AMATH 482 Homework 2

By Dexian Chen

Abstract

The report intends to explore the effect of different Gabor windows with different widths and translation. We apply Gabor filter on several pieces of music file to visualize sound signal and figure out the ideal trade-off between time and frequency in Gabor transform.

Introduction and Overview

We'll analyze a 9 second portion of Handel's Messiah with time-frequency analysis. We'll apply Gaussian filter, Mexican wavelet, and Shannon step-function on this piece. By changing filter's width, we'll do a comparison on the spectrogram for part 1. Then we'll analyze the song "Mary had a little lamb" played on the piano and the recorder. Once we have filtered the overtone, we'll reproduce the music score and differentiate the piano and the recorder in the end.

Theoretical Background

Fourier Transform and its inverse are defined as:

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

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

The famous Fourier transform (**See Equation (1)** and **(2)**) allows us to convert signal information from spatial domain to frequency domain. However, the powerful short time Fourier transform has a limitation that we will fail to locate the time information after we convert to frequency domain. Hence, people come up with Gabor transform (**See Equation (3)**) to locate both the time and frequency information in a small window. This sounds like a perfect solution, but we also need to address the trade-off between time and frequency resolution, because the Gabor transform can't give us much information on time and frequency simultaneously.

$$G[f](t,\omega) = \int_{-\infty}^{\infty} \tilde{f}(\tau)\bar{g}(\tau - t)e^{-i\omega\tau}\, dr \;=\; 1, (f, \overline{g_{t,\omega}}) \tag{3}$$

Another background we need to know is different types of Gabor windows. The most frequently used one is **Gaussian Window**. (**See Equation (4))**

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

Gaussian filter is a simple filter with window width a and center at τ .

There are a lot of other filters. For this project, we randomly choose Mexican hat wavelet and Shannon step-function as our extra filters to see the effect on spectrogram. Since we are reading the spectrogram to get information, it is important to understand the graph's meaning.

Spectrogram is defined as the visual representation of the spectrum of frequencies of a signal when it changes in time. In MATLAB, we use the command **pcolor** to draw the spectrogram. A lot of Spectrograms will later be shown as the results of our time-frequency analysis. These spectrograms are the visual representation of the Gabor Transform which gives us time and frequency information.

Algorithm Implementation and Development

In part one, we load the audio file and initialize our data. Particularly, we assign **L** to be the length of the audio, and **n** be the number of measurements or sampling points. Then we scale the frequency domain by 2π because Fast Fourier Transform (FFT) is 2π periodic. Also, we would set up the timespan with equal vector entries with **n**, which is **tslide**. (Appendix B Line 5-20) The most significant process is the looping, which allows us to apply filter on every stopping point in my **tslide** and we can simultaneously store the maximum frequency index by combining them into a matrix. Hence, once we finish looping, we can find the center frequency by locating it with the index matrix built in the iteration. Finally, we use **pcolor** to draw the spectrograms. Finally, we can just play around with different width and translation span on **tslide** to compare the effects on spectrogram.

In part two, we follow the same process of loading the music file, and set up the initial value. I rename L as Lp and Lr to denote length of piano and vice versa. Then I use the same method as part one to find the center frequency of the music by iterating and applying filter on the vector **y**. One thing we should be careful is that whenever we graph, we need to dividing k or ks by 2π to transfer back to unit in Hertz, because we're analyzing music and interested in Hertz instead of frequency.

Computational Results

Part One: I start to explore the effect of Gaussian filter with fixed time translation, and varying width. As we can see in Figure 1, there's a comparison of four different Gaussian windows.

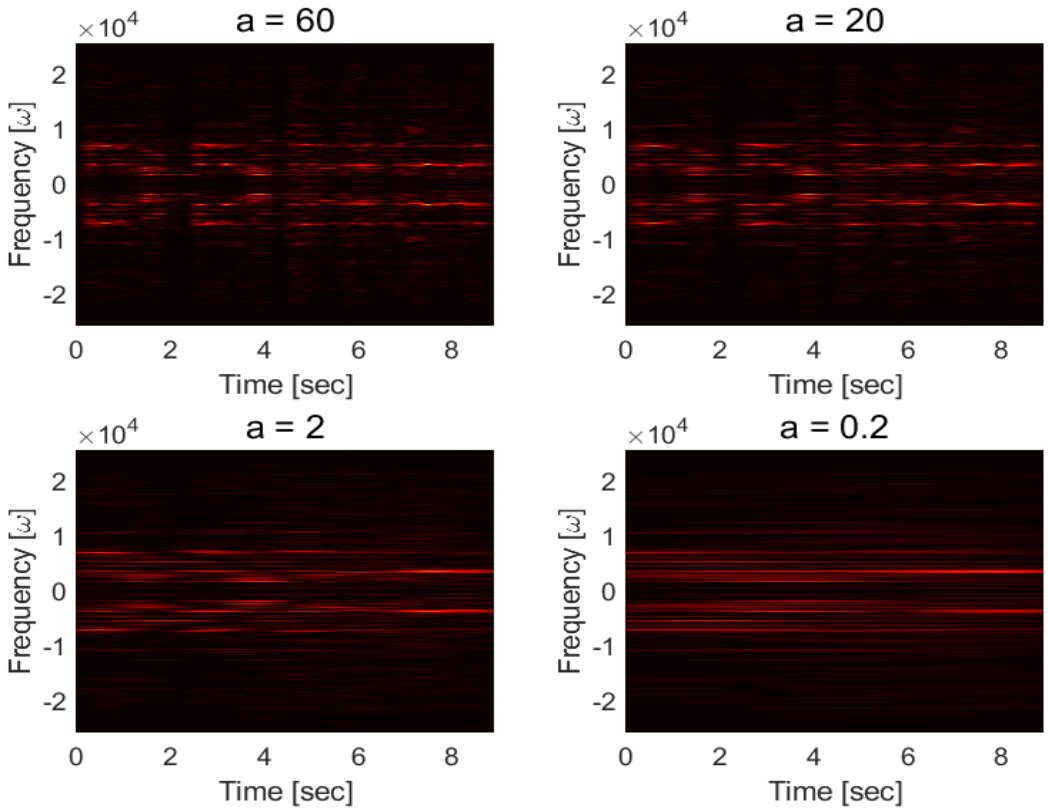

Figure 1: Gaussian Filter Spectrogram with fixed translation 0.1

From observation, it's obvious that the spectrogram is horizontally stretched out and we lose too much resolution on time. Therefore, width 2 and 0.2 are not good choices, and we either choose a = 60 or a = 20 with fixed translation 0.1. We then can conclude that the wider the window is, the higher frequency resolution will be. I also explore the effect of time translation by fixing window width. Figure 2 is a comparison of different time translations with fixed window width 70. In Figure 2, the spectrogram with translation 3 has the worst resolution because the filter iterates through the **tslide** too fast to have appropriate amount of data filtered. Too little data are filtered due to filters translating too fast is called **undersampling**. By comparing the two pictures on the top of Figure 2, we can tell that much information is lost. If we look at the two spectrograms on the bottom of Figure 2, we can tell that these two are very similar. Nothing seems to be wrong, but a very small translation can lead to **oversampling**, which is a

phenomenon that tiny translation causes too much information filtered and being overlapped.

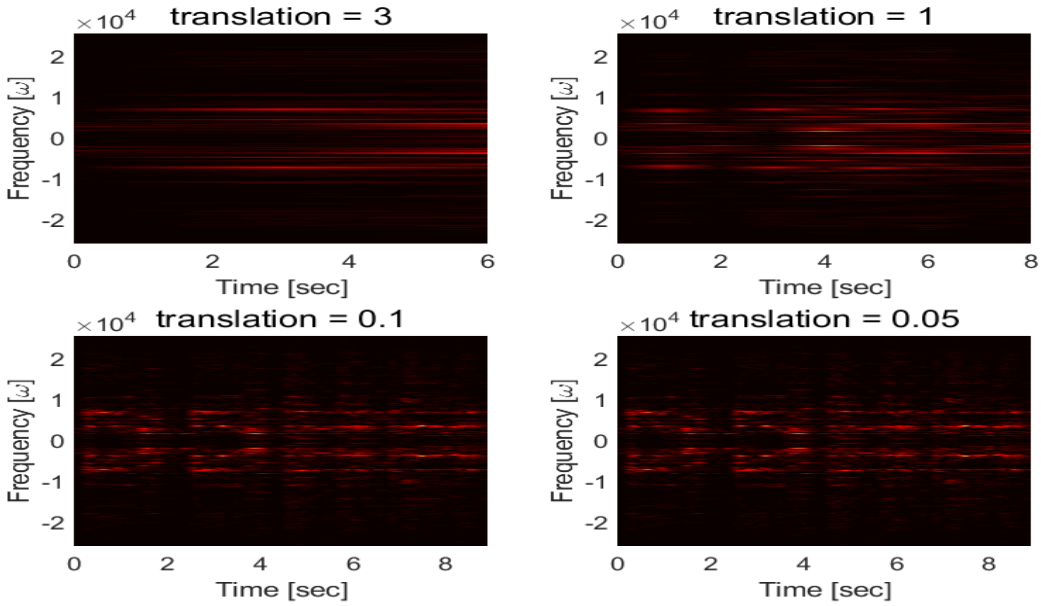

Figure 2: Gaussian Filter Spectrogram with fixed window width a=70

I also try to use the Mexican hat wavelet and Shannon step-function for filtering. It turns out that they are all effective. However, they differ in the choice of width and translation because of different shape. I find out that using same window width on Mexican hat wavelet and Shannon step-function won't give me same level of resolutions on spectrogram. Figure 3 is the collection of spectrograms with different Gabor windows.

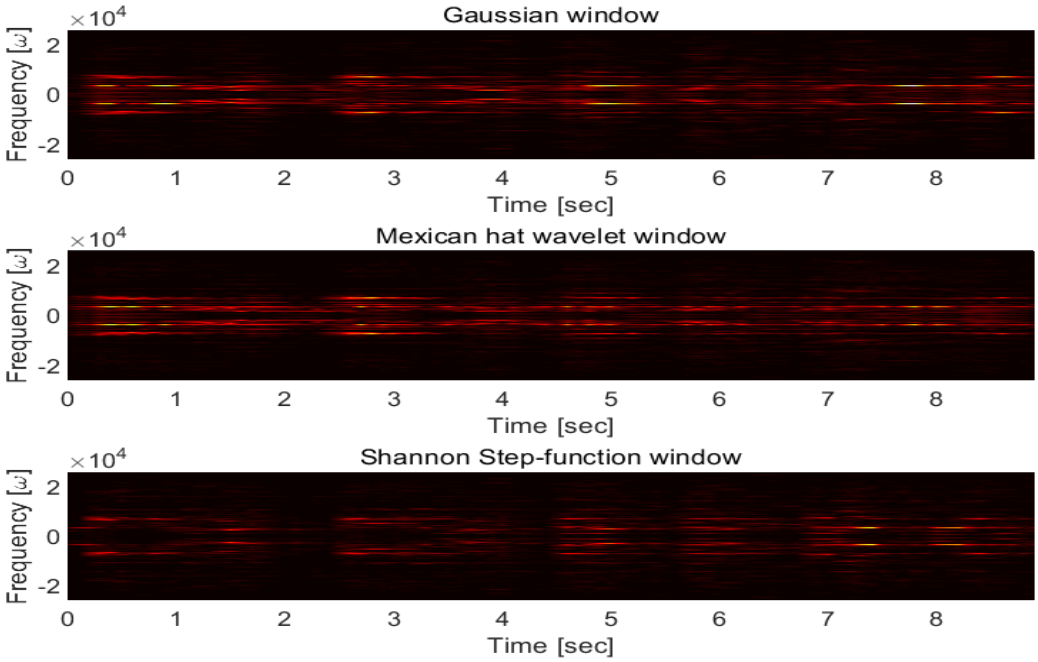

Figure 3: Viable Spectrogram of different Gabor window

Part Two: I apply Gaussian filter with a = 50 translation = 0.1 to filter out the overtones. According to the clean spectrogram (Figure 4) and the picture of music scale provided, we can recognize the notes in Hertz are: 320, 280, 260, 320, 320, 320, 320, 280, 280, 280, 280, 320, 320, 320, 320, 280, 260, 280, 320, 320, 320, 320, 280, 280, 320, 280, 260, 260, 260, 320, 320, 320. We can easily tell that most of the piano notes are in a frequency range of 200 to 400 Hz, floating around 300 in Figure 4. However, the recorder has generally higher frequency around 1000Hz. (See Figure 5)

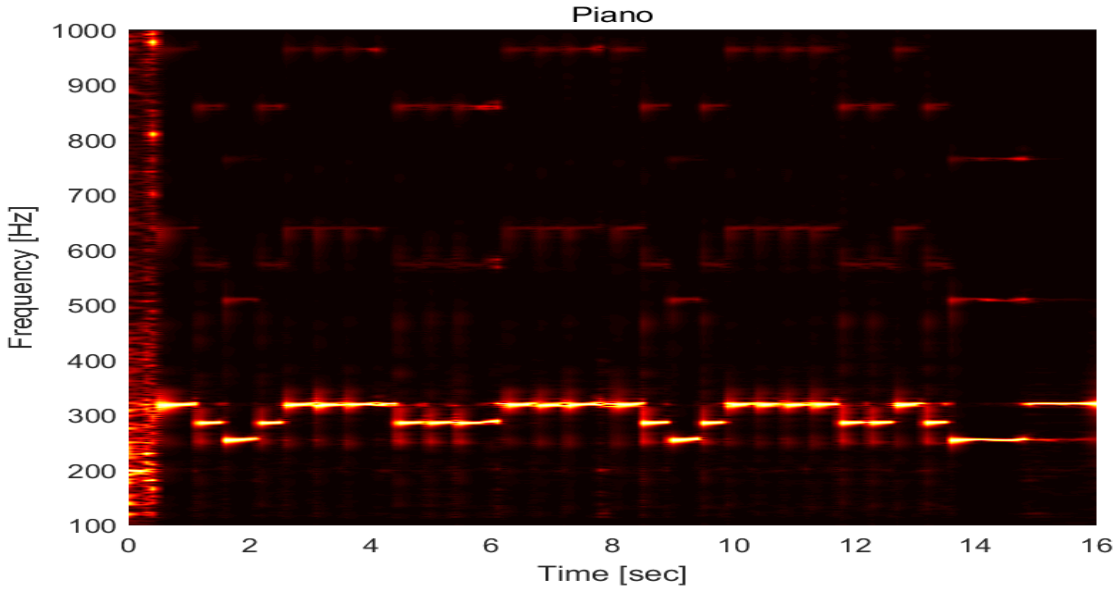

Figure 4: Piano for Marry had a little lamb

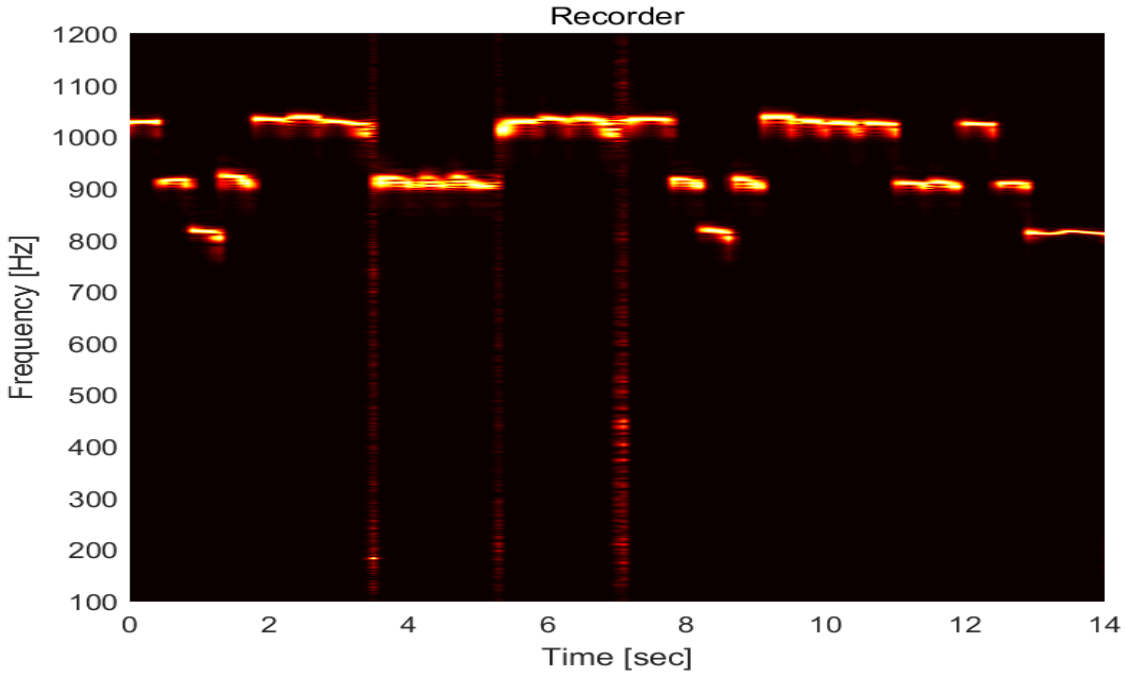

Figure 5: Recorder for Marry had a little lamb

Reading from the spectrogram for recorder, we have notes in Hz: 1030, 910, 810, 910, 1030, 1030, 1030, 1030, 910, 910, 910, 910, 1030, 1030, 1030, 1030, 1030, 910, 810, 910, 1030, 1030, 1030, 1030, 1030, 910, 910, 1030, 910, 810, 810.

Summary and Conclusions

In conclusion, we can use apply Gabor transform with different filters, but we need to choose reasonable translation and window width. Furthermore, different Gabor window will likely have different choice of width and translation. Smart choice is important for avoiding **oversampling** and **undersampling**. With appropriate width and translation, we can filter the overtone and differentiate among different sound. Part two is an application of doing so.

Appendix A

pcolor(C): create a pseudo color plot using the input matrix C. This command gives us the spectrogram in the project.

fft(X): apply FFT on the input element X

audioread(filename): read data from the input file and returned sampled data and sample rate

colormap: set the colormap for the current figure

Appendix B

```
% Homework 1
clear all; close all; clc

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

L=length(v)/Fs;     % music time length
v = v(1:end-1);   % periodic
n=length(v);
t2=linspace(0,L,n+1);
t=t2(1:n);
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
```

```
ks=fftshift(k);

% Gaussian Filter with fixed translation 0.1 and changing window width
figure(1)
a_vec = [60 20 2 0.2];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tslide=0:0.1:L;
    Sgt_spec = zeros(length(tslide),n);
    Sgt_spec = [];
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        Sg=g.*v;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end

    subplot(2,2,jj)
    pcolor(tslide,ks,Sgt_spec.'),
    shading interp
    title(['a = ',num2str(a)],'Fontsize',12)
    xlabel('Time [sec]');
    ylabel('Frequency [\omega]');
    colormap(hot)
end
print(gcf, '-dpng', 'figure 1.png');
% Gaussian filter with fixed width a = 70 and changing translation
figure(2)
translation_vec = [3 1 0.1 0.05];
for jj = 1:length(translation_vec)
    a = 70;
    translation = translation_vec(jj);
    tslide=0:translation:L;
    Sgt_spec = zeros(length(tslide),n);
    Sgt_spec = [];
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        Sg=g.*v;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end

    subplot(2,2,jj)
    pcolor(tslide,ks,Sgt_spec.'),
    shading interp
    title(['translation = ',num2str(translation)],'Fontsize',12)
```

```
    xlabel('Time [sec]');
    ylabel('Frequency [\omega]');
    colormap(hot)
end
print(gcf, '-dpng', 'figure 2.png');

% Compare Gaussian, Mexican hat wavlet and Shannnon step-function window
figure(3)
a = 70;
tslide=0:0.1:L;
% Gaussian window with width a = 70
Sgt_spec = zeros(length(tslide),n);
Sgt_spec = [];
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    Sg=g.*v;
    Sgt=fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end
subplot(3,1,1)
pcolor(tslide,ks,Sgt_spec.'),
shading interp
title(['Gaussian window'])
xlabel('Time [sec]');
ylabel('Frequency [\omega]');
colormap(hot)

% Mexican hat wavelet with width a = 0.1
a = 0.1;
Sgt_spec = zeros(length(tslide),n);
Sgt_spec = [];
for j=1:length(tslide)
    g=2/(sqrt(3*a) * (pi)^(1/4)) * (1-((t-tslide(j))/a).^2).*exp(-(t-tslide(j)).^2 / (2*a^2));
    Sg=g.*v;
    Sgt=fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end
subplot(3,1,2)
pcolor(tslide,ks,Sgt_spec.'),
shading interp
title(['Mexican hat wavelet window'])
xlabel('Time [sec]');
ylabel('Frequency [\omega]');
colormap(hot)

% Shannon Step-function window with width a = 0.1
```

```
Sgt_spec = zeros(length(tslide),n);
Sgt_spec = [];
for j=1:length(tslide)
    g= (abs(t - tslide(j)) < a);
    Sg=g.*v;
    Sgt=fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end
subplot(3,1,3)
pcolor(tslide,ks,Sgt_spec.'),
shading interp
title(['Shannon Step-function window'])
xlabel('Time [sec]');
ylabel('Frequency [\omega]');
colormap(hot)
print(gcf, '-dpng', 'figure 3.png');
%% Part 2
close all; clear all;
% Piano
[y,Fs] = audioread('music1.wav');
tr_piano=length(y)/Fs; % record time in seconds
% plot((1:length(y))/Fs,y);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title('Mary had a little lamb (piano)');
% p8 = audioplayer(y,Fs); playblocking(p8);
y = y'/2;
Lp=16;     % music time length
n=length(y);
t=(1:length(y))/Fs;
k=(2*pi/Lp)*[0:n/2-1 -n/2:-1];
ks_p=fftshift(k);
tslide_p = 0:0.1:Lp;
pgt_spec = [];
score_p = [];
a = 50;
for j=1:length(tslide_p)
    g=exp(-a*(t-tslide_p(j)).^2);
    Sg=g.*y;
    Sgt=fft(Sg);
    pgt_spec = [pgt_spec; abs(fftshift(Sgt))/max(abs(Sgt))];
    [M, I] = max(abs(Sgt));
    score_p = [score_p; abs(k(I))];
end
figure(4)
% subplot(2,1,1)
% plot(tslide_p,score_p/(2*pi));
```

```
% xlabel('Time [sec]');
% ylabel('Frequency(Hz)');
% title('Center Frequency of Piano');
% subplot(2,1,2)
pcolor(tslide_p,(ks_p/(2*pi)),pgt_spec.')
shading interp
title(['Piano'])
xlabel('Time [sec]');
ylabel('Frequency [Hz]');
ylim([100 1000])
colormap(hot)
drawnow
hold on
print(gcf, '-dpng', 'figure 4.png');
%% Recorder
[y,Fs] = audioread('music2.wav');
tr_rec=length(y)/Fs; % record time in seconds
% 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;
Lr=14;     % music time length
n=length(y);
t=(1:length(y))/Fs;
k=(2*pi/Lr)*[0:n/2-1 -n/2:-1];
ks_r=fftshift(k);
tslide_r = 0:0.1:Lr;
rgt_spec = [];
score_r = [];
a = 50;
for j=1:length(tslide_r)
    g=exp(-a*(t-tslide_r(j)).^2);
    Sg=g.*y;
    Sgt=fft(Sg);
    rgt_spec = [rgt_spec; abs(fftshift(Sgt))/max(abs(Sgt))];
    [M, I] = max(abs((Sgt)));
    score_r = [score_r; abs(k(I))];
end
figure(5)
% subplot(2,1,1)
% plot(tslide_r,score_r/(2*pi));
% xlabel('Time [sec]');
% ylabel('Frequency(Hz)');
% title('Center Frequency of Recorder');
% subplot(2,1,2)
```

```
pcolor(tslide_r,ks_r/(2*pi),rgt_spec.')
shading interp
title(['Recorder'])
xlabel('Time [sec]');
ylabel('Frequency [Hz]');
colormap(hot)
ylim([100 1200])
drawnow
hold on
print(gcf, '-dpng', 'figure 5.png');
```