HW 2: Gabor Transforms

Amanda Lee

Abstract

In this assignment, I examine the effects of using different window sizes and the effects of undersampling versus oversampling for the Gabor transform through spectrogram analysis. I also examine the differences between spectrograms made with the Gabor transform using different kinds of filters, such as the Gaussian filter, Mexican hat wavelet, and Shannon filter. In the second part of this assignment, I use time-frequency analysis to determine the notes being played in two recordings of "Mary Had a Little Lamb", one of a piano, and one of a recorder. I then examine the differences between the overtones produced by each instrument.

Introduction and Overview

For the first section, I will use time-frequency analysis on a clip of Handel's "Messiah". I will create spectrograms from the sound clip using different variations on a Gabor Transforms, examining the effect of different window sizes and oversampling versus undersampling, as well as the effect of using different filters.
For the second part, there are two clips of "Mary Had a Little Lamb" played with different instruments. The goal is to use spectrogram analysis to figure out which notes are being played at what times and make a score of the piece. I will also use the spectrograms to analyze the differences in the sounds produced by the piano and the recorder.

Theoretical Background

These are the methods I will use to analyze the clips of music.

Gabor Transform

A Gabor Transform improves on Fourier Transform's the lack of time information by looking at the signal through a certain window and sliding the window along the signal, taking the Fourier Transform of it at each time. This is a form of time-frequency analysis. The windows are created by multiplying the signal by a gaussian centered at the time of interest. This is called filtering and has the effect of "zooming in" on a certain area of the signal. Functions other than gaussians can also be used for filtering. Depending on the width of the window, the analysis will reveal more or less time or frequency information. A wider window will result in more frequency resolution but less time resolution. Conversely, a narrower window will result in less frequency resolution with greater time resolution.
The Discrete Gabor Transform looks at a finite number of windows with a certain amount of overlap between one window and the next. Undersampling happens when the windows are spaced too far apart and some signal information gets lost. Oversampling is when the windows overlap each other. Overlap is necessary so that signal information is not lost. Gabor Transforms can be used to create spectrograms, a way of visually representing frequency information over time. The power of the frequency can represented by a color scale.I will use a number of different types of filters for the first part of the analysis.
I will first use a gaussian filter. The equation is as follows:

$$\mathcal{F}(k) = e^{-a(\tau-t)^2}$$

Where τ is a point in time and t is the time that the filter is centered at.
This is the equation for a Mexican Hat Wavelet:

$$\mathcal{F}(k) = (1 - a(\tau - t)^2)e^{-a(\tau-t)^2/2}$$

Where τ is a point in time and t is the time that the filter is centered at. I will also use a Shannon filter, which is a step function with a value of one inside the time window and a value of zero everywhere else.

Algorithm Implementation and Development

Part I.

To begin a time-frequency analysis on the clip of Handel's Messiah, I load the signal into MATLAB. Then I take the transpose of the vector containing the signal so that it is easier to work with. I construct the time domain of the signal by making a vector that goes from 1 to the length of the signal vector, then dividing each value by the sampling frequency. I then get the length of the signal in seconds from the last value in the time domain vector. Then, I construct the frequency domain of the signal.

After that, I will first experiment with differently sized windows for a Gabor Transform and look at the spectrograms created by each window size. The window sizes will be determined by the value of a in the equation used for the filter. I will first use a gaussian filter for the Gabor Transform. To do this, I make a vector with the different a values I want to test. Then, for each a value, I make a vector of time values, called tslide, to center the windows at along the signal. Then, for each time values, I calculate the value of the filter and apply it to the signal by multiplying it with the signal. I then take the Fourier Transform of the filtered signal to get its frequency information and add it to an array that holds information about the rest of the windows. I can then create a spectrogram using these values.

Next, using one window size, I will test different amounts of how far the window is translated at each step. To do this, I follow the same process as above, but I change the length of my tslide vector to make each spectrogram. I will then repeat the above experiments with different window sizes and different amounts of shifting using a Mexican Hat Wavelet and a Shannon Wavelet instead of a Gaussian. To make the Shannon filter, I just rounded the values from a Gaussian filter to either 1 or 0.

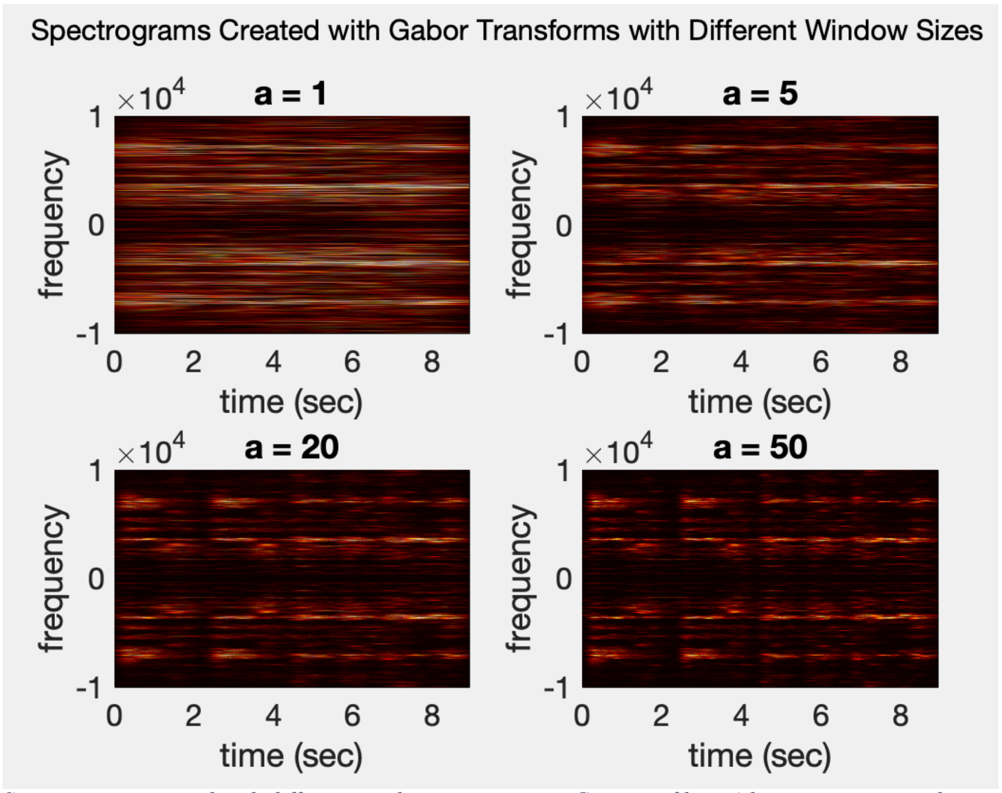

Figure 1: Spectrograms created with different windows sizes using a Gaussian filter. A larger a corresponds to a smaller window.

Part II.

Similar to Part I, I will load in the music recordings, take the transpose of the vector containing the signal, construct the time domains, get the length of each signal in seconds, and construct the frequency domain of each signal. I will then take the Gabor transform of both signals and use them to create spectrograms of each recording to compare the overtones produced by each instrument. While going through each time window, I will save the strongest frequency in a vector. A can then use the spectrograms to see the rhythm of the music being played. I will have to divide the frequencies by 2pi in order to get the frequencies in hertz. Then I can plot the strongest frequencies at each time window and use the values in the plot to get the frequencies of the notes.

Computational Results

Part I.

In general, for different window sizes, a small a value resulted in very stretched out spectrograms with poor time resolution. In the spectrograms below, the a values of 1 and 5 don't show the rhythm of the music clip very clearly. Going from an a value of 5 to an a value of 20, there is a
clear difference in clarity of the time information. The gaps between notes are more visible going from an value of 20 to 50, however, the differences in the spectrogram are more subtle. The information along the time axis is more defined with some loss in frequency information.
For different numbers of window translations, a lot number resulted in spectrograms that looked stretched out, similar to those made with a wide window. However, the undersampled spectrograms had less continuity, with the lines being broken at some points along the time axis. This is likely a result of information loss between the windows. Higher numbers of window translations produced spectrograms that look practically identical. This is likely because different levels of oversampling are not going to have more or less information of the sample contained between each other if they are all oversampled.
Different filters produced spectrograms with slightly different qualities. The gaussian filter produced spectrograms with smoother edges along the time axis than the Shannon filter. Spectrograms made with the Mexican hat wavelet had time information that was much more difficult to make out. Increasing the window size parameter for the Shannon filter from 20 to 50 greatly helped with this, more so than for the spectrograms from the other filters.

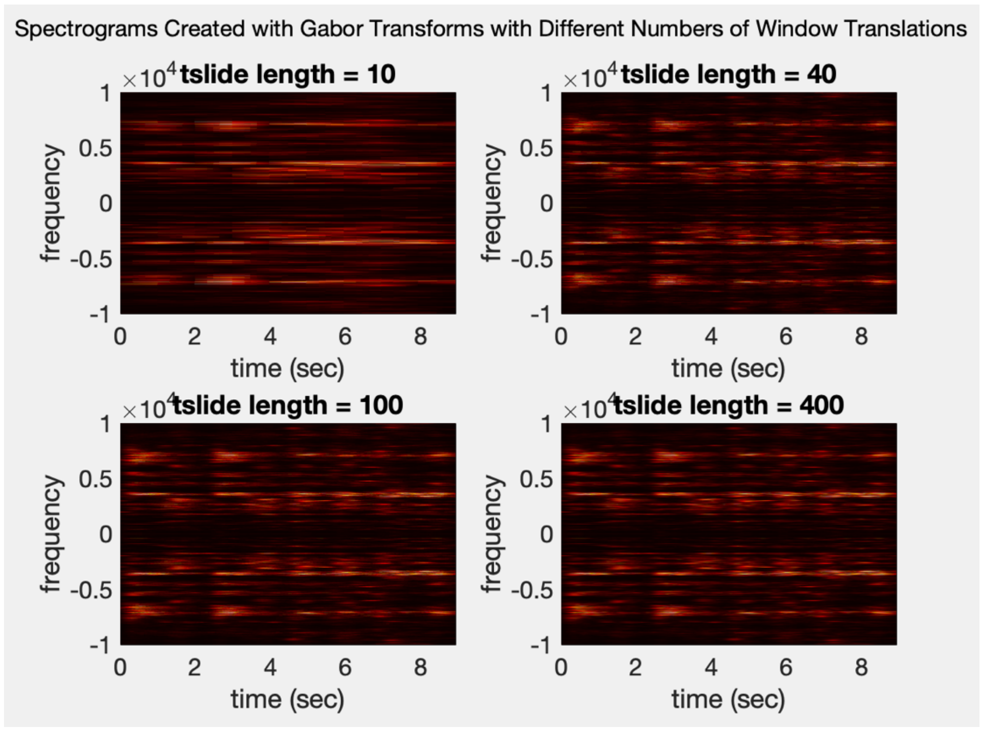

Figure 2: Spectrograms created with different amounts of sampling using a Gaussian filter. A longer tslide vector corresponds with a higher sampling rate for the Gabor transform.

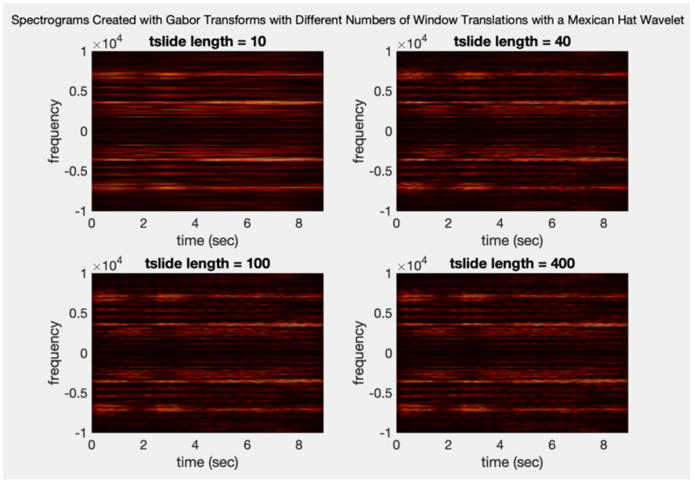

Figure 4: Spectrograms created with different windows sizes using a Mexican hat wavelet..

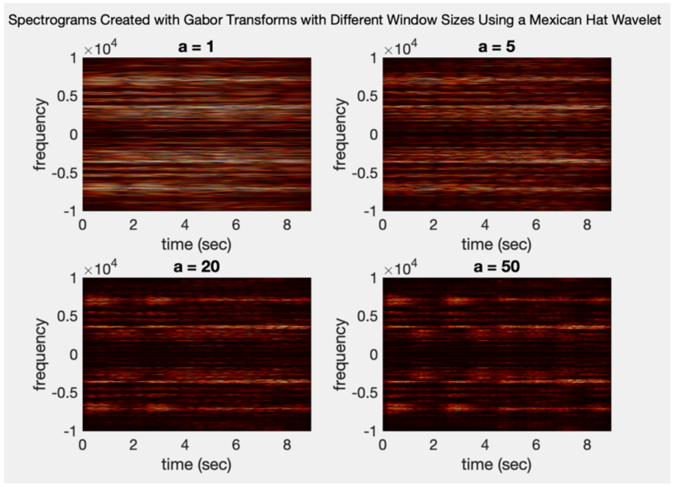

Figure 3: Spectrograms created with different amounts of translation using a Mexican hat wavelet.

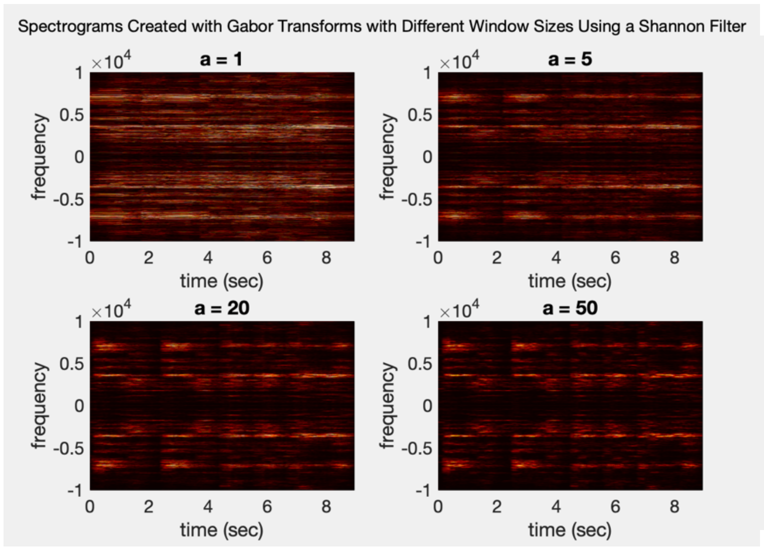

Figure 6: Spectrograms created with different windows sizes using a Shannon filter.

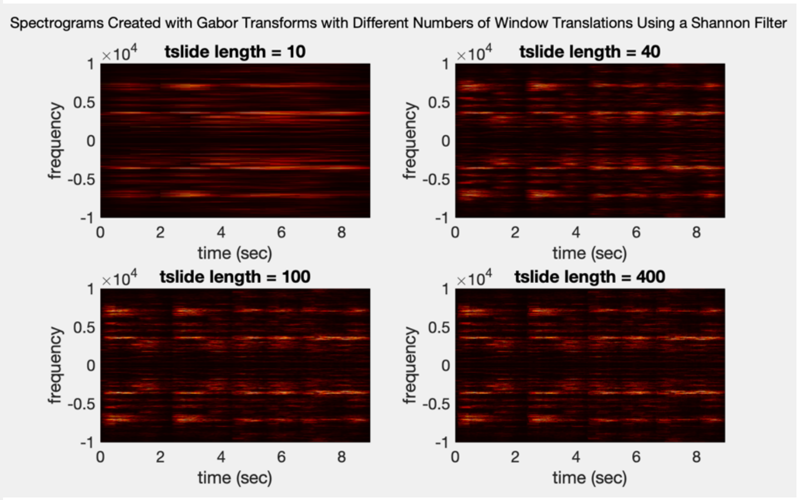

Figure 5: Spectrograms created with different amounts of translation using a Shannon filter.

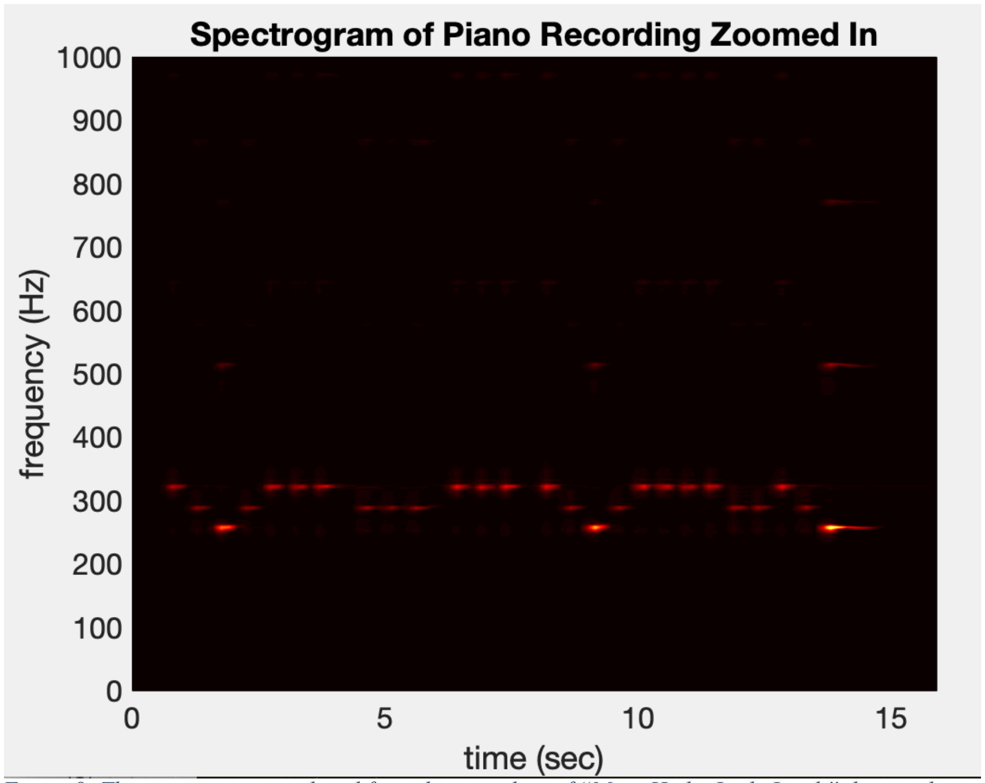

Figure 8: The spectrogram produced from the recording of "Mary Had a Little Lamb" done with a piano. The spectrogram was produced by a Gabor transform using a Gaussian filter and an a value of 50.

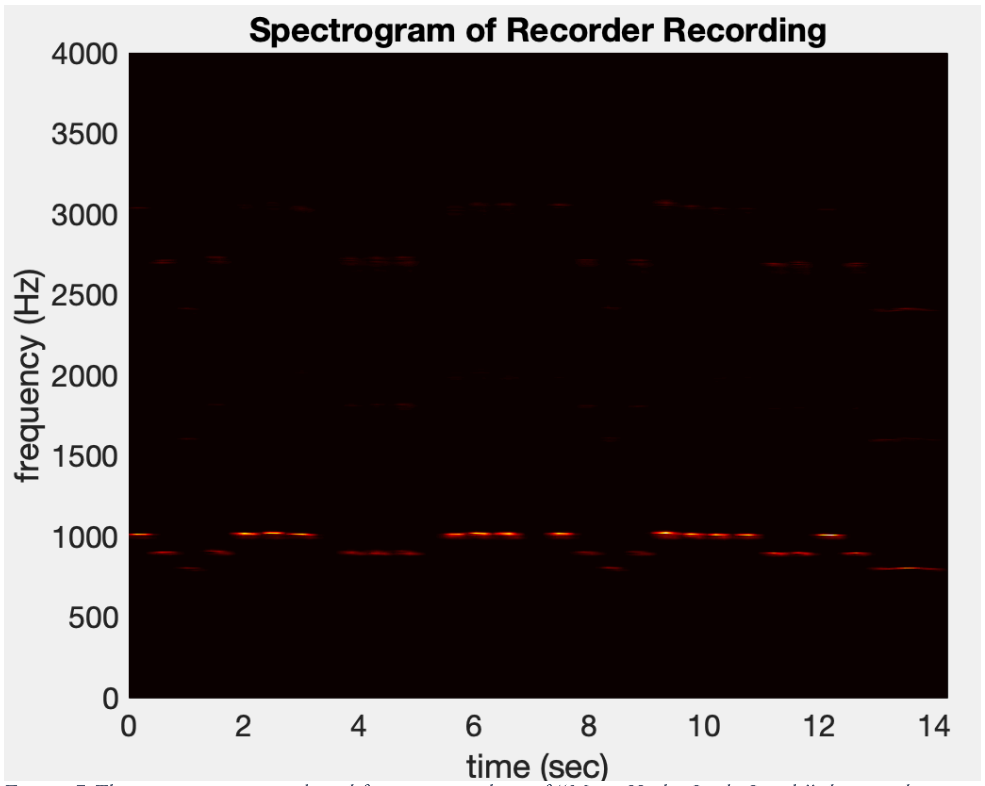

Figure 7:The spectrogram produced from a recording of "Mary Had a Little Lamb" done with a recorder. The spectrogram was produced by a Gabor transform using a Gaussian filter and an a value of 50. There is a clear difference between the spectrograms produced by each instrument.

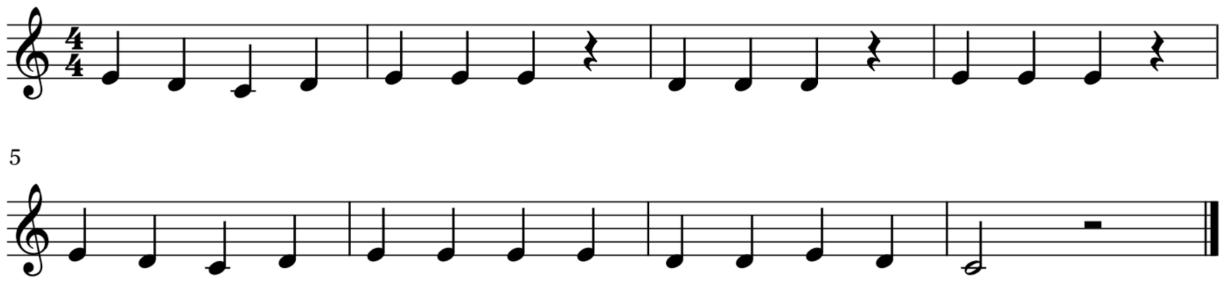

Figure 9: The score for the piano.

Part II.

As seen in figure 7 and 8 the overtones of the recorder are much higher than the overtones from the piano, which makes sense because the recorder is playing notes at a higher frequency. It is also harder to see the overtones on the spectrogram for the piano, meaning that the overtones are quieter.

Using the plots made from the strongest frequencies at each time window, I determined the notes being played. I used the plot in combination with the spectrogram to determine the rhythm of the notes and write the scores for each instrument.

Summary and Conclusion

Using a window that is too large greatly reduces the time resolution given in a Gabor transform. Too large of a window, however, can reduce the amount of frequency resolution. It is important to use an adequately sized window for a Gabor transform for time-frequency analysis.

Appendix A. MATLAB Functions Used

`Y=fft(X)` returns the one dimensional Fast Fourier transform of a vector. This is used to convert the signal to the frequency domain.

`Y=fftshift(X)` shifts the zero-frequency component of `X` to the center of the array. This is used so that the values are arranged properly for plotting.

`pcolor(X,Y,C)` creates a pseudocolor plot using the values in matrix `C`. The vector `X` specifies the x coordinates and the vector `Y` specifies the y coordinates. This is used to create spectrograms.

`[y,Fs]=audioread(filename)` reads data from filename and returns sampled data, `y` and the sampling rate for the data, `Fs`. This is used to read in audio files to analyze.

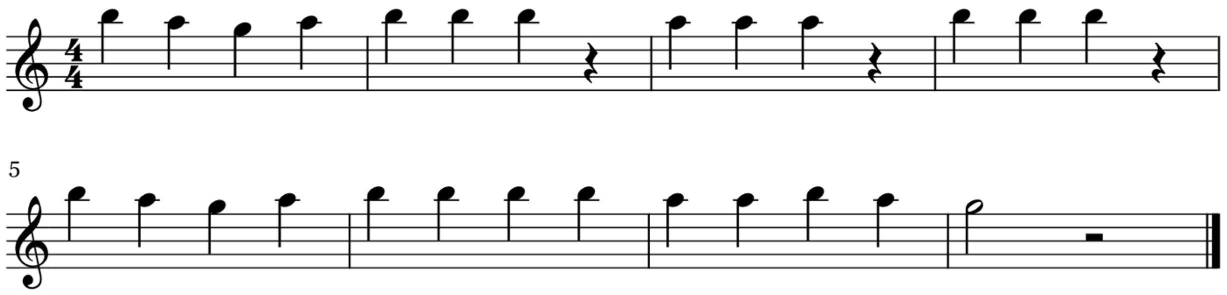

Figure 10: The score for the recorder. The rhythm is the same as the piano's, but the notes for the recorder are over an octave higher.

Appendix B. Matlab Code

Part I.

```
% Part I
clear; close all; clc;
load handel
% Signal
v = y';
% Time domain of signal
t = (1:length(v))/Fs;
% Length of signal in seconds
L = t(length(t));
% Number of samples
n = length(v);
% Construct Frequency Domain
k=(2*pi/L)*[0:(n-1)/2 -(n-1)/2:-1];
ks=fftshift(k);

%plot Time domain
figure(1);
plot(t,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');
%%
%p8 = audioplayer(v,Fs);
%playblocking(p8);
%% Create spectrograms with different window sizes
sgtitle('Spectrograms Created with Gabor Transforms with Different Window
Sizes')
a = [1 5 20 50];
for jj = 1:length(a)
    tslide = linspace(0,L,100);
    vgt_spec = zeros(length(tslide),n);
    for j = 1:length(tslide)
        g = exp(-a(jj) * (t-tslide(j)).^2);
        vg = g.*v;
        vgt = fft(vg);
        vgt_spec(j,:) = fftshift(abs(vgt));
    end

    subplot(2,2,jj)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['a = ',num2str(a(jj))],'Fontsize',10)
    xlabel('time (sec)')
    ylabel('frequency')
    set(gca,'Ylim',[-1*10^4 1*10^4],'Fontsize',16)
    colormap(hot)
end
%% Create spectrograms with different numbers of window translations
```

```
sgtitle('Spectrograms Created with Gabor Transforms with Different Numbers of
Window Translations')
a = 20;
tslide_vec = [10 40 100 400];
for jj = 1:length(tslide_vec)
    tslide = linspace(0,L,tslide_vec(jj));
    vgt_spec = zeros(length(tslide),n);
    for j = 1:length(tslide)
        g = exp(-a * (t-tslide(j)).^2);
        vg = g.*v;
        vgt = fft(vg);
        vgt_spec(j,:) = fftshift(abs(vgt));
    end

    subplot(2,2,jj)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['tslide length = ',num2str(length(tslide))],'Fontsize',10)
    xlabel('time (sec)')
    ylabel('frequency')
    set(gca,'Ylim',[-1*10^4 1*10^4],'Fontsize',16)
    colormap(hot)
end
%% Create spectrograms with different window sizes using a mexican hat
wavelet
sgtitle('Spectrograms Created with Gabor Transforms with Different Window
Sizes Using a Mexican Hat Wavelet')
a = [1 5 20 50];
for jj = 1:length(a)
    tslide = linspace(0,L,100);
    vgt_spec = zeros(length(tslide),n);
    for j = 1:length(tslide)
        g = (1-a(jj)*(t-tslide(j)).^2).* exp(-a(jj)*(t-tslide(j)).^2/2);
        vg = g.*v;
        vgt = fft(vg);
        vgt_spec(j,:) = fftshift(abs(vgt));
    end

    subplot(2,2,jj)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['a = ',num2str(a(jj))],'Fontsize',10)
    xlabel('time (sec)')
    ylabel('frequency')
    set(gca,'Ylim',[-1*10^4 1*10^4],'Fontsize',16)
    colormap(hot)
end
%% Create spectrograms with different numbers of window translations
sgtitle('Spectrograms Created with Gabor Transforms with Different Numbers of
Window Translations with a Mexican Hat Wavelet')
a = 20;
tslide_vec = [10 40 100 400];
for jj = 1:length(tslide_vec)
```

```
    tslide = 0:5:n;
    vgt_spec = zeros(length(tslide),n);
    for j = 1:length(tslide)
        g = (1-a*(t-tslide(j)).^2).* exp(-a*(t-tslide(j)).^2/2);
        vg = g.*v;
        vgt = fft(vg);
        vgt_spec(j,:) = fftshift(abs(vgt));
    end

    subplot(2,2,jj)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['tslide length = ',num2str(length(tslide))],'Fontsize',10)
    xlabel('time (sec)')
    ylabel('frequency')
    set(gca,'Ylim',[-1*10^4 1*10^4],'Fontsize',16)
    colormap(hot)
end
%% Create spectrograms with different window sizes using a Shannon filter
sgtitle('Spectrograms Created with Gabor Transforms with Different Window
Sizes Using a Shannon Filter')
a = [1 5 20 50];
for jj = 1:length(a)
    tslide = linspace(0,L,100);
    vgt_spec = zeros(length(tslide),n);
    for j = 1:length(tslide)
        g = round(exp(-a(jj) * (t-tslide(j)).^2));
        vg = g.*v;
        vgt = fft(vg);
        vgt_spec(j,:) = fftshift(abs(vgt));
    end

    subplot(2,2,jj)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['a = ',num2str(a(jj))],'Fontsize',10)
    xlabel('time (sec)')
    ylabel('frequency')
    set(gca,'Ylim',[-1*10^4 1*10^4],'Fontsize',16)
    colormap(hot)
end
%% Create spectrograms with different numbers of window translations
sgtitle('Spectrograms Created with Gabor Transforms with Different Numbers of
Window Translations Using a Shannon Filter')
a = 20;
tslide_vec = [10 40 100 400];
for jj = 1:length(tslide_vec)
    tslide = linspace(0,L,tslide_vec(jj));
    vgt_spec = zeros(length(tslide),n);
    for j = 1:length(tslide)
        g = round(exp(-a * (t-tslide(j)).^2));
        vg = g.*v;
        vgt = fft(vg);
        vgt_spec(j,:) = fftshift(abs(vgt));
    end

    subplot(2,2,jj)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['tslide length = ',num2str(length(tslide))],'Fontsize',10)
    xlabel('time (sec)')
    ylabel('frequency')
    set(gca,'Ylim',[-1*10^4 1*10^4],'Fontsize',16)
    colormap(hot)
end
```

Part II.

```
% Part II
clear; close all; clc;
[y1,Fs1] = audioread('music1.wav');
[y2,Fs2] = audioread('music2.wav');
% Signal
v1 = y1';
v2 = y2';
% Time domain of signal
t1 = (1:length(v1))/Fs1;
t2 = (1:length(v2))/Fs2;
% Length of signal in seconds
L1 = t1(length(t1));
L2 = t2(length(t2));
% Number of samples
n1 = length(v1);
n2 = length(v2);
% Construct Frequency Domain
k1=(2*pi/L1)*[0:n1/2-1 -n1/2:-1];
ks1=fftshift(k1);
k2=(2*pi/L2)*[0:n2/2-1 -n2/2:-1];
ks2=fftshift(k2);

a1 = 50;
tslide1 = linspace(0,L1,100);
tslide2 = linspace(0,L2,100);

% Gabor Transform for piano
vgt_spec1 = zeros(length(tslide1),n1);
max_freq1 = zeros(length(tslide1),1);
for j = 1:length(tslide1)
    g1 = exp(-a1 * (t1-tslide1(j)).^2);
    vg1 = g1.*v1;
    vgt1 = fft(vg1);
    vgt_spec1(j,:) = fftshift(abs(vgt1));
    [M,I] = max(abs(vgt1));
    max_freq1(j) = abs(k1(I))/(2*pi);
end
```

```
% Gabor Transform for recorder
a2 = 50;
vgt_spec2 = zeros(length(tslide2),n2);
max_freq2 = zeros(length(tslide2),1);
for j = 1:length(tslide2)
    g2 = exp(-a2 * (t2-tslide2(j)).^2);
    vg2 = g2.*v2;
    vgt2 = fft(vg2);
    vgt_spec2(j,:) = fftshift(abs(vgt2));
    [M,I] = max(abs(vgt2));
    max_freq2(j) = abs(k2(I))/(2*pi);
end
%% Plot spectrograms
% Piano
figure(1)
pcolor(tslide1,ks1/(2*pi),vgt_spec1.'),
shading interp
title('Spectrogram of Piano Recording')
xlabel('time (sec)')
ylabel('frequency (Hz)')
set(gca,'Ylim',[0 4000],'Fontsize',16)
colormap(hot)
%%
figure(2)
pcolor(tslide1,ks1/(2*pi),vgt_spec1.'),
shading interp
title('Spectrogram of Piano Recording Zoomed In')
xlabel('time (sec)')
ylabel('frequency (Hz)')
set(gca,'Ylim',[0 1000],'Fontsize',16)
colormap(hot)
%%
% Recorder
figure(3)
pcolor(tslide2,ks2/(2*pi),vgt_spec2.'),
shading interp
title('Spectrogram of Recorder Recording')
xlabel('time (sec)')
ylabel('frequency (Hz)')
set(gca,'Ylim',[0 4000],'Fontsize',16)
colormap(hot)

%% Plots to help identify note frequencies
% Piano
figure(4)
plot(tslide1,max_freq1)
title('Piano Frequencies Over Time')
xlabel('time (sec)')
ylabel('frequency (Hz)')

% Recorder
figure(5)
```

```
plot(tslide2,max_freq2)
title('Recorder Frequencies Over Time')
xlabel('time (sec)')
ylabel('frequency (Hz)')
```