AMATH 582 Homework 2

Christopher Liu

February 7, 2020

Abstract

Fourier transforms are a powerful tool for data analysis as it allows us to analyze the data in the frequency domain. However, there is one main limitation to the Fourier transform as it does not retain information in time. In order to perform a time-frequency analysis of non-stationary signals, we will introduce the concept of Gabor transforms, windowed Fourier transforms. Using sample music from Handel, we will explore how changing parameters in the transforms affects resulting spectrograms. We will then analyze two recordings of the Mary had a Little Lamb, one with a piano and the other with a recorder, to produce a score of the music and to observe differences in frequency content of the two instruments.

1 Introduction and Overview

Using Fourier Transforms to analyze data in the frequency domain has many advantages. For example in our previous assignment, we were able to take advantage of the properties of data in the frequency domain to extract information from noisy data. However, the Fourier transform has a severe limitation in that it does not record the moment in time a specific frequency is present [1]. This limitation becomes apparent if the signal we would like to analyze is changing in time (a non-stationary signal) such as a song.

To overcome this limitation, we introduce the concept of Windowed Fourier Transforms. By discretizing our signal into separate time windows, we are able to capture both frequency and time in the signal. We will achieve this by using Gabor transforms. In this assignment, we will analyze a music sample by Handel to explore how changing the width and increment of the discretization affects the produced spectrogram. Furthermore, we will explore how using different Gabor windows changes the produced spectrograms. Incorporating what we learn from exploring these parameters, we will then reproduce a music score for 2 recordings of Mary had a little lamb (MHLL) on a piano and a recorder.

2 Theoretical Background

Once again we will employ Fourier transforms for our time-frequency analysis to extract the frequency information from the songs. By windowing the Fourier transforms, we will be able to localize the frequency data in time in order to produce spectrograms.

2.1 Windowed Fourier Transforms

2.1.1 Fourier Transform

The Fourier transform (1) and its inverse (Equation 2) are defined as the following, where k are the continuous wavenumbers.

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

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

Using the Fourier transform, we are able to extract the frequency content of a given signal.

2.1.2 Gabor Transform

The Gabor transform, named after Gabor Denes, allows us to localize a signal in both time and frequency domains [1]. We can achieve this by modifying the Fourier transform kernel given by (3) which introduces a new term, $g(\tau - t)$

$$g_{t,\omega}(\tau) = e^{i\omega\tau} g(\tau - t) \tag{3}$$

Thus, the Gabor transform, also known as the short-time Fourier transform is the following,

$$G[f](\tau, \omega) = \tilde{f}_g(t, \omega) = \int_{-\infty}^{\infty} f(\tau)\bar{g}(\tau - t)e^{i\omega\tau} d\tau = (f, \bar{g}_{t,\omega}) \tag{4}$$

where integrating over the parameter τ in the $g(\tau - t)$ term allows us to slide the time window, centered at τ, over the entire signal. In practice we use a discrete version of the Gabor transform for a lattice of points $\nu = m\omega_0$ and $\tau = nt_0$ where m and n are integers and $\omega_0, t_0 > 0$ are constants. The discrete kernel $g_{m,n}$ and transform $\tilde{f}(m, n)$ are defined as the following,

$$g_{m,n}(t) = e^{i2\pi m\omega_0 t} g(t - nt_0) \tag{5}$$

$$\tilde{f}(m, n) = \int_{-\infty}^{\infty} f(t)\bar{g}_{m,n}(t)dt = (f, g_{m,n}) \tag{6}$$

It is important to note that the number of Gabor windows we use will affect our ability to localize the frequency data in time. The more increments we use, the more localized in time we are, but at a computational cost.

2.2 Different Types of Windows

Different types of filters can be used to create the Gabor windows as we are in essence, zeroing out the signal outside of the window while preserving the signal inside. As we will show later, varying the width of the filter has a profound effect on the resulting spectrograms. Because the Heisenberg relationship must hold, a wider filter will capture more frequency content but loses time resolution, while a narrow filter has greater time resolution but captures less frequency content. This is because any portion of the signal with wavelength longer than the window is not captured in the time window [1]. In this exercise we explore 3 different types of filters/windows, the standard Gaussian, the Mexican Hat Wavelet and the Step-function (Shannon) window.

2.2.1 Gaussian

The Gaussian filter used is defined as the following.

$$f(t) = e^{-a(t-\tau)^2} \tag{7}$$

It has a maximum of 1 and has a width a and is centered at $t = \tau$

2.2.2 Mexican Hat Wavelet

The Mexican Hat Wavelet is defined as the following,

$$g(t) = (1 - (t - \tau)^2)e^{-a\frac{(t-\tau)^2}{2}} \tag{8}$$

Similar to the Gaussian, a determines the width of the filter and it is centered at $t = \tau$.

2.2.3 Step-function (Shannon) Window

The Shannon window is our simplest filter as it keeps the signal as it is in the window and discards everything out side of it. It is defined as the following.

$$h(t) = \begin{cases} 1 & \tau - a \leq t \leq \tau + a \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Once again we have a width, a, and the filter is centered at $t = \tau$.

3 Algorithm Implementation and Development

Before we implement the Gabor transform, we need to determine a few key parameters. Namely, the length of the data sets (music) in seconds and the scaled wavenumbers which we will need for the spectrograms. When we import the Handel sample and the 2 recordings of MHLL, we are given the sampled data and the sample rate. We can determine the length of the song in seconds by dividing the total number of samples (the length of the sampled data vector) by the sample rate. To rescale the wavenumbers, we multiply each one by $\frac{1}{L}$ where L is the length of the music sample in seconds.

Now that we have the parameters we need, we can move on to performing a Gabor transform of the data. This is described in detail in Algorithm 1 which performs a Gabor transform for a given increment of the window in time and width. In `MATLAB` we define the time increment vector as `t = 0:increment:L`.

Algorithm 1: Gabor Transform

```
Compute the time increment vector which ranges from 0-L
for j = 1 : (length of t increment vector) do
  Define filter for a given width centered at the jth time increment
  Multiply original signal by windowing filter
  Take the FFT of windowed signal
  Store the absolute value of the FFT
end for
```

Using Algorithm 1, we can vary the type of filter, width and number of time windows we use to carry out the Gabor transform. Note that this algorithm is not used when applying the Shannon window. Rather than redefining a vector of 1s and 0s for each iteration of the loop and multiplying that with the original signal, we directly extract the pertinent samples from the original data. This was done to improve computational speed.

4 Computational Results

4.1 Handel Sample

We begin our analysis by looking at the Handel sample. Note that the number of samples refers to number of discrete time windows, which is the length of the time increment vector ($\frac{1}{L}$ where L), and that the width for the Gaussian and Mexican Hat Wavelet refers to the parameter a.

4.1.1 Varying Width

Figure 1 shows 4 spectrograms with varying width parameters. As expected, as we increase a, which decreases width of the filter, we lose more frequency content but gain better resolution in time. There is a stark difference between the spectrograms for $a = 0.5$ and $a = 10$. For $a = 0.5$, we are able to capture the higher frequencies (above 2000 Hz) but the frequencies look like they are continuous in time. We know this is not true as we are able to listen to the music sample. Conversely for $a = 10$, we do not capture higher frequencies but are able to discern individual peaks corresponding to the choir in time.

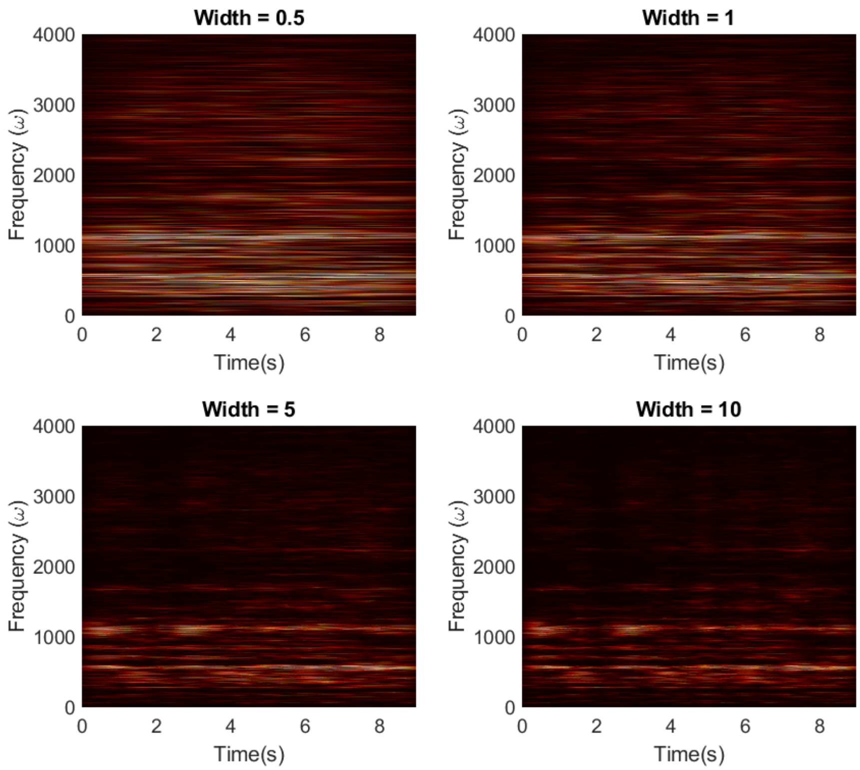


Figure 1: Varying the width of the Gaussian filter with 91 samples for Handel

4.1.2 Varying Number of Samples

Next we explore how varying the number of discretizations of the signal in time affects our spectrograms (shown in Figure 2. As expected, when we have more samples, we get a smoother spectrogram for the frequencies with distinguishable peaks and troughs. When we decrease the sample number to 10, we can still make out troughs and peaks for a given frequency but the spectrogram has a banded appearance to it due to the coarseness of the discretization. Finally, computing 10 samples was significantly faster than for 181.

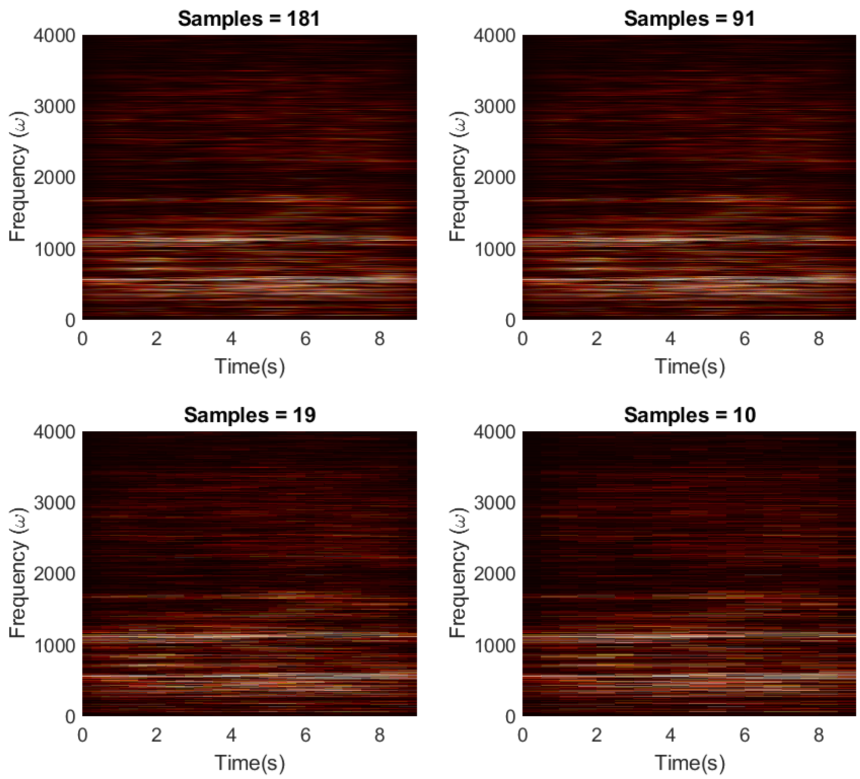
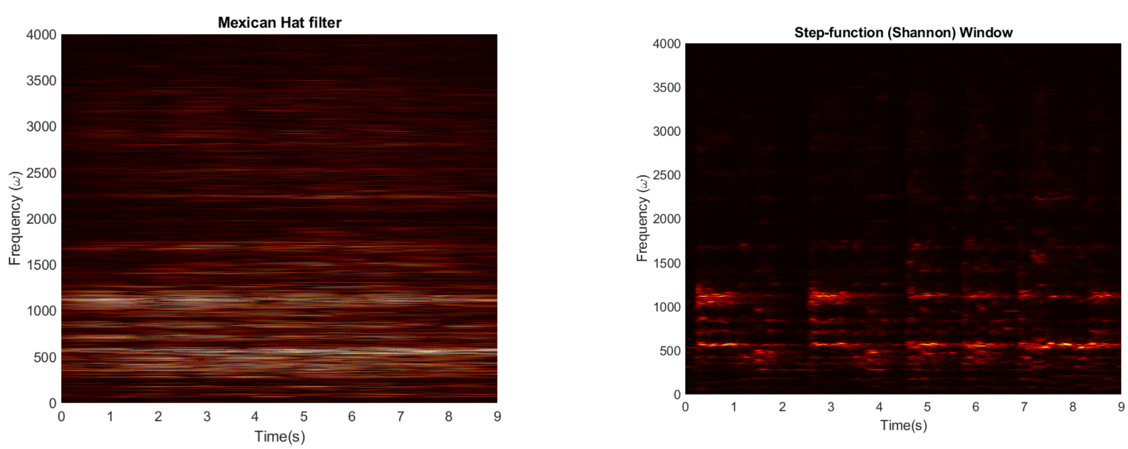


Figure 2: Varying the number of samples of the Gaussian filter with a = 1 for Handel

4.1.3 Different Filters

Finally we explore how changing the filter from a Gaussian to a step-function of hat wavelet affects our resulting spectrograms (Figure 3. For the hat wavelet, we used a = 1 and 91 samples. For the step-function, we had a width of 500 signal samples on either side and discretized the signal into 147 time windows.



(a) Mexican Hat Wavelet

(b) Step-function (Shannon) Window

Figure 3: 2 Different Filters for Handel

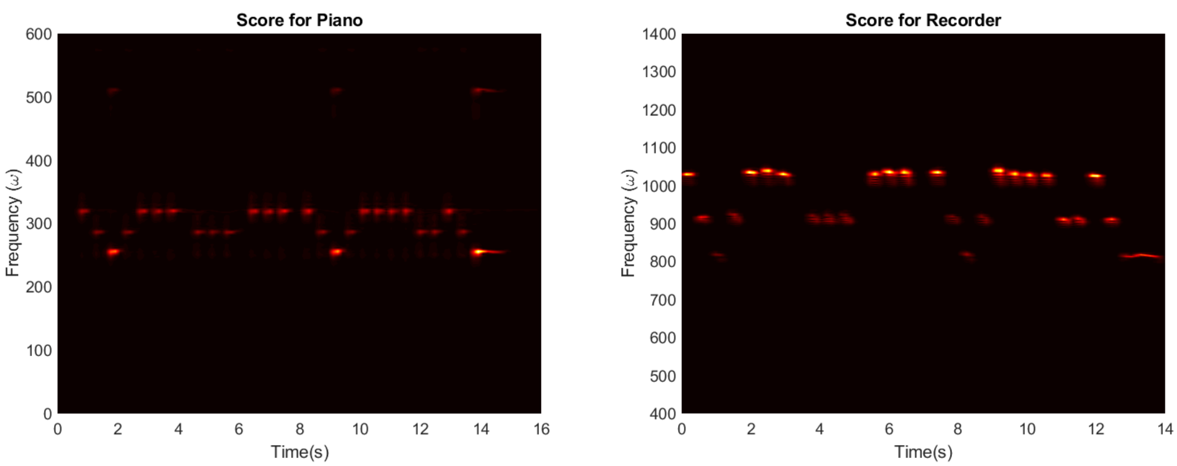
The spectrogram produced by the hat wavelet is similar to the Gaussian. The step-function on the other hand does not look like the Gaussian at all. It seems to capture less frequency content and there is banded pattern similar to the low-sample plot.

4.2 Mary had a Little Lamb

For the second part, we move to analyzing two recordings of MHLL. One with a piano and one with a recorder. We will produce a score of the music using a spectrogram and also analyze the differences in frequency content of the recorder and piano. A Gaussian filter was used for this analysis as it is easy to implement and produces good results.

4.2.1 Spectrograms

Using what we learnt from analyzing Handel, we were able to create scores for the two recordings of MHLL (Figure 4). For the recorder and the piano we used 101 samples and $a = 50$. The increment was chosen using trial and error to produce the clearest spectrogram while minimizing computational time. Since the main frequencies are quite low and we are more interested in the time resolution, a narrow width (or a large width parameter a) was chosen in order to clearly distinguish between successive notes played at the same frequency. Furthermore, having a narrow width allowed us to remove much of the overtones.



(a) Piano Score

(b) Recorder Score

Figure 4: Spectrograms for the piano and the recorder

Observing the frequency spectrogram closely, we find that the piano is rough played in $(C, D, E) = (261.63, 293.66, 329.63)$ Hz and the recorder is roughly in $(G, A, B) = (783.99, 880.00, 987.77)$ Hz. From the two figures, we are also able to see slightly more overtones in the piano score. An instrument generates overtones at integer multiples of the center frequency of the note. This will be more clear in the next part when we look at the overall frequency content of the two instruments. Furthermore, we observe that the frequencies played for each note are much more consistent for the piano than the recorder. This is most likely because when you play the piano you hit the key to produce a sound but for the recorder, there are variations due to the way you blow into the recorder.

4.2.2 Frequency Content and Overtones

Plotting the overall frequency content of two recordings (Figure 5), we can observe the key the song is played in as well as the overtones that are produced by both instruments. Comparing the two plots, we observe that the piano has significantly more overtones than the recorder which is why some find the piano 'better sounding' as it has a different timbre.

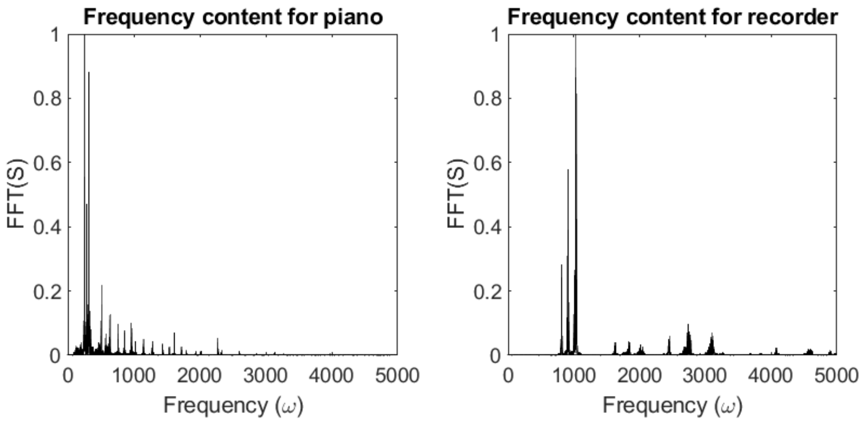


Figure 5: Frequency content of both recordings

5 Summary and Conclusions

In the first part, we were able to show the relationships between the time increment, filter width, filter type and the resulting spectrograms. Increasing width allowed more frequency content to be captured but decreased resolution in time whereas decreasing width gave better resolution in time but less frequency information. Increasing the time increment gave us better resolution in time but increased computational time.

In the second part, we were able to produce scores of the two recordings of MHLL. We determined which notes they were played on, $(C, D, E) = (261.63, 293.66, 329.63)$ for the piano, and $(G, A, B) = (783.99, 880.00, 987.77)$ for the recorder. We also showed that the piano produced more overtones.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

Appendix A MATLAB Functions

Below are the key MATLAB functions implemented.

- `fft(Vg)` returns the multidimentional Fourier transform of an n-dimensional array by using the fast Fourier Transform algorithm.
- `fftshift(Vgt)` rearranges the Fourier transform so that the zero-frequency component is at the center. Used for plotting transformed data.
- `[y, Fs]=audioread(filename)` reads data from the specific file and returns the sampled data, `y`, and the sample rate, `Fs`
- `pcolor(t,k,Vgt)` creates a pseudocolor plot (spectrogram) for time vs frequency using the values of the data after an FFT has been applied
- `colormap` sets the colormap of a figure to a predefined colormap.

Appendix B MATLAB Code

Github Repo: https://github.com/chrismhl/AMATH582/tree/master/Homework2

```
clear all; clc; close all;

load handel.mat

v = y'/2;

L=round(max(abs((1:length(v))/Fs))); n=length(v)-1;
t2=linspace(0,L,n+1); t=t2(1:n);
k=(1/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);

v = v(1:n);
%%
% Initial plots of the signal and the FFT
figure(1)
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Handel Signal, v(n)');

figure(2)
Vt = fft(v(1:n));
plot(ks,abs(fftshift(Vt))/max(abs(Vt)),'k');
set(gca,'Fontsize',[14])
title('Handel Signal after FFT');
xlabel('frequency (\omega)'), ylabel('FFT(S)')

%%
% Sliding Gabor Transform with fixed width

%figure(3)
Sgt_spec=[];
tslide=0:0.1:L;
width = [0.5,1,5,10]; %change as needed

for k = 1:length(width)
    for j=1:length(tslide)
        g=exp(-width(k)*(t-tslide(j)).^2); % Gabor
        Vg=g.*v;
        Vgt=fft(Vg);
        Sgt_spec=[Sgt_spec;
        abs(fftshift(Vgt))];
    end

    subplot(2,2,k)
    pcolor(tslide,ks,Sgt_spec.'),
    shading interp
    set(gca,'Fontsize',[14])
    ylim([0 4000])
    xlabel('Time(s)')
    ylabel('Frequency (\omega)')
    title(['Width = ' num2str(width(k))])
    colormap(hot)
    Sgt_spec=[];
```

```

end

%%
%Over and Undersampling

%figure(4)
Sgt_spec2=[];
increment = [0.05 0.1 0.5 1]; %change increment

width2 = 1; %change as needed
for k = 1:length(increment)
    tslide2=0:increment(k):L;

    for j=1:length(tslide2)
        g=exp(-width2*(t-tslide2(j)).^2); % Gabor
        Vg=g.*v;
        Vgt=fft(Vg);
        Sgt_spec2=[Sgt_spec2;
        abs(fftshift(Vgt))];
    end

    subplot(2,2,k)
    pcolor(tslide2,ks,Sgt_spec2.'),
    shading interp
    set(gca,'Fontsize',[14])
    ylim([0 4000])
    xlabel('Time(s)')
    ylabel('Frequency (\omega)')
    title(['Samples = ' num2str(length(tslide2))])
    colormap(hot)
    Sgt_spec2=[];
end

%%
% Mexican hat wavelet

Sgt_spec3=[];
tslide3=0:0.1:L;
widthmh = 1;  %change as needed

for j=1:length(tslide3)
    mhat = (1-(t-tslide3(j)).^2).*exp(-widthmh*(t-tslide3(j)).^2);
    Vm=mhat.*v;
    Vmt=fft(Vm);
    Sgt_spec3=[Sgt_spec3;
    abs(fftshift(Vmt))];
end

%%
% Plotting spectrogram for mexican hat wavelet

figure(7)
pcolor(tslide3,ks,Sgt_spec3.'),
```

```
shading interp
set(gca,'Fontsize',[14])
ylim([0 4000])
xlabel('Time(s)')
ylabel('Frequency (\omega)')
title('Mexican Hat filter')
colormap(hot)

%%
% Sliding step-function

widthshn = 500;
m = n; %lenght of the filter
step = 500;
Sgt_spec4=[];
Vs=zeros(n,1);

for j = 1:step:m
    if j < widthshn+1
        Vs = zeros(m,1);
        Vs(j:1:j+widthshn) = v(j:1:j+widthshn);

        if j>1
        Vs(1:j) = v(1:j);
        end

    elseif (j+widthshn) >= m
        Vs = zeros(m,1);
        Vs(j - widthshn:1:m) = v(j - widthshn:1:m);
    else
        Vs = zeros(m,1);
        Vs(j - widthshn:1:j+widthshn) = v(j - widthshn:1:j+widthshn);
    end

    Vst=fft(Vs);
    Sgt_spec4=[Sgt_spec4;abs(fftshift(Vst.'))];
end

%%
% Plotting spectrogram for shannon window

figure(8)
tslide4=linspace(0,9,length(1:step:m));
pcolor(tslide4,ks,Sgt_spec4.'),
shading interp
set(gca,'Fontsize',[14])
ylim([0 4000])
xlabel('Time(s)')
ylabel('Frequency (\omega)')
title('Step-function (Shannon) Window')
colormap(hot)

%%
% Part 2
```

```
clc; clear all; close all;

figure(9)
tr_piano=16; % record time in seconds
y2=audioread('music1.wav'); Fs2=length(y2)/tr_piano;
plot((1:length(y2))/Fs2,y2);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (piano)'); drawnow

figure(10)
tr_rec=14; % record time in seconds
y3=audioread('music2.wav'); Fs3=length(y3)/tr_rec;
plot((1:length(y3))/Fs3,y3);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)');

%%
% Frequencies for the piano overtones.

n=length(y2);
t2=linspace(0,tr_piano,n+1); tp=t2(1:n);
kp=(1/tr_piano)*[0:n/2-1 -n/2:-1];
ksp=fftshift(kp);

subplot(1,2,1)
y2t = fft(y2(1:n));
plot(ksp,abs(fftshift(y2t))/max(abs(y2t)),'k');
xlim([0 5000])
title('Frequency content for piano')
%xlim([220 350])
set(gca,'Fontsize',[14])
xlabel('Frequency (\omega)'), ylabel('FFT(S)')

% Frequencies for the recorder overtones.

n=length(y3);
t2=linspace(0,tr_rec,n+1); tr=t2(1:n);
kr=(1/tr_rec)*[0:n/2-1 -n/2:-1];
ksr=fftshift(kr);

subplot(1,2,2)
y3t = fft(y3(1:n));
plot(ksr,abs(fftshift(y3t))/max(abs(y3t)),'k');
xlim([0 5e3])
title('Frequency content for recorder')
set(gca,'Fontsize',[14])
xlabel('Frequency (\omega)'), ylabel('FFT(S)')
%%
%Gabor filter for piano

piano_spec=[];
incr = 0.2;
tslide_p=0:incr:tr_piano;
width = 25; %change as needed
```

```
for j=1:length(tslide_p)
    g=exp(-width*(tp-tslide_p(j)).^2); % Gabor
    Pg=g.*y2.';
    Pgt=fft(Pg);
    piano_spec=[piano_spec;
    abs(fftshift(Pgt))];
end

%%
%Spectrogram for piano
figure(11)

pcolor(tslide_p,ksp,(piano_spec.'/max(max(abs(piano_spec)))))
shading interp
set(gca,'Fontsize',[14])
ylim([0 1000])
title('Score for Piano')
xlabel('Time(s)')
ylabel('Frequency (\omega)')
colormap(hot)
%%
%Gabor filter for recorder

rec_spec=[];
incr = 0.1;
tslide_r=0:incr:tr_rec;
width = 20; %change as needed

for j=1:length(tslide_r)
    g=exp(-width*(tr-tslide_r(j)).^2); % Gabor
    Rg=g.*y3.';
    Rgt=fft(Rg);
    rec_spec=[rec_spec;
    abs(fftshift(Rgt))];
end

%%
%Spectrogram for recorder
figure(12)

pcolor(tslide_r,ksr,(rec_spec.'/max(max(abs(rec_spec)))))
shading interp
set(gca,'Fontsize',[14])
ylim([0 2e3])
xlabel('Time(s)')
ylabel('Frequency (\omega)')
title('Score for Recorder')
colormap(hot)
```