AMATH 582 Homework 2

Reed Nomura

February 7, 2020

Abstract

This article explores the use of windowed Fourier transforms to visualize audio data through spectograms. This article will be split into two parts. In the first part we will be exploring a nine second clip of Handel's Messiah. We will be adjusting several variables to see how they change the resulting spectogram. In the second part, we will be using the methods explored in part 1 in order to compare two clips of Mary Had a Little Lamb.

1 Introduction and Overview

Fourier transforms provide an insight into the frequency data hidden within a data set. Unfortunately, by transforming data into the Fourier domain, the time dimension is lost. We can use Windowed Fourier transforms to reclaim the time dimension lost in Fourier transforms. We will be using windowed Fourier transforms to analyze three audio clips. In part one we will be looking at a nine second clip of Handel's Messiah. In order to understand how windowed Fourier transforms function, we will adjust a plethora of variables to see how each affect the resulting spectograms. In part two, we will use what we learned in part one to analyze two clips of Mary Had a Little Lamb. In the end we will be able to reproduce the score for these clips and we will know a little more about the overtones present in each clip.

2 Theoretical Background

2.1 Fourier Series

Fourier introduced the concept of representing a given function $f(x)$ by a trigonometric series of sines and cosines: [1]

$$f(x) = \frac{a_0}{2} + \sum_{i=1}^{\infty} (a_n \cos nx + b_n \sin nx) \quad x \in (-\pi, \pi]. \tag{1}$$

Through some basic mathematic manipulation, we are able to produce formulas for the Fourier coefficients.

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos nx dx \tag{2}$$

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin nx dx \tag{3}$$

The complex version of the expansion produces the Fourier series on the domain $x \in [-L, L]$ which is given by

$$f(x) = \sum_{-\infty}^{\infty} c_n e^{in\pi x/L} \quad x \in [-L, L]. \tag{4}$$

With the following Fourier coefficient.

$$c_n = \frac{1}{2L} \int_{-L}^{L} f(x) e^{in\pi x/L} dx \tag{5}$$

2.2 Fourier Transform

The Fourier Transform is an integral transform defined over the entire line $x \in [-\infty, \infty]$.[1] The Fourier transform is defined as

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx \tag{6}$$

Thus the Fourier transform is essentially an eigenfunction expansion over all continuous wavenumbers k. And once we are on a finite domain $x \in [-L, L]$, the continuous eigenfunction expansion becomes a discrete sum of eigenfunctions and associated wavenumbers (eigenvalues)[1]. This concept has widespread applications. For the purposes of this paper, Fourier transforms will be used to transform data within specified windows in order to view frequency data in the form of a spectogram.

2.3 Gabor Transforms

Fourier transforms are very useful for capturing frequency information within a sample. However, they are limited in terms of providing information about when those frequencies occurred. Gábor transforms take Fourier transformations of small windows that are translated across the signal. Gábor windows are also referred to as Short Term Fourier Transforms or (STFT)s. In this paper we will be using three different filters to produce Gábor transforms.

2.4 Gaussian

$$g(t) = e^{-a(t-b)^2} \tag{7}$$

This is the first filter at which we will be looking. Additionally, it will be the filter with which we will be predominately experimenting in this paper.

2.5 Mexican Hat Wavelet

$$\psi(t) = 1 - \left(\frac{t}{a}\right)^2 e^{-(t/a)^2/2} \tag{8}$$

This will be one of the Gábor filters we will translate over our sound clip to view the resulting spectogram.

2.6 Shannon

$$s(t) = \begin{cases} 0 & |t| > \frac{a}{2} \\ 1 & |t| \leq \frac{a}{2} \end{cases} \tag{9}$$

This is the final Gábor filter we will translate over our sound clip to view the resulting spectogram. This window has many applications, especially when it comes to viewing sharp edges of photographs.

3 Algorithm Implementation and Development

3.1 Outline Part 1

1. Load Song
2. Create Filters
3. Set parameters for Windowed transform
4. Add windowed Fourier transform to song data
5. Create Spectogram

3.2 Outline Part 2

1. Load Songs
2. Create Filter
3. Set parameters for Windowed transform
4. Add windowed Fourier transforms to songs' data
5. Create Spectogram of the log of the transformed matrices
6. Record and Plot the data

3.3 Algorithms Part 1

Algorithm 1: Applying Gábor Window

```
Import data
Design Filter
Divide the clip by the number of intervals (τ) over which the window will be translated
Set Window Width
for t = 1 : τ do
  Apply Windowed Filter at t
  Perform FFT of Filtered Window
  Store Transformations in a Matrix
end for
Create Spectogram of the Matrix
```

3.4 Algorithms Part 2

Algorithm 2: Finding Central Frequencies

Import data
Design Filter
Divide the clip by the number of intervals(τ) over which the window will be translated
Set Window Width
for $t = 1 : \tau$ **do**
  Apply Windowed Filter at t
  Perform FFT of Filtered Window
  Store Transformations in a Matrix
end for
Create Spectogram of log of the the Matrix
Zoom to find central frequencies

4 Computational Results

4.1 Part 1

The first step in analyzing Handel's Messiah was to look examine how the width of the Gábor window affects the resulting spectogram. The first table has three snapshots of a Gaussian Gábor filter on Handel's Messiah with varying widths. The first has a width value of $a = 1$, the second has a width value of $a = 10$, and the third has a width value of $a = 100$ found in equation 7.

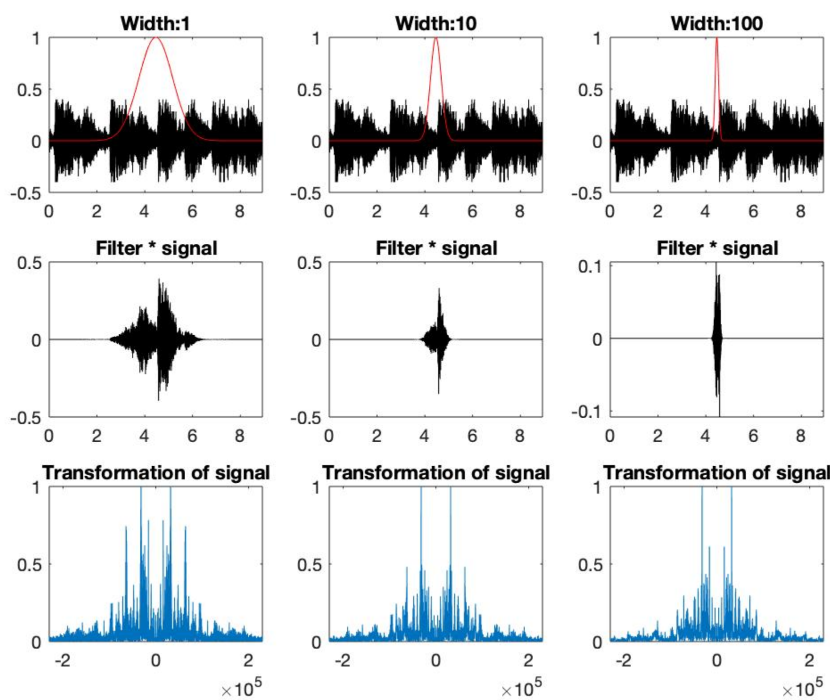


Figure 1: Gaussian Filter snapshots with varying widths

By adjusting the width of the Gausian filters, we can see a distinct difference in resolution for each example. Each of these windows was translated over the music clip over 100 intervals and their resulting spectograms are shown in the table below. As the width increases from 1 to 100, there is a growing localization of the signal in time. However, as the time resolution increases, we begin to loose resolution in the Fourier domain.

Width: 1 | Width: 10 | Width: 100

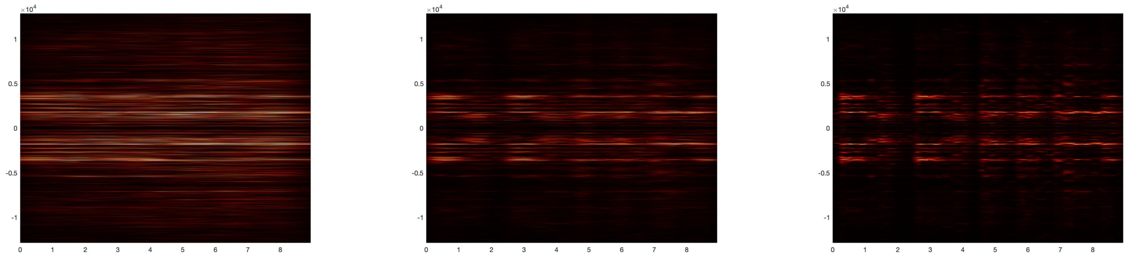


Table 1: Gaussian filter spectograms with varying window widths taken over 100 translations

We can also see a difference in resolution when we adjust the number of translations over which we move the windows. In the following table each spectogram is the result of translating a Gaussian window with a width value of $a = 100$ over the sound clip. The leftmost spectogram is the result of translating over 10 intervals, the middle is translated over 50 intervals, and the last spectogram is translated over 1000 intervals.

Intervals: 10 | Intervals: 50 | Intervals: 1000

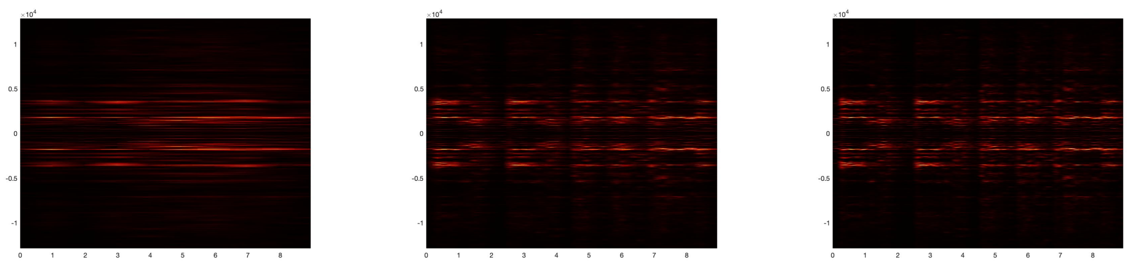


Table 2: Gaussian filter spectograms taken with width of 100

In the case of the leftmost spectogram, we are experiencing severe under sampling. As a result of our large and course translations, we are loosing some information in our spectogram. One might expect that too many translation intervals might cause noise in the spectogram. However, the only issue that arises from oversampling is the cost in time it takes to compute the Fourier Transform of each translation.

In addition to modifying the width of the window and the number of intervals over which our windows are translated, we are able to exchange the type of window we use to analyze the data. In this article we looked primarily at Gaussian window. However, in the following figure we will be looking at the Mexican Hat wavelet and the Shannon filter side-by-side with the Gaussian filter.

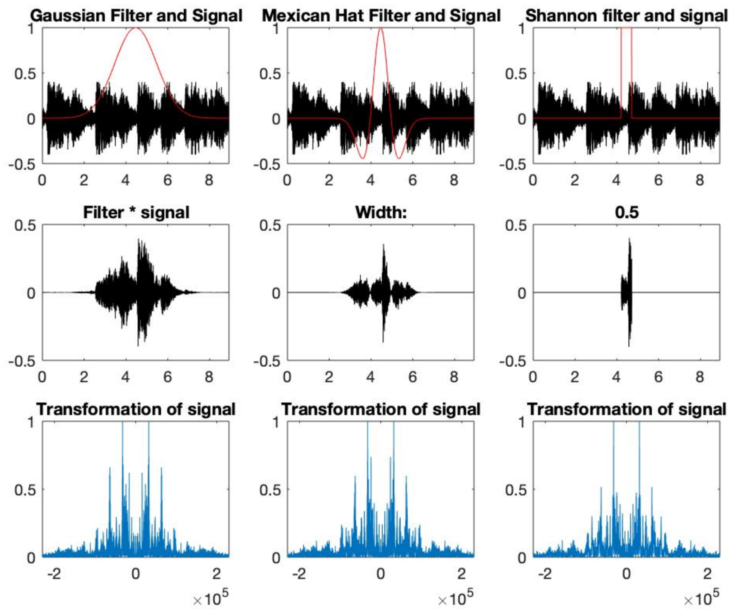


Figure 2: Snapshot of all three filters with a width of 0.5

In the figure above we have three snapshots of windowed Fourier transforms. Each filter has a width value of $a = 0.5$. On the left there is a Gaussian filter, in the middle is a Mexican Hat wavelet, and on the right there is a Shannon filter. Each filter isolates different parts of the signal to be filtered. Each window is translated over the signal over 100 intervals and their spectogram is pictured below. The spectogram follows in the same order as Figure 2.

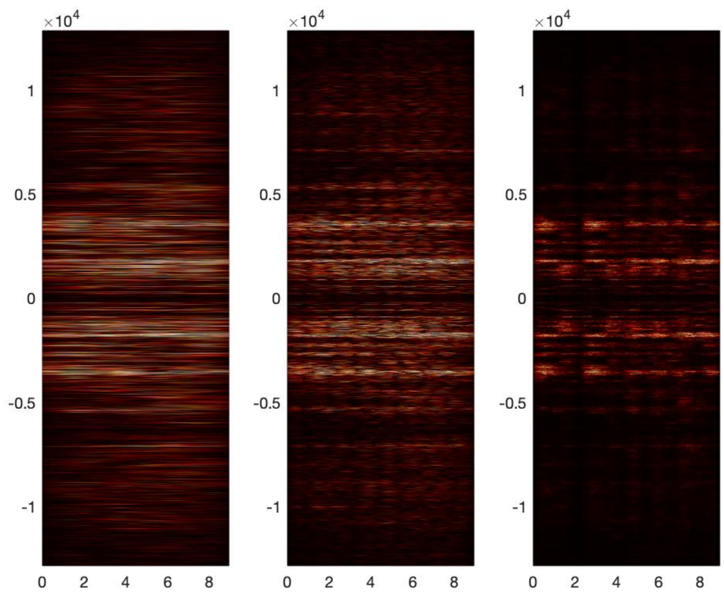


Figure 3: Spectogram of All three filters taken over 100 intervals with a width of .5

With this width, the Shannon filter produces the clearest spectogram. However at other widths, the other two filters produce higher resolution spectograms. In the table below I have included a glimpse into the roll that width plays on the different filters.

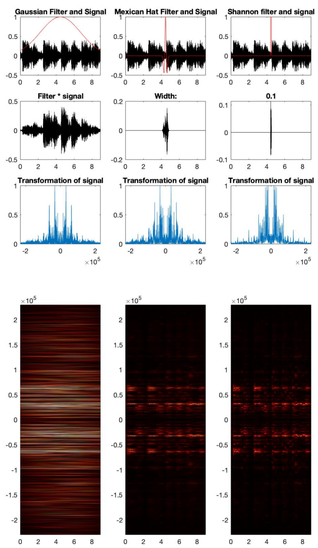
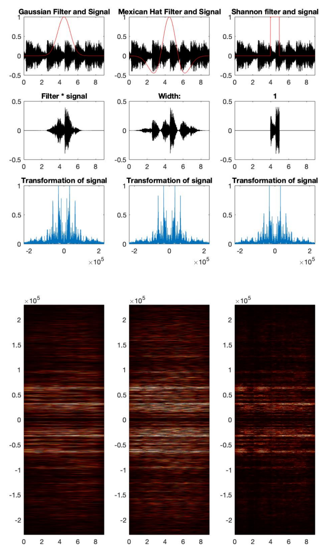
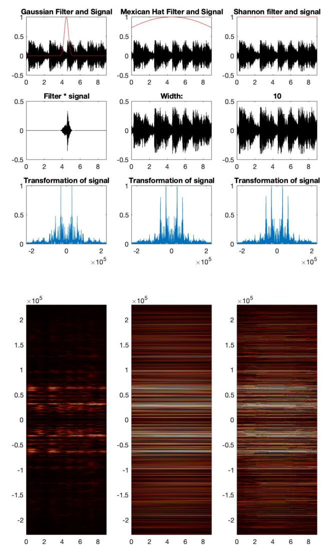
Width:0.1	Width: 1	Width: 10
		

Table 3: Snapshot of all Filters and their spectograms with varying widths

Each filter has a specific set of circumstances for which they are best suited. However, in all cases we see the same inverse relationship between frequency and time resolution.

4.2 Part 2

By using the methods outlined in Part 1, we are able to analyze two clips of Mary Had a Little Lamb. The table below contains two figures. Both figures include the audio clip of Mary Had a Little Lamb and then the unfiltered Fourier transform of the clip. The figure on the left is the piano version while the figure on the right is the recorder version.

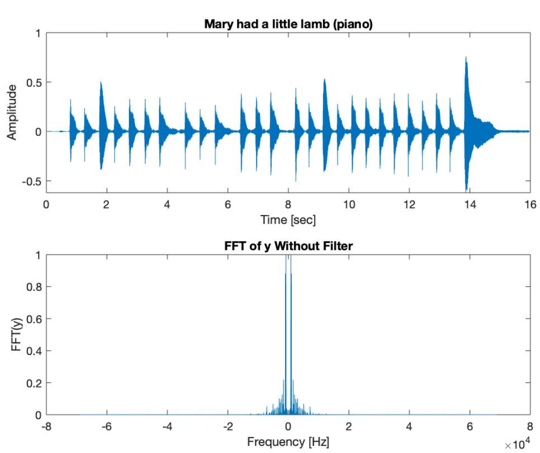
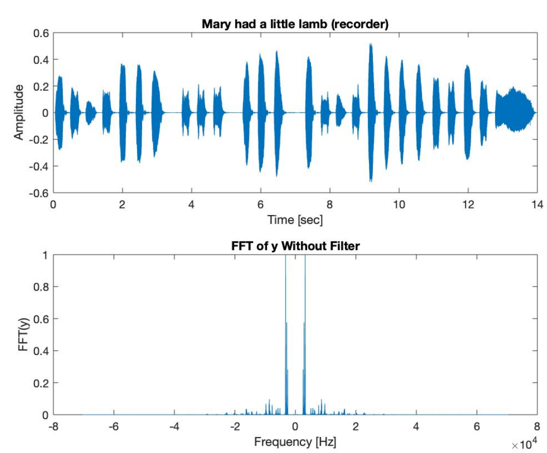
Piano	Recorder
	

Table 4: Unfiltered Fourier Transform of Mary Hand a Little Lamb

A Gaussian windowed Fourier transform was applied to both clips using a width of 100 translated over 100 intervals. Two spectograms were created of these two audio clips by taking the log of this data.

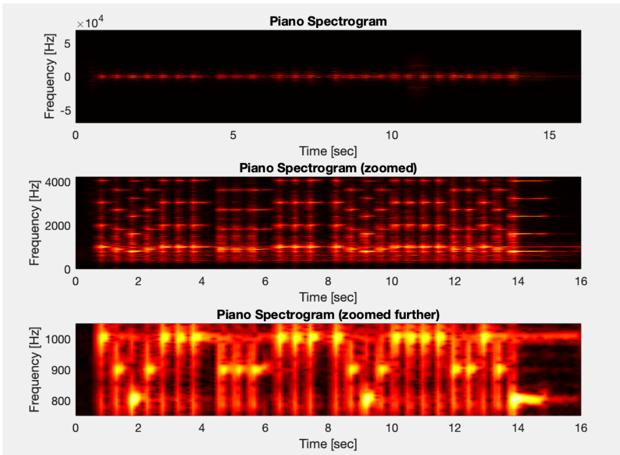
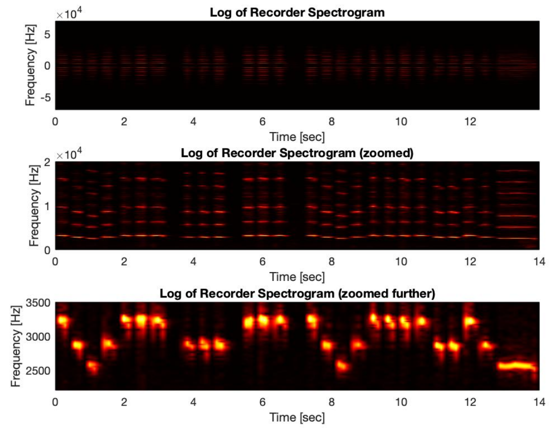
Piano	Recorder
	

Table 5: Spectograms of the Gaussian windowed Fourier transforms with a width of 100 taken over 100 translations

In both of these sound clips we can see that there are three clear notes being played through them. By by zooming in to each individual note, we can see that these central frequencies present in piano clip are 987.77 Hz, 880 Hz, and 783.99 Hz. These frequencies correspond with B5, A5, and G5 respectively. In the recorder clip the frequencies present are 3322.4 Hz, 2960 Hz, and 2637 Hz. These frequencies correspond with G7, F7, and E7 respectively. These notes were easily verified with a keyboard while listening to the clips. The table below shows the musical scores for the piano and recorder pieces.

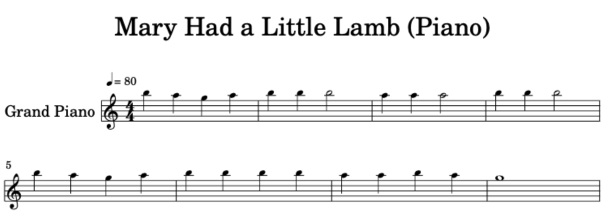
Piano	Recorder
	

Table 6: Music Scores for the piano and recorder clips

In addition to finding the notes based on central frequency. There are some overtones present in each of the sound clips. For ω frequencies found in the piano piece, I can see clear overtones up to 4ω. While, in the recorder piece, I can see overtones up to about 5ω. Additionally, you can see some illumination around each of the notes more in the piano piece than in the recorder piece. In a traditional piano, the sound is created by hammering on strings that vibrate at particular frequencies. Whenever a note is played, harmonic tones are produces as well by vibrating at harmonic frequencies and resonating relevant strings. Digital pianos mimic this sound which is why, even with the same settings, the piano clip appears muddled more with information other than the played central frequencies.

5 Summary and Conclusions

5.1 Part 1

In the first part of this paper we discovered the way that changing the width of a Gábor window, the number of intervals over which the window is translated, and the filter used to create the window will change the way that the resulting spectogram will look. Most notably we saw that as the time resolution increased, frequency resolution decreased. In order to effectively use Gábor windows to analyze data, it is vital to test the settings and take several passes at the data. Some attempts will yield higher frequency resolution and some will provide higher time resolution. However, all of them will help to create a clearer picture of the data being analyzed.

5.2 Part 2

In the second part of this paper we use the tools we learned in part one to analyze two audio clips of Mary Had a Little Lamb. One clip was a piano clip while the other was from a recorder. Through the use of a Gaussian Gábor filter, we were able to create a spectogram for each of these clips. The spectograms showed the central frequencies and the time in which they occurred in the clips. From this we were able to recreate the musical score for each of the two clips.

References

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

[2] *MathWorks Website*. URL: `https://www.mathworks.com/help/matlab/index.html`.

Appendix A MATLAB Functions

- `playblocking(playerObj)`: plays the audio associated with audioplayer object playerObj from beginning to end. playblocking does not return control until playback completes. [2]
- `Y = fft(X)` computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm. [2]
- Y = fftshift(X): rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array. [2]
- `y = linspace(x1,x2,n)`: generates n points. The spacing between the points is $\frac{x2-x1}{n-1}$ [2]
- `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. [2]
- `colormap(map)`: sets the colormap for the current figure to the colormap specified by map. [2]

Appendix B MATLAB Code

This code can be found at: https://github.com/ReedNomura/AMATH-582/blob/master/Homework2.m

```
% AMATH 582 Homework 2
%% Part 1 Handel's Messiah

clear all; close all; clc; %Start Fresh
```

```
load handel % Load Sound Clip

%Set up Parameters
v = y'/2;

figure() %Unfiltered Handel's Messiah

subplot(2,1,1) % Plotting Handel's Messiah
plot((1:length(v))/Fs,v)
  xlabel('Time [sec]');
  ylabel('Amplitude');
  title('Signal of Interest, v(n)');


 % Set up Parameters for FFT
 v = v(1:end-1); %Parse v
 L = (length(v)-1) / Fs; %Time Domain
 n = length(v); % Amount of time (to calculate frequency conversion)
 vt = fft(v); % Fourier Transform of Handel's Messiah
 k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; % Frequency conversion
 ks=fftshift(k);

subplot(2,1,2) % Plotting FFT of Handel's Messiah
 plot(ks, abs(fftshift(vt)) / max(abs(vt)));
 xlabel('Frequency (k)')
 ylabel('FFT(v)')
 title('FFT of v Without Filter')


 %% Play Sound Clip (Handel's Messiah)
    %p8 = audioplayer(v,Fs);
    %playblocking(p8);

%% Filters
% Configure Parameters
 width = 1000;          % Width of filter
 intervals = 100;        % Number of time intervals
 t1 = (0:length(v))/Fs;
 t = t1(1:end-1);
 tslide = linspace(0,t(end-1),intervals);             % Time discretization
 spec = zeros(length(tslide),length(v));             % Preallocate space for spectrogram
 spec_g = zeros(length(tslide),length(v));
 spec_m = zeros(length(tslide),length(v));
 spec_s = zeros(length(tslide),length(v));

% Translate Filter accross intervals
figure()
 for j=1:length(tslide)

     g = exp(-width*(t-tslide(j)).^2); %Gaussian Filter
     m = (1-((t-tslide(j))/width).^2).*exp(-(((t-tslide(j))/width).^2)/2); %Mexican Hat Filter
     s = ((t-tslide(j))>-width/2 & (t-tslide(j))< width/2); %Shannon Filter
     filter = g ; %Pick the filter {g, m, s}
     vf = filter.*v; %Apply Filter
     vft = fft(vf); %Fourier Transform of Filtered
     spec(j,:) = abs(fftshift(vft));             % Store fft in spectrogram

     %Store for all Filters
     vgf_spec = g.*v; %Apply Gausian Filter
     vgft_spec = fft(vgf_spec); %Take Fourier Transform with filter
     spec_g(j,:) = abs(fftshift(vgft_spec));   % Store fft in spectrogram

     vmf_spec = m.*v; %Apply Mexican Hat Filter
     vmft_spec = fft(vmf_spec); %Take Fourier Transform with filter
     spec_m(j,:) = abs(fftshift(vmft_spec));    % Store fft in spectrogram

     vsf_spec = s.*v; %Apply Mexican Hat Filter
```

```
    vsft_spec = fft(vsf_spec); %Take Fourier Transform with filter
    spec_s(j,:) = abs(fftshift(vsft_spec));   % Store fft in spectrogram

    % Annimation of the filter moving accross the signal
    subplot(3,1,1), plot(t,v,'k',t,filter,'r'), title('Gabor filter and signal'), ...
        legend('v','Gabor filter')
    subplot(3,1,2), plot(t,vf,'k'), title('Gabor filter * signal')
    subplot(3,1,3), plot(ks, abs(fftshift(vft))/max(abs(vft))), title('Gabor ...
        transformation of signal')
    drawnow

 end
% Plot spectrogram
 figure()
 pcolor(tslide,ks,spec.'), shading interp
 colormap('hot')

  %% Spectograms for Three Filter Types (Gaussian, Mexican Hat, and Shannon)
  %Setup
  figure()
     subplot(1,3,1),
     pcolor(tslide,ks,spec_g.'), shading interp
     colormap('hot')

     subplot(1,3,2),
     pcolor(tslide,ks,spec_m.'), shading interp
     colormap('hot')

     subplot (1,3,3),
     pcolor(tslide,ks,spec_s.'), shading interp
     colormap('hot')

 %% All in one image Different Filters

v = y'/2;
n = length(v);
t = (1:length(v))/Fs;
L = max(t);
k = (2*pi)*[0:n/2 -n/2:-1];
ks = fftshift(k);
midpoint = L / 2;
center = midpoint;

figure()

%width = .5; % Fixed Width (comment out: for, moving width, and end)

intervals = 100;        % Number of width interval
wslide = linspace(.1,10,intervals);             % width discretization

%for j=1:length(wslide)
%width = wslide(j); %moving width

   %Gausian
     filter_g = exp(-width*((t-center).^2)); %Gaussian Filter
     vgf = filter_g.*v; %Apply Gausian Filter
     vgft = fft(vgf); %Take Fourier Transform with filter

     subplot(3,3,1), plot(t,v,'k',t,filter_g,'r'), title('Gaussian Filter and Signal'), ...
         xlim([0, L])
     subplot(3,3,4), plot(t,vgf,'k'), title('Filter * signal'), xlim([0, L])
     subplot(3,3,7), plot(ks, abs(fftshift(vgft))/max(abs(vgft))), title('Transformation ...
         of signal')

   % Mexican Hat

    filter_m = (1-((t-center)/width).^2).*exp(-(((t-center)/width).^2)/2); %Mexican Hat Filter
    vmf = filter_m.*v; %Apply Mexican Hat Filter
```

```
    vmft = fft(vmf); %Take Fourier Transform with filter

     subplot(3,3,2), plot(t,v,'k',t,filter_m,'r'), title('Mexican Hat Filter and Signal'), ...
         xlim([0, t(end)])
     subplot(3,3,5), plot(t,vmf,'k'), title('Width:'), xlim([0, L])
     subplot(3,3,8), plot(ks, abs(fftshift(vmft))/max(abs(vmft))), title('Transformation ...
         of signal')


 % Shannon Filter

     filter_s = ((t-center)>-width/2 & (t-center)< width/2); %Shannon Filter
     vsf = filter_s.*v; %Apply Shannon Filter
     vsft = fft(vsf); %Take Fourier Transform with filter

     subplot(3,3,3), plot(t,v,'k',t,filter_s,'r'), title('Shannon filter and signal'), ...
         xlim([0, t(end)])
     subplot(3,3,6), plot(t,vsf,'k'), title(width), xlim([0, L])
     subplot(3,3,9), plot(ks, abs(fftshift(vsft))/max(abs(vsft))), title('Transformation ...
         of signal')
     drawnow
%end



%% Part 2 Mary Had a Little Lamb

% Fourier Transform see the notes but not when
% Filter based on those notes to create clean spectogram
% look at how the overtones
clear all , close all, clc;

%Piano
figure()
tr_piano = 16;  % record time in seconds
 y = audioread('music1.wav');
 Fs=length(y)/tr_piano;

 subplot (2,1,1)
 plot((1:length(y))/Fs,y);
 xlabel('Time [sec]'); ylabel('Amplitude');
 title('Mary had a little lamb (piano)');  drawnow

 % Set up Parameters for FFT
 y = y.'; %Parse y
 L = tr_piano; %Time Domain
 n = length(y);
 yt = fft(y); % Fourier Transform of Mary Had a Little Lamb (Piano)
 k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; % Frequency Reframe
 ks=fftshift(k);

 subplot(2,1,2) % Plotting FFT Mary Had a Little Lamb (Piano)
 plot(ks, abs(fftshift(yt)) / max(abs(yt)));
 xlabel('Frequency [Hz]')
 ylabel('FFT(y)')
 title('FFT of y Without Filter')

 %Play Mary Had a Little Lamb (Piano)
 %p8 = audioplayer(y,Fs); playblocking(p8);

% Set up for spectograms

t1 = linspace(0,L,n+1);
t = t1(1:end-1);
width = 1000;
intervals = 200;
tslide = linspace(0, t(end-1), intervals);
spec = zeros(length(tslide),n);
```

```
%% Filters
figure()
   for j = 1:length(tslide)
     g = exp(-width*(t-tslide(j)).^2); %Gaussian Filter
     m = (1-((t-tslide(j))/width).^2).*exp(-(((t-tslide(j))/width).^2)/2); %Mexican Hat Filter
     s = ((t-tslide(j))>-width/2 & (t-tslide(j))< width/2); %Shannon Filter
     filter = g ; %Pick the filter {g, m, s}
     yf = filter.*y; %Apply Filter
     yft = fft(yf); %Fourier Transform of Filtered
     spec(j,:) = abs(fftshift(yft));    % Store fft in spectrogram

    % Plot
    subplot(3,1,1), plot(t,y,'k',t,filter,'r'), title('Gabor filter and signal'), ...
        legend('y','Gabor filter')
    xlabel('Time [sec]'), ylabel('Amplitude')
    subplot(3,1,2), plot(t,yf,'k'), title('Gabor filter * signal')
    xlabel('Time [sec]'), ylabel('Amplitude')
    subplot(3,1,3), plot(ks, abs(fftshift(yft))/max(abs(yft))), title('Gabor ...
        transformation of signal')
    xlabel('Frequency [Hz]'), ylabel('Magnitude')
    drawnow

end
%% Plot full spectrogram
subplot(2,1,1)
pcolor(tslide,ks, log(spec.'+1)), shading interp
colormap('hot'), xlabel('Time [sec]'), ylabel('Frequency [Hz]'), title('Log of Piano ...
    Spectrogram')

% Plot Zoom of spectrogram
subplot (2,1,2)
pcolor(tslide,ks,log(spec.'+1)), shading interp
axis([0 16 1500 2100])
colormap('hot'), xlabel('Time [sec]'), ylabel('Frequency [Hz]'), title('Log of Piano ...
    Spectrogram (zoomed)')





%% Recorder
  figure()
  tr_rec=14;  % record time in seconds
  y = audioread('music2.wav');
  Fs = length(y)/tr_rec;
  subplot(2,1,1);
  plot((1:length(y))/Fs,y);
  xlabel('Time [sec]'); ylabel('Amplitude');
  title('Mary had a little lamb (recorder)');

 % Set up Parameters for FFT
 y = y.'; %Parse y
 L = tr_rec; %Time Domain
 n = length(y);
 yt = fft(y); % Fourier Transform of Mary Had a Little Lamb (Recorder)
 k=(2*pi/(2*L))*[0:(n/2-1) -n/2:-1]; % Frequency Reframe
 ks=fftshift(k);

 subplot(2,1,2) % Plotting FFT Mary Had a Little Lamb (Recorder)
 plot(ks, abs(fftshift(yt)) / max(abs(yt)));
 xlabel('Frequency [Hz]')
 ylabel('FFT(y)')
 title('FFT of y Without Filter')

 %Play Mary Had a Little Lamb (Recorder)
 %p8 = audioplayer(y,Fs); playblocking(p8);

```

```
% Set up for spectograms

t1 = linspace(0,L,n+1);
t = t1(1:end-1);
width = 1000;
intervals = 100;
tslide = linspace(0, t(end-1), intervals);
spec = zeros(length(tslide),n);

%%Filters
figure()
   for j = 1:length(tslide)
     g = exp(-width*(t-tslide(j)).^2); %Gaussian Filter
     m = (1-((t-tslide(j))/width).^2).*exp(-(((t-tslide(j))/width).^2)/2); %Mexican Hat Filter
     s = ((t-tslide(j))>-width/2 & (t-tslide(j))< width/2); %Shannon Filter
     filter = g ; %Pick the filter {g, m, s}
     yf = filter.*y; %Apply Filter
     yft = fft(yf); %Fourier Transform of Filtered
     spec(j,:) = abs(fftshift(yft));    % Store fft in spectrogram

    % Plot Data
    subplot(3,1,1), plot(t,y,'k',t,filter,'r'), title('Gabor filter and signal'), ...
        legend('y','Gabor filter')
    xlabel('Time [sec]'), ylabel('Amplitude')
    subplot(3,1,2), plot(t,yf,'k'), title('Gabor filter * signal')
    xlabel('Time [sec]'), ylabel('Amplitude')
    subplot(3,1,3), plot(ks, abs(fftshift(yft))/max(abs(yft))), title('Gabor ...
        transformation of signal')
    xlabel('Frequency [Hz]'), ylabel('Magnitude')
    drawnow

   end

%% Plot full spectrogram
figure()
subplot (3,1,1)
pcolor(tslide,ks, log(spec.'+1)), shading interp
colormap('hot'), xlabel('Time [sec]'), ylabel('Frequency [Hz]'), title('Log of Recorder ...
    Spectrogram')

% Plot Zoom of spectrogram
subplot (3,1,2)
pcolor(tslide,ks,log(spec.'+1)), shading interp
axis([0 14 0 20000])
colormap('hot'), xlabel('Time [sec]'), ylabel('Frequency [Hz]'), title('Log of Recorder ...
    Spectrogram (zoomed)')

% Plot Zoom of spectrogram
subplot (3,1,3)
pcolor(tslide,ks,log(spec.'+1)), shading interp
axis([0 14 2200 3500])
colormap('hot'), xlabel('Time [sec]'), ylabel('Frequency [Hz]'), title('Log of Recorder ...
    Spectrogram (zoomed further)')
```