Amath 482 HW2
Leqi Wang

Gabor Transform and Its Application in Audio Filtering

Abstract: Using Gabor transform to filter out fundamental frequency of different instruments in two sound clips, one is 14 seconds and another is 60 seconds.

Section I. Introduction and Overview

Gabor transformation is useful to recover information in time domain which we lost by using Fourier transformation. In this assignment, we applied Gabor transform to filter out the fundamental frequencies for different instrument in two sound clips. One is clip of the songs *Sweet Child O' Mine* by Guns N' Roses and *Comfortably Numb* by Pink Floyd.

Section II. Theoretical Background

1. Fourier transform and Gabor Transform:
   a) Drawbacks of Fourier transform: when transforming the time signal, we'll lose the information in time domain. We will only know what frequencies are present in that clip, but do not know where it occurred.
   b) Gabor transform: preserves the information in the time domain by localize a signal in both time and frequency domains.
   c) Equation of Fourier and Gabor:
      i. Fourier transform:

$$F(k) = \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{\infty} e^{-i \cdot kx} \cdot f(x) \cdot dx$$

      ii. Gabor Transform:

$$G[f] = \int_{-\infty}^{\infty} f(\tau)\bar{g}(t - \tau) \cdot e^{-i\omega\tau}\, d\tau$$

      where tau is the variable that allows us to slide across time dimension, and g(t – tau) should be a function with center tau and allow some range of filtering.

Section III. Algorithm Implementation and Development

To acquire the music score in GNR clip (figure1, sectionIV), I start with Gabor transform to make a spectrogram. I divided the clip into small steps with step size(tau) equaling 0.5 and applied a gaussian function with a equaling 1000 as my Gabor filter. After applying Gabor filter in the time domain for each snapshot of tau, I applied

Fourier transform. In the frequency domain, I found the max value in the frequency domain as the center frequency, which refers to the sound frequency that is dominating that period of time, and apply another gaussian filter centered around that frequency to denoise the overtones. Afterwards, I store that snapshot of frequency spectrum in the variable Sgt_spec, and in the end use it to plot the spectrum using a hot map.

For part b (figure2, sectionIV), I divided the Floyd clip into separate fragments, or the calculation is too large for MATLAB to handle. I apply the Gabor transform just like in part a, but since there are different instruments playing and we are required to find the music score for bass, I applied an filter called instrument_filter on the signal so that any signal over 300 will not be recorded. It is realized by creating an array of 0 and 1, so that any frequency that is over 300 will be multiplied by zero.

In order to get the note by finding fundamental frequency, I store the center frequency for each tau period of time, sort them, and match them with the fundamental frequency chart given in Assignment2.

Same for part c (figure3, sectionIV), to filter out frequency of guitar with frequency range from 250 to 1000 Hertz, I applied the instrument_filter(though the range is different from last part) to find the general range of the guitar before finding the central frequency, so that the information in that range will not be lost. I store the Fourier vector in a matrix, and use it to plot the final spectrogram.

Section IV. Computational Results

For GNR, I plotted the fundamental frequency for each notes, and added some width to make it a little bit more visualizable. (figure1)

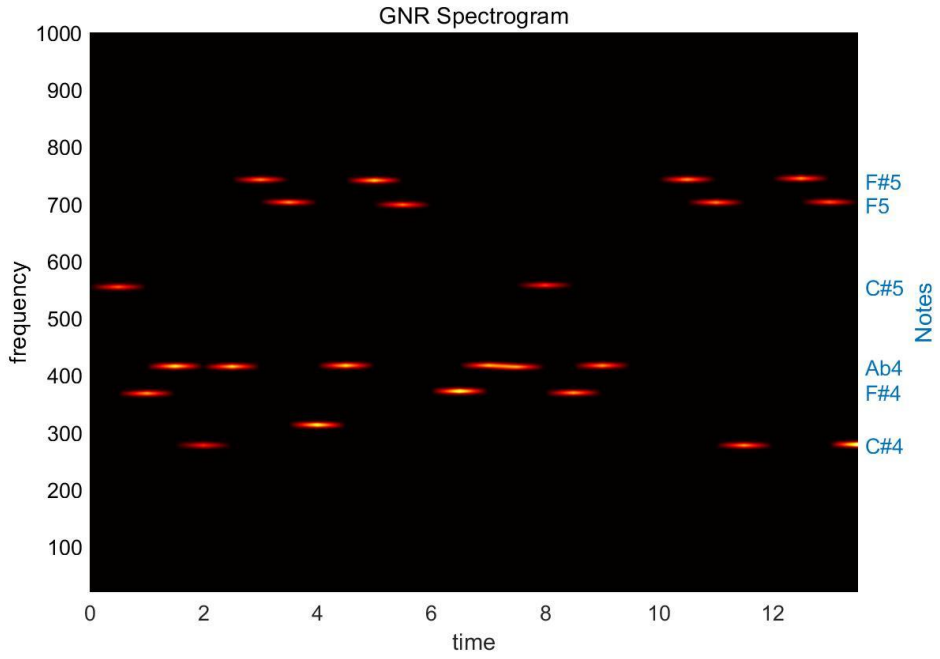

Figure 1:GNR spectrum, 14s, tau = .2

For bass in Floyd:

1. Bass Spectrum:

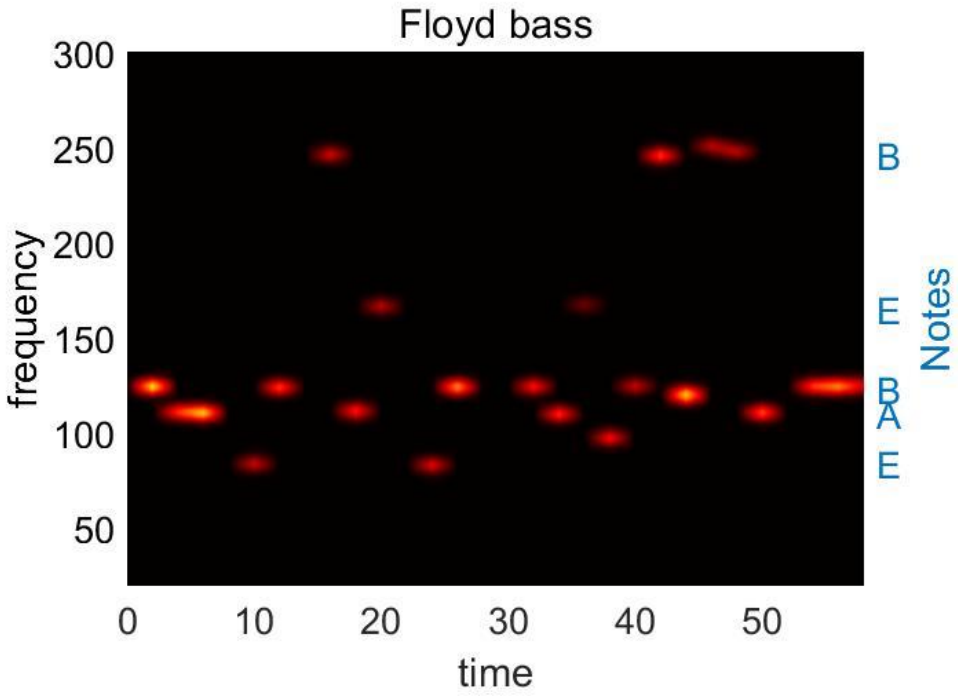

Figure 2: Floyd Bass spectrum, 60sec, tau = 2.

2. Guitar Spectrum:

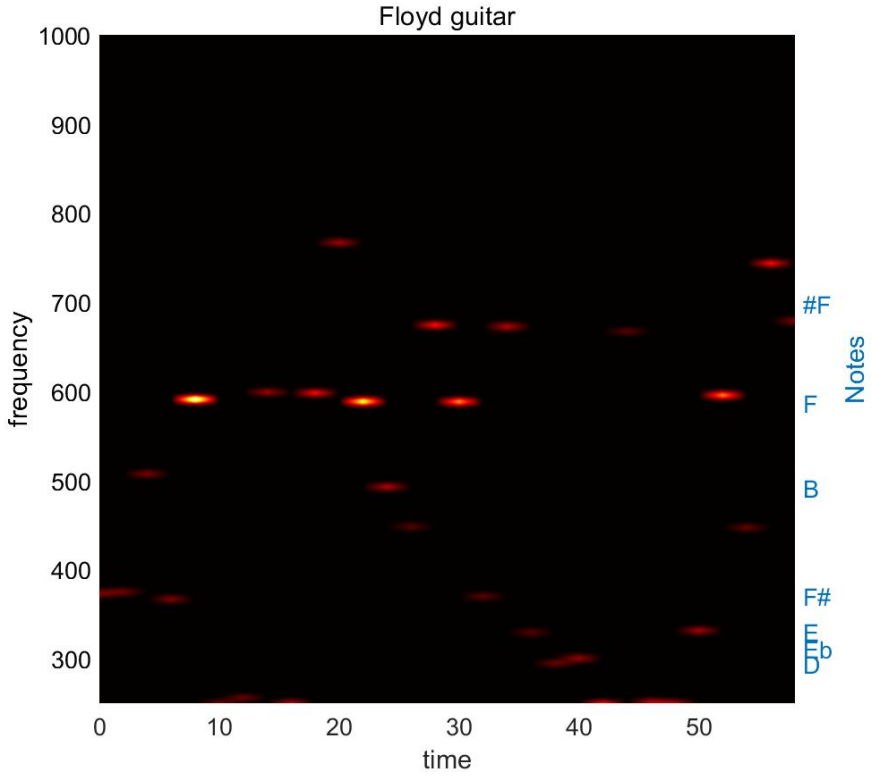

Figure 3: Guitar solo spectrum, 60 sec, tau = 2.

Section V. Summary and Conclusions

Gabor transform is capable of identifying patters of frequencies in localized period of time. It informs us that in certain period what frequencies are present, and therefore it is useful to recover some information in the time dimension.

Appendix A. MATLAB functions used and brief implementation explanation

1. audioread(): read in .m4a file with output S = sample data, Fs = number of samples per sec in Hertz.
2. length(): return the length of an element.
3. linspace(x1,x2,n): generate vector from x1 to x2 with step size n.
4. zeros(): generate a zero matrix with specified size.
5. transpose(): transpose a matrix to implement matrix multiplication.
6. fft() and fftshift(): implement Fourier transform and shift the transformed elements into right order.
7. lowpass(): "filters the input signal x using a lowpass filter with normalized passband frequency wpass in units of π rad/sample", used for Bass frequency filtering.
8. [max_Sgt,ind] = max(Sgt); return the maximum element in an array and its index, used to find central frequency.

Graphing:

1. pcolor(): creates a pseudocolor plot using the values in matrix C.
2. yyaxis left/ yyaxis left: make separate y axis, one indicate frequency.
3. yticklabels(), yticks(), set frequency-note match for each spectrum.

Appendix B. MATLAB codes

```
clear all;
close all;
clc

%%
figure(1)
[S, Fs] = audioread('Floyd.m4a'); % S = sample data, Fs =
number of samples per sec in Hertz
tr_gnr = length(S)/Fs; % record time in seconds
S = S(1: Fs * 58);
plot((1:length(S))/Fs,S);
% title('Sweet Child O Mine');
xlabel('Time [sec]'); ylabel('Amplitude');
% p8 = audioplayer(S,Fs); playblocking(p8);
```

```
L = 58; % spatial domain
n = length(S); % Fourier modes
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (2*pi/L)*[0:n/2-1 -n/2:-1]; ks = fftshift(k);

%%
test = transpose(abs(ks/ 2/ pi - 250));
[kmin, kind] = min(test);
instrument_filter = ones(n, 1);
instrument_filter(1:kind, 1) = 0;

%% Floyd
a = 1000;
tau = 0:2:58;
Sgt_spec = zeros(n, length(tau));
fund = zeros(length(tau),1);
for j = 1:length(tau)
   gabor = exp(-a*(t - tau(j)).^2); % Gabor window function
   Sg = transpose(gabor).*S;
   Sgt = fftshift(abs(fft(Sg))) .* instrument_filter;
   [max_Sgt,ind] = max(Sgt);
   central = abs(ks(ind));
   fund(j,1) = central;
   gauss_filter = exp(-0.001*(ks - central).^2);
%    gauss_filter = 1;
   Sgt_spec(:,j) = transpose(gauss_filter) .* Sgt;
end
%%
figure(2)
pcolor(tau,ks/(2 * pi),Sgt_spec/(2 * pi))
shading interp
yyaxis left
set(gca,'ylim',[250,1000],'Fontsize',16)
ylabel('frequency')
xlabel('time')
colormap(hot)

yyaxis right
ylabel('Notes')
set(gca,'ylim',[250,1000],'Fontsize',16)
yticklabels({'D','Eb','E','F#','B','F','#F'})
yticks([294,311.13,330,370,492,587,698,740])
```

```
% yticklabels({'E','A','B','E', 'B'})
% yticks([83.6842,110,123,164.81,246])

title('Floyd guitar')
%% GNR
a = 1000;
tau = 0:0.5:tr_gnr;
Sgt_spec = zeros(n, length(tau));
fund = zeros(length(tau),1);
for j = 1:length(tau)
   gabor = exp(-a*(t - tau(j)).^2); % Gabor window function
   Sg = transpose(gabor).*S;
   Sgt = fftshift(abs(fft(Sg)));
   [max_Sgt,ind] = max(Sgt);
   central = abs(ks(ind));
   fund(j,1) = central;
   gauss_filter = exp(-0.001*(ks - central).^2);
%    gauss_filter = 1;
   Sgt_spec(:,j) = transpose(gauss_filter) .* Sgt;
end

figure(2)
% pcolor(tau,ks,Sgt_spec)
pcolor(tau,ks/(2 * pi),Sgt_spec/(2 * pi))
shading interp
yyaxis left
set(gca,'ylim',[20,1000],'Fontsize',16)
ylabel('frequency')
xlabel('time')
colormap(hot)

yyaxis right
ylabel('Notes')
set(gca,'ylim',[20,1000],'Fontsize',16)
yticklabels({'C#4', 'F#4', 'Ab4', 'C#5','F5','F#5'})
yticks([277,370,415,554,698,740])
title('GNR Spectrogram')

%%
figure(3)
plot(log(abs(Sgt_spec)/max(abs(Sgt_spec)) + 1))
```