AMATH 582: Homework 2

Hyun Ah Lim

February 7, 2020

Abstract

The Gabor Method was used to analyze three different audio files using MATLAB. This method requires one to implement a 'best fit' signal filtering iteration in the time domain and the Fourier Transform routine. The combination of these two actions, produce data for a spectrogram (a visualization representing frequency and time properties). The three files analyzed were MATLAB's Handel's Messiah and two versions of the children's song Mary had a little lamb (piano vs. recorder). Through the analysis of Handel's Messiah, rich in frequency content, the tradeoff between time resolution and frequency resolution was explored as an example of the Heisenberg uncertainty principle. The choice in filter was also explored using Gaussian, Mexican Hat and Shannon Step filters. The lessons learned in how filter choice and associated parameters of scale (window width) and translation (time step size) impact the resulting spectrogram were used to produce spectrograms on two versions of Mary had a little lamb, that resembled a musical score (where notes and their duration could be resolved). The spectrogram also provided a visual indicator of the difference in sound quality of the two instruments.

1 Introduction and Overview

Three audio signals were provided: Handel's Messiah, Mary had a Little Lamb (piano) and Mary had a Little Lamb (recorder). The challenge was to produce spectrograms, which are visualizations that provide frequency information in relation to time, using the Gabor Method. This method combines the Fourier Transform and signal filtering along a discretized path in the temporal domain. The first piece, Handel's Messiah is a complex audio file and therefore contains equally complex frequency components. This audio file was used to explore the impacts of the filter selected and the two parameter values used in its implementation: scale (filter width) and translation (step size for time discretization). We will see how they illustrate the trade-off between the resolution in time and frequency in accordance with the Heisenberg uncertainty principle. The analysis of the second and third pieces, two single instrument versions of the same song, Mary had a Little Lamb, employs the same method, Gabor Transform, but with the objective to construct a visual representation of the musical score.

2 Theoretical Background

The Gabor Transform, the methodology for this analysis, combines the Fourier Transform routine (see previous analysis of ultrasound in git) and signal filters in the time domain to produce both frequency and time resolution, depicted below in Figure 1.

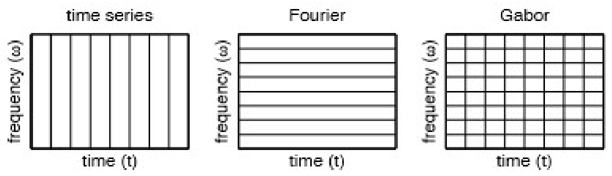

FIGURE 1. THREE CONCEPTUAL DRAWINGS REPRESENT HOW THE GABOR METHOD COMBINES TIME AND FREQUENCY DISCRETIZATION (TEXTBOOK[1])

Implementing the Gabor Method occurs in two major steps: 1) filter the signal in a window of time in the time domain, 2) Transform that filtered signal into its frequency components (Fourier Transform). This step is then repeated, with the filter starting at the beginning of the time series data and moved along the time axis in discrete steps. At each step, the resulting frequency data is compiled along with its time component.

The method requires the user to select a filter function and then two associated parameters: a = *scale* (window width) and b = *translation* (vector of values that defines step size). The three filters chosen in this analysis are below. (eq 1, eq 2, eq 3)

$$\text{Gaussian Filter: } g(t) = e^{-a(t-b)^2} \qquad \text{(eq 1)}$$

$$\text{Mexican Hat Filter: } m(t) = (1 - \left(\left(t - \frac{4.5}{a^2}\right)\right) * e^{-\frac{(t-b)^2}{2a^2}} \qquad \text{(eq 2)}$$

$$\text{Step (Shannon) filter: } s(t) = \begin{cases} 1, & |t-b| \le a \\ 0, & |t-b| > a \end{cases} \qquad \text{(eq 3)}$$

To understand the scale parameter, the Gaussian filter is shown below (figure 2) with varying scale values and the resulting filtered signal (filter * signal):

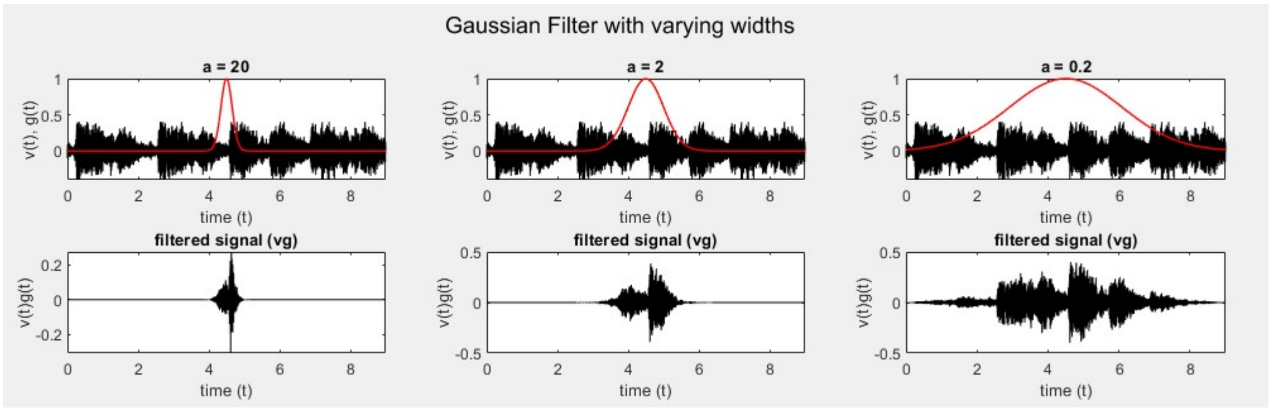

FIGURE 2. THE TOP ROW DEPICTS THE ORIGINAL AUDIO SIGNAL, DISPLAYED WITH A GAUSSIAN FILTER (RED), EACH WITH A VARYING WIDTH. THE RESULTING FILTERED SIGNAL IS THEN SHOWN IN THE SECOND ROW.

It's important to note that the size of your window acts as a constraint in two competing directions. The narrower the window, the better time resolution, however, narrow windows cannot capture lower frequency content. Wider windows capture both the high and lower frequencies but there is no time resolution within that wider window. This is an example of the Heisenberg Uncertainty principle, where you have a trade off between time and frequency resolution; improve one, and the other is degraded. How the translation parameter works is shown below, displaying two different translation parameters (figure 3). One is coarse (step size is large) and the other is fine (step size is smaller).

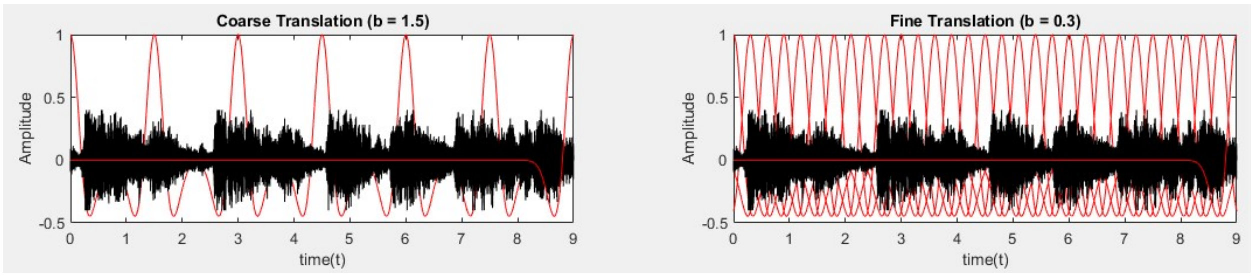

FIGURE 3. COARSE VERSUS FINE TRANSLATION PARAMETERS OF THE MEXICAN HAT FILTER (RED) ARE SUPERIMPOSED ON THE ORIGINAL SIGNAL ABOVE.

You can intuit that the best step size (translation) is impacted by the scale parameter (window width), by either missing frequency content (under sampling; too coarse) or duplicating frequency content due to excessive overlap (oversampling, to fine with respect to window size).

In the end, the Gabor Method, provides a spectrogram, displaying both frequency and time properties of a signal, and he choice of filter along with the two parameters of scale and translation determine its accuracy/quality.

3 Algorithm Implementation and Development

Three audio files were analyzed using the Gabor Method. These are the algorithms developed in MATLAB

3.1 Handel's Messiah (nine second audio file as 731131 x 1 matrix).

3.1.1 Load file, reduce data values by 50% and transpose as 1x731131 matrix

3.1.2 Setup time and frequency vectors for Fourier Transform routine
Discretize the [0,L] time domain using n = 731131 and L = 9 and then make it periodic.
Discretize the frequency domain based on Fourier Modes (n), and normalizing the [0, L] time interval to Fourier's assumed [-π, π] (i.e. multiply vector by $\frac{2\pi}{L}$) . This is our spectral resolution.

3.1.3 Setup Gaussian filter with three different widths to compare filtered signals and resulting frequency content. Plot to view.

3.1.4 Execute Gabor Method using Gaussian Filter with varying scale and translation parameters
3.1.4.1 Create Gaussian filter with parameter a = 15 (wide time window).
3.1.4.2 Set up translation vector tslide = [0:3:9] (coarse translation)
3.1.4.3 Use for loop to iterate over the tslide vector, multiplying the signal at each discretized step in time and then transform to frequency components using fft() and normalize. Compile resulting frequency information vectors at each iteration. Use fftshift() to bring 0 wavenumber (k) value to the center.
3.1.4.4 Plot Spectrogram using pcolor(). Remember to take absolution values, abs() and fftshift() function to bring 0 values to the center. Modify color contrast and range for desired appearance using caxis() function.
3.1.4.5 Repeat steps 3.1.4.1 – 3.1.4.4, with exception of using finer translation steps ([0:0.5:9]).
3.1.4.6 Repeat steps 3.1.4.1 – 3.1.4.4 for narrower window (a = 15) and coarse translation
3.1.4.7 Repeat step 3.1.4.6, with exception of using finer translation steps ([0:0.1:9]).

3.1.5 Setup Mexican hat filter with three different widths to compare filtered signals and resulting frequency content. Plot to view.

3.1.6 Execute Gabor Method, repeat steps 3.1.4 and 3.1.5, using Mexican hat filter instead of the Gaussian (a and b values will vary but repeat scenarios of wide vs narrow and coarse vs. fine)

3.1.7 Execute Gabor Method, repeat steps 3.1.4 and 3.1.5, using Step filter (Shannon) instead of the Gaussian (a and b values will vary but repeat scenarios of wide vs narrow and coarse vs. fine)

3.2 Mary had a little lamb (16 second audio file, piano)

3.2.1 Load music1.wav file using audioread(). Repeat step 3.1.2 using n = 701440 and L = 16.

3.2.2 Execute Fourier Transform (not Gabor Method) and plot to view frequency components.

3.2.3 Identify frequency (k) the dominant tone exists using the max() function. Plot Fourier Transform, zooming into this area of dominant tones to visually confirm and find top three.

3.2.4 Filter out all frequencies greater than 2x the lowest dominant tone (to remove overtones).

3.2.5 Transform filtered frequency data back to temporal domain using ifft() and ifftshift().

3.2.6 Repeat steps 3.1.4.1-3.1.4.4 using filtered audio signal produced in 3.2.5.

3.2.7 Repeat previous step for varying scale and translation parameters and plot spectrogram, adjusting caxis() for best contrast, ylim() to zoom area of the dominant tones in the top half (positive frequencies) of the spectrogram. Additionally, convert wavenumber frequency to Hz (i.e. divide by 2π).

3.3 Repeat 3.2 for Mary had a little lamb (recorder)

4 Computation Results

4.1 Handel's Messiah: Gabor Method, Exploring Scale and Translation Parameter values

4.1.1 Gabor Method: Gaussian filter

The visualizations below (figure 3) represent four implementations of the Gabor Method. The first two employ a wide window (large scale) Gaussian filter with two different translations (coarse versus fine). Note that the wide window with fine translation is an example of oversampling, and a smeared appearance occurs. In the bottom two rows we see narrow window width, also implemented with two different translation parameters. The resulting spectrograms shows us that we lose frequency content (colors are dimmer) but better time resolution. The narrow window width with coarse filter shows us an example of undersampling. The narrow window and fine translation implementation seems a better fit for time resolution however, we can still see that there is a lot of missing frequency information.

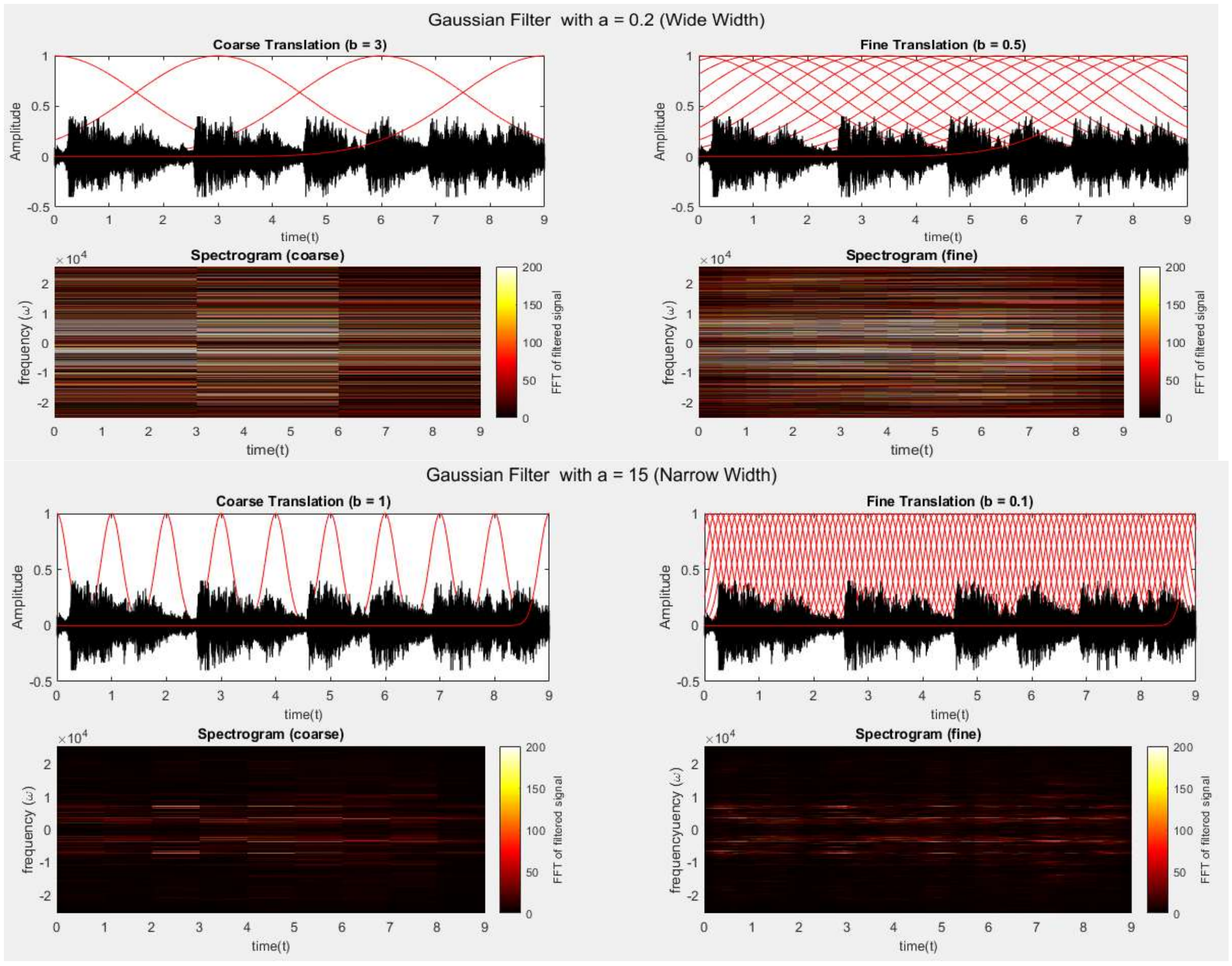

FIGURE 3. FOUR IMPLEMENTATIONS OF THE GABOR METHOD, USING A WIDE VS. NARROW SCALED GAUSSIAN FILTER AND COARSE VS. FINE TRANSLATION. THE RESULTING SPECTROGRAMS ARE DISPLAYED UNDER EACH.

4.1.2 Gabor Method: Mexican Hat filter

Just as in section 4.1.1 the visualizations below in figure 4 represent implementations of the Gabor Method, with the Mexican Hat filter (eq. 2). Again, we see the same type of limitation in frequency content for when the scale parameter defines a narrower window (top two implementations versus the bottom two).

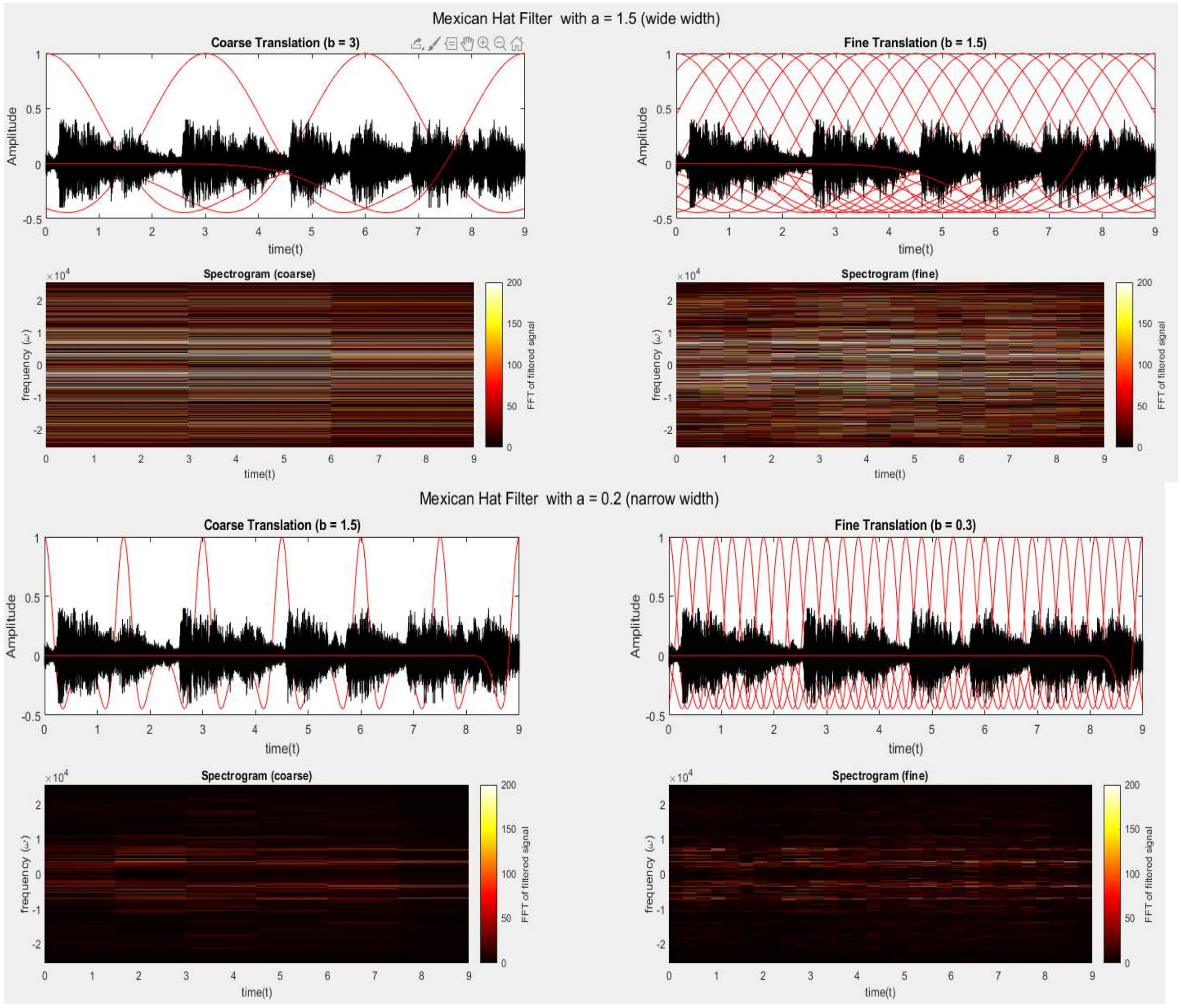

FIGURE 4. FOUR IMPLEMENTATIONS OF THE GABOR METHOD, USING A WIDE VS. NARROW SCALED MEXICAN HAT FILTER AND COARSE VS. FINE TRANSLATION. THE RESULTING SPECTROGRAMS ARE DISPLAYED UNDER EACH.

4.1.3 Gabor Method: Step Filter

Just as in section 4.1.1 and 4.1.2 the visualizations below in figure 5, represent four implementations of the Gabor Method but with the Shannon (step) filter (eq. 3). And again, we see the same type of limitation in frequency content for when the scale parameter produces a narrower window (top two implementations versus the bottom two). The impacts of the translation parameter are again on display. Note the result of undersampling, shown by the narrow window combined with coarse translation.

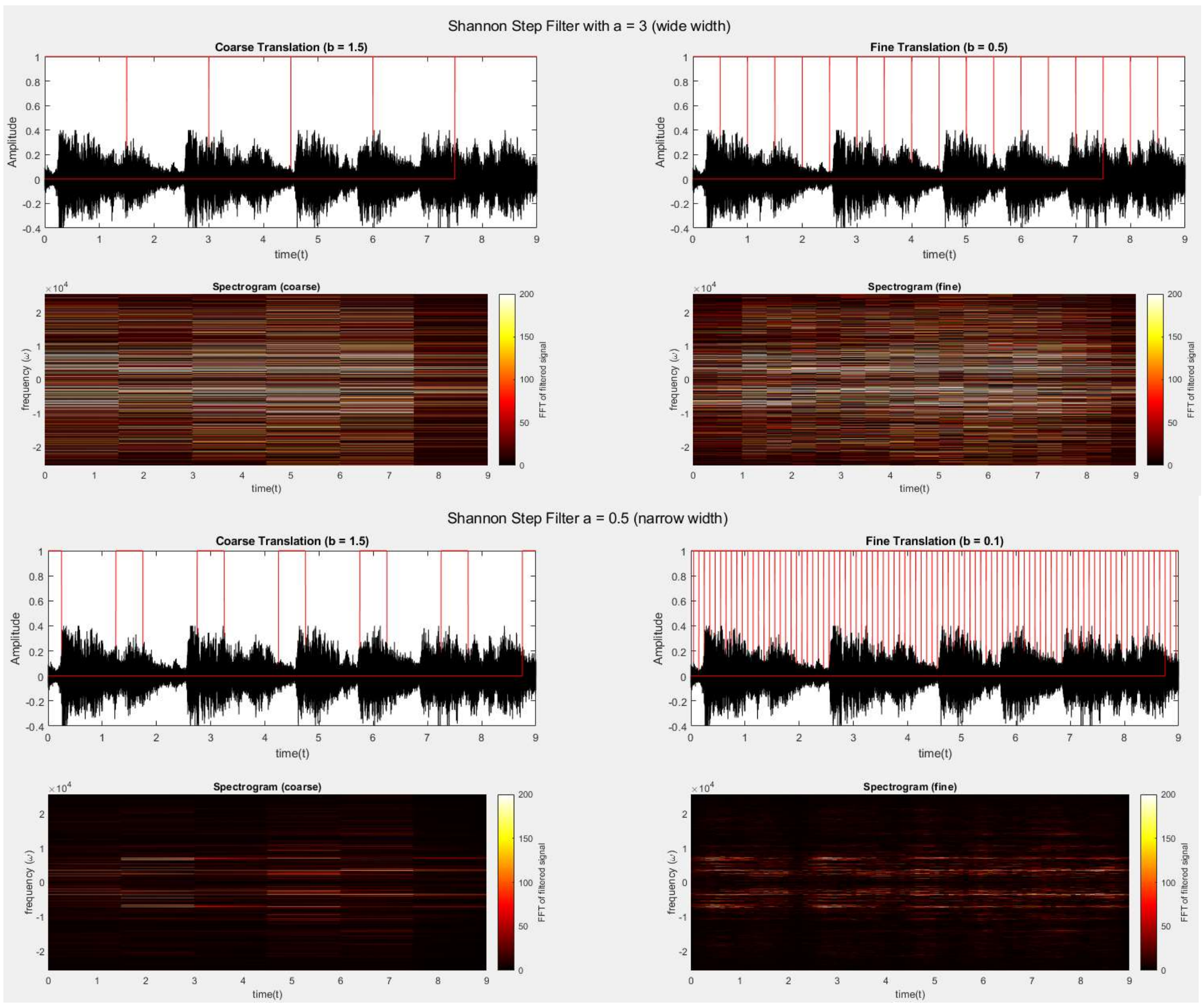

FIGURE 5. FOUR IMPLEMENTATIONS OF THE GABOR METHOD, USING A WIDE VS. NARROW SCALED SHANNON FILTER AND COARSE VS. FINE TRANSLATION. THE RESULTING SPECTROGRAMS ARE DISPLAYED UNDER EACH.

4.2 Mary had a Little Lamb (Piano and Recorder)

4.2.1 Dominant frequencies, and overtones

The dominant frequencies, representing the notes for both songs are shown in the two Fourier Transforms in figure 6. By For the piano, the peaks are $(\omega_1) \approx 1609$, $(\omega_2) \approx 1804$ and $(\omega_3) \approx 2009$. You can also notice mini peaks representing overtones at $(2\omega_1) \approx 3200$, $(2\omega_2) \approx 3600$ and $(2\omega_3) \approx 400$; and for the recorder, $(\omega_1) \approx 5132$, $(\omega_2) \approx 5724$ and $(\omega_3) \approx 6498$ with mini peaks representing overtones at $(2\omega_1) \approx 10220$, $(2\omega_2) \approx 11610$ and $(2\omega_3) \approx 12690$.

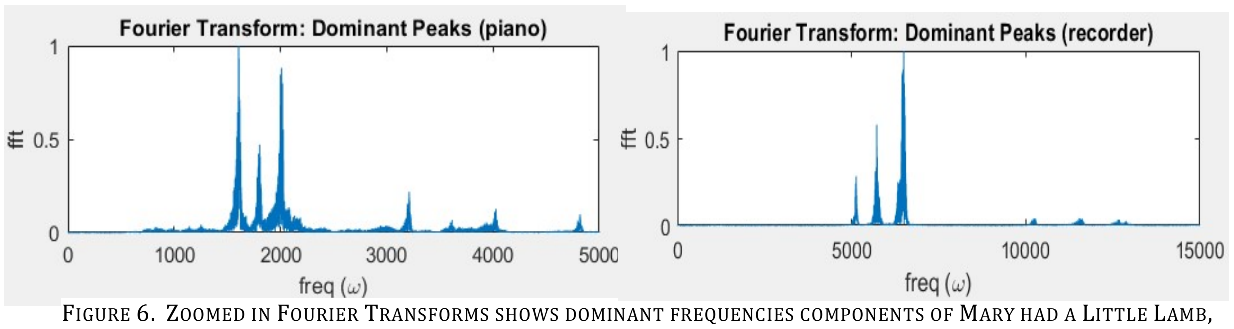

FIGURE 6. ZOOMED IN FOURIER TRANSFORMS SHOWS DOMINANT FREQUENCIES COMPONENTS OF MARY HAD A LITTLE LAMB, ON THE LEFT FOR THE PIANO RECORDING AND ON THE RIGHT FOR THE RECORDER. IN EACH THE THREE LARGE PEAKS, THE DOMINANT TONES CAN BE SEEN. MINI PEAKS CAN BE SEEN AT MULTIPLE VALUES, REPRESENTING THEIR OVERTONES.

The Fourier Transforms displayed above, are normalized and therefore we can compare the sizes of the dominant tones in relation to their overtones. The piano shows more significant overtone content in comparison to the recorder.

4.2.2 Spectrograms

Below are spectrograms for the piano version (figure 7) and the recorder (figure 8). In each the spectrogram is shown just below the actual audio signal to display the time resolution accuracy. Note that the spectrogram shows Hz instead of wavenumber and is zoomed into the positive (top half) of the frequency axis, visually mimicking notes on a musical score.

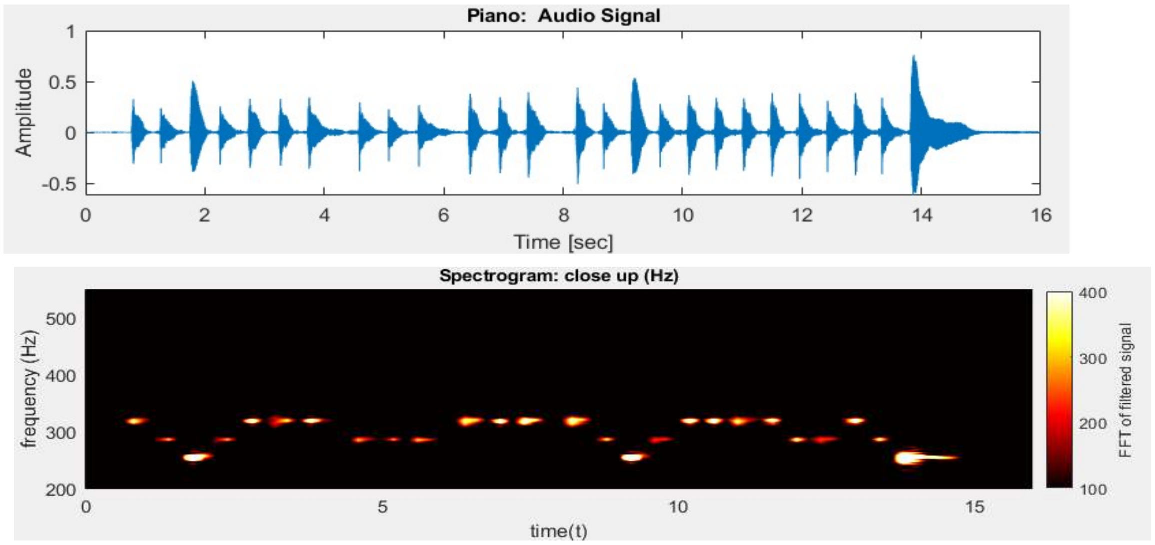

FIGURE 7. AUDIO SIGNAL (TOP) AND SPECTROGRAM (BOTTOM) FOR MARY HAD A LITTLE LAMB (PIANO). THE SPECTROGRAM IS ZOOMED INTO THE AREA OF INTEREST AND TUNED FOR CONTRAST.

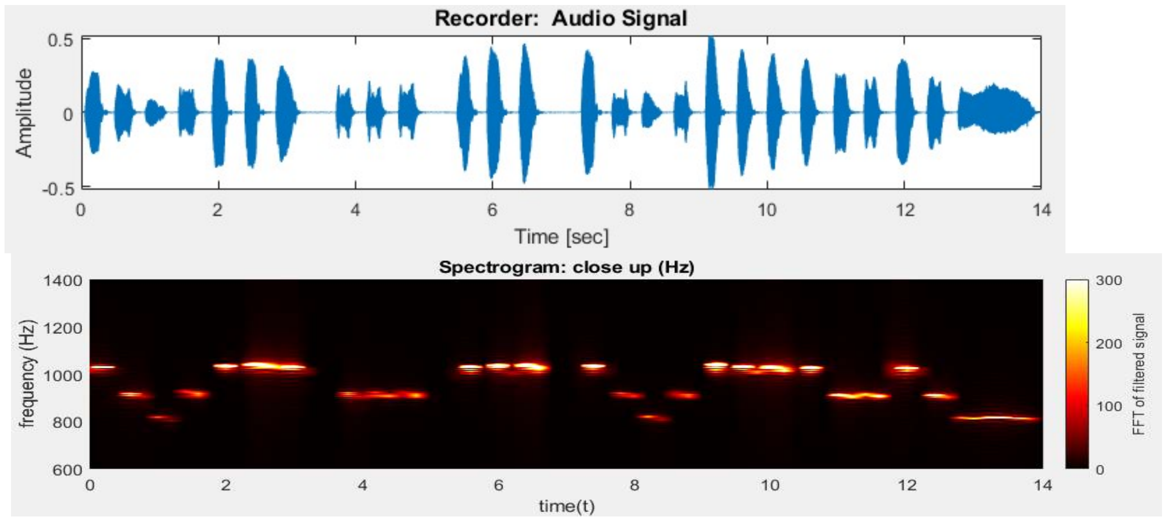

FIGURE 8. AUDIO SIGNAL (TOP) AND SPECTROGRAM (BOTTOM) FOR MARY HAD A LITTLE LAMB (RECORDER). THE SPECTROGRAM IS ZOOMED INTO THE AREA OF INTEREST AND TUNED FOR CONTRAST.

You can see in both spectrograms with frequency in Hz, the positions of the dominant tones match those found in section 4.2.1 above and time resolution visually matches the audio signal. Note that the filter choice and parameters, a and b, were optimized through trial and error. Based on the attempts, it seemed that the Shannon Step filter (a = 0.2 and step size = 0.2) provided a better visualization for mimicking notes. In addition, the color axis was modified to find the best contrast in identifying the dominant tones.

4.2.3 Approximate notes and resulting music score

Converting to the dominant frequency wavenumbers found in 4.2.1, the Hz values corresponding to each of the three dominant tones in each song were calculated. In addition, figure 9 and figure 10 are musical scores depicting the notes played in each song based on these calculations.

$|\omega_1| \approx 1609 \rightarrow 1609/(2*\pi) = 256.08 \approx$ middle C4 (middle C)
$|\omega_2| \approx 1804 \rightarrow 1804/(2*\pi) = 287.11 \approx$ D4
$|\omega_3| \approx 2009 \rightarrow 2009/(2*\pi) = 319.7 \approx$ E4

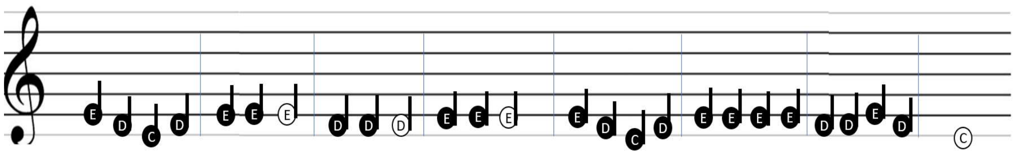

FIGURE 9. MUSICAL SCORE OF MARY HAD A LITTLE LAMB (PIANO) BASED ON FREQUENCY OF DOMINANT NOTES AND TIME RESOLUTION OF SPECTROGRAM.

$|\omega_1| \approx 5132 \rightarrow 5132/(2*\pi) = 816.78 \approx$ G5
$|\omega_2| \approx 5724 \rightarrow 5724/(2*\pi) = 911.00 \approx$ A5
$|\omega_3| \approx 6498 \rightarrow 6498/(2*\pi) = 1034.2 \approx$ B5/C5 (going with B5, the recorder is out of tune)

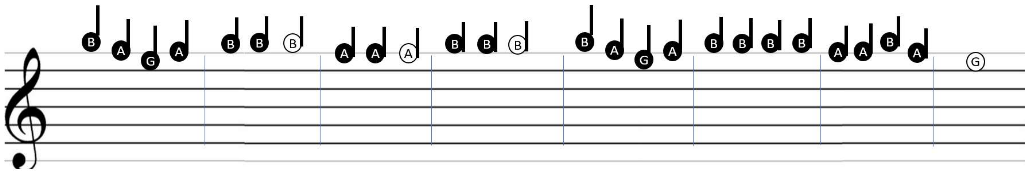

FIGURE 10. MUSICAL SCORE OF MARY HAD A LITTLE LAMB (RECORDER) BASED ON FREQUENCY OF DOMINANT NOTES AND TIME RESOLUTION OF SPECTROGRAM.

4.2.4 Comparison of overtones: Piano vs. Recorder

Note that in sections 4.2.1 and 4.3.1, where we were able to see the normalized frequency components of both versions. The 'mini peaks' representing the overtones for the dominant frequencies were much larger for the piano than the recorder. In addition, if we compare their spectrograms using the same contrast (color axis), we also see more banding near the notes as well (overtones of weaker frequencies) for the piano. Figure 11 below displays this difference.

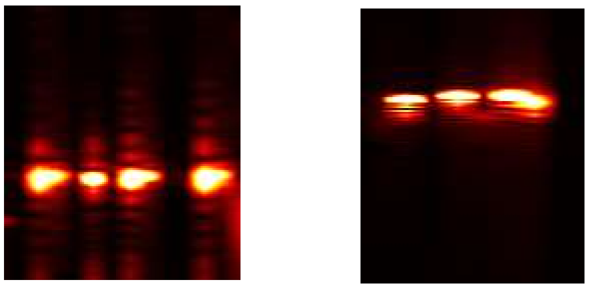

FIGURE 11. THE IMAGE ON THE LEFT IS A SECTION OF THE SPECTROGRAM FOR THE PIANO. THE ONE ON THE RIGHT, THE RECORDER, USING THE SAME COLOR AXIS [0 300]. NOTE THE DIFFERENCE IN BANDING.

5 Summary and Conclusions

The success of the Gabor Method relies heavily on the choices made for the implementation: which filter to employ and their parameters of scale and translation. In the first part of this analysis (Handel's Messiah) we saw the trade-off that occurs between time and frequency resolution. In order to capture more frequency content, we need a scale parameter that produces a wider window that will not omit lower frequencies (longer wavelengths). But since that wide window has no time resolution across its span, we have less specific time localization. Then if we need more specific time localization, that requires a narrower window, and then we lose frequency content. So, it's a tradeoff, where the best choice depends on what you want the spectrogram to tell you. Do you need better time resolution or frequency resolution? Improving one, degrades the other, a representation of the Heisenberg uncertainty principle.

This important principle, the tradeoff between time and frequency resolution, was then used to produce a spectrogram using the Gabor Method that best represents the music score (musical notes) in the recordings of the two audio files of Mary had a little lamb. In addition, we also determined differences in overtone content of the piano recording of Mary had a little lamb (more overtone content for the dominant frequencies), versus the recorder version (less). But we also saw, more mini echoes near the actual notes themselves (overtones of non-dominant frequencies?). Since all overtones impact the quality of a sound, we were able to visualize we can tell the difference between the sound of a piano, and that of a recorder.

References

[1] J. Nathan Kutz 2013, Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data, Oxford University Press
[2] https://newt.phys.unsw.edu.au/music/note/

Appendix A: Relevant MATLAB functions

1. fftn(): This an N dimensional discrete fast Fourier Transform function. Since we were working with 3D spatial data, fftn() was used rather than ft() to transform 3D spatial data to a 3D frequency domain

2. fftshift(): This function was is used to move the 0 values back to the center after the fftn() function was used. Does not affect processing but allows us to better visualize the frequency domain

3. ifftn(): This is the N dimensional Inverse discrete fast Fourier Transform function. This was used to transform data back from the spectral domain to the spatial domain.

4. ifftshift(): This reverses the fftshift() that was performed in the spectral domain, back to the original frequency coordinate order where the 0 values are at the outer edges. This should be performed prior to executing the ifftn() to transform back to the spatial domain.

5. pcolor(): function used to plot a matrix using color. This function was used to generate the spectrogram.

6. colormap(): provides ability to determine what color scheme is used in the pcolor visualization. In our analysis we used colormap(hot).

7. colorbar: This adds a color bar reference (i.e. shows to and from values along the color range of the pcolor() visualization.

8. caxis(): This allows you to determine the scale and endpoints of the color range of the pcolor() visualization. It was used in this analysis to tweak the level of contrast in the spectrogram.

9. 'for loop' (honorable mention): This isn't a function but the 'for loop' allowed us to perform iterations across all 20 measurements with minimal coding.

Appendix B: MATLAB Code

```
%% Part 1, Exploring Gabor window sizes %%
clear all; close all; clc
load handel
%%
v = y'/2; % new vector, transpose of y/2
vf = fft(v);  %% FFT of signal
vfs = fftshift(vf); %% FFT shifted of signal

L=9; n=73113; % set up time domain and fourier modes
t2=linspace(0,L,n+1); t= t2(1:n); % setup time vector based on signal data, periodic
k=(2*pi/L)*[0:n/2 -n/2:-1];  % setup freq vector for odd fourier mode (symmetrical)
ks=fftshift(k); % flip so ks has ordered modes with 0 at center

% plot signal and fft of signal
figure(1)
sgtitle('Signal of Interest v(t)');
subplot(2,1,1);
plot((1:length(v))/Fs,v);
xlim([0 9]);
set(gca,'Fontsize',[10]) ;
xlabel('Time [sec]');
ylabel('Amplitude');
title('v(t)');
subplot(2,1,2);
plot(ks,abs(vfs)/max(abs(vfs)));
xlabel('frequency (\omega)');
ylabel('fft(v)');
title('Fourier Transform - v(t)');

 %%  Gabor Method with Gaussian fIlter
figure(2)
sgtitle('Gaussian Filter with varying widths')
width=[20 2 0.2];  % using 3 different gabor widths

for j=1:3
    g=exp(-width(j)*(t-4.5).^2);  % gabor window is gaussian filter
    subplot(3,3,(j))
    plot(t, v,'k', 'Linewidth', [.25]), hold on
    plot(t,g,'r','Linewidth',[1])
    set(gca,'Fontsize',[9])
    xlim([0 9]);
    ylabel('v(t), g(t)')
    xlabel('time (t)')
    title(['a = ', num2str(width(j))])
    vg = g.*v;
    vgf = fft(vg);
    vgfs_all(j , :) = abs(fftshift(vgf))/max(abs(vgf));  %% FFT of signal

    subplot(3,3, j+3), plot(t,vg,'k')
    xlim([0 9]);
    set(gca,'Fontsize',[9])
    title('filtered signal (vg)')
    ylabel('v(t)g(t)'), xlabel('time (t)')

    subplot(3,3,(j)+6);
    title('Fourier Transform');
    plot(ks, vgfs_all(j, :), 'b', 'Linewidth', [1]);
    set(gca, 'Fontsize', [9]);
    ylabel('fft(vg)');
    xlabel('frequency (\omega)');
   end

%  Sliding Gabor Window (Gaussian), wide window, coarse translation
figure(3)
sgtitle('Gaussian Filter  with a = 0.2 (Wide Width)')
vgt_spec=[];
tslide=[0:3: 9];
for ii=1:length(tslide)
    g=exp(-.2*(t-tslide(ii)).^2); % Gabor
    vg = g.*v;
    vgt=fft(vg);
    vgt_spec=[vgt_spec;  abs(fftshift(vgt))];
    title('Coarse Translation (b = 3)')
    subplot(2,2,1), plot(t,v,'k', t,g, 'r') ;  hold on;
    xlabel('time(t)', 'Fontsize', [10])
     ylabel('Amplitude')
end

subplot(2,2,3)
pcolor(tslide,ks,vgt_spec.'),
shading interp
title('Spectrogram (coarse)')
xlabel('time(t)')
ylabel('frequency (\omega)');
set(gca,'Fontsize',[10])
colormap(hot)
colorbar
h = colorbar; ylabel(h, 'FFT of filtered signal')
caxis([0 200])

% Gaussian (wide window, fine translation)
vgt_spec=[];
tslide=[0:.5: 9];
for ii=1:length(tslide)
    g=exp(-.2*(t-tslide(ii)).^2); % Gabor
    vg = g.*v;
    vgt=fft(vg);
    vgt_spec=[vgt_spec;  abs(fftshift(vgt))];
    subplot(2,2,2), plot(t,v,'k', t,g, 'r') ;  hold on;
        title('Fine Translation (b = 0.5)')
    xlabel('time(t)', 'Fontsize', [10]);
    ylabel('Amplitude')
end

subplot(2,2,4)
pcolor(tslide,ks,vgt_spec.'),
shading interp
 xlabel('time(t)', 'Fontsize', [10]);
ylabel('frequency (\omega)')
set(gca,'Fontsize',[10])
colormap(hot)
colorbar
title('Spectrogram (fine)');
h = colorbar; ylabel(h, 'FFT of filtered signal')
caxis([0 200])

%  Sliding Gabor Window (Gaussian) - narrow window, coarse translation
figure(4)
sgtitle('Gaussian Filter  with a = 15 (Narrow Width)')
vgt_spec=[];
tslide=[0:1: 9];
for ii=1:length(tslide)
   g=exp(-15*(t-tslide(ii)).^2); % Gabor
   vg = g.*v;
   vgt=fft(vg);
   vgt_spec=[vgt_spec;  abs(fftshift(vgt))];
   subplot(2,2,1), plot(t,v,'k', t,g, 'r') ;  hold on;
     xlabel('time(t)', 'Fontsize', [10]);
    ylabel('Amplitude')
    title('Coarse Translation (b = 1)')
end

subplot(2,2,3)
pcolor(tslide,ks,vgt_spec.'),
shading interp
xlabel('time(t)', 'Fontsize', [10]);
ylabel('frequency (\omega)')
colormap(hot)
colorbar
title('Spectrogram (coarse)');
h = colorbar; ylabel(h, 'FFT of filtered signal')
caxis([0 200])

% Gaussian (narrow window, fine translation)
vgt_spec=[];
tslide=[0:.1: 9];
for ii=1:length(tslide)
   g=exp(-15*(t-tslide(ii)).^2); % Gabor
   vg = g.*v;
   vgt=fft(vg);
   vgt_spec=[vgt_spec;  abs(fftshift(vgt))];
```

```
    subplot(2,2,2), plot(t,v,'k', t,g, 'r') ;  hold on;
   title('Fine Translation (b = 0.1)')
    xlabel('time(t)', 'Fontsize', [10]);
     ylabel('Amplitude')
end

subplot(2,2,4)
pcolor(tslide,ks,vgt_spec.'),
shading interp
xlabel('time(t)', 'Fontsize', [10]);
ylabel('frequencyuency (\omega)')
colormap(hot)
colorbar
h = colorbar; ylabel(h, 'FFT of filtered signal')
title('Spectrogram (fine)');
caxis([0 200])

%% best fit Gabot with Gaussian
figure(5)
vgt_spec=[];
tslide=[0:.3: 9];
for ii=1:length(tslide)
    g=exp(-9*(t-tslide(ii)).^2); % Gabor
    vg = g.*v;
    vgt=fft(vg);
    vgt_spec=[vgt_spec;  abs(fftshift(vgt))];
    subplot(2,1,1), plot(t,v,'k', t,g, 'r') ;  hold on;
   title('/sigma = 10 & translation = 0.3')
    xlabel('time(t)', 'Fontsize', [10]);
     ylabel('Amplitude')
end

subplot(2,1,2)
pcolor(tslide,ks,vgt_spec.'),
shading interp
xlabel('time(t)', 'Fontsize', [10]);
ylabel('frequency (\omega)')
colormap(hot)
colorbar
h = colorbar; ylabel(h, 'FFT of filtered signal')
title('Spectrogram (fine)');
caxis([0 200])

%  Gabor Method with mexican hat filter
figure(6)
sgtitle('Mexican Hat Filter of varying widths')
width=[1 .5 0.2];  % using 3 different gabor widths

for jj=1:3
    m=(1-((t-4.5)./width(jj)).^2).*exp(-(t-4.5).^2/(2.*width(jj).^2));  % gabor window is gaussian filter
    subplot(3,3,(jj))
    plot(t, v,'k'), hold on
    plot(t,m,'r','Linewidth',[2])
    set(gca,'Fontsize',[9])
    xlim([0 9]);
    ylabel('v(t), m(t)')
    xlabel('time (t)')
    title( width(jj))
    vm = m.*v;
    vmf = fft(vm);
    vmfs_all(jj, :) = abs(fftshift(vmf))/max(abs(vmf));  %% FFT of signal

    subplot(3,3, jj+3), plot(t,vm,'k')
    set(gca,'Fontsize',[9])
    xlim([0 9]);
    title('filtered signal (vm)')
    ylabel('v(t)m(t)'), xlabel('time (t)')

    subplot(3,3,(jj)+6);
    plot(ks, vmfs_all(jj, :), 'b', 'Linewidth', [1]);
    title('Fourier Transform (vm)')
    set(gca, 'Fontsize', [9]);
    ylabel('fft(vm)');
    xlabel('frequency (\omega)');
end

% gabor window (mexican hat) wide window coarse translation
figure(7)
sgtitle('Mexican Hat Filter  with a = 1.5 (wide width)')
vmt_spec=[];
tslide=0:3:9;
for ii=1:length(tslide)
    m=(1-((t-tslide(ii))./1.5).^2).*exp(-(t-tslide(ii)).^2/(2.*1.5.^2)); % Gabor
    vm = m.*v;
    vmt=fft(vm);
    vmt_spec=[vmt_spec;  abs(fftshift(vmt))];
    subplot(2,2,1), plot(t,v,'k', t,m, 'r'); hold on;
        ylabel('Amplitude');
     xlabel('time(t)')
    title('Coarse Translation (b = 3)')
end
%
subplot(2,2,3),
pcolor(tslide,ks,vmt_spec.'),
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar
title('Spectrogram (coarse)');
h = colorbar; ylabel(h, 'FFT of filtered signal')
caxis([0 200])
ylabel('frequency (\omega)')
```

```
xlabel('time(t)')

% mhat, wide window, fine translation
vmt_spec=[];
tslide=0:.5:9;
for ii=1:length(tslide)
    m=(1-((t-tslide(ii))./1.5).^2).*exp(-(t-tslide(ii)).^2/(2.*1.5.^2)); % Gabor
    vm = m.*v;
    vmt=fft(vm);
    vmt_spec=[vmt_spec;  abs(fftshift(vmt))];
    subplot(2,2,2), plot(t,v,'k', t,m, 'r'); hold on;
     title('Fine Translation (b = 1.5)')
     ylabel('Amplitude');
     xlabel('time(t)')
end

subplot(2,2,4),
pcolor(tslide,ks,vmt_spec.'),
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar
h = colorbar; ylabel(h, 'FFT of filtered signal')
title('Spectrogram (fine)');
caxis([0 200])
ylabel('frequency (\omega)')
xlabel('time(t)')

%% narrow window coarse translation
figure(8)
sgtitle('Mexican Hat Filter  with a = 0.2 (narrow width)')
vmt_spec=[];
tslide=0:1.5:9;
for ii=1:length(tslide)
    m=(1-((t-tslide(ii))./0.2).^2).*exp(-(t-tslide(ii)).^2/(2.*0.2.^2)); % Gabor
    vm = m.*v;
    vmt=fft(vm);
    vmt_spec=[vmt_spec;  abs(fftshift(vmt))];
    subplot(2,2,1), plot(t,v,'k', t,m, 'r'); hold on;
      title('Coarse Translation (b = 1.5)')
       ylabel('Amplitude');
     xlabel('time(t)')
end
subplot(2,2,3),
pcolor(tslide,ks,vmt_spec.'),
title('Spectrogram (coarse)');
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar
caxis([0 200])
```

```
h = colorbar; ylabel(h, 'FFT of filtered signal')
ylabel('frequency (\omega)')
xlabel('time(t)')

% mhat, wide window, fine translation
vmt_spec=[];
tslide=0:.3:9;
for ii=1:length(tslide)
    m=(1-((t-tslide(ii))./0.2).^2).*exp(-(t-tslide(ii)).^2/(2.*0.2.^2)); % Gabor
    vm = m.*v;
    vmt=fft(vm);
    vmt_spec=[vmt_spec;  abs(fftshift(vmt))];
    subplot(2,2,2), plot(t,v,'k', t,m, 'r'); hold on;
      title('Fine Translation (b = 0.3)')
       ylabel('Amplitude');
     xlabel('time(t)')
end
%
subplot(2,2,4),
pcolor(tslide,ks,vmt_spec.'),
title('Spectrogram:  Mexican Hat Filter')
shading interp
set(gca,'Fontsize',[9])
title('Spectrogram (fine)');
colormap('hot')
colorbar
caxis([0 200])
h = colorbar; ylabel(h, 'FFT of filtered signal')
ylabel('frequency (\omega)')
xlabel('time(t)')

%best fit
figure(9)
vmt_spec=[];
tslide=0:.3:9;
for ii=1:length(tslide)
    m=(1-((t-tslide(ii))./0.2).^2).*exp(-(t-tslide(ii)).^2/(2.*0.2.^2)); % Gabor
    vm = m.*v;
    vmt=fft(vm);
    vmt_spec=[vmt_spec;  abs(fftshift(vmt))];
    subplot(2,1,1), plot(t,v,'k', t,m, 'r'); hold on;
    title('Signal of Interest w// Mexican Hat filter, all translations')
  end
%
subplot(2,1,2),
pcolor(tslide,ks,vmt_spec.'),
title('Spectrogram:  Mexican Hat Filter')
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar
```

```
caxis([0 200])

% gabor window (Step Function: Shannon)
figure(10)
sgtitle('Shannon Step Filter with varying widths');
width=[3 1 0.2];  % using 3 different gabor widths

for jj=1:3
    sh = abs(t-4.5) <= width(jj)/2;  % gabor window with Shannon Step filter
    subplot(3,3,(jj))
    plot(t, v,'k'), hold on
    plot(t,sh,'r','Linewidth',[1])
    set(gca,'Fontsize',[9])
    xlim([0 9]);
    ylabel('v(t), m(t)')
    xlabel('time (t)')
    title( width(jj))
    vs = sh.*v;
    vsf = fft(vs);
    vsfs_all(jj, :) = abs(fftshift(vsf))/max(abs(vsf));  %% FFT of signal

    subplot(3,3, jj+3), plot(t,vs,'k')
    set(gca,'Fontsize',[9])
    xlim([0 9]);
    title('filtered signal (vs)')
    ylabel('v(t)s(t)'), xlabel('time (t)')

    subplot(3,3,(jj)+6);
    plot(ks, vsfs_all(jj, :), 'b', 'Linewidth', [1]);
    set(gca, 'Fontsize', [9]);
    ylabel('fft(vs)');
    xlabel('frequency (\omega)');

end
% wide window coarse translation
figure(11)
sgtitle('Shannon Step Filter with a = 3 (wide width)')
vst_spec=[];
tslide=0:1.5:9;
for ii=1:length(tslide)
    sh = abs(t-tslide(ii)) <= 3/2; % shannon filter
    vs = sh.*v;
    vst=fft(vs);
    vst_spec=[vst_spec;  abs(fftshift(vst))];
    subplot(2,2,1), plot(t,v,'k', t,sh, 'r'); hold on;
    title('Coarse Translation (b = 1.5)')
          ylabel('Amplitude');
     xlabel('time(t)')
end

subplot(2,2,3),
pcolor(tslide,ks,vst_spec.'),
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar
caxis([0 200])
h = colorbar; ylabel(h, 'FFT of filtered signal')
ylabel('frequency (\omega)')
    title('Spectrogram (coarse)')
xlabel('time(t)')
% shannon, wide window, fine translation
vst_spec=[];
tslide=0:.5:9;
for ii=1:length(tslide)
    sh = abs(t-tslide(ii)) <= 3/2; % Gabor
    vs = sh.*v;
    vst=fft(vs);
    vst_spec=[vst_spec;  abs(fftshift(vst))];
    subplot(2,2,2), plot(t,v,'k', t,sh, 'r'); hold on;
    title('Fine Translation (b = 0.5)')
     ylabel('Amplitude');
     xlabel('time(t)')
end
%
subplot(2,2,4),
pcolor(tslide,ks,vst_spec.'),
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar
h = colorbar; ylabel(h, 'FFT of filtered signal')
caxis([0 200])
    title('Spectrogram (fine)')
ylabel('frequency (\omega)')
xlabel('time(t)')

% shannon, narrow window, coarse translation
figure(12)
sgtitle('Shannon Step Filter a = 0.5 (narrow width)')
vst_spec=[];
tslide=0:1.5:9;
for ii=1:length(tslide)
    sh = abs(t-tslide(ii)) <= 0.5/2; % Gabor
    vs = sh.*v;
    vst=fft(vs);
    vst_spec=[vst_spec;  abs(fftshift(vst))];
    subplot(2,2,1), plot(t,v,'k', t,sh, 'r'); hold on;
    title('Coarse Translation (b = 1.5)')
     ylabel('Amplitude');
     xlabel('time(t)')
end
```

```
subplot(2,2,3),
pcolor(tslide,ks,vst_spec.'),
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar
caxis([0 200])
ylabel('frequency (\omega)')
xlabel('time(t)')
h = colorbar; ylabel(h, 'FFT of filtered signal')
title('Spectrogram (coarse)')
% shannon, narrow window, fine translation
vst_spec=[];
tslide=0:.1:9;
for ii=1:length(tslide)
    sh = abs(t-tslide(ii)) <= 0.5/2; % Gabor
    vs = sh.*v;
    vst=fft(vs);
    vst_spec=[vst_spec;  abs(fftshift(vst))];
    subplot(2,2,2), plot(t,v,'k', t,sh, 'r'); hold on;
    title('Fine Translation (b = 0.1)')
      ylabel('Amplitude');
     xlabel('time(t)')
end
%
subplot(2,2,4),
pcolor(tslide,ks,vst_spec.'),
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar
caxis([0 200])
ylabel('frequency (\omega)')
xlabel('time(t)')
h = colorbar; ylabel(h, 'FFT of filtered signal')
title('Spectrogram (fine)')

% shannon, narrow window, fine translation
figure(13)
vst_spec=[];
tslide=0:0.1:9;
for ii=1:length(tslide)
    sh = abs(t-tslide(ii)) <= 0.5/2; % Gabor
    vs = sh.*v;
    vst=fft(vs);
    vst_spec=[vst_spec;  abs(fftshift(vst))];
    subplot(2,1,1), plot(t,v,'k'); hold on;
    ylabel('Amplitude')
    xlabel('time(t)')
    title('original audio signal')
```

```
end

subplot(2,1,2);
pcolor(tslide,ks,vst_spec.'),
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
ylabel('frequency (\omega)')
xlabel('time(t)')
h = colorbar; ylabel(h, 'FFT of filtered signal')
title('Spectrogram:  Shannon Filter (a = 0.5, b = 0.1)')

%% PART 2: Piano
clear all; close all; clc
tr_piano = 16; % record time in seconds
y=audioread('music1.wav');

p = y.';
pf = fft(p);  %% FFT of signal
pfs = fftshift(pf); %% FFT shifted of signal

L=tr_piano; n=701440; % set up time domain and fourier modes
t2=linspace(0,L,n+1); t=t2(1:n);  % setup time vector based on signal data, periodic
k=(2*pi/L)*[0:n/2-1 -n/2:-1];   % setup freq vector
ks=fftshift(k); % flip so ks has ordered modes with 0 at center

% plot signal and fft of signal
figure(1)
sgtitle('Mary had a little lamb:  Piano');
%subplot(2,1,1);
Fs = length(p)/tr_piano;
plot((1:length(p))/Fs,p);
xlim([0 16]);
set(gca,'Fontsize',[9]) ;
xlabel('Time [sec]'); ylabel('Amplitude');
title ('Audio Signal');

subplot(2,1,2);
plot(ks,abs(pfs)/max(abs(pfs)));
xlabel('freq (\omega)');
ylabel('fft');
title('Fourier Transform');

figure(2)
subplot(2,1,1);
plot(ks,abs(pfs)/max(abs(pfs)));
xlabel('freq (\omega)');
ylabel('fft');
title('Fourier Transform close up near peaks');
xlim([0 5000]);
```

```
%find max value in three areas with peaks, this is the lowest note
[M1,I1]= max(pfs);
max1_wave = ks(1, I1);% first peak

%zero out anything above 2x this value (covers its overtones in addition
%to any overtones of the three later peaks
ind = abs(ks(1,:)) >3000; ; %grab indices along second column where value greater than lowest
multiple of peak1
pfs_clean = num2cell(pfs); % youre going to need to turn it into a cell array
pfs_clean(1,ind) = {0}; %set those indices to zero
pfs_clean = cell2mat(pfs_clean);


subplot(2,1,2);
plot(ks,abs(pfs_clean)/max(abs(pfs_clean)));
xlabel('freq (\omega)');
ylabel('fft');
title('Fourier Transform close up near peaks, cleaned');
xlim([0 5000]);

% ifft and iffshift to pfs_clean data for gabor filtering
pf_clean = ifftshift(pfs_clean);
p_clean = ifft(pf_clean);



%Gaussian
figure(3)
sgtitle('Gaussian Filter  with a = 10 & b = 0.2')
pgt_spec=[];
tslide=[0:0.2: 16];
for ii=1:length(tslide)
   g=exp(-10*(t-tslide(ii)).^2); % Gabor
   pg = g.*p_clean;
   pgt=fft(pg);
   pgt_spec=[pgt_spec;  abs(fftshift(pgt))];
end
%
subplot(2,1,1)
pcolor(tslide,ks,pgt_spec.'),
shading interp
xlabel('time(t)')
set(gca,'Fontsize',[10])
colormap(hot)
title('Spectrogram');
colorbar
h = colorbar; ylabel(h, 'FFT of filtered signal')
caxis([0 300])
ylim([-4000 4000])

xlabel('time(t)');
```

```
ylabel('frequency (\omega)')
%
subplot(2,1,2)
Hz = ks/(2*pi);
pcolor(tslide,Hz,pgt_spec.'),
shading interp
xlabel('time(t)')
set(gca,'Fontsize',[10])
colormap(hot)
title('Spectrogram: close up (Hz)');
xlabel('time(t)');
ylabel('frequency (Hz)')
colorbar
h = colorbar; ylabel(h, 'FFT of filtered signal')
caxis([0 800])
ylim([100 400])

% Step Function
figure(4)
sgtitle('Shannon step filter w/ a = 0.2 and b = 0.2')
pst_spec=[];
tslide=[0:0.2: 16];
for ii=1:length(tslide)
    sh = abs(t-tslide(ii)) <= 0.2/2; % Shannon filter
    ps = sh.*p_clean;
    pst=fft(ps);
    pst_spec=[pst_spec;  abs(fftshift(pst))];
end

%
subplot(2,1,1)
pcolor(tslide,ks,pst_spec.'),
shading interp
xlabel('time(t)')
set(gca,'Fontsize',[10])
colormap(hot)
title('Spectrogram');
colorbar
h = colorbar; ylabel(h, 'FFT of filtered signal')
caxis([0 800])
ylim([-3500 3500])
xlabel('time(t)');
ylabel('frequency (\omega)')
%
subplot(2,1,2)
Hz = ks/(2*pi);
pcolor(tslide,Hz,pst_spec.'),
shading interp
xlabel('time(t)')
set(gca,'Fontsize',[10])
colormap(hot);
```

```
title('Spectrogram: close up (Hz)');
xlabel('time(t)');
ylabel('frequency (Hz)')
colorbar
h = colorbar; ylabel(h, 'FFT of filtered signal')
caxis([100 400])
ylim([200 550])

%% PART 2: Recorder %%
clear all;
tr_rec= 14; % record time in seconds
y=audioread('music2.wav');
r = y.';
rf = fft(r);  % FFT of signal
rfs = fftshift(rf); %% FFT shifted of signal

L=tr_rec; n=length(y) ; % set up time domain and fourier modes
t2=linspace(0,L,n+1); t=t2(1:n);  % setup time vector based on signal data, periodic
k=(2*pi/L)*[0:n/2-1 -n/2:-1];   % setup freq vector
ks=fftshift(k); % flip so ks has ordered modes with 0 at center

% plot signal and fft of signal
figure(5)
subplot(2,1,1)
sgtitle('Mary had a little lamb:  Recorder');
Fs = length(r)/tr_rec;
plot((1:length(r))/Fs,r);
set(gca,'Fontsize',[10]) ;
xlabel('Time [sec]'); ylabel('Amplitude');
title ('Recorder:  Audio Signal', 'Fontsize', [12]);
subplot(2,1,2)
plot(ks,abs(rfs)/max(abs(rfs)));
xlabel('freq (\omega)');
ylabel('fft');
title('Fourier Transform');

figure(6)
subplot(2,1,1);
plot(ks,abs(rfs)/max(abs(rfs)));
set(gca,'Fontsize',[12]) ;
xlabel('freq (\omega)');
ylabel('fft');
title('Fourier Transform: Dominant Peaks (recorder)', 'Fontsize', [13]);

%zero out anything above 2x this value (covers its overtones in addition
%to any overtones of the three later peaks
ind = abs(ks(1,:)) >10000; ; %grab indices along second column where value greater than lowest
multiple of peak1
rfs_clean = num2cell(rfs); % youre going to need to turn it into a cell array
rfs_clean(1,ind) = {0}; %set those indices to zero
rfs_clean = cell2mat(rfs_clean);
```

```
subplot(2,1,2);
plot(ks,abs(rfs_clean)/max(abs(rfs_clean)));
xlabel('freq (\omega)');
ylabel('fft');
title('Fourier Transform close up near peaks, cleaned');
% ifft and iffshift to pfs_clean data for gabor filtering
rf_clean = ifftshift(rfs_clean);
r_clean = ifft(rf_clean);
% Gaussian
figure(7)
sgtitle('Gaussian Filter  with a = 10 & translation = 0.2')
rgt_spec=[];
tslide=[0:0.2: 16];
for ii=1:length(tslide)
   g=exp(-12*(t-tslide(ii)).^2); % Gabor
   rg = g.*r_clean;
   rgt=fft(rg);
   rgt_spec=[rgt_spec;  abs(fftshift(rgt))];
   subplot(3,1,1), plot(t,r_clean,'k', t,g, 'r') ;
   xlabel('time(t)', 'Fontsize', [10])
    ylabel('Amplitude')
  % subplot(4,1,2), plot(t,pg,'k')
 %  subplot(4,1,3), plot(ks,abs(fftshift(pgt))/max(abs(pgt)))

end
%
subplot(3,1,2)
pcolor(tslide,ks,rgt_spec.'),
shading interp
xlabel('time(t)')
set(gca,'Fontsize',[10])
colormap(hot)
title('Spectrogram');
colorbar
h = colorbar; ylabel(h, 'FFT of filtered signal')
ylim([-25000 25000])
caxis([0 300])
xlim([0 14])

subplot(3,1,3)
Hz = ks/(2*pi);
pcolor(tslide,Hz,rgt_spec.'),
shading interp
xlabel('time(t)')
set(gca,'Fontsize',[10])
colormap(hot)
title('Spectrogram: close up (Hz)');
colorbar
ylabel('frequency (Hz)');
h = colorbar; ylabel(h, 'FFT of filtered signal')
```

```
ylim([400 1400])
caxis([0 400])
xlim([0 14])

% Step Function
figure(8)
sgtitle('Step filter (shannon)  w/ a = 0.2 & translation = 0.2', 'Fontsize', [14])
rst_spec=[];
tslide=[0:0.2: 16];
for ii=1:length(tslide)
    sh = abs(t-tslide(ii)) <= 0.2/2; % Gabor
    rs = sh.*r_clean;
    rst=fft(rs);
    rst_spec=[rst_spec;  abs(fftshift(rst))];
end
%
subplot(2,1,1)
pcolor(tslide,ks,rst_spec.'),
shading interp
xlabel('time(t)')
set(gca,'Fontsize',[12])
colormap(hot)
ylabel('frequency (\omega)');
title('Spectrogram');
colorbar
h = colorbar; ylabel(h, 'FFT of filtered signal')
ylim([-25000 25000])
caxis([0 200])
xlim([0 14])
%
subplot(2,1,2)
Hz = ks/(2*pi);
pcolor(tslide,Hz,rst_spec.'),
shading interp
xlabel('time(t)')
set(gca,'Fontsize',[12])
colormap(hot);
ylabel('frequency (Hz)');
h = colorbar; ylabel(h, 'FFT of filtered signal')
title('Spectrogram: close up (Hz)');
ylim([600 1400])
caxis([0 300])
xlim([0 14])
```

Appendix C: Github

https://github.com/Washington-Turtles/amath582.git