# AMATH 482 Homework 2

Ariel Luo

Feb 10, 2020

**Abstract**

This homework is about analyze a portion of two rock songs by the use of Gabor Transform

## 1 Introduction and Overview

In this homework, we are trying to find the notes of the main melody from a portion of two different songs. The clips provided are Sweet Child O' Mine by Guns N' Roses and Comfortably Numb by Pink Floyd. Our first goal is to identify the musics notes by identifying the frequencies of the notes played by using Gabor Transform. Our second goal is to separate the bass and the guitar solo in Comfortably Numb.

## 2 Theoretical Background

As we learned from our lectures, the Fourier transform can decompose functions in space or time into functions in frequency. However, we are not able to directly see the correlation between time and frequency. The Gabor Transform, also called the Short Time Fourier Transform, can work with data in the time frame. It uses a time filter as a window and the window will slide through the whole signal. The Gabor transform can be represented using the equation:

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

Where g is the filter function centered at $\tau$, and a is the width of the window. If we use a very big a that covers the whole signal, the transform will be the same as the Fourier transform.

## 3 Algorithm Implementation and Development

The algorithm can be summarized and separated into two main parts:

**Algorithm 1:** Filtering for frequencies

Import data from `.m4a` files and read using audioread
Record time, data
Select window size and center
**for** $j = 1 : len(centers)$ **do**
  create filter and apply
  Perform Fast Fourier Transform
  Find peak where the frequency is located Find frequency and store in result
**end for**

I used a window width of 50 for both songs and divide up the time in steps of 0.4 and 0.5. Unlike the Fourier coordinates, the $2\pi$ was taken out in order to obtain the frequencies in Hertz.
When isolating the bass, I used a Gaussian filter centered at the frequencies I found after eliminating the higher frequencies.

| | frequency | note |
|---|---|---|
| 1 | 276.22 | C# |
| 2 | 553.39 | C# |
| 3 | 368.71 | F# |
| 4 | 416.12 | G# |
| 5 | 276.95 | C# |
| 6 | 414.15 | G# |
| 7 | 741.78 | F# |
| 8 | 701.73 | F |
| 9 | 311.47 | D# |
| 10 | 416.99 | G# |
| 11 | 742.58 | F# |
| 12 | 413.35 | G# |
| 13 | 557.10 | C# |
| 14 | 370.02 | F# |
| 15 | 416.99 | G# |
| 16 | 414.95 | G# |
| 17 | 556.23 | C# |
| 18 | 369.07 | F# |
| 19 | 416.70 | G# |
| 20 | 370.74 | F# |
| 21 | 415.97 | G# |
| 22 | 741.86 | F# |
| 23 | 701.80 | F |
| 24 | 277.38 | C# |
| 25 | 415.90 | G# |
| 26 | 743.02 | F# |
| 27 | 702.17 | F |
| 28 | 277.96 | C# |

Table 1: Notes and frequencies in Sweet O' Mine

# 4 Computational Results

The 28 notes I got for Sweet Child O' Mine are
C# C# F# G# C# G# F# F D# G# F# G# C# F# G# G# C# F# G# F# G# F# F C# G# F# F C#
See Table 1 for the frequencies of notes.

The first 30 notes I got for Comfortably Numb are F B B B B A A A E A A A A G D D F B E E B B B B B B B B B
See Table 2 for the frequencies of notes.

# 5 Summary and Conclusions

By applying what we learned in class on Gabor Transform, we successfully filtered out the frequencies from the original data. By checking the frequencies with the corresponding notes, we were able to obtain a rough music sheet(the musics scores) of the song. I also isolated the bass from guitar in Comfortably Numb.

# Appendix A MATLAB Functions

Add your important MATLAB functions here with a brief implementation explanation. This is how to make an **unordered** list:

| | frequency | note |
|---|---|---|
| 1 | 373.89 | F# |
| 2 | 374.77 | B |
| 3 | 124.25 | B |
| 4 | 124.55 | B |
| 5 | 124.54 | B |
| 6 | 123.58 | A |
| 7 | 111.50 | A |
| 8 | 111.80 | A |
| 9 | 332.03 | E |
| 10 | 111.92 | A |
| 11 | 110.55 | A |
| 12 | 110.33 | A |
| 13 | 110.57 | A |
| 14 | 97.88 | G |
| 15 | 590.33 | D |
| 16 | 589.79 | D |
| 17 | 91.11 | F# |
| 18 | 125.34 | B |
| 19 | 83.03 | E |
| 20 | 82.56 | E |
| 21 | 125.29 | B |
| 22 | 124.05 | B |
| 23 | 123.36 | B |
| 24 | 123.70 | B |
| 25 | 252.41 | B |
| 26 | 123.90 | B |
| 27 | 124.92 | B |
| 28 | 247.34 | B |
| 29 | 124.35 | B |
| 30 | 123.70 | B |

Table 2: First 30 Notes and frequencies in Comfotably Numb

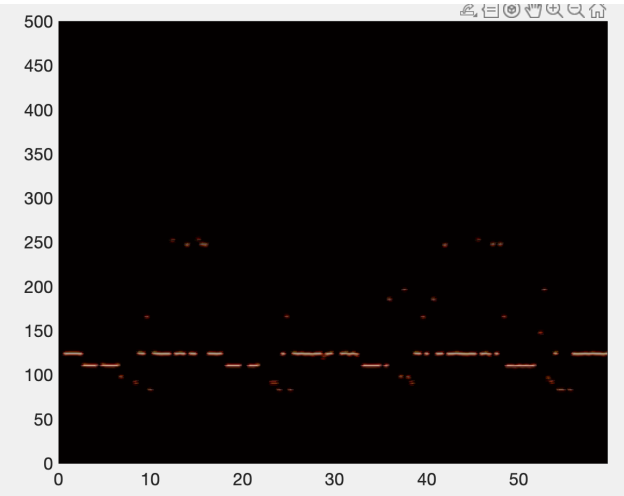


Figure 1: The spectrogram of the bass.

- `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.
- `[I1,I2,...,In]` = ind2sub(sz,ind) returns n arrays I1,I2,...,In containing the equivalent multidimensional subscripts corresponding to the linear indices ind for a multidimensional array of size sz.
- `k = find(X)` returns a vector containing the linear indices of each nonzero element in array X.

# Appendix B MATLAB Code

```
clear all; close all; clc;
%figure(1)
[y, Fs] = audioread('GNR.m4a');
y = y';
trgnr = 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 = trgnr; 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);
a = 50;
tau = 0:0.5:trgnr;
f = zeros(length(tau),1);
for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2);
    yg = g.*y;
    ygt = fft(yg);
    ygtshift = abs(fftshift(ygt));
    x=ind2sub(size(ygtshift),find(ygtshift == max(ygtshift)));
    f(j)=ks(x(2));
end
```

Listing 1: Code for HW2-GNR

```
clear all; close all; clc;
%figure(1)
[y, Fs] = audioread('Floyd.m4a');
y = y';
trgnr = length(y)/Fs; % record time in seconds
L = trgnr; n = length(y);
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (1/L)*[0:(n/2) -n/2:-1]; ks = fftshift(k);
a = 50;
tau = 0:0.4:trgnr;
f = zeros(length(tau),1);

for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2);
    yg = g.*y;
    ygt = fft(yg);
    ygtshift = abs(fftshift(ygt));
    x=ind2sub(size(ygtshift),find(ygtshift == max(ygtshift)));
    f(j)=ks(x(2));
end

for j = 1:length(f)
   if(f(j)>300)
       f(j)=0;
   end
end

filtered_f = zeros(length(tau),1);
for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2);
    filter = exp(-0.2*((k - f(j)).^2));
    nyg = fft(y);
    nygt2 = filter.*nyg;
    inverse = abs(ifft(nygt2));
    inverse = g.*inverse;
    nygt = fft(inverse);
    nygt_spec(:,j)=fftshift(abs(nygt));
    nygtshift = abs(fftshift(nygt));
    ny=ind2sub(size(nygtshift),find(nygtshift == max(nygtshift)));
    filtered_f(j)=ks(ny);
end

  figure(1)
  pcolor(tau,ks,nygt_spec)
  shading interp
  set(gca,'ylim',[0,500],'Fontsize',16)
  colormap(hot)
```

Listing 2: Code for HW2-Floyd