AMATH 482 Homework 2

David Warne

February 10, 2021

Abstract

The Gabor Transform is a powerful tool that can be used for the filtering of frequencies. This method is useful for extracting important information, whether that be patterns, hidden data or useful information that we want to study further.

1 Introduction and Overview

In this assignment, I am essentially given two audio files, one containing music from the popular Gun's N Roses song 'Sweet Child O' Mine', and one containing music from the popular Pink Floyd song 'Comfortably Numb'. Once I extract the amplitude data from these two audio files, I am tasked with implementing a Gabor transform to isolate the music score of each, which allows us to better understand each of the musical pieces!

2 Theoretical Background

In order to better understand the algorithms implemented in this assignment, it is important to talk about the Gabor Transform and why we are using it in this context. From the textbook, [1], we know that the Gabor transform is also referred to as the *short-time fourier transform.* It gets this name because the implementation of a Gabor Transform is essentially a fourier transform that is preceded by an elementwise Gaussian multiplication. This Gaussian can also be referred to as our *window* in time that we are using to look at our signal. Once the transform is complete, we will have a much better understanding of the frequencies within a signal over a given period of time.

Below is our Gabor Transform that I implement to extract data about the frequency for both of the provided songs:

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

Where f(t) refers to the vector of extracted signal data, g(t) refers to the Gaussian, and the e function is the fourier transform. My Gaussian can be seen below:

$$g(t-\tau) = e^{-a*(t-\tau)^2} \tag{2}$$

Where a refers to the size of the window and and τ is the center of the window at any given time t.

3 Algorithm Implementation and Development

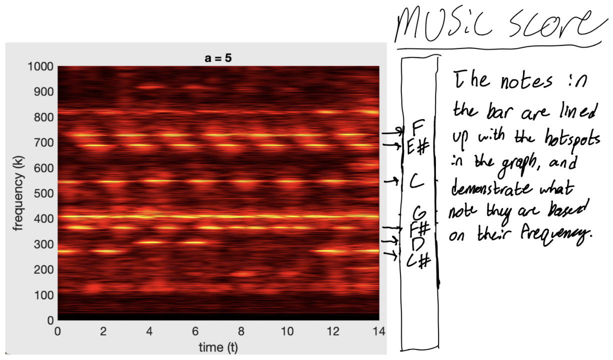
There are essentially two algorithms. One that is used to extract the music score from each song and one that is used to isolate the a given instrument from any signal. See Algorithm 1 for extracting the music score, and see Algorithm 2 for isolating an instrument.

Algorithm 1: Extracting the Music Score

```
Extract data from either GNR.m4a or Floyd.m4a using the audioread() function
Initialize all important variables (i.e. bounds (L, ks, t, $\tau$), and an empty matrix to store values for the spectrogram)
for j = 1:length($\tau$) do
    Calculate the Gaussian
    Use element-wise multiplication to multiply the Gaussian by the signal function extracted from the audio data
    Transform the data into frequency space.
    Shift the absolute value of the fourier space values using fftshift() and save them into the empty spectrogram matrix.
end for
Plot the spectrogram using pcolor() and set the colormap to hot.
NOTE: you may want to plot log(abs(s)+1) to better visualize the data, and make sure to adjust the axis according to the frequencies you are searching for.
```

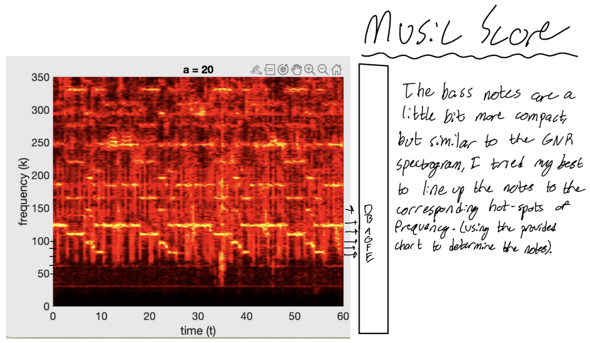
Algorithm 2: Isolating a Specific Frequency

```
Extract data from either GNR.m4a or Floyd.m4a using the audioread() function
Initialize all important variables (i.e. bounds (L, ks, t, $\tau$), and an empty matrix to store values for the spectrogram)
Use the filterDesigner toolbox to create a filter to isolate certain frequencies in the data. For the Floyd data this would ideally be a lowpass filter since we are trying to isolate the bass instrument.
for j = 1:length($\tau$) do
    Calculate the Gaussian
    Use element-wise multiplication to multiply the Gaussian by the signal function extracted from the audio data
    Transform the data into frequency space.
    Apply designated filter to the frequency data to localize the data on the specified frequency.
    Add the filtered data to the empty spectrogram matrix.
end for
Plot the spectrogram using pcolor() and set the colormap to hot.
NOTE: you may want to plot log(abs(s)+1) to better visualize the data, and make sure to adjust the axis according to the frequencies you are searching for.
```



The notes in the bar are lined up with the hotspots in the graph, and demonstrate what note they are based on their frequency.

Figure 1: Pictured is the music score spectrogram I got for the GNR data for the guitar line.



The bass notes are a little bit more compact, but similar to the GNR spectrogram, I tried my best to line up the notes to the corresponding hot-spots of frequency. (using the provided chart to determine the notes).

Figure 2: Pictured is the music score spectrogram I got for the Floyd data for the bass line.

4 Computational Results

- Part 1: The music score from the GNR data can be seen in Figure 1. The music score from the Floyd data can be seen in Figure 2.
- Part 2: I was unable to figure this part out, but here was my thought process on how I tried to get it to work. I know that we are supposed to apply a filter in the frequency domain, so I tried using the find() function in MATLAB to to find indices of elements that were within a specific frequency range,and then altering the vector to only include those values, but kept running into an issue working with the complex numbers. I didn't know how to isolate frequencies when working with complex numbers. I then tried to use the filterDesigner to create a low pass filter that would filter between 50 and 150 Hz, but was unable to figure out how to correctly use the software in a way that would allow me to properly apply the filter. I kept getting matrix dimension errors because I couldn't figure out how to create the filter so that it would match the dimensions of my transformed matrix.
- Part 3: For similar reasons of being unable to figure out how to isolate certain frequency ranges like in part 2, I was also stumped by the same problems for this part. I've included the function for my lowpass filter in the appendices with the rest of my MATLAB code, so any and all feedback would be much appreciated because it was very frustrating not being able to figure this out!!

5 Summary and Conclusions

In conclusion, I think the most important thing to take away from this assignment is how powerful gabor transforms and the filtering of signal data can be. Part 1 allowed us to take an audio file for a song and determine the notes that compose the musical score. Part's 2 and 3 allowed us to use filtration to isolate a specific instrument from an audio track. This is obviously some very powerful calculation, and it just goes to show how applicable practices like fourier/gabor transforms can be when it comes to breaking down and finding patterns in real world data!

References

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

Appendix A MATLAB Functions

Here are the main functions that I used in this assignment:

- `[y, Fs] = audioread(audio.m4a)` returns the data of an input audio file.
- `fft()` returns a fourier transform of the data so that we can work with it in frequency space.
- `fftshift()` shifts the zero-frequency data to the center of our array so that we can visualize the data more clearly.
- `pcolor()` is used to plot the spectrogram.

Appendix B MATLAB Code

Here is the MATLAB code that was used to produce the information in this assignment.

```
%% PART 1
clear all; close all; clc;

% GNR data extraction and viewing the frequencies
figure(1);
[y, Fs] = audioread('GNR.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');

L = ceil(tr_gnr);
n = length(y);
t2 = linspace(0,L,n+1); t = t2(1:n);
% Different scaling on k, for hertz we need to ditch the 2pi
k = (1/L)*[0:n/2-1 -n/2:-1]; % Notice the 1/L instead of 2*pi/L
ks = fftshift(k);

a = 5;
tau = 0:.1:L; % step through the song with iteration
Ygt_spec = []; % Clear at each loop iteration
for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2); % Gabor Transform
    Yg = g.*y'; % need to transpose y so that the dimensions match
    Ygt = fft(Yg);
    Ygt_spec(:,j) = fftshift(abs(Ygt));
end

% spectrogram for visualization
figure(2);
pcolor(tau,ks,log(abs(Ygt_spec) + 1))
shading interp
set(gca,'ylim',[0 1000],'Fontsize',16)
colormap(hot)
xlabel('time (t)'), ylabel('frequency (k)')
title(['a = ',num2str(a)],'Fontsize',16)
```

Listing 1: Code for Part 1 - GNR

```
%% PART 1 Floyd
clear all; close all; clc;

% Floyd data extraction and viewing the frequencies
figure(1);
[y, Fs] = audioread('Floyd.m4a');
y = y(1:end-1); % make the dataset have an even number of elements
tr_floyd = length(y)/Fs; % record time in seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Comfortably Numb');

n = length(y);
song_length = ceil(tr_floyd);
L = song_length;
a = 20;
k = (1/L)*[0:n/2-1 -n/2:-1]; % Notice the 1/L instead of 2*pi/L
ks = fftshift(k);

Ygt_spec = [];
t2 = linspace(0,L,n+1); t = t2(1:n);
tau = 0:.5:L; % step through the song with iteration

for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2); % Gabor Transform
    Yg = g.*y'; % need to transpose y so that the dimensions match
    Ygt = fft(Yg);

    Ygt_spec(:,j) = fftshift(abs(Ygt));
end

figure(2);
% Make spectrogram for Floyd data
pcolor(tau,ks,log(abs(Ygt_spec)+1))
shading interp
set(gca,'ylim',[0 350],'Fontsize',16)
colormap(hot)
xlabel('time (t)'), ylabel('frequency (k)')
title(['a = ',num2str(a)],'Fontsize',16)
```

Listing 2: Code for Part 1 - Floyd

```
%% PART 2

% trynig to isolate the bass using a filter in frequency space
clear all; close all; clc;

% Floyd data extraction and viewing the frequencies
figure(1);
[y, Fs] = audioread('Floyd.m4a');
y = y(1:end-1); % make the dataset have an even number of elements
tr_floyd = length(y)/Fs; % record time in seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Comfortably Numb');

% initialize variables

n = length(y);
song_length = ceil(tr_floyd);
L = song_length;
a = 20;
k = (1/L)*[0:n/2-1 -n/2:-1]; % Notice the 1/L instead of 2*pi/L
ks = fftshift(k);

Ygt_spec = [];
t2 = linspace(0,L,n+1); t = t2(1:n);
tau = 0:.5:L; % step through the song with iteration
Hd = bass_filter;
filt = Hd.sosMatrix;

for j = 1:length(tau)
    g = exp(-a*(t - tau(j)).^2); % Gabor Transform
    Yg = g.*y'; % need to transpose y so that the dimensions match
    Ygt = fft(y);

    % not sure how to properly create the filter and the method of using
    % find() was unsuccessful /:
    Ygt_filtered = filt.*Ygt;

    Ygt_spec(:,j) = fftshift(abs(Ygt));
end

% Spectrogram for visualization
pcolor(tau,ks,log(abs(Ygt_spec)+1))
shading interp
set(gca,'ylim',[0 350],'Fontsize',16)
colormap(hot)
xlabel('time (t)'), ylabel('frequency (k)')
title(['a = ',num2str(a)],'Fontsize',16)
```

Listing 3: Code for Part 2

```matlab
function Hd = bass_filter
%BASS_FILTER Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.9 and Signal Processing Toolbox 8.5.
% Generated on: 10-Feb-2021 22:31:49

% Butterworth Lowpass filter designed using FDESIGN.LOWPASS.

% All frequency values are in Hz.
Fs = 48000;  % Sampling Frequency

Fpass = 50;          % Passband Frequency
Fstop = 150;         % Stopband Frequency
Apass = 1;           % Passband Ripple (dB)
Astop = 80;          % Stopband Attenuation (dB)
match = 'stopband';  % Band to match exactly

% Construct an FDESIGN object and call its BUTTER method.
h  = fdesign.lowpass(Fpass, Fstop, Apass, Astop, Fs);
Hd = design(h, 'butter', 'MatchExactly', match);

% [EOF]
```

Listing 4: Code for Lowpass Filter