Computational Methods for Data Analysis

Homework 2

Due on February 7, 2020 at 11:00pm

Professor J. Nathan Kutz

Section A

Chandan Sharma Subedi

Abstract

Frequency analysis of a signal through Fourier Transform is a very useful tool to analyze stationary signal. However, when the frequency signature of a signal changes over time, frequency analysis does not give adequate information to analyze the signal. The paper discusses a novel time-frequency analysis based on Gabor Transform that extends the idea of Fourier Tranform to produce a spectrogram of a signal. The spectrogram is later used to identify notes played on the piano and recorder recordings of the song *"Mary had a little lamb."*.

1 Introduction and Overview

Fourier Transform was an influential idea that revolutionlized radar technology in 1940s. A Fourier Transform of a signal gives the frequency content of the signal. However, when frequencies in the signal change with time, Fourier Transform becomes less useful. It does not tell us at what point each frequency is active. The figure 1 shows that even though Fourier Transform gives all frequencies in the signal, it is not clear when 10 Hz frequency is active.

Gabor in 1946 came up with an idea of using a sliding window that would allow us to assume the signal under the window as stationary. Through sliding window, the time information of the frequency spectrum is captured that gives us a 2D data with good time and frequency resolution (as shown in figure 2). Section 2 gives a detailed theoretical background on Gabor Transform and sliding windows. Section 3 and 4 discuss the implementation of Gabor Transform to analyze music recording and results of the analysis.

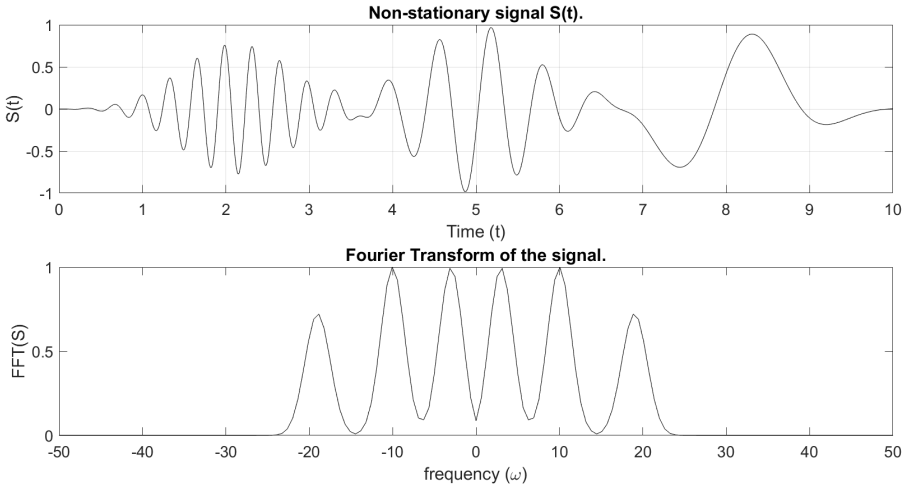
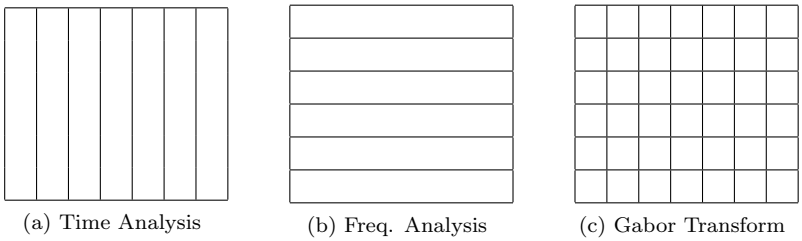


Figure 1: The figure shows a non-stationary signal and its fast fourier transform.



(a) Time Analysis (b) Freq. Analysis (c) Gabor Transform

Figure 2: Time analysis gives a perfect resolution on when each frequencies are active, but no information about what those frequencies are. Frequency analysis gives perfect resolution on what all the frequencies are in the signal, but no information on when those frequencies are active. Time-Frequency analyis through Gabor Transform gives a good resolution (not perfect) on both time and frequency.

2 Theoretical Background

The discrete Gabor Tranform discretizes the signal through sliding window. Each window captures a piece of the signal, which is then fourier tranformed to gives its frequency spectrum. As we slide the window through the signal, we obtain a varying frequency spectrum. Combining these spectrums, we obtain a spectrogram that gives us a visual representation of how frequency spectrum varies over time in the signal.

Fourier Tranform of any periodic function in the interval $t \in (-\pi, \pi]$ is a complex valued function of frequency whose modulus is the amount of that frequency and argument is the phase of the sinusoid. Fourier transform is defined as

$$F(k) = \int_{-\infty}^{\infty} f(t) e^{-ikt} dt$$

The Gabor Transform of the function $f(t)$ is defined as

$$G_f(\tau, k) = \int_{-\infty}^{\infty} f(t) \overline{g}_a(\tau - t) e^{-ikt} dt$$

$g_a(\tau - t)$ is the sliding window obtained through the parameterization of function $g(t)$ over a scaling factor a and a shifting factor τ. The scaling factor a determines the width of the window. In this assingment, following three sliding windows were examined:

$$\textit{Gaussian:} \qquad g_a(\tau - t) = e^{-a(\tau - t)^2}$$

$$\textit{Mexican Hat:} \qquad g_a(\tau - t) = (1 - a(\tau - t)^2) e^{-\left(\frac{a(\tau - t)^2}{2}\right)}$$

$$\textit{Rectangle:} \qquad g_a(\tau - t) = \begin{cases} 1 & \text{if } (\tau + a/2) > t > (\tau - a/2) \\ 1/2 & \text{if } t = (\tau + a/2),\ t = (\tau - a/2) \\ 0 & \text{otherwise.} \end{cases}$$

All these functions have one similarity that allows them to be used as a sliding window. When we mulitply the signal with these functions, the signal that lies inside the window remains more or less unchanged, while the signal that lies outside the window approaches zero. Thus, these functions act as a window that isolates a part of a signal at a particular instance of time.

A spectrogram is generated by combining frequency spectrums obtained through the fourier tranform of windowed signals. Thus, it gives a visual representation of how frequency spectrum is changing over time. The resolution of spectrogram, obtained through Gabor Transform, depends strongly on the sliding window and its scaling and sliding parameters. This stems from the fact that the fundamental idea of Gabor Transform is to make some compromise on spectral resolution so as to gain some temporal resolution.

- Effect of scaling parameter on resolution of spectrogram:
  If we choose a scaling parameter that gives a very narrow sliding window, then we will obtain a good temporal resolution but a poor spectral resolution. If a is the width of the narrow sliding window, the smallest frequency it can identify is $1/a$. Thus, we will lose all low frequencies in the spectrum. On the other hand, if we choose a wide sliding window, then we will obtain a good spectral resolution but a poor temporal resolution. Thus, a scaling factor should be chosen that gives a good resolution for both time and frequency.

- Effect of sliding parameter on resolution of spectrogram:
  Undersampling: If we slide window at time steps greater than half of the window width, then we are bound to completely miss a part of the signal. This will reduce both temporal and spectral resolution of the spectrogram.
  Oversampling: If we slide window at very small time steps, it will slightly improve the temporal resolution but at a massive computational cost.

3 Algorithm Implementation and Development

3.1 Part I

In Part I of the assignment, a portion of the Handel's Messiah was analyzed using Gabor Transform as the method of time-frequency analysis. Three different sliding window functions were used with varying scaling and sliding parameters. A MATLAB script was written to perform the following four computational procedures.

- **Generate a vector representing the data:**
  The recording of the Handel's Messiah was tranformed into a column vector. The corresponding time vector was generated using the samping rate Fs.
- **Construct the sliding window function:**
  A sliding window was constructed by choosing a function (Gaussian, Mexican Hat or Rectangle) and parameterizing it with scaling parameter a and sliding parameter τ. A sliding time-step Δt was chosen to find the sliding parameter at each step.
- **Fast Fourier Transform of windowed signal:**
  At each step of the slide, windowed signal was Fast Fourier transformed to generate the frequency spectrum. The spectrum was saved in an array for plotting.
- **Generate Spectrogram:**
  All frequency spectrums were plotted against time using *pcolor* MATLAB function.

3.2 Part II

In Part II, we use Gabor filtering to reproduce the music score of the song *"Mary had a little lamb"* recorded in piano and recorder. We use same steps as in Part I to first generate spectrogram and then use music sheet to find the score.

- **Generate vectors representing the piano and recorder data:**
- **Construct the Gaussian sliding window function:**
- **Fast Fourier Transform of windowed signal:**
- **Generate Logarithmic spectrograms:**
- **Reproduce the music score in piano recording:**
  The logarithmic spectrogram was zoomed near center freqeuncies and using the music sheet given in the assignment, the music score was reproduced.

4 Computational Results

4.1 Part I

The audio considered for the analysis was a 8.9249 sec long audio recording of the Handel's Massiah, sampled at 8192 Hz. The audio was loaded from the MATLAB and a vector was created representing the audio. Using the sampling rate, the corresponding time vector was also calculated. A Gaussian sliding window was constructed with width (FWHM) ≈ 0.17sec. The smallest frequency that a human ear can hear is 20Hz. The wavelength of that signal will be $1/20 = 0.05$ sec. The window width guarantees that all frequencies audible to human ear will be detected during Gabor Transform. The figure 3 shows the process of filtering/windowing signal at each sliding step by the Gaussian window and finding its corresponding frequency spectrum through FFT. The spectrogram was generated by plotting all frequency spectrums against time using MATLAB *pcolor* function.

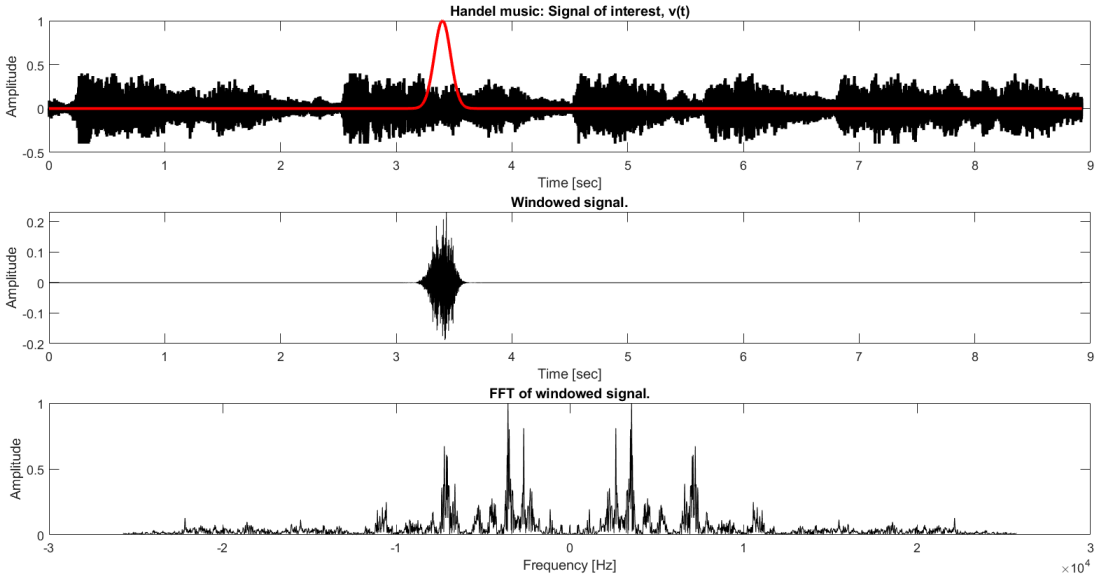


Figure 3: The signal of interest is plotted along with the Gaussian sliding window (red). The corresponding filtered signal and its frequency spectrum is also plotted.

The effect of scaling was explore by generating spectrogram for four different values of a that changes the width of the window. The width of the Gaussian function was defined in terms of full width at half maximum (FWHM), which is the distance between half of maximum value. FWHM for our Gaussian function was derived as follow

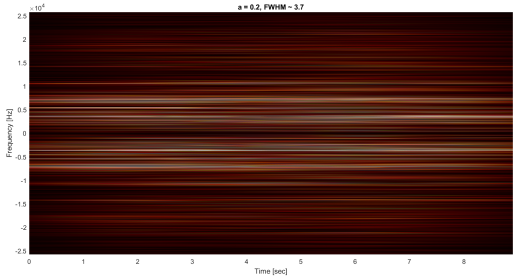
$$\frac{1}{2} = e^{-a(\tau - t^*)^2} \implies (\tau - t^*) = \sqrt{\frac{\ln 2}{a}} \implies \Delta = 2\sqrt{\frac{\ln 2}{a}}$$

The figure 4 shows how spectrogram resolution changes when the width of sliding window is changed. For large width, we saw a very good frequency resolution but a poor temporal resolution. However as window width was decreased, we saw improvement in temporal resolution but at the expense of spectral resolution. This makes sense, since smaller window will fail to capture low frequency signals, while increasing the time data. The figure 5 shows zoomed spectrograms produced by different window function with approximately the same width (≈ 0.2). We can see that Mexican Hat function captures more frequency content and with less temporal resolution (observe right most part of the spectrogram) because it does not approach 0 as fast as the other two functions. However Mexican Hat function trims more data inside its window because of its sharp peak. Spectrogram with Shannon function is close to one with Gaussian function. Because of sharp edges of Shannon function, we see discreteness in its spectrogram.

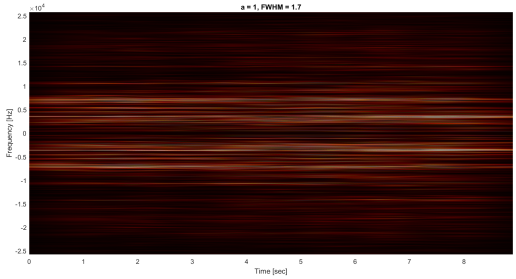
The figure 4 shows the effect of undersampling and oversampling. For large τ, the signal was undersampled as a result of which both frequency and temporal resolution deteriorated. Infact, nearly half of the signal was not used during course sampling. On the other hand, for very small τ, we get great temporal and spectral resolution but at the expense of computation time. While the resolution did not change significantly when τ was changed from 0.05 to 0.01, the computation time increased by 6 folds.

4.2 Part II

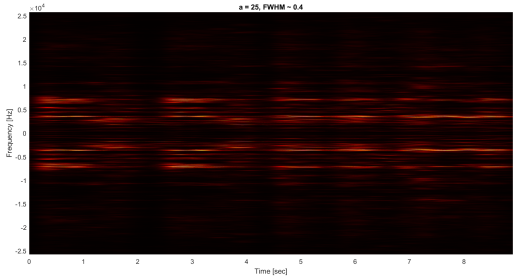
In this section, first the piano and recorder recordings of the song were tranformed into vectors (as shown in figure 7. Using sampling rates for each recordings, corresponding time vectors were calculated. A Gaussian sliding window was constructed with width (FWHM) ≈ 0.2 sec. The sliding step was chosen to be $\Delta\tau = 0.05$



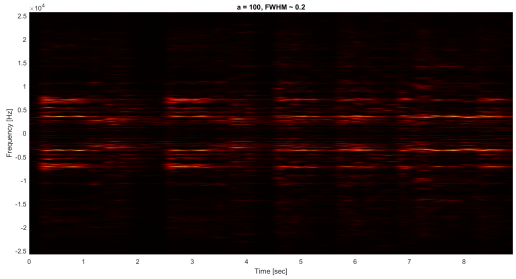
(a) Spectrogram using Gaussian filter of FWHM ≈ 3.7



(b) Spectrogram using Gaussian filter of FWHM ≈ 1.7

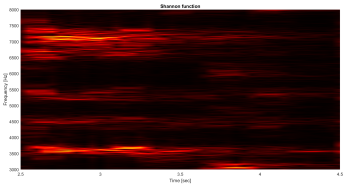


(c) Spectrogram using Gaussian filter of FWHM ≈ 0.4

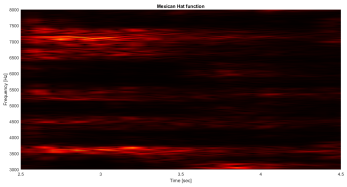


(d) Spectrogram using Gaussian filter of FWHM ≈ 0.2

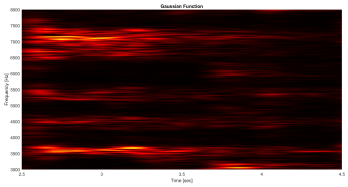
Figure 4: Four spectrograms show the effect of scaling parameter a on the spectral and temporal resolution.



(a) Shannon function.

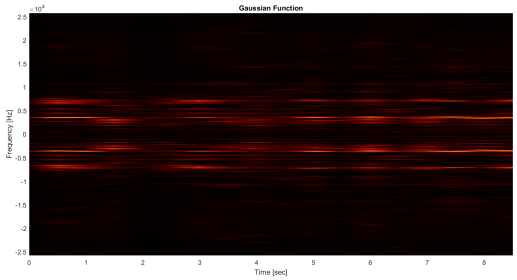


(b) Mexican Hat function.

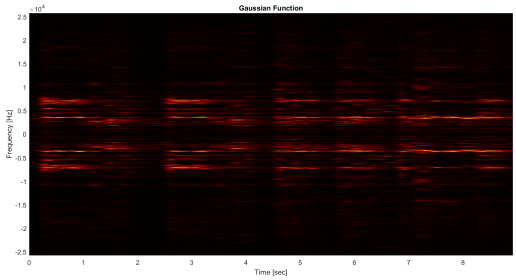


(c) Gaussian function.

Figure 5: Three zoomed spectrograms shows the effect of sliding window function on the spectral and temporal resolution (for same window width).



(a) Undersampling with $\Delta\tau = 0.5$.

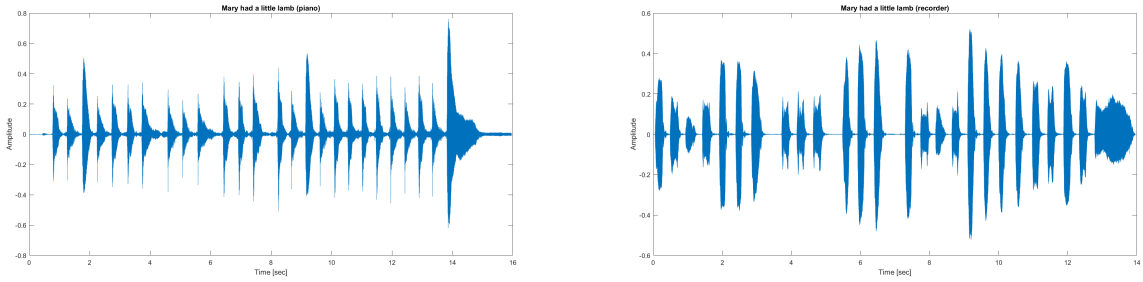


(b) Oversampling with $\Delta\tau = 0.01$.

Figure 6: Spectrograms show the effect of sliding parameter τ on the spectral and temporal resolution using Gaussian window function of FWHM ≈ 0.2.

sec. A logarithmic spectrogram was generated to find principal frequencies that has highest magnitudes. Bright spots in zoomed logarithmic spectrograms correspond to principal frequencies while less bright spots indicate overtones. Using the music chart and the zoomed spectrogram of piano recording, the music score was found to be: **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**.

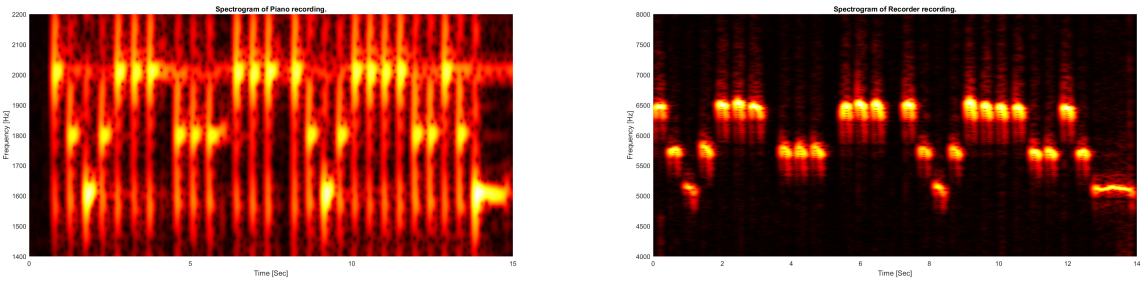
By comparing spectrograms in the figure 8, we can see that piano generates excessive overtones than recorder. These overtones, which are interger multiples of principal frequencies, gives the piano music a rich sound (often known as timbre of the instrument).



(a) Piano recording.

(b) Recorder recording.

Figure 7: Figure shows piano and recorder recordings of the song *"Mary had a little lamb"*.



(a) Zoomed logarithmic spectrogram of piano.

(b) Zoomed logarithmic spectrogram of recorder.

Figure 8: Figure shows zoomed logarithmic spectrograms of piano and recorder recordings using Gaussian sliding window.

5 Summary and Conclusions

In this paper, Gabor transform was analyzed using music recordings. In the first part, using Handel's Messiah recording, the effect of changing sliding and scaling parameters on the temporal and spectral resolution of spectrogram was studied. It was concluded that Gabor transform provides temporal resolution but at the expense of spectral resolution. The sliding window and its parameters should be chosen so that both temporal and spectral resolutions are maximized. For both data, it was found that the Gaussian sliding window of width 0.2 with sliding time step of 0.05 gave good resolutions. Spectrograms for 3 different sliding window function (Gaussian, Mexican Hat and Shannon) was also analyzed. In the second part, music score of a piano recording was reproduced using Gabor tranform. By comparing spectrograms of recorder and piano recordings, it was found that the richness of sound in piano comes from overtones. Using the music sheet for piano, the music score was reproduced.

Appendix A MATLAB Functions

Some important MATLAB functions used during the implementation.

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `ks = fftshift(k)` rearranges FT by shifting zero frequency component to the center of the array. Exchanges two halves of the array.
- `Y = fftn(X)` returns multidimensional Fourier transform of an N-D array using FFT algorithm
- `pcolor(X,Y,C)` specifies the x- and y-coordinates for the vertices. The size of C must match the size of the x-y coordinate grid. For example, if X and Y define an m-by-n grid, then C must be an m-by-n matrix.
- `colormap(map)` sets the colormap for the current figure to the colormap specified by map.
- `shading interp` varies the color in each line segment and face by interpolating the colormap index or true color value across the line or face.

Appendix B MATLAB Code

The code is published on the github repository AMATH-582 under Gabor Transform directory.

MATLAB code for Part I of the assignment.

```
%% Clear workspace
close all; clear all; clc
%% Load the Handel's Messiah music
load handel
v = y'/2;
t = (1:length(v))/Fs;
L = t(end);
N = length(t);
figure(1);
plot(t, v);
xlabel("Time [sec]");
ylabel("Amplitude");
title("Handel music: Signal of interest, v(t)")

%% Shannon Window (Step-function)
figure(2);
widths = [5 2 1 0.1];
rec=@(x,a,b) ones(1,numel(x)).*(t<(a+b/2) & t>(a-b/2));
for j = 1:length(widths)
    shannon = rec(t, 4, widths(j));
    subplot(4, 1, j)
    plot(t,v,'k'), hold on
    plot(t,shannon, 'r', 'Linewidth', [2])
    set(gca, 'Fontsize', [4])
    ylabel('v(t), g(t)')
end
xlabel('time (t)')

%% Gaussian Window
figure(3);
widths = [0.2 1 5 50];
for j = 1:length(widths)
    gauss = exp(-widths(j)*(t-4).^2);
    subplot(4, 1, j)
    plot(t,gauss, 'r', 'Linewidth', [2])
    set(gca, 'Fontsize', [4])
    ylabel('v(t), g(t)')
end
xlabel('time (t)')

%% Mexican Hat Window
figure(4);
widths = [0.2 1 25 100];
for j = 1:length(widths)
    hat = (1-widths(j)*(t-4).^2).*exp(-(widths(j)*(t-4).^2)/2);
    subplot(4, 1, j)
    plot(t,hat, 'r', 'Linewidth', [2])
    set(gca, 'Fontsize', [4])
    ylabel('v(t), g(t)')
```

```
end
xlabel('time (t)')
%% Gabor Transform using Gaussian window
figure(5);
spectrogram = [];
tslide = [0:0.01:L];
k = (2*pi)/(L)*[0:N/2 -N/2:-1];      % Odd N
ks = fftshift(k);
for j = 1:length(tslide)
    gauss = exp(-100*(t-tslide(j)).^2);      % width~0.25sec
    vf = gauss.*v;
    vft = fft(vf);
    spectrogram = [spectrogram; abs(fftshift(vft))];
    subplot(3,1,1);
    plot(t,v,'k',t,gauss,'r',"Linewidth",[2]);
    xlabel("Time [sec]"); ylabel("Amplitude")
    subplot(3,1,2), plot(t,vf,'k')
    xlabel("Time [sec]"); ylabel("Amplitude")
    subplot(3,1,3), plot(ks, abs(fftshift(vft))/max(abs(vft)), 'k')
    xlabel("Frequency [Hz]"); ylabel("Amplitude")
    drawnow
    pause(0.1)
end
%% Gabor Transform using Mexican hat function
figure(6);
spectrogram = [];
tslide = [0:0.05:L];
k = (2*pi)/(L)*[0:N/2 -N/2:-1];      % Odd N
ks = fftshift(k);
for j = 1:length(tslide)
    hat = (1-100*(t-tslide(j)).^2).*exp(-(100*(t-tslide(j)).^2)/2);
    vf = hat.*v;
    vft = fft(vf);
    spectrogram = [spectrogram; abs(fftshift(vft))];
    subplot(3,1,1), plot(t,v,'k',t,hat,'r',"Linewidth",[2])
    subplot(3,1,2), plot(t,vf,'k')
    subplot(3,1,3), plot(ks, abs(fftshift(vft))/max(abs(vft)), 'k')
    drawnow
    pause(0.1)
end

%% Gabor Transform using Shannon function (rectangular function)
figure(7);
spectrogram = [];
tslide = [0:0.05:L];
k = (2*pi)/(L)*[0:N/2 -N/2:-1];       % Odd N
ks = fftshift(k);
for j = 1:length(tslide)
    shannon = rec(t, tslide(j), 0.2);
    vf = shannon.*v;
    vft = fft(vf);
    spectrogram = [spectrogram; abs(fftshift(vft))];
    subplot(3,1,1), plot(t,v,'k',t,shannon,'r',"Linewidth",[2])
    subplot(3,1,2), plot(t,vf,'k')
```

```
    subplot(3,1,3), plot(ks, abs(fftshift(vft))/max(abs(vft)), 'k')
    drawnow
    pause(0.1)
end
%% Spectrogram
figure(8);
pcolor(tslide, ks, spectrogram.')
shading interp
colormap(hot)
% axis([2.5 4.5 3000 8000])
xlabel("Time [sec]")
ylabel("Frequency [Hz]")
% title("a = 0.2, FWHM ~ 3.7")
% title("a = 1, FWHM ~ 1.7")
% title("a = 25, FWHM ~ 0.4")
% title("a = 100, FWHM ~ 0.2")


% title("Shannon function")
% title("Mexican Hat function")
title("Gaussian Function")

%% Gabor Transform Plots for paper.
clear all; close all; clc

L=10; n=2048;
t2=linspace(0,L,n+1); t=t2(1:n);
k=(2*pi/L)*[0:n/2-1 -n/2:-1]; ks=fftshift(k);
S = sin(10*t).*exp(-(t-5).^2) + ...
    cos(19*t).*tanh(0.5*t).*exp(-(t-2).^2) + ...
    cos(3*t).*tanh(0.5*t).*exp(-(t-8).^2);
St=fft(S);

figure(9); % Time domain
subplot(2,1,1)
plot(t,S,"k")
set(gca,"Fontsize",[14]),
xlabel("Time (t)"), ylabel("S(t)")
title("Non-stationary signal S(t).")
grid on
subplot(2,1,2) % Fourier domain
plot(ks,abs(fftshift(St))/max(abs(St)),"k");
axis([-50 50 0 1])
set(gca,"Fontsize",[14])
xlabel("frequency (\omega)"), ylabel("FFT(S)")
title("Fourier Transform of the signal.")
```

MATLAB code for Part II of the assignment.

```
%% Clean Workspace
close all; clear all; clc;

%% Load song as vector
% Load the piano recording of the song and create a vector
tr_piano=16; % record time in seconds
y_piano=audioread("music1.wav")';
Fs=length(y_piano)/tr_piano;
t_piano=(1:length(y_piano))/Fs;
figure(1);
plot(t_piano,y_piano);
xlabel("Time [sec]"); ylabel("Amplitude");
title("Mary had a little lamb (piano)");

% Load the recorder recording of the song and create a vector
tr_rec=14; % record time in seconds
y_recorder=audioread("music2.wav")';
Fs=length(y_recorder)/tr_rec;
t_recorder=(1:length(y_recorder))/Fs;
figure(2);
plot(t_recorder,y_recorder);
xlabel("Time [sec]"); ylabel("Amplitude");
title("Mary had a little lamb (recorder)");

%% Gabor Transform of piano recording using Gaussian window.
figure(3);
L = t_piano(end);
N = length(t_piano);
spectrogram = [];
tslide = [0:0.05:L];
k = (2*pi/L)*[0:N/2-1 -N/2:-1];   % Even N
ks = fftshift(k);
for j = 1:length(tslide)
    gauss = exp(-100*(t_piano-tslide(j)).^2);       % Guassian Window
    yf_piano = gauss.*y_piano;
    yft_piano = fft(yf_piano);
    spectrogram = [spectrogram; abs(fftshift(yft_piano))];
    subplot(3,1,1), plot(t_piano,y_piano,'k',t_piano,gauss,'r',"Linewidth
       ",[2])
    subplot(3,1,2), plot(t_piano,yf_piano,'k')
    subplot(3,1,3), plot(ks, abs(fftshift(yft_piano))/max(abs(yft_piano)),
        'k')
    drawnow
    pause(0.1)
end

% Logarithmic Spectrogram
Spectrogram
figure(4);
pcolor(tslide, ks, log(spectrogram.' + 1))
axis([0 15 1400 2200])
shading interp
colormap(hot)
```

```
xlabel("Time [Sec]")
ylabel("Frequency [Hz]")

% Zoomed Logarithmic Spectrogram
figure(5);
pcolor(tslide, ks, log(spectrogram.' + 1))
shading interp
colormap(hot)
xlabel("Time [Sec]")
ylabel("Frequency [Hz]")

%% Gabor Transform of recorder recording using Gaussian window.
figure(6);
L = t_recorder(end);
N = length(t_recorder);
spectrogram = [];
tslide = [0:0.05:L];
k = (2*pi/L)*[0:N/2-1 -N/2:-1];   % Even N
ks = fftshift(k);
for j = 1:length(tslide)
    gauss = exp(-100*(t_recorder-tslide(j)).^2);    % Gaussian Window
    yf_recorder = gauss.*y_recorder;
    yft_recorder = fft(yf_recorder);
    spectrogram = [spectrogram; abs(fftshift(yft_recorder))];
    subplot(3,1,1), plot(t_recorder,y_recorder,'k',t_recorder,gauss,'r',"
       Linewidth",[2])
    subplot(3,1,2), plot(t_recorder,yf_recorder,'k')
    subplot(3,1,3), plot(ks, abs(fftshift(yft_recorder))/max(abs(
       yft_recorder)), 'k')
    drawnow
    pause(0.1)
end

% Logarithmic Spectrogram
figure(7);
pcolor(tslide, ks, log(spectrogram.' + 1))
axis([0 14 4000 8000])
shading interp
colormap(hot)
xlabel("Time [Sec]")
ylabel("Frequency [Hz]")
title("Spectrogram of Recorder recording.")

% Zoomed Logarithmic Spectrogram
figure(8);
pcolor(tslide, ks, log(spectrogram.' + 1))
shading interp
colormap(hot)
xlabel("Time [Sec]")
ylabel("Frequency [Hz]")
```