Homework 2: Gábor Transforms

Maximilian Walter
AMATH 482
University of Washington

This assignment looks at different filtering methods to evaluate music files. Using different types of Gábor transforms I was able to decompose music into their frequencies and then determine the notes being played. I evaluate how different parameters, filter width and time steps, effect the resolution of the spectrograms as well as the tradeoffs for using different types of wavelets.

Introduction and Overview

In part I, I look at the different parameters of these filters to see the tradeoffs between time and frequency resolution. I look at window size and time steps size using the Gaussian Filter. I then look at different wavelets, Mexican Hat (Ricker) wavelet and the Shannon Window and their ability to filter the signal. In part II. I use two recordings of the same song, Mary had a Little Lamb, played on two different instruments, a piano and a flute, and used filtering to determine the notes being played.

Theoretical Background

Fourier Series and Transform

The Fourier series and transform are analysis tools to help us better understand a signals process. The Fourier series is a mathematical equation that allows the user to recreate a signal by combining a series of Sines and Cosines at different frequencies. The Fourier Transform is a tool to analyze an incoming signal by decomposing it into the frequencies that make up that signal. Similar to music, each chord has a series of notes, that together make up the sound we hear. The Fourier transform allows the user to pick out the individual notes and their strengths, which in this case is frequency. The transform relies heavily on Euler's formula, which allows the function to have complex exponential values, which we can represent with sines and cosines.

Euler's Formula

$$e^{i\theta} = cos\theta + isin\theta \tag{1}$$

This formula allows us to take the transform using the following equation.

Fourier Transform

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

Gábor Transforms

In this paper I used three different Gábor transforms to filter my signal. The first filter, the most basic, is the Gaussian Filter. This simply uses a Gaussian curve to filter the signal over across a sliding window. The original signal is multiplied by these filters in the time domain and then the signal is taken into the frequency domain using the Fourier transform. The width of the filter is determined by alpha, and the position of the filter is determined by tau.

Gaussian Filter

$$Gaussian\ Filter = e^{-a(t-tau)^2} \tag{3}$$

The second filter used is the Mexican Hat Wavelet. This, as the name implies, belongs into the wavelet family. This filter has improved time localization compared to the Gaussian filter, as the wavelet oscillates around the center then returns back to zero.

Mexican Hat (Ricker) Wavelet

$$Mexican\ Hat = (1 - a(t - tau)^2) * e^{\frac{1-a(t-tau)}{2}} \tag{4}$$

Shannon Window

$$Shannon\ Window = Heaviside\left(t - \left(tau - \frac{window}{2}\right)\right) - Heaviside(t - \left(tau + \frac{window}{2}\right)) \tag{5}$$

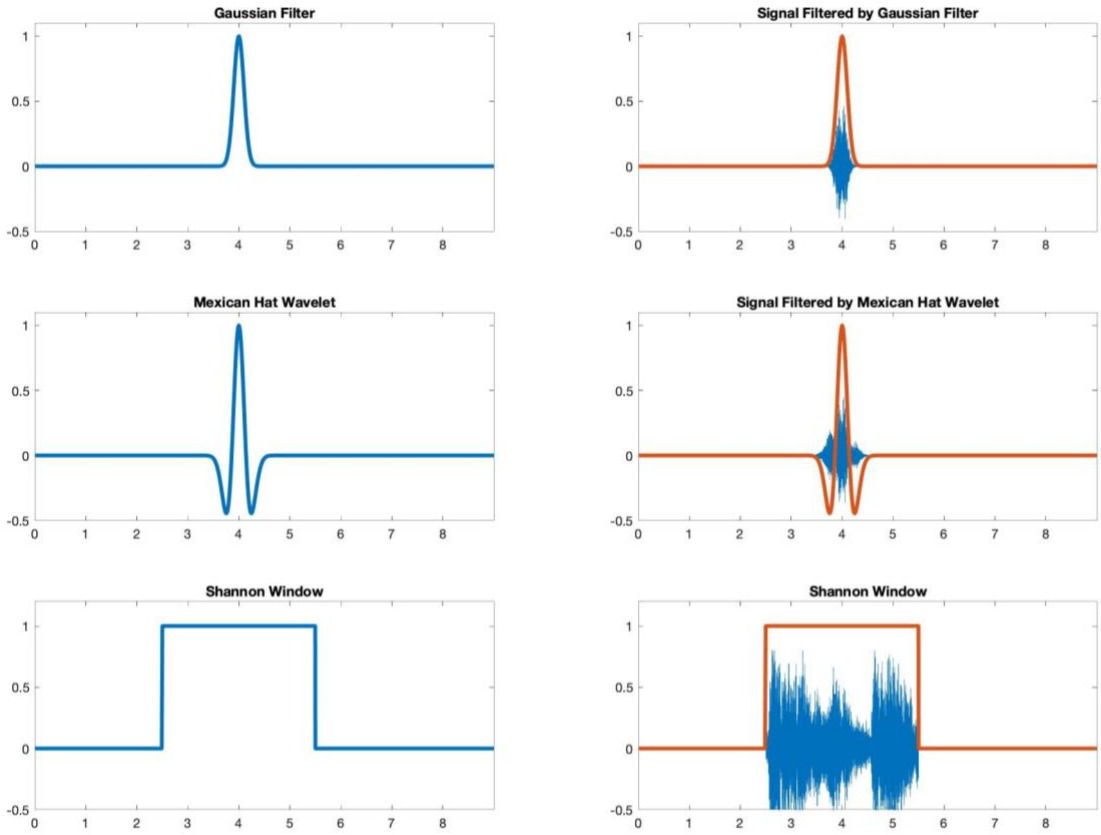

Figure 1: The shape of each filter and its effect on the signal.

Algorithm Implementation and Development

Part I.

Variables

This section uses the built-in audio file, *handel*, which is a 9 second recording of Handel's Hallelujah Chorus. I then down sample this data by 5 to improve speed for testing. The file, when loaded in, gives the signal as well as the sampling frequency. I then set up the length, sample, time, and frequency variables for the rest of the code. For my spectrograms, I am using the Nyquist theorem in limiting my y axis, but because of down sampling, a ¼ ratio is used instead of ½.

Parameter Testing

The first initial test was done using the Gaussian filter, with a width of 10, and a time window of 0.1 seconds. I created a loop that for the length of my sliding window, multiplying the signal by the Gaussian filter then Fourier transforming this into the frequency domain. For plotting, the data was then shifted and only the absolute value was used, removing the complex

components of the signal. This was stored for each time point. To plot, I looked at the original signal in time, the unfiltered signal in frequency, the filtered signal in frequency, and finally a spectrogram of the entire set of data. The results can be seen in figure 2.

I then tested how the width of the filter, alpha, effected the resolution of my spectrogram. I began by looking at alpha from 0:100 in increments of 10. For this report, I only show the results in increments of 25. The same process is used to filter the signal as well placing the results into a subplot. The results can be found in figure 3.

For my third evaluation, I looked at the sampling window and how different ranges of time effects my spectrogram. I used a sliding window of size 0.1:1.5 for testing in increments of 0.1. For the results, I will only be showing the range of 0.1:1.1 in increments of 1.1. the same filtering process was used for this section, with the addition of a *clear* variable at the beginning of each loop to reset my storage variable. The results can be found in figure 4.

Filter Testing

For this section, I look at Gaussian, Mexican Hat, and Shannon filters and their effects on my temporal/spectral resolution. The previous sections allowed me to see that an alpha of 50 and a time window of 0.1 provided the best results, and these variables are used in the remaining methods.

The Gaussian Filter was in used in the previous section, but the curve and effect on the signal can be seen in figure 1.

The Mexican Hat Wavelet is shown in equation (4). The wave and effect on filtering the signal is shown in figure 1. The implementation of this wavelet is similar to that of the Gaussian, where the wavelet is just multiplied by the signal. The output is then Fourier transformed and shifted for plotting.

The final wavelet tested was the Shannon Window. This required the use of Heaviside functions to correctly implement. Using the sliding window, the built in Heaviside function was used. A width was determined outside of the loop, using 0.1, and this was subtracted from Tau. To center the starting point around zero, the width was divided by two. This causes the initial step to 1. To complete the square wave, another Heaviside function, shifted by the width, adding the value to tau. For plotting, the width was increased to 3. This can be seen in figure 1.

Part II.

This section used two music files, each playing Mary had a Little Lamb, music file 1 on a piano and music file 2 on a recorder. The goal was to filter the signal and determine the notes being played and recreate the music score.

Variables

Like before, the file was loaded into the script. The output contained the signal, y, and the sampling frequency, Fs. To improve speed, the signal was down sampled by 5 again. From this data, the length of the piece, the number of samples, and the time and frequency indices could be made. The signal comes out as a column vector that must be transposed into a row vector before being used.

Filtering

Similar to Part I, a Gaussian filter was used. The width used in this section was an alpha of 100 and a t slide size of 0.1 seconds. My initial for loop creates the filter and then filters the signal. I then place this into the frequency space using a Fourier transform and shift and store the absolute value in a matrix for later plotting. I also find the maximum value of the signal at each time point and find the index using the max command. This allows me to find this value in the frequency space and store it in a p_note (piano note) variable. This is done to find the frequency at each time, and thus determine the note being played. The exact same method is used for the second music file to find the maximum frequency for the notes as well as the spectral and temporal information needed for the spectrogram. This was then plotted using the *pcolor* command that takes the time and spectral space, as well as the resulting signal. To be in the correct space, the *ks* variable is divided by 2*pi to match the output data.

After looking at the spectrogram, there are overtones, or harmonics that are visible in each recording (figure. 7-9). To remove these, I implemented a low pass filter to remove these overtones. This was done using the built-in command, *lowpass,* that removed frequencies above the given threshold. I used a threshold of 370 Hz for the piano piece.

Determining Notes

After storing all the maximum frequency values and their indices, I could then determine the notes being played. I then labeled the frequency axis with the corresponding note. I could then use this to create a score using Noteflight.

Computation Results

Part I.

Parameter Sweeping

As stated, the first evaluation was done using the Gaussian filter and a baseline was set with an a = 10 and a time window of 0.1 seconds. The results can be seen in figure 2.

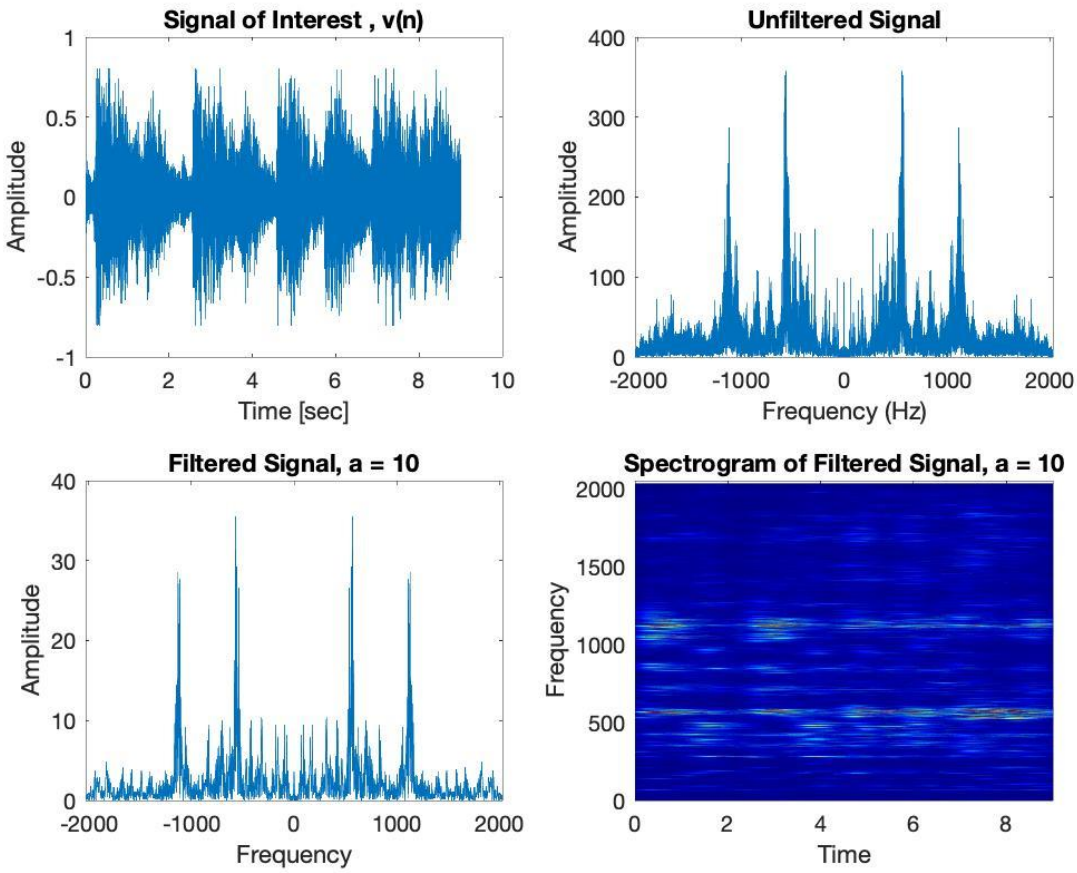

Figure 2: Gaussian Signal for baseline results. Top left is the original signal. Top Right is the unfiltered signal in the frequency space. Bottom left is the filtered signal in frequency space. Bottom right is the spectrogram after filtering.

After evaluating the plots from changing the alpha value, an alpha value of 50 was used as it provided the best temporal and spectral resolution. This was done to minimize the over filtering being done that is reducing the amplitude in the spectral space. The spectrogram doesn't greatly improve between 50 and 100, therefore 50 was the chosen value. This can be seen in figure 3.

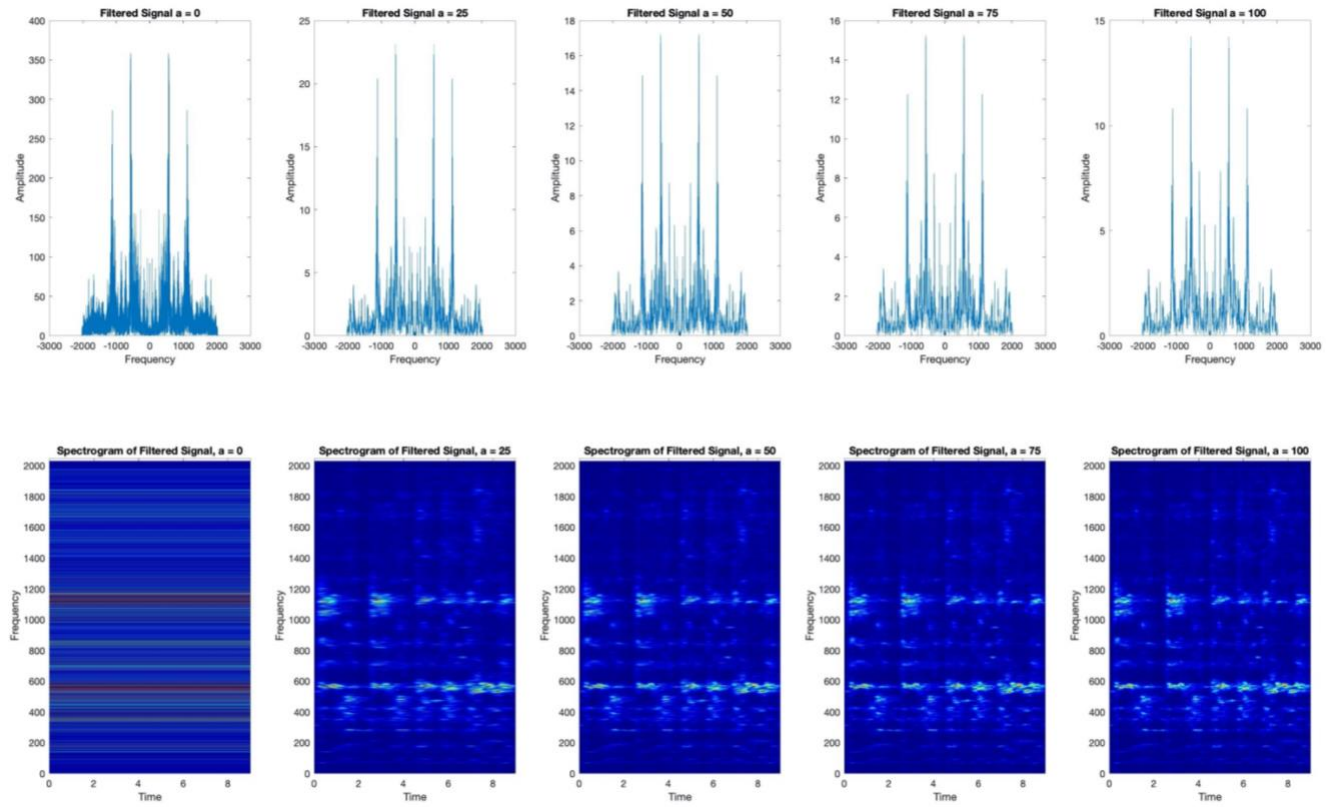

Figure 3: Altering the filter width from 0:100 in steps of 25. The top shows the filtered signal in frequency space and the bottom shows the resulting spectrogram.

The final test was to look at each time window size. The optimal time size was the smallest, using a time window of 0.1 seconds. The results can be seen in figure 4. As the window size increased, the spectrogram becomes smeared in the time domain, but the frequency resolution increases. This is expected the expected result. What I found interesting was the increase in amplitude at the 0.7 window in the filtered spectral data. This could be that the window lands directly on a high amplitude low frequency point in the song.

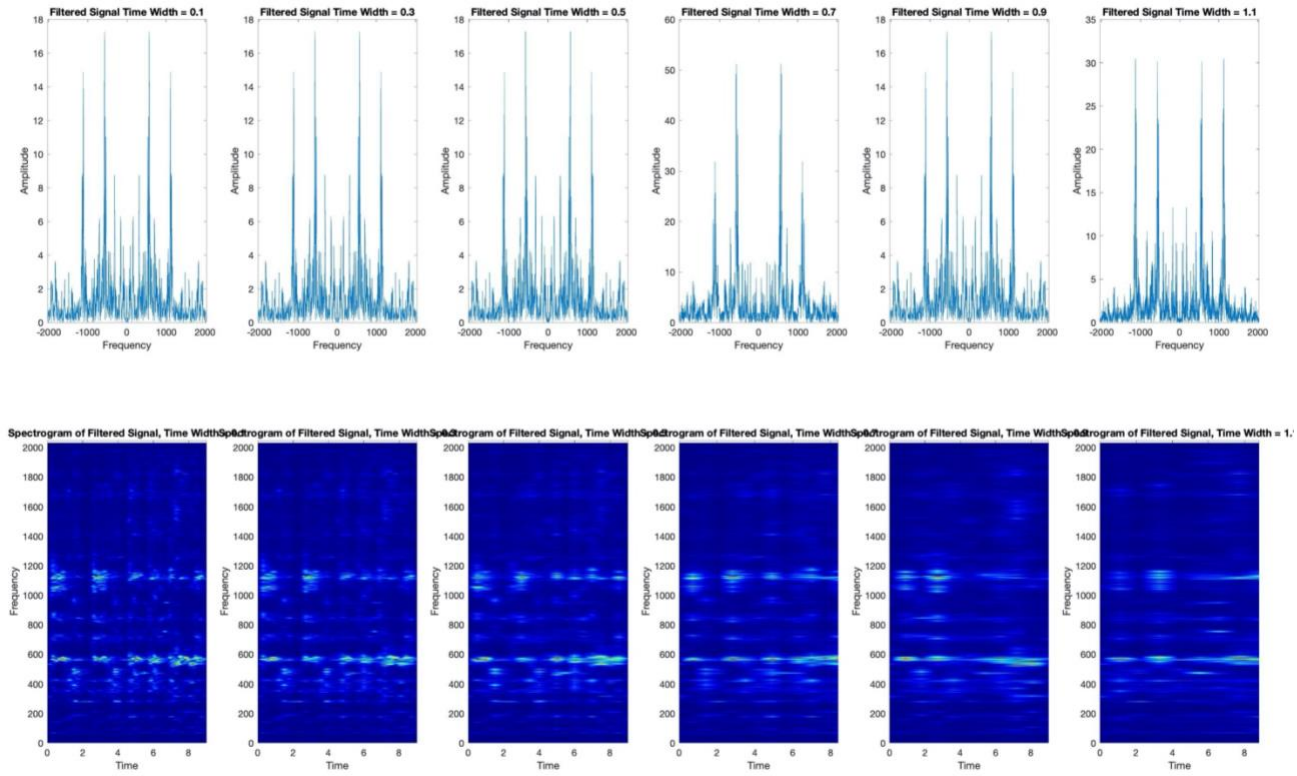

Figure 4: Altering time window size in a Gaussian Filter. Top shows filtered signal in frequency, the bottom shows the resulting spectrogram. The titles are overlapping but are the same as the top.

Different Filters

Looking at the results from the Mexican Hat and Shannon Window, the both produce similar spectrograms **(**Figure 5 and 6 respectively) I think, for this piece, the Mexican Hat provides a bit better spectral resolution as seen in the slightly smaller bins. There is also less amplitude suppression compared to that of the Shannon Window.

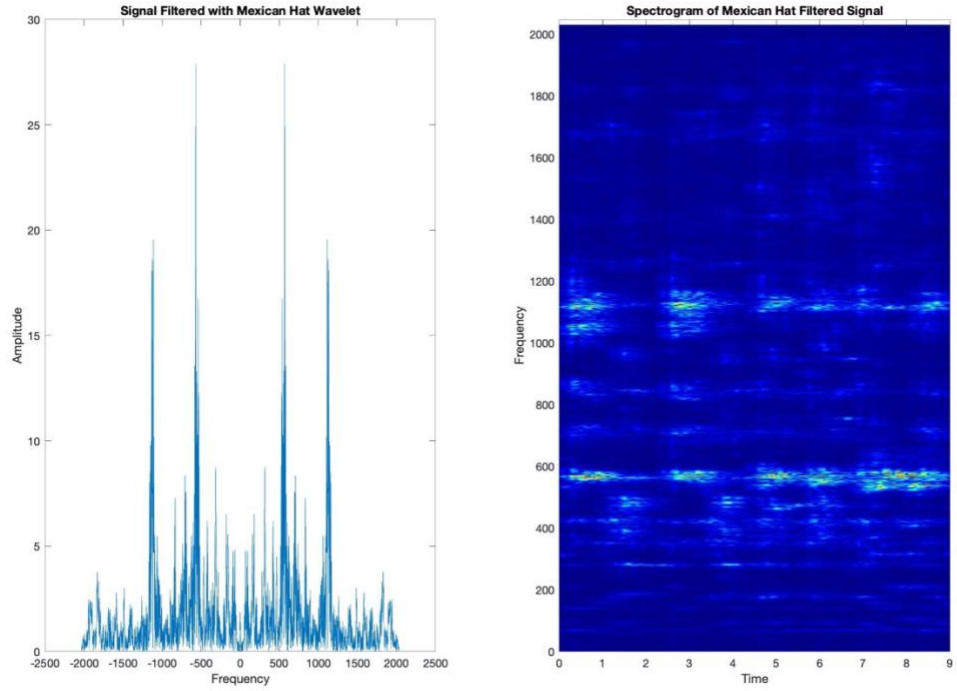

Figure 5: Mexican Hat Filter. The left is the resulting filtered signal in the frequency domain. On the right is the resulting spectrogram.

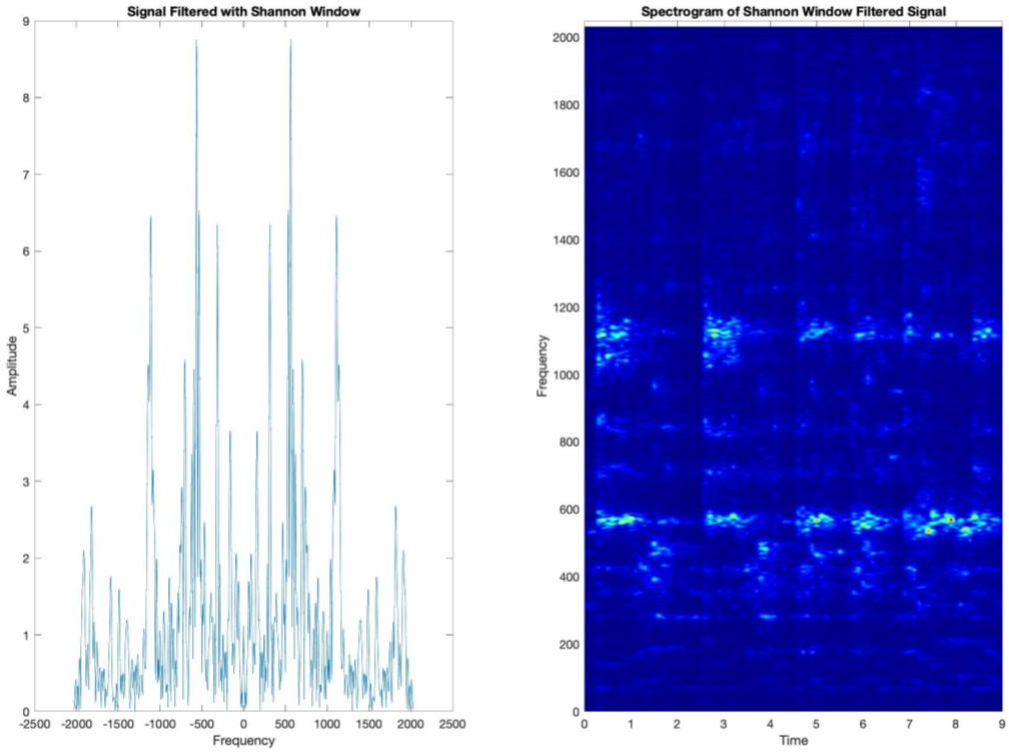

Figure 6: Shannon window filter. The left is the resulting filtered signal in the frequency domain. On the right is the resulting spectrogram.

Part II.

The initial score and spectrograms are shown in figures 7,8,9. Here we can clearly see the overtones in the recorder piece in the notes, with the visibility being more in the spectrogram for the piano. The final MATLAB music score for both the piano and recorder, as well as its filtered spectrogram, is in figure 10. To show in traditional notes, I used the software noteflight to build the score. These can be seen in figures 11 and 12.

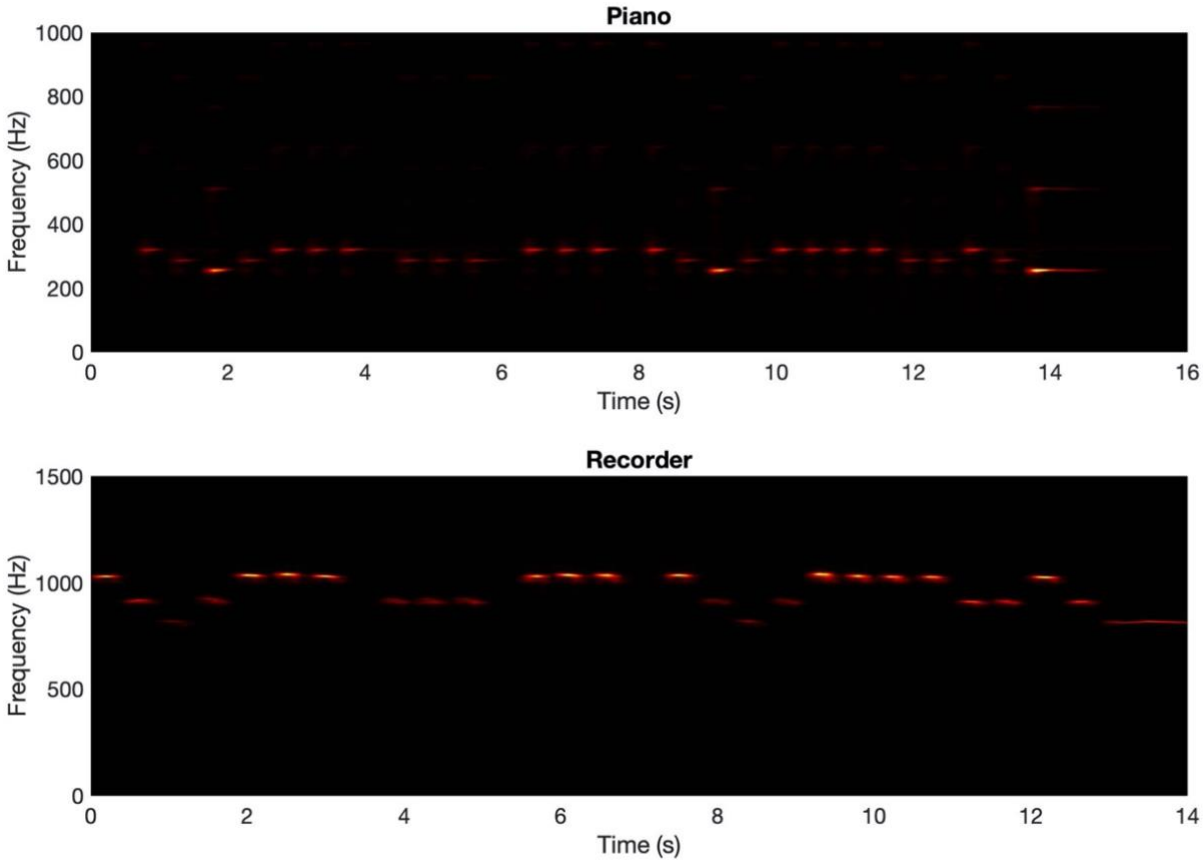

Figure 7: Top shows the unfiltered spectrogram from the piano. The bottom shows the unfiltered spectrogram from the recorder.

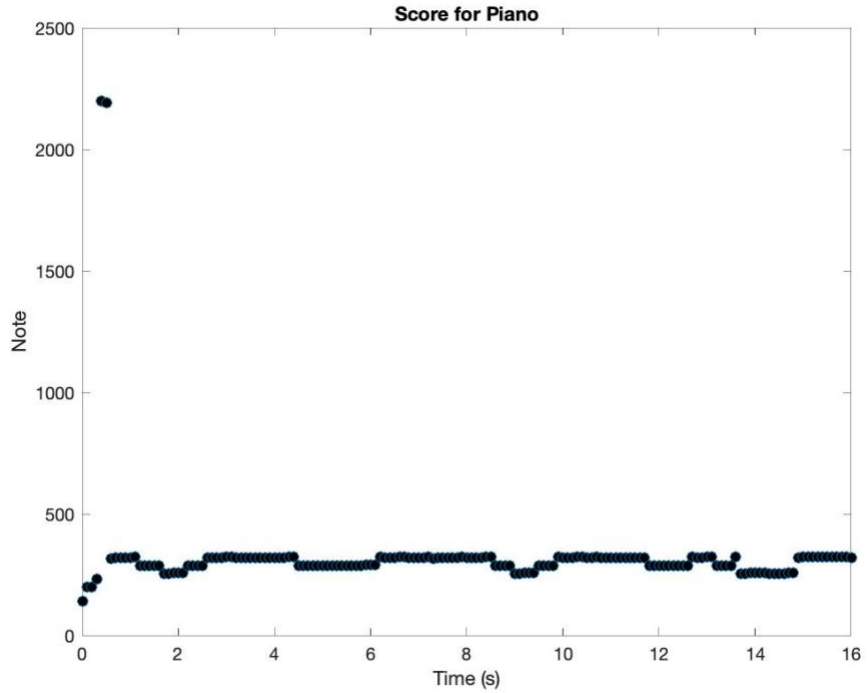

Figure 8: The unfiltered score for the piano.

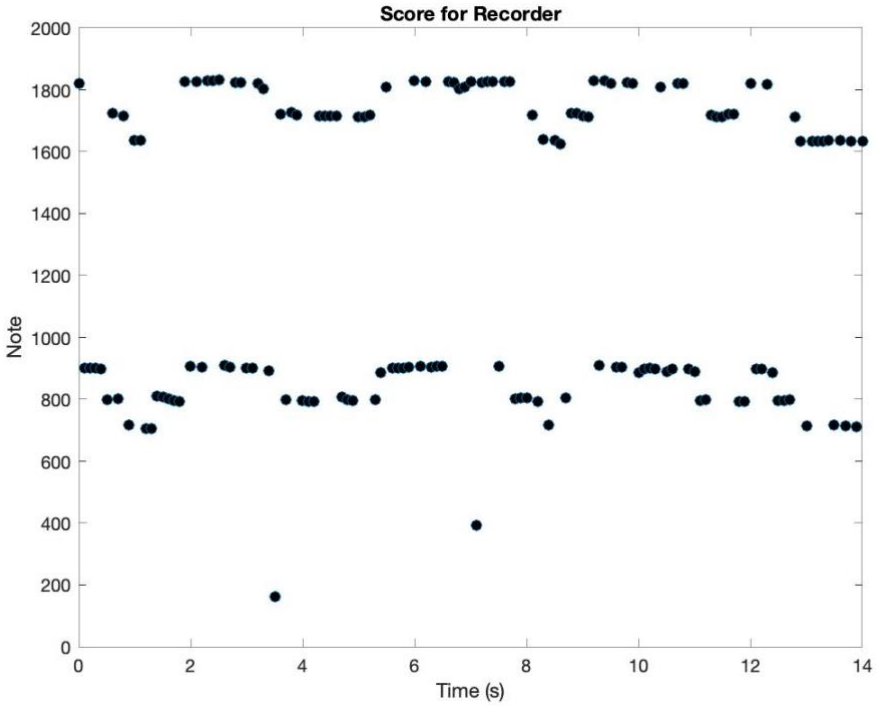

Figure 9: Unfiltered score of the recorder, showing clearly the overtones.

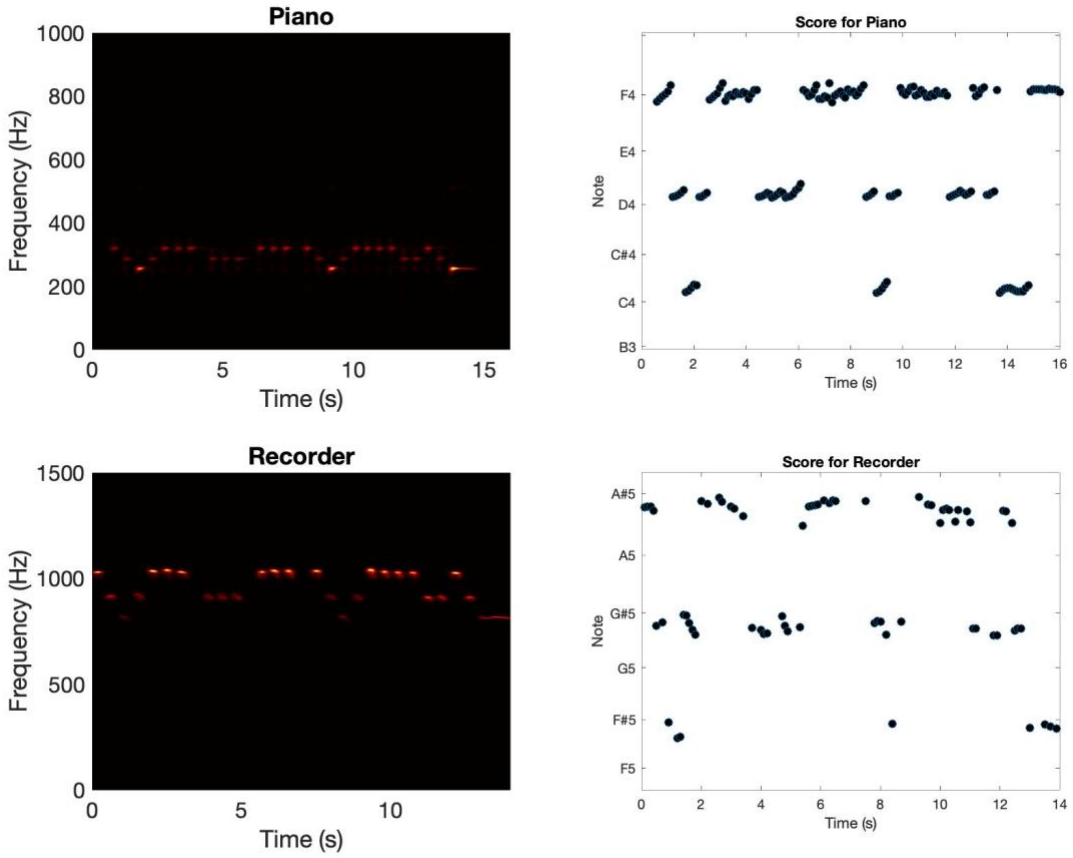

Figure 10: Filtered spectrogram for both the Piano and Recorder on the left. On the right, the notes are shown with their corresponding note frequency labeled.

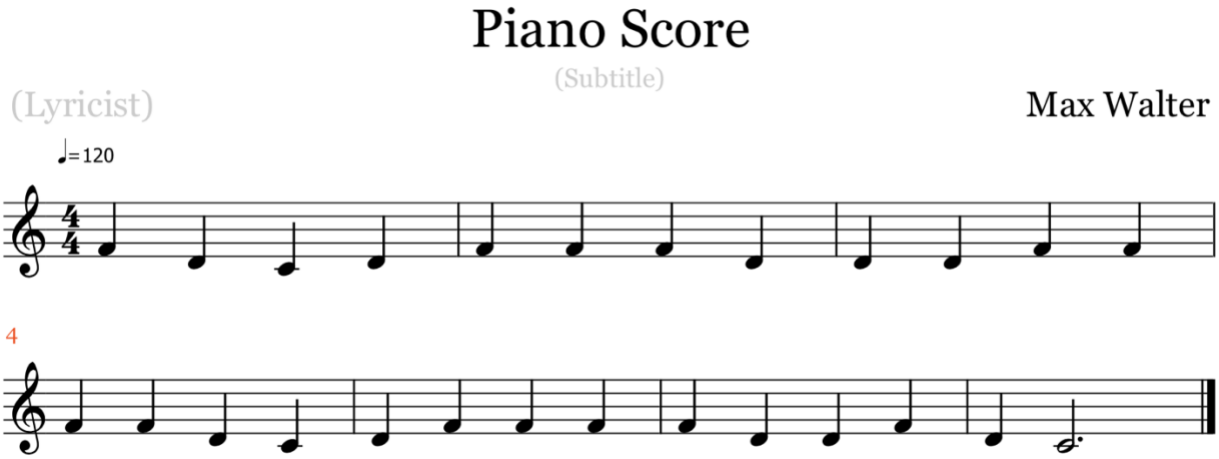

Figure 11: Piano Music Score translated from MATLAB to Noteflight.

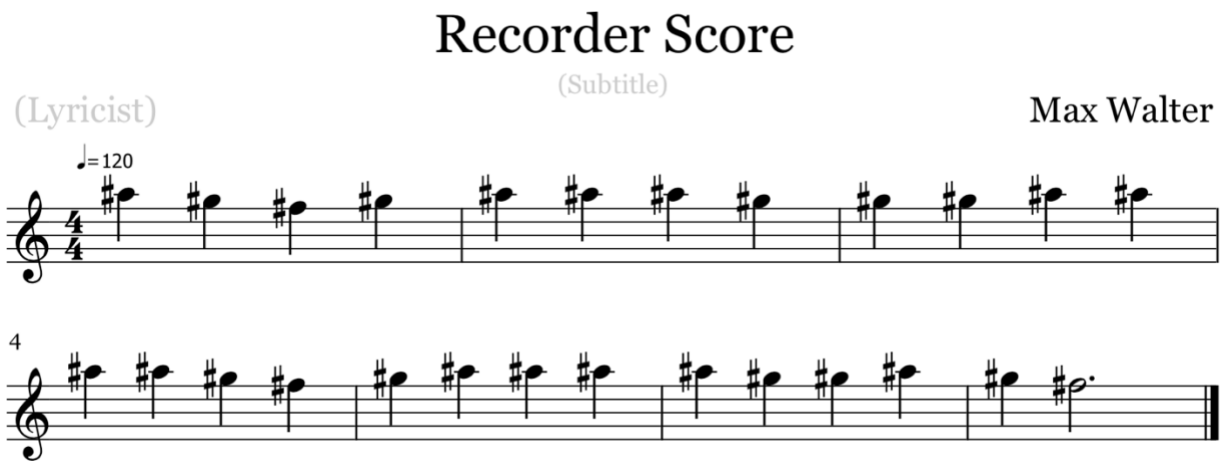

Figure 12: Recorder score translated from MATLAB to Noteflight.

Summary and Conclusions

After evaluating the width and time window for the gaussian signal I was able to determine that and alpha of 50 and a time window of 0.1 provided the best results. There is a large tradeoff between temporal and spectral resolution which follows the Heisenberg uncertainty principle as stated in class. The recorder was played at a much higher frequency as seen in the spectrogram of the two pieces. Both had overtones that could be removed using a targeted low pass filter with the recorder losing a lot of spectral detail in the process, and thus was omitted for this report.

Appendix A.

Fft(x): Takes the Fourier transform of X. We used this to transform our signal from the time into the frequency domain.

Fftshift(x): Rearranges X so that the zero-frequency component is centered around 0. We used this to center our data for plotting.

abs(x): takes the absolute value of X. This was used to remove complex numbers from out fourier transformed signal.

[max, I] – max(A): This gives the maximum value of an array, A and its index in that array. This was used to find the index value for each note.

[y, Fs] = audioread('file.wav'): This function loads in an .wav file. This was done to load in our sample signal.

X = downsample(y,n): This function takes the original signal and removes points. In our case this was done to reduce the size of our signal to improve speed and performance.

Pcolor(x,y,A): This function plots a spectrogram by plotting the time, spectral, and output of the fourier transform.

Appendix B.

Part I.

```
%Max Walter
%AMATH 482
%Homework 2: Gabor Transform and Signal Processing
%2/7/2020
%University of Washington
clear all; close all; clc

%Part I. Looking at different filters manipulating width and time window.

%Loading in built in music file of Handel's Hallelujah Chorus

load handel %Outputs [y Fs]
y = downsample(y,2); %Downsampling for improved performance
v = y'; %Comes out as a column vector
vt = fft(v); %Initial Fourier Transform of the Data

%p8 = audioplayer(v,Fs); playblocking(p8); %Code uses to play the file

%Defining Parameters
L=9; n=length(y);
t2=linspace(0,L,n+1);
t=t2(1:n);
k = (1/L)*[0:(n/2) -n/2:-1];
ks = fftshift(k);

%Plotting Original Time Signal
figure()
subplot(2,2,1)
plot(t,v);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Signal of Interest , v(n)');

%Plotting Unfiltered Signal
subplot(2,2,2)
plot(ks, abs(fftshift(vt)));
xlabel('Frequency (Hz)'); ylabel('Amplitude');
title('Unfiltered Signal');

%Gaussian Filter
%g = exp(-a*(t-tau).^2);

a = 10; %Width
t_slide = 0:0.1:L; %Sliding Window

for j = 1:length(t_slide)
    g = exp(-a*(t-t_slide(j)).^2);
```

```
    sg = g.*v; %Filtered Signal
    sgt = fft(sg); %FFT of Filtered Signal
    ssgt(j,:) = abs(fftshift(sgt)); %Shifted filter Signal for making
spectrogram.
end

%Plotting Filtered Signal
subplot(2,2,3)
plot(ks, abs(fftshift(sgt)));
xlabel('Frequency'); ylabel('Amplitude');
title('Filtered Signal, a = 10');

%Plotting Spectrogram
subplot(2,2,4)
pcolor(t_slide,ks,ssgt.'),
xlabel('Time'); ylabel('Frequency');
title('Spectrogram of Filtered Signal, a = 10');
set(gca,'Ylim',[0 Fs/4])

shading interp
colormap(jet)

%%
%%%%%
%Plotting Different Width


%Plotting Original Signal
figure()
subplot(2,1,1)
plot(t,v);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Signal of Interest , v(n)');

%Plotting Unfiltered
subplot(2,1,2)
plot(ks, abs(fftshift(vt)));
xlabel('Frequency (Hz)'); ylabel('Amplitude');
title('Unfiltered Signal');

%Running through different Widths
IDX = 1;
figure()
for a = 0:25:100

    for j = 1:length(t_slide)
        g = exp(-a*(t-t_slide(j)).^2);
        sg = g.*v;
        sgt = fft(sg);
        ssgt(j,:) = abs(fftshift(sgt));
    end

    subplot(2,5,IDX)
    plot(ks, abs(fftshift(sgt)));
    xlabel('Frequency'); ylabel('Amplitude');
```

```
    title(['Filtered Signal a = ' num2str(a)]);

    IDX = IDX + 5;
    subplot(2,5,IDX)
    pcolor(t_slide,ks,ssgt.'),
    xlabel('Time'); ylabel('Frequency');
    title(['Spectrogram of Filtered Signal, a = ' num2str(a)]);

    IDX = IDX - 4;
    shading interp
    set(gca,'Ylim',[0 Fs/4])
    colormap(jet)
end

%%
%%%%%%%%
%Testing Different Sized Sampling Windows
%Looking at sizes from 0.1 second to 1.5 seconds

a = 50; %Redefining filterd width from previous section
IDX = 1;
figure()
%Filtering Times
for t_slide_w = 0.1:0.2:1.1
    clear ssgt;

    t_slide = 0:t_slide_w:L;

    %Running New slide over signal through Gaussian Filter
    for j = 1:length(t_slide)
        g = exp(-a*(t-t_slide(j)).^2);
        sg = g.*v; %filtered signal
        sgt = fft(sg); %FFT of filtered signal
        ssgt(j,:) = abs(fftshift(sgt)); %Shifted filter Signal
    end

    %Plotting Filtered Signal
    subplot(2,6,IDX)
    plot(ks, abs(fftshift(sgt)));
    xlabel('Frequency'); ylabel('Amplitude');
    title(['Filtered Signal Time Width = ' num2str(t_slide_w)]);
    IDX = IDX + 6;

    %Plotting Unfiltered Signal
    subplot(2,6,IDX)
    pcolor(t_slide,ks,ssgt.'),
    xlabel('Time'); ylabel('Frequency');
    title(['Spectrogram of Filtered Signal, Time Width = '
num2str(t_slide_w)]);
    IDX = IDX - 5;

    shading interp
    set(gca,'Ylim',[0 Fs/4])
    colormap(jet)
```

```
end

%%
%%%%%%
%Mexican Hat Filter

%Using Values from previous tests
t_slide = 0:0.1:L;
a = 50;


for j = 1:length(t_slide)
    m_hat = (1-a*(t-t_slide(j)).^2).*exp((-1*a*(t-t_slide(j)).^2)/2);
%Mexican Hat Function
    sg = m_hat.*v;
    sgt = fft(sg);
    ssgt(j,:) = abs(fftshift(sgt));
end


figure()
%Filtered Signal with Mexican Hat Wavelet
subplot(1,2,1)
plot(ks, abs(fftshift(sgt)));
xlabel('Frequency'); ylabel('Amplitude');
title('Signal Filtered with Mexican Hat Wavelet');

subplot(1,2,2)
pcolor(t_slide,ks,ssgt.'),
xlabel('Time'); ylabel('Frequency');
title('Spectrogram of Mexican Hat Filtered Signal');

shading interp
set(gca,'Ylim',[0 Fs/4])
colormap(jet)


%%
%%%%%%%%%
%Shannon Window
%Heaviside Function to Step and Act as Shannon window
%First part ramps up holds at 1, then ramps down

t_slide = 0:0.1:L;
width = 0.1; %Selecting hold width of step function

for j = 1:length(t_slide)
    sg = (heaviside(t-(t_slide(j) - width/2)) - heaviside(t-(t_slide(j) +
width/2))).*v;
    sgt = fft(sg); %FFT of filtered signal
    ssgt(j,:) = abs(fftshift(sgt));
end

figure()
```

```
%Signal after Shannon Window
subplot(1,2,1)
plot(ks, abs(fftshift(sgt)));
xlabel('Frequency'); ylabel('Amplitude');
title('Signal Filtered with Shannon Window');

%Spectrogram of Signal
subplot(1,2,2)
pcolor(t_slide,ks,ssgt.'),
xlabel('Time'); ylabel('Frequency');
title('Spectrogram of Shannon Window Filtered Signal');
set(gca,'Ylim',[0 Fs/4])

shading interp
colormap(jet)

%%
%%%%
%Plotting Commands for functions used to filter

tau = 4;
a = 50;
width = 3;

%Gaussian
G = exp(-a*(t-tau).^2);

%Mexican Hat
m_hat = (1-a*(t-tau).^2).*exp((-1*a*(t-tau).^2)/2);

%Shannon
shannon = heaviside(t-(tau - width/2)) - heaviside(t-(tau + width/2));

figure()
subplot(3,2,1)
plot(t,G,'Linewidth',3)
axis([-inf inf -0.5 1.1])
title('Gaussian Filter')

subplot(3,2,2)
plot(t,G.*v)
hold on
plot(t,G,'Linewidth',3)
axis([-inf inf -0.5 1.1])
title('Signal Filtered by Gaussian Filter')

subplot(3,2,3)
plot(t,m_hat,'Linewidth',3)
axis([-inf inf -0.5 1.1])
title('Mexican Hat Wavelet')

subplot(3,2,4)
plot(t,m_hat.*v)
hold on
plot(t,m_hat,'Linewidth',3)
```

```
axis([-inf inf -0.5 1.1])
title('Signal Filtered by Mexican Hat Wavelet')

subplot(3,2,5)
plot(t,shannon,'Linewidth',3)
axis([-inf inf -0.5 1.2])
title('Shannon Window')

subplot(3,2,6)
plot(t,shannon.*v)
hold on
plot(t,shannon,'Linewidth',3)
axis([-inf inf -0.5 1.2])
title('Shannon Window')
```

Part II.

```
%AMATH 482 HW 2
%Code by Max Walter
%1/30/2020 V2

%%
%Part II.
clear all; close all; clc

%Piano
%figure()
[y,Fs] = audioread('music1.wav');
tr_piano=length(y)/Fs; % record time in seconds
L = round(length(y)/Fs);
y = downsample(y,5);

% plot((1:length(y))/Fs,y);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title('Mary had a little lamb (piano)');
%p8 = audioplayer(y,Fs); playblocking(p8);

piano_signal = y';
n = length(y);
t2p = linspace(0,tr_piano,n+1);
tp=t2p(1:n);

kp = (2*pi/L)*[0:(n/2)-1 -n/2:-1];
ksp = fftshift(kp);

t_slide_p = 0:0.1:L;
a = 100;

pnotes = zeros(length(t_slide_p),1); %Initilizing pnotes vector
spp = zeros(length(t_slide_p),length(y)); %Initilizing Shifted Signal Matrix

for j = 1:length(t_slide_p)
    g = exp(-a*(tp - t_slide_p(j)).^2);
```

```
    fps = g.*piano_signal; %filtered piano signal
    lfps = lowpass(fps,370,Fs);
    fpst = fft(lfps); %filtered Piano Signal Transform

    [M,I] = max(fpst); %Finding Index where signal is maximum for note
    pnotes(j,:) = abs(kp(I))/(2*pi); %Using Finding maximum index in
frequency.
    spp(j,:) = abs(fftshift(fpst)); %Shifting signal for spectrogram
end

% figure()
% plot(ksp, fftshift(fpst))

figure();
subplot(2,2,1)
pcolor(t_slide_p , ksp/(2*pi) ,spp.')
shading interp
set(gca,'Ylim',[0 1000],'Fontsize',16)
colormap(hot)
xlabel("Time (s)");
ylabel("Frequency (Hz)");
title("Piano");

%Recorder

[y,Fs] = audioread('music2.wav');
tr_recorder=length(y)/Fs; % record time in seconds
Lr = round(length(y)/Fs);
y = downsample(y,5);

% plot((1:length(y))/Fs,y);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title('Mary had a little lamb (piano)');
%p8 = audioplayer(y,Fs); playblocking(p8);

recorder_signal = y';
n = length(y);
t2r = linspace(0,tr_recorder,n+1);
tr=t2r(1:n);

kr = (2*pi/Lr)*[0:(n/2) -n/2:-1];
ksr = fftshift(kr);

t_slide_r = 0:0.1:Lr;
a = 100;



rnotes = zeros(length(t_slide_r),1);
spr = zeros(length(t_slide_r),length(y));

for j = 1:length(t_slide_r)
    g = exp(-a*(tr - t_slide_r(j)).^2);
    frs = g.*recorder_signal; %filtered recorder signal
    frst = fft(frs); %filtered Recorder Signal Transform
```

```
    [M,I] = max(frst);
    rnotes(j,:) = abs(kp(I))/(2*pi);
    spr(j,:) = abs(fftshift(frst));
end


subplot(2,2,3)
pcolor(t_slide_r , ksr/(2*pi) ,spr.')
shading interp
set(gca,'Ylim',[0 1500],'Fontsize',16)
colormap(hot)
xlabel("Time (s)");
ylabel("Frequency (Hz)");
title("Recorder");


subplot(2,2,2)
plot(t_slide_p,pnotes+10,'o','MarkerFaceColor', 'k');
yticks([246.94, 261.63, 277.18, 293.66, 311.13, 329.63, 349.23]);
yticklabels({'B3','C4','C#4','D4','E4','F4'});
ylim ([246 350])
title("Score for Piano");
xlabel("Time (s)"); ylabel("Note");

subplot(2,2,4)
plot(t_slide_r,rnotes + 20,'o', 'MarkerFaceColor', 'k')
yticks([698.46, 739.99, 783.99, 830.61, 880, 932.33]);
yticklabels({'F5','F#5','G5','G#5','A5','A#5'});
ylim ([680 950])
title("Score for Recorder");
xlabel("Time (s)"); ylabel("Note");
```