Gábor Transforms and Time-Frequency Analysis

Yuxin Jiang
AMATH 482 HW2

Abstract

Although the Fourier Transform is one of the most effective methods for analysis of signals, in order to explore a signal with various frequencies, Gábor Transform is used to localize both time and frequency. In this report, we explored the method of Gábor Transform using a portion of Handel's Messiah by using Gábor filtering, changing the width of the Gábor Transform, exploring oversampling and undersampling, and utilizing different Gábor windows. We also filtered out the music score of piano clip and recorder clip of Mary Had a Little Lamb using Gábor filtering.

I. Introduction and Overview

Limitations of Fourier tranform such as failure to capture the *moment in time* and failure to analyze non-stationary signal occurred. Therefore, we utilized Gábor Transform to localize both time and frequency properties in a given signal. Since the method is computed by discretizing both the time and frequency domain, situations like oversampling and undersampling need to be avoided, and choice of window widths, different types of Gábor windows, and translations need to be considered to ensure the accuracy. The report would illustrate the features in two parts.

Part I is based on a 9-second piece of music from Handel's Messiah. With application of Gábor filtering, we would visualize the influence of different window widths, different translations, and different types of Gábor windows by the spectrograms. Part II is based on two music clips of Mary Had a Little Lamb in piano and recorder version. We reproduced the music scores after filtering and explored the difference of recorder and piano on overtones.

II. Theoretical Background

A. Gábor Transform and Short-time Fourier Transform (STFT)

The Gábor Transform, or the Short-time Fourier Transform (STFT), is defined as

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

where the bar denotes the complex conjugate of the function, with the function $g(\tau - t)$ is utilized as a time filter to localize the signal over a specific window of time. The Gábor [1]Transform also provides two key principles for joint time-frequency analysis, translation of a short-time window, τ, and scaling of a short-time window, a. However, information of any signal outside of the window provided by Gábor Transformation is lost. We could extract the spectral information by moving τ through the data in a specific time range, the width a.

[1] J.Nathan Kutz, *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*

If $0 < t_0,\ \omega_0 < 1$, then the signal is over-sampled. If $t_0,\ \omega_0 > 1$, then the signal is under-sampled. There are[2] also drawbacks of the Gábor Transform due to the Heinsenberg relationship. The shorter time filtering window will offer us less information about the frequency contents, while the longer time filtering window will retain more frequency components with the expense of losing time components, in contrast.

B. Wavelets

The fundamental principle of wavelet theory is to allow the scaling window a to vary to extract time resolution. Wavelet is originated from the fact that scaling window explores through smaller and smaller pieces of waves from the large signal. The wavelet function starts with the mother wavelet:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$$

where $a \neq 0$ and b are real constants. a is the scaling parameter, and b is the translation parameter previously denoted by τ. There are various mother wavelets designed specifically to meet beneficial properties for a given problem. In part I, we used Gaussian window, Mexican hat wavelet and Shannon window for comparison. In part II, we used the Gaussian window.

The Gaussian window is defined as

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

The Mexican hat wavelet is defined as

$$g(t) = \frac{2}{\sqrt{3a}\pi^{\frac{1}{4}}}\left(1 - \left(\frac{t-b}{a}\right)^2\right)e^{-\frac{t-b^2}{2a^2}}$$

The Shannon window is defined as

$$g(t) = |t-b| < a$$

where a is the window width parameter, and b is the translation parameter. For Gaussian window, the larger a is, the narrower the window width is, given the negative sign before a.

III. Algorithm Implementation and Development

Part I

We were given a 9-second music piece with 73113 sample points with first and last data as 0. In order to keep an even number for the size, I only kept the first 73112 points, which is also the Fourier mode. I defined the length as 9 since it is a 9-second audio. I then defined the vector 't' using **(1:length(v))/Fs** which is the time. I have also created the wavenumbers 'k' and rescaled it by $\frac{2\pi}{L}$ since the FFT assumes 2π periodic domain. I have also used **fftshift** to get 'ks'.

I created the variable 'a' as the window width, and 'tslide' to denote the translation of time. I then created a for loop to iterate through the 'tslide' so that the data is centered around it throughout the process. After defining the Gábor filter as 'g', I multiplied our data 'v' by the filter and applied Fourier transform by using **fft**. To keep track of the data, I created a matrix 'vgt_spec' to include the absolute value of the filtered 'v' and applied **fftshift** to the results to keep the data in

[2] J.Nathan Kutz, *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*

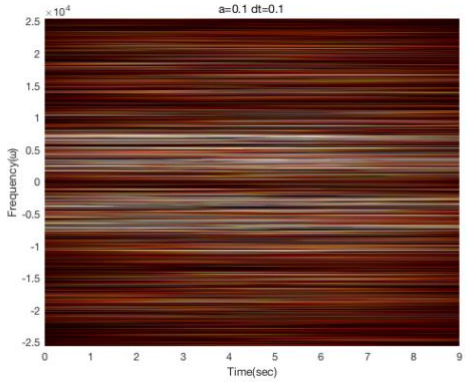

Figure 1: Spectrogram with Large Window Width

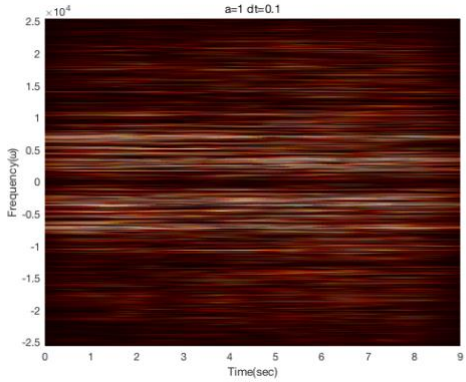

Figure 2: Spectrogram with Medium Window Width

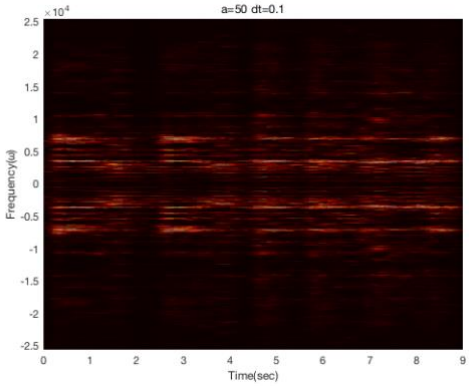

Figure 3: Spectrogram with Small Window Width

mathematically correct order because the Fourier transform has shifted the data. After the results are stored in the matrix 'vgt_spec' at every time step, I was able to plot the spectrogram using the matrix by **pcolor** and setting it in a heat map by **colormap(hot)**.

In order to explore the influence of window width, translations, and different Gábor windows, I changed several parameters to compare the different spectrograms. I have changed the value 'a' to adjust different window width, with 50 representing a really narrow window, 1 as a fair window and 0.1 as a really large window. I have also changed the translation parameter in 'tslide' to 0.1, 0.05 and 5, showing the effect of fair sampling, oversampling and undersampling. Also, I have also changed the filter function 'g' in the for loop from Gaussian filter to Mexican hat wavelet and Shannon wavelet to see the influence of different Gábor windows. I repeated the algorithm described in the first paragraph with different parameters and functions to see the influence of window width and translations.

Part II

In part II I only used the Gaussian filter with $a = 30\ and\ \Delta\ t = 0.2$ to reproduce the audio score. I created length, Fourier mode, time, frequency and window width just as the previous part. For every iteration in the for loop, I used the Gaussian filter and used **fft** on the results. Additionally, in order to get a clearer picture of the scores and overtone filtering, I used the **max** command in order to find the indexes the corresponding wavenumbers in frequency domain, taking the absolute values. Because we wished to see the score in Hertz to match the music notes provided, we needed to divide the results by 2π to change from the angular frequency ω. I then plotted the results in Hertz and also the spectrograms. I repeated my steps for both the piano and recorder version for comparison.

IV. Computational Results

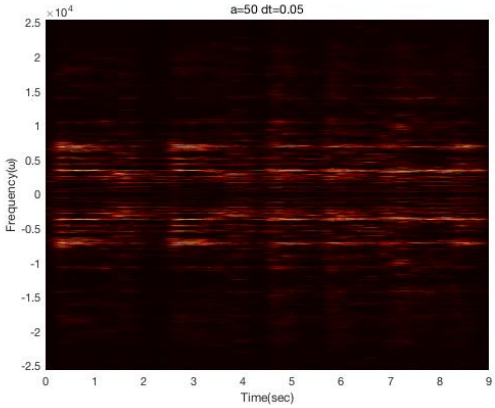

Figure 4: Spectrogram of Oversampled Data

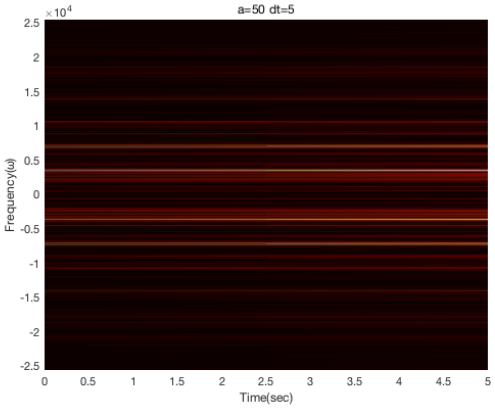

Figure 5: Spectrogram of Undersampled Data

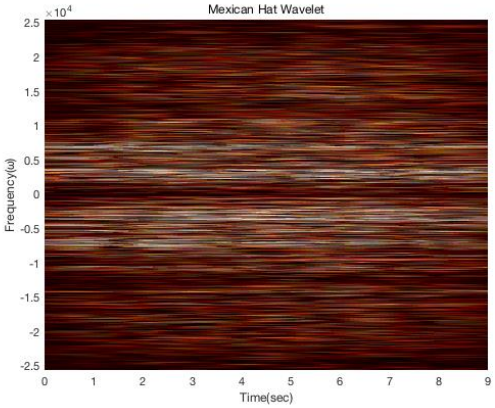

Figure 6: Spectrogram of Mexican Hat Filtered data

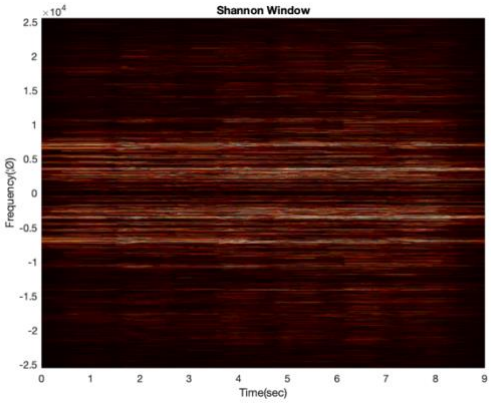

Figure 7: Spectrogram of Shannon Filtered data

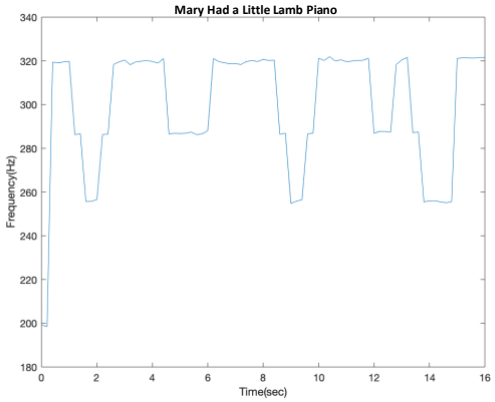

Figure 8: Music Score of Piano Version

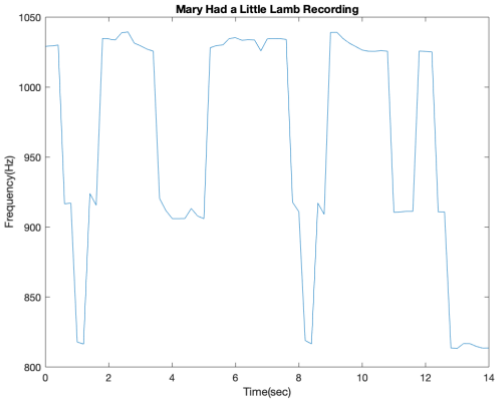

Figure 9: Music Score of Recorder Version

Part I

Spectrograms with different window width, translations and Gábor windows are shown above. Comparing figure 1 and figure 3, we can see that a broad window width can offer us a good resolution in frequency but nearly no information regarding time. Comparing figure 2 and figure 3, we could see that a narrower window width provide us with great information in time resolution but also trades off the frequency information. Overall, $a = 50$ provides us with a better picture and I chose $a = 50$ to test different translations. Comparing figure 3 and figure 4, it is not obvious but from the little difference, if we have small translation, difference between each time fraction is not as obvious, which is the case of oversampling. Similarly, comparing figure 3 and figure 5,

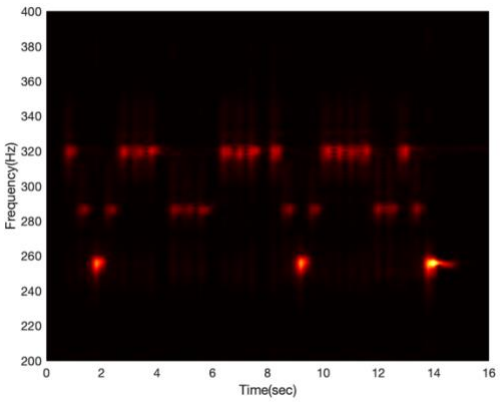

Figure 10: Spectrum of Piano Version

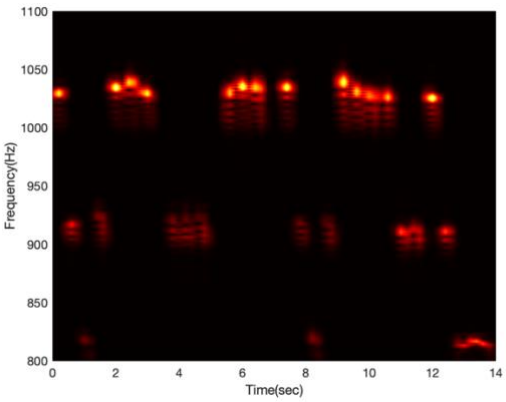

Figure 11: Spectrum of Recorder Version

we could not get good resolutions on both time and frequency since the translation is too large, which is the case of undersampling. I then tested the Mexican Hat wavelet and Shannon window with $a = 1$ and $\Delta\, t = 0.1$. Comparing figure 3, 6 and 7, we could find out given same parameters, Mexican Hat wavelet and Shannon window offered us better frequency resolution than Gaussian window, where Gaussian window offered us better time resolution.

Part II

From figure 8 and 9, we could clearly see the change of frequency. According to the music scale provided, the music score for piano clip is:

E D C D E E E D D D E E E E D C D E E E E D D E D C

The score for the recorder version is:

B A G A B B B A A A B B B B A G A B B B B A A B A G

Comparing figure 1 and 2, we could see that both the piano and recorder follows the same pattern but the range of the frequency, or the tone, is around 240-340 for piano, and 700-1050 for the recorder. We could also see that there seems to be more overtones and undertones in piano but mostly undertones in recorder based on the residuals outside of the brightest red dots.

V. Summary and Conclusions

In conclusion, while using Gábor transform, we should first set a reasonable translation based on the data length to avoid oversampling and undersampling problems. Then we need to choose appropriate window width based on the translation chosen, so that both time and frequency resolutions could be evaluated accurately. Additionally, we should take different types of wavelets into consideration since different Gábor windows aimed at different beneficial properties for a given problem. In our case in part I, Gaussian window with window width as 50 and translation as 0.1 would be a relatively rational choice. In part II, we used Gábor filtering to reproduce the music scores and observed more overtones in piano audio clip and less overtones in recorder version after examining the audio data of Mary Had a Little Lamb.

Appendix A: MATLAB Functions Used and Brief Implementation Explanation

[y,Fs] = audioread(filename) reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.

[M,N] = max (Y) returns the row indexes of largest elements in each column.

colormap map sets the colormap for the current figure to one of the predefined colormaps.

pcolor(X, Y, Z) specifies the x- and y-coordinates for the vertices. The size of C must match the size of the x-y coordinate grid. For example, if X and Y define an m-by-n grid, then C must be an m-by-n matrix.

plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X.

Y = fft (X) returns the fast Fourier transform on X.

Y = fftshift (X) rearranges a Fourier transform by shifting the zero-frequency component to transform algorithm. the center or the array. [3]

[3] "Makers of MATLAB and Simulink." MathWorks, www.mathworks.com/.

Appendix B: MATLAB Code

```
%% Part I
clear all; close all; clc;
load handel
v=y'/2;

%Setting up the signal
L=9;
v=v(1:(length(v)-1));
n=length(v);
t=(1:length(v))/Fs;
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);

%Creating Gaussian window a=1 delta translation=0.1
a = 1;
tslide=0:0.1:9;
vgt_spec = zeros(length(tslide),n);
vgt_spec = [];
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

figure(1)
pcolor(tslide,ks,vgt_spec.'),
shading interp
xlabel('Time(sec)')
ylabel('Frequency(¶ÿ)')
colormap(hot)
title('a=1 dt=0.1')

%Changing window width to a=50 translation=0.1
a=50;
tslide=0:0.1:9;
vgt_spec = zeros(length(tslide),n);
vgt_spec = [];
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

figure(2)
pcolor(tslide,ks,vgt_spec.'),
shading interp
xlabel('Time(sec)')
ylabel('Frequency(¶ÿ)')
colormap(hot)
title('a=50 dt=0.1')
```

```
%Changing window width to a=0.1 translation=0.1
a=0.1;
tslide=0:0.1:9;
vgt_spec = zeros(length(tslide),n);
vgt_spec = [];
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

figure(3)
pcolor(tslide,ks,vgt_spec.'),
shading interp
xlabel('Time(sec)')
ylabel('Frequency(¶ÿ)')
colormap(hot)
title('a=0.1 dt=0.1')

%Changing sample rate to a=50 translation=0.05
a=50;
tslide=0:0.05:9;
vgt_spec = zeros(length(tslide),n);
vgt_spec = [];
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

figure(4)
pcolor(tslide,ks,vgt_spec.'),
shading interp
xlabel('Time(sec)')
ylabel('Frequency(¶ÿ)')
colormap(hot)
title('a=50 dt=0.05')

%Changing sample rate to a=50 translation=5
a=50;
tslide=0:5:9;
vgt_spec = zeros(length(tslide),n);
vgt_spec = [];
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

figure(5)
pcolor(tslide,ks,vgt_spec.'),
shading interp
```

```
xlabel('Time(sec)')
ylabel('Frequency(¶ÿ)')
colormap(hot)
title('a=50 dt=5')

%Creating Mexican Hat Wavelet window
a=1;
tslide=0:0.1:9;
vgt_spec = zeros(length(tslide),n);
vgt_spec = [];
for j=1:length(tslide)
    g=(2/(sqrt(3*a)*(pi)^(1/4)))*(1-((t-tslide(j))/a).^2).*exp(-(t-
tslide(j)).^2/(2*a^2));
    vg=g.*v;
    vgt=fft(vg);
    vgt_spec(j,:) = fftshift(abs(vgt));
end

figure(6)
pcolor(tslide,ks,vgt_spec.'),
shading interp
xlabel('Time(sec)')
ylabel('Frequency(¶ÿ)')
colormap(hot)
title('Mexican Hat Wavelet')

%Creating Shannon Window
a=1;
tslide=0:0.1:9;
vgt_spec = zeros(length(tslide),n);
vgt_spec = [];
for j=1:length(tslide)
    g=(abs(t-tslide(j))<a);
    vg=g.*v;
    vgt=fft(vg);
    vgt_spec(j,:) = fftshift(abs(vgt));
end

figure(7)
pcolor(tslide,ks,vgt_spec.'),
shading interp
xlabel('Time(sec)')
ylabel('Frequency(¶ÿ)')
colormap(hot)
title('Shannon Window')

%% Part 2 audio 1
clear all; close all; clc;

y= audioread('music1.wav');
Fs=length(y)/16;
y=y'/2;
L=16;
n=length(y);
t=(1:length(y))/Fs;
```

```
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);
a=30;
tslide=0:0.2:16;
ygt_spec = zeros(length(tslide),n);
ygt_spec = [];
y_score=[];
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    yg=g.*y;
    ygt=fft(yg);
    ygt_spec(j,:) = fftshift(abs(ygt));
    [M,N]=max(abs(ygt));
    y_score=[y_score; abs(k(N))/(2*pi)];
end

figure(1)
pcolor(tslide,ks/(2*pi),ygt_spec.'),
set(gca,'Ylim', [200 400],'Fontsize', [12])
shading interp
xlabel('Time(sec)')
ylabel('Frequency(Hz)')
colormap(hot)
title('Mary Had a Little Lamb Piano')

figure(2)
plot(tslide, y_score)
xlabel('Time(sec)')
ylabel('Frequency(Hz)')

%% Part 2 audio 2

clear all; close all; clc;

y= audioread('music2.wav');
Fs=length(y)/14;
y=y'/2;
L=14;
n=length(y);
t=(1:length(y))/Fs;
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);
a=30;
tslide=0:0.2:14;
ygt_spec = zeros(length(tslide),n);
ygt_spec = [];
y_score=[];
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    yg=g.*y;
    ygt=fft(yg);
    ygt_spec(j,:) = fftshift(abs(ygt));
    [M,N]=max(abs(ygt));
    y_score=[y_score; abs(k(N))/(2*pi)];
end
```

```
figure(1)
pcolor(tslide,ks/(2*pi),ygt_spec.'),
set(gca,'Ylim', [800 1100],'Fontsize', [12])
shading interp
xlabel('Time(sec)')
ylabel('Frequency(Hz)')
colormap(hot)

figure(2)
plot(tslide, y_score)
xlabel('Time(sec)')
ylabel('Frequency(Hz)')
title('Mary Had a Little Lamb Recording')
```