Gábor Transforms in Music Analysis

Alyssa Mell

Abstract

Using Gábor filtering we produce spectrograms in order to study a portion of Handel's Messiah with time-frequency analysis. We study this piece of music with different window widths, translation lengths, and types of Gábor windows in order to get a full understanding of the time-frequency signature of the piece. We then study two additional music files of *Mary had a little lamb* and use Gábor filtering to reproduce their music scores. Finally, we consider the difference in the time-frequency analysis of a piano and a recorder for this song.

Sec. I. Introduction and Overview

The Gábor Transform is a useful tool in analyzing the time-frequency spectrum of different music pieces. The spectrograms produced by this analysis give a visual representation of how the frequency spectrum and amplitude of the notes change over time. Since each type of note on an instrument has a specific frequency signature, this makes it easy to determine which notes are being played at each point in time. From this analysis it is then possible to reproduce the music score of the original piece.

In this paper, we will first study spectrograms of 9 seconds of Handel's Messiah. We create these spectrograms by using multiple Gábor windows with various widths and translation distances in order to better understand the time-frequency signature of the piece. Then we study two different recordings of the song, *Mary had a little lamb*. The first recording is played on the piano and the second is on the recorder. We create and study spectrograms of each piece in order to reproduce their music scores. Finally, we study the difference between the two instruments by analyzing their timbres. For a given center frequency, the timbre is related to the overtones, or the consecutive multiples, of that frequency. By studying these overtones we can make comparisons between the timbres of the two instruments.

Sec. II. Theoretical Background

The main method used for data analysis in this paper is the Gábor transform, which is useful in producing spectrograms. The Gábor transform, as shown in equation 1, is similar to the Fourier transform used in the previous assignment, but it differs in that it provides information about both the time and the frequency of the data, rather than solely providing frequency information. The Heisenberg-Gábor uncertainty principle states that, the more you know about the time or position of a signal, the less you know about its frequency, and vice versa. Therefore, since the Gábor transform provides additional time information, it is not as accurate in its frequency values as the Fourier transform. However, Gábor filtering makes an especially useful tool in analyzing things such as the scores of music pieces where you need to know about the frequency of the signal at different points in time.

There are multiple types of Gábor filters that can be used in analyzing the data. The three main types used in this paper are the Gaussian window, the Mexican hat wavelet, and the Shannon window. The general Gábor transform and each of these filters are defined respectively as follows:

$$F(t,w) = \int_{-\infty}^{\infty} f(\tau) g(\tau - t) e^{-iw\tau} d\tau \tag{1}$$

$$g(\tau - t) = e^{-\alpha(t-\tau)^2} \tag{2}$$

$$g(\tau - t) = (1 - \alpha(t-\tau)^2) e^{-\alpha(t-\tau)^2/2} \tag{3}$$

$$g(\tau - t) = \begin{cases} 1 & a \leq \tau - t \leq b \\ 0 & else \end{cases} \tag{4}$$

For the Gábor transform in equation 1, $g(\tau - t)$ centers the filter at a specific point in time, and w represents the corresponding frequencies at that time. In the Gaussian window defined by equation 2 and the Mexican Hat wavelet defined by equation 3, α represents the width parameter of the filter and τ represents the center. In equation 4, the Shannon window, with width $b - a$ is a uniform function set to equal one when $\tau - t$ is between the values a and b. The different results observed when using these three different filters will be explained in more detail in Sec. IV below.

In order to produce a spectrogram, these filters slide across the data by changing their center position at each point in time. The resulting data is plotted using a psuedocolor plot which allows us to see the changes in frequency of the data over time. The spectrograms provided in this paper have an x-axis representing time and a y-axis representing frequency. If we use a wide filter for our spectrogram, this will provide us with a greater amount of frequency information, but less precision in time. Therefore we will see more distinctive changes in color on the plot when moving up and down on the y-axis, but less horizontal distinction. In contrast, if we use a thinner filter and therefore have more accurate time data, then we will see more distinctive changes in color when moving left and right across the x-axis, but less vertical distinction. These comparisons and more will be plotted and described and in more detail in the sections below.

Sec. III. Algorithm Implementation and Development

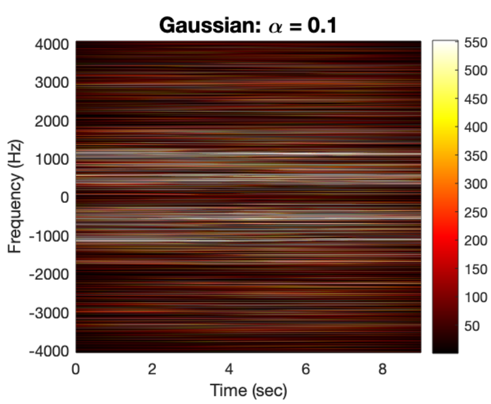
In this section there will often be references to the MATLAB code which is located in Appendix B. The specific line of the code that is being referenced will be stated at the end of the sentence. The frequency units used throughout the code is set to be in Hertz since this is how the provided music scale was measured.

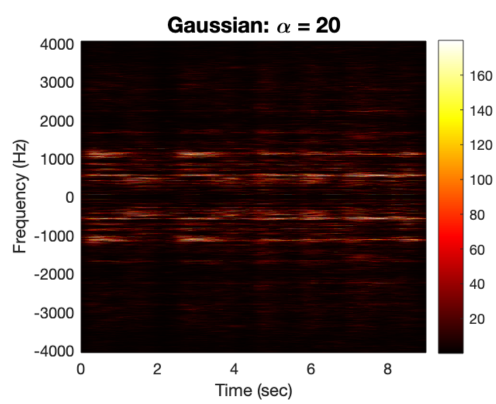
The first part of this problem required an open-ended analysis of Handel's Messiah. In order to analyze this piece, we created multiple spectrograms in MATLAB using the three types of windows defined above in equations 2, 3, and 4. These equations acted as the filter functions that were multiplied by the signal at each point in time. The general method for creating a spectrogram for each of these types of filters was the same.

In order to create the spectrograms, a for loop was created and the center of the filter function was shifted over at each time iteration until the entire recording was covered. For the Gaussian filter and the Mexican Hat wavelet, this was done by changing τ to shift the center of the filter over the entire time interval, with the values of τ denoted as tslide() (lines 38, 64). For the Shannon step function, this was done by creating a vector full of zeros and ones and shifting the positions of the ones across the vector with each step forward in time (lines 91). This vector was created such that it would cover the entire length of the given signal (line 90). At the end of each iteration, the filtered and transformed data was placed into a vector that held the updated data for each time step. After data was collected for the entire time interval, it was plotted using pcolor() and colormap(hot) which showed the higher intensity signals as a bright yellow and lower intensity signals as dark red for all frequencies and points in time. The filters were used with various widths and translation distances in order to get a good number of spectrograms for analysis. These are included and interpreted in Sec. IV.

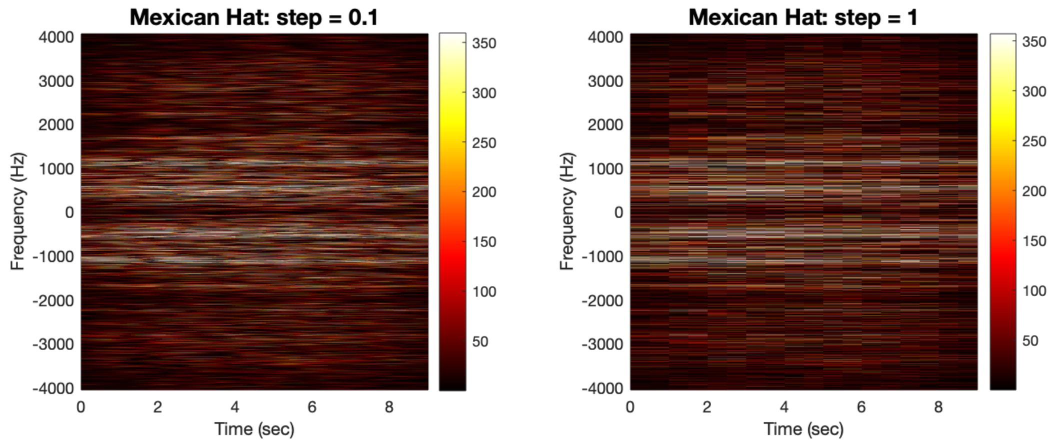
The methods used to analyze the song *Mary had a little lamb* were very similar to those mentioned above. A Gaussian filter was used in creating a spectrogram for both the piano and recorder data (lines 151, 189). These spectrograms were plotted using appropriate y-axis limits so that the overtones would not appear in the plot, therefore producing a clean score (see Figures 5 and 6).

Sec. IV. Computational Results





Figures 1 and 2: These plots show the Gaussian filter spectrogram with two different widths, smaller α values mean greater width. Therefore Figure 1 on the left has a filter with a greater width.



Figures 3 and 4: These plots show the Mexican Hat wavelet spectrogram with two different translation distances. The step size represents the translation distance, or change in tau at each time. Therefore Figure 3 on the left has a smaller translation distance.

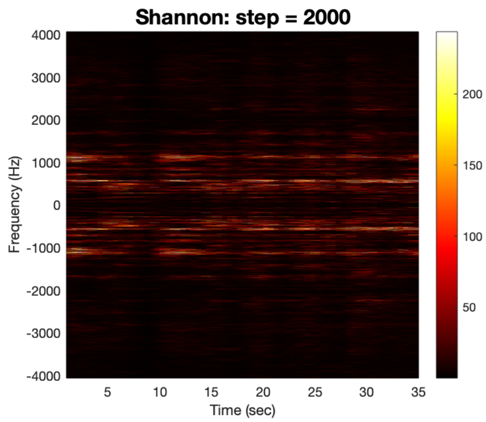
Figures 1 through 6 show the different spectrograms that were seen for Handel's Messiah when changing the types of Gábor filters used, their window widths, and their translation distance. When changing the translation distance of the Gaussian filter, it produced very similar results to those shown in Figure 3 and 4, so that figure was not included in this paper. Also, when changing the width of the Mexican hat wavelet, it produces very similar figures to those shown in Figures 1 and 2, so those plots are also not included in this paper. The brighter the point on the spectrogram, the stronger the signal. In this case, greater signal strength represents a louder signal at that frequency or time.

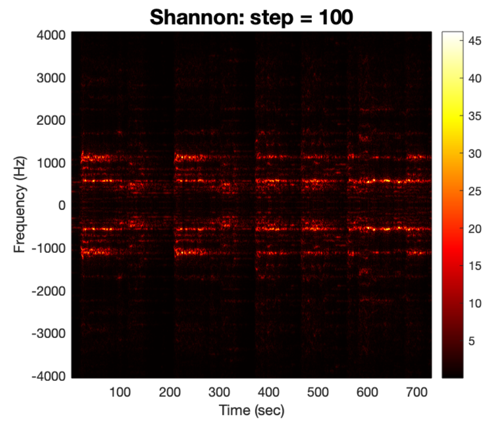
From these plots we can determine that when we decrease the filter width such as in Figure 2, the amount of information that we know about the frequencies decreases. This is clear by observing how the top and bottom of the plot go dark in Figure 2 compared to Figure 1, showing less range in frequency values. However, this also increases what we know about the time that the frequencies occur. The smaller filter width begins to show some darker vertical lines which indicate that there is nothing being played at that point in time.

Figures 3 and 4 show another interesting variation of the filter. Here, figure 4 has a greater translation length, so there is less overlapping data at each point in time. Our data actually appears to be somewhat disjoint in Figure 4 where there are clear color divisions as the intensity of each frequency changes over time. This is seen at even intervals, where certain times are skipped over, causing the color changes in the plot to lack uniformity.

Figures 5 and 6 show a Shannon filter with different widths. The smaller width in Figure 6 provides more time information because the dark vertical lines separating the frequencies at different points in time are more distinct.

In our analysis of *Mary had a little lamb*, the spectrograms in Figures 7 and 8 produced a very clear distinction between notes being played. These notes appear at frequencies around 260Hz to 320Hz for the piano and around 800Hz to 1000Hz for the recorder. This difference is due to the two instruments playing different scores.





Figures 5 and 6: These plots show the Shannon step-function spectrogram with two different widths and step sizes. The larger step size also has a larger width, so it covers all of the data in fewer steps. Figure 5 on the left has a larger width and step size. Figure 5 has width 4000, and figure 6 has width 200.

From our analysis of *Mary had a little lamb*, given the frequencies in the spectrogram in Figure 7, the music score for the piano can be reproduced as: EDCD EEE DDD EEE EDCD EEEE DDEDC. Notes that are being held longer will have a larger blank space following them. Therefore we can determine that the notes written in black above last approximately 0.5 seconds, the notes in red last approximately 1 second, and the note in green lasts approximately 2 seconds.

Given the frequencies in the spectrogram in Figure 8, the music score for the recorder can be reproduced as: BAGA BBB AAA BBB BAGA BBBB AABAG. The total length of the recording for the recorder was slightly shorter than the piano. Therefore, the length of each note should be slightly less, but they will still follow the same general pattern as the piano, with the notes in black being the shortest, then red, and then green.

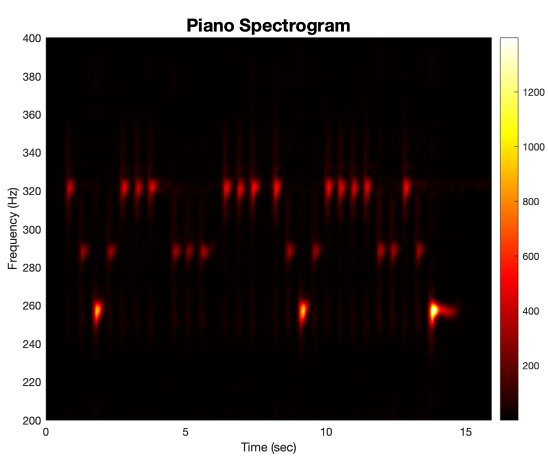


Figure 7: This plot shows the Gaussian filtered spectrogram for the piano.

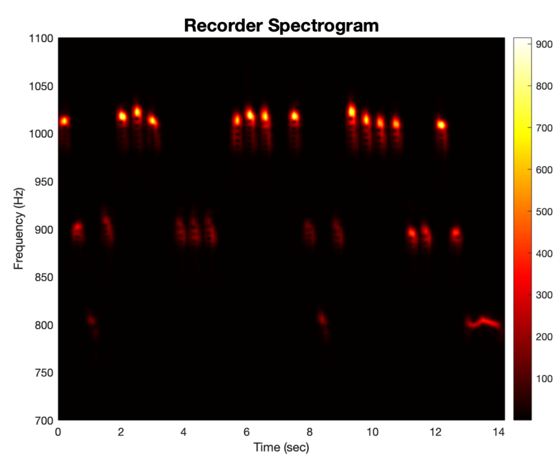


Figure 8: This plot shows the Gaussian filtered spectrogram for the recorder.

Sec. V. Summary and Conclusions

From analyzing Handel's Messiah, we can conclude that increasing the width of a filter increases the amount of information we have about frequencies but reduces our precision in time. Also, if time increments are made to translate too far with each step, graphs can lose uniformity since some times will be skipped over. This causes the intensity of the frequencies at certain times to no longer match up. Overall, the Gaussian filter and Mexican Hat filter appeared to produce similar spectrograms, while the Shannon filter seemed to provide more time information than frequency at the parameters we provided.

A couple important differences to note between the recorder and the piano can be seen in the spectrograms of Figure 7 and Figure 8. Despite playing the same song, the recorder and piano cover very different ranges of frequencies, with the lowest and highest frequencies separating each other by about 200Hz for the recorder, and 60Hz for the piano. Therefore the spacing between the overtones will be greater for the recorder. Also, the recorder emphasizes high frequencies while the piano emphasizes lower ones. So, even if the two instruments were played at the exact same frequency, the piano would be have stronger low overtones while the recorder would have stronger high overtones. This is a fundamental difference in the timbre of the two instruments.

Appendix A. MATLAB functions

Listed below are the important MATLAB functions used in this paper, along with a brief explanation of how they are implemented.

- [Y, FS] = `audioread`(FILENAME) : reads an audiofile and return the sampled data in Y and the sample rate in FS.

- `audioplayer`(Y, FS) : creates an audioplayer object for the given signal Y, and sample rate FS.

- `playblocking`(OBJ) : plays a given audio sample object from the beginning.

- `pcolor`(X,Y,C) : creates a pseudocolor plot of C on a grid defined by X and Y. A pseudocolor plot is a flat surface plot viewed from above.

- `colormap`(hot) : sets a red color scheme for the pseudocolor plot.

- `colorbar` : provides a scale for the colors used in the colormap.

- `shading` interp : controls the color shading of pcolor(), interp sets the shading to interpolated. This means the color in each segment varies linearly and is interpolated at the end.

Appendix B. MATLAB Codes

```
%% Part 1 Data
clear; close all; clc

% Plots portion of music that will be analyzed
load handel
v = y';

% Plots music signal
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');


% Plays the music
%p8 = audioplayer(v,Fs);
%playblocking(p8);

% Initializes variables for length, time, frequency, and number of
    modes
L=9; % seconds
n=length(v);
t2=linspace(0,L,n+1); t=t2(1:n);
k=(1/L)*[0:(n-1)/2 -(n-1)/2:-1]; % frequency measures in hertz
ks=fftshift(k);

%% Gaussian Filter Spectrogram
% Uses a guassian filter to create a spectrogram of the music
    frequencies
% over time. Changes the width of the filter for each iteration.
% In order to test for different translation distances, replace
    a_vec with
% a = 1 and change the tslide interval at each iteration as done
    for the
% Mexican Hat wavelet below.
a_vec = [0.1 20];
for ii = 1:length(a_vec)
    a = a_vec(ii);
    tslide=0:0.1:L;
    vgt_spec = zeros(length(tslide),n);
    for jj=1:length(tslide)
        g = exp(-a*(t-tslide(jj)).^2);
```

```
        vg=g.*v;
        vgt=fft(vg);
        vgt_spec(jj,:) = fftshift(abs(vgt));
    end
    subplot(1,length(a_vec),ii)
    pcolor(tslide,ks,vgt_spec.')
    shading interp
    title(['Gaussian: a = ',num2str(a)],'Fontsize',14)
    xlabel('Time (sec)'); ylabel('Frequency (Hz)');
    colorbar
    colormap(hot)
end

%% Mexican Hat Filter Spectrogram
% Uses a Mexican Hat filter to create a spectrogram of the music frequencies
% over time. Changes the translational distance of the filter for each
% iteration.
% In order to test for different widths, replace t_vec with t = 0.1 and
% change the a value at each iteration as done for the gaussian filter
% above.
t_vec = [0.1 1];
for ii = 1:length(t_vec)
    tslide=0:t_vec(ii):L;
    vgt_spec = zeros(length(tslide),n);
    for jj=1:length(tslide)
        g =(1-(t-tslide(jj)).^2).*exp(-((t-tslide(jj)).^2)/2);
        % Use this instead when changing width of Mexican Hat filter
        % g =(1-a*(t-tslide(jj)).^2).*exp(-(a*(t-tslide(jj)).^2)/2);
        vg=g.*v;
        vgt=fft(vg);
        vgt_spec(jj,:) = fftshift(abs(vgt));
    end
    subplot(1,length(t_vec),ii)
    pcolor(tslide,ks,vgt_spec.')
    shading interp
    title(['Mexican Hat: t = ',num2str(t_vec(ii))],'Fontsize',14)
    xlabel('Time (sec)'); ylabel('Frequency (Hz)');
    colorbar
    colormap(hot)
```

```
end

%% Shannon Step-Function Filter Spectrogram
% Uses a Shannon Step-Function filter to create a spectrogram of the music
% frequencies over time.

% Creates plot using filter width 4000 and step size 2000.
step_1 = 1;
step_2 = 4000;
step = 2000;
vgt_spec = zeros(35,n);
for jj=1:35
    g = zeros(1,length(v));
    g(step_1:step_2) = 1;
    vg=g.*v;
    vgt=fft(vg);
    vgt_spec(jj,:) = fftshift(abs(vgt));
    step_1 = step_1 + 2000;
    step_2 = step_2 + 2000;
end
subplot(1,2,1)
pcolor(1:35,ks,vgt_spec.')
shading interp
title('Shannon: step = 2000','Fontsize',16)
xlabel('Time (sec)'); ylabel('Frequency (Hz)');
colorbar
colormap(hot)

% Creates plot using filter width 200 and step size 100.
step_1 = 1;
step_2 = 200;
vgt_spec = zeros(729,n);
for jj=1:729
    g = zeros(1,length(v));
    g(step_1:step_2) = 1;
    vg=g.*v;
    vgt=fft(vg);
    vgt_spec(jj,:) = fftshift(abs(vgt));
    step_1 = step_1 + 100;
    step_2 = step_2 + 100;
end
subplot(1,2,2);
pcolor(1:729,ks,vgt_spec.')
shading interp
```

```
title('Shannon: step = 100','Fontsize',16)
xlabel('Time (sec)'); ylabel('Frequency (Hz)');
colorbar
colormap(hot)

%% Part 2 - Piano
clear; close all; clc

% Reads in the audio file and stores and plots the data.
[y,Fs] = audioread('music1.wav');
v = y.';
tr_piano=length(y)/Fs;  % record time in seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (piano)');
p8 = audioplayer(y,Fs); playblocking(p8); % Plays the music

% Initializes variables for length, time, frequency, and number of modes
L=tr_piano; n=length(v);
t2=linspace(0,L,n+1);
t=t2(1:n);
k=(1/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);

% Creates a spectrogram for the piano using a Gaussian filter.
a = 100; % filter width parameter
tslide=0:0.1:L; % translation distance
vgt_spec = zeros(length(tslide),n);
for jj=1:length(tslide)
    g = exp(-a*(t-tslide(jj)).^2);
    vg=g.*v;
    vgt=fft(vg);
    vgt_spec(jj,:) = fftshift(abs(vgt));
end
pcolor(tslide,ks,vgt_spec.')
shading interp
title('Piano Spectrogram','Fontsize',16)
ylim([200 400]); % sets frequency limits for graph to remove overtones
xlabel('Time (sec)'); ylabel('Frequency (Hz)');
colorbar
colormap(hot)

%% Part 2 - Recorder
```

```
clear; close all; clc

% Reads in the audio file and stores and plots the data.
[y,Fs] = audioread('music2.wav');
v = y.';
tr_rec=length(y)/Fs;  % record time in seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)');
p8 = audioplayer(y,Fs); playblocking(p8); % Plays the music

% Initializes variables for length, time, frequency, and number of
    modes
L=tr_rec; n=length(v);
t2=linspace(0,L,n+1);
t=t2(1:n);
k=(1/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);
vt = fft(v);

% Creates a spectrogram for the recorder using a Gaussian filter.
a = 100; % filter width parameter
tslide=0:0.1:L; % translation distance
vgt_spec = zeros(length(tslide),n);
for jj=1:length(tslide)
    g = exp(-a*(t-tslide(jj)).^2);
    vg=g.*v;
    vgt=fft(vg);
    vgt_spec(jj,:) = fftshift(abs(vgt));
end
pcolor(tslide,ks,vgt_spec.')
shading interp
title('Recorder Spectrogram','Fontsize',16)
ylim([700 1100]); % sets frequency limits for graph to remove
    overtones
xlabel('Time (sec)'); ylabel('Frequency (Hz)');
colorbar
colormap(hot)
```