Gabor transforms

Khoa Vo

Abstract: This project will discuss about one of the fundamental techniques in signal analysis, Gabor Transform, and how it is applied to extract the time-frequency information. This project will identify the drawbacks of the Gabor Transform and explore what is tradeoff associated with the Gabor Transform.

1. Introduction and Overview

This project has two parts. In part I, we will analyze and explore 9 seconds of Handels Messiah to examine how the width, the translation parameter, and the choice of wavelets affect the time-frequency resolution. In part II, we will need to perform Gabor Transform to produce the music scores for a portion of *Mary had a little lamb* played on the piano and recorded.

2. Theoretical Background

2.1. Fourier Analysis

Fast Fourier Transform (FFT) is an algorithm that is widely used for applications in engineering, data science, music, and mathematics. FFT algorithm converts the signal in the original domain (time or space) into a representation in frequency domain. FFT algorithms are known to have time complexity of $O(N \log N)$. It finds the transform on the interval $x \in [-L, L]$ and discretizes the range of x into 2^n points. FFT is defined by the formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \tag{1}$$

Its inverse (IFFT) is defined by the formula:

$$x_k = 1/N \sum_{n=0}^{N-1} X_k e^{-i2\pi kn/N} \tag{2}$$

2.2. Gabor Transform

Fast Fourier Transform is an important and foundational method for the analysis of signal. However, Fast Fourier Transform fails to capture the moment in time when various frequencies were actually exhibited.

To obtain the frequency content of the data, without losing all the spatial information, we need to perform the Fast Fourier Transform on a specific window in time. This technique is called Gabor Transform, named after Dennis Gabor, is a special case of the short-time Fourier transform (STFT). The Gabor Transform is defined by the formula:

$$\mathcal{G}[f] = \int_{-\infty}^{\infty} f(\tau) g(\tau - t) e^{-i\omega\tau} d\tau \tag{3}$$

with $g(\tau - t)$ inducing localization of the Fourier integral around $t = \tau$.

Drawbacks of Gabor Transform is that if we transform on a window that is too short comparing to the wavelength, then a large portion of frequency content of the data will be lost. However, if we choose a large window, we will lost more spatial information. Therefore, we need to check back and forth to determine which window size is the best for us to capture the spatial and frequency content.

2.3. Spectrogram

The Gabor Transform can be used to analyze both the time and frequency properties of a given signal. A visual representation of how the frequencies of the signal varies in the spatial domain is called *Spectrogram.*

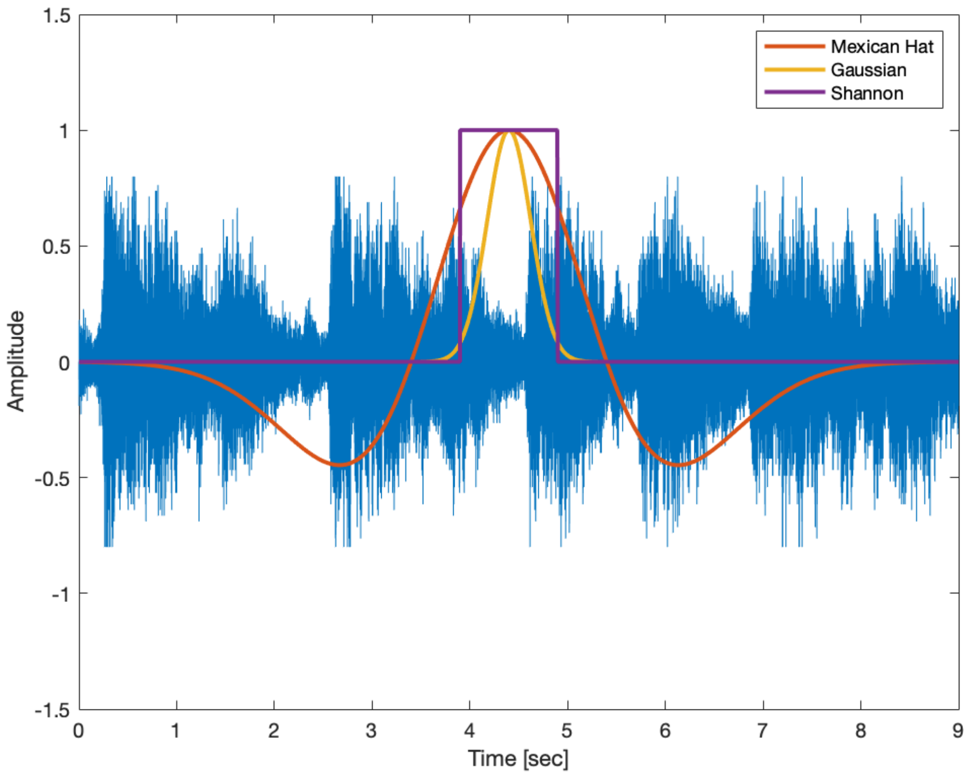


Figure 1: Difference between 3 functions: Gaussian, Mexican Hat, and Shannon.

2.4. Wavelet Analysis

The Gabor Transform limits the high level of time-frequency resolution that can be obtained, as it trades off the accuracy in time (frequency) for the accuracy in frequency (time). A modification of Gabor Transform is to allow the scaling window to vary in order to extract out higher-frequencies and better time resolution. Wavelet analysis begins with a function known as the *mother wavelet*, defined by:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a}) \tag{4}$$

with a is the scaling parameter and b is the translation parameter.

There are other functions that can be used for the Gabor windows:

Gaussian filter:

$$g = e^{-a(t-\tau)^2} \tag{5}$$

Mexican Hat Wavelet:

$$\psi_t = (1-(t-\tau)^2)e^{\frac{-a(t-\tau)^2}{2}} \tag{6}$$

Shannon:

$$g = \begin{cases} 1 & \text{for } t \in [\tau - \frac{d}{2}, \tau + \frac{d}{2}], \text{ with } d \text{ is the width of the window} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The difference between those functions is shown in **Figure 1**.

3. Algorithm Implementation and Development

3.1. Part I: Analyze a portion of Handels Messiah

3.1.1. Exploring how the in window width affects the Spectrogram

First, we need to load the data and get the amplitude v as 1x73113 vector. We want it periodic, so we only need first 73112 columns. Then, since FFT shifts the data so we will need to define k from 0 to $n/2-1$ and $-n/2$ to -1, in which n is the length of vector v and then define ks by shifting k using the command "fftshift". Since command the 'fft' assumes that we are working on the 2π periodic domain. Therefore, we need to rescale 'k' by the factor of $2\pi/L$, with L is 9 because the piece is 9 seconds.

For this part, I will choose the translation parameter to be 0.1 and time *tslice* goes from 0 to 9. I also construct a vector for different width values, including 0.1, 10, 100, and 1000 to explore how the Spectrogram changes as we increase the width of the Gaussian filter. We iterate though width vector, then iterate though the translations of Gabor Transform, construct the Gaussian filter as **Equation (5)** with the width accordingly, filter the original data, and perform the Fast Fourier transform. Since the 'fft' shifts the data, so we need to use 'fftshift' to shift the transformed data back to its mathematically correct position and take the real part of it by using absolute function, 'abs'. Finnaly, we need to scale ks by a factor of $1/(2*pi)$ when we plot the spectrogram.

3.1.2. Exploring the Spectrogram

The last section, we explore the width of 100 gives the Spectrogram with good time-frequency resolution, hence for this section, i will use the width of 100 to explore how change the translation parameter affects the Spectrogram. First, I construct a vector of translation parameters, including 0.01, 0.1, 1, and 5. We iterate through the vector, then iterate though the translations, filter by applying the Gaussian Filter with $a = 100$, and perform the Fast Fourier transform. Since the 'fft' shifts the data, so we need to use 'fftshift' to shift the transformed data back to its mathematically correct position and take the real part of it by using function, 'abs'. Finnaly, we need to scale ks by a factor of $1/(2*pi)$ when we plot the spectrogram.

3.1.3. Exploring the Wavelets

The previous sections have shown that for this problem the width of 100 and the translation parameter of 0.1 gives us good time-frequency resolution, hence I will use those value to explore how different wavelets affect the Spectrogram. We use the same algorithm as previous parts, however, we now will apply 3 different filters (**Equation (5)**, **Equation (6)**, and **Equation (7)**) to plot the Spectrogram.

3.2. Exploring Piano vs. Recorder

We start by loading and converting both pieces of music into vectors, construct the translation parameter = 0.25 from 0 to the length of each music piece. Applying the Gausian Filter, perform the same algorithm as Part I and then find the the highest frequency in that window. I also scale the original amplitude by a factor of 1000 and 4000 respectively for piano and recorder. It is easier to determine what is the frequency in a moment.

4. Computational Results

4.1. Exploring how the change in window width affects the Spectrogram

Figure 2 shows that the Spectrogram for window width of 0.1 gives lots of frequency information, but we cannot see when those frequencies happen in the time domain. As we increase the width, the amount of frequency information decrease and obtain more time information.

4.2. Exploring Oversampling, Normal Sampling, and Undersampling

Figure 3 shows that with translation of 0.01, we traded off the frequency information for the time information, this is called oversampling. Otherwise, with we translation of 5, we traded off the time information for the frequency information.

4.3. Gaussian, Mexican Hat Wavelet, and Shannon

Figure 4 compares how applying 3 different filters results in 3 different Spectrograms.

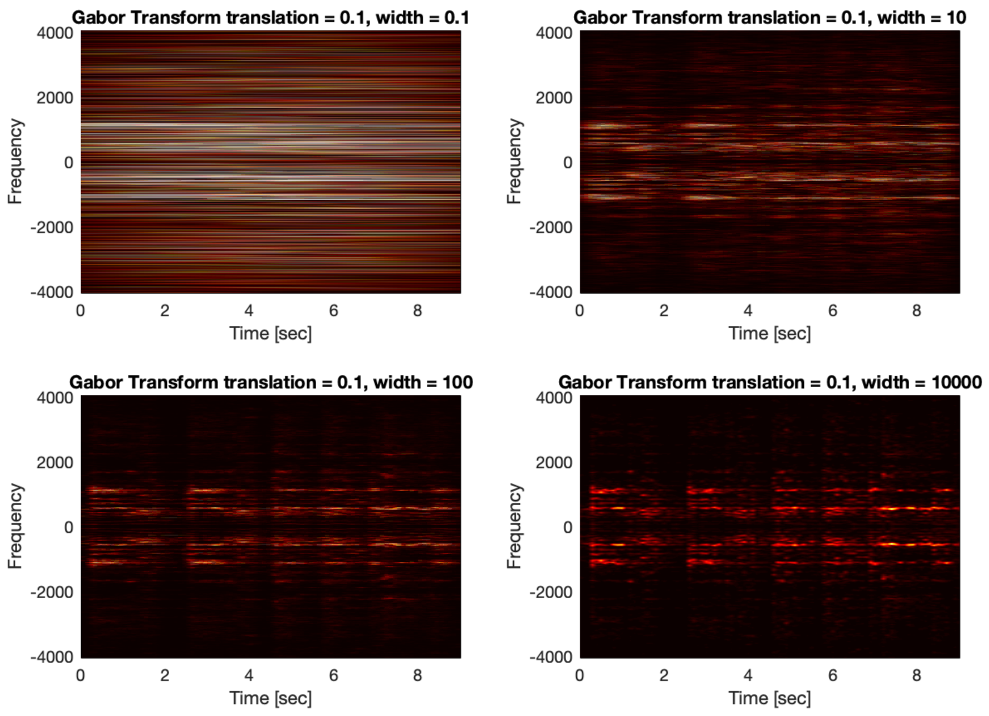


Figure 2: The Spectrogram for 4 different window widths: 0.1, 10, 100 and 1000

4.4. Piano vs. Recorder

Using **Figure 5.a**, I estimate the frequencies for piano:

321 288 256 288 319 320 320 284 284 284 320 320 320 319 288 258 288 321 320 323 319 287 287 320 287 254

which is according to:

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

Using **Figure 5.b**, I estimate the frequencies for recorder:

1046 901 804 908 1005 1006 1007 900 900 900 1052 1052 1052 1052 900 806 894 1036 1036 1036 1036 849 911 994 845 794

which is according to:

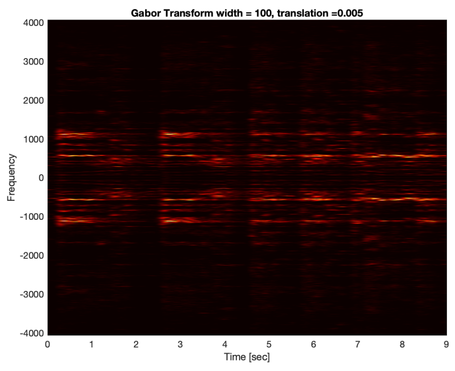
C A G A B B B A A A B B B B A G A B B B B A A B A G

5. Summary and Conclusions

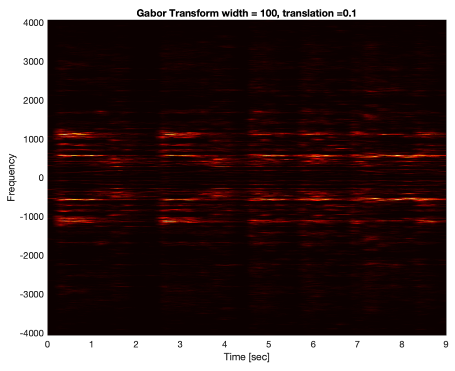
Gabor transform is the fundamental technique to analyze the signal. Choosing a good window size and translating factor to perform Gabor Transform can give a nice time-frequency resolution, otherwise we will trade off the time (frequency) for frequency (time). After filtering, the recorder has higher frequencies than the piano has, roughly about 3 times higher.

Appendix A.

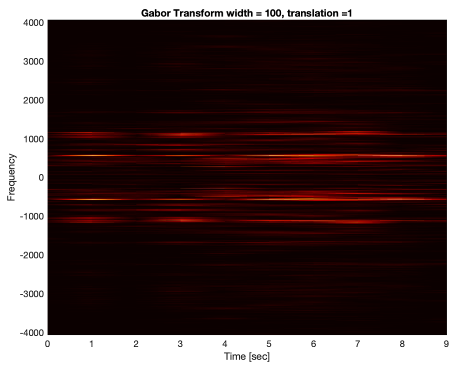
fft() : Fast Fourier Transform
fftshift(): Rearrange a Fourier Transform by shifting the zero-frequency component to the center
pcolor(): Plot a heat map in MatLab



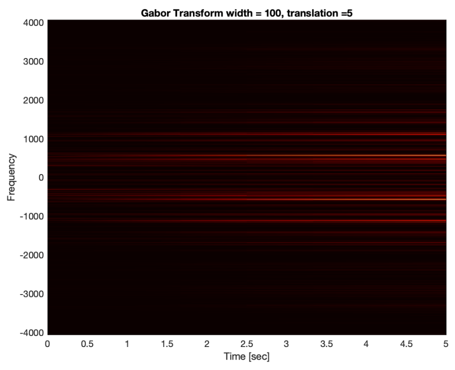
(a) Oversampling, translation parameter = 0.005



(b) Translation parameter = 0.1

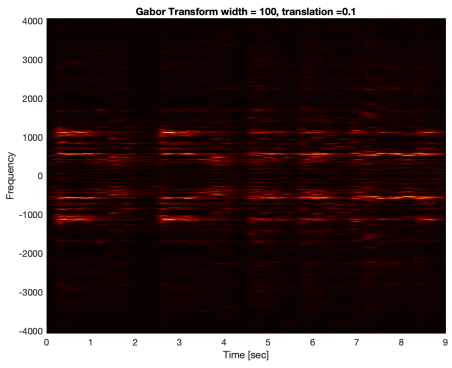


(c) Translation parameter = 1

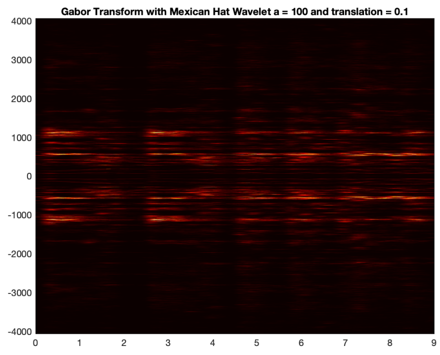


(d) Undersampling, translation parameter = 5

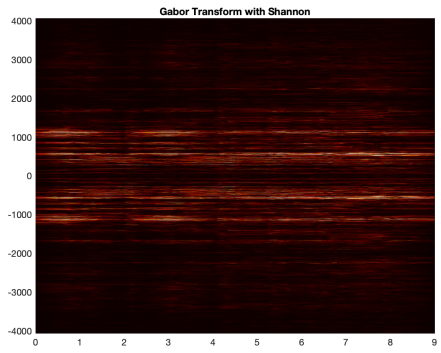
Figure 3: The Spectrogram for 4 different translation parameters: 0.01, 0.1, 1, 5



(a) Gaussian

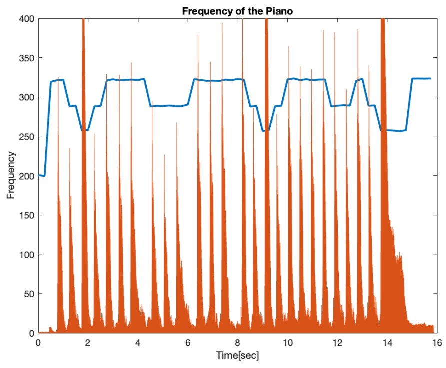


(b) Mexican Hat Wavelet

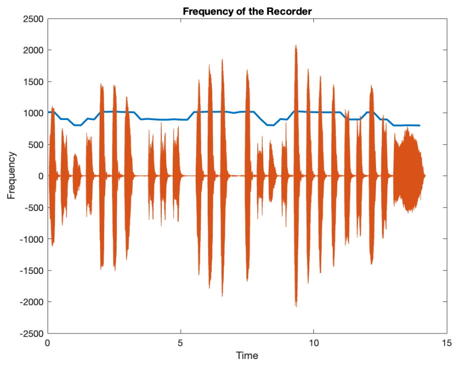


(c) Shannon

Figure 4: Difference of Gaussian, Mexican Hat Wavelet, and Shannon



(a) Frequency of Piano



(b) Frequency of Recorder

Figure 5: Frequency of Piano and Recorder

Appendix B.

Part I

```
clear all; close all; clc
  load handel
  v = y';
  v1 = v;
  v = v(1:length(v) - 1);
  %p8 = audioplayer(v,Fs);playblocking(p8);
  L=9; 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);
  tslide=0:0.1:9;
  %% comparing 3 windows
  j = 45
  tslide = 0:0.1:9;
  gaussian
  g=exp(-10*(t-tslide(j)).^2);
  mexican hat
  mxh = (1 - (t-tslide(j)).^2).*exp(-(t-tslide(j)).^2/2);
  shannon filter
  s = zeros(1, length(t));
  s = zeros(1, length(t));
  index_of_tslidej = round(tslide(45)/9*length(t)); % index of tslide
  index_width = round(0.5/9*length(t));
  if index_of_tslidej-index_width <= 0
      s(1:index_of_tslidej+index_width) = ones();
  elseif index_of_tslidej+index_width > length(t);
      s(index_of_tslidej-index_width:length(t)) = ones();
  else
      s(index_of_tslidej-index_width:index_of_tslidej+index_width) = ones();
  end;
  plot(t, v); hold on
  a1 = plot(t, mxh, 'linewidth',[2]); L1 = 'Mexican Hat';
  a2 = plot(t, g, 'linewidth',[2]); L2 = 'Gaussian';
  a3 = plot(t, s, 'linewidth',[2]); L3 = 'Shannon';
  ylim([-1.5 1.5])
  ylabel('Amplitude')
  xlabel('Time [sec]')
  legend([a1;a2;a3], L1, L2, L3)

  %% exploring changing width
  a_vector = [0.1 10 100 10000]; %different width
  tslide=0:0.1:9;
  Sgt_spec = zeros(length(tslide),n);
  for i=1:length(a_vector)
     a = a_vector(i);
     for j=1:length(tslide)
         g=exp(-a.*(t-tslide(j)).^2);
         vfilter = g.*v;
         vfiltert= fft(vfilter);
         Sgt_spec(j,:) = fftshift(abs(vfiltert));
     end

     subplot(2, 2, i)
     pcolor(tslide,ks/(2*pi),Sgt_spec.'),
     xlabel('Time [sec]')
     ylabel('Frequency')
     shading interp
     colormap(hot)
```

```
   title(['Gabor Transform translation = 0.1, width = ', num2str(a)])
   saveas(gca, ['gabor_trans.png'])
end

%% exploring change translation
t_step = [0.005]; % large step mean undersampling, small step mean oversampling
for i=1:length(t_step)
    a = 100;
    step = t_step(i);
    tslide=0:step:9;
    Sgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        g=exp(-a.*(t-tslide(j)).^2);
        vfilter = g.*v;
        vfiltert= fft(vfilter);
        Sgt_spec(j,:) = fftshift(abs(vfiltert)); % We don't want to scale it
    end
    pcolor(tslide,ks/(2*pi),Sgt_spec.')
    xlabel('Time [sec]')
    ylabel('Frequency')
    shading interp
    colormap(hot)
    title(['Gabor Transform width = 100, translation =',num2str(step)])
    saveas(gcf, ['translation=',num2str(step),'.png'])
end

%%mexican hat
a = 100;
tslide=0:0.1:9;
for j=1:length(tslide)
    filter = (1 - (t-tslide(j)).^2).*exp(-a*(t-tslide(j)).^2/2);
    vfilter = v.*filter;
    vfiltert= fft(vfilter);
    Sgt_spec(j,:) = fftshift(abs(vfiltert));
end
pcolor(tslide,ks/(2*pi),Sgt_spec.'),
shading interp
colormap(hot)
title(['Gabor Transform with Mexican Hat Wavelet a = 100 and translation = 0.1'])
saveas(gcf, ['mexican_hat.png'])

%% shannon filter
Sgt_spec = zeros(length(tslide),n);
for j=1:length(tslide)
    s = zeros(1, length(t));
    index_of_tslidej = round(tslide(j)/9*length(t)); % index of tslide
    index_width = round(0.5/9*length(t));
    if index_of_tslidej-index_width <= 0 && index_of_tslidej+index_width > length(t)
        s(1:length(t)) = ones();
    elseif index_of_tslidej-index_width <= 0
        s(1:index_of_tslidej+index_width) = ones();
    elseif index_of_tslidej+index_width > length(t)
        s(index_of_tslidej-index_width:length(t)) = ones();
    else
        s(index_of_tslidej-index_width:index_of_tslidej+index_width) = ones();
    end
    filter = abs(t-tslide(j)) < 1/2;
    vfilter = v.*s;
    vfiltert= fft(vfilter);
    Sgt_spec(j,:) = fftshift(abs(vfiltert));
end
pcolor(tslide,ks/(2*pi),Sgt_spec.'),
    shading interp
    colormap(hot)
    title(['Gabor Transform with Shannon'])
    saveas(gca, 'shannon.png')
```

Part II

```
  clear all; close all; clc

    %piano-------------------------------------------------------------
    [y,Fs] = audioread('music1.wav');
    tr_piano=length(y)/Fs; % record time in seconds
    n = length(y);
    t2=linspace(0,tr_piano,n+1); t=t2(2:n+1);
    k=(2*pi/tr_piano)*[0:n/2-1 -n/2:-1];
    ks=fftshift(k);
    v1 = y';
    tslide = 0:0.25:tr_piano;

    Sgt_spec = zeros(length(tslide),n);
    music = [];
    for j=1:length(tslide)
       g=exp(-40*(t-tslide(j)).^2);
       v1filter = g.*v1;
       v1filtert= fft(v1filter);
       Sgt_spec(j,:) = abs(fftshift(v1filtert));

       index = find(v1filtert == max(v1filtert(:)));
       music = [music, abs(k(index))/(2*pi)];
    end
    pcolor(tslide,ks/(2*pi),abs(Sgt_spec).'),
    shading interp
    colormap(hot)
    ylim([0 500])

    title(['Spectrogram of Piano sound'])
    plot(tslide, music, 'Linewidth', 2), hold on

    plot((1:length(y))/Fs,y*1000)
    ylim([0 400])
    xlabel('Time[sec]')
    ylabel('Frequency')
    title('Frequency of the Piano')

    %recording----------------------------------------------------------------------------

    [y,Fs] = audioread('music2.wav');
    tr_rec=length(y)/Fs; % record time in seconds
    n = length(y);
    t2=linspace(0,tr_rec,n+1); t=t2(2:n+1);
    k=(2*pi/tr_rec)*[0:n/2-1 -n/2:-1];
    ks=fftshift(k);
    v2 = y';
    tslide=0:0.25:tr_rec;
    Sgt_spec = zeros(length(tslide),n);
    music = [];
    for j=1:length(tslide)
       g=exp(-40.*(t-tslide(j)).^2);
       v2filter = v2.*g;
       v2filtert= fft(v2filter);
       Sgt_spec(j,:) = fftshift(abs(v2filtert));
```