Finding Meaning in Music: An Analytical Approach

Kuan H. Chen
Chemical Engineering
Applied & Computational Mathematical Sciences

Abstract

One thousand years into the future, human efforts in advancing science came at the expense artistic expression – music, a concept recorded only in ancient text, was a product of past civilizations. Sound archives from the ancient times are discovered, and what might be described as *melodic* is listened to for the first time in a millennium. Researchers seek to recreate what they are hearing, but discover that the archives are riddled with noise that interrupts the melodies. By using the Gabor Transformation (GT), the Fast Fourier Transformation (FFT) algorithm, and filters, researchers are able to formally document this mysterious creation of the past, ushering in the start of Renaissance in the 32nd century.

Keywords: GT, order, characteristic frequency, FFT, resolution

Finding Meaning in Music: An Analytical Approach

I. Introduction

A musical note is nothing more special than the various sounds that we hear day to day. In a physical sense, it is simply a sound that occurs at a well-defined frequency. The significance of music can only be appreciated when a series of musical notes is arranged into a particular *order*. This order is what makes sound transcend into music. In this report, three passages of music titled *Handel*, *Mary Had a Little Lamb (Piano)* (MHALLP) and *Mary Had a Little Lamb (Recorder)* (MHALLR) are analyzed in order to document the musical notes that appear as well as the order at which they appear. To complicate efforts, these passages are riddled with noise at various frequencies, making it harder to identify which frequencies are responsible for creating the musical notes in question. The following sections explore the GT that is used to discern patterns, the Fast Fourier Transformation (FFT) and inverse FFT (iFFT) used to process signals, and an interpretation of the musical score that was recreated from the passages.

II. Theoretical Background

1. **Gabor Transformation:**

   When handling a series of data that varies with time, it is often useful to be able to discern an underlying pattern buried under the signals. The GT allows one to analyze a series of data in smaller "windows" to take a closer look at how data in each window differs from one another with respect to time. Various types of transformations have been devised; the Gaussian, Mexican Hat, and Shannon filters are shown as follows as Equation 1, 2, and 3, respectively,

$$\mathbf{1.}\ \psi_{Gaussian} = e^{-w_g(t-t_0)^2}$$

$$\mathbf{2.}\ \psi_{MH} = (1-(t-t_0))^2 e^{-\frac{w_{mh}(t-t_0)^2}{2}}$$

$$\mathbf{3.}\ \psi_{Shannon} = \begin{cases} 0\ \forall & t-t_0 < 0 \\ 1\ \forall & 0 < t-t_0 < w_s \\ 0\ \forall & w_s < t-t_0 \end{cases}$$

   where w_g, w_{mh}, and w_s are parameters that govern the window width of the filters. Based on the window width, different amounts of data can be captured, resulting in differences in time resolution.

2. **Characteristic Frequency:**

   Each note that appears in the passage has an associated characteristic frequency to it. Among the frequencies that are captured by the GT, the one corresponding to a musical note should be dominant and the rest can be disregarded as noise. Filtering out noise can be achieved simply by keeping a list of the frequencies that correspond to the most dominant signals in every window. The musical score can be recreated by matching these frequencies with time.

3. **Fourier Transformation:**

The Fourier Transformation is used to decompose a function in the time domain into the frequencies that make it up, the formula for which is shown in Equation 4.

$$\mathbf{4.}\, F(k) = \int_{-\infty}^{\infty} f(t)e^{-ikt}dt$$

After the transformation, the function becomes one that is dependent on wave number instead of time. In the wave number domain, the magnitude of signals represents the amount of detected signals at a given wave number. In this context, it is useful in finding the characteristic frequency of a note that is captured by the GT since wave numbers and frequencies differ by a factor of 2π. A dominant signal in the wave number domain will correspond to the characteristic frequency of a note.

4. **Resolution:**

The quality of the musical score can be judged by two parameters – how clearly defined the pattern with respect to time is, and how much noise is retained. The clarity of the pattern is characterized as the *time resolution* – the clearer the pattern the higher the time resolution; the amount of noise retained is characterized as the *frequency resolution* – the less noise retained the higher the frequency resolution. Both the time and frequency resolution of the score are related directly to the width of the window defined by the GT.

High resolution in both time and frequency is impossible, as there is an inherent trade-off property that exists between the two. The Heisenberg Uncertainty Principle, which states that it is impossible to measure both the position and momentum of a particle at the same time, can be evoked to illustrate this property and is shown in Equation 4.

$$\mathbf{4.}\ \sigma_x \sigma_p \geq \frac{h}{2\pi}$$

If a GT with a wide window length is used, more data points will be captured and the frequencies of dominant signals are guaranteed to be retained after filtering. This results in a high frequency resolution. However, with many frequencies captured in one window, it is impossible to decipher the order in which these frequencies occur. This results in a low time resolution. If a GT with a narrow window length is used, less data points will be captured, all of which can be noise. After filtering, the frequency retained may be noise itself, causing a lot of noise to appear on the score. This results in low frequency resolution. However, since data points are captured at more well-defined time points, it is easier to see a pattern emerge.

III. Algorithm Implementation and Development

Part 1

This section uses signals from the music file *handel* to illustrate the usage of GT and how their window widths may affect data analysis.

Figure 1 depicts the sound signals that were generated from *handel* over a period of 8.9 seconds.

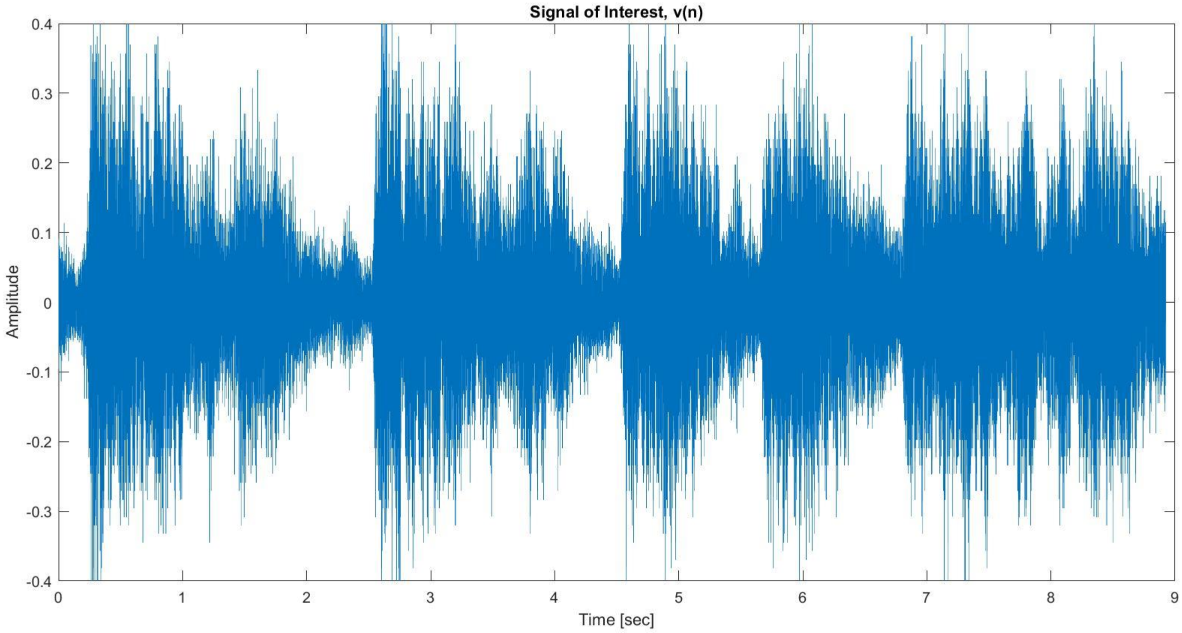

Figure 1. Sound signals generated from `handel`

Figure 2 depicts the effects of the Gaussian filter of various width factors on this data set.

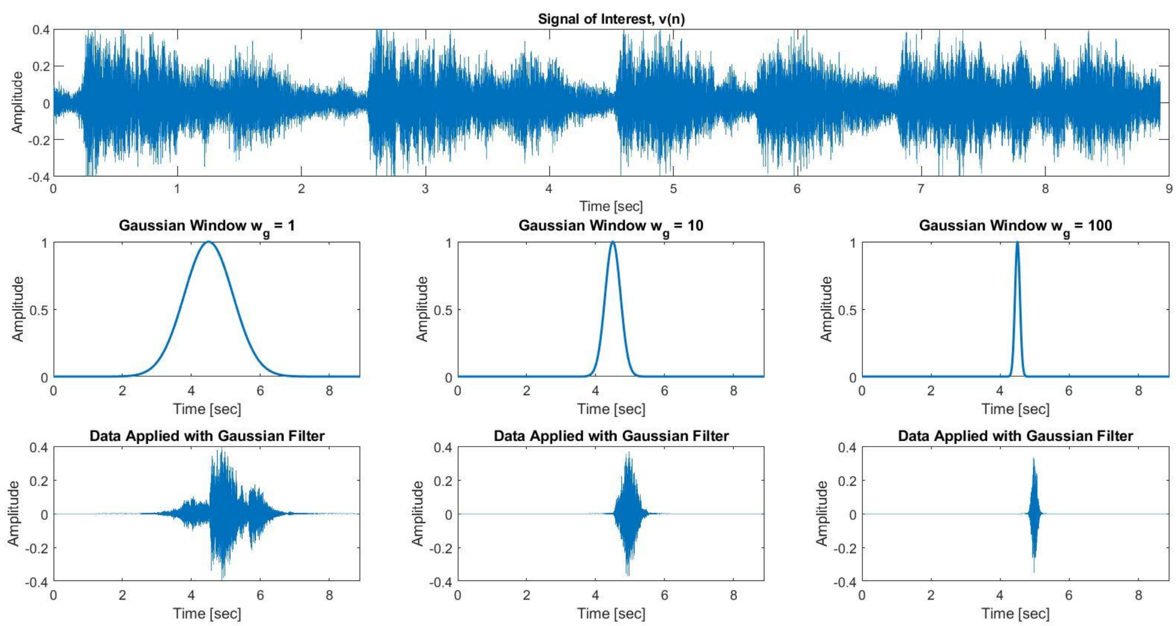

Figure 2. The Gaussian filter applied to the data set at w_g = 1, 10, and 100

Since the primary purpose of the GT is to identify patterns, as discussed in the previous section, $w_g = 100$ was chosen to maximize time resolution. To analyze subsets of the data with respect to time, the Gaussian filter was slid across the data set. Each subset of data was then taken the FFT of to find the set of frequencies that occurred in that window. These steps can be achieved using the following MATLAB commands:

```
gabor = @(w,t,n)exp(-w*(tspan-t).^n);
tslide = 0:0.1:(length(v))/Fs;
for j = 1:length(tslide)
    vg = gabor(width,tslide(j),power).*v;
    dataInWindowTransformedG = [dataInWindowTransformedG;abs(fftshift(vgt))];
end
```

In order to generate a spectrogram, `dataInWindowTransformedG` is first normalized by dividing all its components by its maximal value, as seen in the following command:

```
maxInG = max(dataInWindowTransformedG(i,:));
dataInWindowNormalizedG = [dataInWindowNormalizedG; dataInWindowTransformedG(i,:)/maxInG];
```

To generate the spectrograms, the following commands are used:

```
pcolor(tslide,ks,dataInWindowNormalizedG.'), shading interp
set(gca,'Fontsize',[14])
colormap(hot)
title('a) Gaussian')
xlabel('Time [sec]')
ylabel('Frequency')
```

Figure 3 shows two spectrograms that were generated using the Gaussian filter with $w_g = 100$ and 1, respectively:

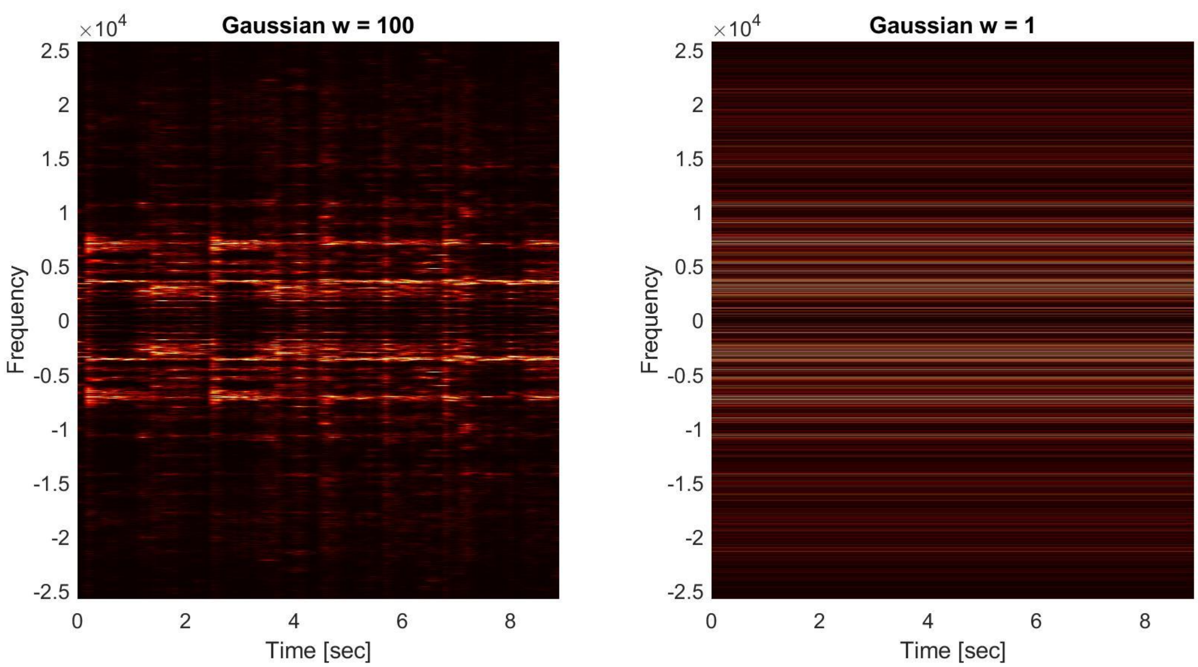

Figure 3. Two spectrograms are generated using Gaussian filters of $w_g = 100$ and 1

As can be seen, with a Gaussian filter of $w_g = 100$, varying patterns with respect to time emerge, corresponding to a high resolution in time; with a Gaussian filter of $w_g = 1$, no clear variation in patterns can be seen, which represent a low time resolution.

A similar procedure can be implemented to see how different types of GT may impact the resulting spectrogram. Figure 4 depicts the Gaussian, Mexican Hat, and Shannon filters of similar window lengths and their effects on the data set; Figure 5 depicts the spectrograms that these filters generate.

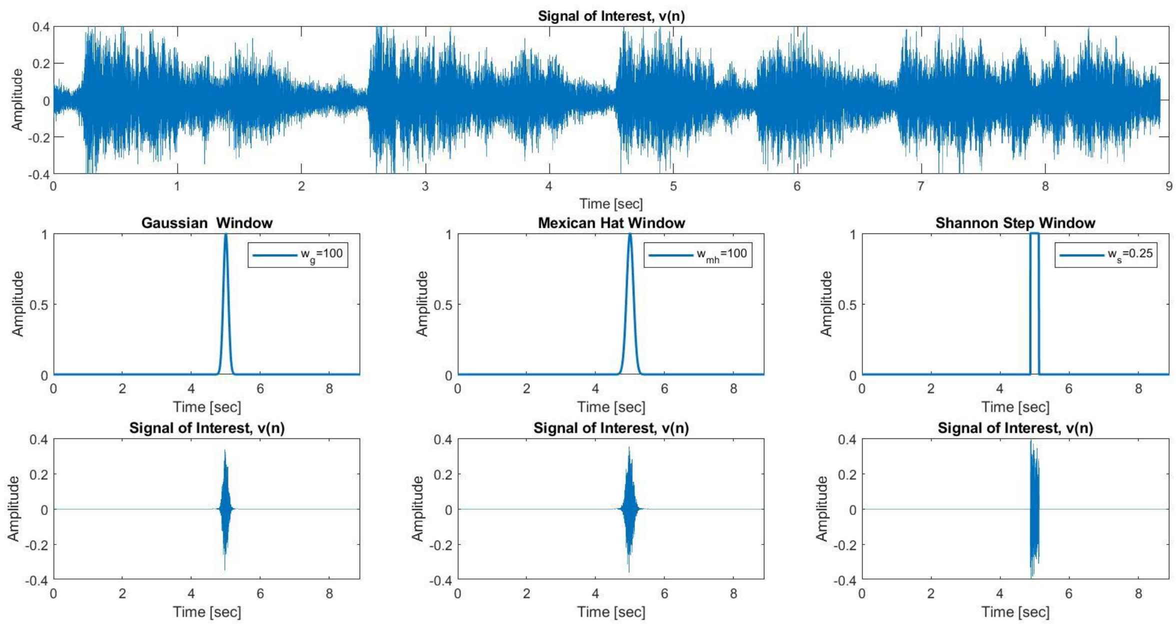

Figure 4. The Gaussian, Mexican Hat, and Shannon filters

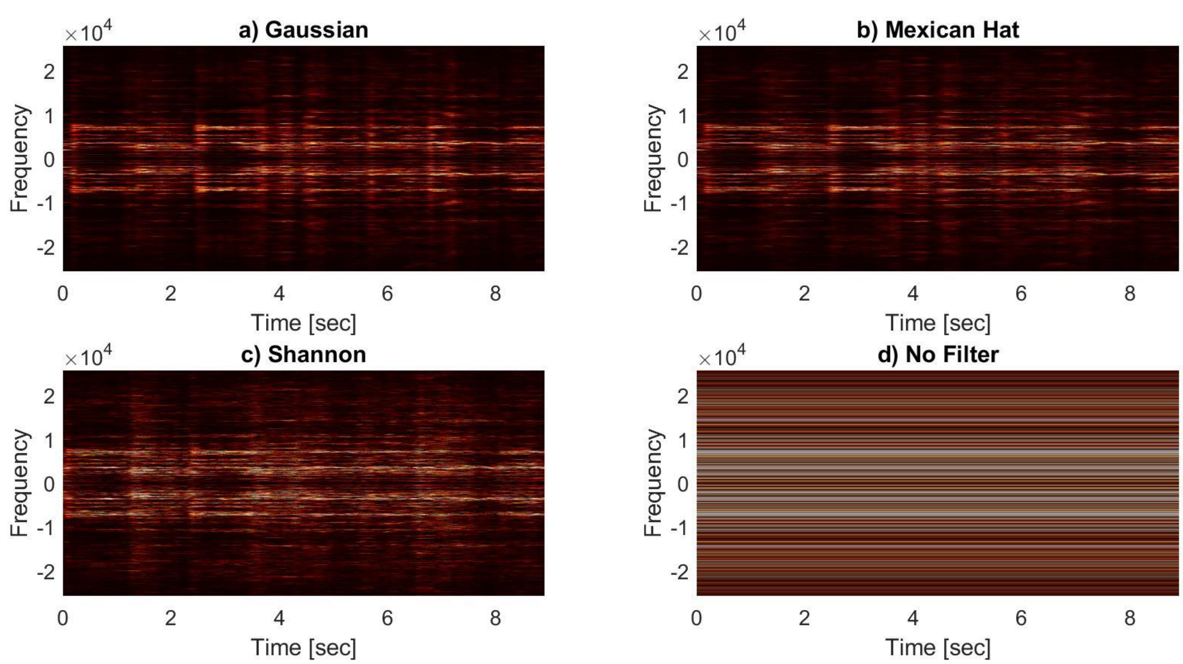

Figure 5. Spectrograms generated with the Gaussian, Mexican Hat, and Shannon filters, with one without any filter

As one can see, the different filters generate similar spectrograms, suggesting that the width of the filters is the primary determinant of time and frequency resolution.

Part 2

In this section, mysterious data files titled MHALLP and MHALLR are analyzed in an attempt to recreate the original score. Figure 6 shows the sound signals of MHALLP over a time period of 16 seconds.

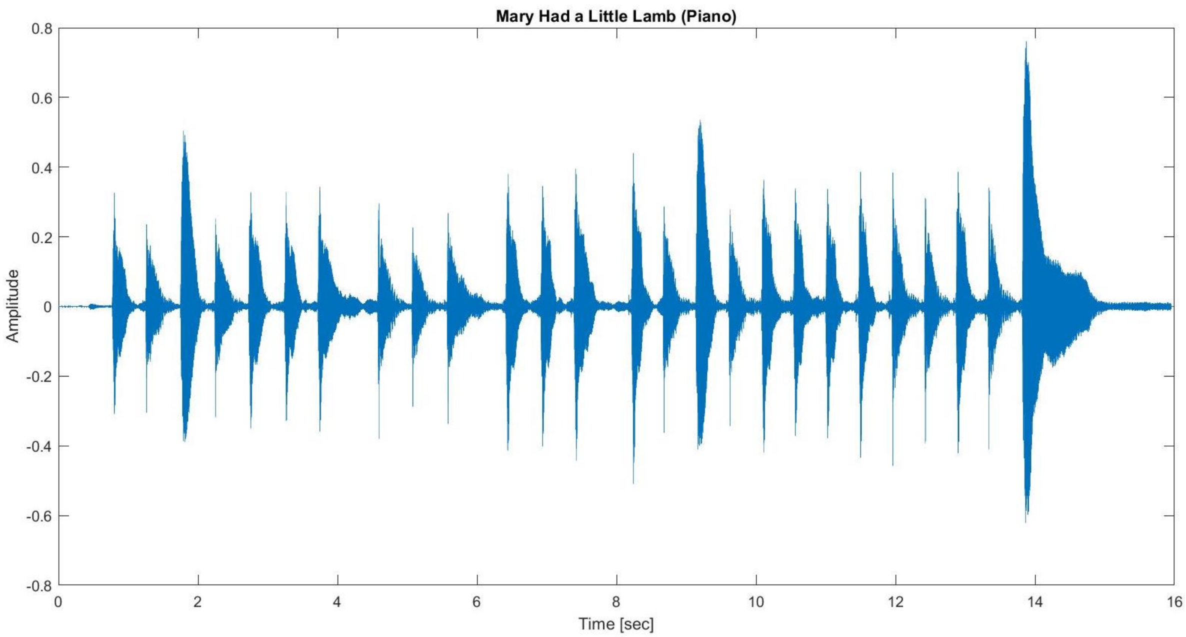

Figure 6. Sound signals of MMHALLP

Each large spike in amplitude can be seen as a potential musical note to capture. The goal is to identify at which frequencies these spikes occur and document them to recreate the score. This analysis is done using a Gaussian filter of $w_g = 10$ as a basis.

Figure 7 shows the effect of the Gaussian filter on the data set.

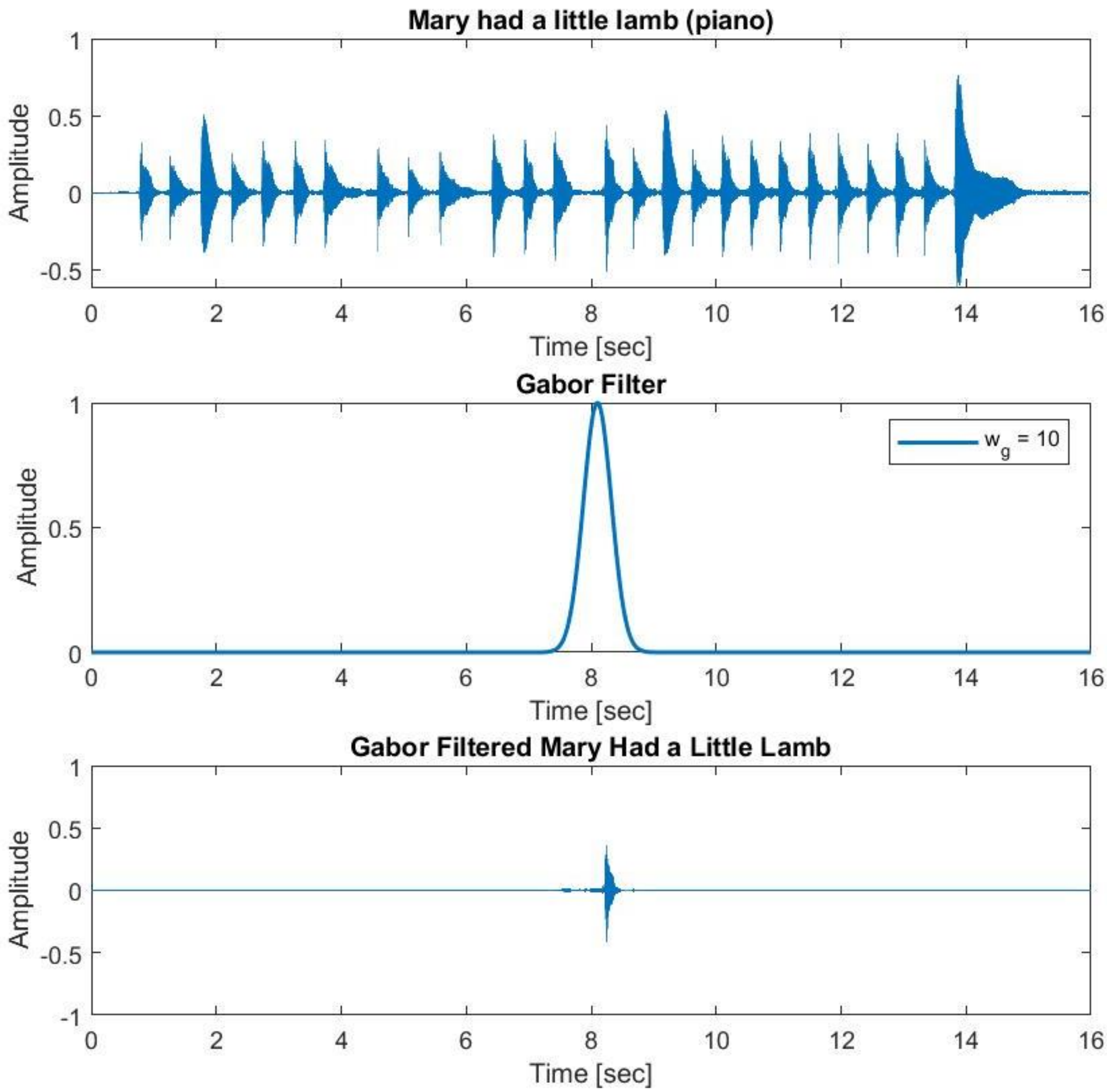

Figure 7. Effect of Gaussian filter of $w_g = 10$ on data set

To find the dominant signals that exist in each window of time, the FFT is applied to the data in each window with the following MATLAB commands:

```
dataInWindowTransformed = [];
for a = 1:length(tslidePiano)
    gaborWindow = gabor(10,tslidePiano(a),2);
    dataInWindow = gaborWindow'.*data;
    dataInWindowTransformed = [dataInWindowTransformed...
        abs(fftshift(fft(dataInWindow)))]
end
```

The most dominant signals in the window as well as their corresponding frequencies can then be found using the `max` function as follows:

```
dataInWindowNormalized = [];
frequencies = [];
for i = 1:length(tslidePiano)
    [maxInWindow,I] = max(dataInWindowTransformed(:,i));
    frequencies = [frequencies abs(ks(I)/2/pi)];
    dataInWindowNormalized = [dataInWindowNormalized...
        dataInWindowTransformed(:,i)/maxInWindow];
end
```

By plotting frequencies with respect to time, a music score, as shown in Figure 8 in the following section, is generated.

IV. Computational Results

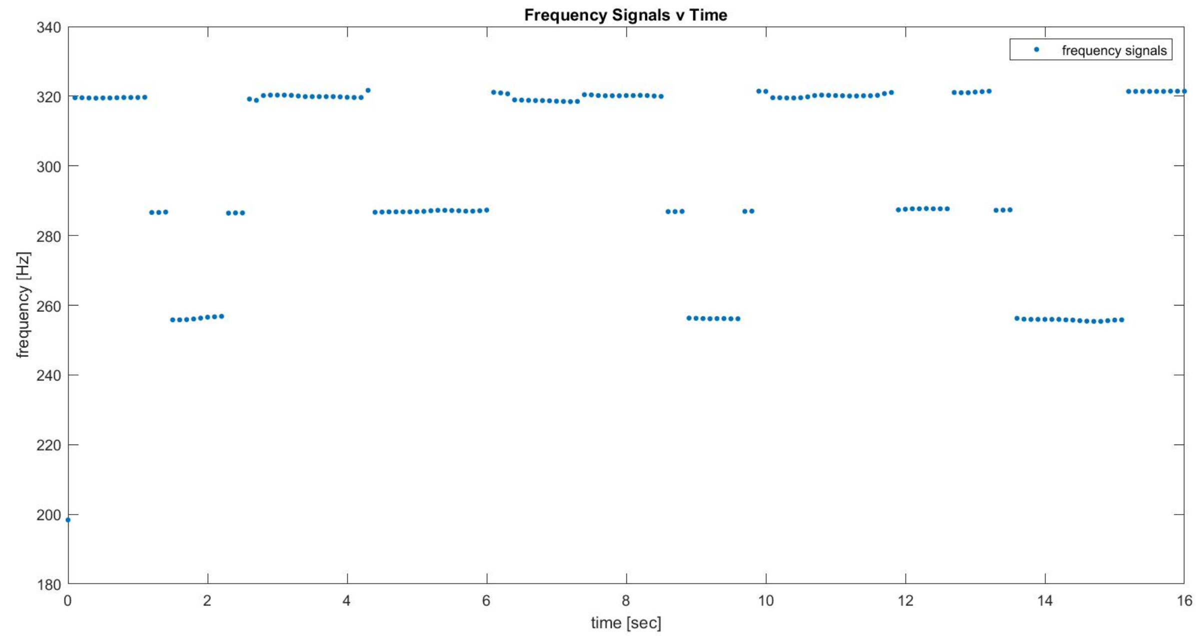

Figure 8. Music score generated for MHALLP from filter of w_g = 10

This is a music score that successfully captures the pattern of the music without retaining excessive amounts of noise. It can be seen from this score that there are three musical notes that frequently reoccur at around 319.5, 286.6, and 256.0 Hz. Points that are not within a short range from these values can be considered noise. Based on the chart provided in Appendix C, these notes can be identified as three of the following five notes – C_4, $C^{\#}_4$, D_4, $D^{\#}_4$, E_4.

If a width factor of w_g = 100 was used instead, the music score in Figure 9 would result.

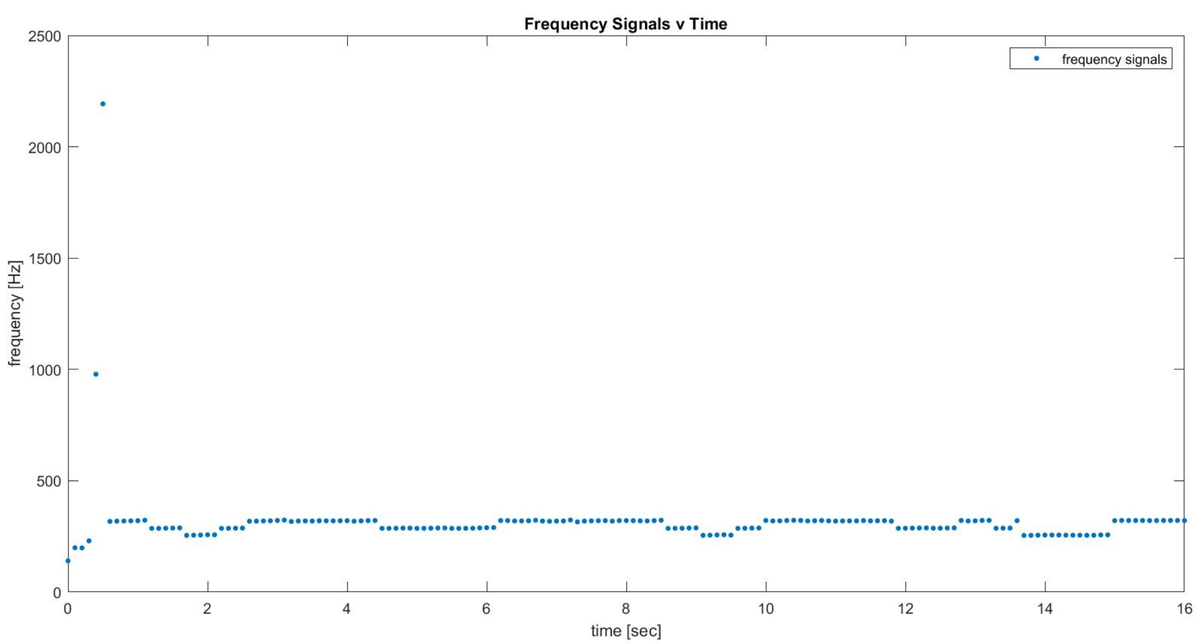

Figure 9. Music score generated for MHALLP from filter of w_g = 10

As one can see, the pattern of the score is much more defined, representing a high time resolution. However, more noise appears in the score, represented by the high amounts of deviations exhibited towards the beginning of the score.

On the other hand, if a filter of $w_g = 1$ was used, the score in Figure 10 will be generated.

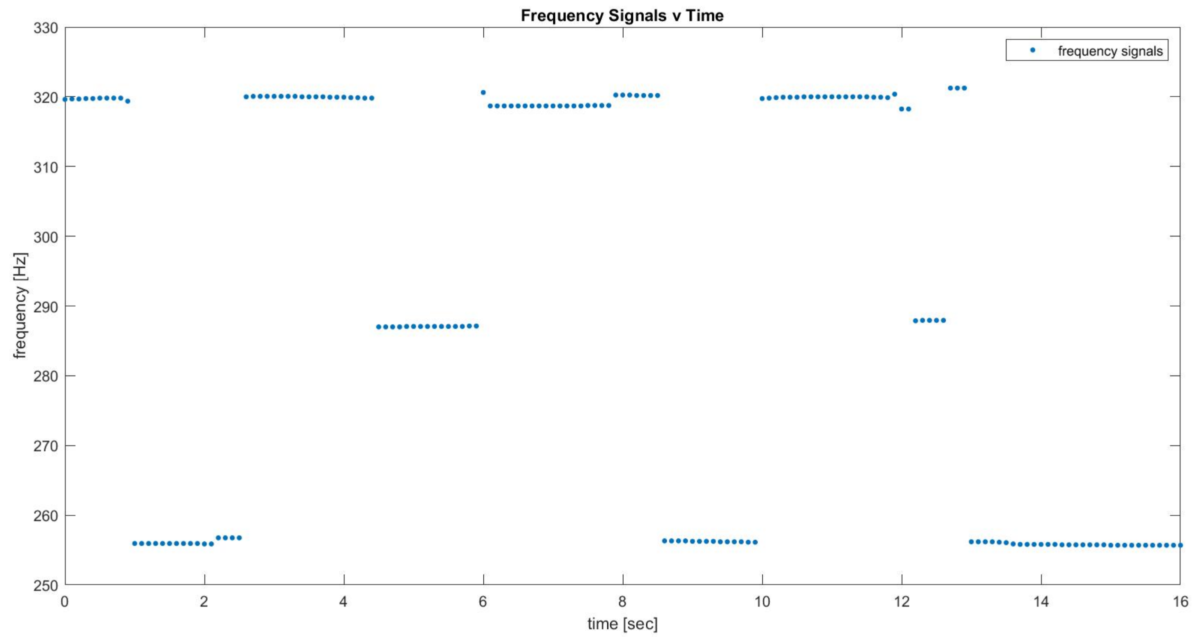

Figure 11. Music score generated for MHALLP from filter of $w_g = 1$

In this case, much less noise is retained compared to when the filter of $w_g = 10$ was used. However, some notes that were seen in the other two cases were not retained in this score, and the pattern is incomplete. Therefore, this score represents one with little time resolution but high frequency resolution.

The same procedure discussed to analyze MHALLP can be applied to MHALLR. The data set presented in Figure 12 can be transformed into a music score by using a Gaussian filter of $w_g = 10$ as shown in Figure 13.

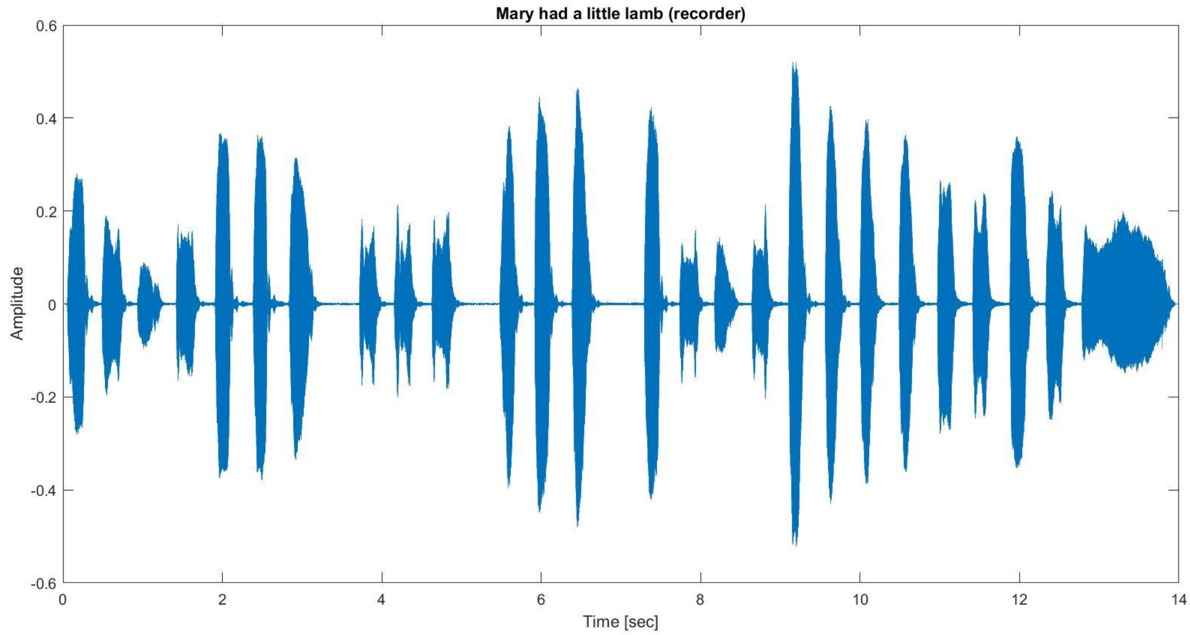

Figure 12. Sound signals of MHALLR

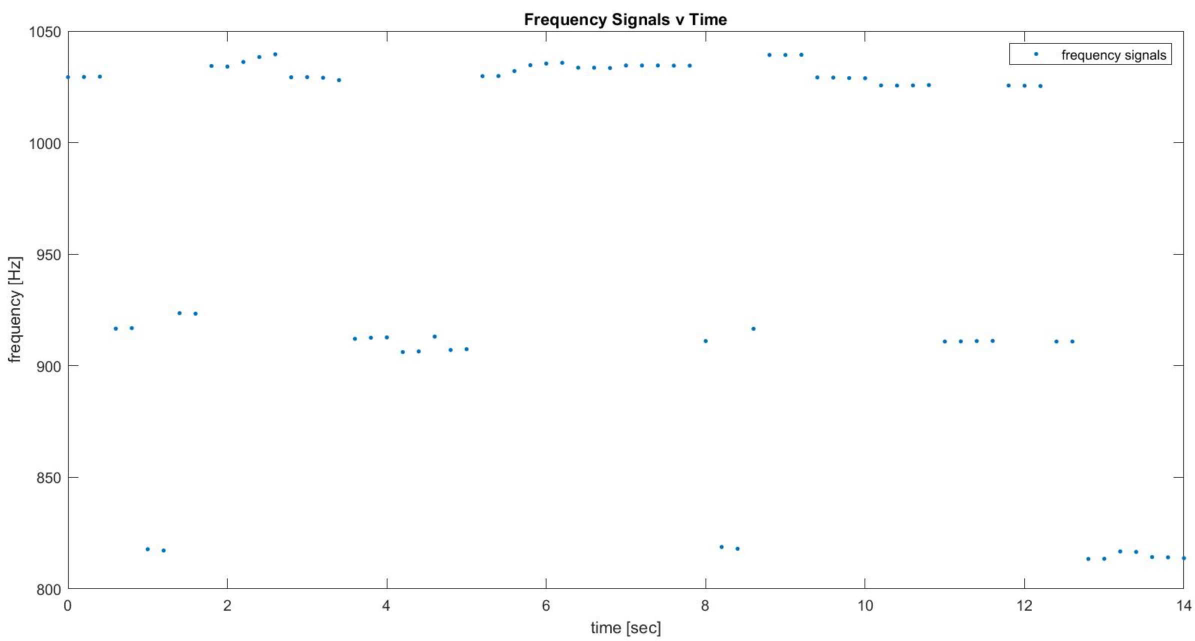

Figure 13. Music score generated for MHALLR with filter of w_g = 10

According to this score, it can be seen that three dominant musical notes emerge from the analysis with frequencies around 1029, 916.6 and 817.5 Hz. Based on the chart provided in Appendix C, these notes correspond to three of the following notes – B_5, $A^{\#}_5$, A_5, $G^{\#}_5$, G_5.

V. Summary and Conclusions

Using *Handel* as an example, the significance of the window width of a GT is explored. Based on conclusions drawn from the analysis of *Handel*, two passages of sound data were analyzed to recreate the musical score that they each represented. These analyses concluded that three notes from each data set frequently reoccurred – notes at frequency 319.5, 286.6, and 256.0 Hz for MHALLP, and notes at frequency 1029, 916.6 and 817.5 Hz for MHALLR. It is impossible to match these numbers perfectly to the literature values provided in Appendix C due to the imperfections of perhaps, the recording devices or analysis methods, but this the first step to reembracing humanities past tie to music. May this be the beginning of a glorious Renaissance that is powered by science!

Appendix A

The following are the MATLAB functions that were used to complete this report:

- `heaviside(tspan-t+w/2)`—The step function used to define the Shannon filter
- `pcolor(tslide,ks,allDataNormalized.'), shading interp`—Used to generate spectrograms

Appendix B

```
%% Part one -- Handel

%% Hallelujah
clear all;close all; clc
load handel
v = y'/2;
power = 2;
width = 100;
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');

L=length(v)/Fs;
n=length(v);
tspan = linspace(0,length(v)/Fs,length(v)+1);
tspan=tspan(1:n);
k=(2*pi/L)*[0:n/2 -n/2:-1]; ks=fftshift(k);

subplot(3,3,[1 2 3])
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');
%%
gabor = @(w,t,n)exp(-w*(tspan-t).^n);

subplot(3,3,4)
plot((1:length(v))/Fs, gabor(1, 4.5, 2), 'LineWidth', 1.5)
xlabel('Time [sec]');
ylabel('Amplitude');
title('Gaussian Window w_g = 1')
xlim([0 8.9])

subplot(3,3,5)
plot((1:length(v))/Fs, gabor(10, 4.5, 2), 'LineWidth', 1.5)
xlabel('Time [sec]');
ylabel('Amplitude');
title('Gaussian Window w_g = 10')
xlim([0 8.9])

subplot(3,3,6)
plot((1:length(v))/Fs, gabor(100, 4.5, 2), 'LineWidth', 1.5)
xlabel('Time [sec]');
ylabel('Amplitude');
title('Gaussian Window w_g = 100')
xlim([0 8.9])

subplot(3,3,7)
plot((1:length(v))/Fs, gabor(1, 5, 2).*v)
xlabel('Time [sec]');
ylabel('Amplitude');
title('Data Applied with Gaussian Filter')
```

```
xlim([0 8.9])

subplot(3,3,8)
plot((1:length(v))/Fs, gabor(10, 5, 2).*v)
xlabel('Time [sec]');
ylabel('Amplitude');
title('Data Applied with Gaussian Filter')
xlim([0 8.9])

subplot(3,3,9)
plot((1:length(v))/Fs, gabor(100, 5, 2).*v)
xlabel('Time [sec]');
ylabel('Amplitude');
title('Data Applied with Gaussian Filter')
xlim([0 8.9])
%% Hallelujah with different Gabor Filters



mexican = @(w,t) (1-(tspan-t).^2).*exp(-w*((tspan-t).^2)/2);
shannonStep = @(w, t) heaviside(tspan-t+w/2)-heaviside(tspan-t-w/2);
subplot(3,3,[1 2 3])
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');
for lol = 1:5
    gaborAndMexicanWidth = [0.1 1 10 100 100];
    shannonWidth = [2 1 0.5 0.25 0.1];
    subplot(3,3,4)
    plot(tspan,gabor(gaborAndMexicanWidth(lol),5,power), 'LineWidth', 1.5)
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Gaussian  Window');
    legend(strcat('w_g=',num2str(gaborAndMexicanWidth(lol))))
    xlim([0 8.9])

    subplot(3,3,5)
    plot(tspan,mexican(gaborAndMexicanWidth(lol), 5), 'LineWidth', 1.5)
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Mexican Hat Window');
    ylim([0,1])
    legend(strcat('w_m_h=',num2str(gaborAndMexicanWidth(lol))))
    xlim([0 8.9])

    subplot(3,3,6)
    plot(tspan, shannonStep(shannonWidth(lol),5), 'LineWidth', 1.5)
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Shannon Step Window');
    legend(strcat('w_s=',num2str(shannonWidth(lol))))
    xlim([0 8.9])

    vg = v.*gabor(gaborAndMexicanWidth(lol),5,power);
    vg2 = v.*gabor(1,5,power);
```

```
    subplot(3,3,7)
    plot(tspan,vg);
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Signal of Interest, v(n)');
    xlim([0 8.9])

    vm = v.*mexican(gaborAndMexicanWidth(lol),5);
    subplot(3,3,8)
    plot(tspan,vm);
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Signal of Interest, v(n)');
    xlim([0 8.9])

    vs = v.*shannonStep(shannonWidth(lol),5);
    subplot(3,3,9)
    plot(tspan,vs)
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Signal of Interest, v(n)');
    xlim([0 8.9])
    pause
end
%
close all;
subplot(4,1,1)
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');
dataInWindowTransformedG = [];
dataInWindowTransformedG2 = [];
dataInWindowTransformedM = [];
dataInWindowTransformedS = [];
allDataTransformed = [];
tslide = 0:0.1:(length(v))/Fs;
for j = 1:length(tslide)
    subplot(4,1,2)
    plot(tspan,gabor(width,tslide(j),power))
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Gabor Window');
    subplot(4,1,3)
    vg = gabor(width,tslide(j),power).*v;
    vm = mexican(width,tslide(j)).*v;
    vs = shannonStep(0.5, tslide(j)).*v;
    plot(tspan,vg)
    ylim([-0.5 0.5])
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Signal of Interest, v(n)');
    subplot(4,1,4)
    vgt = fft(vg);
    vgt2 = fft(vg2);
    vmt = fft(vm);
```

```
    vst = fft(vs);
    vt = fft(v);

    dataInWindowTransformedG = [dataInWindowTransformedG;abs(fftshift(vgt))];
    dataInWindowTransformedG2 = [dataInWindowTransformedG2;abs(fftshift(vgt2))];
    dataInWindowTransformedM = [dataInWindowTransformedM;abs(fftshift(vmt))];
    dataInWindowTransformedS = [dataInWindowTransformedS;abs(fftshift(vst))];
    allDataTransformed = [allDataTransformed; abs(fftshift(vt))];
    plot(ks,abs(fftshift(vgt)))
    xlabel('Wave Number')
    ylabel('Amplitude');
    title('Signal of Interest, v(n)');
    ylim([0 100])
    drawnow
    pause(0.05)
end
%
close all;
dataInWindowNormalizedG = [];
dataInWindowNormalizedG2 = [];
dataInWindowNormalizedM = [];
dataInWindowNormalizedS = [];
allDataNormalized = [];
for i = 1:length(tslide)
    maxInG = max(dataInWindowTransformedG(i,:));
    maxInG2 = max(dataInWindowTransformedG2(i,:));
    maxInM = max(dataInWindowTransformedM(i,:));
    maxInS = max(dataInWindowTransformedS(i,:));
    maxInAll = max(allDataTransformed(i,:));
    dataInWindowNormalizedG = [dataInWindowNormalizedG;
dataInWindowTransformedG(i,:)/maxInG];
    dataInWindowNormalizedG2 = [dataInWindowNormalizedG2;
dataInWindowTransformedG2(i,:)/maxInG2];
    dataInWindowNormalizedM = [dataInWindowNormalizedM;
dataInWindowTransformedM(i,:)/maxInM];
    dataInWindowNormalizedS = [dataInWindowNormalizedS;
dataInWindowTransformedS(i,:)/maxInS];
    allDataNormalized = [allDataNormalized; allDataTransformed(i,:)/maxInAll];
end
close all;
for j = 1:length(tslide)
    subplot(3,1,1)
    plot(tspan,gabor(width,tslide(j),2))
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Gabor Window');
    subplot(3,1,2)
    plot(tspan,mexican(100,tslide(j)))
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Mexican Hat Window');
    subplot(3,1,3)
    plot(tspan,shannonStep(0.5,tslide(j)))
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Shannon Step Window');
```

```
    pause(0.1)
end


for j = 1:length(tslide)
    subplot(4,1,2)
    vg = gabor(width,tslide(j),2).*v;
    plot(tspan,vg)
    ylim([-0.5 0.5])
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Signal of Interest (Gabor), v(n)');

    subplot(4,1,3)
    vm = mexican(100,tslide(j)).*v;
    plot(tspan,vm)
    ylim([-0.5 0.5])
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Signal of Interest (Mexican Hat), v(n)');

    subplot(4,1,4)
    vs = shannonStep(0.5,tslide(j)).*v;
    plot(tspan,vs)
    ylim([-0.5 0.5])
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Signal of Interest (Shannon Step), v(n)');
    pause(0.1)
end

%%
close all;
subplot(2,2,1)
pcolor(tslide,ks,dataInWindowNormalizedG.'), shading interp
set(gca,'Fontsize',[14])
colormap(hot)
title('a) Gaussian')
xlabel('Time [sec]')
ylabel('Frequency')


subplot(2,2,2)
pcolor(tslide,ks,dataInWindowNormalizedM.'), shading interp
set(gca,'Fontsize',[14])
colormap(hot)
title('b) Mexican Hat')
xlabel('Time [sec]')
ylabel('Frequency')

subplot(2,2,3)
pcolor(tslide,ks,dataInWindowNormalizedS.'), shading interp
set(gca,'Fontsize',[14])
colormap(hot)
title('c) Shannon')
xlabel('Time [sec]')
```

```
ylabel('Frequency')

subplot(2,2,4)
pcolor(tslide,ks,allDataNormalized.'), shading interp
set(gca,'Fontsize',[14])
colormap(hot)
title('d) No Filter')
xlabel('Time [sec]')
ylabel('Frequency')

%% Part Two -- Lambs
clear all; close all; clc
data=audioread('music1.wav');
numSubplot = 3;
tr_piano=16; % record time in seconds
tspanPiano = linspace(0,tr_piano,length(data));
tslidePiano = 0:0.1:tr_piano;
Fs=length(data)/tr_piano;
plot((1:length(data))/Fs,data);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Mary Had a Little Lamb (Piano)');
subplot(numSubplot,1,1)
plot((1:length(data))/Fs,data);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Mary had a little lamb (piano)'); drawnow

L = length(data)/Fs;
n = length(data);
tspan = linspace(0,n/Fs,n+1);
tspan = tspan(1:n);
krange= [0:n/2-1, -n/2:-1];
krange= (2*pi/L)*krange;
ks=fftshift(krange);
dataTransformed = fft(data);
dataTransformedShift = fftshift(dataTransformed); % use for plotting


gabor = @(w,t,n)exp(-w*(tspanPiano-t).^n);


dataInWindowTransformed = [];
for a = 1:length(tslidePiano)
    gaborWindow = gabor(1,tslidePiano(a),2);
    subplot(numSubplot, 1, 2)
    plot(tspanPiano, gaborWindow, 'LineWidth', 1.5)
    title('Gabor Filter')
    xlabel('Time [sec]')
    ylabel('Amplitude')
    dataInWindow = gaborWindow'.*data;
    legend('w_g = 10')
    subplot(numSubplot, 1, 3)
    plot(tspanPiano, dataInWindow)
    title('Gabor Filtered Mary Had a Little Lamb')
    ylim([-1 1])
```

```
    xlabel('Time [sec]')
    ylabel('Amplitude')
    dataInWindowTransformed = [dataInWindowTransformed
abs(fftshift(fft(dataInWindow)))];
    pause(0.01)
end


dataInWindowNormalized = [];
frequencies = [];
for i = 1:length(tslidePiano)
   [maxInWindow,I] = max(dataInWindowTransformed(:,i));
   frequencies = [frequencies abs(ks(I)/2/pi)];
   dataInWindowNormalized = [dataInWindowNormalized...
       dataInWindowTransformed(:,i)/maxInWindow];
end
close all;

plot(tslidePiano, frequencies, '.', 'MarkerSize',10)
legend('frequency signals')
title('Frequency Signals v Time')
xlabel('time [sec]')
ylabel('frequency [Hz]')

%% Part 2 -- Recorder
clear all; close all; clc
numSubplot = 4;
tr_rec=14; % record time in seconds
dataRec=audioread('music2.wav');
Fs=length(dataRec)/tr_rec;
tspanRec = linspace(0,tr_rec,length(dataRec));
tslideRec = 0:0.2:tr_rec;
%subplot(numSubplot,1,1)
plot((1:length(dataRec))/Fs,dataRec);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Mary had a little lamb (recorder)');
%%
% p8 = audioplayer(y,Fs); playblocking(p8);
L = length(dataRec)/Fs;
n = length(dataRec);
tspan = linspace(0,n/Fs,n+1);
tspan = tspan(1:n);
krange= [0:n/2-1, -n/2:-1];
krange= (2*pi/L)*krange;
ks=fftshift(krange);
dataTransformed = fft(dataRec);
dataTransformedShift = fftshift(dataTransformed); % use for plotting
subplot(numSubplot,1,2)
plot(ks,abs(dataTransformedShift)/max(abs(dataTransformed)))
xlabel('frequency domain')
ylabel('amplitude')
title('data transformed')
gabor = @(w,t,n)exp(-w*(tspanRec-t).^n);

dataInWindowTransformed = [];
```

```
for a = 1:length(tslideRec)
    gaborWindow = gabor(10,tslideRec(a),2);
    subplot(numSubplot, 1, 3)
    plot(tspanRec, gaborWindow)
    title('Gabor Filter')
    xlabel('t Recorder')
    ylabel('Amplitude')
    dataInWindow = gaborWindow'.*dataRec;
    subplot(numSubplot, 1, 4)
    plot(tspanRec, dataInWindow)
    title('Gabor Filtered Mary Had a Little Lamb')
    ylim([-1 1])
    xlabel('t Recorder')
    ylabel('Amplitude')
    dataInWindowTransformed = [dataInWindowTransformed
abs(fftshift(fft(dataInWindow)))];
    pause(0.05)
end


dataInWindowNormalized = [];
frequencies = [];
for i = 1:length(tslideRec)
   [maxInWindow,I] = max(dataInWindowTransformed(:,i));
   frequencies = [frequencies abs(ks(I)/2/pi)];
   dataInWindowNormalized = [dataInWindowNormalized
dataInWindowTransformed(:,i)/maxInWindow];

end
%%
close all;
plot(tslideRec, frequencies, '.', 'MarkerSize',8)
legend('frequency signals')
title('Frequency Signals v Time')
xlabel('time [sec]')
ylabel('frequency [Hz]')
```

Appendix C

Note	Frequency (Hz)
C_0	16.35
$C^{\#}_0/D^b_0$	17.32
D_0	18.35
$D^{\#}_0/E^b_0$	19.45
E_0	20.6
F_0	21.83
$F^{\#}_0/G^b_0$	23.12
G_0	24.5
$G^{\#}_0/A^b_0$	25.96
A_0	27.5
$A^{\#}_0/B^b_0$	29.14
B_0	30.87
C_1	32.7
$C^{\#}_1/D^b_1$	34.65
D_1	36.71
$D^{\#}_1/E^b_1$	38.89
E_1	41.2
F_1	43.65
$F^{\#}_1/G^b_1$	46.25
G_1	49
$G^{\#}_1/A^b_1$	51.91
A_1	55
$A^{\#}_1/B^b_1$	58.27
B_1	61.74
C_2	65.41
$C^{\#}_2/D^b_2$	69.3
D_2	73.42
$D^{\#}_2/E^b_2$	77.78
E_2	82.41
F_2	87.31
$F^{\#}_2/G^b_2$	92.5
G_2	98
$G^{\#}_2/A^b_2$	103.83
A_2	110
$A^{\#}_2/B^b_2$	116.54
B_2	123.47
C_3	130.81
$C^{\#}_3/D^b_3$	138.59

D_3	146.83
$D^{\#}_3/E^{b}_3$	155.56
E_3	164.81
F_3	174.61
$F^{\#}_3/G^{b}_3$	185
G_3	196
$G^{\#}_3/A^{b}_3$	207.65
A_3	220
$A^{\#}_3/B^{b}_3$	233.08
B_3	246.94
C_4	261.63
$C^{\#}_4/D^{b}_4$	277.18
D_4	293.66
$D^{\#}_4/E^{b}_4$	311.13
E_4	329.63
F_4	349.23
$F^{\#}_4/G^{b}_4$	369.99
G_4	392
$G^{\#}_4/A^{b}_4$	415.3
A_4	440
$A^{\#}_4/B^{b}_4$	466.16
B_4	493.88
C_5	523.25
$C^{\#}_5/D^{b}_5$	554.37
D_5	587.33
$D^{\#}_5/E^{b}_5$	622.25
E_5	659.25
F_5	698.46
$F^{\#}_5/G^{b}_5$	739.99
G_5	783.99
$G^{\#}_5/A^{b}_5$	830.61
A_5	880
$A^{\#}_5/B^{b}_5$	932.33
B_5	987.77
C_6	1046.5
$C^{\#}_6/D^{b}_6$	1108.73
D_6	1174.66
$D^{\#}_6/E^{b}_6$	1244.51
E_6	1318.51
F_6	1396.91
$F^{\#}_6/G^{b}_6$	1479.98

G_6	1567.98
$G^{\#}_6/A^b_6$	1661.22
A_6	1760
$A^{\#}_6/B^b_6$	1864.66
B_6	1975.53
C_7	2093
$C^{\#}_7/D^b_7$	2217.46
D_7	2349.32
$D^{\#}_7/E^b_7$	2489.02
E_7	2637.02
F_7	2793.83
$F^{\#}_7/G^b_7$	2959.96
G_7	3135.96
$G^{\#}_7/A^b_7$	3322.44
A_7	3520
$A^{\#}_7/B^b_7$	3729.31
B_7	3951.07
C_8	4186.01
$C^{\#}_8/D^b_8$	4434.92
D_8	4698.63
$D^{\#}_8/E^b_8$	4978.03
E_8	5274.04
F_8	5587.65
$F^{\#}_8/G^b_8$	5919.91
G_8	6271.93
$G^{\#}_8/A^b_8$	6644.88
A_8	7040
$A^{\#}_8/B^b_8$	7458.62
B_8	7902.13