HW 2

Zoheb Siddiqui

February 14, 2019

Introduction

Abstract

This homework is an introduction to time frequency analysis through Gabor window filtering. We first experiment with different filters and widths. Then we use the Gabor window filtering method to analyze frequencies of a song played by two different instruments.

Part 1

The purpose of this question is to experiment with time-frequency analysis using wavelets on a song that exists in Matlab.

We create Spectograms for a number of filters while experimenting with the width of the filters, and the speed at which the filter is slid across the signal.

We observe the Spectogram for each variation to gain a better understanding of how the Gabor wavelet filtering method works.

Part 2

After we become familiar with the different filters and the effects of different filters, changes in the width parameters and over/under sampling we shall use these Gabor wavelets to identify notes/frequencies in a small segment of a song, 'Mary had a little lamb', played on two different instruments.

Theoretical Background

Part 1

We know that when we Fourier transform signals we gain information about their frequency however we lose all information regarding the time.

To combat this loss we try a different method of transforming. We select a window in time and transform the data just for that window after filtering it. This gives us frequency information about the signal in that time frame. Thus we have some information about both

frequency and time at the same time. This information can be represented in the form of a Spectogram.

Part 2

For each time slice, we filter the data and Fourier transform and shift it. The frequency signal with the maximum amplitude is the one we are looking for in each time slice. The frequency at this point is the frequency of the note being played. Thus, we would like to find the index of the point with the maximum amplitude in the Fourier domain and then use that index to find the frequency for that time slice. This frequency is angular and must be converted to Hertz. To do this, we divide it by 2π. This frequency, now in Hz, can be used to identify the note being played.

Algorithm Implementation

Part 1

We first design 3 different filters. The Gaussian filter, the Shannon filter and the Ricker filter. The filters have a width parameter α and can be centered at different time intervals by altering the time parameter τ.

For each filter:
    For each width:
        Declare empty matrix to store Fourier transform values for each time slice
        For each slice in time:
            Create a filter centered around that time slice
            Filter the signal using each filter
            Take the fast Fourier transform of the filtered signal
            Take the fftshift of the Fourier transform
            Append the absolute value of the shifted signal to the Matrix declared
        Plot Spectograms for that filter width

Part 2

The algorithm is the same for both the piano and the recorder. We are gonna be using a Gaussian filter to filter each time slice and a fixed width.

Declare empty vector to store frequency values
For each slice in time:
    Create a filter centered around that time slice
    Filter the signal using each filter
    Take the fftshift of the Fourier transform
    Find the index of the maximum value of abs(fftshift)
    Find the shifted abs(frequency) at this index.

Convert frequency to Hz.

Add frequency to the vector

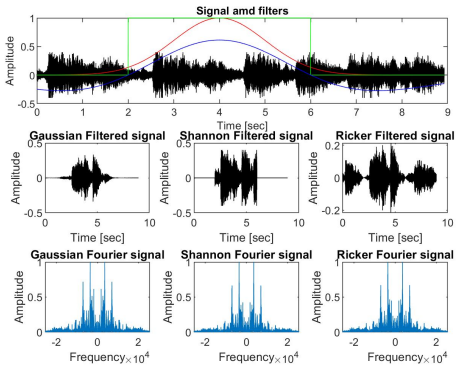
Plot the frequency vector and use the values to find score

Computational Results

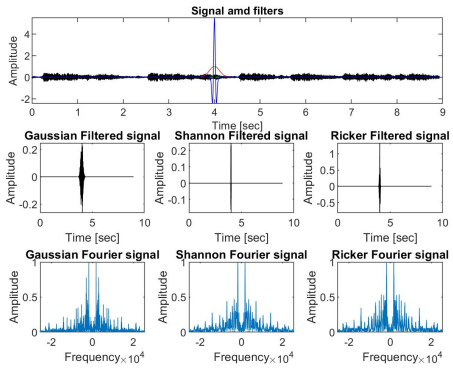
Part 1

We used three filters(Gaussian, Shannon and Ricker) with 2 different widths [50, 0.5]. For each filter and width we first over-sample and then we under-sample.

The filters and the Fourier transforms look like:



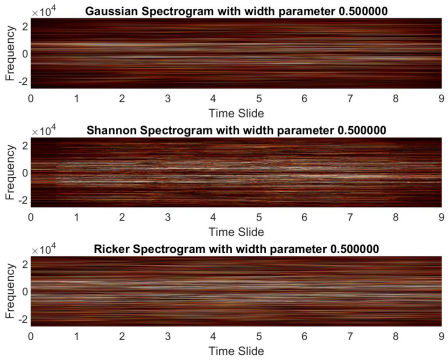
(a) Filter width parameter = 0.5



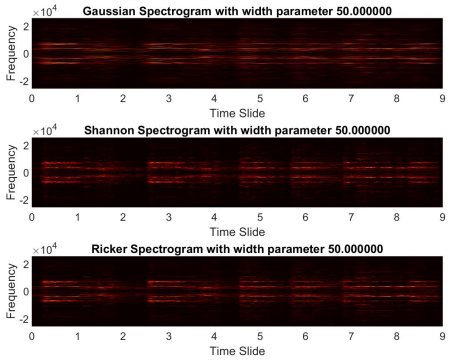
(b) Filter width parameter = 40

Figure 1: Signal and Filter

The results for over-sampling:



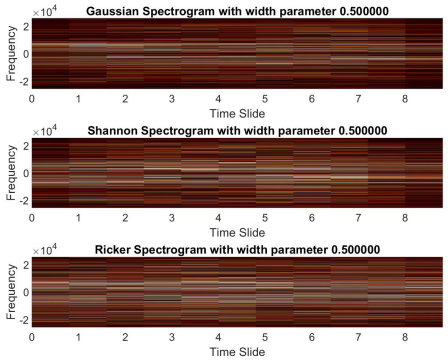
(a) Filter width parameter = 0.5



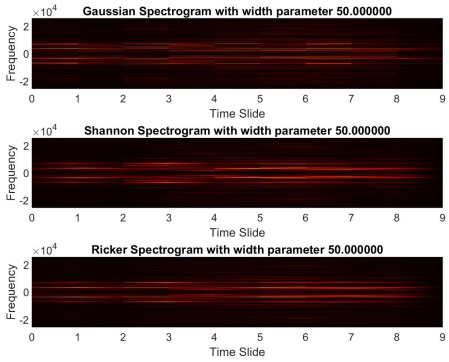
(b) Filter width parameter = 50

Figure 2: Oversampled Spectogram

The results for under-sampling:



(a) Filter width parameter = 0.5

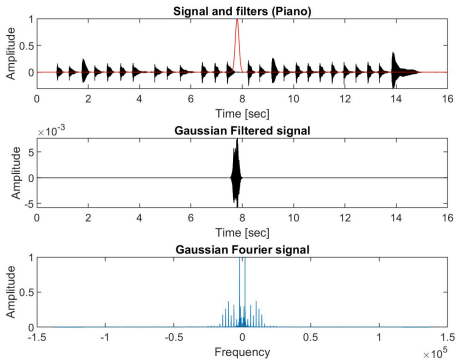


(b) Filter width parameter = 40

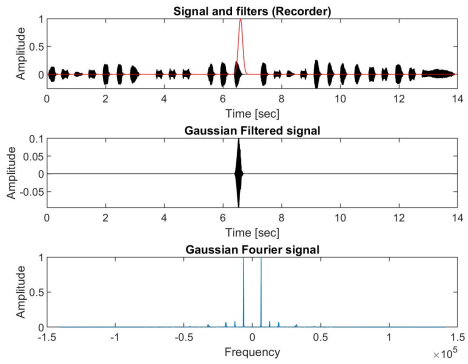
Figure 3: Undersampled Spectogram

Part 2

The filters and Fourier transforms on both data sets:



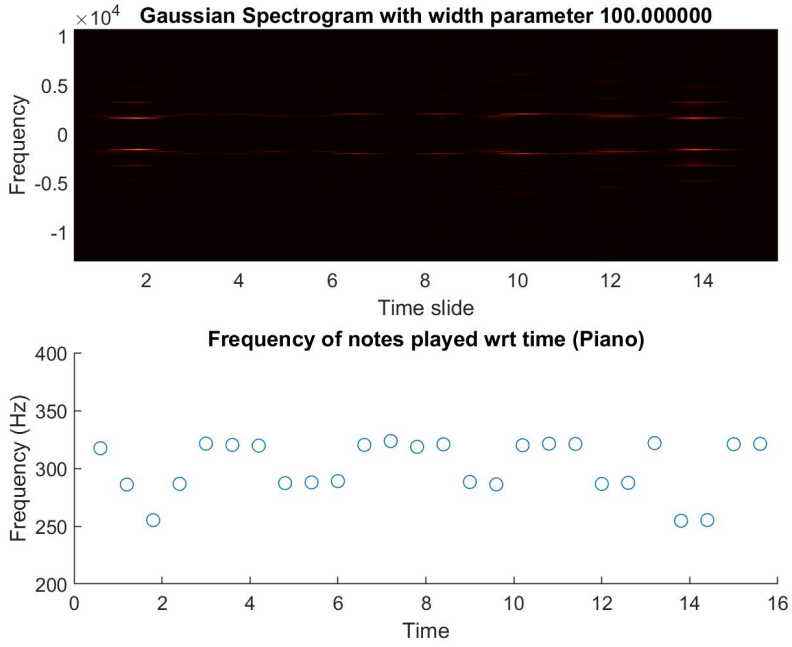
(a) Filter width parameter = 100



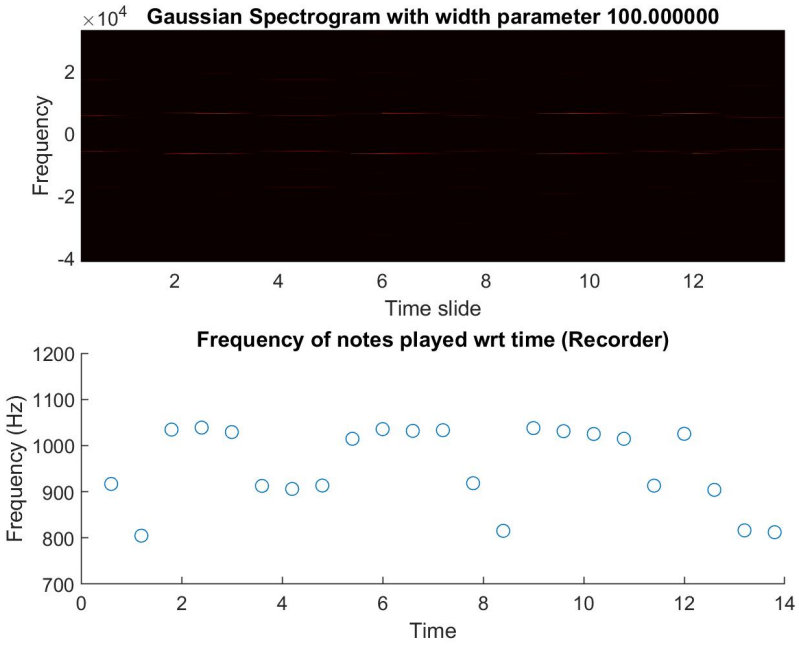
(b) Filter width parameter = 100

Figure 4: Signal and Filter

The Spectogram and frequency distribution or the piano is:



The Spectogram and frequency distribution for the recorder is:



The notes played on the piano are(in order):

[e, c#, c, c#, e, e, e, c#, c#, c#, e, e, e, e, c#, c#, e, e, e, c#, c#, e, c, c, e, e].

The notes played on the recorder are(in order):

[a#, g, c, c, c, a#, a#, a#, c, c, c, c, a#, g, c, c, c, c, a#, c, a#, g, g].

Summary

Part 1

We observe that for the same width parameters, the Gaussian filter tends to be wider than the other two. As the width parameter decreases the filter becomes wider and we lose frequency localization as multiple frequencies are filtered out at the same time. The Spectograms reflect this as it looks extremely unclear. When we have a narrow filter, we gain localized information about frequency and the Spectogram tends to be clearer.

The sampling rate also affects the clarity of the Spectogram. When we oversample we gain a better understanding of how frequency changes over time. However oversampling takes a lot of memory.

When we undersample the distribution of frequencies looks less clear as we overlook transition in frequency.

Both under and over sampling are bad for different reasons. The ideal thing to do is have the filter transition at a rate in between over and under sampling so we get a clear Spectogram while not consuming a lot of memory.

Part 2

We had audio data on a section of a song played by 2 different instruments. we used a Gabor filter at different slices in time for both instruments to figure out what note is being played at each section in time. We then compare the frequencies and the quality(timbre) of the sound and how overtones affect the timbre.

When we hear the audio we can see that the Piano sounds fuller and the Recorder sounds less rich. This is due to the presence of overtones in the piano. If we look at the Spectograms, we can clearly see them in the case of the piano.
Furthermore, the recorder is shriller as it plays at a higher frequency than the piano and this is evident when we see the scatter-plot frequency distribution.

Appendices

Matlab Function Used

fft(): Computes the fast Fourier transform of a vector.

fftshift(): Shifts the Fourier transform so that the frequencies with maximum amplitude are in the center.

pcolor(): Used to create a pseudo color plot.

colormap(): Used to alter the color of the plot.

audioread(): Used to read in an audio file as a vector.

scatter(): Used to plot a scatter-plot.

Code

Part 1

```
clear all; close all; clc

load handel
v = y'/2;

t = (1:length(v))/Fs;
L = 9;
n=length(v);
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);

v = v(1:73112);
t = t(1:73112);

width=[50 0.5];
tslide = 0:0.05:L;

for i = 1:length(width)
    Sgt_spec = [];
    Sst_spec = [];
    Smt_spec = [];
    figure(2*i - 1)
    for j=1:length(tslide)

        g = exp(-width(i)*(t - tslide(j)).^2); %gaussian filter
        s = (abs(t-tslide(j)) <= 1/width(i)); %shannon filter
        m = 2.*(1 - ((t-tslide(j))/width(i).^-1).^2)...
            .*exp(-((t-tslide(j)).^2)/...
            (2.*width(i).^-2))/(sqrt(3.*width(i).^-1)...
            .*pi^(1/4));%Ricker hat filter

        Sg = g.*v; %filtered with gaussian
```

```
Ss = s.*v; %filtered with shannon
Sm = m.*v; %mexican hat filter

Sgt = fft(Sg); %fft gaussian
Sst = fft(Ss); %fft shannon
Smt = fft(Sm); %fft mexican

subplot(3,3,1:3),
plot(t,v,'k',t,g,'r',t,s,'g',t,m,'b')
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal amd filters');

subplot(3,3,4),
plot(t,Sg,'k')
xlabel('Time [sec]');
ylabel('Amplitude');
title('Gaussian Filtered signal');

subplot(3,3,5),
plot(t,Ss,'k')
xlabel('Time [sec]');
ylabel('Amplitude');
title('Shannon Filtered signal');

subplot(3,3,6),
plot(t,Sm,'k')
xlabel('Time [sec]');
ylabel('Amplitude');
title('Ricker Filtered signal');

subplot(3,3,7),
plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)))
xlabel('Frequency');
ylabel('Amplitude');
title('Gaussian Fourier signal');

subplot(3,3,8),
plot(ks,abs(fftshift(Sst))/max(abs(Sst)))
xlabel('Frequency');
ylabel('Amplitude');
title('Shannon Fourier signal');

subplot(3,3,9),
plot(ks,abs(fftshift(Smt))/max(abs(Smt)))
```

```
        xlabel('Frequency');
        ylabel('Amplitude');
        title('Ricker Fourier signal');

        drawnow

        Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];
        Sst_spec = [Sst_spec; abs(fftshift(Sst))];
        Smt_spec = [Smt_spec; abs(fftshift(Smt))];
    end

    figure(2*i)
    subplot(3,1,1)
    pcolor(tslide,ks,Sgt_spec.'), ...
        shading interp, colormap(hot)
    str = sprintf('Gaussian Spectrogram with width parameter %f',...
        width(i));
    title(str)
    xlabel('Time Slide')
    ylabel('Frequency')

    subplot(3,1,2)
    pcolor(tslide,ks,Sst_spec.'), shading interp,...
        colormap(hot)
    str = sprintf('Shannon Spectrogram with width parameter %f',...
        width(i));
    title(str)
    xlabel('Time Slide')
    ylabel('Frequency')

    subplot(3,1,3)
    pcolor(tslide,ks,Smt_spec.'), shading interp,...
        colormap(hot)
    str = sprintf('Ricker Spectrogram with width parameter %f',...
        width(i));
    title(str)
    xlabel('Time Slide')
    ylabel('Frequency')
end
```

Part 2

```
clear all; close all; clc

L=16; % record time in seconds
```

```
y=audioread('music1.wav');
Fs=length(y)/L;
v = y'/2;
t = (1:length(v))/Fs;
n=length(v);
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);

width=[100];
tslide = 0:0.6:L;

for i = 1:length(width)
    figure(2*i - 1)
    title('Piano')

    Sgt_spec = [];
    frequencies = [];
    for j=1:length(tslide)
        g = exp(-width(i)*(t - tslide(j)).^2);
        Sg = g.*v; %filtered with gaussian
        Sgt = fft(Sg); %fft gaussian

        %signal and filter plot
        subplot(3,1,1)
        plot(t,v,'k',t,g,'r')
        xlabel('Time [sec]');
        ylabel('Amplitude');
        title('Signal and filters (Piano)');

        subplot(3,1,2),
        plot(t,Sg,'k')
        xlabel('Time [sec]');
        ylabel('Amplitude');
        title('Gaussian Filtered signal');

        subplot(3,1,3),
        plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)))
        xlabel('Frequency');
        ylabel('Amplitude');
        title('Gaussian Fourier signal');

        drawnow

        [val, index] = max(abs(fftshift(Sgt)));
        frequency = ks(index)/(2*pi);
```

```
        frequencies = [frequencies, frequency];

        Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];
    end
    frequencies = abs(frequencies);
    %disp(frequencies);
    figure(2*i)

    subplot(2,1,1)
    pcolor(tslide,ks,Sgt_spec.'), ...
        shading interp, colormap(hot)
    %axis([0 length(tslide) 300000 400000])
    str = sprintf('Gaussian Spectrogram with width parameter %f',...
        width(i));
    title(str)
    xlabel('Time slide')
    ylabel('Frequency')

    subplot(2,1,2)
    scatter(tslide(2:length(tslide)),frequencies(2:length(tslide)));
    xlabel('Time');
    ylabel('Frequency (Hz)');
    title('Frequency of notes played wrt time (Piano)');
    axis([0 L 200 400])
    drawnow
end


%%
%clear all; close all; clc

L=14; % record time in seconds
y=audioread('music2.wav');
Fs=length(y)/L;
v = y'/2;
t = (1:length(v))/Fs;
n=length(v);
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);

width=[100];
tslide = 0:0.6:L;

for i = 1:length(width)
    figure(4*i - 1)
```

```
title('Recorder')

Sgt_spec = [];
frequencies = [];
for j=1:length(tslide)
    g = exp(-width(i)*(t - tslide(j)).^2);
    Sg = g.*v; %filtered with gaussian
    Sgt = fft(Sg); %fft gaussian

    %signal and filter plot
    subplot(3,1,1)
    plot(t,v,'k',t,g,'r')
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Signal and filters (Recorder)');

    subplot(3,1,2),
    plot(t,Sg,'k')
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Gaussian Filtered signal');

    subplot(3,1,3),
    plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)))
    xlabel('Frequency');
    ylabel('Amplitude');
    title('Gaussian Fourier signal');

    drawnow

    [val, index] = max(abs(fftshift(Sgt)));
    frequency = ks(index)/(2*pi);
    frequencies = [frequencies, frequency];

    Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];
end
frequencies = abs(frequencies);
%disp(frequencies);
figure(4*i)

subplot(2,1,1)
pcolor(tslide,ks,Sgt_spec.'), ...
    shading interp, colormap(hot)
%axis([0 length(tslide) 300000 400000])
str = sprintf('Gaussian Spectrogram with width parameter %f',...
```

```
        width(i));
    title(str)
    xlabel('Time slide')
    ylabel('Frequency')

    subplot(2,1,2)
    scatter(tslide(2:length(tslide)),frequencies(2:length(tslide)));
    xlabel('Time');
    ylabel('Frequency (Hz)');
    title('Frequency of notes played wrt time (Recorder)');
    axis([0 L 700 1200])
    drawnow
end
```