John Curtis

Assignment 2: Rock & Roll and the Gabor Transform

Abstract

This paper explores methods to perform frequency analysis on music clips (*Sweet Child O' Mine* by Guns N' Roses and *Comfortably Numb* by Pink Floyd) and identify the instruments and notes that comprise them. The Gabor transform and spectrograms were used to identify the frequencies of instruments and find what notes were being played. Frequency filtering was also used to isolate certain instruments to make it easier to identify the notes. Through these methods, a basic music score for the bass of *Sweet Child O' Mine* and the guitar of *Comfortably Numb* were found, however the guitar of *Comfortably Numb* was much harder to isolate.

Introduction and Overview

Being able to decompose the frequencies depending on time in some data can be useful in analysing what the data is made up of. The data can be many things but in this case it is the audio data of 2 songs: *Sweet Child O' Mine* by Guns N' Roses and *Comfortably Numb* by Pink Floyd. The data being used are two 14 and 60 second clips of these two songs respectively. We will be using the Gabor transform to find the frequencies making up each song over time and use this to try and isolate different components of the songs as well as determining the notes played using the frequencies.

Theoretical Background

The Fourier transform can be used to decompose a function into the fundamental frequencies of sine and cosine functions that sum to make up the initial function. However, when performing the Fourier transform, time information is lost as if all frequencies occur at the same time. To get around this and keep some information about time, we will use what is known as the Gabor transform. The process of the Gabor transform involves having a filter function applied to the initial function and several Fourier transforms are taken of the function as the filter is moved across the region being analysed. The mathematical definition for the Gabor transform is as follows:

$$\widetilde{f}_g(\tau,k) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-ik}\,dt \qquad Eq.1$$

Where $f(t)$ is the initial function, $g(t-\tau)$ is the filter function centred at τ and $\widetilde{f}_g(\tau,k)$ is the Gabor transformed function. This then gives us the amount of each frequency contained at each time interval of the function. Much like the discrete Fourier transform, there is also a discrete form of the Gabor transform which we will be using where the function is now split up into finite components, the equation for which is the same as equation 1 except $k = m\omega_0$ and $\tau = nt_0$.

We can then use the discrete Gabor transform to create a spectrogram where which is the transformed function plotted with time on the x axis and frequency on the y axis. In the spectrogram, each vertical slice at each time represents the Fourier transform of the function when the filter is centred at that time. The spectrogram then allows us to investigate what the frequency components are at each time.

Algorithm Implementation and Development

The algorithm to analyse the songs was written in Matlab.

First, I created a vector of the wavenumbers scaled by $\frac{1}{len \quad of\ song\ clip\ in\ seconds}$ so that the frequencies will be in Hz. Then I set the steps for the filter to move across the function, and then looped through each step. During each iteration of the loop, I defined the filter function centred at the current step in time. The filter I used was a Gaussian function which has the equation:

$$e^{-a(t-t_0)^2} \qquad Eq.\ 2$$

Where t are the frequencies and t_0 is the time at which the filter is centred. I then multiplied the music clip data with the filter, and then took the Fourier transform of it at each time step. At the end I was left with a matrix containing the Gabor transform of the data containing the frequency data at each time.

I used this data to plot the spectrogram by having the time on the x axis, the frequencies on the y axis and then the Gabor transform of the function defining the colour of each point. I could then use this spectrogram to find the notes played in each clip by looking at the most present frequencies at each time.

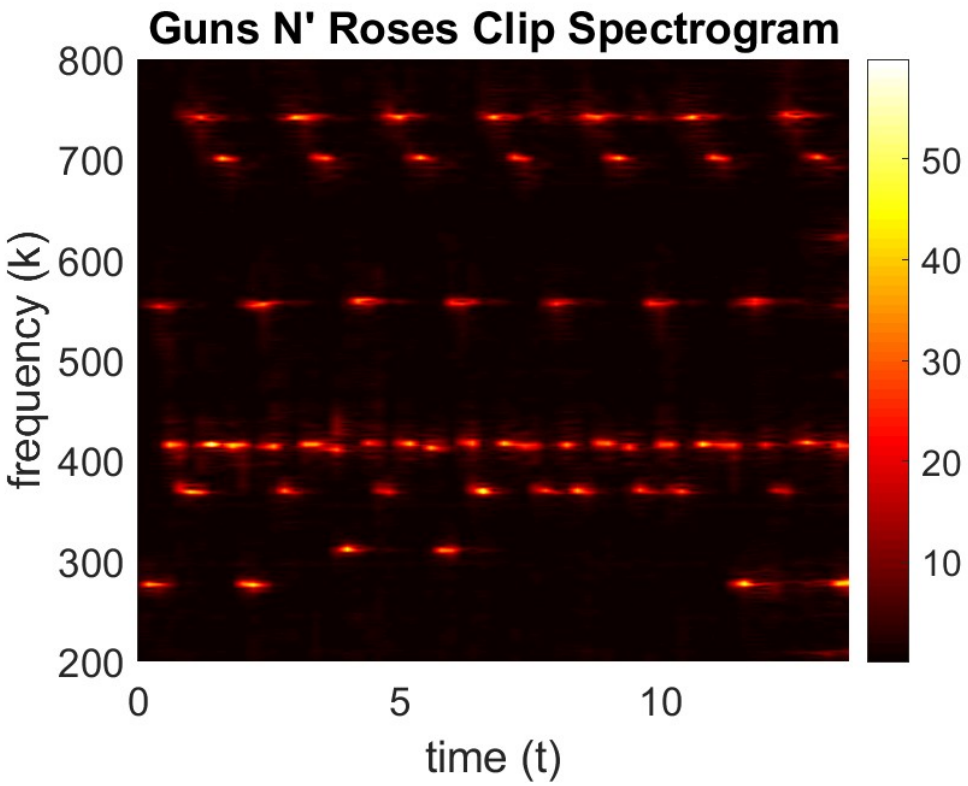
I repeated this for both the Guns N' Roses and Pink Floyd clip. The Guns N' Roses clip contained just one instrument so it was easy to determine the notes. However, for the Pink Floyd clip there are several different instruments playing at the same time so it was more difficult to find the notes being played by the guitar and the bass.

To overcome this I used frequency filtering to isolate the frequencies of the instruments to better determine the notes of each instrument. To do this I defined another Gaussian filter function as defined in equation 2, but instead of applying it to time I applied to frequency. At each time step iteration in the Gabor transform process, I multiplied the Fourier transform of the data at that specific time with the filter function, and then did that for all time steps as earlier.

This left me with the Gabor transform of the song but filtered around a certain frequency. The width and centre of this filter were adjusted to isolate both the bass and guitar of the Pink Floyd clip separately. So the spectrograms for them contained primarily the frequencies for the bass and guitar respectively, making it easier to find the notes of played by the instruments.

Computational Results

The spectrogram for the Guns N' Roses Clip with filter width $a = 50$ and time steps of 0.2 from the start to the end of the clip is as follows:



From this spectrogram we can create the music score for the guitar:

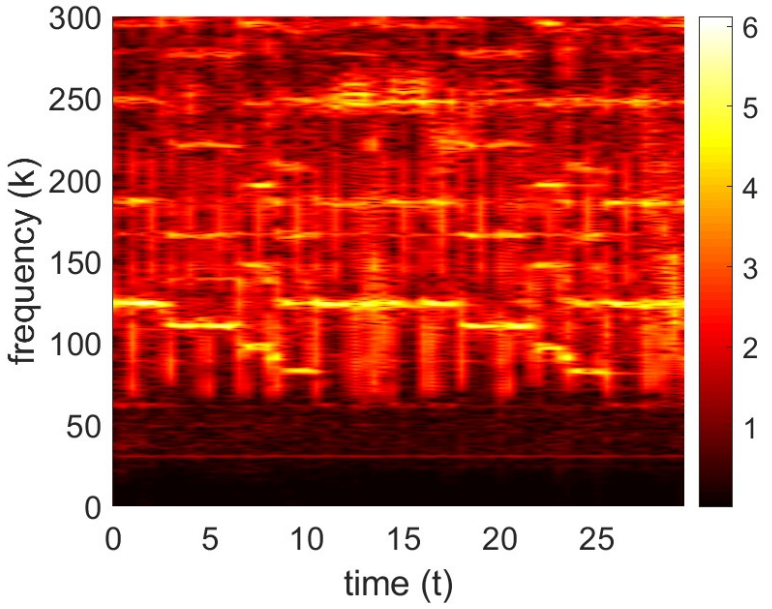
C#4 C#5 G#4 F#4 F#5 G#4 F5 G#4 x2

D#4 C#5 G#4 F#4 F#5 G#4 F5 G#4 x2

F#4 C#5 G#4 F#4 F#5 G#4 F5 G#4 x2

C#4 C#5 G#4 F#4 F#5 G#4 F5 G#4 x2

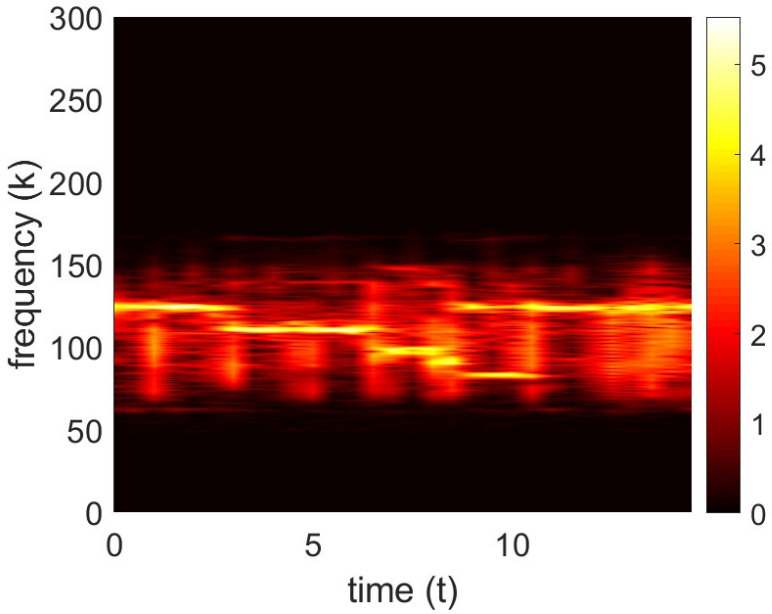
The spectrogram for the Pink Floyd Clip with filter width $a = 10$ and time steps of 0.5 from the start to the end of the clip is as follows (note that the spectrogram is zoomed into a specific time and frequency range to better see the bass notes, and that the log of the spectrogram was plotted):



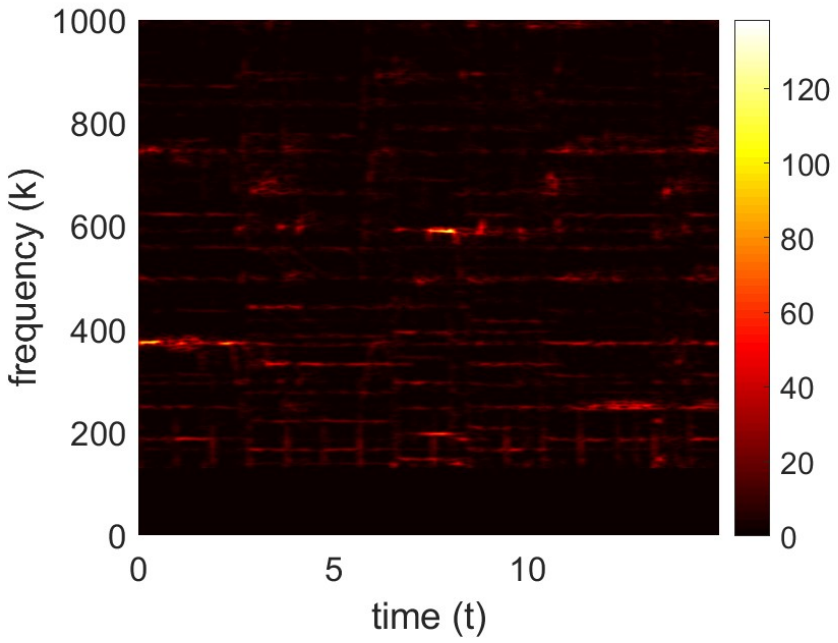
From this spectrogram we can create the music score for the bass:

B2 A2 G2 F#2 E2 B2 Which repeats for the entirety of the clip.

To isolate the bass, a frequency filter centred at $k = 100$ with width $a = 0.0001$ was used, giving the following spectrogram (again using the log of spectrogram):



To isolate the guitar, I applied a high pass filter so all frequencies below 130 Hz (the highest the bass frequency was in the clip) giving the following spectrogram:



From this we can see that the first two distinct notes are F#4 and E4.

However, the spectrogram is still very messy and it is difficult to tell which frequencies correspond to guitar notes. I tried applying a Gaussian filter but since the guitar notes have such a wide range and it is hard to tell what frequencies to centre the filter around, it did not make it any clearer.

Summary and Conclusion

From the spectrograms, we found the music score of the guitar in *Sweet Child O' Mine* by Guns N' Roses and the bass in *Comfortably Numb* by Pink Floyd. These were relatively easy to identify since they stood out from the rest of the spectrogram. The guitar in *Comfortably Numb* was much harder to identify however since the notes changed very quickly and there were many other overtones present that added lots of noise. It is much harder to use a frequency filter to isolate a frequency when it is difficult to identify where the filter should be centred.

Overall however, the method of using the Gabor transform and spectrograms to identify the frequencies and notes in a song does work and can be useful, but is significantly more useful and effective when the song is relatively simple and has few instruments or the instrument stands out in frequency space.

Appendix A: MatLab Functions Used

- **[y,Fs] = audioread(filename)** – reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.
    - https://au.mathworks.com/help/matlab/ref/audioread.html
- **L = length(X)** – returns the length of the largest array dimension in X.
    - https://au.mathworks.com/help/matlab/ref/length.html
- **Y = abs(X)** – Returns the absolute value of each element in array X. If X is complex then instead returns the complex magnitude.
    - https://au.mathworks.com/help/matlab/ref/abs.html
- **Y = exp(X)** – Returns the exponential e^x for each element in array X.
    - https://au.mathworks.com/help/matlab/ref/exp.html
- **Y = linspace(x1,x2,n)** – Generates n points between x1 and x2.
    - https://au.mathworks.com/help/matlab/ref/linspace.html
- **Y = fftn(X)** – Returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm.
    - https://au.mathworks.com/help/matlab/ref/fftn.html
- **Y = fftshift(X)** – Rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.
    - https://au.mathworks.com/help/matlab/ref/fftshift.html
- **Y = ifftn(X)** – Returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm.
    - https://au.mathworks.com/help/matlab/ref/ifftn.html
- **Y = ifftshift(X)** – Rearranges a zero-frequency-shifted Fourier transform Y back to the original transform output.
    - https://au.mathworks.com/help/matlab/ref/ifftshift.html
- **pcolor(C)** – Creates a pseudocolor plot using the values in matrix C.
    - https://au.mathworks.com/help/matlab/ref/pcolor.html
- **print(filename,formattype)** – Saves the current figure to a file using the specified file format.
    - https://au.mathworks.com/help/matlab/ref/print.html

Appendix B: MatLab Code

```
%% GNR Guitar Score
clear; close all; clc;

[y, Fs] = audioread('GNR.m4a');
trgnr = length(y)/Fs;
t = linspace(0,trgnr,length(y));
k = (1/trgnr)*[0:length(y)/2-1 -length(y)/2:-1];
ks = fftshift(k);
S = y';

a = 50;
tau = 0:0.2:trgnr;

for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2);
    Sg = g.*S;
    Sgt = fft(Sg);
    Sgt_spec(:,j) = fftshift(abs(Sgt));
end

pcolor(tau,ks,Sgt_spec)
shading interp
set(gca,'ylim',[200 800],'Fontsize',16)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
title("Guns N' Roses Clip Spectrogram")
print('GNR_spectrogram','-dpng')

%% Floyd Bass Score
clear; close all; clc;

[y, Fs] = audioread('Floyd.m4a');
y = y(1:(length(y)-1)/2);
trgnr = length(y)/Fs;
t = linspace(0,trgnr,length(y));
k = (1/trgnr)*[0:length(y)/2-1 -length(y)/2:-1];
ks = fftshift(k);
S = y';

a = 10;
tau = 0:0.5:trgnr;

for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2);
    Sg = g.*S;
```

```
    Sgt = fft(Sg);
    Sgt_spec(:,j) = fftshift(abs(Sgt));
end

pcolor(tau,ks,log(abs(Sgt_spec)+1))
shading interp
set(gca,'ylim',[0 300],'Fontsize',16)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
print('Floyd_spectrogram','-dpng')

%% Floyd Bass Isolation
clear; close all; clc;

[y, Fs] = audioread('Floyd.m4a');
y = y(1:(length(y)-1)/4);
trgnr = length(y)/Fs;
t = linspace(0,trgnr,length(y));
k = (1/trgnr)*[0:length(y)/2-1 -length(y)/2:-1];
ks = fftshift(k);
S = y';

Fa = 0.001;
Ftau = 100;
f = exp(-Fa*(ks - Ftau).^2);

Ga = 10;
Gtau = 0:0.5:trgnr;
for j = 1:length(Gtau)
    g = exp(-Ga*(t - Gtau(j)).^2);
    Sg = g.*S;
    Sgt = fftshift(fft(Sg));
    Sgtf = f.*Sgt;
    Sgt_spec(:,j) = Sgtf;
end

pcolor(Gtau,ks,log(abs(Sgt_spec)+1))
shading interp
set(gca,'ylim',[0 300],'Fontsize',16)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
print('Floyd_bass_spectrogram','-dpng')

%% Floyd Guitar Isolation
clear; close all; clc;
```

```
[y, Fs] = audioread('Floyd.m4a');
y = y(1:(length(y)-1)/4);
trgnr = length(y)/Fs;
t = linspace(0,trgnr,length(y));
k = (1/trgnr)*[0:length(y)/2-1 -length(y)/2:-1];
ks = fftshift(k);
S = y';

f = ks;
f(f < 130) = 0;
f(f > 0) = 1 ;

% Fa = 0.0001;
% Ftau = 200;
% f = exp(-Fa*(ks - Ftau).^2);

Ga = 100;
Gtau = 0:0.1:trgnr;
for j = 1:length(Gtau)
   g = exp(-Ga*(t - Gtau(j)).^2);
   Sg = g.*S;
   Sgt = fftshift(fft(Sg));
   Sgtf = f.*Sgt;
   Sgt_spec(:,j) = Sgtf;
end

pcolor(Gtau,ks,abs(Sgt_spec))
shading interp
set(gca,'ylim',[0 1000],'Fontsize',16)
colormap(hot)
colorbar
xlabel('time (t)'), ylabel('frequency (k)')
print('Floyd_guitar_spectrogram','-dpng')
```