AMATH582 HW2 GABOR TRANSFORMS

SHUANG WU

Jan. 28th

Abstract

Fourier transform is a very important method when we want to analyze the signals and frequencies. But there is a huge limitation for this method that we cannot get any time information after we do the transformation. If we can divide the time domain into small windows and use FFT for each of the window to figure out the frequency, though lose some frequency information, we will have both information for frequency and time simultaneously. This method was proposed by Gabor Denes, whose method creates a kernel which will filter the signal out for a specific time domain.

1 INTRODUCTION

In this project, I will first analyze a portion of Handel's Messiah with time-frequency analysis. I will find the spectrogram of part of the Handel's Messiah, like 9 seconds. I will also explore many interesting things during this process, like how the width of that window affects the spectrogram and what is the idea of oversampling and undersampling. The most important thing is how different types window will affect the spectrogram. For the second part, I have a music track named *Mary Had a Little Lamb* that is played on both the recorder and piano. I want to find the music score for these two different pieces music and find the difference between the recorder and the piano.

2 THEORETICAL BACKGROUND

2.1 *Gabor transformation*[1]

In order to filter out the specific location of part of the signal, Gabor built a kernel:

$$g_{t,\omega}(\tau) = e^{i\omega\tau} g(\tau - t)$$

The term $g(\tau - t)$ was used to locate the specific time domain. The equation for Gabor's transform was also named STFT(short-time Fourier transform):

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

where the bar means complex conjugate.

I will first go through all the properties as a list and then give a plot that was introduced in AMATH 582 class notes from Professor Kutz to help understand how this work.

1. The energy is bounded by the Schwarz inequality so that

$$|\bar{f}_g(t,\omega)| \leq \|f\|\|g\|$$

[1] All the figures and equations in this section are come from the 582 notes from Kutz

2. The energy in the signal plane around the neighborhood of (t, ω) is calculated from:

$$|\bar{f}_g(t,\omega)|^2 = |\int_{-\infty}^{\infty} f(\tau)g(\tau - t)e^{-i\omega\tau}d\tau|^2$$

3. The time-frequency spread around a Gabor window is computed from the variance:

$$\sigma_t^2 = \int_{-\infty}^{\infty} \tau^2 |g(\tau)|^2 d\tau$$

$$\sigma_\omega^2 = \frac{1}{2\pi}\int_{-\infty}^{\infty} v^2 |\bar{g}(v)|^2 dv$$

where $\sigma_t\sigma_\omega$ is independent of t and ω and is governed by the Heinsenberg uncertainty principle.

4. Garbor transform is linear

5. Can be inverted with the formula:

$$f(\tau) = \frac{1}{2\pi}\frac{1}{\|g\|^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \bar{f}_g(t,\omega)g(\tau - t)e^{i\omega\tau}d\omega dt$$

Let's now look at the cartoon to understand how it works:

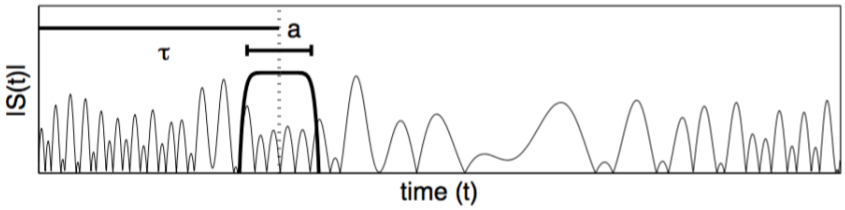

Figure 1: Time filtering window centered at τ with width a

The figure 1 shows how Gabor transform work. For a window with width a, we move the window to the position centered at τ. Thus, before we do the FFT we already have the time content, and if we do the FFT, we will have the specific frequency for that specific time domain. As we increase the τ from 0 to the end of the time, we will observe all the frequencies related to it's specific time domain like shift in the window.

Look at another cartoon, figure 2, which is also the most important one that demonstrates the basic idea of the Garbo transform:

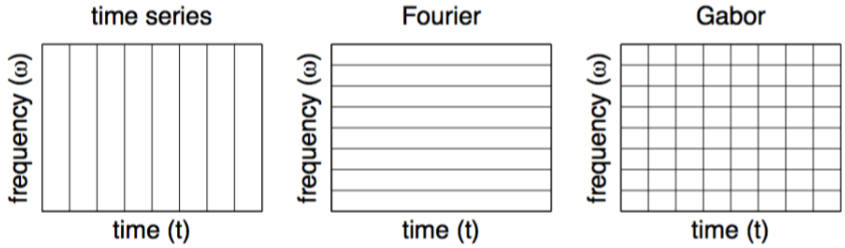

Figure 2: First plot is the time series method, the second plot is Fourier method, and the last one is Gabor method.

In the figure 2, we can observe a very good resolution in time domain, but we have no idea with the frequency domain from the first plot. Then we do the FFT, which transfer the time domain into frequency domain. Although we will get a very nice resolution plot in frequency domain, we have no idea

about the time domain. At last, Gabor transform creates a window to pull out a specific time information, and then transfer to frequency domain that combines the time and frequency information.

But every coin has two sides. In the figure 1, if we have a wavelength that is longer than the window with width a, we will totally lose that signal. Maybe we can try to make the window wider, but time domain will be narrower by no surprise. If we shrink the width of the window, we can have a better resolution of the time domain, but we will lose some information about the frequency that has longer wavelength than the width a. This is like a trade off, we need use our resolution of time to trade some resolution of frequency, and vice versa.

2.2 *Wavelets*

In order to improve the Gabor transform, wavelet theory was introduced. In this theory, we first use a wider window to filter out the low frequency, and then we make the window narrower to filter out the high frequency and better time resolution. We first need a function named mother wavelet:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$$

for a$\neq$ 0 and b are real constants. In this case, a is the same a as before to stand the width of the window and b is the τ as before to indicate the translation or transform rate. Different mother wavelets can be decided on the different types of problems.

I'd like to talk about 3 different wavelets or the type of window that I use in the later section.

2.3 *Gaussian wavelet*

The most normal wavelet named Gaussian wavelet with the equation:

$$\psi(\tau - t) = e^{-a(t-\tau)^2}$$

for a is the width and τ is the location of the center.

2.4 *The Mexican Hat Wavelet*

This is one of the more common wavelets defined by the second moment of a Gaussian in the frequency domain. The general equation is:

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

for a is the width and τ is the location of the center.

2.5 *Super-Gaussian wavelet*

This is almost the same as Gaussian wavelet but changes the power from 2 to 10, named super Gaussian.

$$\psi(\tau - t) = e^{-a(t-\tau)^{10}}$$

for a is the width and τ is the location of the center.

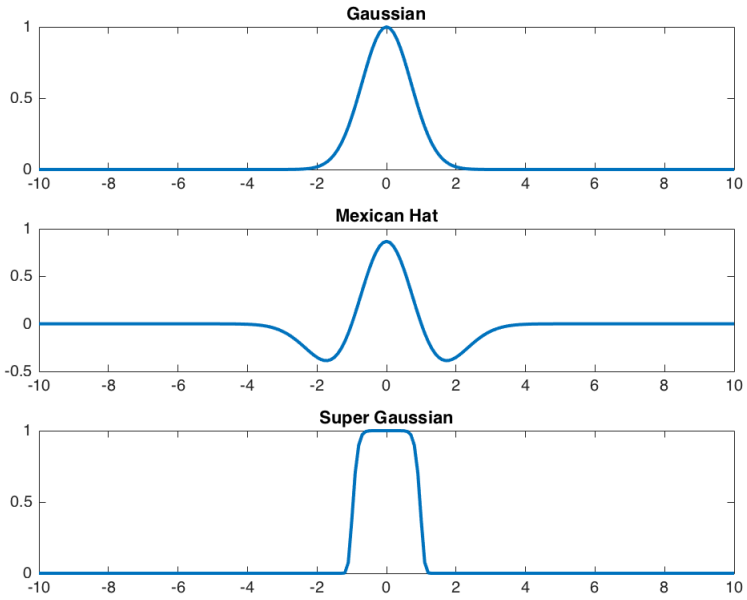

Figure 3: Different window type. The first one is the Gaussian wavelets. The second one is Mexican Hat wavelets, and the last one is called Super Gaussian wavelets.

3 ALGORITHM IMPLEMENTATION

3.1 *Part I*

1. Build up the basic term

   Before starting doing the transform, I need to load the data and know the size of this data. I already known this data is 9 seconds long and has a total of 73113 sample points. So we set the length $L = 9$. I also need to set my sample points n to be $73113 - 1$, which equals to 73112. The reason I minus one from the sample points is: the number is odd which will not give a integer after we divided by 2 when defines the Fourier domain, k. Minus one will be fine, because the data size is very large.

   After set the default numbers, I can set my t use the linsapce and k, in frequency domain by transfering the domain into 2π domain. I also do the `fftshift` for my k to make sure the plot will have the right position. Another parameter that is very important to set named tslide, which is associated with τ, defined by $0 : \tau : L$.

   I choose the translation rate be 0.1, which will have tslide from 0 to 9 by 0.1. This tslide will give me 91 points which means the window will shift from left to right 91 times. Also, I set a empty vector that can store all the spectral data later.

2. Build for loop to do the transform and save all the spectral data points.

   In the `for` loop, the most important thing is building the kernel or the type of the window.

   I introduced three types of windows in the previous section including Gaussian, super Gaussian and Mexican hat. I directly write the equa-

tion I have before to calculate the new vector g.

After getting the kernal g, I use the g times the original data and then do the FFT. This will FFT the signal in that small window for a specific time domain. Last, I want to save the absolute value of the `fftshift` of the after-transfer value to the spectral vector. At the end of the `for` loop, I will have all the spectral data points ready and I can continue plotting it.

3. Plot the spectrogram

   Use the code from the lecture note to plot the spectrogram.
   In the part I, it is not required to produce the frequency in hertz. I do not use the $ks/2\pi$ to stand the frequency domain. But in part two, I do this transfer to find the specific hertz frequency.

3.2 *Part II*

The steps for analyzing both pieces of music will be the same, the only difference is the different data.

1. Build the basic variable and import data.
   Import the audio data using audioread in `Matlab`. After importing the data, I can find the size for each of the music track. Because both size number and number of sample points are even, I can directly set n equals to that number. For L, it is just the length or how many seconds of the music piece: 16 for the first and 14 for the second. After getting L and n, I define the variable t, k and ks as the same as before in the part I.

   Tslide here will set from 0 to L, and the transform rate is set to be 0.1 here. Although this will take a lot of time to run this program, compared with the rate equals to 0.5, I find the rate 0.1 will give the full information I need for both time and frequency domain. Also, I choose the width of the window to be -40 and the type of window to be normal Gaussian, which also gives me very good information for both domain.

2. Build `for` loop to do the transform and save all the spectral data into spectral vector.

   After having all the parameters, I build a `for` loop like what I did in part I and do the same thing in the `for` loop just like in part I.

3. Plot the spectrogram and figure out the best y limit when plot.

   After having all the data points, I can plot the spectrogram. When I first plot the graph, it is better to set no limits for y-axis. And after I got the first draft of the plot, I can find which part or which area of the music notes is brighter, and then zoom in for that specific area of our first plot to find the specific music score. Then I produce a plot including both brighter and less brighter parts, which are the overtones of the brighter part. This will be my second plot for analysing the overtones and differences.

   There is one important thing that I did when I making the plot. I did $ks/(2 * \pi)$ transfer here, because the ks is in Fourier domain which is $\omega = 2\pi f$ associated with 2π. If I want to know the specific hertz of the music notes, I need do the transformation or just divide by 2π to get

the actual hertz frequency(o 4000).

There is one more thing I need mention, which is the way of changing the limit of the axis for y and finding the brighter part is not a theoretical way to deal with this problem. It is a convince way but not that strict. If I want to do it theoretically, I need first find the central frequency for each of the music note and then build a filter with central frequency to filter music notes out in the frequency domain. After getting the actual frequency data, I can then compare with music scale to find what the music score is. In this project, because the music is not that complicated and the brighter part is very easy to observe by eye, I will not go through the step such as filtering as that has already been done in project 1 (marble).

4 COMPUTATION RESULTS

4.1 *Part I*

Controling the width a = -20 and $\tau = 0.1$, I create the following spectrogram:

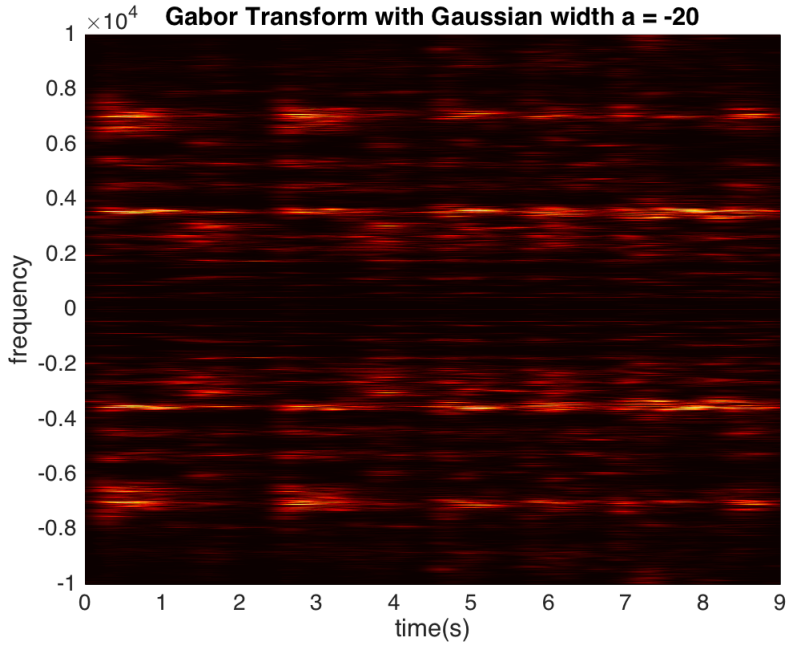

Figure 4: Spectrogram for Gaussian when a=-20 and $\tau = 0.1$

From this plot, it is clear that when at a specific time area, I can find some frequency information in the time area which combines the information for time and frequency together.

Now let's try to change the width a in this Gaussian window. Figure 5.

Because in my Gaussian equation, there is a negative sign before the value a, so the larger the absolute value of a is, the smaller the window is. From the figure 5, when I have $a = 0$, which is just do the FFT directly, I will have a very nice resolution in frequency domain and have no information in time domain. When I have a equals to -0.2, which is a wider window that take more frequency and less time information, the plot shows I will have a good resolution in frequency domain instead of time domain. As I continue making the window smaller, I will have good resolution for the time domain and will have poor resolution for the frequency domain, shown in the last

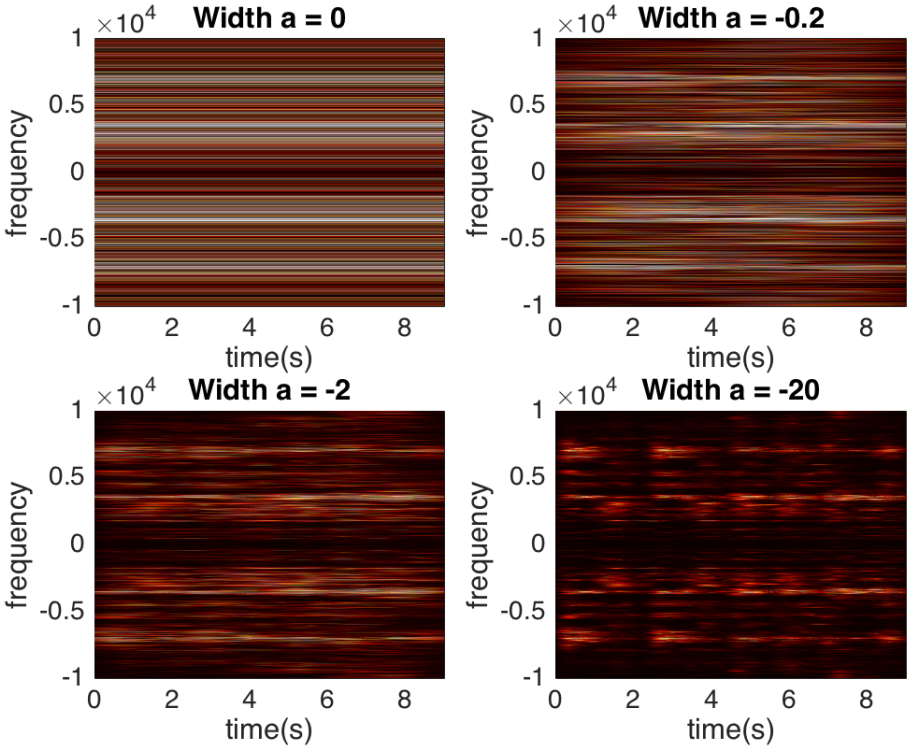

Figure 5: Spectrogram for Gaussian when a vary

plot with a equals to -20. This is the same as I talked before that there is a trade off between time and frequency.

Now let's discover what happen if I change the translation rate τ, figure 6.

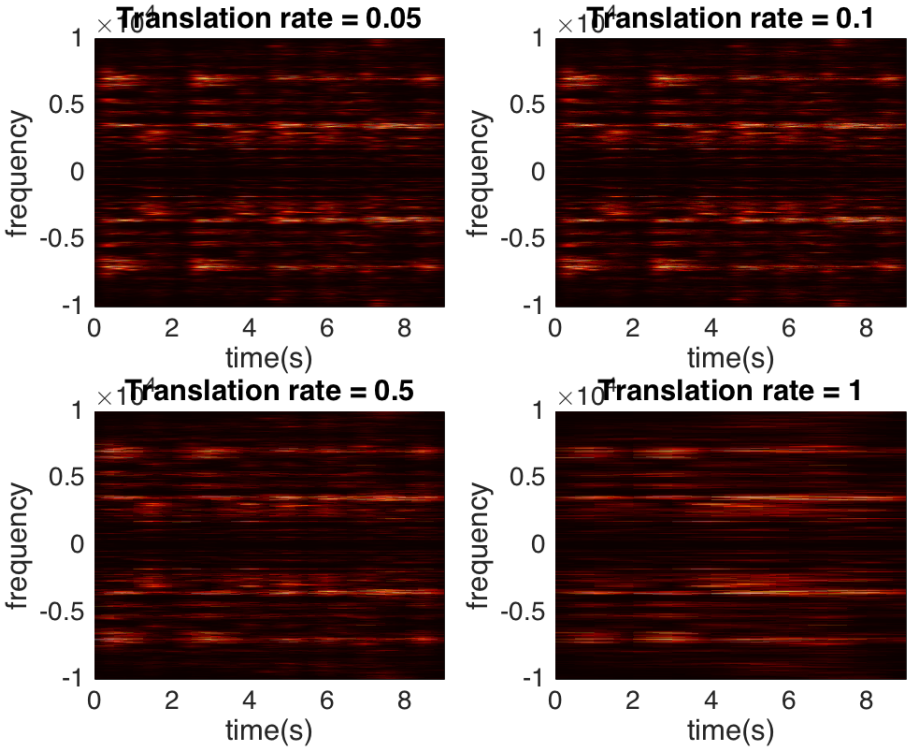

Figure 6: Spectrogram for Gaussian when τ vary

When I have a very small translation rate, which means I have a lot of sample data, also named oversampling, the resolution looks good just like before. When we control the a steady and do not make the rate larger, I will find that the resolution becomes worse. The reason this situation happens is when I have a very big rate like 1, I only have 10 windows to analyze, less data will cause worse resolution in both time and frequency domain. Does this mean that I want the rate as small as possible? No, because for example, if I have rate which is 0.01, I will have 901 windows to analyze. This will take a lot of time and huge amount of work. Also, discovering from the figure 6, the two plots look like same when rate is 0.05 and 0.1. So I need to find small rate which is right, but I need to find the rate that is not very small to increase our work but make the same plot at end.

After exploring the effects by different widths and translation rates, I finally look at the effects by different type of windows.

I use three windows in this problem: Gaussian, Super Gaussian and Mexican hat.

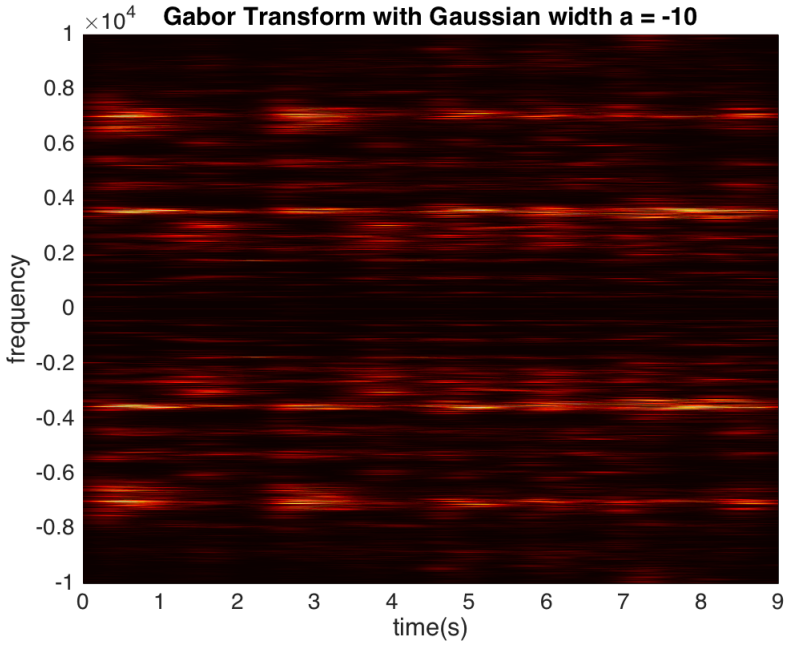

Figure 7: Spectrogram for Gaussian when a=-10, $\tau = 0.1$

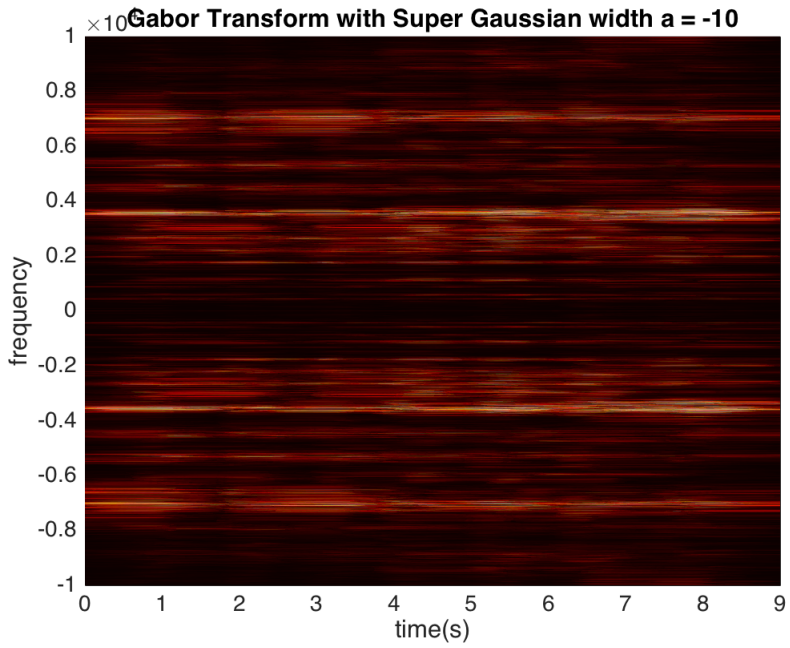

Figure 8: Spectrogram for Super Gaussian when a=-10, $\tau = 0.1$

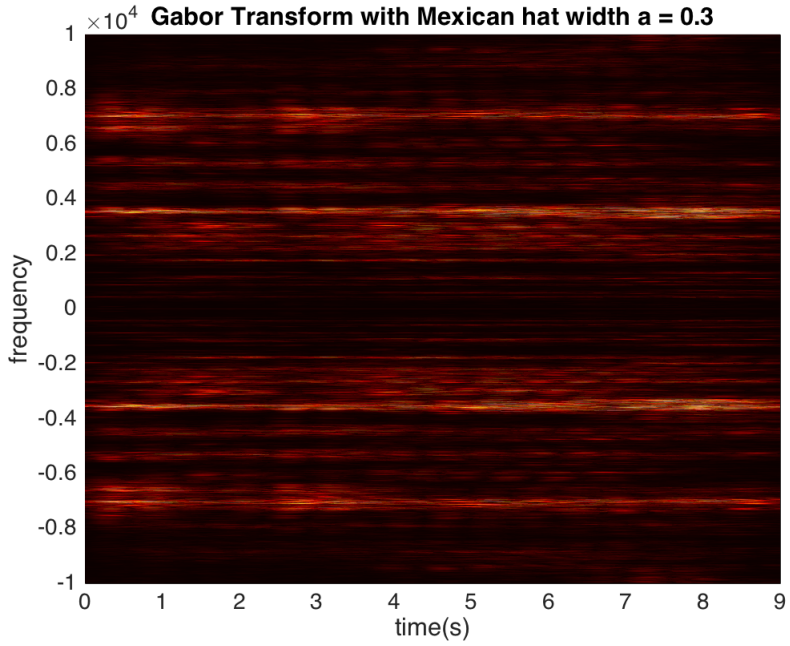

Figure 9: Spectrogram for Mexican hat when a=0.3, $\tau = 0.1$

It is hard to detect by naked eye, but it is obvious that the Gaussian have a good resolution in time when other two have good resolutions in frequency. In fact, I cannot compare the previous Gaussian with the Mexican hat because the different width values, but if I set the same width value for Mexican hat, the window fails to do the job. Maybe more work researches need to be done related to this part. But at least, super Gaussian have good resolution in frequency and normal Gaussian have good resolution in time.

4.2 *part II*

Use the Gabor transform with the Gaussian window, where width is -40 and $\tau = 0.1$, and this means I have a total of 1601 windows for the piano music and 1401 windows for the recorder music. I also try some width like -10, -20 and associated with $\tau = 0.5$ or $\tau = 0.1$. Those case do not give very good resolution for the time domain, which means I cannot observe how many music notes from the plot because that will be a line. After getting the best resolution parameters, I do the same thing as in the previous part and sketch the spectrogram.

For the piano case, I have the plot.

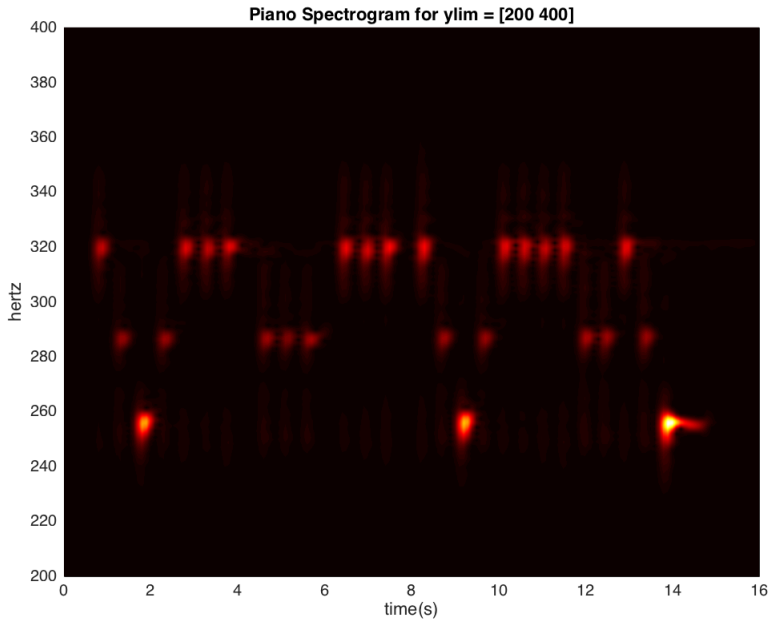

Figure 10: Spectrogram for Piano when a=-40, $\tau = 0.1$

In the figure 10, I zoom in at a specific area for y from 200 to 400. The reason is when I ploted the graph without any specific limit for y-axis, I found this area is the area most brighter. The less brighter music notes will be mentioned later. One thing need to pay attention here is that If I plot the spectrogram with *ks*, Fourier domain, I have no idea about what the hertz is, Because that domain is from -10000 to 10000. In order to find the hertz for the music, I use *ks* divided by 2π to match the hertz notes.

Thus from the plot we can easily read the notes in hertz:

320Hz, 280Hz, 260Hz, 280Hz, 320Hz, 320Hz, 320Hz, 280Hz, 280Hz, 280Hz, 320Hz, 320Hz, 320Hz, 320Hz, 280Hz, 260Hz, 280Hz, 320Hz, 320Hz, 320Hz, 320Hz, 280Hz, 280Hz, 320Hz, 280Hz, 260Hz

For total 26 music notes. Compared with the music scale that has been given, we have the approximate music score:

E4, D4, C4, D4, E4, E4, E4, D4, D4, D4, E4, E4, E4, E4, D4, C4, D4, E4, E4, E4, E4, D4, D4, E4, D4, C4.

I notice the piano is a little bit out of tone because the frequency is a little lower than the actual frequency of the music score.

For the recorder, I do the same thing as I did for the piano part. The only difference is the area I zoom in, because this part has high frequency, I zoom in the most bright part and have the plot figure 11.

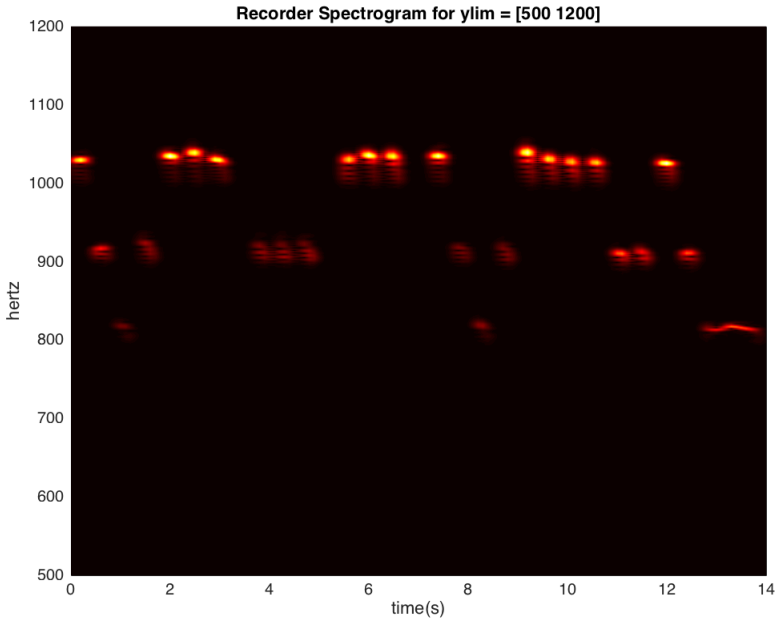

Figure 11: Spectrogram for Recorder when a=-40, $\tau = 0.1$

From this plot I have the hertz:

1000Hz, 900Hz, 800Hz, 900Hz, 1000Hz, 1000Hz, 1000Hz, 900Hz, 900Hz, 900Hz, 1000Hz, 1000Hz, 1000Hz, 1000Hz, 900Hz, 800Hz, 900Hz, 1000Hz, 1000Hz, 1000Hz, 1000Hz, 900Hz, 900Hz, 1000Hz, 900Hz, 800Hz.

For total 26 music notes and then transfer to music score, I have:

B6, A6, G5, A6, B6, B6, B6, A6, A6, A6, B6, B6, B6, B6, A6, G5, A6, B6, B6, B6, B6, A6, A6, B6, A6, G5

I notice that the recorder is out of tone because the frequency is higher than the actual frequency of the music note.

To observe the difference between the recorder and piano, let's look at a wider area for each of the plot, figure 12 and figure 13.

Maybe it is very hard to find the difference. But in general, I have the piano music notes in lower frequency(Hz) domain and the recorder in higher frequency(Hz) domain. Also, if I look at the overtones for each type, piano has a little bit more overtones than the recorder has. One thing that can also be observed from the two plot or even the previous two, figure 9 and figure 10, will be more clear. The music notes for the piano with the same hertz are in the same line, but the music notes for the recorder with the same hertz are not very strictly in the same line: some notes are higher and some are lower. The length of the notes also give us very useful information that how long in seconds that note has been played.

For the overtone, I know the definition that if I am playing a note at frequency ω_0, an instrument will generate overtones at $2\omega_0$, $3\omega_0$,$\cdots$ and so forth. For the piano, I can easily find 3 observably overtones for 250Hz, overtone at

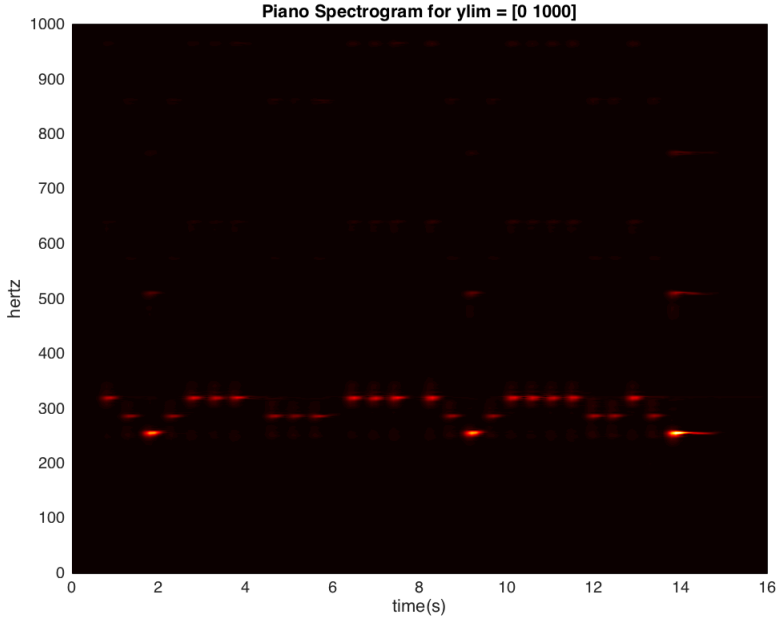

Figure 12: Spectrogram for Piano when a=-40, $\tau = 0.1$ with wider area

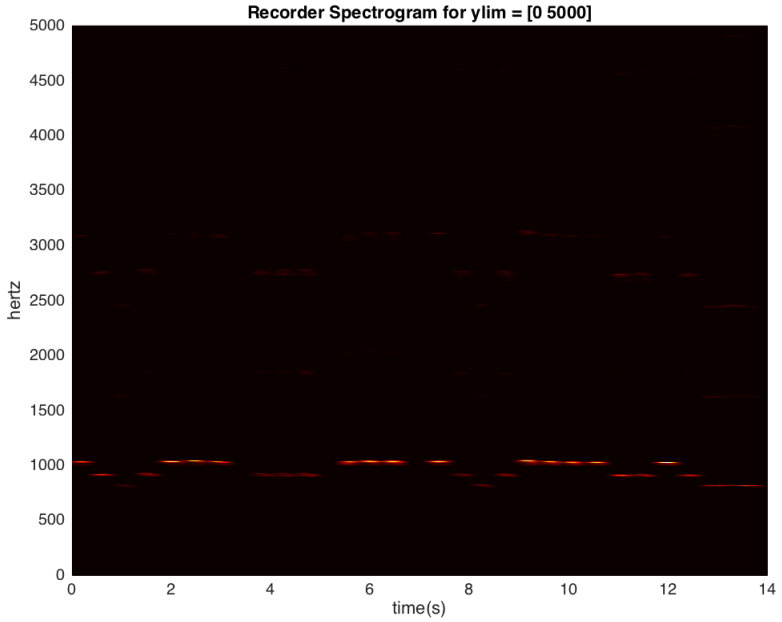

Figure 13: Spectrogram for Recorder when a=-40, $\tau = 0.1$ with wider area

500Hz. And I also observe that the 3 250Hz are brighter than other music notes in the same area, that may be the reason the overtones been created and very easy to observe.

For the recorder, it is a little hard to see any overtones, because all the music notes are not that bright to make overtones in its twice frequency. Also, it may because that the music is not played in instrument, it cannot produce very strong overtones.

5 SUMMARY AND CONCLUSION

By using the Gabor's transform with different types of windows, I can now have the information for the time domain and frequency domain at the same time. But the resolution may decrease as the other resolutions increase. Also, the width and the translation rate and even the type of windows will affect the resolution in bot time domain and frequency domain. In conclusion, when I need to use the Gabor's transform, I need to find the suitable kernel or the type of windows first. Then I decid how many information I need for each domain and the value of parameters by specific applications cases. Also using this method to analyze the music, I can recreate the music and find the difference or overtones for the music.

6 APPENDIX A

MATLAB Functions Used and Brief Implementation Explanations[2]

1. `fft`
   Fourier transform the time domain into frequency domain.
2. `audioread`
   Read the audio into the matlab as a data set.
3. `fftshift`
   Change into 2π domain
4. `ifftshift`
   Change back to the regular domain
5. `abs`
   Take the absolute value
6. `xlabel`
   Label the information for x-axis
7. `ylabel`
   Label the information for y-axis
8. `title`
   Label the information for the title
9. `pcolor`
   Plot the surface view
10. `Shading`
    Color shading mode.
11. `interp`
    Resample data at a higher rate using lowpass interpolation.
12. `gca`
    Get handle to current axis.
13. `Colormap hot` Sets the current figure's colormap to Black-red-yellow-white color map.
14. `figure`
    Number the figure
15. `audioplayer(Y, Fs)`
    Creates an audioplayer object for signal Y, using sample rate Fs. A handle to the object is returned.

[2] Copy from the Matlab

7 MATLAB CODES

7.1 *Part I*

```
 clear all; close all; clc
load handel
v = y'/2;
v = v(1:73112);
% plot((1:length(v))/Fs,v);
% xlabel('Time [sec]');
% ylabel('Amplitude');
% title('Signal of Interest, v(n)');
%
% p8 = audioplayer(v,Fs);
% playblocking(p8);

L=9; n=73112;
t2=linspace(0,L,n+1); t=t2(1:n);

k=(2*pi/L)*[0:(n/2-1) -n/2:-1];
ks=fftshift(k);

vt=fft(v);

% figure(1), plot(t,v)
% figure(2), plot(ks,abs(fftshift(vt)))

%% filter

%g=exp(-0.1*(t-4).^2); %Gaussian
% g=exp(-0.1*(t-4).^10); %Super Gaussian
% figure(3), %plot(t,S,'k',t,g,'r','Linewidth',[2])

% Spec=[];
tslide=0:0.1:9;
% a=[0 0.2 2 20];
a=[20 20 20 20];
in=[0.05 0.1 0.5 1];
for jj=1:4
Spec=[];
tslide=0:in(jj):9;
for j=1:length(tslide)
g=exp(-a(jj)*(t-tslide(j)).^2); %Gaussian
% g=exp(-a*(t-tslide(j)).^10); %Super Gaussian
vg=g.*v;
vgt=fft(vg);
Spec=[Spec; abs(fftshift(vgt))];

% subplot(3,1,1)
% plot(t,v,'k',t,g,'r','Linewidth',[2])
% xlabel('time(s)')
% ylabel('v(t),g(t)')
% axis([0 9 -1 1])
% subplot(3,1,2)
% plot(t,vg,'Linewidth',[2])
% xlabel('time(s)')
% ylabel('v(t)*g(t)')
% axis([0 9 -1 1])
% subplot(3,1,3)
```

```
% plot(ks,abs(fftshift(vgt))/max(abs(vgt)),'Linewidth',[2])
% xlabel('frequency')
% ylabel('FFT(v*g)')
% drawnow
end

subplot(2,2,jj)
pcolor(tslide,ks,Spec.'), shading interp
set(gca,'Ylim',[-10000 10000],'Fontsize',[14])
title(['Translation rate = ',num2str(in(jj))])
xlabel('time(s)')
ylabel('frequency(Hz)')
colormap hot

end

% %% spectrogram plot
% figure(4)
% pcolor(tslide,ks,Spec.'), shading interp
% set(gca,'Ylim',[-10000 10000],'Fontsize',[14])
% xlabel('time(s)')
% ylabel('frequency(Hz)')
% colormap hot
```

For different window type:

```
    clear all; close all; clc
load handel
v = y'/2;
v = v(1:73112);

L=9; n=73112;
t2=linspace(0,L,n+1); t=t2(1:n);

k=(2*pi/L)*[0:(n/2-1) -n/2:-1];
ks=fftshift(k);

vt=fft(v);


Spec=[];
tslide=0:0.1:9;
a=10;
for j=1:length(tslide)
g=exp(-a*(t-tslide(j)).^2); %Gaussian
% g=exp(-a*(t-tslide(j)).^10); %Super Gaussian
vg=g.*v;
vgt=fft(vg);
Spec=[Spec; abs(fftshift(vgt))];

end
subplot(3,1,1)
pcolor(tslide,ks,Spec.'), shading interp
set(gca,'Ylim',[-10000 10000],'Fontsize',[14])
xlabel('time(s)')
ylabel('frequency(Hz)')
colormap hot
```

```
Spec=[];
tslide=0:0.1:9;
a=10;
for j=1:length(tslide)
% g=exp(-a*(t-tslide(j)).^2); %Gaussian
g=exp(-a*(t-tslide(j)).^10); %Super Gaussian
vg=g.*v;
vgt=fft(vg);
Spec=[Spec; abs(fftshift(vgt))];
end
subplot(3,1,2)
pcolor(tslide,ks,Spec.'), shading interp
set(gca,'Ylim',[-10000 10000],'Fontsize',[14])
xlabel('time(s)')
ylabel('frequency(Hz)')
colormap hot

Spec=[];
tslide=0:0.1:9;
a=1;
for j=1:length(tslide)
% g=exp(-a(jj)*(t-tslide(j)).^2); %Gaussian
% g=exp(-a*(t-tslide(j)).^10); %Super Gaussian
g=2/(sqrt(3*a)*pi^(1/4))*(1-((t-tslide(j)).^2/a^2)).*exp((-(t-tslide(j)).^2)/(2*a^2));
vg=g.*v;
vgt=fft(vg);
Spec=[Spec; abs(fftshift(vgt))];
% subplot(3,1,1)
% plot(t,v,'k',t,g,'r','Linewidth',[2])
% xlabel('time(s)')
% ylabel('v(t),g(t)')
% axis([0 9 -1 1])
% subplot(3,1,2)
% plot(t,vg,'Linewidth',[2])
% xlabel('time(s)')
% ylabel('v(t)*g(t)')
% axis([0 9 -1 1])
% subplot(3,1,3)
% plot(ks,abs(fftshift(vgt))/max(abs(vgt)),'Linewidth',[2])
% xlabel('frequency')
% ylabel('FFT(v*g)')
%drawnow
end
subplot(3,1,3)
pcolor(tslide,ks,Spec.'), shading interp
set(gca,'Ylim',[-10000 10000],'Fontsize',[14])
xlabel('time(s)')
ylabel('frequency(Hz)')
colormap hot
```

7.2 *part II*

Piano:

```
    clear all; close all; clc
tr_piano=16;  % record time in seconds
y=audioread('music1.wav');
Fs=length(y)/tr_piano;
```

```
% plot((1:length(y))/Fs,y);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title('Mary had a little lamb (piano)');
% drawnow
% p8 = audioplayer(y,Fs); playblocking(p8);
y=y.';
L=16; n=701440;
t2=linspace(0,L,n+1); t=t2(1:n);

k=(2*pi/L)*[0:(n/2-1) -n/2:-1];
ks=fftshift(k);

yt=fft(y);


Spec=[];
tslide=0:0.1:16;
a=40;
for j=1:length(tslide)
g=exp(-a*(t-tslide(j)).^2);%Gaussian
yg=g.*y;
ygt=fft(yg);
Spec=[Spec; abs(fftshift(ygt))];
end
figure(1)
pcolor(tslide,ks/(2*pi),Spec.'), shading interp
set(gca,'Ylim',[200 400],'Fontsize',[10])
colormap hot
xlabel('time(s)')
ylabel('hertz')
title('Piano Spectrogram for ylim = [200 400]')
figure(2)
pcolor(tslide,ks/(2*pi),Spec.'), shading interp
set(gca,'Ylim',[0 1000],'Fontsize',[10])
colormap hot
xlabel('time(s)')
title('Piano Spectrogram for ylim = [0 1000]')
ylabel('hertz')
```

Recorder:

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

 y=y.';
L=14; n=627712;
t2=linspace(0,L,n+1); t=t2(1:n);

k=(2*pi/L)*[0:(n/2-1) -n/2:-1];
ks=fftshift(k);

yt=fft(y);
```

```
Spec=[];
tslide=0:0.1:14;
a=40;
for j=1:length(tslide)
g=exp(-a*(t-tslide(j)).^2);%Gaussian
yg=g.*y;
ygt=fft(yg);
Spec=[Spec; abs(fftshift(ygt))];
end
figure(1)
pcolor(tslide,ks/(2*pi),Spec.'), shading interp
set(gca,'Ylim',[500 1200],'Fontsize',[10])
colormap hot
xlabel('time(s)')
ylabel('hertz')
title('Recorder Spectrogram for ylim = [500 1200]')
figure(2)
pcolor(tslide,ks/(2*pi),Spec.'), shading interp
set(gca,'Ylim',[0 5000],'Fontsize',[10])
colormap hot
xlabel('time(s)')
title('Recorder Spectrogram for ylim = [0 5000]')
ylabel('hertz')
```