Gábor Transform

Eileen Chang

February 7th, 2020

Abstract

Since Fourier transform can only analyze the time or the frequency. It can't analyze the time and frequency at the same time. Thus, Gábor transform plays an important role as analysis both the time and the frequency at the same time. Through generating spectrograms for the Gábor transform, we are able to know the amplitude of frequency at some time. For Gábor transform, the width of the window and the time step are the main studies of this paper. Choosing the wrong width or the time step will give us a bad analysis. Thus, we are going to explore the suitable width and time step for the audio signal.

1 Introduction

Time-frequency analysis gives us a study for both the time domain and the frequency domain. Since being able to know which frequency happens in what time will help us getting more correctly analysis, it is better to use Gábor transform to get both information about the time and the frequency. In this paper, we are going to analyze the three audio, Handel's Messiah and Mary Had a Little Lamb that plays by a piano and a recorder.

Spectrograms of the audio signals are created by implementing the Gábor transform, using different functions as the sliding window. Using spectrograms can let us know more information about the frequency that happens at a specific time. Thus, in this paper, we are going to explore the different coefficients in Gaussian function and the time step in the time domain to see how it will affect the spectrograms. Also, we are going to compare the Gaussian function with the Mexican Hat and Shannon wavelets. We will also talk about undersampling and oversampling as well.

2 Theoretical Background

2.1 Time-Frequency Analysis: Windowed Fourier Transforms (Short Time Fourier Transform, STFT)

Time and frequency are both really important when analysis the data. Fourier transform treat the frequency and time as two individual parts. It means that

when we get the frequency, then we completely lost the information about time. Thus, the windowed Fourier transform is a good technology to get both the information of time and frequency. It will capture the small part of the signal without losing any information of time and frequency.

2.2 Gábor Transform

Gábor transform is the windowed Fourier transform that we applied Gaussian filter into it. The definition of the Gábor transform captures the entire time-frequency content of the signal. It is a function of the two variables t and ω. Thus, the Gábor transform is computed by discretizing the time and frequency domain. This is the Gábor equation

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

where the bar represents the complex conjugate of the function. The Gábor kernel,

$$g_{t,\omega}(\tau) = e^{i\omega\tau}g(\tau - t), \tag{2}$$

was introduced with the aim of localizing both time and frequency. The time filtering window $g(\tau - t)$ is centered at τ with width a. When considering the spectral content of this window, any portion of the signal with a wavelength longer than the window is completely lost. The drawback of the Gábor transform is that when analysis the signal, it will only change the position but the width and shape of window will not change. Since the frequency is the reciprocal of the period, it will be more practical if the width of the window varies. The high frequency is captured better with the narrow window and the low frequency will be more correctly when the width of the window is wide.

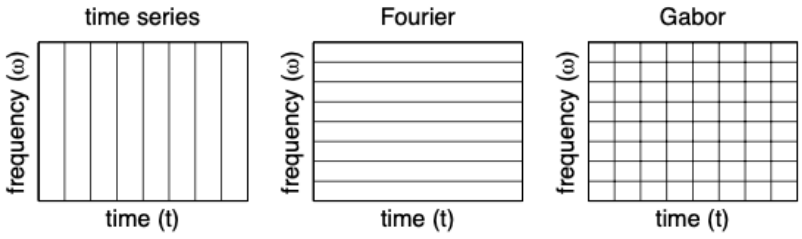


Figure 1: There are trade-offs between these three graphs. The left graph is time series analysis. It only gives us the information about time. The middle graph is the frequency analysis. It only indicates that the information about frequency. The right graph is the Gábor transform. It tells us both information about time and frequency.

2.3 Spectrogram

The spectrogram is a spectrum graph but includes the time information. When applied to an audio signal, spectrograms are sometimes called sonographs, voiceprints, or voicegrams. When the data is represented in a 3D plot they may be called waterfalls. In this project, the spectrogram is constructed using frequency data gathered using the Gábor transform on three audio signals.

3 Algorithm Implementation and Development

1. Load the audios(handel, music1, music2) into the Matlab.
   - Need to transpose the vector because it is a column.
2. Calculate its length and the sampling rate (number of points).
3. Set the frequency components by scaling the time domain with $\frac{2\pi}{2L}$, so the frequency domain will be $[-2\pi, 2\pi]$.
   - When the number of points are even, the grid vector of the frequency domain should be defined as $\frac{2\pi}{2L}$ * [0:$\frac{n}{2}$-1 -$\frac{n}{2}$:-1]
   - When the number of points are odd, the grid vector of the frequency domain should be defined as $\frac{2\pi}{2L}$ * [0:$\frac{n-1}{2}$ -$\frac{n-1}{2}$:-1]
   - In this case, we have odd number of points, so we are going to use the second one.
4. Swap the first half and second half of frequency components by using the commend fftshift, so the function will transform back to its mathematically correct positions.
5. Assign values for the width of the window (Gaussian, Mexican Hat, and Shannon)
   - We will keep changing the value of these variables since the aim is to find the best value for these variables.
6. Define the time step vector for the moving windows.
7. Using for loop for translating the window
   - Define a filter function for Gaussian function, Mexican Hat wavelet, and Shannon wavelet.
   - Multiply the signal with the filter function.
   - Take the Fourier transform for the result.
   - Using the commend fftshift then take the absolute of the Fourier result. Add the result data into the vector Sgtvector, which will be used to generate the spectrograms.

- Plot the signal with for the filtering function and Fourier transform.

8. After finish running the for loop, we will use the vector Sgtvector to plot the spectrogram with the time on the x-axis and frequency on the y-axis.

4 Computational Results

4.1 The Handel's Messiah

This is a 8.924 seconds audio of the Handel's Messiah. It is discretized into 73113 points. And the sampling rate is 8192 Hz. Figure 2 is the graph of the interesting signal of Handel's Messiah that shows in time.

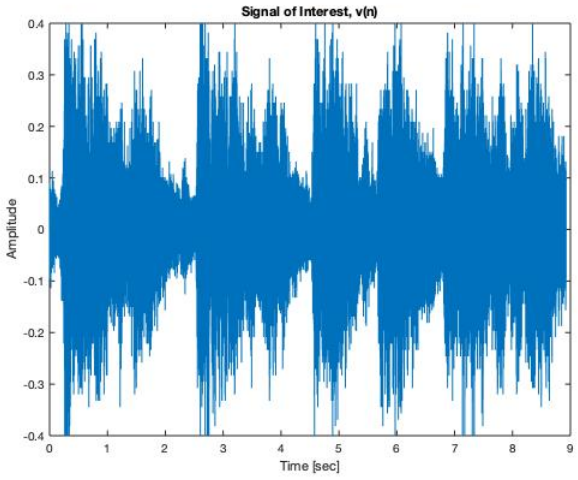


Figure 2: This is the signal of Handel's Messiah.

We apply the Gaussian filter in the Gábor transform to analysis the signal. Figure 3 gives us an audio signal with the moving window on the top. The middle is the graph that showing us the result of Gaussian function and the signal. The bottom is the Fourier transform of the resulting function.

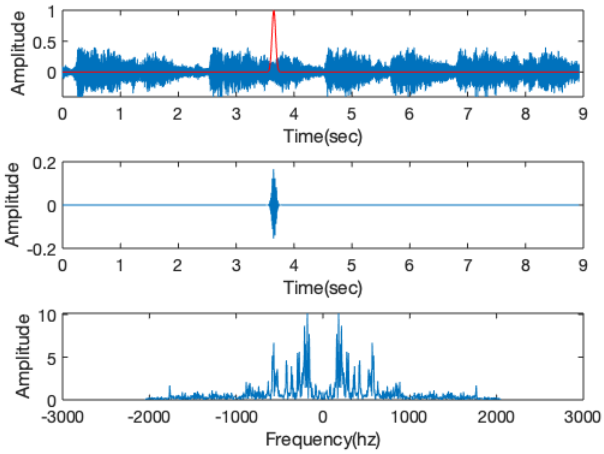


Figure 3: Top: An audio signal with moving window; Middle: The result of Gaussian filter and the signal; Bottom: the Fourier transform of the resulting function.

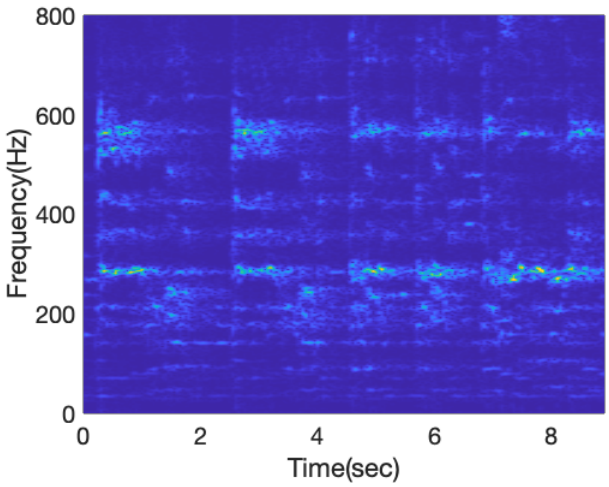
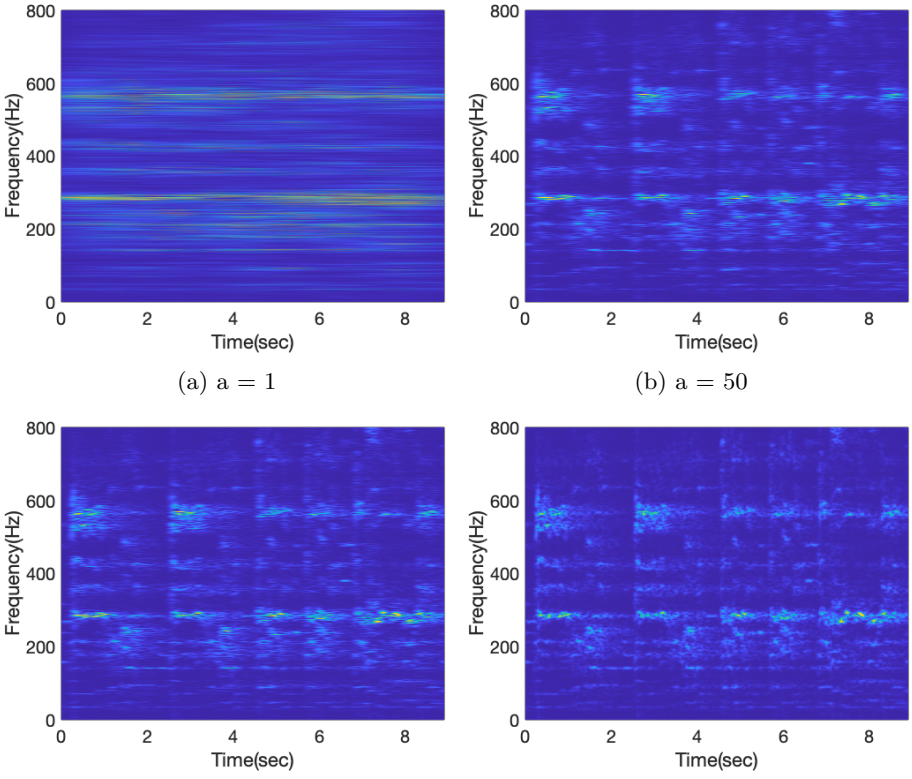


Figure 4: Spectrogram of Handel's Messiah, a = 500, Δt = 0.05

Figure 4 shows us the spectrogram of the Gábor with Gaussian filter. The width of the window is 500 and the time step is 0.05 seconds. This graph captured both the time and the frequency well. In the graph, we can see that in 500-600 Hz and 200-300 Hz, they have high amplitude. They are corresponding to the "Hallelujah" in the audio. The high frequency part is sung by women

and the low frequency part is sung by men.



(a) a = 1

(b) a = 50

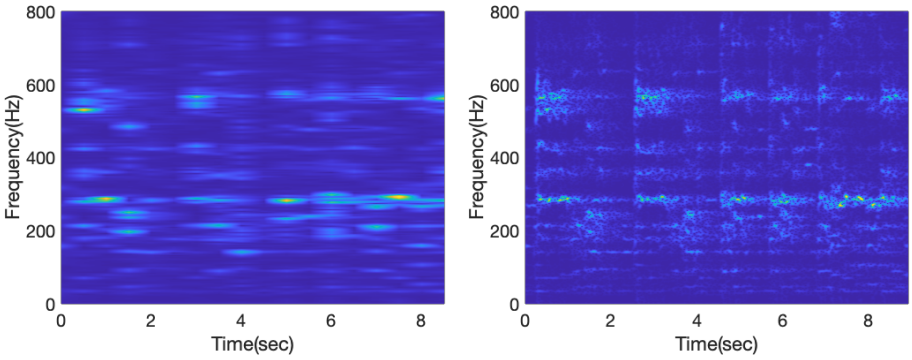
(c) a = 100

(d) a = 500

Figure 5: Spectrogram with different width of Gaussian Windows, $\Delta t = 0.05$

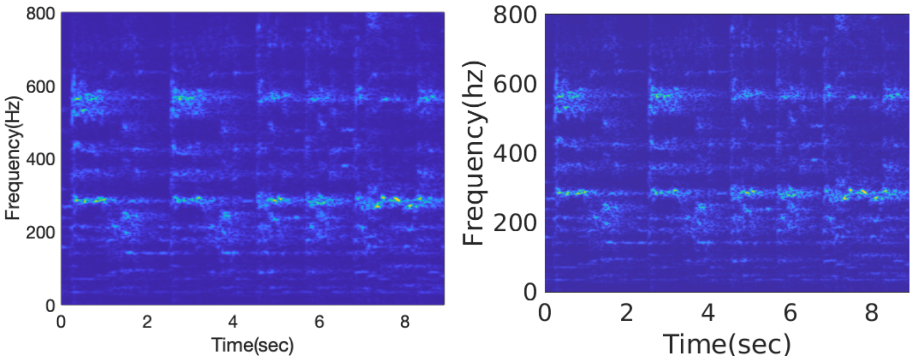
Figure 5 is the spectrograms with different width of Gaussian window with $\Delta t = 0.05$. From the graph, we can know that if the width of the window is narrow, then we can get more information about the time but less information about the frequency. Opposite, when the width of the window is wide, then we will know more information about frequency but less information about the time. When a = 500, it performs well in the both time and frequency domain. As a decrease, we lose more information about the frequency. Thus, when a = 1, it performs very poor at capturing the frequency.

Figure 6 is the results of under and over sampling. For under sampling, the time step is 0.1 second and a is 500. Since the time step is 0.1 second, it misses some samples from the signal. Thus, from the graph, we can know that there are some samples that are missing. The oversampling graph has enough signal to captured both time and frequency well.



(a) Under Sampling (b) Over Sampling

Figure 6: Result of Under and Over Sampling



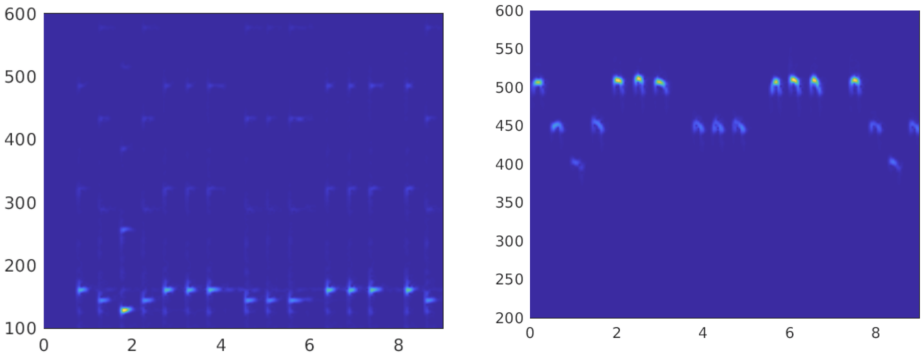
(a) The Mexican Hat (b) Shannon Wavelet

Figure 7: Result of Mexican Hat Wavelet and Shannon Wavelet

From Figure 7, we can see that there is no significant difference between Mexican Hat and Shannon Window. The time step of both wavelets is 0.05 seconds, same as Gaussian transform. So, from the spectrograms of these three wavelets, we can conclude that there is not significant difference between them.

4.2 Mary Had a Little Lamb

There are two kinds of instruments for playing the song - Mary Had a Little Lamb. One is the piano, the other one is recorder.



(a) Piano (b) Recorder

Figure 8: Spectrograms of Mary Had a Little Lamb with two different instruments - Piano and Recorder (x-axis unit: time(sec), y-axis unit: frequency(Hz)

The frequencies of the piano in this recording ranged from 100 Hz to 200 Hz, while the frequencies recorder ranged from 400 Hz to 550 Hz. We know that recorder has the higher frequency, so it hears like having higher pitch than the piano one. Since every node has its own frequency in Hz, we can tell the song is constructed by these nodes:
Mi, Re, Do, Re, Mi, Mi, Mi
Re, Re, Re, Mi, Mi, Mi
Mi, Re, Do, Re, Mi, Mi, Mi
Re, Re, Mi, Re, Do.

5 Summary and Conclusions

Gábor transform allows us to get both the time and the frequency of the signal at the same time. The width of the window and the time step are the core things that we study in this paper. After running the code, we find out that when the width of the window is 500 and the time step is 0.05 second, Matlab generates good spectrograms for this three audio. Through the spectrograms, we can see the amplitude of frequency corresponding to the time. We also explore the undersampling/oversampling of the data and the different wavelets. Thus, from Handel's Messiah, we find out that using the Gaussian, Mexican Hat, and Shannon wavelets as filter will generate the similar spectrograms for the audio. And from the song - Mary Had a Little Lamb, even though the notes are the same, through different instruments, it will generate different amplitudes of frequency so people can hear the difference between the music that plays by different instruments.

6 Appendix A.

1. abs()
   - Return an absolute value.
   - A = abs(a).
2. fftshift()
   - Shift zero-frequency component to center of spectrum
   - Y = fftshift(X) rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.
3. pcolor()
   - pcolor(C) creates a pseudocolor plot using the values in matrix C.
   - pcolor(X,Y,C) specifies the x- and y-coordinates for the vertices. The size of C must match the size of the x-y coordinate grid.
4. plot()
   - 2-D line plot
   - plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X.
5. length()
   - Length of largest array dimension.
   - length(X) returns the length of the largest array dimension in X.
6. audioread()
   - Read audio file.
   - $[y, Fs] = audioread(filename)$ reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.

7 Appendix B.

Matlab code for Handel's Messiah

```
load handel
v = y'/2;
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');
```

```
a = 500;
ShannonWindow = 0.25;
b = 100;

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

tfinal = length(v)/Fs;
t = (1:length(v))/Fs;

Sgtvector = [];
t_step = 0:0.01:5;

figure(2)
for i = 1:length(t_step)
    g = exp(-a*(t-t_step(i)).^2);%Gaussian
    g = (1-b*(t-t_step(i).^2).*exp(-a*(t-t_step(i)).^2);
        %Mexican Hat
    %Shannon window
    if t_step(i)+ShannonWindow > 8.92
        break
    end
    g = 0*[1:length(v)];
    if t_step(i) <2*ShannonWindow
        tend = floor((t_step(i)+ShannonWindow)*8192;
        g(1:tend) = 1;
    elseif tfinal -t_step(i) < 2*ShannonWindow
        tstart = ceil((t_step(i) *8192))+1;
        g(tstart:end) = 1;
    else
        tstart = floor((t_step(i)-ShannonWindow)*8192);
        tend = floor((t_step(i)+ShannonWindow)*8192);
        g(tstart:tend) = 1;
    end
    Sg = g.*v;
    Sgt = fft(Sg);
    Sgtvector = [Sgtvector; abs(fftshift(Sgt))];
%     subplot(3,1,1), plot(t, v, t, g, 'r');
%     axis([0 length(v)/Fs -0.5 1]);
%     xlabel('Time [sec]');
%     ylabel('Amplitude');
%
%     subplot(3,1,2), plot(t, Sg);
%     axis([0 length(v)/Fs -0.5 1]);
%     xlabel('Time [sec]');
```

```
%       ylabel('Amplitude');
%
%       subplot(3,1,3), plot(ks/(2*pi), abs(fftshift(Sgt)))
    ;
%       xlabel('Frequency(hz)');
%       ylabel('Amplitude');
%       set(gca, 'Fontsize', 14);
%
%       pause(0.0000001)
end

figure(3)
pcolor(t_step, ks/(2*pi), Sgtvector.'), shading interp
set(gca, 'Ylim', [0 800], 'Fontsize', (20));
xlabel('Time(sec)');
ylabel('Frequency(hz)');
```

Matlabe code for Mary Had a Little Lamb

```
clear all; clc; close all;
%[y,Fs] = audioread('music1.wav');
[y,Fs] = audioread('music2.wav');
tr_piano=length(y)/Fs;  % record time in seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Mary had a little lamb (piano)');

v = y'/2;
a = 500;

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

tfinal = length(v)/Fs;
t = (1:length(v))/Fs;

Sgtvector = [];
t_step = 0:0.05:tfinal;

figure(2)
for i = 1:length(t_step)
    g = exp(-a*(t-t_step(i)).^2);
    Sg = g.*v;
    Sgt = fft(Sg);
    Sgtvector = [Sgtvector; abs(fftshift(Sgt))];
```

```
    subplot(3,1,1), plot(t, v, t, g, 'r');
    axis([0 length(v)/Fs -0.5 1]);
    xlabel('Time [sec]');
    ylabel('Amplitude');

    subplot(3,1,2), plot(t, Sg);
    axis([0 length(v)/Fs -0.5 1]);
    xlabel('Time [sec]');
    ylabel('Amplitude');

    subplot(3,1,3), plot(ks, abs(fftshift(Sgt)));
    xlabel('Frequency(hz)');
    ylabel('Amplitude');

    pause(0.0000001)
end

figure(3)
pcolor(t_step, ks/(2*pi), Sgtvector.'), shading interp
set(gca, 'Ylim', [100 600], 'Fontsize', (14));
```