Joshua Whalen
AMATH 482
February 7, 2020

Assignment 2: Gábor Transforms

Abstract
The purpose of this assignment was to investigate the different techniques involved in filtering frequencies of music recordings i order to generate the original frequencies or notes of the music. This assignment required the use of the Gábor transform and multiple window types to filter the frequencies in the time domain, and find the frequencies with the highest amplitudes. The results of this assignment are divided into two parts. The first part shows analysis of a famous music segment by Handel presented in multiple spectrograms. I investigated the effects of window width, type, and translation on the overall spectrogram quality. The second part shows the process of generating a music score by filtering the music sample, and correlating the resulting frequencies with the corresponding musical notes.

Introduction
The Fourier transform is known as the basis behind most all signal processing, but it has large limitations. In particular, the Fourier transform can transform the signal recorded in the time domain to the frequency domain, but it doesn't retain any information from the time domain. This means all of the low and high frequencies of a musical piece will be captured, but there is no way to tell when they occur in the time domain. Therefore, the Fourier transform is great for constant or periodic frequencies like with sonar and radar, but not for musical pieces that have multiple frequencies all changing in time. As signal processing methods advanced, the limitations of the Fourier transform were very apparent and it wasn't long before the Gábor transform was created, which was the first method to capture both time and frequency information at the same time. The method involves modifying the original Fourier transform by sliding through the time domain and filtering for high amplitude frequencies. This will tracks the highest amplitude frequencies and the time they occur. However, this method is also highly dependent on the width of the Gábor window and the translation of the window. The application of this signal processing method will be investigated further in this paper.

Theoretical Background
The equations for the Fourier transform and the Gábor transform are shown below. Since the Gábor was based on the Fourier, it is clear that they are very similar, but the Gábor includes a "g(tao -t)" term which denotes the sliding filter function. This sliding function can be in many different shapes and sizes.

Fourier: $\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{-ikx} f(x)dx$ Gábor: $\int_{-\infty}^{\infty} f(\tau)\bar{g}(\tau - t)e^{-i\omega\tau}d\tau$

In this paper, we will explore the standard Gaussian filter window as well as a Shannon filter window. The tao refers to the time domain that the function is integrating over while the t represents the postion of the filter in the time domain. This transform will be taken across the time domain at all time points separated by the sampling translation "t". It is important to consider the width of the filter window and the spacing of the sampling in the time domain in order to not over or under sample. This is illustrated in the figure below. As shown in the figure, depending on how the sampling is spaced in the time domain the filtering can overlap greatly, which will increase the computational load without gaining much new information. Conversely, if the windows do not overlap at all there will be areas of the original signal that won't be sampled at all, which will result in poor spectrogram resolution.

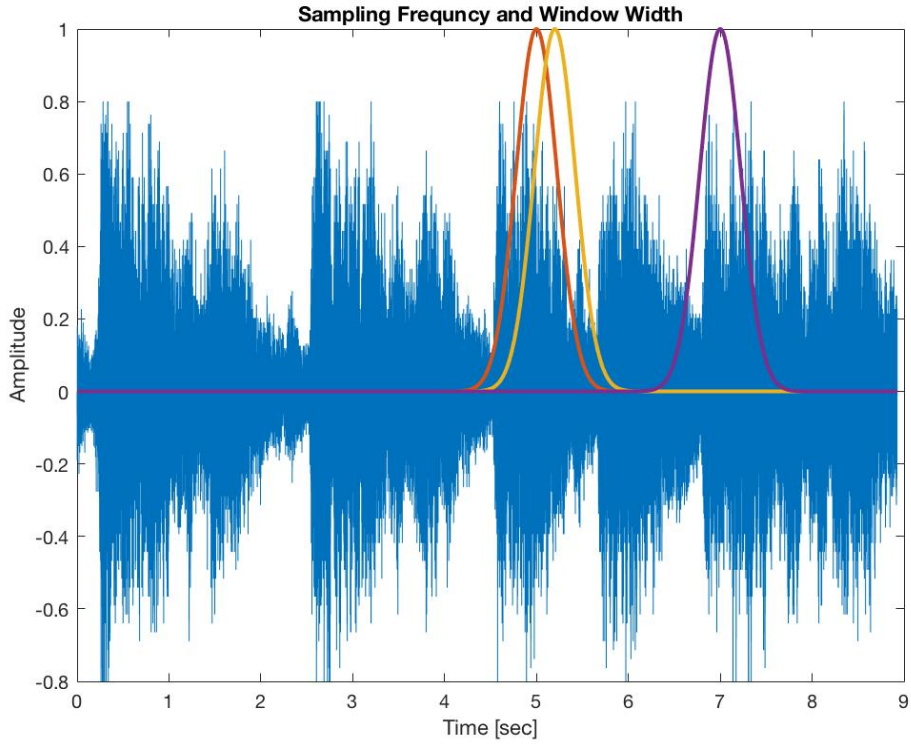

Fig 1: This plot shows a case of over sampling between the red and yellow curves and a under sampling between the red and blue curves due to nonidealized translation step size of the Gábor filter window.

Although the Gábor method was revolutionary for incorporating both time and frequency in the same plot, it too has limitations. With this method it is still impossible to get very fine resolution in time and frequency at the same time. As the window width is decreased, the time resolution will increase, but the ability to catch lower frequencies is lost, and frequency resolution will decrease. Expectedly, as the window width increased a higher range of frequencies can be captured, but the position in time of those frequencies is lost. Balancing the resolution of time and frequency is large drawback to the Gábor transform.

Algorithm Implementation

This section will go over the code used to develop each plot throughout this paper, and the code used to produce the plots is posted in appendix B. First for part 1, after loading in the Handel music segment, I had to define the parameters for the space for Gábor transform. This involved defining the length of the signal in time, which was found by dividing the number of data points by the sampling frequency. Then I defined the filtering window type and sampling frequency of the sliding window. For the first plots, a simple Gaussian filter was used. The original signal was multiplied by the filter in the time domain iterating through all steps in the tslide variable, which defined the "tao" translation steps in time. The window width and translation step size was then optimized to produce the most fine signal in both the time and frequency domain. Next, the Gaussian filter was compared to the Shannon filter. The translation step size was held constant for both windows, and the width of each was varied as shown in Fig 2. The results of the comparison is shown in the results section of the paper.

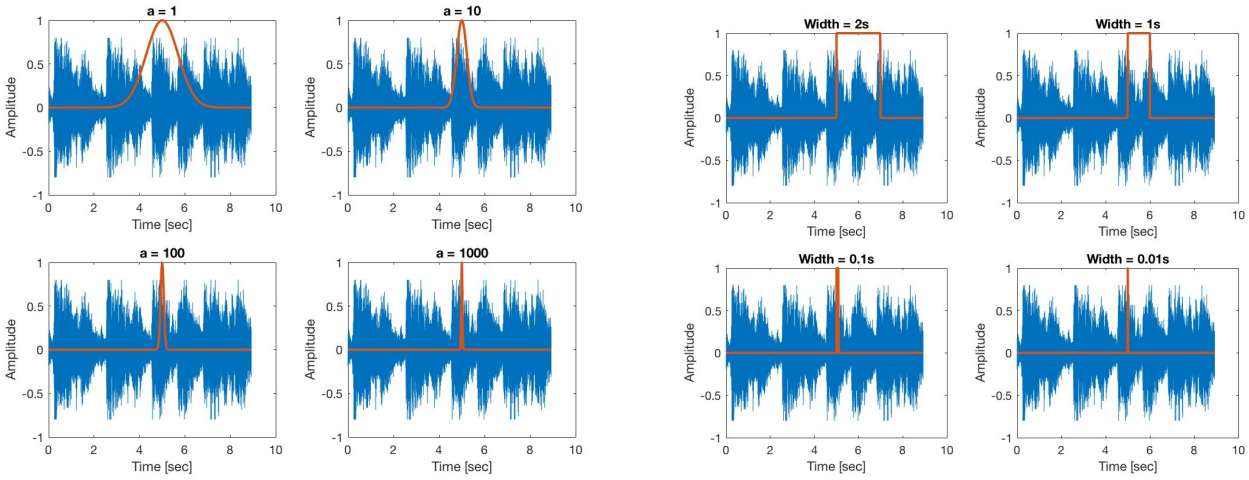

Fig 2: This figure shows variation in filter widths used later in a Gábor transform. Gaussian (left) & Shannon (right).

To vary the width of each filter, a for loop was generated that redefined the filter equation each iteration. The equations for the Gaussian filter and Shannon filter are shown below. The Gaussian filter width can be modified using the constant "a". For this comparison the filter was set to a = 1, 10, 100, 1000. The Shannon filter is a simple set of two step functions, where the width is defined in the second step function as width. The width was set to 2s, 1s, 0.1s, and 0.01s.

Gaussian Filter: $e^{(-a(\tau-tslide))^2}$ Shannon Filter: $H(\tau - tslide) - H(\tau - tslide - width)$

In the second part of the assignment, we filtered two separated recordings of Mary had a Little Lamb. One was recorded using a piano and the other with a recorder. The process for filtering the signals was the same, but each recording had a different amount of overtone due to the different timbre of each instrument. The filtering window used for each Gábor transform is shown in Fig 3. In order to remove the over tones, I also implemented a second filter in the frequency domain centered around the max frequency. This filtered out the overtones and left only the original frequency.

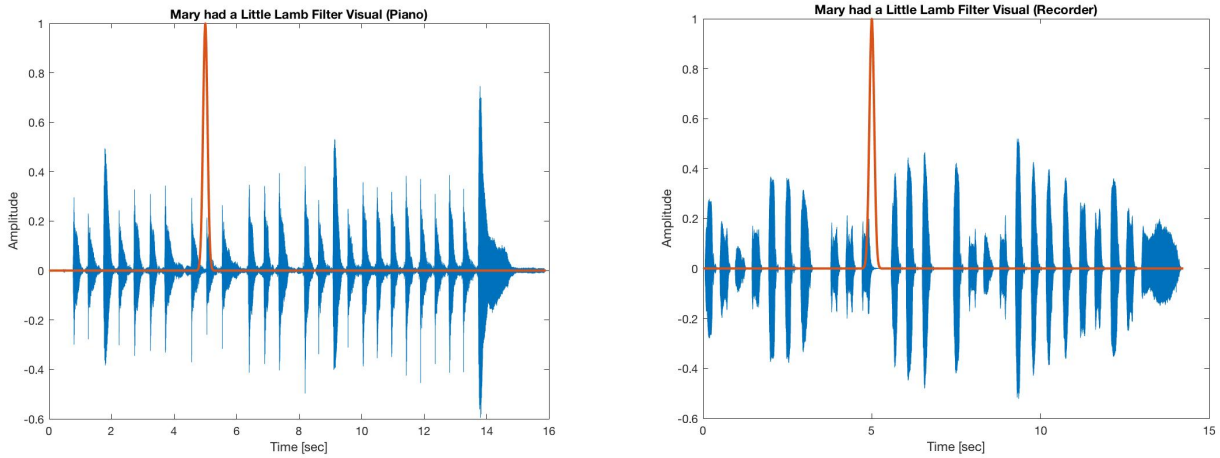

Fig 3: This figure shows the Gaussian filter window used during the Gábor transform for the piano (left) and recorder (right).

Computational Results: Handel Messiah Analysis

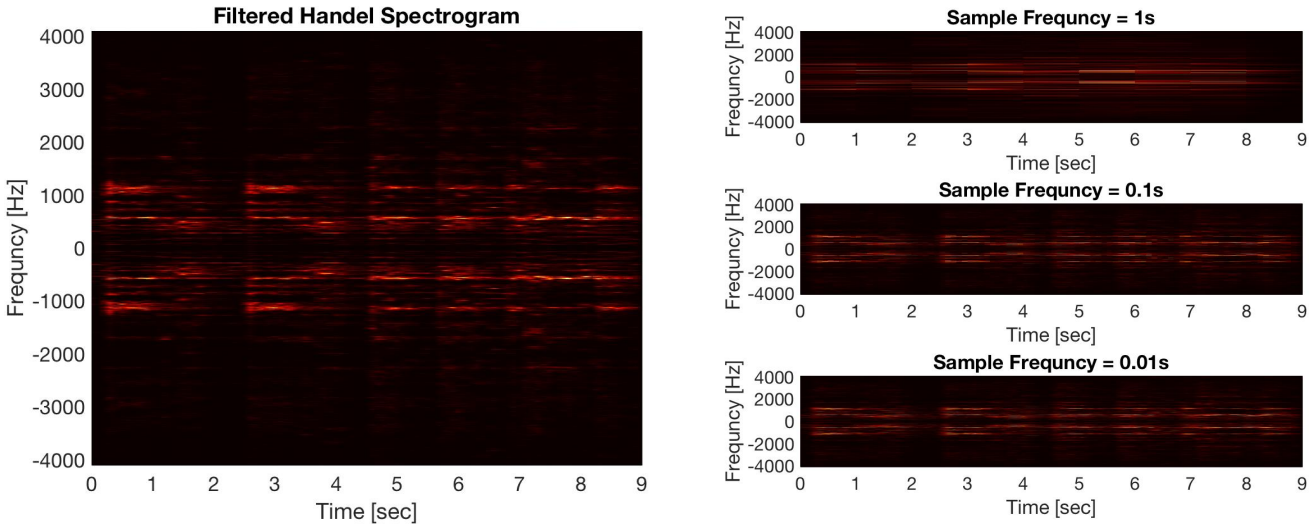

Fig 4: The ideal spectrogram for Handel's Messiah is shown on the left with the varying sampled spectrograms on the right.

A segment from musical piece Messiah by Handel was analyzed to find the most prominent frequencies and notes. The translation step size of the filter window was tested at sampling frequencies of 1s, 0.1s, and 0.001s. As shown in Fig 3, the 1s sample frequency seems to be severely under sampled resulting in a loss of resolution in the time domain. While the sampling frequency of 0.1s and 0.01s seem to be the same meaning 0.01s is probably over sampling, which is computationally expensive with no benefit. The frequencies on the spectrogram shown in the Fig 4 seem to be set around 1000 Hz and 500Hz, which makes sense because this piece is includes very high notes and those frequencies correlate to the upper vocal range for women and men. This spectrogram was able to effectively capture the frequencies in the time domain as shown in the 2 longer notes from seconds 1-4, the shorter notes from 5-6, and the longer trill from seconds 7-9. This spectrogram was even able to capture the variation in amplitude of the frequencies in the first half versus the second half of the piece, which corresponds to the change from a loud projective singing to the softer build up.

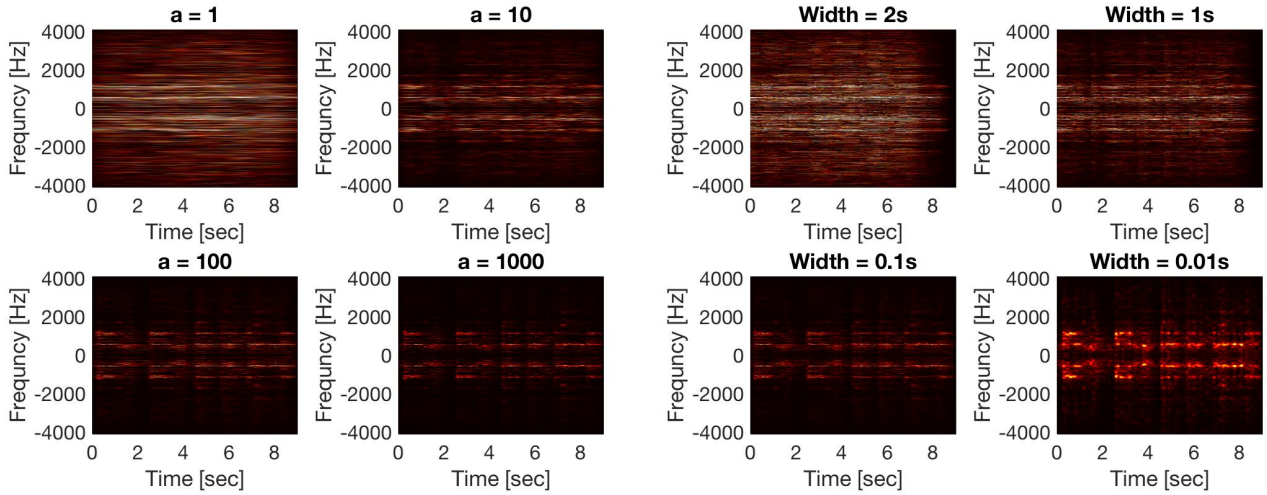

Fig 5: This figure shows the resulting spectrograms with varying window width sizes. The Gaussian filter is on the left and the Shannon filter is on the right. The filters are shown in the time domain in Fig 2.

After establishing the ideal spectrogram for the Gaussian window, I made a new spectrogram with a Shannon window. The spectrograms set at varying window widths are shown in Fig 4. The shape of the two filters are not extremely different so it is understandable that their spectrograms look very similar. But it is interesting to note that the Shannon window can look over sampled very quickly if the width is too small. This is different with the Gaussian window because if the step size is half the window size in a Gaussian there will be less than half overlap in the window. However, with the Shannon filter, there would be exactly half overlap of the window. Therefore, it is easier to oversample with the Shannon filter compared to the Gaussian filter. On the other hand, the Shannon filter is easier to find an ideal window width because if you have a set translation step size and make the window size the same, there will be no overlap and minimal under sampling. This makes the Gaussian filter more robust overall but the Shannon filter is easier to use.

Computational Results: Mary had a Little Lamb Score Generation

The second part of the assignment was to generate the muscal score to Mary had a Little Lamb based on recordings from a piano and recorder. The spectrograms for the Gábor filtered signals before and after filtering out the overtones are shown in Fig 6. The piano frequencies ranged between 330 and 260 Hz, which means it was played in the key of middle C, while the recorder had frequencies ranging from 1000 and 750 Hz, which means it was played in the key of G.

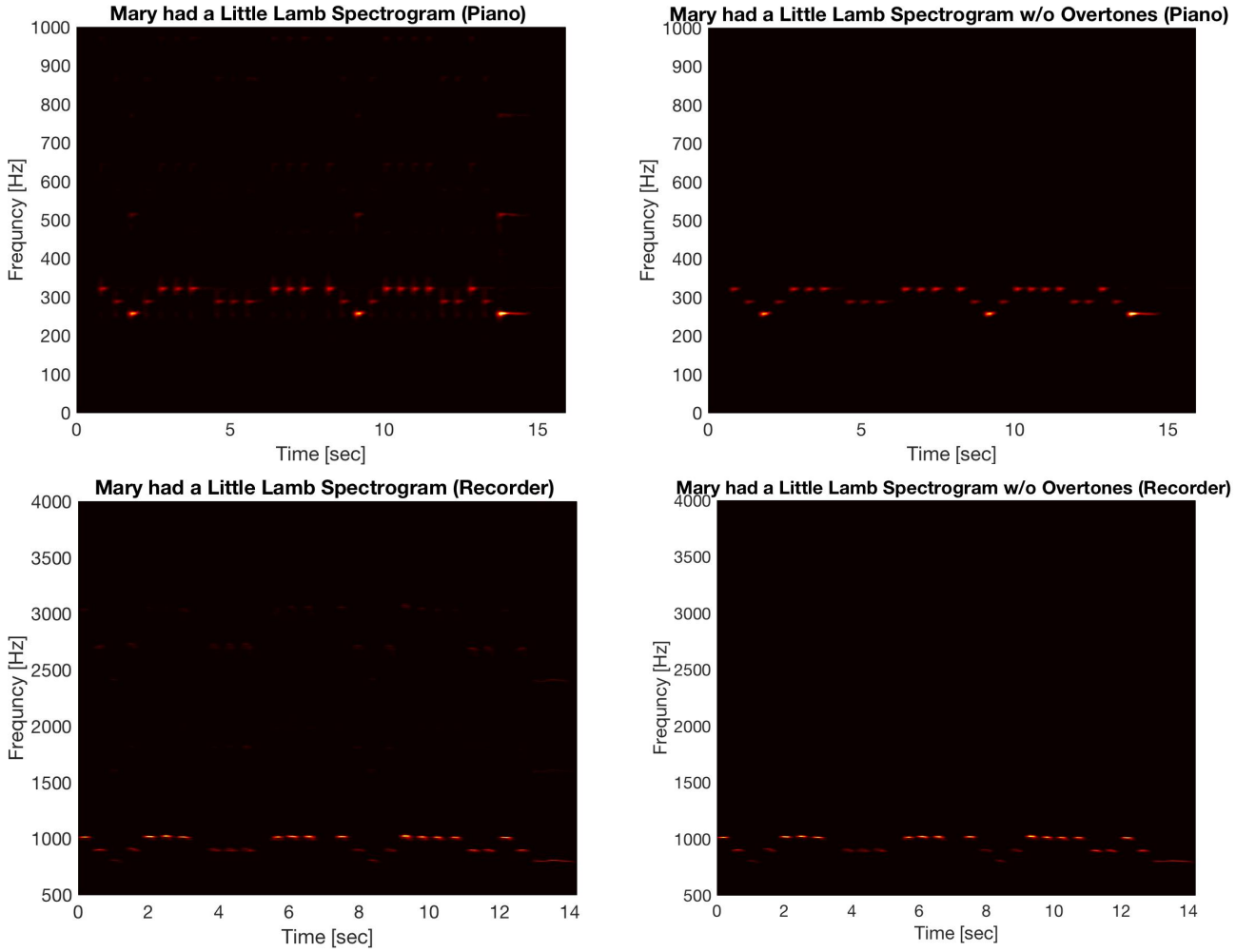

Fig 6: Spectrograms for piano and recorder with overtones on the left and without overtones on the right.

Besides having different frequency ranges, the recorder and piano also have different timbre, which are shown in the spectrogram containing the overtones of the instrument. The piano seems to have each set of overtones at similar amplitudes, while the recorder seems to skip the first overtone and shows the second overtone at higher amplitude. This means the piano has a timbre that has higher octave overtones at a similar magnitudes, and the recorder has a timbre that emphasizes the second octave overtone more than the other octaves. By using the frequencies shown in the spectrograms, I recreated the original score for each instrument shown in Fig 7.

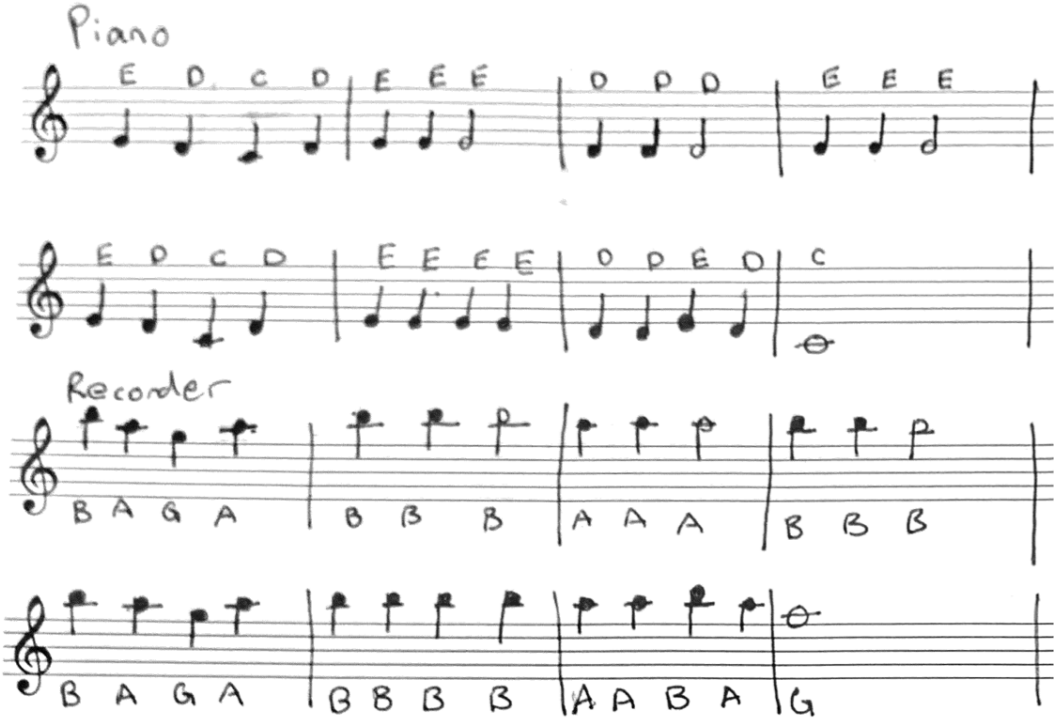

Fig 7: Original score for the piano and recorder based on the spectrograms in fig 6.

Conclusion

In conclusion, the Gábor transform is a powerful signal processing method that captures both time and frequency information. In this paper, the Gábor transform has been shown to filter the complex musical piece of Messiah by Handel and produce the main frequencies of the piece and where they occur in time. Additionally, the Gábor transform was used to recreate a musical scores based on frequencies from recorded music. The Gábor transform is an extremely useful signal processing method that allows analysis of frequency and time, and is compatible with many different filter windows that have varying benefits.

Appendix A: Functions & Explanations

- load / audioread: used to load data from music recordings for analysis.
- fft: used for the Gábor transform, shifts data from the time domain to the frequency domain.
- fftshift: used for plotting in the frequency domain, shifts data to align in the middle of the graph verses the edges.
- heaviside: used in the Shannon filter, MATLAB generated step-function.

Appendix B: Code

```
%% definitions and refresh
close all; clear all; clc;
load handel
v = y';
n = length(y);
L = n/Fs;
t = (1:length(y))/(Fs);
k=(1/L)*[0:((n+1)/2-1) (-n/2):-1];
ks=fftshift(k);
a = [1,10,100,1000];
tslide=0:0.1:9;
width = [2, 1, 0.1, 0.01];
%% Gaus filter
for k = 1:length(a)
    ghold=exp(-a(k)*(t-5).^2);
    subplot(2,2,k)
    plot(t,v);
    hold on
    plot(t,ghold,'linewidth',2)
    xlabel('Time [sec]');
    ylabel('Amplitude');
    title('Signal of Interest, v(n)');
end
subplot(2,2,1)
title('a = 1')
xlabel('Time [sec]');
ylabel('Amplitude');
subplot(2,2,2)
title('a = 10')
xlabel('Time [sec]');
ylabel('Amplitude');
subplot(2,2,3)
title('a = 100')
xlabel('Time [sec]');
ylabel('Amplitude');
subplot(2,2,4)
title('a = 1000')
xlabel('Time [sec]');
ylabel('Amplitude');
%% Gaus plot
figure(2)
for k = 1:length(a)
    Sgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        g=exp(-a(k)*(t-tslide(j)).^2);
        Sg=v.*g;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = abs(fftshift(Sgt)); % We don't want to scale it
    end
    subplot(2,2,k)
    pcolor(tslide,ks,Sgt_spec.'),
    shading interp
    set(gca,'Fontsize',16)
    colormap(hot)
end
subplot(2,2,1)
```

```
title('a = 1')
xlabel('Time [sec]');
ylabel('Frequncy [Hz]');
subplot(2,2,2)
title('a = 10')
xlabel('Time [sec]');
ylabel('Frequncy [Hz]');
subplot(2,2,3)
title('a = 100')
xlabel('Time [sec]');
ylabel('Frequncy [Hz]');
subplot(2,2,4)
title('a = 1000')
xlabel('Time [sec]');
ylabel('Frequncy [Hz]');
%% Sampling freq
tspace = [1, 0.1, 0.01];
for k = 1:length(tspace)
    tslide = 0:tspace(k):9;
    Sgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        g=exp(-100*(t-tslide(j)).^2);
        Sg=v.*g;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = abs(fftshift(Sgt)); % We don't want to scale it
    end
    subplot(3,1,k)
    pcolor(tslide,ks,Sgt_spec.'),
    shading interp
    set(gca,'Fontsize',16)
    colormap(hot)
end
subplot(3,1,1)
title('Sample Frequncy = 1s')
xlabel('Time [sec]');
ylabel('Frequncy [Hz]');
subplot(3,1,2)
title('Sample Frequncy = 0.1s')
xlabel('Time [sec]');
ylabel('Frequncy [Hz]');
subplot(3,1,3)
title('Sample Frequncy = 0.01s')
xlabel('Time [sec]');
ylabel('Frequncy [Hz]');
%% Shannon plot
for k = 1:length(width)
    for j=1:length(tslide)
        g = -heaviside(t-tslide(j)-width(k)) + heaviside(t-tslide(j));
        Sg=v.*g;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = abs(fftshift(Sgt)); % We don't want to scale it
    end
    subplot(2,2,k)
    pcolor(tslide,ks,Sgt_spec.'),
    shading interp
    set(gca,'Fontsize',16)
    colormap(hot)
end
```

```
subplot(2,2,1)
title('Width = 2s')
xlabel('Time [sec]');
ylabel('Frequncy [Hz]');
subplot(2,2,2)
title('Width = 1s')
xlabel('Time [sec]');
ylabel('Frequncy [Hz]');
subplot(2,2,3)
title('Width = 0.1s')
xlabel('Time [sec]');
ylabel('Frequncy [Hz]');
subplot(2,2,4)
title('Width = 0.01s')
xlabel('Time [sec]');
ylabel('Frequncy [Hz]');
%% Shannon filter
figure()
for k = 1:length(width)
    ghold = -heaviside(t-5-width(k)) + heaviside(t-5);
    subplot(2,2,k)
    plot(t,v);
    hold on
    plot(t,ghold,'linewidth',2)
end
subplot(2,2,1)
title('Width = 2s')
xlabel('Time [sec]');
ylabel('Amplitude');
subplot(2,2,2)
title('Width = 1s')
xlabel('Time [sec]');
ylabel('Amplitude');
subplot(2,2,3)
title('Width = 0.1s')
xlabel('Time [sec]');
ylabel('Amplitude');
subplot(2,2,4)
title('Width = 0.01s')
xlabel('Time [sec]');
ylabel('Amplitude');

%% piano + filter
close all; clear all; clc;
[ypiano,Fs] = audioread('music1.wav');
Fs = Fs/16;
vpiano = ypiano';
vpiano = downsample(vpiano, 16);
tr_piano=length(vpiano)/Fs;
npiano = length(vpiano);
tpiano = (1:npiano)/Fs;
a = 100;
k=(1/tr_piano)*[0:(npiano/2-1) (-npiano/2):-1];
ks=fftshift(k);
tslide=0:0.1:tr_piano;
Sgt_spec = zeros(length(tslide),npiano);

 for j=1:length(tslide)
```

```
        g=exp(-a*(tpiano-tslide(j)).^2);
        Sg=g.*vpiano;
        Sgt=fft(Sg);
        [maxt,tin]=max(abs(Sgt));
        Sgtf = Sgt.*exp(-0.01*(k-k(tin)).^2);
        Sgt_spec(j,:) = fftshift(abs(Sgtf));
 end
figure()
pcolor(tslide,ks,Sgt_spec.'),
shading interp
set(gca,'Ylim',[0,1000],'Fontsize',16)
colormap(hot)
title('Mary had a Little Lamb Spectrogram w/o Overtones (Piano) ')
xlabel('Time [sec]');
ylabel('Frequncy [Hz]');

figure()
plot((1:length(vpiano))/Fs,vpiano);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a Little Lamb Filter Visual (Piano)');
hold on
g=exp(-100*(tpiano-5).^2);
plot(tpiano,g,'linewidth',2)


%% recorder + filter

close all; clear all; clc;
[yrec,Fs] = audioread('music2.wav');
vrec = yrec';
tr_rec=length(vrec)/Fs;
nrec = length(vrec);
trec = (1:nrec)/Fs;
a = 100;
k=(1/tr_rec)*[0:(nrec/2-1) (-nrec/2):-1];
ks=fftshift(k);
tslide=0:0.1:tr_rec;
Sgt_spec = zeros(length(tslide),nrec);

 for j=1:length(tslide)
        g=exp(-a*(trec-tslide(j)).^2);
        Sg=g.*vrec;
        Sgt=fft(Sg);
        [maxt,tin]=max(abs(Sgt));
        Sgtf = Sgt.*exp(-0.001*(k-k(tin)).^2);
        Sgt_spec(j,:) = fftshift(abs(Sgtf));
 end
figure()
pcolor(tslide,ks,Sgt_spec.'),
shading interp
set(gca,'Ylim',[500,4000],'Fontsize',16)
colormap(hot)
title('Mary had a Little Lamb Spectrogram w/o Overtones (Recorder) ')
xlabel('Time [sec]');
ylabel('Frequncy [Hz]');

figure()
```

```
plot((1:length(vrec))/Fs,vrec);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a Little Lamb Filter Visual (Recorder)');
hold on
g=exp(-100*(trec-5).^2);
plot(trec,g,'linewidth',2)
```