AMATH 582 HW2: Gabor Transforms

Huasong Zhang
Department of Applied Mathematics
University of Washington
Seattle, WA 98195
huasongz@uw.edu

Abstract

This project has two parts. For the first part, we need to apply different wavelets on a piece of music record and observe the differences caused by various type of wavelets and different window sizes of the wavelets. While for the second part, we applied the same wavelet to two different sound, one is from piano and the other one is from recorder. After applied wavelets, we can see clearly how sounds are different in many aspects. From the first part of the assignment, we can observe the impacts of wavelets; and from the second part, we will notice the difference in sound.

1 Introduction and Overview

In our life, there are so many sounds and they all have different frequencies. That is why we have this wonderful world which is composed of various sounds. In order to gain more information of the sounds, it is hard to do things on something that we cannot see. That is why we need Hz, spectrogram and other things to represent the sounds visually and numerically. And also, the sounds always are mixed with noises; that's why we need to use the filter to extract the most essential components from the sound. Therefore, in this assignment, we are going to analyze three music signals by applying some techniques.

2 Theoretical Background

First of all, a sound signal is a wave in time domain. In order to observe it, we will need Fourier Transform to transform it to frequency domain. Fourier Transform is an eigenfunction expansion over the wave numbers k by using the formula:
$F(k) = \frac{1}{\sqrt{(2\pi)}} \int_{-\infty}^{\infty} f(x) e^{-ikx} dx$
and its inverse is
$f(x) = \frac{1}{\sqrt{(2\pi)}} \int_{-\infty}^{\infty} F(k) e^{ikx} dk$
The kernel e^{-ikx} represents the oscillation and k is the wave number. However, when we use Fourier Transform it just tells us the frequencies present in the song signal, it doesn't tell us when it was played. However, for a song, the most important thing is when the node is played, which we are missing. Also, for a non-stationary song signal, the time and frequencies change with time. It is impossible to just use one Fourier Transform to represent the whole song signal. Therefore, the Fourier Transform is not appropriate in this case and the Gabor transform was introduced. Gabor transform works by applying filter on the signal and the filter moves with respect to time, which can filter each segment of the signal. Therefore, we can not only localize in frequency but also localize in time. We will get a complete picture of the frequency of the signal and when this frequency happens. The formula for Gabor transform is:
$g_{t,\omega}(\tau) = e^{-i\omega\tau} g(t-\tau)$
$G[f](t,\omega) = f_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau) \bar{g}(\tau - t) e^{-i\omega\tau} d\tau$
g is the new Fourier kernel which replace the original kernel e^{-ikx} in Fourier Transform. This kernel has a new term $g(t-\tau)$ which represents it will move with respects to time and applies on every piece of the signal.

3 Algorithm Implementation and Development

Part I

First of all we load our data into the working space. v is the signal. Since this music is 9 seconds, so we define $L = 9$. And we want n data points equally distributed on this time domain and this n is from the number of data we have in the dataset. Since we assume it is periodic, so we divided the entire time domain into $n + 1$ points and take the first n. After that, we define k which has range of 2π. After everything is set up, we now need to apply Gabor transform.

Since the filter goes with time, we need a for loop and out our wavelet inside the for loop. At first, we define the tslide, which decides the step size we want the filter go each time. If the step size is too large, we will have undersampling, since the filter goes too fast; if we have step size too small, it is the case of oversampling, and we may take too many repeated information of the signal and it will take longer time and take more memory. We also need to build an empty set spc in order to save the signal information and plot spectrogram later.

Inside the for loop, we first define our wavelet and denoted as g. g contains the $(t - tslide(j))$ term, which makes it moves with respect to time. Then we apply the filter to the signal by multiplying them together and use FFT to do Fourier transform of the filtered signal at this specific time. And we save the abs(fftshift(vft)) in the spc as our spectrogram data. We also want to see how the filter moves and how it affects and extracts signal information. Therefore, we want to plot a 3 by 1 plot inside the for loop. On the top, we plot the original signal and filter. The second one, we plot the filtered signal in time domain. And for the third one, we plot the filtered signal in Fourier domain. Outside the for loop, we can use the information we saved in spc to plot the spectrogram using pcolor.

For changing step size and filters, I put three different step sizes and 12 filters with four kinds and three window widths for each kind of filter. I will comment out the others and just use one step size and one filter each time.

Part II

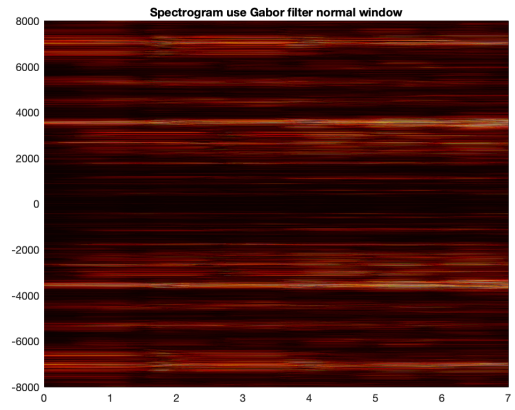
For piano, since the music is 16 seconds, so I set the $L = 16$. And the way to set up t and k is the same way as I did in the first part. Since we need to find out the Hz of this piece of music. The only thing which is different is I built an empty set called Max in order to save the information of maximum frequency each time. After the signal is filtered and transformed in to Fourier domain, I find out where the maximum frequency is. Then, use the index to locate in k and save it. We want to find the music score, which means we want to find the Hz range for this piece of music. Hz represents how many time something happens in one second. Therefore, after we find the index of maximum frequency, we want to use it to locate in wavenumber which is defined as number of wavelength per unit. And this unit is 2π. Therefore, when we plot the nodes, we divide the Max by 2π to get the correct Hz with respect to time.

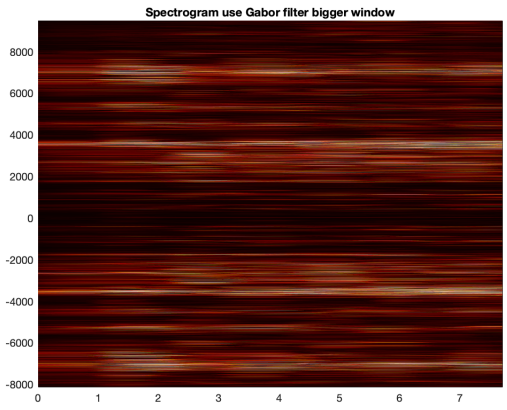
For the recorder piece of signal, the process is exactly the same. Then we can compare the two nodes plots and observe the differences from them.

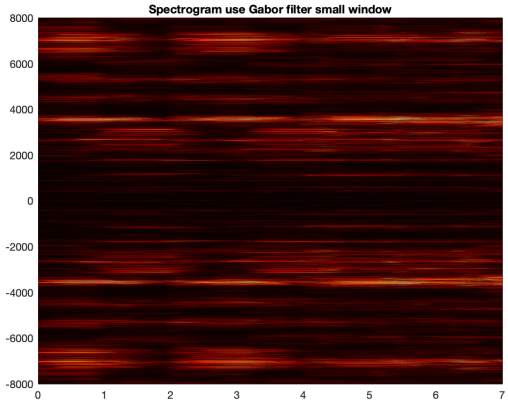
4 Computational Results

Part I

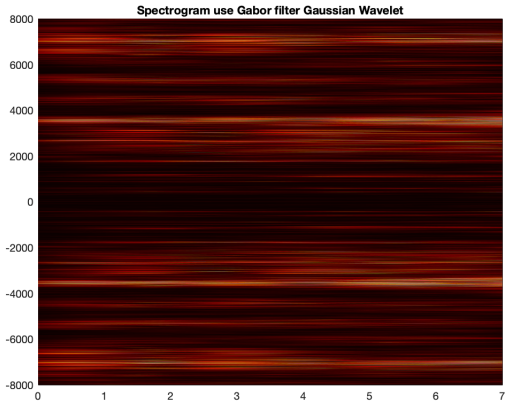
Wavelet used in class $\begin{cases} g = e^{(-(t-\tau)^{10})} & \text{normal window} \\ g = e^{(-0.01*(t-\tau)^{10})} & \text{big window} \\ g = e^{(-10*(t-\tau)^{10})} & \text{small window} \end{cases}$

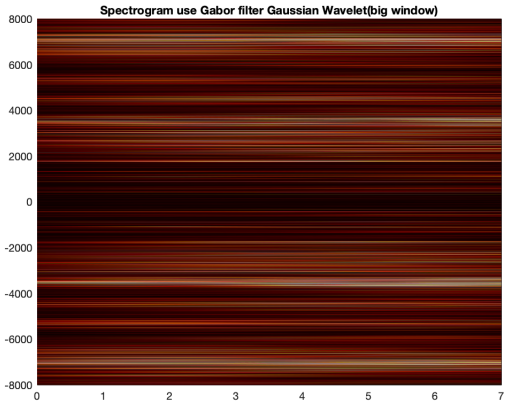


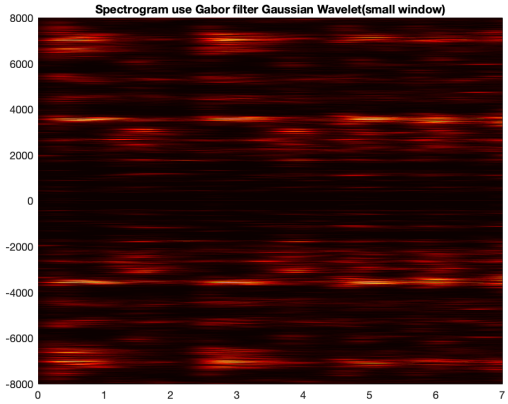


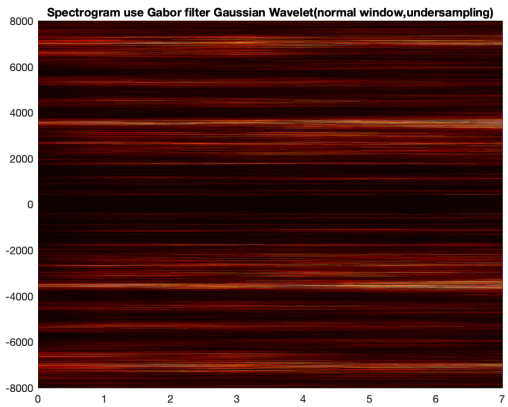


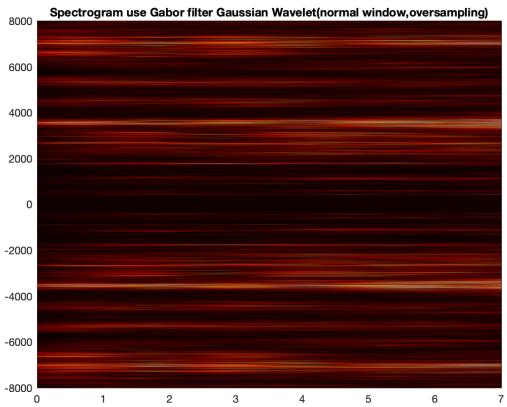
Gaussian Wavelet $\begin{cases} g = e^{(-(t-\tau)^2)} & \text{normal window} \\ g = e^{(-0.01*(t-\tau)^2)} & \text{big window} \\ g = e^{(-10*(t-\tau)^2)} & \text{small window} \end{cases}$



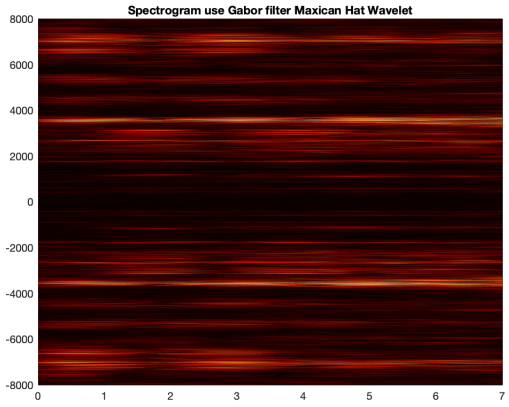


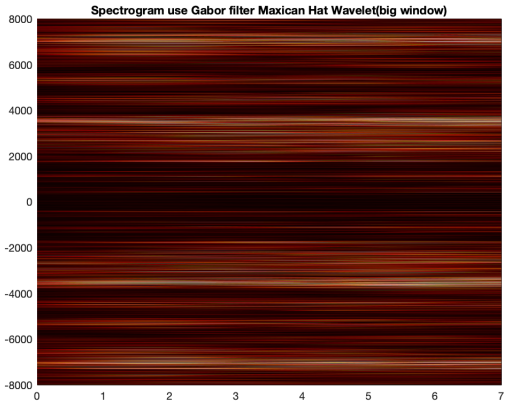


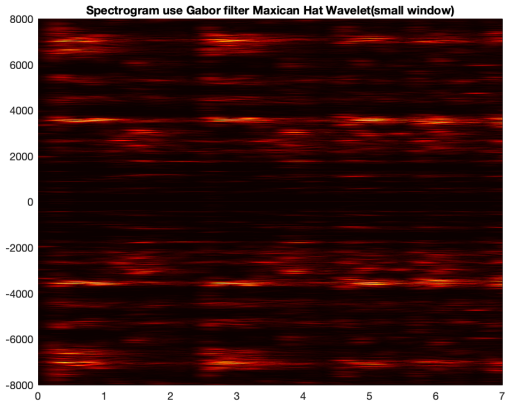




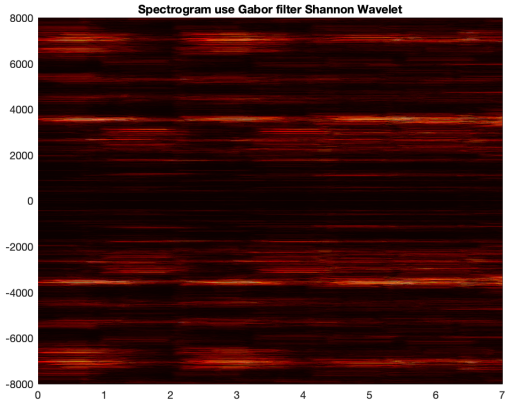
Maxican Hat Wavelet $\begin{cases} g = (1-(t-\tau)^2).*e^{(-(t-\tau)^2)} & \text{normal window} \\ g = (1-0.1*(t-\tau)^2).*e^{(-0.1*(t-\tau)^2)} & \text{big window} \\ g = (1-10*(t-\tau)^2).*e^{(-10*(t-\tau)^2)} & \text{small window} \end{cases}$

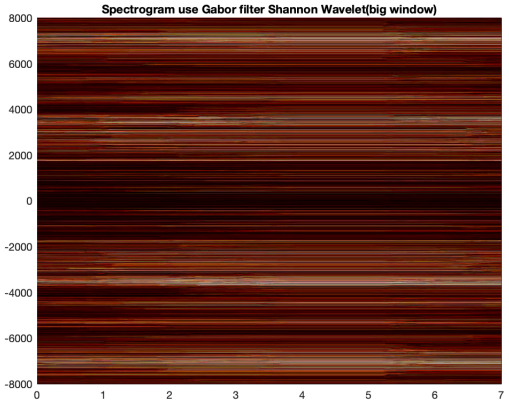


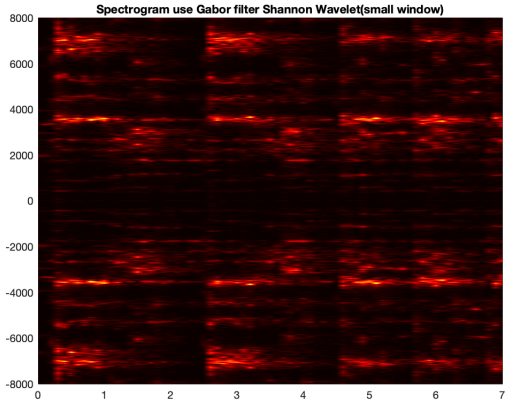


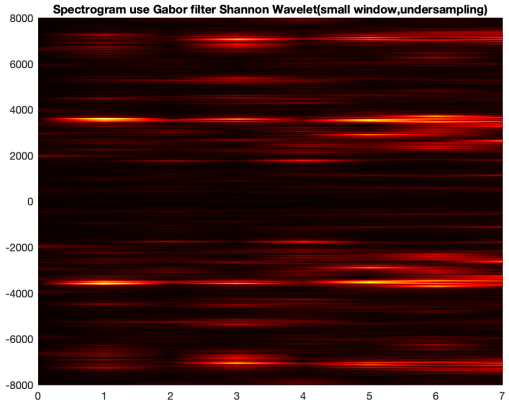


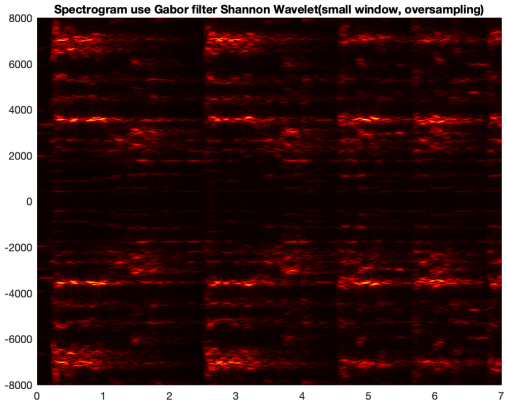
Shannon Wavelet $\begin{cases} g = |t-\tau| <= 0.5 & \text{normal window} \\ g = 0.1*|t-\tau| <= 0.5 & \text{big window} \\ g = 10*|t-\tau| <= 0.5 & \text{small window} \end{cases}$





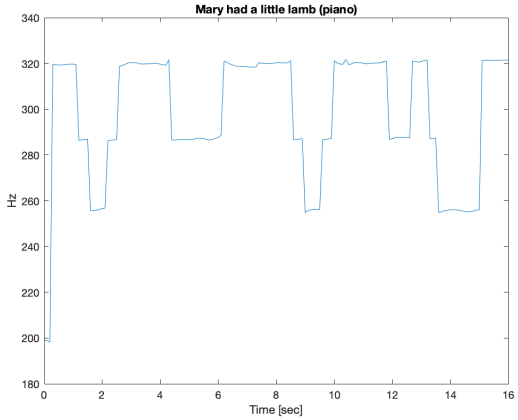


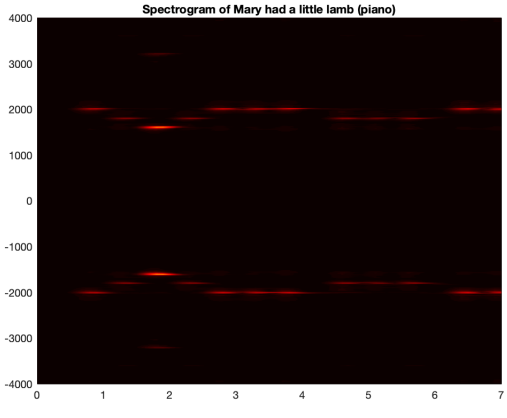




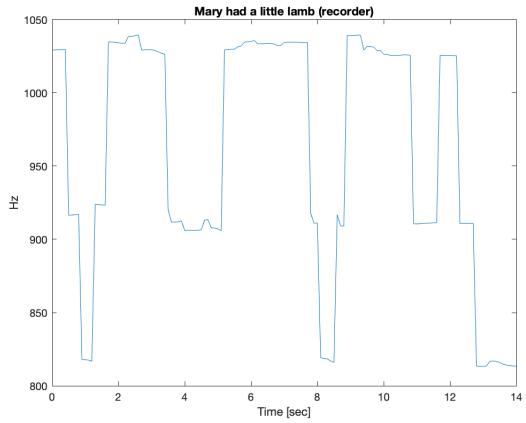
Part II

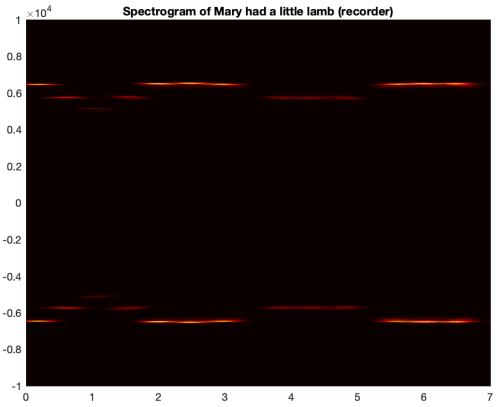
Piano





Recorder





5 Summary and Conclusions

Part I

For each group of spectrograms we can see that for wavelets with smaller window size, the spectrogram is more clear; for large window size, the spectrogram is vague. This indicates that when the window size is large, too many frequencies will be taken and extracted; there is no emphasizes. When we apply a wavelet with smaller window size, the most essential information from the signal can be taken out, and from the spectrogram, we can see clearly where the nodes are and their frequencies.

Speaking of undersampling and oversampling, we can see that from the Gaussian wavelet plots that for the same width of filter, undersampling will make the spectrogram even more vague a little bit. While, if we oversampling, it makes the spectrogram become more clear a little bit, but it's still not as clear as the results from a smaller window width. So the undersampling and oversampling on the same filter does not have significant effects on a filter that originallt does not perform very well.

Applying undersampling and oversampling on the small width of Shannon filter, it can be seen that the spectrogram from undersampling is more smooth, while the oversampling one is discrete and every piece of information from the signal is extracted and saved.

Overall, it seems that oversmapling is always better than undersampling. And if we cannot decide the sample size, it is better to oversampling. Also, small width of the filter is better than large width.

Part II

From the plots we can see that for piano, the music scale for piano is between 280 and 320, and 800 to 1050 for recorder. It seems that the Hz for recorder is more spread out comparing to piano since it ranges from 800 to 1050 which have 250 differences; while the Hz for piano only has 40 differences. Also, recorder has higher Hz. The lowest is for recorder is around 800, while the highest Hz for piano is about 320. That is the reason why the timbre of piano and recorder are different.

Therefore, from the time frequency analysis, we can extract the Hz information of a music signal. Based on the music score, we are able to identify where this piece of music signal comes from.

Appendix A MATLAB functions used and brief implementation explanation

fft: does Fourier Transform on signal
fftshift: since fft will switch the left side and right side of the signal and multiply the modes by -1, fftshift is puting the signal back in the right position and take the absolute of the amplitude.

Appendix B MATLAB codes

Part I

```
clear all;clc
load handel
v = y'/2;
figure(1)
```

```
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');
% This is in time domain

% p8 = audioplayer(v,Fs);
% playblocking(p8);

L = 9;
n = length(v);
t2 = (1:n+1)/Fs;
t = t2(1:n);
k = (2*pi/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

vt = fft(v(1:end-1));
figure(2)
subplot(2,1,1)
plot(t,v)
subplot(2,1,2)
plot(ks,fftshift(vt))

%% Gabor

tslide = 0:0.1:9;
%tslide = 0:1:9;
%tslide = 0:0.02:9;
spc = [];
for j = 1:length(tslide)
    %g = exp(-(t-tslide(j)).^10);
    %g = exp(-0.01*(t-tslide(j)).^10); %bigger window
    %g = exp(-100*(t-tslide(j)).^10); %smaller window: localizing in time
    %g = exp(-(t-tslide(j)).^2); %Gaussian Wavelet
    %g = exp(-0.1*(t-tslide(j)).^2); %Gaussian Big
    %g = exp(-10*(t-tslide(j)).^2); %Gaussian Small
    %g = (1-(t-tslide(j)).^2).*exp(-(t-tslide(j)).^2); %Mexican hat
    %g = (1-0.1*(t-tslide(j)).^2).*exp(-0.1*(t-tslide(j)).^2); %Mexican hat big
    %g = (1-10*(t-tslide(j)).^2).*exp(-10*(t-tslide(j)).^2); %Mexican hat small
    %g = abs(t-tslide(j)) <= 0.5; %Shannon function
    %g = 0.1*abs(t-tslide(j)) <= 0.5; %Shannon function big
    g = 10*abs(t-tslide(j)) <= 0.5; %Shannon function small

    vf = g.*v; % signal filter
    vft = fft(vf(1:end-1));
    spc = [spc;abs(fftshift(vft))]; %save fourier transform of the signal

    figure(3)
    subplot(3,1,1)
    plot(t,v,'k-',t,g,'r-','Linewidth',[2])
    subplot(3,1,2)
    plot(t,vf,'k-','Linewidth',[2])
    axis([0 9 -0.5 0.5])
    subplot(3,1,3)
    plot(fftshift(k),abs(fftshift(vft))/max(abs(fftshift(vft))),'m','Linewidth',[2])
    axis([-200 200 0 0.2])
    pause(0.2)
end

figure(4)
pcolor(tslide,fftshift(k),spc.'),shading interp, colormap(hot)
axis([0 7 -8000 8000])
title('Spectrogram use Gabor filter Shannon Wavelet(small window)')
```

Part II

```
%% Piano
clear all;clc
L = 16;
n = 701440;
t2 = linspace(0,L,n+1);
t = t2(1:n); % make n+1 points and throw out the last point
k = (2*pi/L)*[0:n/2-1 -n/2:-1];

tr_piano=16; % record time in seconds
y1=audioread('music1.wav');
Fs1=length(y1)/tr_piano;
y1 = y1.';
figure(1)
plot((1:length(y1))/Fs1,y1);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (piano)'); drawnow
%p8 = audioplayer(y1,Fs1); playblocking(p8);

figure(2)
tslide = 0:0.1:16;
Max = [];
spc = [];
for j = 1:length(tslide)
    g = exp(-20*(t-tslide(j)).^2);
    Sf = g.*y1; % signal filter
    Sft = fft(Sf);
    [f_max,ind] = max(Sft);
    Max = [Max;k(ind)];
    spc = [spc;abs(fftshift(Sft))];

    subplot(3,1,1)
    plot(t,y1,'k-',t,g,'r-','Linewidth',[2])
    subplot(3,1,2)
    plot(t,Sf,'k-','Linewidth',[2])
    axis([0 16 -1 1])
    subplot(3,1,3)
    plot(fftshift(k),abs(fftshift(Sft))/max(abs(fftshift(Sft))),'m','Linewidth',[2])
    axis([-100 100 0 0.1])
    pause(0.2)
end
figure(3)
plot(tslide,abs(Max./(2*pi)))
xlabel('Time [sec]'); ylabel('Hz');
title('Mary had a little lamb (piano)')

figure(4)
pcolor(tslide,fftshift(k),spc.'),shading interp, colormap(hot)
axis([0 7 -4000 4000])
title('Spectrogram of Mary had a little lamb (piano)')
%% Record
clear all;clc
L = 14;
n = 627712;
t2 = linspace(0,L,n+1);
t = t2(1:n); % make n+1 points and throw out the last point
k = (2*pi/L)*[0:n/2-1 -n/2:-1];

figure(5)
tr_rec=14; % record time in seconds
y2=audioread('music2.wav'); Fs2=length(y2)/tr_rec;
y2 = y2.';
plot((1:length(y2))/Fs2,y2);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)');
%p8 = audioplayer(y,Fs); playblocking(p8);
```

```
tslide = 0:0.1:14;
Max2 = [];
spc2 = [];
for j = 1:length(tslide)
    g = exp(-20*(t-tslide(j)).^2);
    Sf2 = g.*y2; % signal filter
    Sft2 = fft(Sf2);
    [f_max2,ind] = max(Sft2);
    Max2 = [Max2;k(ind)];
    spc2 = [spc2;abs(fftshift(Sft2))];

%     figure(6)
%     subplot(3,1,1)
%     plot(t,y2,'k-',t,g,'r-','Linewidth',[2])
%     subplot(3,1,2)
%     plot(t,Sf2,'k-','Linewidth',[2])
%     axis([0 14 -1 1])
%     subplot(3,1,3)
%     plot(fftshift(k),abs(fftshift(Sft2))/max(abs(fftshift(Sft2))),'m','Linewidth',[2])
%     axis([-100 100 0 0.1])
%     pause(0.2)
end
figure(7)
plot(tslide,abs(Max2./(2*pi)))
xlabel('Time [sec]'); ylabel('Hz');
title('Mary had a little lamb (recorder)')
figure(8)
pcolor(tslide,fftshift(k),spc2.'),shading interp, colormap(hot)
axis([0 7 -10000 10000])
title('Spectrogram of Mary had a little lamb (recorder)')
```