AMATH 482 Homework 2

Jiaqi Su

February 10, 2021

Abstract

Different notes played by instruments are characterized by different frequencies in a music scale. Besides, the range of notes capable to play varies among different instruments. Therefore, identifying dominant frequencies in a audio piece signal can give information about the notes of the melody played by a certain instrument. The goal of this project is to separate the solo of guitar and bass in the sound clips of two famous rock n roll songs: *Sweet Child O' Mine* by Gun N' Roses and *Comfortably Numb* by Pink Floyd, and to identify the notes played by each of the instruments.

1 Introduction and Overview

The goal of this problem is to reproduce the notes played by guitar in *Sweet Child O' Mine* and that of guitar and bass in *Comfortably Numb*. The main difficulty is to separate the desired instrument using filters. The data we read from the audio clip is discrete data sampled at some frequency, meaning the certain numbers of sample points we take in a second. Given this discrete sample data, we first need to plot the an overview of frequency over time to get an idea of the possible notes played. After that we can filter out overtones or parts not played by our target instrument to isolate the solo of the target instrument.

2 Theoretical Background

Fourier transform is widely used in signal processing to decompose signal into function of different frequencies. However, the Fourier transform only provide information about the existence of frequencies while losing information about the how different frequencies appear and change over time. To obtain information both in frequency and time domain, we introduce the Gabor Transform, or short-time Fourier transform (STTF), which creates filter window at centered at each time point and slides the window through the time range to collect information in both time domain and frequency domain.

The Gabor Transform of function $f(t)$ at time τ is defined by

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

where $f_g(\tilde{\tau,k})$ gives us the information about frequency near time τ and g is the filter function of our choice. Therefore, as we shift τ across the time range, we obtain frequency information at different time.

For the filter function g, the commonly-used choice is the Gaussian function:

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

The parameter a determines the width(standard deviation) of the filter while the parameter τ is the center of the sliding time window. The width parameter a is essential in determining the trade-off of time and frequency information. A greater a represents a larger window to recover more information about the frequency, and at the same time leads to fewer time slides and less information about the time. So having an appropriate window can maximize the value of the information provided by the Gabor Transform.

As we have a discrete version of Fourier Transform, we also have a discrete version of Gabor Transform to solve problems in the real world, where the data is a set of discrete points instead of a continuous function $f(x)$. Consider a discrete set of frequency:

$$k = m\mu_0 \tag{3}$$
$$\tau = nt_0 \tag{4}$$

where m and n are integers and μ_0 and t_0 are positive constants. Then the discrete Gabor Transform is:

$$f_g(\tilde{m}, n) = \int_{-\inf}^{\inf} f(t)g(t - mt_0)e^{-i\mu_0 t}dt \tag{5}$$

3 Algorithm Implementation and Development

3.1 Gun N' Roses Guitar Notes

The first step is to decompose the frequencies in the signal of the song clip across time, so that we can match the frequencies with notes in music scale. To implement this algorithm, we need following developments:

```
Algorithm 1: GNR guitar notes
    Import sampled data $y$ from audio clip GNR.m4a
    Set up the time and Fourier domain
    for $j$ = 1:time step:time range $L$ do
        Create time filter $g$ centered at current time point
        Apply the filter $g$ to sampled data $y$
        Using fft to switch to the frequency domain
        Save frequency information at current time into $j$ th column of spectrogram matrix
    end for
    Plot the spectrogram and adding labels of significant notes
```

1. When setting up the time and frequency domain, the frequency should be re-scaled to $2\pi/L$ where L is our spatial domain since FFT assumes 2π periodic signal. And when plotting the spectrogram, to display the frequencies in Hertz, the frequency should be scaled back by a factor $1/2\pi$.

2. MATLAB implementation gives the FFT result in a shifted version such the nonzero frequency is at left and the negative part is at the right. Therefore, fftshift is needed each time we use FFT to shift the zero frequency back to the center as a standard form.

3. Parameter a in the filter function of the Gabor transform determines the width of the filter and therefore affects the the filtered results. In this way, different value widths should be tried to have a clear overview of the frequencies over time.

3.2 Comfortably Numb Bass and Guitar Notes

Compared to *Sweet Child O' Mine* which only has guitar in the song clip, *Comfortably Numb* has guitar and bass playing different notes in the song clip. Therefore, we need to seperate the two instruments based on their frequency range and clean the overtones for better identifications of notes. The algorithms of finding bass or guitar notes are the same.

To implement this algorithm, we need following developments:

1. Parameter a and τ in the filter function of the Gabor transform and the regular Gaussian filter both determine the width of the filter and influence results. An appropriate a in the Gabor transform is needed to have a clear overview of the frequencies over time, and an appropriate τ is needed in the filter using fundamental functions to clean the overtones without eliminating target frequencies of notes played in the song.

Algorithm 2: floyd separate notes

```
Import sampled data $y$ from audio clip Floyd.m4a
Set up the time and Fourier domain
for $j$ = 1:time step:time range $L$ do
    Create time filter $g$ centered at current time point
    Apply the filter $g$ to sampled data $y$
    Using fft to switch to the frequency domain
    Save frequency information at current time into $j$ th column of spectrogram matrix $ygt_spec$
end for
Create and apply an array to filter out frequencies that exceed the range of bass $(20 - 260Hz)$ or the range of guitar $(300 - 1000Hz)$
for $j$ = 1:time step:time range $L$ do
    Extract the signal data $ygt_spec(:, j)$ at $j$ th time point
    Find the fundamental frequency where the signal data has maximum at current time.
    Create a Gaussian filter using the fundamental frequency and apply to the signal data $ygt_spec(:, j)$
end for
Plot the spectrogram and adding labels of significant notes
```

4 Computational Results

4.1 Notes of Guitar in *Sweet Child O' Mine*

Using Algorithm 1, the frequencies and corresponding notes played by guitar in *Sweet Child O' Mine* is shown in the graph 1.

4.2 Notes of Bass in *Comfortably Numb*

Using Algorithm 2, the frequencies and corresponding notes played by bass in Comfortably Numb is shown in the graph 2.

4.3 Notes of Guitar in *Comfortably Numb*

Using Algorithm 2, the frequencies and corresponding notes played by guitar in *Comfortably Numb* is shown in the graph 3.

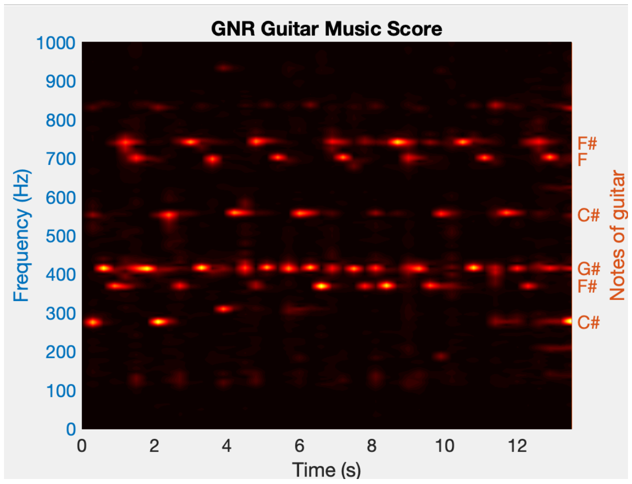

Figure 1: Here is a picture of the spectrogram with notes played by guitar *Sweet Child O' Mine*

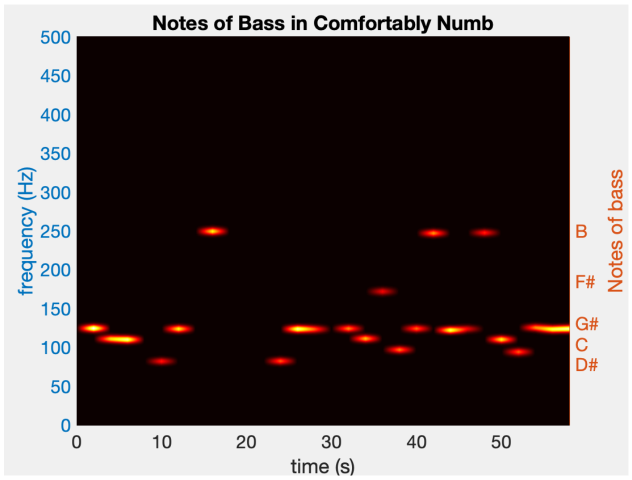

Figure 2: Here is a picture of the spectrogram with notes played by bass in *Comfortably Numb*

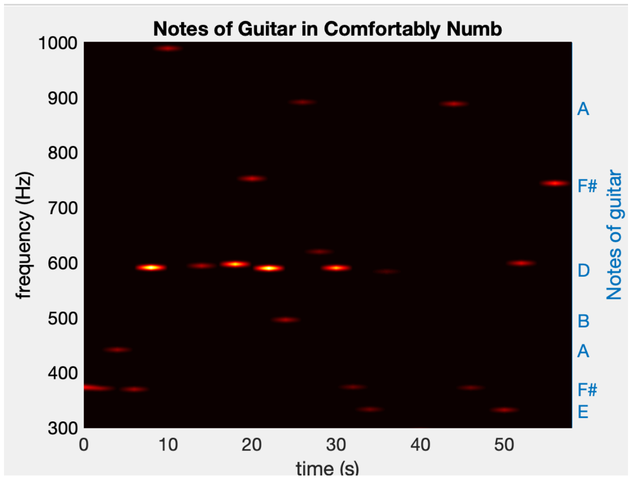

Figure 3: Here is a picture of the spectrogram with notes played by guitar in *Comfortably Numb*

5 Summary and Conclusions

In conclusion, the Gabor Transform provides information both in the time domain and frequency domian through using FFT across the time range. And applying appropriate filters based on the fundamental frequency (certain note) at each time point helps up filter out noise of overtones. Finally by matching frequencies to the music scale, we can identify the notes in the audio clips.

Appendix A MATLAB Functions

MATLAB functions used in the implementations of Algorithms.

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`. In Algorithm 2, this is used to set up the time and frequency domain of our problem.

- `y = fftshift(x)` returns a shifted version of `x` through swapping the left and right halves of `x`. It is used to visualize the fast Fourier transform to a spectrum with 0 frequency at the center. As in Algorithm 2 and algorithm 1, this is used after each fft to shift 0 frequency back to center as a standard form.

- `y = fft(x)` applies fast Fourier transform on the vector `x`. The output is complex and all nonzero entries are put before the negative ones.
- `y = abs(x)` returns the absolute value of `x`. If `x` is a complex number, `y` is the complex modulus of `x`. In Algorithm 1, the FFT returns complex number so we need to take complex modulus using `abs()` before finding maximum to locate the fundamental frequency.
- `[M,I] = max(x)` returns maximum entry value `M` in `x` and the index `I` of that maximum value. In Algorithm 1, `max()` is used to find the fundamental frequency of the notes played by certain instrument as each time point.
- `[M,I] = min(x)` returns minimum entry value `M` in `x` and the index `I` of that minimum value. In Algorithm 1, `min()` is used to create an array filter to filter out frequencies outside the range of desired instrument.
- `set(gca,'ytick', v, 'yticklabel',label)` adds labels stored in *label* to y axis at desired points store in v.

Appendix B MATLAB Code

```
% Clean workspace
clear all; close all; clc

% Import and convert audio pieces to vectors (GNR)
[y, Fs] = audioread('GNR.m4a');
tr_gnr = length(y)/Fs; % record time in seconds


% Set up time and Fourier domain (GNR)
L = tr_gnr;
n = length(y);
t2 = linspace(0,L,n+1);
t = t2(1:n);
k = (2*pi/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

% Apply Gabor filter on GNR using Gaussian function
tau = 0:0.3:L;
a = 800;
ygt_spec = zeros(n,length(tau));
for j = 1:length(tau)
   g = exp(-a*(t - tau(j)).^2); % Window function
   yg = g.*y';
   ygt = fft(yg);
   ygt_spec(:, j) = fftshift(abs(ygt))/(2*pi);
end

figure(1)
yyaxis left
pcolor(tau, ks/(2*pi),ygt_spec)
shading interp
set(gca,'ylim',[0 1000],'Fontsize',16)
colormap(hot)
xlabel('Time (s)'), ylabel('Frequency (Hz)')
title('GNR Guitar Music Score')

yyaxis right
pcolor(tau,ks/(2*pi),ygt_spec)
shading interp
set(gca,'ylim',[0 1000],'Fontsize',16)
colormap(hot)
ylabel('Notes of guitar')
set(gca,'ytick',[277,370,415,554,698,740])
set(gca,'Yticklabel',{'C#','F#','G#','C#','F','F#'})
```

Listing 1: Code for GNR *Sweet Child O' Mine*

```
% Clean workspace
clear all; close all; clc
% Import and convert audio pieces to vectors (Floyd)
[y2, Fs2] = audioread('Floyd.m4a');
tr_floyd = length(y2)/Fs2;
y2s = y2(1:length(y2)-1);
% Set up time and Fourier domain (Floyd)
L2 = tr_floyd; n2 = length(y2s);
t2s = linspace(0,L2,n2+1); ts = t2s(1:n2);
k2 = (2*pi/L2)*[0:n2/2-1 -n2/2:-1]; ks2 = fftshift(k2);
% Apply Gabor filter on Floyd using Gaussian function
tau2 = 0:2:L2; a2 = 1000;
ygt2_spec = zeros(n2,length(tau2));
yg2 = zeros(n2,length(tau2));
for j = 1:length(tau2)
   g = exp(-a2*(ts-tau2(j)).^2); % Window function
   yg2(:,j) = g.*y2s';
   ygt2 = fft(yg2(:,j));
   ygt2_spec(:,j) = fftshift(abs(ygt2));
end
figure(1)
pcolor(tau2,ks2/(2*pi),ygt2_spec/(2*pi))
shading interp
set(gca,'ylim',[0 800],'Fontsize',16)
colormap(hot)
xlabel('time (s)'), ylabel('frequency (Hz)') title('Floyd Music Score')
%% Isolate bass in Floyd
ygtf2_spec = zeros(n2,length(tau2));
ksfilter = abs(ks2/(2*pi)-260);
[kmin,kind] = min(ksfilter);
arrayfilter = ones(n2,1);
arrayfilter(kind+1:end,1) = 0;
for j = 1:length(tau2)
    bassrange = arrayfilter.*ygt2_spec(:,j);
    [m,ind] = max(bassrange);
    k0 = abs(ks2(ind));
    taug2 = 0.001;
    filter = exp(-taug2*(ks2-k0).^2);
    ygtf2_spec(:,j) = filter'.*bassrange;
end
% Plot the fig
figure(2)
yyaxis left
pcolor(tau2,ks2/(2*pi),ygtf2_spec/(2*pi))
shading interp
set(gca,'ylim',[0 500],'Fontsize',16)
colormap(hot)
xlabel('time (s)'), ylabel('frequency (Hz)') title('Notes of Bass in Comfortably Numb')

yyaxis right
pcolor(tau2,ks2/(2*pi),ygtf2_spec/(2*pi))
shading interp
set(gca,'ylim',[0 500],'Fontsize',16)
colormap(hot)
ylabel('Notes of bass')
set(gca,'ytick',[78,104,131,185,250])
set(gca,'Yticklabel',{'D#','C','G#','F#','B'})
```

Listing 2: Code for bass notes in Floyd *Comforably Numb*

```
% Clean workspace
clear all; close all; clc
% Import and convert audio pieces to vectors (Floyd)
[y2, Fs2] = audioread('Floyd.m4a');
tr_floyd = length(y2)/Fs2;
y2s = y2(1:length(y2)-1);
% Set up time and Fourier domain (Floyd)
L2 = tr_floyd; n2 = length(y2s);
t2s = linspace(0,L2,n2+1); ts = t2s(1:n2);
k2 = (2*pi/L2)*[0:n2/2-1 -n2/2:-1]; ks2 = fftshift(k2);
% Apply Gabor filter on Floyd using Gaussian function
tau2 = 0:2:L2; a2 = 1000;
ygt2_spec = zeros(n2,length(tau2));
yg2 = zeros(n2,length(tau2));
for j = 1:length(tau2)
   g = exp(-a2*(ts-tau2(j)).^2); % Window function
   yg2(:,j) = g.*y2s';
   ygt2 = fft(yg2(:,j));
   ygt2_spec(:,j) = fftshift(abs(ygt2));
end
figure(1)
pcolor(tau2,ks2/(2*pi),ygt2_spec/(2*pi))
shading interp
set(gca,'ylim',[0 800],'Fontsize',16)
colormap(hot)
xlabel('time (s)'), ylabel('frequency (Hz)') title('Floyd Music Score')
%% Isolate guitar
ygtf2_spec = zeros(n2,length(tau2));
ksfilter = abs(ks2/(2*pi)-250);
[kmin,kind] = min(ksfilter);
arrayfilter = zeros(n2,1);
ksfilter2 = abs(ks2/(2*pi)-1000);
[kmin2,kind2] = min(ksfilter2);
arrayfilter(kind+1:kind2,1) = 1;
for j = 1:length(tau2)
    guitarrange = arrayfilter.*ygt2_spec(:,j);
    [m,ind] = max(guitarrange);
    k0 = abs(ks2(ind));
    taug2 = 0.001;
    filter = exp(-taug2*(ks2-k0).^2);
    ygtf2_spec(:,j) = filter'.*guitarrange;
end
% plot the guitar notes
figure(3)
pcolor(tau2,ks2/(2*pi),ygtf2_spec/(2*pi))
shading interp
set(gca,'ylim',[300 1000],'Fontsize',16)
colormap(hot)
xlabel('time (s)'), ylabel('frequency (Hz)') title('Notes of Guitar in Comfortably Numb')

yyaxis right
pcolor(tau2,ks2/(2*pi),ygtf2_spec/(2*pi))
shading interp
set(gca,'ylim',[300 1000],'Fontsize',16)
colormap(hot)
ylabel('Notes of guitar')
set(gca,'ytick',[330,370,440,494,587,740,880])
set(gca,'Yticklabel',{'E','F#','A','B','D','F#','A'})
```

Listing 3: Code for guitar notes in Floyd *Comforably Numb*