Gabor Transform and Sound Analysis

Yiling Sun

AMATH 482 HW02 Winter 2019

abstract

Garbor Transform is a signal analysis method improved from Fourier Transform. It contains a special wavelet function that discretes the data to gain both time and frequency information. This report explores the method of Garbor Transform, briefly introducing three type of wavelets and its application on sound frequency analysis with music sample *Mary had a liitle lamb*.

1. Introduction and Overview

Gabor Transform is a method to analyze both frequency and time properties of given signal, developed from Fourier Transform. In application, the type of wavelet window, width of window and the translation of window significantly affect the result resolution in time and frequency domain, and sometimes it will lead to the problem of "oversampling" and "undersampling". This report is divided into two parts to explore the characteristics and application of Garbor Transform.

In Part I, we are given a 9 second portion of the classic work Handels Messiah. We will modify the parameters of the width and translation of wavelet window and examine how they affect the time and frequency resolution through spectrograms. Three types of wavelet would be explored: Gaussain wavelet, Mexican Hat wavelet and Shannon wavelet.

In Part II, We apply the Garbor Transform with Gaussain wavelet on the sound samples Mary had a little lamb on both the piano and recorder. We reproduce the music scores of these two piece of music through their spectrograms. Also, we examine their spectrograms and compare the overtone to see the difference between the piano and recorder.

2. Theoretical Background

2.1 Fourier Transform

Fourier Transform translates the data between time domain and frequency domain. Basically, it can represent function as sum of sines and cosines. The Fourier transfrom is defined as

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

, where $e^ikx = coskx + isinkx$(Euler's formula) is the Kernel describing the oscillatory behavior, and k is the wave number.

In the view of Matlab, the commands for Fourier Transform and its reverse are **fft(u)** and **ifft(u)** with operation account $O(n \log n)$. To be praticle in Matlab, finite number of points frequencies on finite domain is required. Also, there are two restrictions worth-noticing. First, based on the characteristic of $coskx+isink$, the periodic boundary condition is necessary. Second, the Fourie mode of frequencies is in 2π periodic domain. Data need to be scaled into $2\pi/L$ periodic domain beforehand.

2.2 Garbor Transform/short-time Fourier transform (STFT)

Due to the inability of Fourier Transform to localize the various frequencies in time domain, the Garbor Transform, or short-time Fourier Transform is introduced with the aim of localizing various frequencies in both time and frequency domain.

Mathematically, based on the Fourier Transform, Gabor adds a new term $g(\tau - t)$, as a time filter, to the Fourier kernel for locating the signal over a specific time window. Hence, Garbor Transform is defined as

$$\mathcal{G}[f](t,\omega) = \tilde{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)\overline{g}(\tau - t)e^{-i\omega\tau}\, d\tau = (f, \overline{g}_{t,w}), \tag{2}$$

where the bar on g denotes the complex conjugate of function. Different types of wavelets are applied as time window here. Fundamentally, the time window is centered on τ and with the width **a**, which control the translation and the scaling of the window. The information outside the time window is severely damped. So, once τ moves through data, we can extract the signal information in that specific time range, which determined by the width **a**.

The Garbor Transform gains the signal information in both time and frequency domain by discretizing the time and frequency domain. But it still comes with the disadvantage of Heinsenberg uncertainty principle that the higher resolution in time, the lower resolution in frequency. This is a trade off must occur on the fixed time filtering window.

2.3 Wavelets

In order to gain more time and frequency information, wavelet theory is introduced to make improvement on Garbor Transform. There are various type of wavelets developed. The wavelet function starts with the **mother wavelet**:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a}), \tag{3}$$

where a $\neq 0$ and b are real constants. The parameter a controls the width of window whereas the parameter b denotes the translation. To use moving filter in Garbor transform, we simply add a translation term to the center of basis wavelet window. In the following, 3 types of wavelet are introduced and examined: Guassain wavelet, Mexican Hat wavelet and step-function window. Figure 1-3 illustrates their basic shape in order.

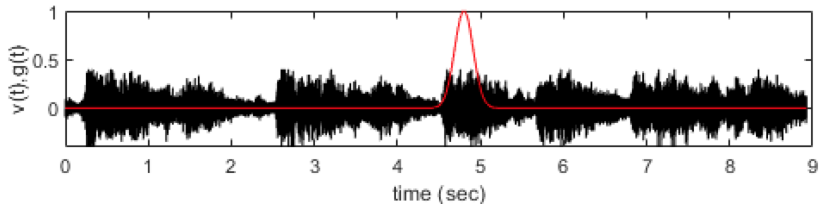

Figure 1: Illustration of the shape of Gaussian wavelet with width paramter 1 and translation 0.1 on the frequencies data of Handels Messiah

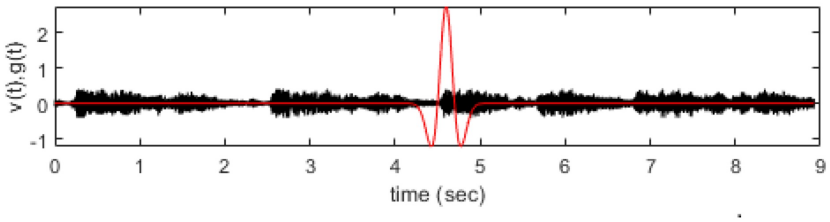

Figure 2: Illustration of the shape of Mexican Hat wavelet

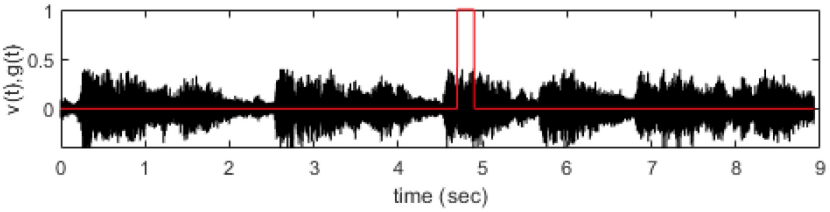

Figure 3: Illustration of the shape of Shannon wavelet

(a) Guassain Window

Guassain filter is the simplest filter to eliminate unnecessary information. With the width of window a and the the location of its center τ, its function is defined as

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

Due to the negative sign before the parameter a, the larger parameter a, the narrower the window width.

(b) Mexican hat wavelet

Mexican hat wavelet is commonly used. It has excellent localization properties in both time and frequency due to the minimal time-bandwidth product of the Gaussian function.[1] Different from the Guassain wavelet, the smaller parameter a relates to the narrower window width. Its function is defined as

$$g(t-\tau) = \frac{2}{\sqrt{3a}\pi^{\frac{1}{4}}}(1-(\frac{(t-\tau)}{a})^2)e^{-\frac{((t-\tau))^2}{2a^2}} \tag{5}$$

(c) Shannon wavelet

The Shannon filter is simply a step function with value of unity within the transmitted band and zero outside of it.[2] It suppresses all frequencies outside of a given filter box.

$$g(t-\tau) = |t-\tau| < a \tag{6}$$

2.3 Spectrogram

A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. It can be generated by Fourier Transform or Garbor Transform. For Garbor Transform, the spectrogram is composed of the time-frequencies data extracted from filtering at each window center. It gives us an idea that what frequency is prominent at that given time span. It is usually depicted as heat map.

[1] Description from 582 notes by J. Nathan Kutz
[2] Description from 582 notes by J. Nathan Kutz

3. Algorithm Implementation and Development

Part I

Prepare Data

We are given a 9 second audio data with 73113 sample points in row and the first and last data is 0. It is periodic, so I delete the last point that our data become a even number 73112 in size, which is also our Fourier mode that is equal to the length of given time span. Also, we set the length L to be 9. Then, I rescale Fourier Domain by $2\pi/L$, since FFT assumes 2π periodic domain.

Explore the window type, window width and window translation

We have three variables, window type, window width [**a**] and window translation [**translation**]. I use the following method to test the effect of these variables. The translation determines the stops of the window center over the time span, which discretes time domain in parts. Given the translation, I build a **forloop** to allow my wavelet to extract the frequencies information centered at time stops. The data we get from this filtering is the frequency information within the small time span. We collect the time-frequency information at each stops into a matrix **vgtspec** that set beforehand. Then, we use **pcolor(tslide,ks,vgtspec.')** to create its spectrogram. The **vgtspec** need to be transposed.

In order to see how these variables affect on the time-frequency resolution, I simply make modification on these parameters and change my wavelet.

Part II

Prepare Data

We are given files play the song *Mary had a little lamb* on both the piano and recorder. I read in music by using **audioread**. They have length **L** 16 and 14. Still, the Fourier mode is equal to the length of time span. One important thing here is to set the k space and rescale it by 2π for finding the frequency later.

Find music score

This section basically follows the method as Part I. We use Gaussain filter and choose the time slide to be 0.25, since the sound of one note is longer than 0.25 second. For the width, I would test several different width to find the best by compare the spectrogram. After I find my idea parameters a, I filter out the overtone by picking up the maximum frequency over each small time span with **max** method and locate the specific frequency in the Fourier domain k we set beforehand. At the same time, we translate the angular

frequency into Hertz, dividing them by $2 * pi$, to match the music notes. Then, I can have a plot for the music score over time.

4. Computational Results

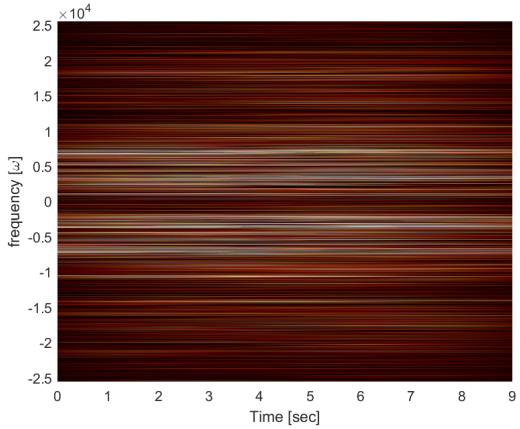

Figure 4: spectrogram under Guassain filter with parameter a=0.1

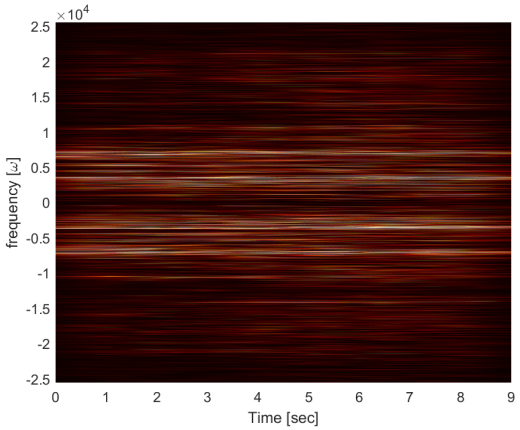

Figure 5: spectrogram with Guassain filter with parameter a=1

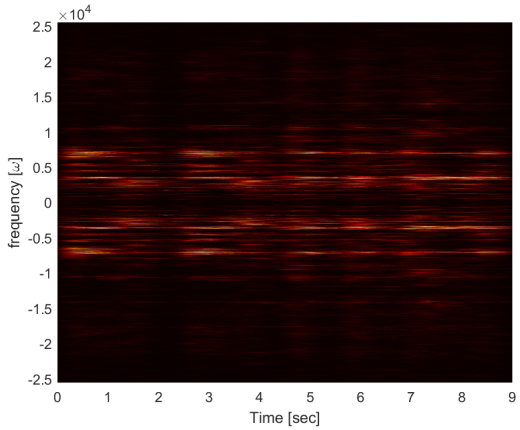

Figure 6: spectrogram with Guassain filter with parameter a=20

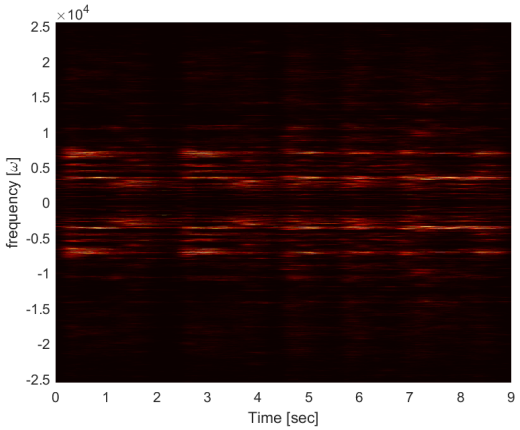

Figure 7: spectrogram with Guassain filter with parameter a=40

Part I

Start with Gaussain filter and parameters, width 1 and translation 0., Figure 1 shows the signal v(t) and the wavelet g(t) over time span. As we observed, this filter seems too wide for the time span. Figure 5 gives the spectrogram under this wavelet. Data are stretched out horizontally. We cannot tell much information from this spectrogram. To compare, I generate the spectrogram with width 0.1, 20 and 40 in Figure 4,6,7. Figure 4 gives us a very good resolution in frequency but almost no information in time, While Figure 3 and Figure 4 have good time resolution but trade off the frequency information. After all, 20 looks like a more

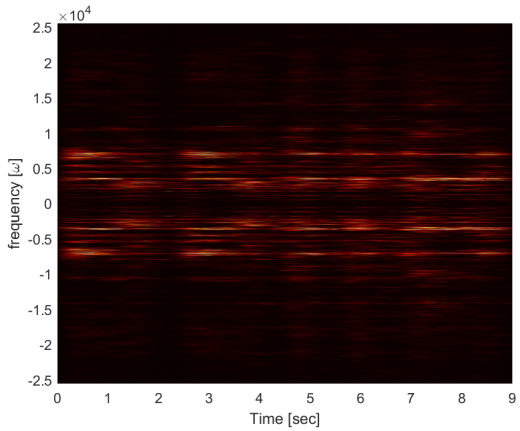

Figure 8: Spectrogram under Guassain filter with a=20 and translation 0.01

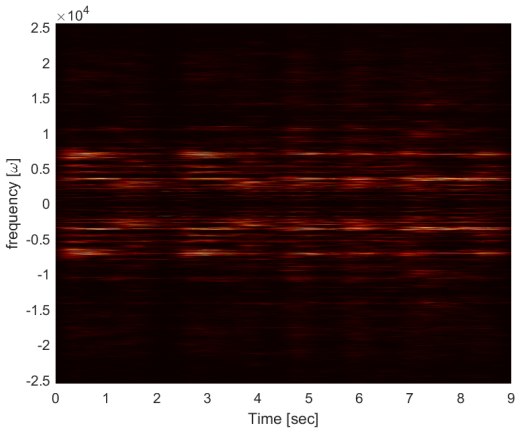

Figure 9: spectrogram under Guassain filter with a=20 and translation 0.05

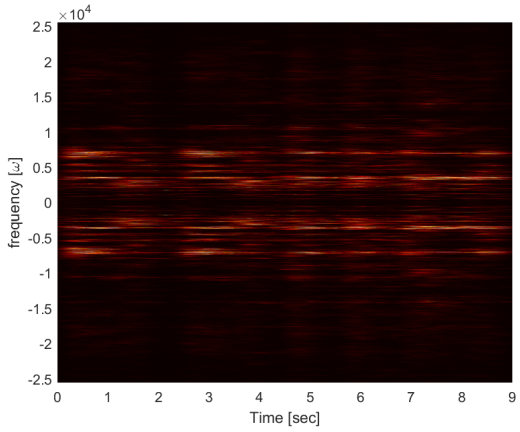

Figure 10: spectrogram under Guassain filter with a=20 and translation 0.1

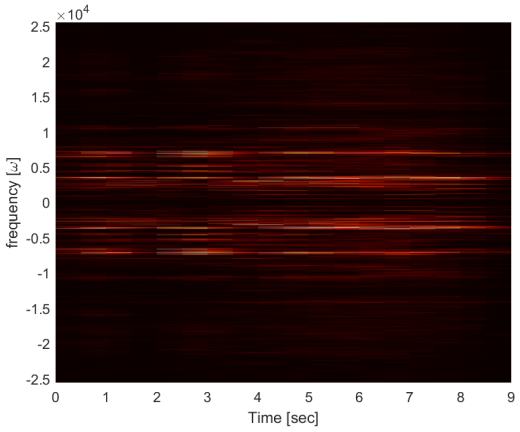

Figure 11: spectrogram under Guassain filter with a=20 and translation 1

reasonable parameter for width with translation 0.1.

Then, with fixed width parameter 20, I start to test the translation. Figure 8-11 demonstrate the spectrograms with Gaussain filter with translation 0.01, 0.05, 0.1, 1. When I have very small translation, we cannot find much difference from the spectrogram between 0.05 and 0.01. This case is so called "**oversampling**". When we already get enough sample data, we do not need the translation to be smaller. When I have large translation, 1, I get bad resolution on both time and frequency, Because, compared to my window width, the translation is too large. With only 10 set of sample, the frequency information is limited in each data set. This case is so called "**undersampling**"

Then, I test the Mexican Hat wavelet. With translation 0.1, as the width parameter a decrease, the resolution in frequency become better and better. In Figure 12, I found that, under the same time resolution, Mexican Hat wavelet can give a better frequency resolution than that with Gaussian filter. Whereas, still under the same time resolution, translation 0.1, Shannon wavelet also can give a better frequency resolution that that with Gaussain wavelet, but not as good as Mexican hat wavelet, as in Figure 13.

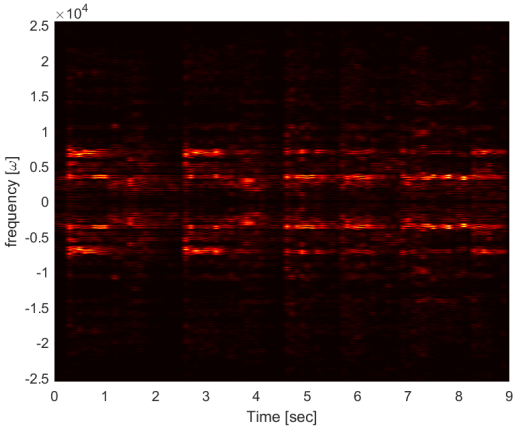

Figure 12: Spectrogram with Mexican hat wavelet with t=0.1 and a = 0.01

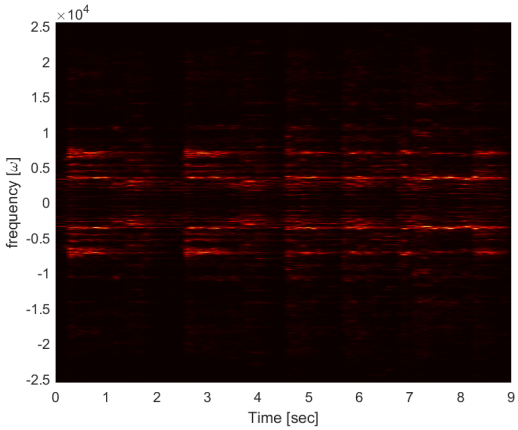

Figure 13: Spectrogram with Shannon wavelet with t=0.1 and a = 0.05

Part II

For the given music on piano and recorder, with Gaussain wavelet and window width parameter 40 and translation 0.25 in both cases, Figure 12 and 13 give a big picture of their spectrograms. For overtone, which is 2 ω,3 ω produced along with the fundamental tone ω When a resonant system is excited. The fundamental tone are around 200∼350 for the piano and 700∼1100 for recorder. Also, more sustainable overtone exists on piano, whereas on the recorder, the overtone almost disappear after the second overtone.

The plots of music score in time for piano and recoreder are made in Figure 16 and Figure 17. we read the notes for piano are:

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 the recorder,:

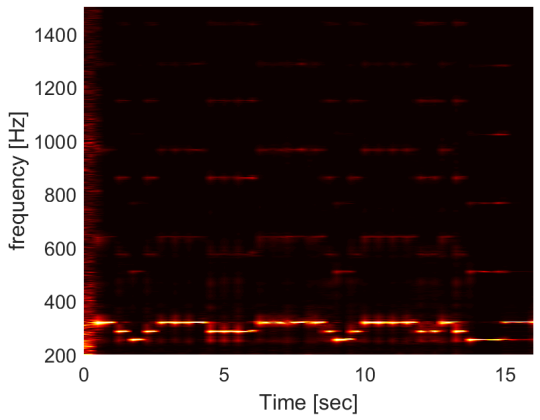

Figure 14: Spectrogram for the music on piano with Gaussain wavelet t=0.1 a=40

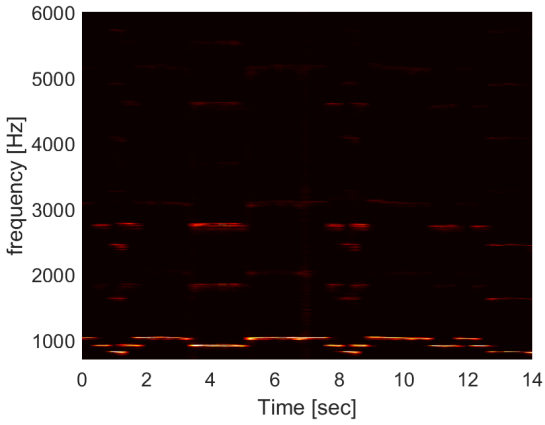

Figure 15: Spectrogram for the music on recorder with Gaussain wavelet t=0.1 a=40

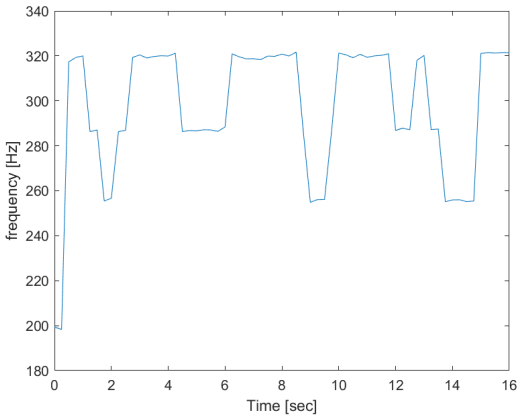

Figure 16: Music score for the music on piano

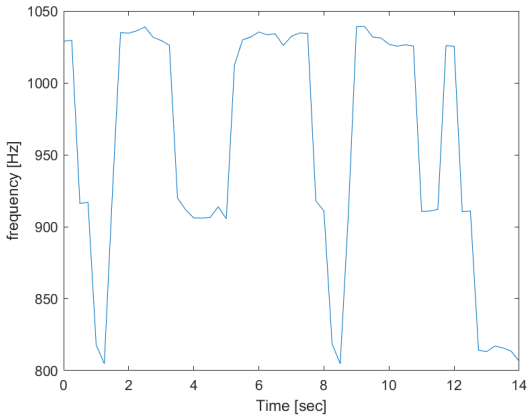

Figure 17: Music socre for the music on recorder

1030Hz, 920Hz, 820Hz, 920Hz,1030Hz,1030Hz,1030Hz,920Hz,920Hz, 920Hz, 1030Hz,1030Hz,1030Hz,1030Hz,920Hz,820Hz,920Hz,1030Hz,1030Hz,1030Hz,1030Hz, 920Hz,920Hz,1030Hz,920Hz,820Hz

Compared to the music scale in Herz, we get the music score for piano:

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

for the recorder:

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

5. Summary and Conclusions

After all, we can conclude that when we do the wavelet transform, firstly we should have a reasonable translation based on the data length (avoid "oversampling" and "undersampling"), then, we can find the desired frequency resolution by modify the width of wavelet. Also, we should take type of wavelets in to consideration, because different type of wavelet have various shape characteristics that aim at different feature data. In our part I case, it is observed that Mexican Hat wavelet gives the best time-frequency result. Also, Garbor transform is a good tool to do the sound frequency analysis. It can reproduce the music score and analyze the timbers of instrument.

6. Appendix A MATLAB functions used and brief implementation explanation

Y = fftn(X) returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm.[3]

X = ifftn(Y) returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm.[4]

Y = fftshift(X) rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.[5]

a = abs(A) return the absolute value of each element in A

colormap sets the colormap for the current figure to the colormap specified by map. **pcolor** draws a pseudocolor plot of the elements of C at the locations specified by X and Y.

plot3(x,y,z,...) plots points with x,y,x coefficient in 3D and connects points up with line.

[3]Description from MathWorks
[4]Description from MathWorks
[5]Description from MathWorks

7. Appendix B MATLAB codes

Part I

```
clear all; close all; clc;
load handel
v = y'/2;

L = 9;  %length of the piece
v = v(1:length(v)-1);  % periodic
n = length(v); %Fourier mode
t = (1:length(v))/Fs;
k = (2*pi/L)*[0:n/2-1 -n/2:-1]; ks = fftshift(k);

% % original data
% subplot(2,1,1)
% plot(t,v)
% xlabel('Time [sec]');
% ylabel('Amplitude');
% title('Signal of Interest');
%
% % FFT
% vt=fft(v);
% subplot(2,1,2)
% plot(ks,abs(fftshift(vt))/max(abs(vt)))
% xlabel('Frequency [\omega]');
% ylabel('Amplitude');
% title('Signal in frequency domain');
% saveas(gcf,'signal.png')

% Gabor filter
a= 0.1;
tslide = 0:0.05:9;
vgt_spec=[];
for j = 1:length(tslide)
    %g = exp(-a*(t-tslide(j)).^2);
    %g = (2/(sqrt(3*a)*pi^(1/4)))*(1-((t-tslide(j)).^2/a^2)).*exp(-(t-tslide(j)).^2/(2*a^2));
    g = (abs(t-tslide(j)) < a);
    vg = g.*v;
    vgt = fft(vg);
    vgt_spec = [vgt_spec; abs(fftshift(vgt))];
    subplot(3,1,1),plot(t,v,'k',t,g,'r')
```

```
    xlabel('time (sec)'), ylabel('v(t),g(t)')
    subplot(3,1,2),plot(t,vg,'k')
    subplot(3,1,3),plot(t,abs(fftshift(vgt))/max(abs(vgt)),'k')
    axis([-50 50 0 1])
    drawnow
    pause(0.1)
end
figure()
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;

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);

%saveas(gcf,'siganl.png')

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

% yt = fft(y);
% figure(2)
% plot(ks,fftshift(abs(yt)));

tslide = 0:0.25:L;
ygt_spec = [];
score = [];
```

```
for j = 1:length(tslide)
    g = exp(-40*(t-tslide(j)).^2);
    yg = g.*y;
    ygt = fft(yg);
    ygt_spec = [ygt_spec; abs(fftshift(ygt))/max(abs(ygt))];

    [M,I]=max(abs(ygt));
    score=[score;abs(k(I))];
%
%     subplot(3,1,1),plot(t,y,'k',t,g,'r')
%     xlabel('time (sec)'), ylabel('y(t),g(t)')
%     subplot(3,1,2),plot(t,yg,'k')
%     subplot(3,1,3),plot(t,abs(fftshift(ygt))/max(abs(ygt)),'k')
%     axis([-50 50 0 1])
%     drawnow
%     pause(0.1)
end
figure()
pcolor(tslide,ks/(2*pi),ygt_spec.'), shading interp
set(gca,'Ylim',[200 350],'Fontsize',[14])
xlabel('Time [sec]');
ylabel('frequency [Hz]');
colormap(hot)
drawnow, hold on
saveas(gcf,'piano.png')

figure()
pcolor(tslide,ks/(2*pi),ygt_spec.'), shading interp
set(gca,'Ylim',[200 1500],'Fontsize',[14])
xlabel('Time [sec]');
ylabel('frequency [Hz]');
colormap(hot)
drawnow, hold on
saveas(gcf,'overtone.png')

% figure()
% plot(tslide, score./(2*pi))
% xlabel('Time [sec]');
% ylabel('frequency [Hz]');
% saveas(gcf,'score.png')
```

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

% yt = fft(y);
% figure(2)
% plot(ks,fftshift(abs(yt)));

tslide = 0:0.25:L;
ygt_spec = [];
score = [];
for j = 1:length(tslide)
    g = exp(-40*(t-tslide(j)).^2);
    yg = g.*y;
    ygt = fft(yg);
    ygt_spec = [ygt_spec; abs(fftshift(ygt))/max(abs(ygt))];

    [M,I]=max(abs(ygt));
    score=[score;abs(k(I))];

    subplot(3,1,1),plot(t,y,'k',t,g,'r')
    xlabel('time (sec)'), ylabel('y(t),g(t)')
    subplot(3,1,2),plot(t,yg,'k')
    subplot(3,1,3),plot(t,abs(fftshift(ygt))/max(abs(ygt)),'k')
    axis([-50 50 0 1])
    drawnow
    pause(0.1)
end
figure()
pcolor(tslide,ks/(2*pi),ygt_spec.'), shading interp
set(gca,'Ylim',[700 1100],'Fontsize',[14])
xlabel('Time [sec]');
ylabel('frequency [Hz]');
```

```
45 colormap(hot)
46 drawnow, hold on
47 saveas(gcf,'record.png')
48
49 figure()
50 pcolor(tslide,ks/(2*pi),ygt_spec.'), shading interp
51 set(gca,'Ylim',[700 6000],'Fontsize',[14])
52 xlabel('Time [sec]');
53 ylabel('frequency [Hz]');
54 colormap(hot)
55 drawnow, hold on
56 saveas(gcf,'overtone2.png')
57 %
58 % figure()
59 % plot(tslide, score./(2*pi))
60 % xlabel('Time [sec]');
61 % ylabel('frequency [Hz]');
62 % saveas(gcf,'score2.png')
```