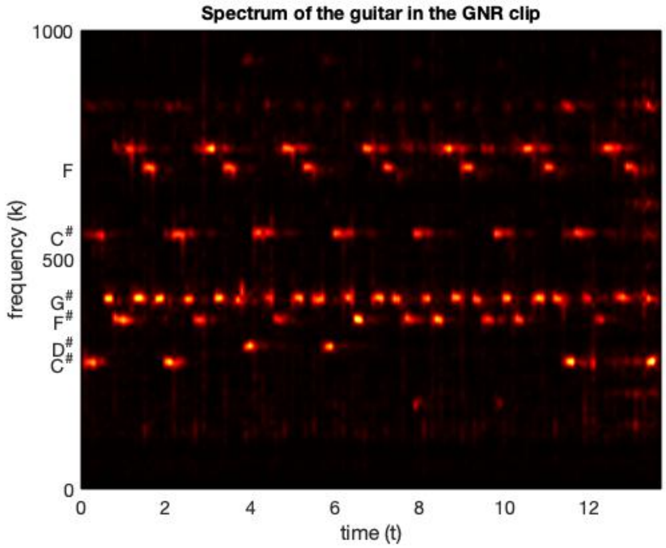

Figure 1: Spectrum of the guitar in the GNR clip

first use a band-pass filter in frequency domain to only examine the frequencies of music notes that are inside the guitar frequency range, 200-500 Hertz. Then, at each center of the window, τ, we multiplied the signal by a (discrete) window function and apply the Fourier transform (FFT). After that, we slide the window over the time domain and repeat the process. In this case, we set $\tau = 2.5$ due to the computational constraints and use the default filter function, the Gaussian function with $a = 10$ so that the resulting spectrum tells us both information about time and frequency. By looking at the spectrum of entire song, we notice that the music notes are repeated about every ten seconds. Therefore, we can build a spectrum of the first ten seconds in *Comfortably Numb*, applying the same Gabor transform implementation but with $\tau = 0.5$ to get a better idea of which music notes are being played. There might be still some bass notes in this frequency range, but we have already identified these notes with their overtones, which are the integer multiplies of the fundamental frequencies, we are still able to figure out which music notes are being played by the guitar.

4 Computational Results

4.1 the Guitar in *Sweet Child O' Mine*

The figure (1) is the spectrum of the guitar in *Sweet Child O' Mine*. We can read off the frequency of each spot from the y-axis of the spectrum and convert it into the corresponding music note. After that, we read off the time of each music note being played by the guitar from the x-axis to put them in chronological order. The music score derived from the spectrum for the guitar in *Sweet Child O' Mine* is:
$C\# - C\# - G\# - F\# - F\# - G\# - F - G\# - C\# - C\# - G\# - F\# - F\# - G\# - F - G\#$
$D\# - C\# - G\# - F\# - F\# - G\# - F - G\# - D\# - C\# - G\# - F\# - F\# - G\# - F - G\#$
$F\# - C\# - G\# - F\# - F\# - G\# - F - G\# - F\# - C\# - G\# - F\# - F\# - G\# - F - G\# - C\#$

4.2 the Bass in *Comfortably Numb*

The figure (2) is the entire spectrum of the bass in *Comfortably Numb*. Clearly, it indicates the music notes are repeated about every ten seconds. The figure (3) is the spectrum of the 10s bass in *Comfortably Numb*. We can read off the frequency of each spot from the y-axis of the spectrum and convert it into the corresponding music note. After that, we read off the time of each music note being played by the bass from the x-axis to put them in chronological order. The music score derived from the spectrum for the bass in

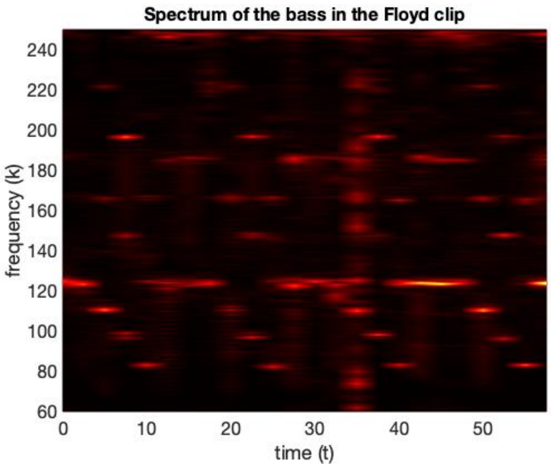

Figure 2: Spectrum of the bass in the Floyd clip

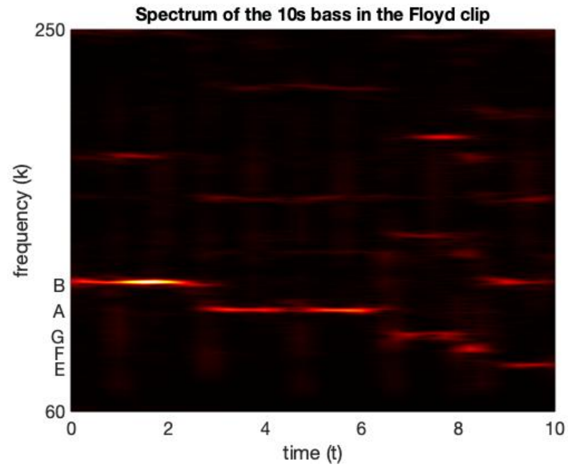

Figure 3: Spectrum of the 10s bass in the Floyd clip

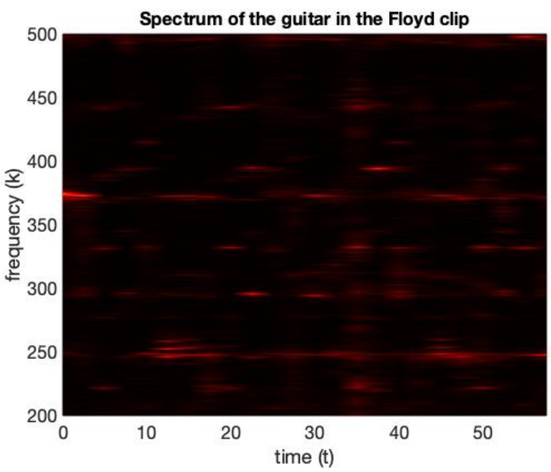

Figure 4: Spectrum of the guitar in the Floyd clip

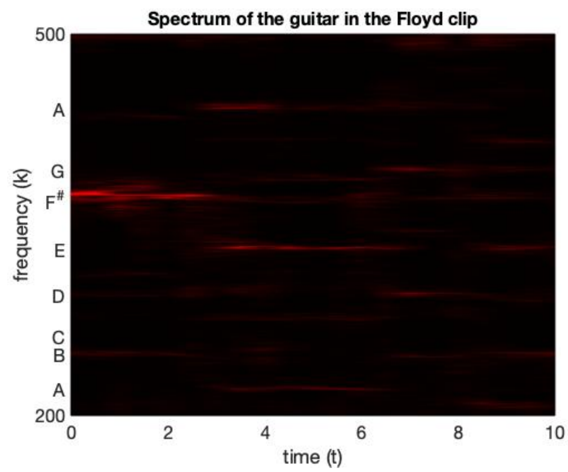

Figure 5: Spectrum of the 10s guitar in the Floyd clip

Comfortably Numb is:

$$B - B - B - A - A - A - G - F - E,$$

and this pattern is repeated over and over.

4.3 the Guitar in *Comfortably Numb*

The figure (4) is the entire spectrum of the guitar in *Comfortably Numb*. Clearly, it indicates the music notes are repeated about every ten seconds. The figure (5) is the spectrum of the 10s guitar in *Comfortably Numb*. We can read off the frequency of each spot from the y-axis of the spectrum and convert it into the corresponding music note. After that, we read off the time of each music note being played from the x-axis to put them in chronological order. When looking for the music notes played by the guitar from the spectrum, it is important to subtract the music notes played by the bass with its overtones, which we have already identified. The music score derived from the spectrum for the guitar in *Comfortably Numb* is:

$$BF^{\#} - AEA - DG - BCF^{\#},$$

and this pattern is repeated over and over.

5 Summary and Conclusions

We succeed in reproducing the music score for the guitar in *Sweet Child O' Mine* through the use of the Gabor filtering. Both guitar and bass are played in *Comfortably Numb*. We first use a band-pass filter in frequency domain to only examine the frequencies of music notes that are inside the bass frequency range, 60-250 Hertz, so that we can isolate the bass. Then we focus on the frequencies of music notes that are inside the guitar frequency range, 200-500, and filter out the bass notes and its overtones. We are now able to put much of the guitar solo together in *Comfortably Numb*. We find out that the music notes played by both guitar and bass are repeated about every ten seconds, so it is sufficient for us to reproduce the music scores for both guitar and bass by only looking at the first ten seconds of the song.

Appendix A MATLAB Functions

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[y,Fs] = audioread(filename)` reads data from the file named `filename`, and returns sampled data, `y`, and a sample rate for that data, `Fs`.
- `Y = fft(X)` computes the discrete Fourier transform (DFT) of `X` using a fast Fourier transform (FFT) algorithm.
- `Y = fftshift(X)` rearranges a Fourier transform `X` by shifting the zero-frequency component to the center of the array.
- `X = ifft(Y)` computes the inverse discrete Fourier transform of `Y` using a fast Fourier transform algorithm. `X` is the same size as `Y`.
- `pcolor(X,Y,C)` specifies the x- and y-coordinates for the vertices. The size of `C` must match the size of the x-y coordinate grid. For example, if `X` and `Y` define an m-by-n grid, then `C` must be an m-by-n matrix.
- `yticks(ticks)` sets the y-axis tick values, which are the locations along the y-axis where the tick marks appear. Specify `ticks` as a vector of increasing values; for example, [0 2 4 6]. This command affects the current axes.
- `yticklabels(labels)` sets the y-axis tick labels for the current axes. Specify `labels` as a string array or a cell array of character vectors; for example, 'January','February','March'. If you specify the labels, then the y-axis tick values and tick labels no longer update automatically based on changes to the axes.

Appendix B MATLAB Code

```matlab
%% guitar in the GNR clip
clear all; close all; clc

[y, Fs] = audioread('GNR.m4a');
tr_gnr = length(y)/Fs; % record time in seconds

S = y'; % Signal
L = tr_gnr;
n = length(S);
t2 = linspace(0,L,n+1); t = t2(1:n); % vector of time points
k = (1/L)*[0:n/2-1 -n/2:-1]; ks = fftshift(k); % Notice the 1/L instead of 2*
    pi/L

% Apply the Gabor transform implementation
```

```
tau = 0:0.1:L; %the center of the window
a = 1000;
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

% construct the spectrum
pcolor(tau,ks,Sgt_spec)
shading interp
set(gca, 'ylim', [0 1000], 'FontSize', 14)
xlabel('time (t)'), ylabel('frequency (k)')
colormap(hot)
yticks([0 278 311 370 415 500 554 698 1000]);
yticklabels({'0', 'C^#', 'D^#', 'F^#', 'G^#', '500', 'C^#', 'F', '1000'});
title('Spectrum of the guitar in the GNR clip', 'Fontsize', 14)

%% isolate the bass in Comfortably Numb
clear all; close all; clc

[y, Fs] = audioread('Floyd.m4a');
tr_gnr = length(y)/Fs; % record time in seconds

S = y'; % Signal
L = tr_gnr;
n = length(S);
t2 = linspace(0,L,n+1); t = t2(1:n); % vector of time points
k = (1/L)*[0:n/2-1 -n/2:-1]; ks = fftshift(k); % Notice the 1/L instead of 2*pi/L

% bandpass filter
S_fft = fft(S);
S_filter = S_fft.*fftshift(60<abs(S_fft)<250);
S_bass = ifft(S_filter);

% Apply the Gabor transform implementation
tau = 0:2.5:L; %the center of the window
a = 10;
for i = 1:length(tau)
    g = exp(-a*(t-tau(i)).^2); % Window function
    Sg = g.*S_bass;
    Sgt = fft(Sg);
    Sgt_spec(:,i) = fftshift(abs(Sgt));
end

% construct the spectrum
Sgt_spec(end,:) = [];
pcolor(tau,ks,Sgt_spec)
shading interp
set(gca, 'ylim', [60 250], 'FontSize', 16)
xlabel('time (t)'), ylabel('frequency (k)')
colormap(hot)
```

```
title('Spectrum of the bass in the Floyd clip', 'Fontsize', 16)

%% isolate 10s of the bass in Comfortably Numb
clear all; close all; clc

[y, Fs] = audioread('10sFloyd.m4a');
tr_gnr = length(y)/Fs; % record time in seconds

S = y'; % Signal
L = tr_gnr;
n = length(S);
t2 = linspace(0,L,n+1); t = t2(1:n); % vector of time points
k = (1/L)*[0:n/2-1 -n/2:-1]; ks = fftshift(k); % Notice the 1/L instead of 2*
    pi/L

% bandpass filter
S_fft = fft(S);
S_filter = S_fft.*fftshift(60<abs(S_fft)<250);
S_bass = ifft(S_filter);

% Apply the Gabor transform implementation
tau = 0:0.25:L; %the center of the window
a = 10;
for i = 1:length(tau)
    g = exp(-a*(t-tau(i)).^2); % Window function
    Sg = g.*S_bass;
    Sgt = fft(Sg);
    Sgt_spec(:,i) = fftshift(abs(Sgt));
end

% construct the spectrum
Sgt_spec(end,:) = [];
pcolor(tau,ks,Sgt_spec)
shading interp
set(gca, 'ylim', [60 250], 'FontSize', 16)
xlabel('time (t)'), ylabel('frequency (k)')
colormap(hot)
yticks([60 81 89 97 110 123 250]);
yticklabels({'60', 'E', 'F', 'G', 'A', 'B', '250'});
title('Spectrum of the 10s bass in the Floyd clip', 'Fontsize', 16)

%% the Guitar in Comfortably Numb
clear all; close all; clc

[y, Fs] = audioread('Floyd.m4a');
tr_gnr = length(y)/Fs; % record time in seconds

S = y'; % Signal
L = tr_gnr;
n = length(S);
t2 = linspace(0,L,n+1); t = t2(1:n); % vector of time points
k = (1/L)*[0:n/2-1 -n/2:-1]; ks = fftshift(k); % Notice the 1/L instead of 2*
    pi/L

```

```
% bandpass filter
S_fft = fft(S);
S_filter = S_fft.*fftshift(200<abs(S_fft)<500);
S_guitar = ifft(S_filter);

% Apply the Gabor transform implementation
tau = 0:2.5:L; %the center of the window
a = 10;
for i = 1:length(tau)
    g = exp(-a*(t-tau(i)).^2); % Window function
    Sg = g.*S_guitar;
    Sgt = fft(Sg);
    Sgt_spec(:,i) = fftshift(abs(Sgt));
end

% construct the spectrum
Sgt_spec(end,:) = [];
pcolor(tau,ks,Sgt_spec)
shading interp
set(gca, 'ylim', [200 500], 'FontSize', 16)
xlabel('time (t)'), ylabel('frequency (k)')
colormap(hot)
title('Spectrum of the guitar in the Floyd clip', 'Fontsize', 16)

%% the 10s of Guitar in Comfortably Numb
clear all; close all; clc

[y, Fs] = audioread('10sFloyd.m4a');
tr_gnr = length(y)/Fs; % record time in seconds

S = y'; % Signal
L = tr_gnr;
n = length(S);
t2 = linspace(0,L,n+1); t = t2(1:n); % vector of time points
k = (1/L)*[0:n/2-1 -n/2:-1]; ks = fftshift(k); % Notice the 1/L instead of 2*
    pi/L

% bandpass filter
S_fft = fft(S);
S_filter = S_fft.*fftshift(200<abs(S_fft)<500);
S_guitar = ifft(S_filter);

% Apply the Gabor transform implementation
tau = 0:0.5:L; %the center of the window
a = 10;
for i = 1:length(tau)
    g = exp(-a*(t-tau(i)).^2); % Window function
    Sg = g.*S_guitar;
    Sgt = fft(Sg);
    Sgt_spec(:,i) = fftshift(abs(Sgt));
end

% construct the spectrum
Sgt_spec(end,:) = [];
```

```
pcolor(tau,ks,Sgt_spec)
shading interp
set(gca, 'ylim', [200 500], 'FontSize', 16)
xlabel('time (t)'), ylabel('frequency (k)')
colormap(hot)
yticks([200 220 247 261 294 330 370 392 440 500]);
yticklabels({'200', 'A', 'B', 'C', 'D', 'E', 'F^#', 'G', 'A', '500'});
title('Spectrum of the guitar in the Floyd clip', 'Fontsize', 16)
```