Getting Back in Tune: Time-Frequency Analysis and Music

Timothy Lu

Abstract

This report follows the process of applying Gábor transform, a windowed Fourier transform, in order to perform time-frequency analysis on three different audio files. First, the differences in filtered data derived from various wavelet formulas were observed. Then, effects of varying width and over/under sampling in the Gábor transform were observed on frequency data by plotting the data on spectrograms and observing the differences. Then we apply the Gábor transforms in order to observe timbre through the overtones present and derive a score in the form of frequency data.

1. Introduction and Overview

Usually whenever music is used in sound editing software, it is displayed as a wave – with time and amplitude being the defining points of data visualized. While this data gives us an idea of what the music will sound like, frequency data could also be helpful to unearth where the music came from. This does not only mean the score (the collection of musical notes) that comprised a given song, but the timbre of the instrument(s) playing the song as well.

The following is an application of time-frequency analysis of three different audio files in order to:

1) Use a 9-second snippet of Handel's Messiah (an audio file already distributed with MATLAB) to test the functionality and limitations of Gábor filters.
2) Derive the original score of music from two audio files playing Mary Had a Little Lamb (in .wav format).
3) Analyze the timbre of two different instruments using time-frequency data derived from the two files in step (2).

To accomplish these steps, we will use Fourier Transforms to move data into the frequency domain, then apply Gábor filters (in the form of wavelets) to unearth the frequencies associated with each time slice.

2. Theoretical Background

The Fourier Transform is a form of spectral transform associated with a *Fourier Series* which "represents functions and their derivatives as sums of cosines and sines" (Kutz, 25). The *Fast-Fourier Transform* (FFT) is an operation which performs a Fourier Transform in *O(N log N)*. While the Fourier Transform is a powerful tool for signal processing, a limitation lies in that "the transform fails to capture the moment in time when various frequencies were actually exhibited" (Kutz, 48). Thus, in order to capture these differences in frequency over time, the Fourier Transform can be taken over smaller windows of time.

2.1 Gábor Transform and Basic Wavelets

The *Gábor transform* is a windowed Fourier transform method that permits us "a fast an easy way to analysis both the time and frequency properties of a given signal" (Kutz, 68). Also known as the *short-time Fourier transform (STFT)*, the transform is defined as:

$$G[f](t,\omega) = \bar{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)\bar{g}(\tau - t)e^{-i\omega\tau}d\tau = (f, \bar{g}_{t,\omega}) \tag{1}$$

where the bar denotes the complex conjugate of the function (Kutz, 50). $g(\tau - t)$ then, is a filter that is used to localize the overall signal at each window of time. It is worth noting that there are other simple windowed Fourier transforms such as the *Zak transform,* or the *Wagner-Ville Distribution* (Kutz, 55).

Some simple Gábor windows, all used in this report, are a *Gaussian time-filter,* the *Mexican hat wavelet*, and a *Shannon filter* (or a step function). The *Gaussian time-filter*:

$$F(t) = \exp(-\tau(t - t_0)^2) \tag{2}$$

is a simple gaussian filter where the filter where τ represents the width of the filter and $t - t_0$ is the center of the filter (or where the filter is localized).

The *Mexican Hat Wavelet* is "a second moment of a Gaussian in the frequency domain" (Kutz, 59). Called the Mexican hat because of its shape, it is defined as the following:

$$\psi(t) = (1 - b * (t - t_0)^2) * \exp(-\tau * (t - t_0)^2) \tag{3}$$

Where τ and $t - t_0$ are the same as the gaussian time-filter, representing width and location respectively. b represents how far the "lip of the hat" dips.

A *Shannon filter* is just a step function that returns 1 within a given width around the center and 0 otherwise. It can be represented as the following:

$$F(t) = \begin{cases} if\ abs(t - center) \leq 2 * width & 1 \\ else & 0 \end{cases} \tag{4}$$

Overtones in music are tones that are playing "over" the central frequency. For our purposes, overtones are tones that are an "octave" (or given a frequency ω_0, scaled frequencies of ω_0 such as $2\omega_0$ or $3\omega_0$).

2.2 Spectrograms

A *spectrogram* is a way to "represent (a) signal in both the time and frequency domain" (Kutz, 68). Spectrograms are a representation of frequency over time. For our purposes, spectrograms will be constructed from data from Handel's Messiah as well as the two audio files of Mary Had a Little Lamb, in order to examine the frequency data in each moment of time. This visualization can be a way to represent the score of our music as well (with frequencies or notes plotted versus time).

3. Algorithm Implementation and Development

We are given a few points of data for each audio file: sample rate, amplitude, and duration of the audio. As stated in part 1, Handel's Messiah will be analyzed differently to the two recordings of Mary Had A Little Lamb. However, all audio files will require the same basic steps:

1) Read in the data, extract the length of the audio file, the number of data points, sample rate, and derive the time steps corresponding to each amplitude data point.
2) Determine a time translation and width to apply for Gábor filtering.
3) For each time value, filter the data, and take the FFT of the filtered data. Store the absolute value of the FFT output in a matrix.
4) Plot the output of the FFT in a spectrogram.

To set up, after extracting sample rate, duration, number of points from the audio files, we must setup a range of frequency values. These are scaled by $\frac{2\pi}{L}$ as the FFT operation assumes 2π periodic signals.

The time translation is a unit (in seconds) by which we increment in every iteration of the loop. This serves as the center of our filters. This variable is `tslide(j)` in Appendix B. In the loop, we center the filter around the current time step, then we multiply the filter by our unfiltered data, such that we have a window of the data.

We then take the FFT of the filtered data and store the windowed FFT into a matrix of values for our spectrogram.

The Spectrogram is constructed from the range of frequency values we created as the y axis, with time as the x axis and the transpose of the windowed FFT data as the color data (with more intense color where there are positive values in frequency). We transpose this matrix such that the axes line up with the range of frequency values we created.

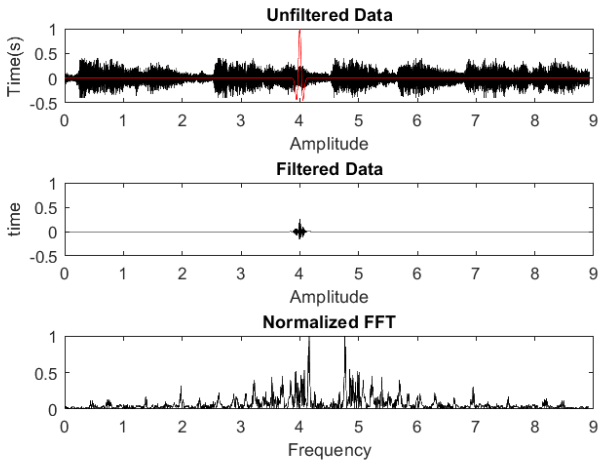

Figure 1 - Visualization of the Mexican hat wavelet applied to data

4. Computational Results

Computational results, for these purposes, will consist of data read into our algorithm, and spectrograms outputted from the Gábor filtering as well as observations about the data drawn from the spectrograms.

4.1 Handel's Messiah

The audio file for Handel's Messiah built into MATLAB ran for roughly 9 seconds with 73113 points of amplitude data (cut down to 73112 points to work around the FFT) and a sample rate of 8192 Hz.

In order to check the effects of difference in width, we altered τ (in the following figures denoted as a) from equation (2). The spectrograms are depicted in figure 2. What we notice is that with a larger width (a smaller value of τ) the time-frequency data starts to meld together. This is reminiscent of a straight Fourier Transform of the data without clear windows. With a smaller filter, and thus a smaller window, there are clear differences in step. The resolution in both the time and frequency domain increases with a more precise filter width.

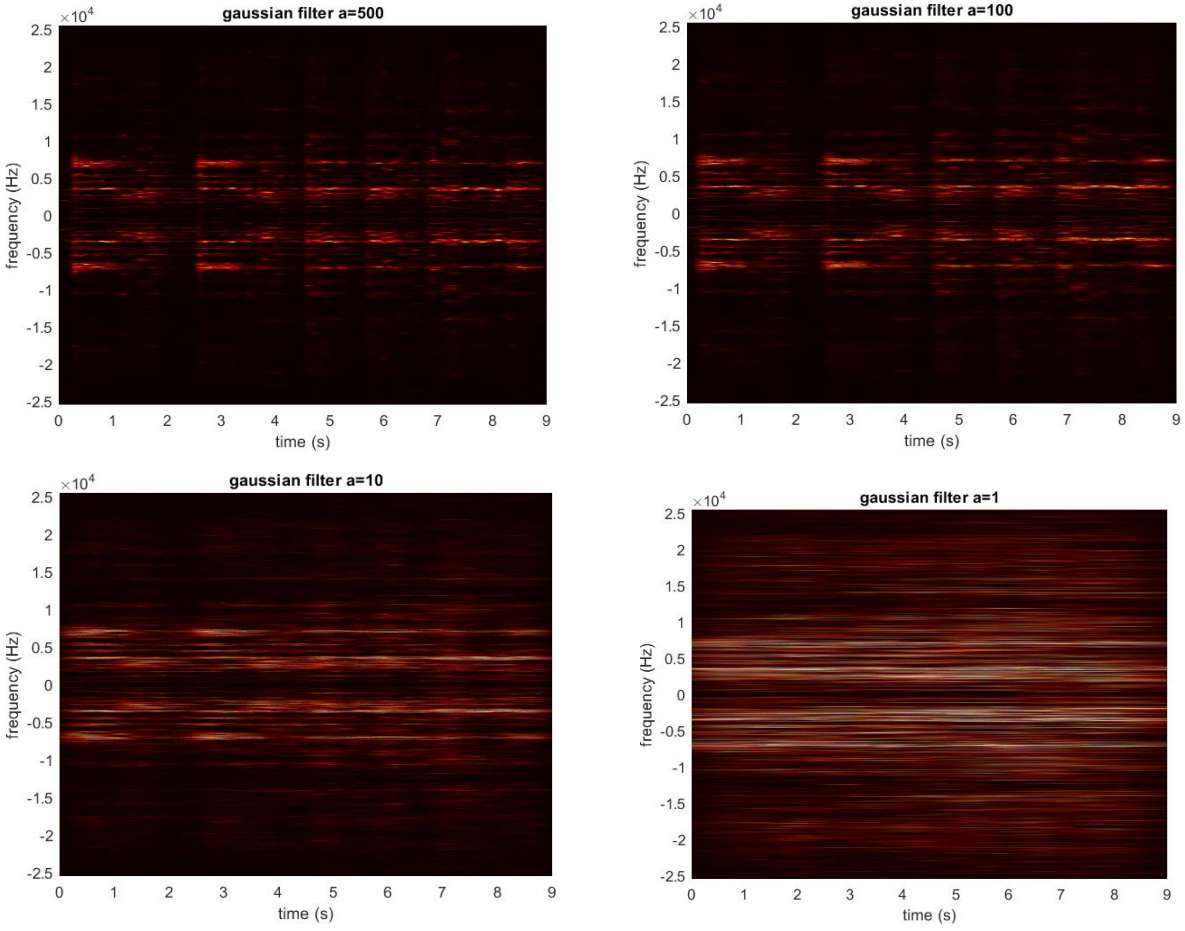

Figure 2 - Effects on Gábor filtering due to difference in a (tau)

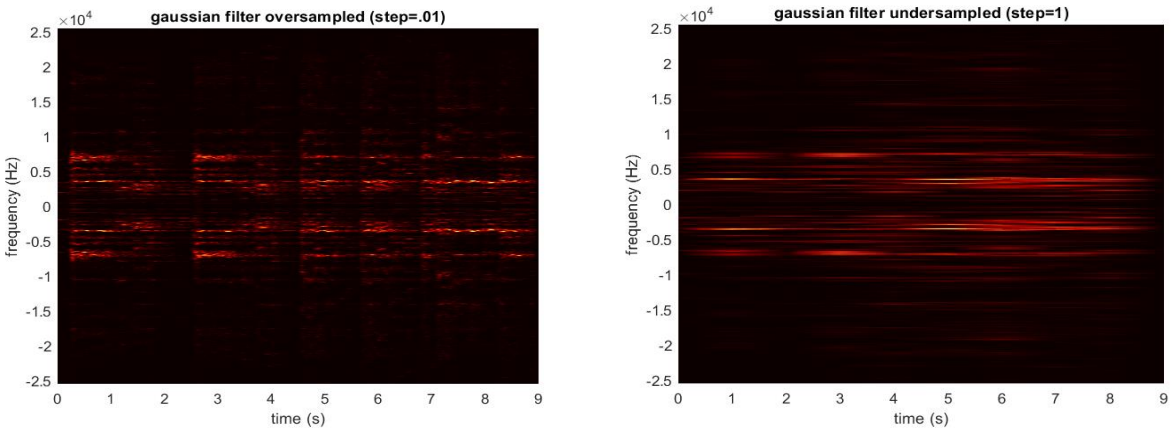

Figure 3 - Oversampling and under sampling (small time translations versus large time translations)

When examining the effects of over/under sampling – or using time steps that are too large or too small, we see time steps that are too small are computationally inefficient, while time steps that are too large give insufficient time resolution. The results of this experiment are in figure 3. Each of these used a width of $\tau = 500$.

When analyzing the difference in spectrogram created by differing wavelets, we explored the effects of different wavelet filters on the results of our filtering. The Shannon filter used a width of $width = 0.25$ as specified in equation (4). The Mexican Hat wavelet used a width of $\tau = 500$ and $b = 1000$. The results are plotted in Figure 4.

From our spectrograms we can conclude that, although the Mexican hat and Shannon step function produce distinctly different sets of data when put through FFT, if the width is narrow enough, the time-frequency data returned is relatively similar in terms of shape and resolution.

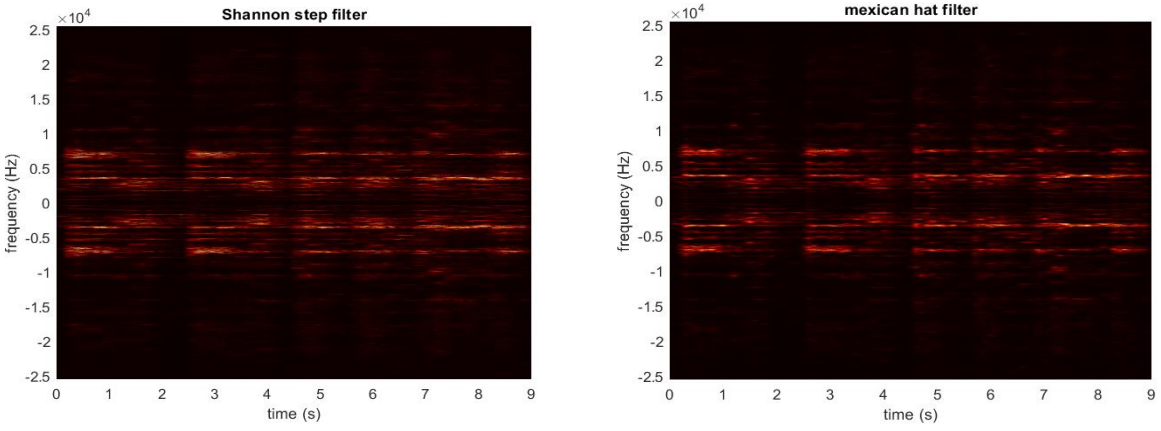

Figure 4 - Mexican Hat wavelet (left) versus Shannon step function (right)

4.2 Mary Had a Little Lamb

For these two pieces, the data that was read in/given for the piano and recorder audio files are as denoted in figure 5.

Instrument	Duration (sec)	Sample rate (Hz)	Number of data points
Piano	16	43840	701440
Recorder	14	44837	627712

Figure 5 - Summary of data read in from Mary Had a Little Lamb audio files

We then applied the basic algorithm as described in section 3, with a gaussian filter of width $\tau = 500$. The time for the piano piece was 0.1 and the time step for the recorder piece was 0.2. Two noticeable observations about the spectrograms are the concentrated frequencies at 250 to 400 Hz (for the piano), and 750 to 1050 HZ (for the recorder). Both the piano and recorder frequency data display significant overtones. Moreover, the overtones in piano are relatively even in intensity, while the overtones with the recorder audio were more intense at a higher frequency. This imbalance in intensity creates a different timbre, or tone of the instrument.

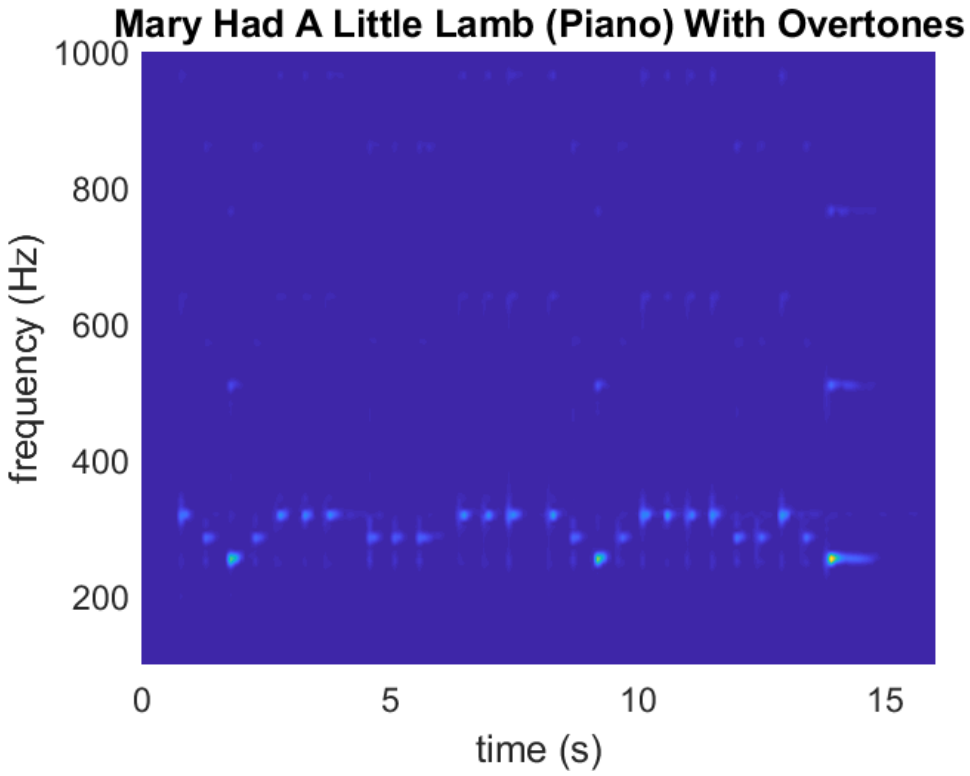

Figure 6 - Piano Score with overtones

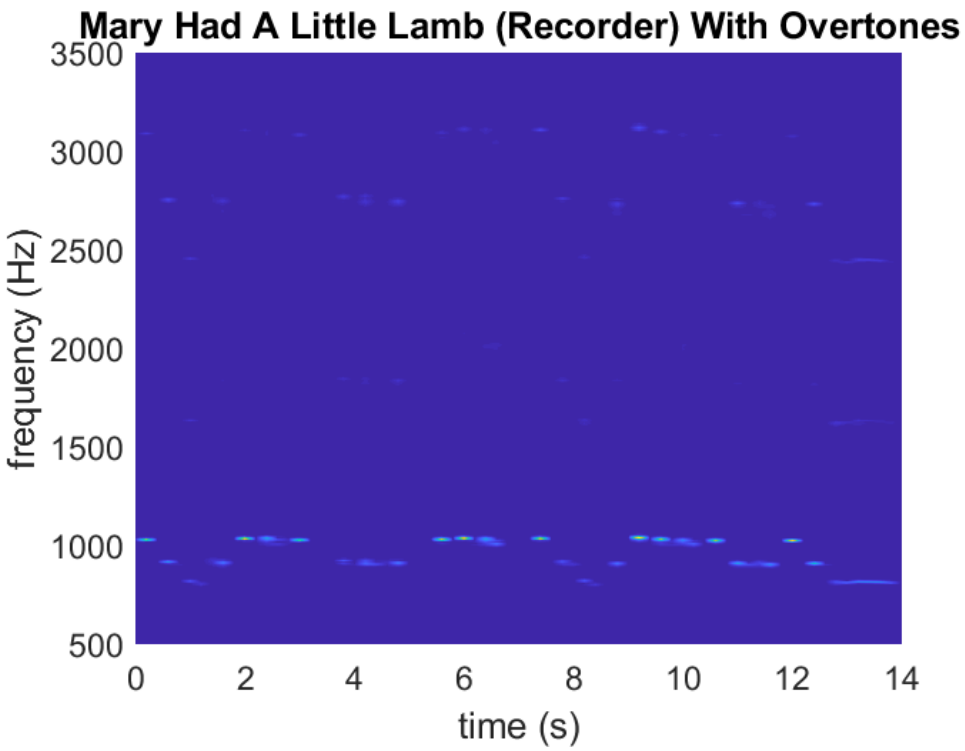

Figure 7 - Recorder score with overtones

V. Summary and Conclusions

A few conclusions can be drawn from the different audio files. The first audio file we analyzed was examined for the effects of changing various properties of a Gábor window such as time translation or width of a filter. While smaller time translations or oversampling lead to great computational burden on the machine, under-sampling lead to poor time resolution. A large width of a filter lead to poor time resolution again. The conclusion we draw from this, is that the parameters used to filter data for time-frequency analysis must fit the data such that there is ample time resolution as well as decent computational speed.

The other audio files were compared for their difference in instrument. First, we examined the central frequencies for each time step in the filtering process. This gave us a score which we plotted in spectrograms (figure 6,7). From these frequencies and the frequencies of notes (MIT), we can recreate the exact notes that were being played on each instrument as depicted in figure 8 and 9. We notice that the recorder has generally higher frequencies and is playing different notes as the piano.

Figure 6 and 7 show the difference in timbre between the recorder and piano through the difference in overtones. The piano has more pronounced overtones which are equal in intensity. This is contrasted with the timbre of the recorder, where the overtones for higher frequencies (3ω) are significantly more intense than the frequencies of the overtones with lover frequencies. This causes the sound of the recorder to be more “harsh” as there are significantly pronounced overtones.

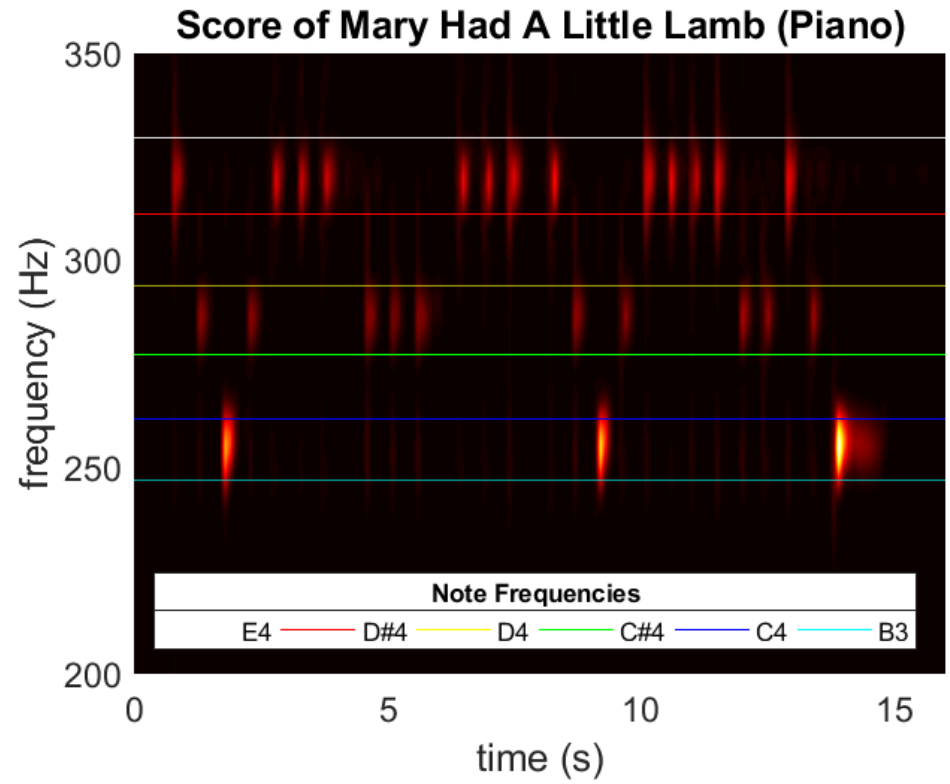

Figure 8 - Score of Mary Had a Little Lamb (Piano)

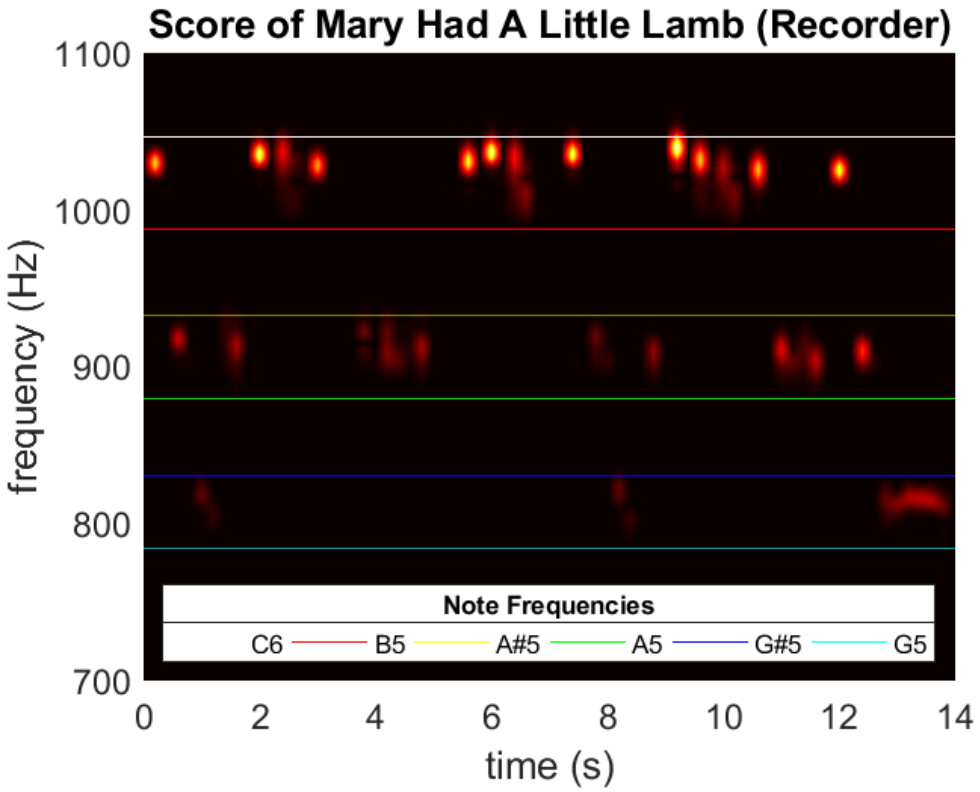

Figure 9 - Score of Mary Had a Little Lamb (Recorder)

References

1) Kutz, *AMATH 582 Notes Part 3*
2) *What Is Computer Science?*, pages.mtu.edu/~suits/notefreqs.html.

Appendix A

Command	Implementation
plot	Used to plot lines of common frequencies.
audioread	Used to read in the data from our audiofiles. “load” was used for Handel’s Messiah.
fft	Used to Fourier Transform our data into the frequency domain.
fftshift	Used to shift our frequency data drawn from fft into a position we can use.
pcolor	Plots the data into a spectrogram.
colormap	Changes the color of the spectrogram for visibility.
abs	Gives the absolute value of an input. Used when applying fftshift. Also used to adjust the Shannon window.
length	Gives the largest index of a matrix/vector. Used to extract the number of data measurements in a given audio file or list of objects.
exp	Used to represent the gaussian filter and Mexican hat filter.

Appendix B

1. Read data from Handel's Messiah

```
clear all; close all; clc;

load handel
v = y'/2;
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');

v = v(1:length(v) - 1);
L = 9;
n = length(v);
t = (1:length(v))/Fs;


k=(2*pi/L)*[0:n/2-1 -n/2:-1]; ks=fftshift(k);
%p8 = audioplayer(v,Fs);
%playblocking(p8);
```

2.Gaussian filter applied to Handel's Messiah

```
%% gaussian filtering on Handel
Vgt_spec=[];

step = L/90;
tslide=0:.01:L;

a = 500;

for j=1:length(tslide)
    tf = t-tslide(j);

    g=exp(-a*tf.^2); % Gaussian

    Vg=g.*v; Vgt=fft(Vg);

    Vgt_spec=[Vgt_spec; abs(fftshift(Vgt))];

    subplot(3,1,1), plot(t,v,'k',t,g,'r')
    axis([0 L -0.5 1]);
    subplot(3,1,2), plot(t,Vg,'k')
    axis([0 L -0.5 1]);
    subplot(3,1,3), plot(t,abs(fftshift(Vgt))/max(abs(Vgt)),'k')
    drawnow
    pause(0.01)
end


subplot(1,1,1), pcolor(tslide,ks,Vgt_spec.'), shading interp
colormap(hot)
title("mexican hat filter")
xlabel("time (s)");
ylabel("frequency (Hz)");
```

3.Mexican hat wavelet filter applied to Handel's Messiah

```
%% Mexican Hat

Vgt_spec=[];

step = L/90;
tslide=0:.1:L;

a = 500;
b = 1000;

for j=1:length(tslide)
    tf = t-tslide(j);

    g=(1-b*tf.^2).*exp(-a*tf.^2); % Mexican hat

    Vg=g.*v; Vgt=fft(Vg);

    Vgt_spec=[Vgt_spec; abs(fftshift(Vgt))];

    subplot(3,1,1), plot(t,v,'k',t,g,'r')
    axis([0 L -0.5 1]);
    title('Unfiltered Data')
    xlabel('Amplitude')
    ylabel('Time(s)')
    subplot(3,1,2), plot(t,Vg,'k')
    axis([0 L -0.5 1]);
    title('Filtered Data')
    xlabel('Amplitude')
    ylabel('time')
    subplot(3,1,3), plot(t,abs(fftshift(Vgt))/max(abs(Vgt)),'k')
    title('Normalized FFT')
    xlabel('Frequency')
    ylabel('Amplitude')
    drawnow
    pause(0.01)
end



subplot(1,1,1),pcolor(tslide,ks,Vgt_spec.'), shading interp
colormap(hot)
title("mexican hat filter");

xlabel("time (s)");
ylabel("frequency (Hz)");
```

4.Shannon filter applied to Handel's Messiah

```
%% Shannon

Vgt_spec=[];

step = L/90;
tslide=0:.1:L;

width=0.25;
idxWidth = round(width/2/(L/n));

for j=1:length(tslide)
    tf = t-tslide(j);

    g=abs(tf) <= width/2;

    Vg=g.*v; Vgt=fft(Vg);

    Vgt_spec=[Vgt_spec; abs(fftshift(Vgt))];

    subplot(3,1,1), plot(t,v,'k',t,g,'r')
    axis([0 L -0.5 1]);
    subplot(3,1,2), plot(t,Vg,'k')
    axis([0 L -0.5 1]);
    subplot(3,1,3), plot(t,abs(fftshift(Vgt))/max(abs(Vgt)),'k')
    drawnow
    pause(0.01)
end



subplot(1,1,1),pcolor(tslide,ks,Vgt_spec.'), shading interp
colormap(hot)
title("Shannon step filter");

xlabel("time (s)");
ylabel("frequency (Hz)");
```

5.Reading and filtering music1.wav

```
%% part 2

clear all; close all; clc;

tr_piano=16; % record time in seconds
y=audioread('music1.wav'); Fs=length(y)/tr_piano;
plot((1:length(y))/Fs,y)
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (piano)'); drawnow
%p8 = audioplayer(y,Fs); playblocking(p8);
%figure(2)

v = y'/2;
v = v(1:length(v));
L = length(v)/Fs;
n = length(v);
t = (1:length(v))/Fs;

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

%% gabor filtering
Vgt_spec=[];

step = .1;
tslide=0:step:L;

tau = 500;


for j=1:length(tslide)
    g=exp(-tau*(t-tslide(j)).^2); % Gaussian

    Vg=g.*v; Vgt=fft(Vg);

    Vgt_spec=[Vgt_spec; abs(fftshift(Vgt))];

    subplot(3,1,1), plot(t,v,'k',t,g,'r')
    axis([0 L -0.5 1]);
    subplot(3,1,2), plot(t,Vg,'k')
    axis([0 L -0.5 1]);
    subplot(3,1,3), plot(ks,abs(fftshift(Vgt)),'k')
    drawnow
    pause(0.01)
end
```

6. Spectrogram and Score of music1.wav

```
%% Spectogram - Gaussian


%Vgt_spec = Vgt_spec./max(abs(Vgt_spec));

subplot(1,1,1), pcolor(tslide,ks/(2*pi),Vgt_spec.'), shading interp
%color(hot)

set(gca,{'Ylim', 'Fontsize'}, {[100 1000],  [14]})

title("Mary Had A Little Lamb (Piano) With Overtones")
xlabel("time (s)");
ylabel("frequency (Hz)");
%% Score the music

pcolor(tslide,ks/(2*pi),Vgt_spec.'), shading interp
colormap(hot);
hold on
pb= plot([0 16],[246.94 246.94],'c') %B_3 freq
hold on;
pc = plot([0 16],[261.63 261.63],'b'); %C_4 freq
hold on;
pcs=plot([0 16],[277.18 277.18],'g') %C#_4 freq
hold on
pd=plot([0 16],[293.66   293.66],'y'); %D_4 freq
hold on;
pds=plot([0 16],[311.13  311.13 ],'r') %D#4 freq
hold on;
pe = plot([0 16],[329.63 329.63],'w') %E_4 freq

set(gca,'Ylim', [200 350],'Fontsize',[14])
title("Score of Mary Had A Little Lamb (Piano)")
xlabel("time (s)");
ylabel("frequency (Hz)");

lgd = legend([pe, pds, pd, pcs, pc, pb],'E4','D#4','D4','C#4','C4','B3');
lgd.FontSize = 10;
lgd.Title.String='Note Frequencies';
```

7. Reading and filtering music2.wav

```
%% Recorder
clear all; close all; clc

tr_rec=14; % record time in seconds
y=audioread('music2.wav'); Fs=length(y)/tr_rec;
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)');
p8 = audioplayer(y,Fs); playblocking(p8);

v = y'/2;
v = v(1:length(v));
L = length(v)/Fs;
n = length(v);
t = (1:length(v))/Fs;

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

%% gabor filtering
Vgt_spec=[];

step = .2;
tslide=0:step:L;

tau = 500;


for j=1:length(tslide)
    g=exp(-tau*(t-tslide(j)).^2); % Gaussian

    Vg=g.*v; Vgt=fft(Vg);

    Vgt_spec=[Vgt_spec; abs(fftshift(Vgt))];

    subplot(3,1,1), plot(t,v,'k',t,g,'r')
    axis([0 L -0.5 1]);
    subplot(3,1,2), plot(t,Vg,'k')
    axis([0 L -0.5 1]);
    subplot(3,1,3), plot(ks,abs(fftshift(Vgt)),'k')
    drawnow
    pause(0.01)
end
```

8. Spectrogram and Score of music2.wav

```
%% Spectogram - Gaussian


subplot(1,1,1), pcolor(tslide,ks/(2*pi),Vgt_spec.'), shading interp
set(gca,{'Ylim', 'Fontsize'}, {[500 3500],  [14]})

title("Mary Had A Little Lamb (Recorder) With Overtones")
xlabel("time (s)");
ylabel("frequency (Hz)");

%% Score the music
pcolor(tslide,ks/(2*pi),Vgt_spec.'), shading interp
colormap(hot);
hold on;
pb = plot([0 16],[783.99 783.99],'c') %G4 freq
hold on;
pc= plot([0 16],[830.61 830.61],'b') %G#4 freq
hold on;
pcs = plot([0 16],[880.00 880.00],'g'); %A4 freq
hold on;
pd=plot([0 16],[932.33 932.33],'y') %A#4 freq
hold on
pds=plot([0 16],[987.77 987.77],'r'); %B4 freq
hold on;
pe=plot([0 16],[1046.50 1046.50],'w') %C5 freq

set(gca,'Ylim', [700 1100],'Fontsize',[14])
title("Score of Mary Had A Little Lamb (Recorder)")
xlabel("time (s)");
ylabel("frequency (Hz)");

lgd = legend([pe,pds,pd,pcs,pc,pb],'C6','B5','A#5','A5','G#5','G5');
lgd.FontSize = 10;
lgd.Title.String='Note Frequencies';
```