# HW2_A Gabor Transforms Problem

Yanjie Liu

## Abstract:

The goal of this report is to analyze audio file by Gabor transform. Specifically, the Fourier transform is used to transform from time domain to frequency domain and the Gabor filter is used to visualize the frequency of the audio file by producing the spectrograms. Then this report explore different Gabor windows like Gaussian wavelet, Mexican hat wavelet, and Shannon wavelet to do the sound frequency analysis.

## Introduction and Overview

Gabor transform is developed from Fourier transform and it is used to analyze signal in both frequency and time domain. In this report, the useful methods of Gabor transforms include changing the wavelet window, choosing different width of windows, and rate of window translation which could lead to 'oversampling' and 'undersampling'.

In part I, a portion of Handel's Messiah is given. This piece of sound signal will be transformed into time-frequency domain. Then by choosing different parameters of the Gabor filter, we will examine how they affect the spectrograms.

In part II, two audio files are given. The content of both files is Mary had a little lamb and instruments of each are piano and recorder. We will use Gaussian wavelet to reproduce the music scores by their spectrograms.

## Theoretical Background

The Fourier transform converts a function of time (a signal) into its constituent frequencies which is consisted by a set of cosine and sine function. For a given function f(x), the Fourier transform of f(x) is

$$\mathrm{F}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx \qquad (1)$$

In this function, the function of frequency, F(x), is generated by the integral of the product of the Euler's formula and f(x) on an infinite domain. And its inverse function is

$$\mathrm{f}(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k) dk \tag{2}$$

In this report, we use Fast Fourier Transform, since fft has complexity O(NlogN) which is much faster than the previous algorithm with complexity O(N^2) and the accuracy properties of fft is better than the standard discretization schemes.

## Gabor Transform

The Gabor transform is a special case of the short-time Fourier transform. The function to be transformed is first multiplied by a Gaussian function, which can be regarded as a window function, and the resulting function is then transformed with a Fourier transform to derive the time-frequency analysis. The Gabor transform of a signal x(t) is defined by this formula:

$$\mathrm{G}(\tau, \omega) = \int_{-\infty}^{\infty} x(t) e^{-\tau(t-\tau)^2} e^{-j\omega t} dt \tag{3}$$

In this report, we will use different types of wavelets as time windows which is centered on $\tau$ and with a width a. These two parameters are used to control the translation and the scale of the window.

## Gabor Wavelets

Gabor Wavelets are sued as a basis for Gabor transforms in information theory applications. The Gabor wavelets will minimize the product of the standard deviations in the time and frequency domain, which means the uncertainty during the transformation will be minimized.

### Gaussian wavelet

The method of applying Gabor transformation is to multiply the time filter Gabor function $g(t)$ with the original signal. The function of Gaussian filter is

$$\mathrm{g}(t - \tau) = e^{-a(t-\tau)^2} \tag{4}$$

In this function, $\mathrm{a}$ measures the bandwidth of the filter, t is the time, and $\tau$ is translation parameter, the center-frequency of the desire signal field. According to this equation, the larger $a$ is, the narrower the Gaussian window width will be.

### Mexican hat wavelet

The Mexican hat wavelet is one of the more common wavelets which is essentially a second moment of a Gaussian in the frequency domain. And the Mexican hat wavelet has excellent localization properties in both time and frequency due to the minimal time-bandwidth product of the Gaussian function. The function of Mexican hat wavelet and its transform are

$$\psi(t) = (1-t^2)e^{\frac{-t^2}{2}} \quad (4)$$

$$\hat{\psi}(\omega) = \sqrt{2\pi}\omega^2 e^{-\omega^2/2} \quad (5)$$

In this report, we will use complex Mexican hat wavelet to analyze the audio file for high temporal precision time-frequency analysis. The function of complex Mexican hat wavelet is

$$\hat{\psi}(t-\tau) = \frac{2}{\sqrt{3a}\pi^{1/4}}(1-\frac{(t-\tau)^2}{a^2})e^{-\frac{1}{2a^2}(t-\tau)^2} \quad (6)$$

**Shannon wavelet**

Shannon wavelet is a step function with value of unity within the transmitted band and zero outside of it. The function of Shannon filter is

$$\psi^{(sha)}(t-\tau) = |t-\tau| < a \quad (7)$$

# Algorithm Implementation and Development

## Part I

## Set-up

Before the start of analysis, we need to load the sound signal into Matlab. Since there is an odd number of data in this audio file, I delete the last column to make the size is even. Since the data is periodic, the deletion will not affect the analysis. After we obtain the length of the audio file, $L$ , we transform the time domain interval into the frequency domain interval by scaling by $\frac{2\pi}{\mathrm{L}}$ , which allows the later calculation of Fourier transform. Then we use fftshift to swap the frequency domain, which serves to eliminate the effect of fft on axis.

## Explore Gabor transform and draw spectrograms

In order to explore how the window width of the Gabor transform affect the spectrogram, I build up a for loop to make the Gabor filter go over the sound signal. And result of the for loop will be used to construct the spectrogram. The for loop allows us to explore how the window width affect the product of the filter function and signal in frequency domain. By changing the width of filter window, we can explore the idea of oversampling and undersampling. Then we will change the filter function in the for loop to explore how the Mexican hat wavelet and a step-function window affect the spectrogram.

## Part II

### Set-up

Before reproducing the music score, we need to load the audio file in Matlab. Then we obtain the length of the audio file, $L$ , we transform the time domain interval into the frequency domain interval by scaling by $\frac{2\pi}{\mathrm{L}}$ , which allows the later calculation of Fourier transform. Then we use fftshift to swap the frequency domain, which serves to eliminate the effect of fft on axis.

### Apply Gaussian filter and draw spectrogram

The process of this part is very similar with the part I. Here we use Gaussian filter to denoise the overtones. The number of notes with minimum duration is 32 in both pieces. Then we divide the length of the audio file by 32 to get the length of each interval which is around 0.5 and 0.4 second. Then we choose 0.2 as the interval. After testing different value of the window width, we choose 100 to construct the spectrogram.

## Computational Result

### Part I

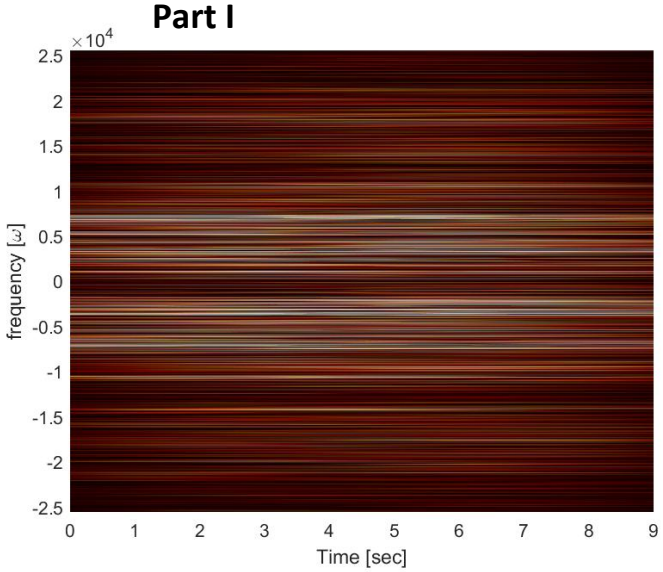


Figure 1: spectrogram under Gaussian filter with a = 0.1

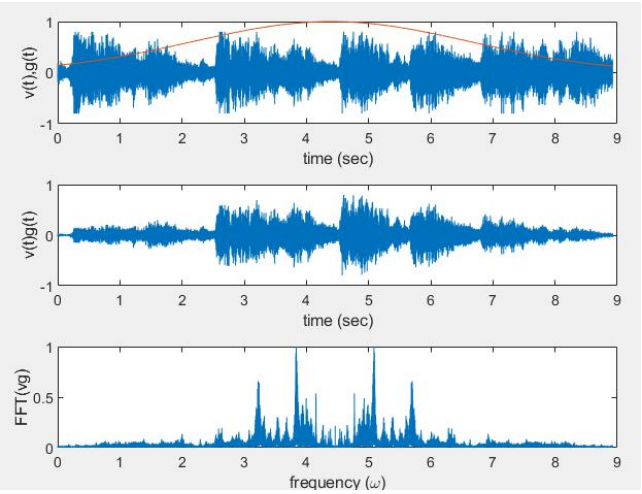


Figure 2: Gaussian filter and signal data and the Fourier transform of their product

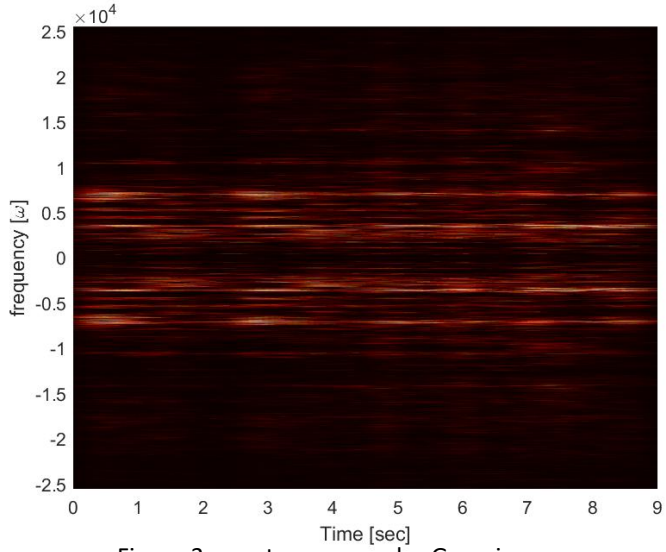


Figure 3: spectrogram under Gaussian filter with a = 10

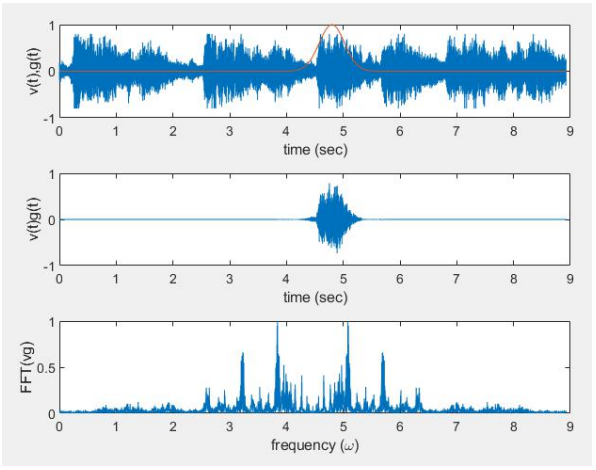


Figure 4: Gaussian filter and signal data and the Fourier transform of their product

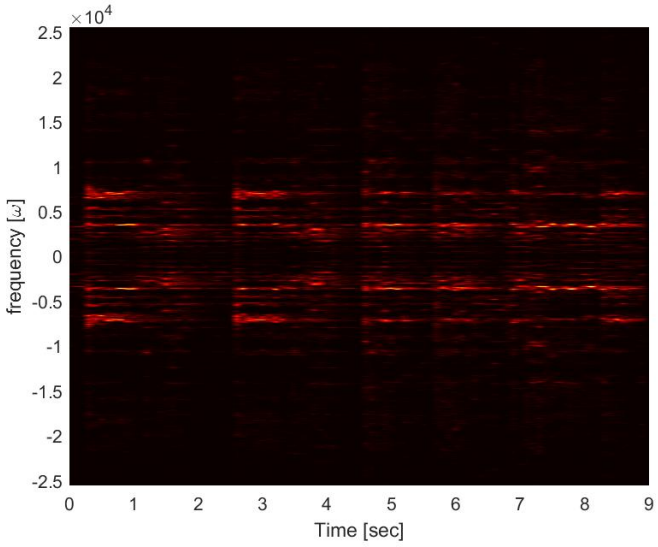


Figure 5: spectrogram under Gaussian filter with a = 1000

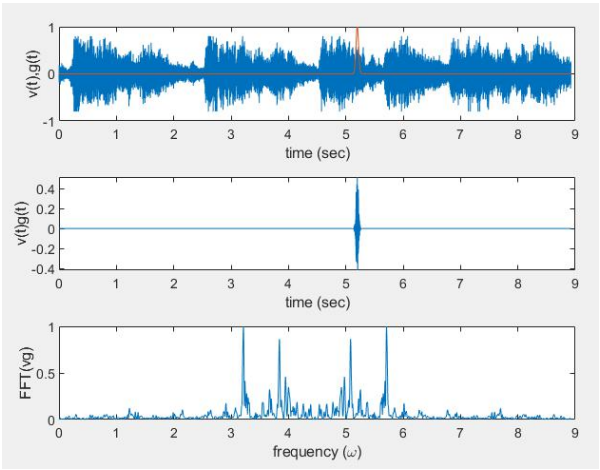


Figure 6: Gaussian filter and signal data and the Fourier transform of their product

For figure 1 and 2, we explore the large window width and oversampling. For figure 5 and 6, we explore the small window width and undersampling. For figure 3 and 4, they provide the comparison example to other figures. From these figures, we can conclude that the narrower Gabor window will make the spectrogram have more resolution. And the wider Gabor window will make the spectrogram have less resolution. And the oversampling will make the spectrogram obscure. The undersampling will make the spectrogram miss some information from the signal data.

## Part II

In this part, we use Gaussian wavelet with window width equivalent to 100 to reproduce spectrogram. In the below figures, we are able to discover that certain music score appears regularly at different frequency. Then we can discover the music scores from the spectrogram. Also the overtones have much dimmer light in the spectrogram.

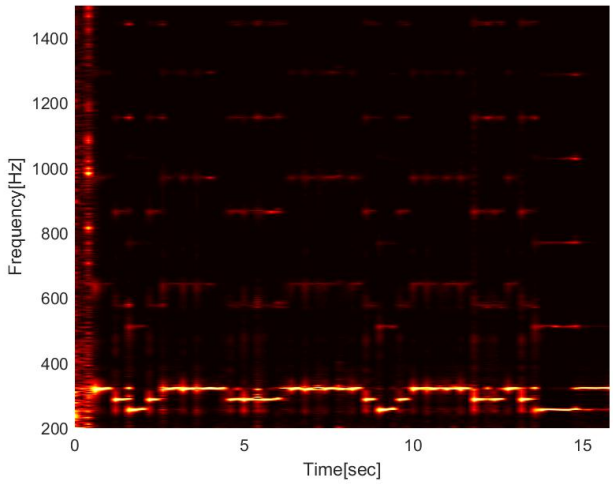


Figure 7: Piano spectrogram to show overtones

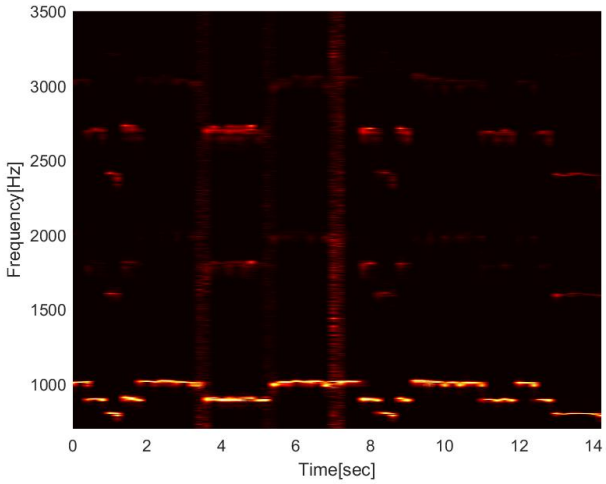


Figure 8: Recorder spectrogram to show overtones

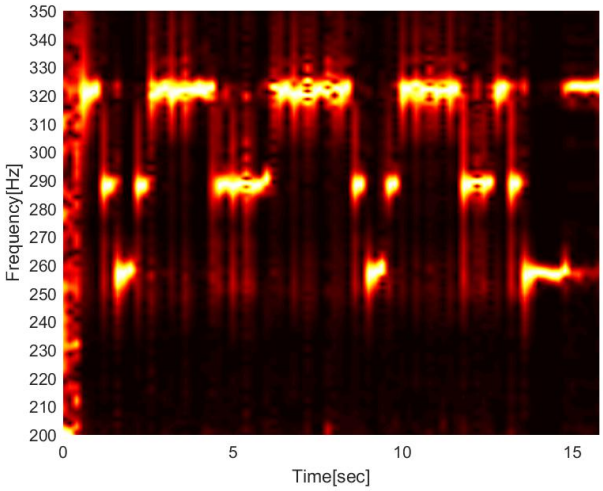


Figure 9: precise piano spectrogram

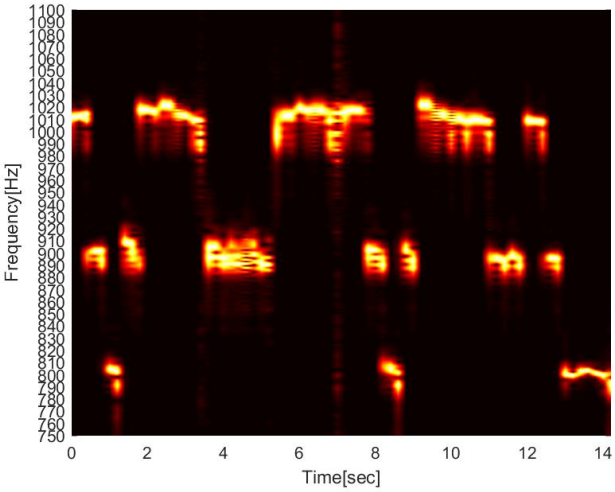


Figure 10: precise recorder spectrogram

According to figure 9 and 10, the frequency of piano piece is:

320Hz 290Hz 260Hz 290Hz 320Hz 320Hz 320Hz 290Hz 290Hz 290Hz 320Hz 320Hz 320Hz 320Hz 290Hz 260Hz 290Hz 320Hz 320Hz 320Hz 320Hz 290Hz 290Hz 320Hz 290Hz 260Hz

And the corresponding music score is:

E D C D E E E D D D E E E E D C D E E E E D D E D C

The frequency for recorder piece is

1020Hz 900Hz 810Hz 900Hz 1020Hz 1020Hz 1020Hz 900Hz 900Hz 900Hz 1020Hz 1020Hz 1020Hz 1020Hz 900Hz 810Hz 900Hz 1020Hz 1020Hz 1020Hz 1020Hz 900Hz 900Hz 1020Hz 900Hz 810Hz

And the corresponding music score is:

B A G A B B B A A A B B B B A G A B B B B A A B A G

## Summary and Conclusions

In this report , we explore the application of the Gabor transform. Specifically we figure out the relationship between Gabor window width and the resolution of the spectrogram. And the effect of oversampling and undersampling toward the spectrogram. The Gabor transforms is widely applicable in the time-frequency analysis and it is a good tool to do the sound frequency analysis.

# Appendix A

[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.

[M,I] = max(___) also returns the index into the operating dimension that corresponds to the maximum value of A for any of the previous syntaxes.

pcolor(X,Y,C) specifies the x- and y-coordinates for the vertices. The size of C must match the size of the x-y coordinate grid. For example, if X and Y define an m-by-n grid, then C must be an m-by-n matrix.

colormap(map) sets the colormap for the current figure to the colormap specified by map.

description from https://www.mathworks.com/help/matlab/

# Appendix B

## Part I

```
clear all; close all; clc;
load handel

v = y';
L=9; n=length(v);
v(end) = [];
t = (1:length(v))/Fs;
k = (2*pi/L)*[0:n/2-1 -n/2:-1]; ks = fftshift(k);

subplot(2,1,1)
plot(t,v)
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');

vt=fft(v);
subplot(2,1,2)
plot(ks,fftshift(abs(vt))/max(abs(vt)))
xlabel('Frequency [\omega]');
ylabel('Amplitude');
title('Signal in Frequency Domain');
saveas(gcf,'signal.png')

% Gabor, Mexican hat wavelet, and Shannon wavelet
a= 1000;
tslide = 0:0.5:9;
vgt_spec=[];
for ii = 1:length(tslide)
   % Gabor filter
   g = exp(-a*(t-tslide(ii)).^2);
   % Mexican hat wavelet
   %g =
(2/(sqrt(3*a)*pi^(1/4)))*(1-((t-tslide(ii)).^2/a)).*exp(-(t-tslide(ii)).^2/(2*a^2));
   % Shannon
   %g = abs(t - tslide(ii)) < a;
   vg = g.*v;
   vgt = fft(vg);
   vgt_spec = [vgt_spec; abs(fftshift(vgt))];
   subplot(3,1,1),plot(t,v,t,g)
   xlabel('time (sec)'), ylabel('v(t),g(t)')
   subplot(3,1,2),plot(t,vg)
   xlabel('time (sec)'), ylabel('v(t)g(t)')
   subplot(3,1,3),plot(t,fftshift(abs(vgt))/max(abs(vgt)))
   xlabel('frequency (\omega)'), ylabel('FFT(vg)')
   axis([0 9 0 1])
   drawnow
   pause(0.1)
end
```

```
figure(2)
pcolor(tslide,ks,vgt_spec.'), shading interp
xlabel('Time [sec]');
ylabel('frequency [\omega]');
colormap(hot)
drawnow, hold on
saveas(gcf,'spectrograms.png')
```

## Part II

```
clear all; close all; clc;

%[y,Fs] = audioread('music1.wav');
[y,Fs] = audioread('music2.wav');
L = length(y)/Fs; % record time in seconds
%p8 = audioplayer(y,Fs);
%playblocking(p8);
v = y.';
n = length(v);
t = (1:length(v))/Fs;
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);

%plot(ks,fftshift(abs(fft(v))));

tslide = 0:0.2:L;
spec = [];
Hertz = [];
a = 60;
for ii = 1:length(tslide)
    g = exp(-a*(t-tslide(ii)).^2);
    vg = g.*v;
    vgt = fft(vg);
    spec = [spec; fftshift(abs(vgt))/max(abs(vgt))];
    [M,I] = max(abs(vgt));
    Hertz = [Hertz; abs(k(I))];
end

figure(2)
pcolor(tslide,ks/(2*pi),spec.'), shading interp
xlabel('Time[sec]');
ylabel('Frequency[Hz]');
%set(gca,'Ylim',[200 350]);
%yticks(linspace(200,350,16));
set(gca,'Ylim',[750 1100]);
yticks(linspace(750,1100,36));
colormap(hot);
%saveas(gcf,'piano.png');
saveas(gcf,'record.png');
```