Rock & Roll and Gabor Transform

Yichen Shen

January 24, 2020

Abstract

This paper introduce the theoretical background, the algorithm implementation and development, and the computational results of the research on utilizing Gabor transform to visualize frequency and so to convey the note information from the given music data. This paper as mentioned, is going to implement Gabor transform and filtering to the data in order to abstract frequency density for notes and lift a single instrument from the track.

1 Introduction and Overview

1.1 Background and Context

Two of the greatest Rock and Roll music are being introduced in this topic, which are: *Sweet Child O' Mine* by Guns N' Roses and *Comfortably Numb* by Pink Floyd. These two pieces enjoy a high reputation and stable position in the music industry; with GNR holding a rank of 37 and Floyd holding a rank of 4 in the Guitar World ranking. With the enthusiasm for music, we can use computational methods to analyze the composition of these two pieces of music.

1.2 Topic Importance

Data like waves would produce information in terms of frequency and its variation in times. In this case, using Fourier Transform is no longer legitimate since it would loose the information in the time domain. Therefore, we want to adapt a new model: Gabor Transform in order to help with dealing the problems with time vs. frequency. Sound is one of the most common form of information isotopic carriers it is important to understand this modality, and understand how to extract information from the given music clip.

1.3 Objectives of Research

1. Reproducing the music score for guitar in the GNR clip as well as for the bass in the Floyd clip.
2. Utilize filter to lift the bass from the Comfortably Numb.
3. Similarly, extract the guitar solo in the Comfortably Numb.

2 Theoretical Background

2.1 Gabor Transform

Shift a filter function by τ then multiply it by a function $f(t)$, gives us a filtered function $f(t)g(t-\tau)$. The center of which is going to be at τ. Developed from this, we get the *Gabor transform* (STFT) function given by:

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

In which, the $\tilde{f}_g(\tau,k)$ present the frequency components and we can make the assumption that the function g is real and symmetric and the $L_2 norm$ of g is set to unity.

This is a similar to the Fourier Transform but Gabor transform return the value in frequency domain so that we can use this to target our frequency and find the desire note based on the frequency.

2.2 Discrete Gabor Transform

When using the Gabor Transform we essentially need a discrete version so that we are shifting the window by some discrete set of values. It is also essential to know the discrete frequencies: $k = m\omega_0$, and $\tau = nt_0$. The discrete Gabor transform is given by the following formula:

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

It is also an important feature to know that we want to include a big window and a small ω_0 for a good frequency solution.

2.3 Filter with Gaussian function

When receiving a signal with noise, it is hard to detect where the pulse is coming from. Therefore, one needs to filter the noise in order to better analyze the signal. The filter that can be use is a function and then multiply it with the signal in the frequency domain. One simply choice of the filter functions is the Gaussian function:

$$F(k) = e^{-\tau(k-k_0)^2} \tag{3}$$

τ is the width of the filter and constant k_0 is the centre of the filter in the frequency domain. The smaller the width the more signal it filter as the Gaussian decay is slower.

3 Algorithm Implementation and Development

We want to modify the vector of the read audio, and by doing so we need our time interval to be the length of the audio and the sample at a n space in which n is the length of the vector. Since we are utilizing the Gabor transform, it is important to note that we are re-scaling the frequency domain at a step of $1/L$ in stead of $2\pi/L$.

3.1 Apply Gabor transform and plot spectrogram

The draw back of Fourier Transform in visualizing the time and frequency has mentioned above, so we are using Gabor Transform instead. In order to perform the Gabor Transform, we implement our Algorithm 1:

1. Set up a variable τ to represent the time interval steps with the increment. Also, define the filter to be a.
2. In the loop, modify the vector with the Gabor Transform and center the filter at corresponding time in each loop.
3. In the loop, save our filtered frequency and time information in a form of matrix with respect to each loop.
4. Out of the loop, use spectrogram command to plot the frequency and time relationship and analyze the frequency with respect to the information points.

Algorithm 1: Apply Gabor Transform and Plot Spectrogram

```
Read the audio from GNR/Floyd.m4a
Turn the audio information into vectors y and define the filter a and time interval steps to be $\tau$ with time steps.
for $j = 1 : length(tau)$ do
    Apply the Gabor window function to the vector y
    Apply the FFT transform on the filtered vector
    Store the data into a matrix Sgtspec
end for
Plot the frequency spectrogram with respect to the matrix Sgtspecs frequency and time information
```

3.2 Filtering the Bass and Guitar

In order for us to extract the sound of the instrument with certain frequency, we can use filter to filter around the center of wanted instrument's frequency in each piece and define the maximum value in order to filter out any overtone or unwanted noise. To best isolate the instrument from a clip, we need to filter around the instrument's frequency and update it with the music is being played. We want to create a loop to first store the frequency and time information as a matrix as like whats being done in Algorithm 1, then implement the Algorithm 2 to filter around the center of each time step:

1. Repeat what is being done in Algorithm 1 and enter the loop.
2. In the loop, after creating the matrix, we are going to define the frequency domain of the wanted instrument, for Bass, we choose to analyze frequency under 250Hz, and for Guitar, we analyze frequency above 250Hz (knowing that the bass has a frequency lower than 250Hz).
3. In the loop, find the position of those frequencies and their corresponding amplitude, then extract the maximum amplitude to be the center of filter.
4. In the loop, since we know frequencies have both positive and negative amplitude, we can apply two filter to each of them, then update their sum in the matrix.
5. In the loop, we will going to transform the frequency back to its music audio, to do this we can use `IFFT` transform to the filtered frequency vectors.

4 Computational Results

4.1 Produce music score

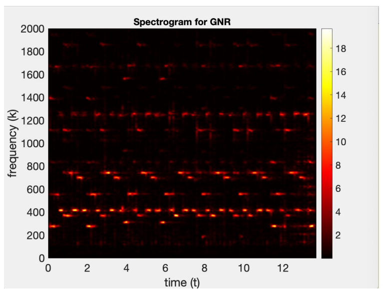
4.1.1 Guitar in GNR clip

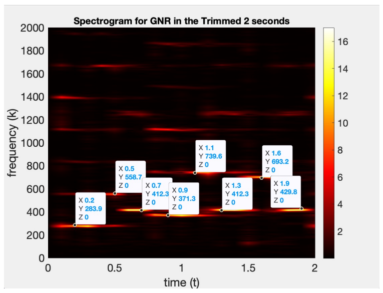
Given that music clip GNR only has guitar playing in it, in order to find the note, we can take a look at the whole picture (Figure 1). We set the limit for y to be positive and up to 2000, and ignore the repeat negative frequency. In the data, the most density frequency is the lightest, so in order to find the corresponding frequency, we can take a look at the range around 200Hz and 800Hz. It is easier to take a closer look at the spectrogram in a smaller time interval, as since the notes is repetitive except for a few different notes in GNR. Therefore, we can trim the original music clip and turn it into the spectrogram of each syllable. After we proceed the spectrogram for the first syllable, we can obtain the corresponding frequency from the graph (Figure 2). In order to spot the note, we need to refer to a reference music scale along with frequency (Figure 3). The first note has a frequency of roughly 283.9, therefore, we know it is a $\#C_4$. Similarly, we can find the following is $\#C_5, \#G_4, \#F_4, \#F_5, \#G_4, F_5, \#G_4$. After process the same steps for the following syllables, we are able to produce the different notes, which are always the first one and are: $\#C_4^{;}\#D_4, \#D_4, \#F_4, \#F_4$.

```
Algorithm 2: Filtering the Bass and Guitar
Read the audio from GNR/Floyd.m4a
Turn the audio information into vectors y and define the filter a and time interval steps to be $\tau$ with time steps.
for $j = 1 : length(tau)$ do
    Apply the Gabor window function to the vector y
    Apply the FFT transform on the filtered vector
    Store the data into a matrix Sgtspec
    find frequency vector that are below or above the define frequency interval and extract the maximum frequency in the frequency domain using [M,I] = (A).
    Trace the maximum amplitude location in the vector using find, then use this information to find the corresponding element int eh frequency vector.
    Define two filter with one filtering positive frequencies and the other filtering negative frequencies. Then add them up to store in the matrix.
    Apply ifft() to transfer the frequency back to music audio.
end for
Plot the frequency spectrogram with respect to the matrix Sgtspecs frequency and time information
Plot the time and amplitude graph with the found maximum amplitude and time.
```

4.1.2 Bass in Floyd clip

In this clip, the situation is slightly different. Instead of having one instrument, Floyd contains a combination of instruments which can be hard to identify the note for just one instrument. First, we again produce a spectrogram for the Floyd clip (Figure 4). Bass generally has a lower frequency, thus we can take a look at the lower frequency domain which in this case we choose the range to be 0 to 300Hz. Again, we trim the clip into smaller time interval and analyze the note (Figure 5,6,7). In this spectrogram we can see that the first note's frequency is 124.1, which correspond to B_2. Similarly, the followings notes are: $A_2, G_2, \#F_2, E_2, B_3, D_3, B_3, A_3$.





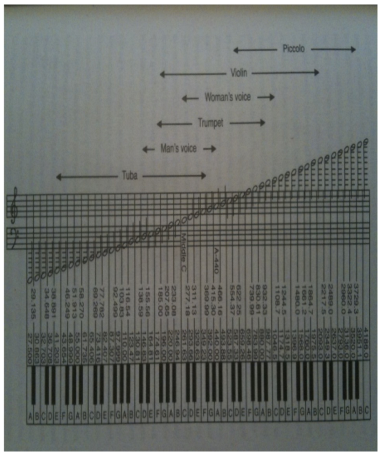
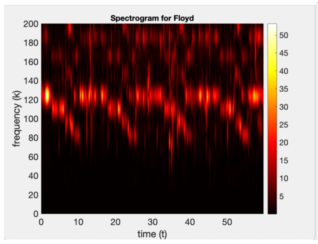
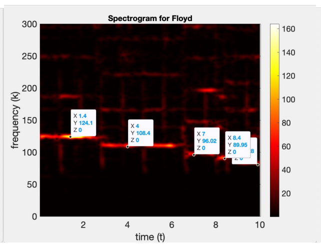


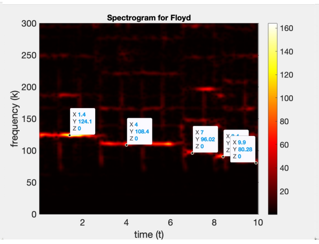
Figure 1: The spectrogram of GNR

Figure 2: The spectrogram of GNR in the trimmed 2 seconds.

Figure 3: The picture of the music scale corresponding to the frequency in Hz.







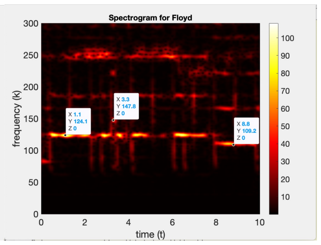


Figure 4: The spectrogram of Floyd

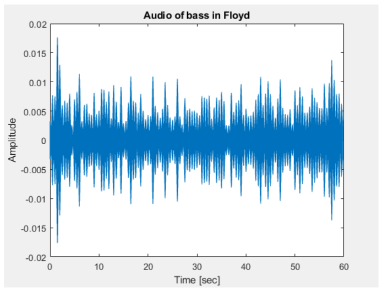
Figure 5: The spectrogram of Floyd in the trimmed 10 seconds.

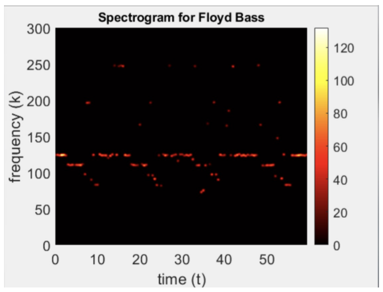
Figure 6: The spectrogram of Floyd in the trimmed 10 seconds.

Figure 7: The spectrogram of Floyd in the second trimmed 10 seconds.

4.2 Filter bass frequency

After applying the filter for the Floyd clip, we get a spectrogram and an audio amplitude graph for the isolated bass frequency and amplitude. By define the interval of frequency in below 250Hz, we are able to find the bass notes and filter out the overtones. Figure 8 gives us the sound amplitude graph for the bass while Figure 9 and 10 give the spectrogram of Bass in Floyd respect to the entire clip and the trimmed 10 seconds. By looking at the frequency, we are able to obtain the notes from the frequency density.





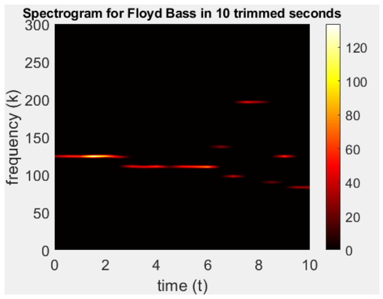


Figure 8: The sound graph of Bass in Floyd

Figure 9: The spectrogram of Bass in Floyd.

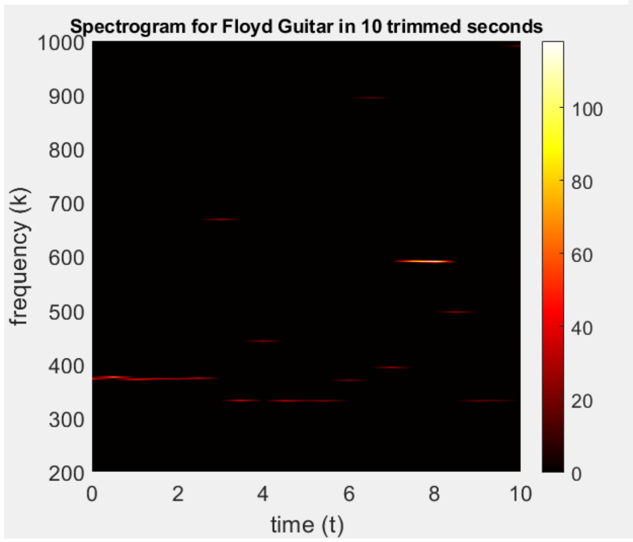
Figure 10: The spectrogram of Bass in Floyd in the trimmed 10 seconds.

4.3 Filter guitar frequency

Similar to filtering Bass frequencies, we compute via algorithm 2 to get a spectrogram. In Figure 11, we trimmed the clip into a 10 seconds interval and we able to see the frequency density of the guitar, however, due to the fact that guitar is playing in a fast tempo and there are multiple music instruments, the frequency density are tiny to spot. From the graph and with the similar method from previous part, we can obtain our notes as: $\#F_4, E_4, G_4, D_5, B_3$.

5 Summary and Conclusions

This research paper studies on the topic of utilizing Gabor transform and filtering in order to identify the music instruments with different frequency from a clip of audio. In order to employ these tools, this paper prepares the theoretical background which includes the (Discrete) Gabor Transform and Gaussian filter



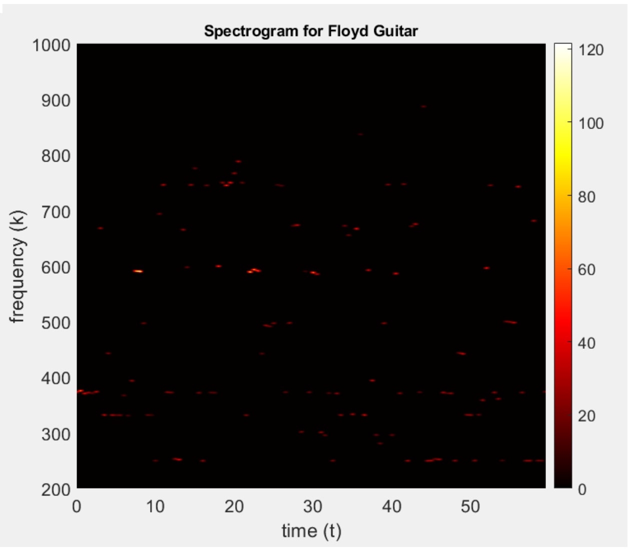


Figure 11: The spectrogram of Guitar in Floyd.

Figure 12: The spectrogram of Guitar in Floyd in the trimmed 10 seconds.

function. After implement the algorithm, the paper documents the computational results which includes the Bass notes in Floyd clip and the guitar notes in GNR clip, as well as isolating the bass in Comfortably Numb and track the guitar solo in the Comfortably Numb. The significance of this research is the implementation indicates that one can uses Gabor transform to analyze data on the frequency and time domain coincidentally and are able to identify the sound with different frequency, since sound is a big media to convey information in the real world, it is important to understand the background theology and algorithms to prepare for the further implementation on sound information.

Appendix A MATLAB Functions

Important MATLAB functions with a brief implementation explanation.

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[X,Y] = meshgrid(x,y)` returns 2-D grid coordinates based on the coordinates contained in the vectors `x` and `y`. X is a matrix where each row is a copy of `x`, and `Y` is a matrix where each column is a copy of `y`. The grid represented by the coordinates `X` and `Y` has `length(y)` rows and `length(x)` columns.
- `fftn(X)` uses the Fast Fourier Transform algorithm to return the multidimensional Fourier Transform of an N-D array.
- `fftshift(A)` shift the zero frequency component to center of the spectrum
- `ifftn(A)` returns the multidimensional inverse discrete Fourier Transform, using FFT, of A with same size.
- `ifftshift(A)` shift the zero-frequency-shifted Fourier Transform X back to the original transform output.
- `Pcolor(A)` crates a pseudocolor plot with the value in matrix A.
- `[M,I] = max(A)` returns the maximum elements of an array A.

Appendix B MATLAB Code

```
%% Load music clip for GNR
figure(1)
[y, Fs] = audioread('GNR_T.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');
p8 = audioplayer(y,Fs); playblocking(p8);

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


%%% Use Gabor Transform
% Creating a spectrogram
a = 1000;
tau = 0:0.1:L;

Sgt_spec = zeros(length(y),length(tau));
for j = 1:length(tau)
    g = exp(-a*(t-tau(j)).^2); % window function
    Sg = g.*y';
    Sgt = fft(Sg);
    Sgt_spec(:,j) = fftshift(abs(Sgt));
end

figure(2)
pcolor(tau,ks,Sgt_spec)
shading interp
set(gca,'ylim',[0 2000],'Fontsize',16)
colormap(hot);
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title("Spectrogram for GNR in the Trimmed 2 seconds", 'Fontsize', 14)
```

Listing 1: Code to perform algorithms 1

```
%% Load music clip for Floyd
figure(3)
[y, Fs] = audioread('Floyd_T2.m4a');
tr_floyd = length(y)/Fs; % record time in seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Comfortably Numb');
p8 = audioplayer(y,Fs); playblocking(p8);

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

%%% Use Gabor Transform
% Creating a spectrogram
a = 100;
tau = 0:0.1:L;

Sgt_spec = zeros(length(y),length(tau));
for j = 1:length(tau)
    g = exp(-a*(t-tau(j)).^2); % window function
    Sg = g.*y';
    Sgt = fft(Sg);
    Sgt_spec(:,j) = fftshift(abs(Sgt));
end

figure(4)
dim_agree = pcolor(tau,ks,Sgt_spec(1:length(ks),:));
shading interp
set(gca,'ylim',[0 300],'Fontsize',16)
colormap(hot);
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title("Spectrogram for Floyd", 'Fontsize', 14)
```

Listing 2: Code to perform algorithms 1

```
%% Filtering the Bass

close all, clear

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


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

% Gaussian filter
a = 150;
tau = 0:0.5:L;
time = zeros(length(ks),length(tau));
for j = 1:length(tau)
    g = exp(-a*(t-tau(j)).^2);
    Sg = g.*y';
    Sgt = fft(Sg);
    Sgt_spec(:,j) = fftshift(abs(Sgt));
    Sgt_spec_time = Sgt_spec(:,j);
    positive_ks = abs(ks);
    amp = positive_ks < 250;
    amp_vec = Sgt_spec_time(amp);
    [M,I] = max(amp_vec);
    amp_loci = find(Sgt_spec_time == M);
    max_freq = ks(amp_loci);

    new_tau = 0.3;
    filter_1 = exp(-new_tau*(ks-max_freq(1)).^2);
    filter_2 = exp(-new_tau*(ks-max_freq(2)).^2);
    temp = Sgt_spec(1:length(ks));
    Sgt_spec(1:length(ks),j) = Sgt_spec(1:length(ks),j).'.*filter_1 + Sgt_spec(1:length(ks),j).'.*filte

    % transfer to music audio
    Sgt = fftshift(Sgt);
    sgft = Sgt(1:length(ks)).*filter_1 + Sgt(1:length(ks)).*filter_2;
    Sgf = ifft(sgft);
    time = Sgf.' + time;

end
plot((1:length(ks))/Fs,time);
xlabel('Time [sec]'); ylabel('Amplitude');
title("Audio of bass in Floyd");


figure(5)
pcolor(tau,ks,Sgt_spec(1:length(ks),:))
shading interp
set(gca, 'ylim', [0 300], 'Fontsize', 16)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title("Spectrogram for Floyd Bass", 'Fontsize', 14);
```

Listing 3: Code to perform algorithms 2

```
%% Filtering the Guitar

close all, clear

[y, Fs] = audioread('Floyd_T1.m4a');
tr_floyd = length(y)/Fs; % record time in seconds


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

% Gaussian filter
a = 150;
tau = 0:0.5:L;
time = zeros(length(ks),length(tau));
for j = 1:length(tau)
    g = exp(-a*(t-tau(j)).^2);
    Sg = g.*y';
    Sgt = fft(Sg);
    Sgt_spec(:,j) = fftshift(abs(Sgt));
    Sgt_spec_time = Sgt_spec(:,j);
    positive_ks = abs(ks);
    amp = positive_ks > 250;
    amp_vec = Sgt_spec_time(amp);
    [M,I] = max(amp_vec);
    amp_loci = find(Sgt_spec_time == M);
    max_freq = ks(amp_loci);

    new_tau = 0.3;
    filter_1 = exp(-new_tau*(ks-max_freq(1)).^2);
    filter_2 = exp(-new_tau*(ks-max_freq(2)).^2);
    temp = Sgt_spec(1:length(ks));
    Sgt_spec(1:length(ks),j) = Sgt_spec(1:length(ks),j).'.*filter_1 + Sgt_spec(1:length(ks),j).'.*filte

    % transfer to music audio
    Sgt = fftshift(Sgt);
    sgft = Sgt(1:length(ks)).*filter_1 + Sgt(1:length(ks)).*filter_2;
    Sgf = ifft(sgft);
    time = Sgf.' + time;

end
plot((1:length(ks))/Fs,time);
xlabel('Time [sec]'); ylabel('Amplitude');
title("Audio of Guitar in Floyd");


figure(6)
pcolor(tau,ks,Sgt_spec(1:length(ks),:))
shading interp
set(gca, 'ylim', [200 1000], 'Fontsize', 16)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title("Spectrogram for Floyd Guitar", 'Fontsize', 14);
```

Listing 4: Code to perform algorithms 2