AMATH 482 HW 2

Liyuan Tang

February 2019

Abstract

In this homework, I will analyze a portion of Handel's Messiah with time-frequency analysis. First, I will use Gábor filtering to produce the spectrogram of the piece of work. Then I will change some parameters to see how they affect the spectrogram. Finally, I will use different Gábor windows in order to find how they are going to influence the spectrograms. In the next part, we are given the data of music on recorder and piano. First, I am going to use Gábor filtering as well to reproduce the music score for these pieces. And then, I will compare the difference between a recorder and a piano.

1 Introduction

There are two parts in homework. The first part is to analyze a portion of Handel's Messiah with time-frequency analysis. I will first use Gábor filtering to generate the transformed data and plot the spectrogram. Then I will change the window size and time step to see how they affect the spectrogram since there is a trade off between the accuracy in time and frequency. Finally, starting with the Gaussian window, I will change different Gábor windows such as Mexican hat wavelet and Shannon window to generate three spectrograms and compare how they look like.

The second part is to reproduce the music score of two given data which play the same piece of *Mary had a little lamb* on piano and recorder. I will use Gábor filtering to get the frequency in order to reproduce the music score and compare the difference between them.

2 Theoretical Background

2.1 Gábor Transforms

The Gábor transforms is known as the short-time Fourier transform(STFT). It is defined as:

$$\mathcal{G}[f](t,\omega) = \tilde{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)\bar{g}(\tau - t)e^{-i\omega\tau}d\tau = (f, \bar{g}_{t,\omega}), \tag{1}$$

where the bar denotes the complex conjugate of the function. The function g(τ - t) represents a time filter for localizing the signal over a specific window of time. Thus, the Gábor transform is a function of t and ω, that allows a fast and easy way to analysis both the time and frequency properties of a given signal. It established two key principles which are the translation of a short-time window and scaling of that window to capture finer time resolution. One of the applications of Gábor transform is to produce a spectrogram that represents the signal in both the time and frequency domain.

It also has several properties. The Gábor transform is linear and can be inverted. And there is a trade off between time and frequency resolutions which means high accuracy in one of them will make the expense resolution in the other parameter.

2.2 Wavelet Analysis

Since the Gábor transform trades off the accuracy in time and frequency, we can make a modification to allow the scaling window to vary in order to improve the time resolution. We can first start with a low-frequency by using a broad scaling window. Then we can shorten the window width in order to extract out high-frequencies and better time resolution. By keeping a catalogue of the extracting process, both time and frequency resolution of a given signal can be obtained. This is the fundamental principle of wavelet theory. It begins with a function called mother wavelet:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a}) \tag{2}$$

where $a \neq 0$ and b are real constants. a is the scaling parameter and b is the translation parameter.

In this problem, we are going to use Mexican hat wavelet which is:

$$\psi(t) = \frac{2}{\sqrt{3\sigma}\pi^{1/4}}(1-(\frac{t}{\sigma})^2)e^{-\frac{t^2}{2\sigma^2}} \tag{3}$$

where σ represents the width. This wavelet is the second moment of a Gaussian in the frequency domain.

And we will use a Shannon (step function) filter. The Shannon filter is simply a step function with value of unity within the certain size and zero outside of it.

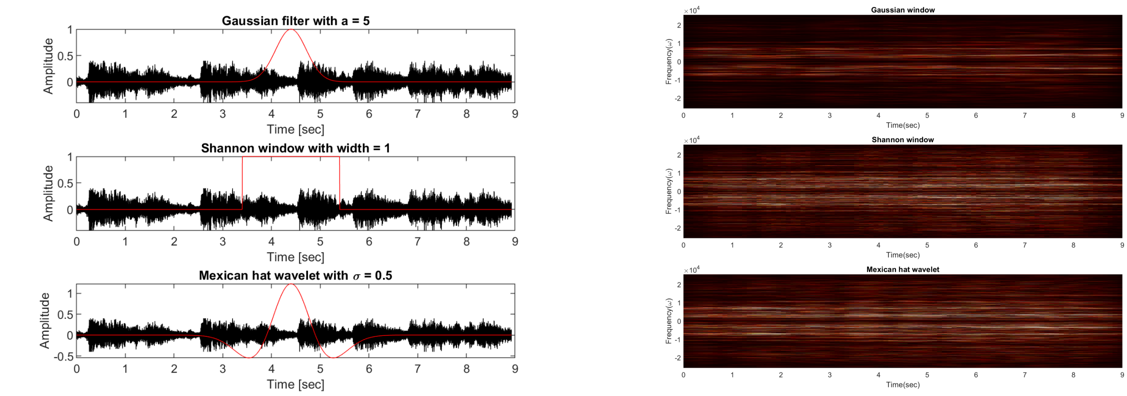
3 Algorithm Implementation and Development

For the first part, we need to load the data by using "load handel" which gives us the data "v" and "Fs" that indicates the sample rate for the data. Since it has odd number of points which is 73113, I just removed the last point for Fourier transform. Then, I created the vector "t" by using "(1:length(v))/Fs" which will generate a row vector of time. I created the wavenumber "k" from [0, L] to [-L, 0] and rescaled it by 2π / L since "fft" assumes working on a 2π periodic domain, where L is the length of v. And I used "fftshift" to get "ks" just like I did in the last homework.

Since we are asked to use Gábor filtering to produce the spectrograms, I made a vector called "tslide" which translates the filter. I used "tslide=0:0.1:9" which time step is 0.1 second. Then I used a for loop which enable me to translate the Gaussian window:

$$g = e^{-a(t-tslide(j))^2}. \tag{4}$$

And then I multiplied this filter by the data "v", and applied Fourier transform to the result. I took the absolute value and used "fftshift" to shift the result since Fourier



(a) The shape of three different Gábor windows

(b) The corresponding spectrograms

Figure 1: Three different Gábor windows and the corresponding spectrograms

transform will shift the data and we need to shift it back. Finally, I stored the results in a matrix, vgt_spec, at every time step, so in the end I had a matrix with 91 $\times$ 73112 which each row represents the data at different time step. With this matrix, I can plot the spectrogram by "pcolor(tslide, ks, vgt_spec.'), shading intertp" and "colormap(hot)". Here we need to take transpose of vgt_spec because the row represents time and column represents the frequency when I store the data but what we want is the column represents the time and row represents the frequency.

The next part of this question is to change the window width of Gábor transform. Here, we can just change the value of "a" in the filter which will affect the width. I changed it to several values to see how it gonna affect the spectrograms. Notice that when we increased the value of "a", the width will decrease due to the negative sign in Equation(4).

Then, it asked to explore the spectrograms with oversampling and undersampling. That is to change the time step in "tslide". The orginal time step is 0.1, I changed it to 0.05, 0.5 and 1 to compare the difference in the corresponding spectrograms while remaining all the other values as the same.

The last question is to different Gábor windows such as Mexican wavelet and a step function window. For the Mexican wavelet, I used the Equation (3) with $\sigma = 0.5$. For the step-function window, I created a function:

$$s = (abs(t - tslide(j)) < width). \tag{5}$$

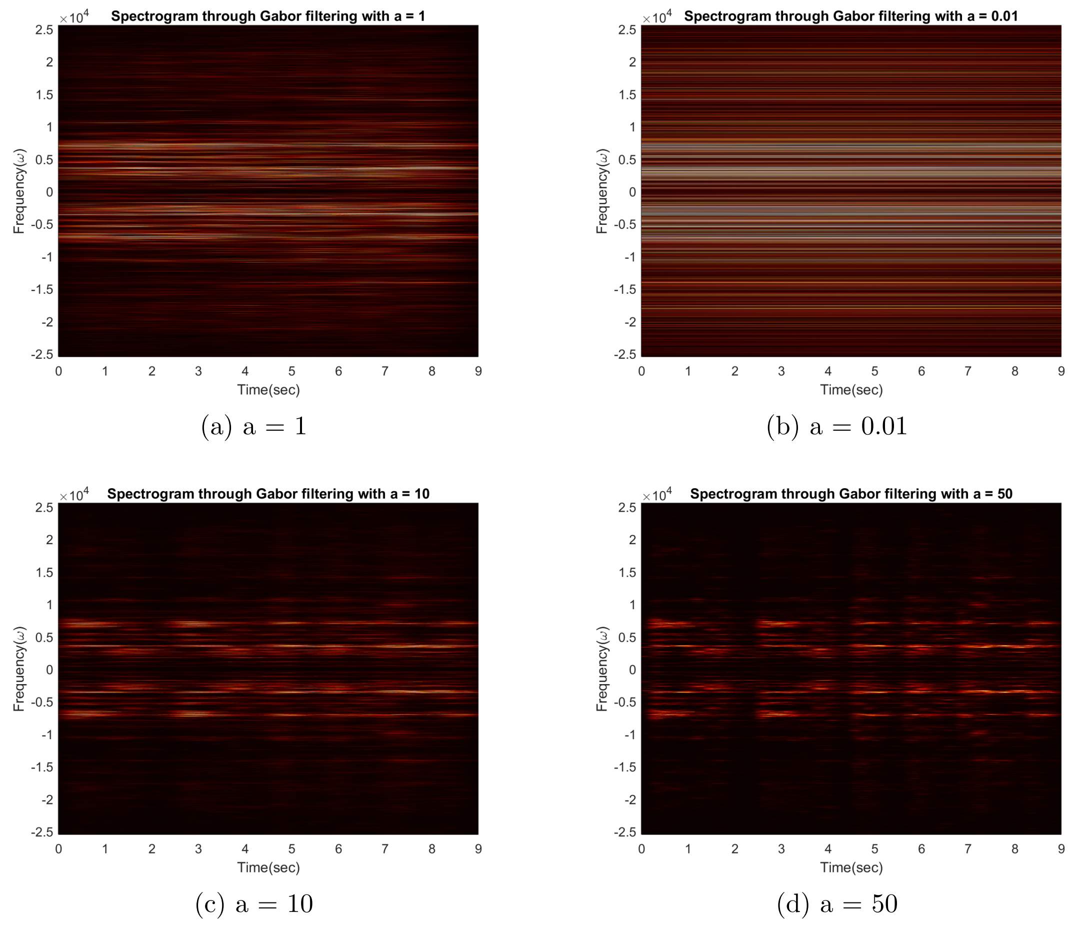
This function will return a vector with the same size as t. The index of it will be 1 if the corresponding index of t - tslide(j) is less than width and 0 otherwise. As a result, the 1s will occur in [tslide(j) - width, tslide(j) + width], centered at tslide(j). The shapes of three Gábor windows are in Figure 1(a). Then we can use the results to produce three different spectrograms.

The part 2 of the question asked us to reproduce the music score for those two music. As the setup, I used "y1 = audioread(music1.wav)" to load the data. "F1s=length(y1)/tr_piano" which Fs1 is the data points per second and tr_piano is the record time in seconds. Then I created t1, k1 and ks1 just as what I did in the previous part. I made tslide = 0:0.1:16.

Just as part 1, I used a for loop. For every iteration, I used the Gaussian filter(Equation (4)) times the data $y1$ and then took fft of the result. Here I choose a = 10 since it gave me a clear figure. After that, I used [V, I] = max(abs(Sgt)) in order to find the maximum index of the data. And I used that index to find corresponding wavenumber and took the absolute value. Here we need to plug the index in k because k is the shifted wavenumbers which matches the shifted data after applying "fft". Since we want the music scale in Hertz while the wavenumbers are measured in angular frequency(ω), we need to divide the result by 2π. Then, I stored the value in a matrix in every iteration and I plotted the frequency versus tslide in the end(Figure 4(a)). Then, I tried to match the frequency with the music scale provided in the problem. And I did almost the same thing for the data of recorder except changing the some of the variables. I choose $a = 40$ and tslide = 0:0.1:14. I think these two value give me a relative clear figure of the frequency(Figure 4(b)).

4 Computational Results

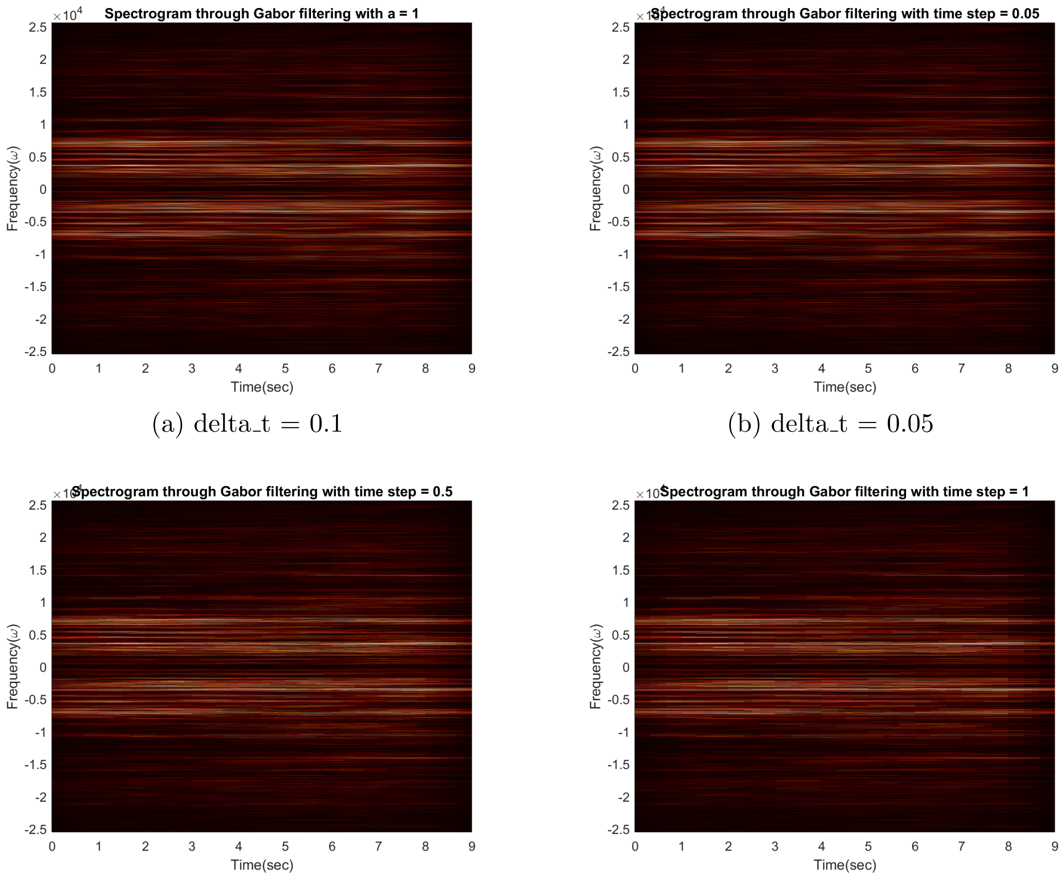
Part 1:
1. The spectrogram is Figure 2(a). I used Gaussian window described by Equation(4) with $a = 1$ and the time step is 0.1.
2. In Figure 2, there are 4 different window width. As the a increases, the width of the window will decrease. When a is large, the figure is clear for locating where the signal is



(a) a = 1 (b) a = 0.01

(c) a = 10 (d) a = 50

Figure 2: Spectrograms produced by different Gaussian windows by only changing the parameter a in the Equation(4).



(a) delta_t = 0.1

(b) delta_t = 0.05

(c) delta_t = 0.5

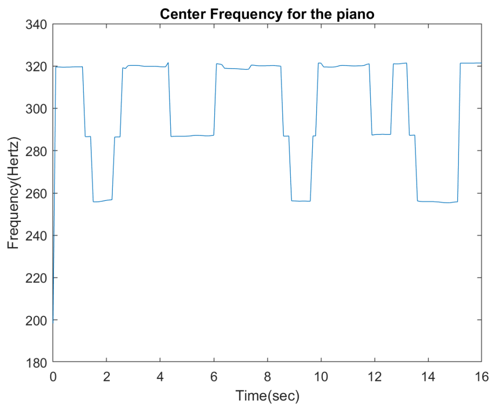
(d) delta_t = 1

Figure 3: Spectrograms produced by different Gaussian windows by only changing the parameter a in the Equation(4).

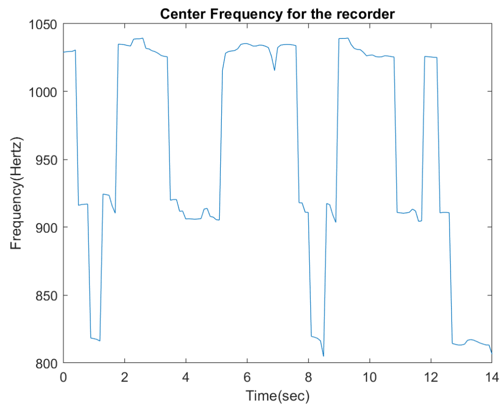
in time content. By comparing those spectrograms, it can be seen that a broader window (smaller a value) will give more accurate information on frequency content which means less accurate of localization of where the signal is in time content.
3. I changed the time step in "tslide" while keeping "a" the same in order to get oversampling(small time step) and undersampling(large time step). The four spectrograms in Figure 3 are generated by different time steps. As we can see, when the time step is small, we have many duplicate data and the spectrograms look almost the same. However, when the time step is big, such as 1, we are having undersampling. In this situation, there are not enough data to produce a spectrogram with good resolutions in time and frequency.
4. I used the Gaussian window with $a = 1$, the Mexican hat wavelet(Equation 3) with $\sigma = 0.5$ and Shannon window(Equation 5) with $width = 1$ whose shapes are shown in Figure 1(a). I generated three spectrograms in Figure 1(b). These three spectrograms look similar to each other. But the step function have a relatively bad resolution in time and frequency. Gaussian window gives the best resolution in both time and frequency due to the Heisenberg's uncertainty principle.

Part 2:

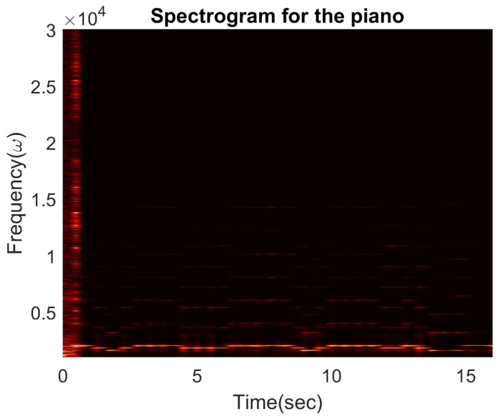
1. I used Gaussian window with $a = 55$ for the piano and $a = 40$ for the recorder. And the time step for the piano is 0.25 while the time step for the recorder is 0.1. Figure 4(a) represents the center frequency of the piano and Figure 4(b) represents the center frequency of the recorder.



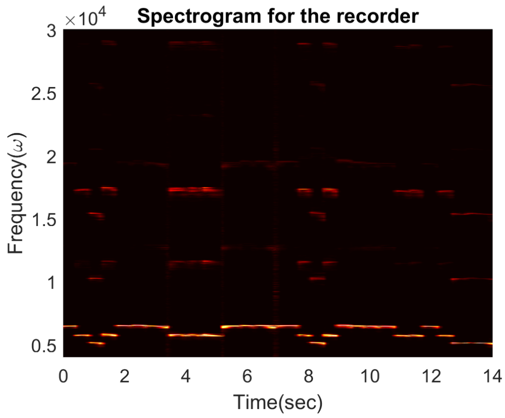
(a) Music score for the piano



(b) Music score for the recorder



(c) The spectrogram of the piano



(d) The spectrogram of the recorder

Figure 4: Center frequency and spectrograms for the piano and recorder

Based on the provided figure of music scale, the music scale for the piano is: **EDCDEEEDDDEEE EDCDEEEEDDEDC.** And the music scale for the recorder is: **BAGABBBAAABBB BAGABBBBAABAG.**

2. From Figure 4 (a) and (b), we can see that they have a similar shape but the frequency of the recorder is much higher than the frequency of the piano. Also, from the spectrograms, it can be seen that piano has more overtones than the recorder.

5 Conclusion

In this homework, I used Gábor filtering to produce the spectrogram of the sound signal. When I changed the window width, I found broader windows will give more accurate information on frequency content and less accurate information on time content. I changed the time step of moving the filter. When the time step is big(undersampling), the spectrogram will have bad resolution in time and frequency. When the time step is small(oversampling), we will have many duplicate data. I used different Gábor windows such as Shannon window and Mexican hat wavelet which generated similar spectrograms.

In the second part, we had two data played the same piece of *Mary had a little lamb* of piano and recorder. I used Gábor filtering to reproduce the music score by finding the center frequency in Hertz. The difference between the piano and the recorder is that the recorder has a higher frequency and piano has more overtones.

6 Appendix A

1. *vgt*=fft(*vg*). "fft" computes the discrete Fourier transform (DFT) of the vector *vg* using a fast Fourier transform (FFT) algorithm and it will return the Fourier transform of the vector.

2. ks = fftshift(k). "fftshift" will rearrange a Fourier transform by shifting the zero-frequency component to the center. "ifftshift" is its inverse.

3. pcolor(tslide,ks,vgt_spec.'). It draws a pseudocolor plot of the elements of vgt_spec.' at the locations specified by tslide and ks.

4. colormap(map). It sets the colormap for the current figure to the colormap specified by map. In this homework, we used colormap(hot) which "hot" is a predefined colormap in Matlab.

5. y1 = audioread('music1.wav'). This command reads data from the file named "music.wav", and returns sampled data, y1, and a sample rate for that data, Fs.

7 Appendix B

```
clear all; close all; clc;
load handel
v = y'/2;
v = v(1:73112);
L = 9;
n = length(v);
t = (1:length(v))/Fs;
k=(2*pi/(L))*[0:(n-1)/2 -(n)/2:-1]; ks=fftshift(k);

%% 1.Gaussian. width = 1, Δ_t = 0.1
vgt_spec=[];
tslide=0:0.1:9;
a = 1;
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    vg=g.*v; vgt=fft(vg);
    vgt_spec=[vgt_spec; abs(fftshift(vgt))];
end
pcolor(tslide, ks, vgt_spec.'), shading interp
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
title('Spectrogram through Gabor filtering with a = 1');
colormap(hot)

%% 2. Gaussian. width = 50, Δ_t = 0.1
close all;
vgt_spec=[];
tslide=0:0.1:9;
a = 50;
for j=1:length(tslide)
```

```
    g=exp(-a*(t-tslide(j)).^2);
    vg=g.*v; vgt=fft(vg);
    vgt_spec=[vgt_spec; abs(fftshift(vgt))];
end
pcolor(tslide,ks,vgt_spec.'), shading interp
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
title('Spectrogram through Gabor filtering with a = 50');
colormap(hot)
%% 2. Gaussian. width = 0.01, Δ_t = 0.1
close all;
vgt_spec=[];
tslide=0:0.1:9;
a = 0.01;
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    vg=g.*v; vgt=fft(vg);
    vgt_spec=[vgt_spec; abs(fftshift(vgt))];
end
pcolor(tslide,ks,vgt_spec.'), shading interp
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
title('Spectrogram through Gabor filtering with a = 0.01');
colormap(hot)
%% 2. Gaussian. width = 10, Δ_t = 0.1
close all;
vgt_spec=[];
tslide=0:0.1:9;
a = 10;
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    vg=g.*v; vgt=fft(vg);
    vgt_spec=[vgt_spec; abs(fftshift(vgt))];
end
pcolor(tslide,ks,vgt_spec.'), shading interp
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
title('Spectrogram through Gabor filtering with a = 10');
colormap(hot)

%% 3. Gaussian. width = 1, Δ_t = 0.05
close all;
vgt_spec=[];
tslide=0:0.05:9;
a = 1;
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    vg=g.*v; vgt=fft(vg);
    vgt_spec=[vgt_spec; abs(fftshift(vgt))];
end
pcolor(tslide,ks,vgt_spec.'), shading interp
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
title('Spectrogram through Gabor filtering with time step = 0.05');
colormap(hot)
%% 3. Gaussian. width = 1, Δ_t = 0.5
close all;
vgt_spec=[];
```

```matlab
tslide=0:0.5:9;
a = 1;
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    vg=g.*v; vgt=fft(vg);
    vgt_spec=[vgt_spec; abs(fftshift(vgt))];
end
pcolor(tslide,ks,vgt_spec.'), shading interp
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
title('Spectrogram through Gabor filtering with time step = 0.5');
colormap(hot)
%% 3. Gaussian. width = 1, Δ_t = 1
close all;
vgt_spec=[];
tslide=0:1:9;
a = 1;
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    vg=g.*v; vgt=fft(vg);
    vgt_spec=[vgt_spec; abs(fftshift(vgt))];
end
figure(2)
pcolor(tslide,ks,vgt_spec.'), shading interp
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
title('Spectrogram through Gabor filtering with time step = 1');
colormap(hot)

%% 4:
% Gaussian filter
close all;
vgt_spec=[];
tslide=0:0.1:9;
a  = 5;
for j=1:length(tslide)
    g= exp(-a.*(t - tslide(j)).^2);
    vg=g.*v;
    vgt = fft(vg);
    vgt_spec=[vgt_spec; abs(fftshift(vgt))];
end
subplot(3,1,1);
pcolor(tslide,ks,vgt_spec.'), shading interp
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
title('Gaussian window')
colormap(hot)
% step function
vgt_spec=[];
tslide=0:0.1:9;
width = 1;
for j=1:length(tslide)
    s = (abs(t - tslide(j)) < width);
    vg=s.*v;
    vgt = fft(vg);
    vgt_spec=[vgt_spec; abs(fftshift(vgt))];
end
subplot(3,1,2);
```

```
pcolor(tslide,ks,vgt_spec.'), shading interp
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
title('Shannon window');
colormap(hot)
% Mexican hat wavelet
vgt_spec=[];
tslide=0:.1:9;
a = 0.5;
for j=1:length(tslide)
    m = 2/(sqrt(3*a)*(pi)^(1/4))*(1-((t - tslide(j))/a).^2)...
        .* exp(-(t - tslide(j)).^2 / (2*a^2));
    vg=m.*v; vgt = fft(vg);
    vgt_spec=[vgt_spec; abs(fftshift(vgt))];
end
subplot(3,1,3)
pcolor(tslide,ks,vgt_spec.'), shading interp
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
title('Mexican hat wavelet');
colormap(hot)

%% plot 3 filters in the same figure
j = 45;
g= exp(-5.*(t - tslide(j)).^2);
s = (abs(t - tslide(j)) < width);
m = 2/(sqrt(3*a)*(pi)^(1/4))*(1-((t - tslide(j))/a).^2)...
        .* exp(-(t - tslide(j)).^2 / (2*a^2));
figure(2);

subplot(3,1,1), plot(t,v,'k',t,g,'r');
title('Gaussian filter with a = 5');
xlabel('Time [sec]');
ylabel('Amplitude');
subplot(3,1,2), plot(t,v,'k',t,s,'r');
xlabel('Time [sec]');
ylabel('Amplitude');
title('Shannon window with width = 1');
subplot(3,1,3), plot(t,v,'k',t,m,'r');
xlabel('Time [sec]');
ylabel('Amplitude');
title('Mexican hat wavelet with \sigma = 0.5');

%% Part 2.
clear all; close all; clc;
%% piano
tr_piano=16; % record time in seconds
y1=audioread('music1.wav'); Fs1=length(y1)/tr_piano;
y1 = y1'/ 2;
L1 = tr_piano;
n1 = length(y1);
t1 = (1:length(y1))/Fs1;
k1=(2*pi/L1)*[0:n1/2-1 -n1/2:-1]; ks1=fftshift(k1);
freq1 = [];
Sgt_spec=[];
tslide=0:0.1:L1;
a = 10;
for j=1:length(tslide)
```

```
    g=exp(-a*(t1-tslide(j)).^2);
    Sg=g.*y1;
    Sgt=fft(Sg);
    [V, I] = max(abs(Sgt));
    freq1 = [freq1; abs(k1(I))/(2*pi)];
    Sgt_spec=[Sgt_spec; abs(fftshift(Sgt)) / max(abs(Sgt))];
end

figure(1)
plot(tslide,freq1)
title('Center Frequency for the piano');
xlabel('Time(sec)');
ylabel('Frequency(Hertz)');

figure(2)
pcolor(tslide, ks1, abs(Sgt_spec).'), shading interp
set(gca,'Ylim',[1000 30000],'Fontsize',[14])
title('Spectrogram for the piano')
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
colormap(hot)

%% record
tr_rec=14; % record time in seconds
y2=audioread('music2.wav'); Fs2=length(y2)/tr_rec;
y2 = y2';
n2 = length(y2);
t2 = (1:length(y2))/Fs2;
k2=(2*pi/(tr_rec))*[0:(n2/2-1) -n2/2:-1]; ks2=fftshift(k2);
freq2 = [];
Sgt_spec2=[];
tslide2=0:0.1:14;
a = 40;
for j=1:length(tslide2)
    g2=exp(-a*(t2-tslide2(j)).^2); % Gabor
    Sg2=g2.*y2; Sgt2=fft(Sg2);
    [V, I] = max(abs(Sgt2));
    freq2 = [freq2; abs(k2(I)/(2*pi))];
end

figure(3)
plot(tslide2, freq2);
title('Center Frequency for the recorder');
xlabel('Time(sec)');
ylabel('Frequency(Hertz)');

figure(4)
pcolor(tslide2, ks2, abs(Sgt_spec2).'), shading interp
title('Spectrogram for the recorder')
set(gca,'Ylim',[4000 30000],'Fontsize',[14])
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
colormap(hot)
```