AMATH 482 Homework 2: Gábor Transforms

Rose Ju

February 7, 2020

Abstract

The purpose of this report is to analyze a portion of the Handel's Messiah with time frequency analysis. We will be using Gábor filtering to produce the spectrogram of the piece of classic work. Then, we will be exploring the different window widths and sampling rate to see how they affect the spectrogram. Lastly, we will be using different Gábor windows to see to see how they influence the spectrograms. Additionally, we will be applying Gábor filtering to a specific piece of music on the recorder and piano and reproduce the music score for the piece. Then we will be comparing and contrasting on the differences between a recorder and piano in their time-frequency analysis.

I. Introduction and Overview

We will be analyzing a portion of Handel's Messiah with a time frequency analysis. The time frequency analysis comprises the techniques that study a given signal simultaneously in both time and frequency domains. Often times, this analysis is used for audio signals due to their variety of frequencies at different times. In this report, we will use the Gábor filtering to create the transformed spectrograms while using different functions or wavelets as the sliding window.

Then we will be examining the effects of the various parameters, such as window widths, sampling rates and different functions to the spectrograms.

Additionally, we will be exploring the music piece *Mary Had a Little Lamb* from the recorder and piano version. We will be using the Gábor transform to generate their musical scores. Then, we will be analyzing the differences between the piano and recorder version of the transformed spectrogram.

II. Theoretical Background

i. Gábor Transform

The Gábor transforms is a special case of short time Fourier transform. It is defined as

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

where $g_{(t,\omega)}(\tau) = e^{i\omega\tau}g(\tau - t)$ and the bar denotes the complex conjugate of the function.

The function $g(\tau - t)$ is a time filter for localizing the signal over a chosen time window. By integrating the time filtering windows down the entire signal over the parameter τ, we are able to pick up the frequency information at every second. τ represents the time at which the function is centered and ω is the width of the function. Thus, we can see that Gábor transform is a function constructed by τ and ω.

Gábor transform is an easy way to analysis both the time and frequency information simultaneously. Whereas in Fourier analysis, the frequency is highly accurate but by tradeoff it cannot localize in time. Similarly, for the time series method, accuracy is achieved for the time domain meanwhile all its frequency resolution is lost. Therefore, the Gábor transform takes in both methods and exchanges some measures of the time accuracy for the frequency resolution. It translates a short time window and scales it to better capture its time resolution.

ii. Gábor Windows and Wavelets

There are many functions that can be used for modifying the scaling Gábor window in order to improve the time resolution of the Gábor transform. One of the most common function used is the Gaussian function. It is defined as

$$g(t) = e^{-a(t-b)^2} \quad (2)$$

where a defines the width of the filter and b defines the center of the function.

In this report, we are going to be exploring the Mexican hat wavelet. It is defined as

$$\psi(t) = \frac{2}{\sqrt{3\sigma}\pi^{\frac{1}{4}}}(1 - \left(\frac{t}{\sigma}\right)^2 e^{-\frac{t^2}{2\sigma^2}} \tag{3}$$

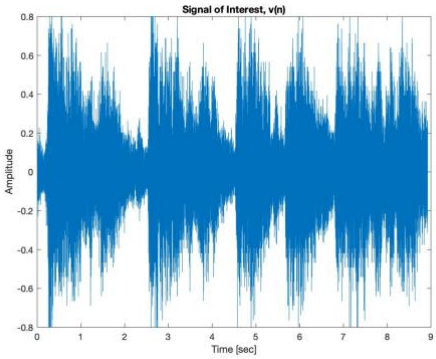
where σ is the width of the wavelet. The Mexican hat wavelet is also the second derivative of the Gaussian wave, defined as,

$$g(t) = (1 - ct)e^{-a(t-b)^2} \tag{4}$$

where c is the maximum negative value of the function. Another window function that we will be using is the Shannon window. The Shannon filter is simply a step function window with value of zero outside of the specified domain.

III. Algorithm Implementation and Development

- Load in the audio data file and calculate the length and the number of points (or sampling rate) for the audio file. Since the original data is in a column form, we need to transpose the data.



(Figure 1: The signal of Interest which will be analyzed later)

- Before applying the fast Fourier Transform, we need to rescale the frequency domain by $\frac{2\pi}{L}$ because the algorithm is 2π periodic signals. Then, we apply `fftshift()` to get the shifted frequencies.
- Set up a time-slide vector that translates the filter for the sliding window.
- Use a loop to translate a function `func` with the time-slide vector. The function will be set to be either a Gaussian window, Mexican hat wavelet or a Shannon window (step function window).

- After successfully setup the filter, we multiply it by the audio signal and perform a Fourier transform. Then we take the absolute value to the multiplication and apply `fftshift()` to the absolute value since we used Fourier transform. Finally, we get the matrix for the resulted Gábor transform spectrogram at each time step.
- Plot the signal with each of the translating window and along with their corresponding spectrograms.
- Explore the effects of width of Gábor transform. We will be changing the α value in equation (2) which is the width of the filter.
- Explore the effects of sampling rate. We will change the time steps in time-slide vector.
- Next part of the study, we will be redoing the steps previously to produce the music score for both the recorder and the piano. Then, we will be experimenting with the width of the filter and the time steps for the signals to find the best matching frequencies.

IV. Computational Results

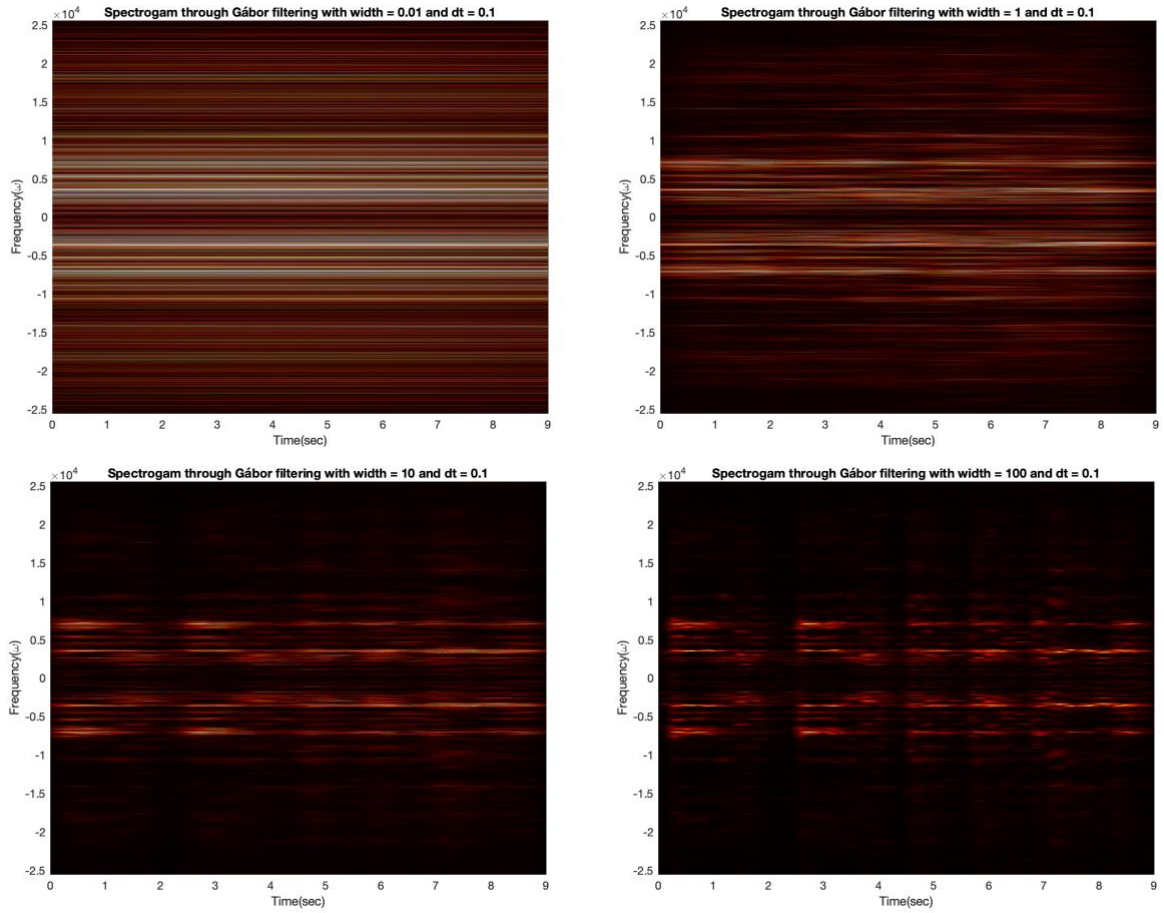


Figure 2: Spectrograms demonstrating the increase in width with $\Delta t = 0.1$

In figure 2, we showed the Spectrogram of the music piece of Handel and investigated in the change of window width while remaining the time steps constant. We can see from the picture that as we increase the value of a, the width of the window decreases. With a large a value, the figure can easily allocate the signal in the time content. If we were to compare the four spectrograms, we find that, with a smaller a value, the graph gives more accurate frequency information. However, in return, we are not able to determine the exact time of the sign.

In figure 3, we showed the Spectrogram of the music piece changing the time step value within the time slide vector while remaining a constant. We define the spectrograms with large Δt to be under-sampling and small Δt to be over-sampling. The figure shows the four generated spectrogram results. We can see that when we have under-sampling it is difficult to get an accurate read on frequency since not all of the data is read. All the frequencies are divided into big blocks. On the other hand, we get accurate frequency and time resolution for oversampling. However, the run-time of the method would also greatly increase.

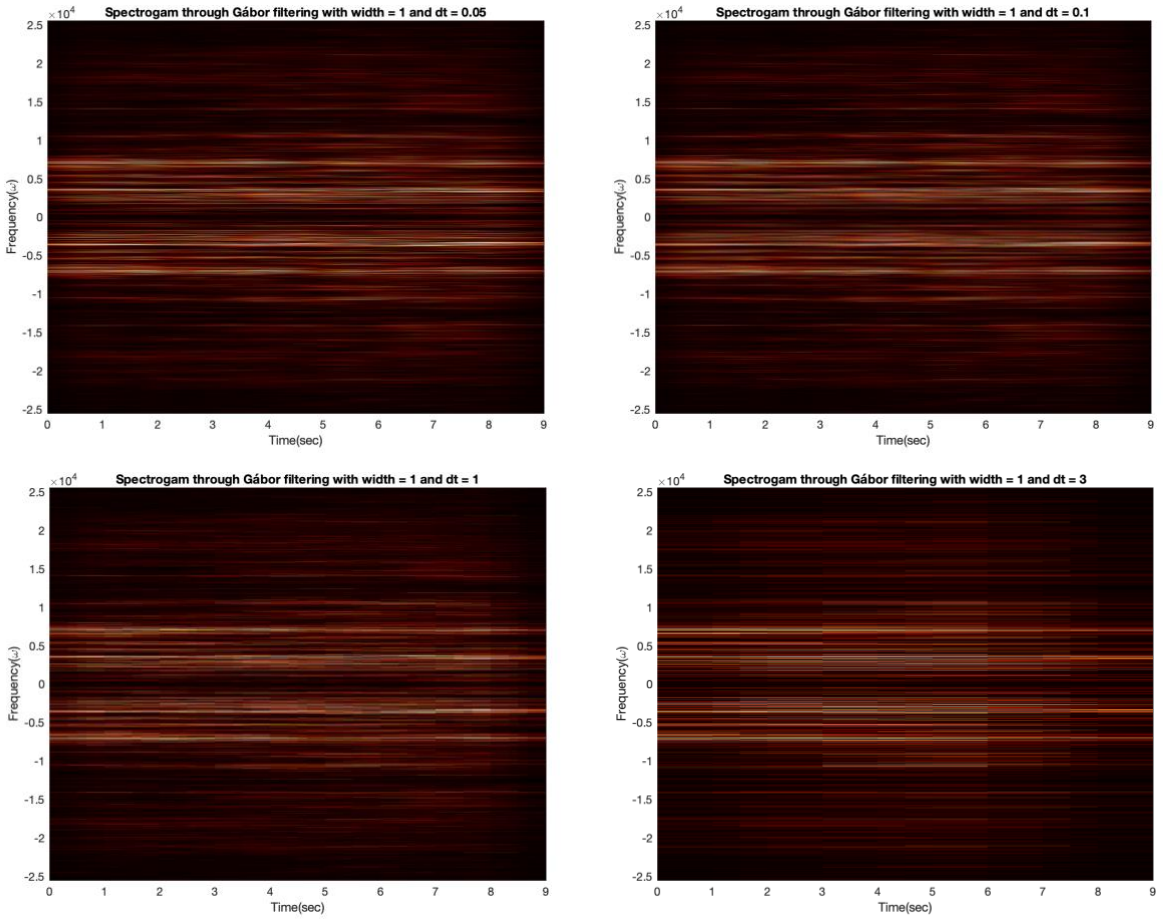


Figure 3: Spectrograms demonstrating change in Δt with width = 0.1

Next, we explored the spectrograms produced using different functions, Gaussian function, Mexican hat function and Shannon window, shown in Figure 4. For the Mexican hat function, we used $\Delta t = 0.1$ with width = 0.5. Then for the Shannon window, we set $\Delta t = 0.1$ with width =1. We can see that the spectrogram generated by the Mexican hat wavelet results in accurate resolutions in both frequency and time domain. However, the Shannon window demonstrates a poor result. It gives inaccurate resolutions in both time and frequency. However, we can clearly see that the gaussian window has by far the best resolutions in both factors.

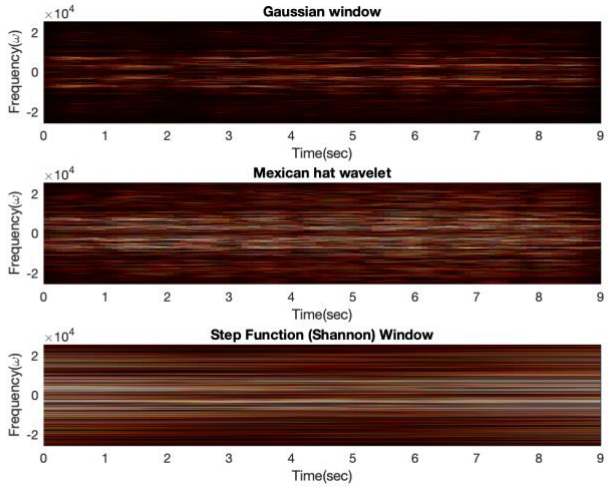


Figure 4: Spectrograms generated by different functions

In the next part, we analyzed the piano and the recorder version of “Mary had a Little Lamb”. The results are shown in Figure 5. For both spectrograms, we used Gábor transform and the Gaussian function with 0.1 sec time step. However, for computational and resolution purposes, we take the window width a to be 10 for the piano version and 50 for the recorder version. Base

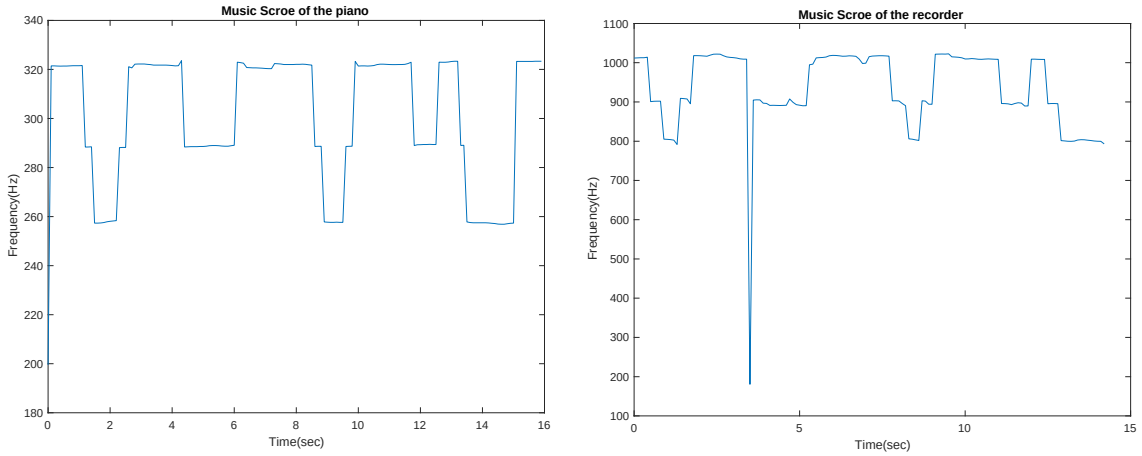


Figure 5: Music scores for the piano and recorder by Gaussian window

on music scale with respect to the frequency, we find the music scores for the piano to be: EDCDEEEDDDEEE EDCDEEEEDDECD. For the music scores of the recorder, it is CAGACCCAAACCC CAGACCCCAAGA. We can see the difference between the piano and the recorder from the Figure 5. The frequency from the recorder is much higher than the frequency of the piano despite for the fact that they have the same shape. The reason why they have a difference in resulted frequency is because the type of instruments is different and the sounds are unique to each instrument.

V. Summary and Conclusions

We used Gábor transform to create spectrograms for audio signals. In this study, we explored the effects of the window width, time steps and different functions. We found that window width with 1 and time step of 0.1 seconds would generate an accurate frequency and time resolution. In addition, we found that with increasing window width we get accurate frequency resolution but loses the accuracy time values. Then we found that with large time steps, we are unable to read every frequency. However, with small time steps, over-sampling significantly increases our run time.

We found that the difference between piano and recorder version of “Mary had a Little Lamb” is their frequency range. They both have the same shape in the end result. However, the frequency of the piano ranges from 260 Hz to 320 Hz. Whereas, the recorder version of the music ranges from 880 Hz to 1000 Hz. The difference in frequencies is expect due to the different instruments and the way they sound.

VI. Appendix A – MATLAB functions

`abs(X)`– calculates the absolute value of the input element. If input in complex, then returns the complex magnitude. This was used to normalize our dataset.

`fftn(X)` **–** performs fast fourier transform to the given data. We used this to transform our data set into frequency domain.

`fftshift(X)` **–** rearranges the elements of X and place 0-frequency to the center of the vector. We used this to translate our data for better visualization.

`load` – load data from MAT-file into the tensor. We used this to load the dataset into our workspace.

`max(X)` – calculates the maximum value of the input element. The function was used to find the maximum

VII. Appendix B – MATLAB codes

```
close all; clear all; clc;
load handel
v = y';
L = 9;
n = length(v);
t = (1:length(v))/Fs;
k = (2*pi/L) * [0:(n-1)/2 -n/2:-1];
ks = fftshift(k);
plot(t,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');
%% Gaussian a = (0.01,1,10,100) dt = 0.1
a = [0.01,1,10,100];
dt = 0.1;
ts = 0:dt:L;

for j = 1:length(a)
    sgtspec = [];
    curr = a(j);
    for i = 1:length(ts)
        func = exp(-curr*(t-ts(i)).^2);
        sg = func.*v;
        sgt = fft(sg);
        sgtspec =[sgtspec; abs(fftshift(sgt))];
    end
    figure(j)
    pcolor(ts, ks, sgtspec.'), shading interp
    xlabel('Time(sec)');
    ylabel('Frequency(\omega)');
    title("Spectrogam through G·bor filtering with width = " + num2str(curr)
+ " and dt = 0.1");
    colormap(hot)
end

%% Gaussian a = 1 dt = (0.05,0.1,0.5,1)
a = 1;
dt = [0.05,0.1,1,3];

for j = 1:length(dt)
    sgtspec = [];
    ts = 0:dt(j):L;
    for i = 1:length(ts)
        func = exp(-a*(t-ts(i)).^2);
        sg = func.*v;
        sgt = fft(sg);
        sgtspec =[sgtspec; abs(fftshift(sgt))];
    end
    figure(j)
    pcolor(ts, ks, sgtspec.'), shading interp
    xlabel('Time(sec)');
    ylabel('Frequency(\omega)');
```

```
    title("Spectrogam through G·bor filtering with width = 1 and dt = " +
num2str(dt(j)));
    colormap(hot)
end

%% Gaussian filter
sgtspec = [];
a = 10;
ts = 0:0.1:L;
for i = 1:length(ts)
    func = exp(-a*(t-ts(i)).^2);
    sg = func.*v;
    sgt = fft(sg);
    sgtspec =[sgtspec; abs(fftshift(sgt))];
end
subplot(3,1,1);
pcolor(ts, ks, sgtspec.'), shading interp
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
title("Gaussian window");
colormap(hot)

% Mexican hat wavelet
sgtspec = [];
a = 0.5;
ts = 0:0.1:L;
for i = 1:length(ts)
    func = 2/(sqrt(3*a)*pi^(1/4))*(1-((t-ts(i))/a).^2).*exp(-(t-
ts(i)).^2/(2*a^2));
    sg = func.*v;
    sgt = fft(sg);
    sgtspec =[sgtspec; abs(fftshift(sgt))];
end
subplot(3,1,2);
pcolor(ts, ks, sgtspec.'), shading interp
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
title("Mexican hat wavelet");
colormap(hot)

% Step Function (Shannon) Window
sgtspec = [];
a = 1;
ts = 0:0.1:L;
for i = 1:length(ts)
    func = abs(t-ts(i));
    sg = func.*v;
    sgt = fft(sg);
    sgtspec =[sgtspec; abs(fftshift(sgt))];
end
subplot(3,1,3);
pcolor(ts, ks, sgtspec.'), shading interp
xlabel('Time(sec)');
ylabel('Frequency(\omega)');
title("Step Function (Shannon) Window");
colormap(hot)
```

```
%%
[y,Fs] = audioread('music1.wav');
y = y'/2;
tr_piano=length(y)/Fs; % record time in seconds
n = length(y);
L = tr_piano;
t = (1:length(y))/Fs;
k = (2*pi/L) * [0:(n-1)/2 -n/2:-1];
ks = fftshift(k);
pfreq = [];
sgtspec = [];

ts = 0:0.1:L;
a = 10;
for i = 1:length(ts)
    func = exp(-a*(t-ts(i)).^2);
    sg = func.*y;
    sgt = fft(sg);
    [V,I] = max(abs(sgt));
    pfreq = [pfreq; abs(k(I))];
    sgtspec =[sgtspec; abs(fftshift(sgt))/max(abs(sgt))];
end

figure(1)
plot(ts,pfreq/(2*pi))
title('Music Scroe of the piano');
xlabel('Time(sec)');
ylabel('Frequency(Hz)');

%%
[y,Fs] = audioread('music2.wav');
y = y'/2;
tr_piano=length(y)/Fs; % record time in seconds
n = length(y);
L = tr_piano;
t = (1:length(y))/Fs;
k = (2*pi/L) * [0:(n-1)/2 -n/2:-1];
ks = fftshift(k);
pfreq = [];
sgtspec = [];

ts = 0:0.1:L;
a = 50;
for i = 1:length(ts)
    func = exp(-a*(t-ts(i)).^2);
    sg = func.*y;
    sgt = fft(sg);
    [V,I] = max(abs(sgt));
    pfreq = [pfreq; abs(k(I))];
    sgtspec =[sgtspec; abs(fftshift(sgt))/max(abs(sgt))];
end

figure(1)
plot(ts,pfreq/(2*pi))
title('Music Scroe of the recorder');
```

```
xlabel('Time(sec)');
ylabel('Frequency(Hz)');
```