Exploring The Gabor Transform Through Rock and Roll

Elliot Jennis

February 2021

Abstract

Sound files provide an excellent example for why the Fourier Transform on its own is a flawed tool for signal processing. Good time resolution and frequency resolution are not possible to achieve simultaneously. When time resolution is required for a given problem without sacrificing frequency resolution, other techniques must be used. The Gabor Transform and its disctertized versions provide a decent but imperfect solution to this dilemma. The following paper shows the strengths and weaknesses of the Gabor Transform through the lens of audio processing. Data taken from audio clips is transformed into the signal domain and individual notes played by different instruments are identified with varying degrees of success.

Introduction and Overview

Following the development of the Fourier transform (FT) and the signal processing techniques that relied on this tool, further work was needed to address the uncertainty principle it created. This problem was the lack of time resolution in the signal domain. This was not a problem for applications involving signals fixed in frequency. However, for other applications where frequency changes with time, this problem was enormous. An obvious case is that of a sound recording. Unless the recording is of a single uninterrupted note, the content of the recording's frequency will change with time. Consider the case of two audio clips from the famous rock and roll songs 'Sweet Child'O Mine' by Guns and Roses (SCOM), and 'Comfortably Numb' by Pink Floyd (CN). To identify the notes as well as the individual instruments in either clip requires time dependant frequency analysis. To achieve this task, a signal processing tool known as the Gabor Transform (GT) is used. This tool uses a series of time 'windows' to characterise the frequency content at each slice of time.

Theoretical Background

The Gabor Transform was developed by British-Hungarian scientist Dennis Gabor in the latter half of the 20th century [1]. The basic principle behind his approach was to break up a signal into windows of time for which a FT would be applied. Frequency content for each window would be known with the number of windows used on a given signal directly affecting the time resolution of the signal in frequency space (Figure 1). Mathematically, this

process is derived by modifying the kernel of the Fourier transform to that of the expression shown in Equation 1[1].

$$g_{t,\omega}(\tau) = e^{i\omega\tau} g(\tau - t) \tag{1}$$

In this case τ represents the location in time which the Gabor window is centered. Rewriting the FT with this new kernel defined the function known as the Gabor Transform or the short-time Fourier Transform (Equation 2)[1].

$$G[f](t,\omega) = \tilde{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)\bar{g}(\tau - t)e^{-i\omega\tau} d\tau \tag{2}$$

By integrating with respect to τ, the Gabor window can be translated over the entire signal. Further specificity can be made regarding the size of the Gabor window by multiplying the kernel by a constant a. A wider window size will allow for greater resolution in frequency and the capture of larger structures in the frequency domain indicating components of lower frequency. However, as a result of their greater width some time resolution is lost. A smaller window size will lead to greater time resolution while ignoring the frequency content with wavelengths longer than the width of the window. This is the biggest drawback of the Gabor transform. It does not completely solve the problem of time and frequency resolution rather it shifts the uncertainly principle. Capturing low frequency and high time resolution is now very difficult, whereas capturing high frequency with high time resolution is quite trivial.

The bar above the g in Equation 2 represents the complex conjugate of the function g. Making a few simple assumptions allows for this bar to be thrown away [1]. These assumptions being that g is real and symmetric with its norm being 1. These assumptions allow for the derivation of the inverse Gabor transform denoted in Equation 3[1].

$$f(\tau) = \frac{1}{2\pi}\frac{1}{\|g\|^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{f}_g(t,\omega)g(\tau - t)e^{-i\omega\tau} d\omega d\tau \tag{3}$$

The double integral is required to cover the frequency and time components of the GT whereas before with the FT only a single integration was required since it did not vary with time [1].

Algorithm Implementation and Development

The algorithm used to develop the code for this particular problem involved 3 basic steps:

1. Shorten the data into smaller usable clips (for SCOM 5 seconds, for CN 10-15 seconds)
2. Create a Gabor filter and iterate to find the correct window size and time step intervals
3. Use the Gabor filter in conjunction with an FFT to transform the date into frequency space and plot the spectrograms of the filtered frequency content

Additional steps used for filtering within frequency space will be discussed later in this section. The length of the SCOM data was about 15 seconds and the CN clip was over a minute long. Using the Gabor transform to analyze the frequency content of the entire two

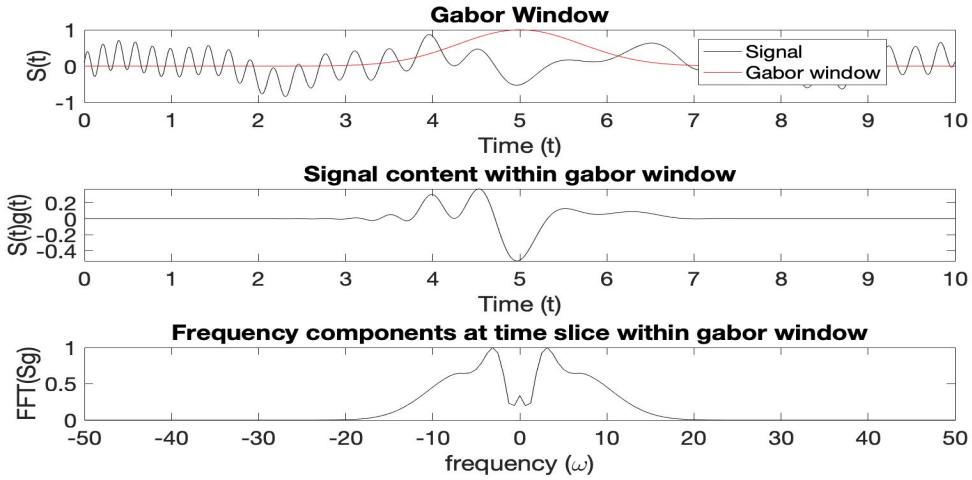

Figure 1: Graphical depiction how the Gabor window captures frequency content from data

clips would take a lot of processing power and memory to run all the necessary calculations . The computer used for this task was an older MacBook Pro, so even with shortened clips the code still took many minutes to run. To cut down the data, the L value was altered, with the time and n parameters both being defined by this shortened L value. For the SCOM clip this length was chosen to be 5 seconds and for CN this value was 15 for the bass and 10 for the guitar solo. By dividing the sample rate given when importing the data, n could be defined in terms of L as well. This value allowed the scaling of time and frequency space to only include the length of the clip L that was required. It also allowed for the use of exactly n terms of data when using the GT. This scaling is shown in Listing 1 as well as the beginning lines of Listings 2 and 3.

The K value used in this problem also differs from the typical way it is defined when used for a Fast Fourier Transform. The frequency units needed for this problem were in Hz. When using the traditional periodic definition of K, the frequency content scaling does not match up with the notes in frequency space. By simply removing 2π from the definition of K, the frequency matrix can be changed to units of Hz, making the spectrogram plot line up nicely with the actual values of the notes played.

To find the correct window size and time step length for the Gabor filter applied to both SCOM and CN, the method of try, check, and revise was repeatedly used. The form of the Gabor filter used in the code is $g = (\alpha)exp(\beta(t - tslide))^2$ (Listing 1,2,3). In this form, α represented an intensity value increasing or decreasing the brightness of the spectrogram. β directly influences the Gabor window size. The higher this number the smaller the width of the Gabor window. The variable $tslide$ represented a vector defined as $tslide = 0 : Tstep : L$. The time step value tells the Gabor window how far to move per iteration of the algorithm. The smaller this value, the slower the Gabor widow moves through the data set. Initially, time resolution was incredibly poor but as window size and the time step distance decreased, the notes came into view.

The CN data set had an additional layer of complexity. To identify different instruments within a single sound clip, a band-pass filter was created to show specific regions of frequency

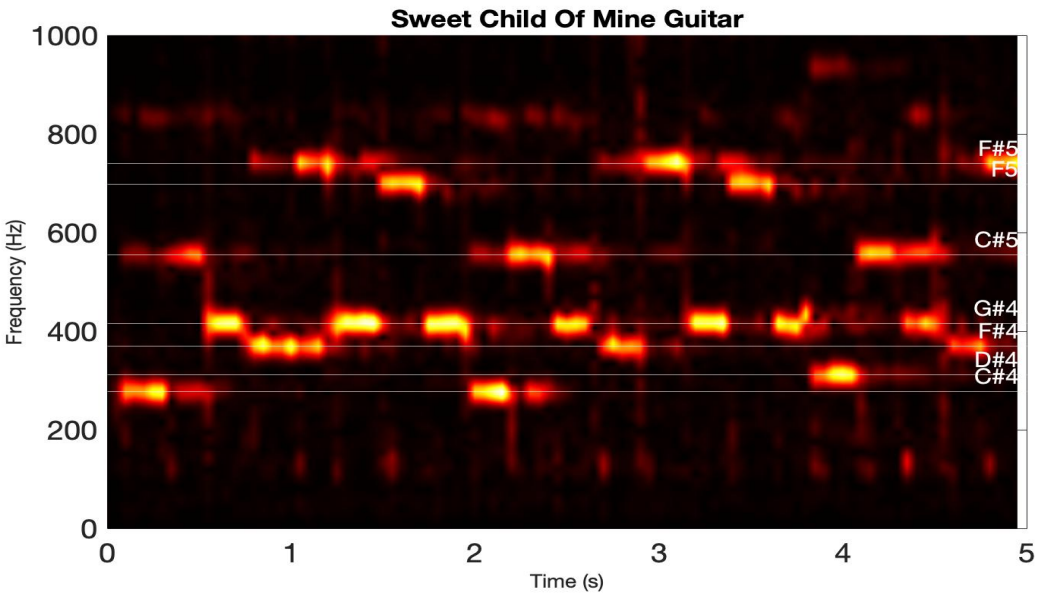

Figure 2: Spectrogram of first 5 seconds of Sweet Child'O Mine

space. For the bass, higher and middle frequencies were filtered out, and for the guitar lower and higher frequencies were filtered out.

Computational Results

Sweet Child'O Mine

To visualize the results from the SCOM data-set, a spectrogram plot was created with white lines corresponding to different musical notes overlaid on-top (Figure 2). Matching the locations of high signal strength with the horizontal lines allows for identification of specific notes at different instances of time[2]. The sequence of notes for the first 5 seconds are described in Table 1. The value for α was .05 and the value for β was 5000. The time step value chosen was 0.05. The code used to generate this data is shown in Listing 1.

Notes Played SCOM												
Notes	$C_4^\#$	$C_5^\#$	$G_4^\#$	$F_4^\#$	$F_5^\#$	$G_4^\#$	F_5	$G_4^\#$	$C_4^\#$	2-8	$D_4^\#$	2-8
Position	1	2	3	4	5	6	7	8	9	10-16	17	18-24

Table 1: First six stanzas of Sweet Child'O Mine[2]

Comfortably Numb

A similar procedure to the process described above for SCOM was done for the CN data set. Spectrogram plots were created with lines representing the frequencies of known musical

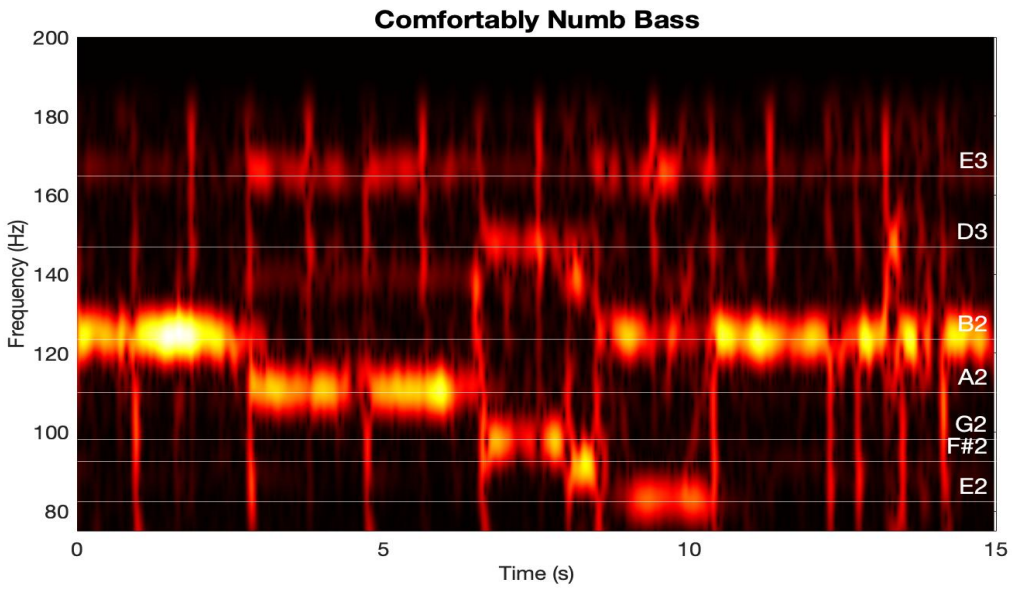

Figure 3: Spectrogram of first 15 seconds of Comfortably Numb (75-200 Hz)

notes overlaid on top of them. For the bass guitar, (Figure 3) the value for α was 0.07 and for β was 300. The time step interval chosen was 0.03. For the guitar solo itself (Figure 4), the value for α was 0.1 and for β was 450. The time step interval was 0.03. The discrepancy in α is due to the strength of the guitar signal relative to the bass. On the spectrogram plots, the bass notes are very apparent whereas the guitar is much more muddled, and requires a bigger boost to be visible. The value for β is higher for the guitar as well to reflect the shorter wavelengths of the notes played by the guitar. A wider Gabor window is not necessary to capture these frequencies, so smaller window is used to achieve greater time resolution. Additionally, for the bass notes, overtones are visible in the top section of the spectrogram plot between 140 and 180 Hz. These signals were ignored as they did not represent the actual notes played. The full code for the bass guitar is shown in Listing 2 while the code for the guitar solo is in Listing 3.

Notes Played CN Bass							
Notes	B_2	A_2	G_2	$F_2^{\#}$	B_2	E_2	B_2
Position	1	2	3	4	5	6	7

Table 2: Bass notes played in Comfortably Numb[2]

Notes Played CN Guitar										
Notes	$F_4^{\#}$	$D4$	$A4$	$E4$	$F_4^{\#}$	G_4	B_4	D_5	B_4	D_5
Position	1	2	3	4	5	6	7	8	9	10

Table 3: First ten seconds of Comfortably Numb Guitar Solo[2]

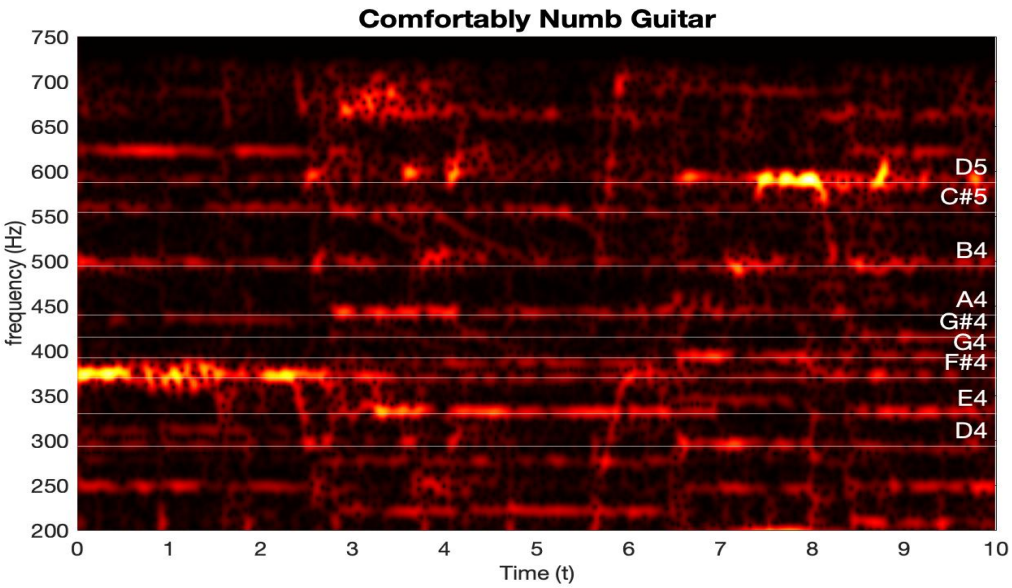

Figure 4: Spectrogram of first 10 Seconds of Comfortably Numb (200-600 Hz)

Summary and Conclusions

The quality of the specrtrograms for the three different problems presented above vary widely. The SCOM data set is comprised of only one instrument, the guitar. This simplifies the processing steps required to identify individual notes. There will be noise in the data but the underlying guitar signal will remain clear. The biggest challenge was finding the correct settings for the Gabor filter (window size and time step). There is a a high degree of confidence in the end results for the SCOM dataset.

The CN data set presented a different challenge. The audio file given included the entire file, no individual instrument tracks were given. The task was to see if separating out instruments from the audio-file was possible. A band-pass filter was used to accomplish this as the bass will play notes of a much lower frequency than the guitar. Beginning with the bass, these notes were not difficult to identify. The combination of long wavelength and low frequency made the locations of each bass note appear clearly on the spectrogram. The guitar was an entirely different story. No combination of settings for Gabor window and time step would make the frequencies appear more clearly than in Figure 4. Yet even then, Figure 4 is still quite unclear about where individual notes are. There is a strong signal close to 370 Hz at the beginning and a strong signal at 600 Hz towards the end. In between, the graph leaves much up to interpretation.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

[2] B. H. Suits. *Physics of Music - Notes*. 1998. URL: https://pages.mtu.edu/~suits/notefreqs.html.

A MATLAB Functions

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.

- `y = bandpass(x,wpass)` filters the input signal `x` using a bandpass filter with a passband frequency range specified by the two-element vector `wpass` and expressed in normalized units of rad/sample. `bandpass` uses a minimum-order filter with a stopband attenuation of 60 dB and compensates for the delay introduced by the filter. If `x` is a matrix, the function filters each column independently.

- `[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 = fftn(X)` returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm. The N-D transform is equivalent to computing the 1-D transform along each dimension of X. The output Y is the same size as X.

- `Y = fftshift(X)` rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.

- `pcolor(C)` creates a pseudocolor plot using the values in matrix C. A pseudocolor plot displays matrix data as an array of colored cells (known as faces). MATLAB® creates this plot as a flat surface in the x-y plane. The surface is defined by a grid of x- and y-coordinates that correspond to the corners (or vertices) of the faces. The grid covers the region `X=1:n` and `Y=1:m`, where `[m,n] = size(C)`. Matrix C specifies the colors at the vertices. The color of each face depends on the color at one of its four surrounding vertices. Of the four vertices, the one that comes first in the x-y grid determines the color of the face.

- `shading interp` varies the color in each line segment and face by interpolating the colormap index or true color value across the line or face.

- `c = hot` returns the hot colormap as a three-column array with the same number of rows as the colormap for the current figure. If no figure exists, then the number of rows is equal to the default length of 256. Each row in the array contains the red, green, and blue intensities for a specific color. The intensities are in the range [0,1], and the color scheme looks like this image.

B MATLAB Code

```
%%Part 1 Sweet Child'O Mine
[y1, Fs1] = audioread('GNR.m4a');
time1 = length(y1)/Fs1; % record time in seconds

L=5; %domain size (could be 10 seconds, hours, etc)
n=length(y1)/time1*L;
t2=linspace(0,L,n+1);
t=t2(1:n);
k=(1/L)*[0:n/2-1 -n/2:-1]; %frequency Hz
ks=fftshift(k);

%% Gabor Filter
ygt_spec1=[];
tslide=0:0.05:t(end);
for j=1:length(tslide)
g=.05*exp(-5000*(t-tslide(j)).^2); % Gabor window size
yg1=g.*transpose(y1(1:n));
ygt1=fft(yg1);
ygt_spec1=[ygt_spec1; fftshift(abs(ygt1))];
end

figure(1)
pcolor(tslide,ks,log(abs(ygt_spec1).'+1));
shading interp
set(gca,'Fontsize',[20]);
xlabel('Time (s)','FontSize',14), ylabel('Frequency (Hz)','FontSize',14),title('Sweet Ch
axis([0 L 0 1000]);
colormap(hot);
hold on, yline(277.18,'w','C#4','Fontsize',16), hold on, yline(311.13,'w','D#4','Fontsiz
yline(369.99,'w','F#4','Fontsize',16), hold on, yline(415.3,'w','G#4','Fontsize',16), ho
hold on, yline(698.46,'w','F5','Fontsize',16), hold on, yline(739.99,'w','F#5','Fontsize
```

Listing 1: Sweet Child'O Mine Guitar

```
% % Part 2 Comfortably Numb Bass
clear all; clc;
[y2, Fs2] = audioread('Floyd.m4a');
time2 = length(y2)/Fs2; % record time in seconds
L=15; n=length(y2)/time2*L;
t2=linspace(0,L,n+1); t=t2(1:n);
k=(1/L)*[0:n/2-1 -n/2:-1]; ks=fftshift(k);
% Bandpass filter
y_filter = bandpass(y2,[75,175],Fs2);
ygt_spec2=[];
tslide=0:0.03:t(end);
for j=1:length(tslide)
g=0.07*exp(-300*(t-tslide(j)).^2)'; % Gabor window
yg2=g.*y_filter(1:n); %GNR in Gabor Window time domain
ygt2=fft(yg2)'; %fft of gabor window GNR
ygt_spec2=[ygt_spec2; fftshift(abs(ygt2))];
end

figure(1)
pcolor(tslide,ks(330751:334976),log(abs(ygt_spec2(:,330751:334976).'+1))), shading inter
set(gca,'Fontsize',[14]);
colormap(hot)
axis([0 L 75 200])
xlabel('Time (s)','Fontsize',14), ylabel('Frequency (Hz)','Fontsize',14);
title('Comfortably Numb Bass','FontSize',18);
 hold on
 yline(82.31,'w','E2','FontSize',16),hold on,yline(92.5,'w','F#2','FontSize',16),hold on
 yline(98.00,'w','G2','FontSize',16), hold on, yline(110,'w','A2','FontSize',16),hold on
 yline(123.47,'w','B2','FontSize',16), hold on,
 yline(146.83,'w','D3','FontSize',16), hold on, yline(164.81,'w','E3','FontSize',16);
```

Listing 2: Comfortably Numb Base Guitar

```
% Part 3 Comfortably Numb Guitar
clear all; clc
[y2, Fs2] = audioread('Floyd.m4a');
time2 = length(y2)/Fs2; % record time in seconds
L=10; n=length(y2)/time2*L;
t2=linspace(0,L,n+1); t=t2(1:n);
k=(1/L)*[0:n/2-1 -n/2:-1]; ks=fftshift(k);
% Bandpass filter
y_filter = bandpass(y2,[200,700],Fs2);
ygt_spec2=[];
tslide=0:0.03:t(end);
for j=1:length(tslide)
g=0.1*exp(-450*(t-tslide(j)).^2)'; % Gabor window
yg2=g.*y_filter(1:n); %GNR in Gabor Window time domain
ygt2=fft(yg2)'; %fft of gabor window GNR
ygt_spec2=[ygt_spec2; fftshift(abs(ygt2))];
end


figure(3)
pcolor(tslide,ks,log(abs(ygt_spec2).'+1));
shading interp
set(gca,'Fontsize',[14]);
colormap(hot)
axis([0 L 200 750])
xlabel('Time (t)','Fontsize',14), ylabel('frequency (Hz)','Fontsize',14);
title('Comfortably Numb Guitar','FontSize',18);
hold on, yline(293.66,'w','D4','Fontsize',16), hold on, yline(329.63,'w','E4','Fontsize'
yline(369.99,'w','F#4','Fontsize',16), hold on, yline(392,'w','G4','Fontsize',16), hold
yline(415.3,'w','G#4','Fontsize',16),hold on
yline(440,'w','A4','Fontsize',16),hold on, yline(493.88,'w','B4','Fontsize',16), hold on
yline(554.37,'w','C#5','Fontsize',16),hold on, yline(587.33,'w','D5','Fontsize',16);
```

Listing 3: Comfortably Numb Guitar Solo