Amath482 – Winter 2020

Gábor transforms

Chenxi Di (1663044)

February 7, 2020

Abstract

The Fourier Transform is a fundamental tool for the analysis of signal. It is clear that when transforming a signal over time, the FT effectively captures the frequency content. Unfortunately, the transform will lose the time information. Developed by Gábor Dénes, the Gábor transform, also known as the short-time Fourier transform (STFT) is frequently used to localize a signal in both the time and frequency domains. This study aims to analyse some portions of with two music with time-frequency analysis.

I. Introduction and Overview

To analyze the 9 second piece of music over time and frequency, I used Gábor transform. Specifically, I produced the spectrograms of the piece of work through Gábor transform. From here, I will explore the roles of window width, translations of the Gábor window and different Gábor windows applied in Gábor transform.

For the second part, I also analyzed the frequency and time information of the song 'Mary had a little lamb' and employed the Gábor transform to reproduce the music score produced by the piano and recorder. From here, I will explore the possible difference between two instruments in the time frequency analysis.

II. Theoretical Background

Gábor Transforms

The limitation of Fourier Transform was early realized in the development of radar and sonar detection. To localize a signal in both the time and frequency domains, the new kernel was introduced:

$$g_{t,\omega}(\tau) = e^{-i\omega\tau} g(\tau - t) \tag{1}$$

The time filtering window is centered at τ with a width a. Then the frequency content of one window is extracted and τ is changed to capture frequency of another time window. The Gábor transform, also called as short-time Fourier transform (STFT) is then defined as:

$$\mathcal{G}[f](t,\omega) = \tilde{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau) g(\tau - t) e^{-i\omega\tau} d\tau \tag{2}$$

with $g(\tau - t)$ resulting the localization of Fourier integral around $t = \tau$.

Gábor Windows

Gábor transform allows to analyze the time and frequency features of a given signal. The important part for Gábor transform is to multiply the time filter Gábor function g(t) with its original signal to construct a windowed section of the signal. Three examples of Gábor filtering function are:

$$\text{Gaussian} : g(t) = e^{-a(t-\tau)^2} \tag{3}$$

$$\text{Mexican Hat} : g(t) = (1 - a(t-\tau)^2) e^{-a(t-\tau)^2/2} \tag{4}$$

$$\text{Shannon} : g(t) = 1 \text{ if } |t - \tau| \leq a/2, \text{ otherwise } 0 \tag{5}$$

where a refers to the width of the window, τ refers to the center of that window.
However, the main challenge for Gábor transform is to find a balance between the time and frequency resolution. As the time filtering window becomes longer, much of the information regarding frequency will disappear. In contrast, shorter window allows more frequency content, but this is achieved at the expense of losing the time resolution of the given signal.

III. Algorithm Implementation and Development

In part one, I will conduct a time frequency analysis on the piece of music in Handel's Messiah by using Gábor transform. Since I will use Fast Fourier Transform in the process, I need to first determine the domain L and Fourier modes. Note that the data has a time domain of (1:length(v))/Fs. Therefore L is equal to length(v)/Fs, which is 8.92 ≈ 9. Since there are 73113 data points, I choose to eliminate the last point to make it as an even number such that the domain can be discretized into $73112 = 2^i$ points in order to apply FFT. As the original Fourier Transform assumes 2π as the period, our k has to be rescaled to $2\pi/L$. To capture the time information, I will set several time windows. Specifically, I will use tslide: $tslide = 0 : \tau : L$ where τ represents center of the window, therefore the time window will slide to another center by a length of τ at each loop. From here, I will apply the Gábor filter and use FFT to capture the frequency content in each time window. The functions of the scaling a, translation τ and filter function will be discussed by following different spectrograms.

In part two, I will also analyze the frequency and time information of the song 'Mary had a little lamb'. To use Gábor transform, I need to first set up some parameters. According to the data, the time domain for the piano file is $Lpiano = 16$, the time domain for the recorder file is $Lrecorder = 14$. Furthermore, there are even data points in both files. As the original Fourier Transform assumes 2π as the period, our k has to be rescaled to $2\pi/L$. To capture the time information, I will set several time windows. Similar to the part I, I will build two spectrograms of the same pieces of music by piano and recorder using Gaussian funtion, window width a = 10, $\tau = 0.1$. When plotting the spectrogram, we need to divide the ks by 2π since they have units of Hz.

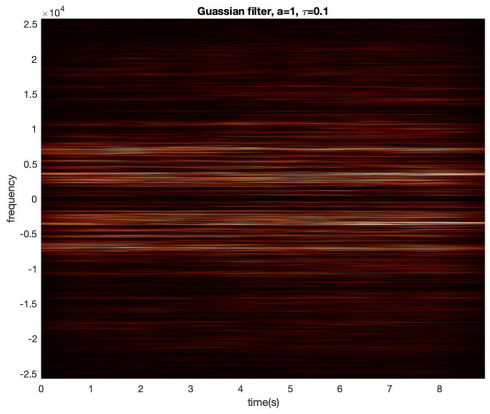
IV. Computational Results

Part I

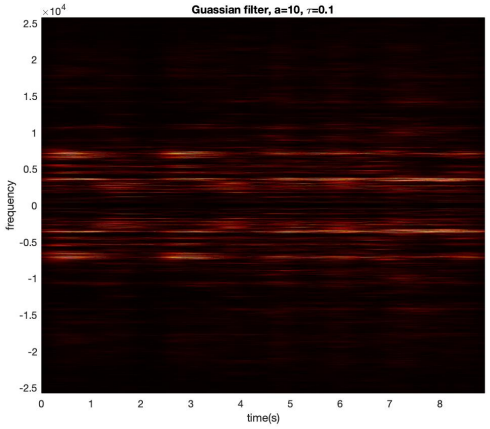
1. Window width of the Gábor transform effects on spectrogram: When $\tau = 0.1$, as the window width a increases from 1 to 10 (Figure 1a to Figure 1b), we can clearly see less horizontal 'red' line. So there are much of the frequency information lost. But in Figure1b, it is easier to identify a precise time at which the signals change. Therefore, the narrower window will allow us to see more frequency content but less time information.

The wider window will give us a better time resolution but less frequency information.

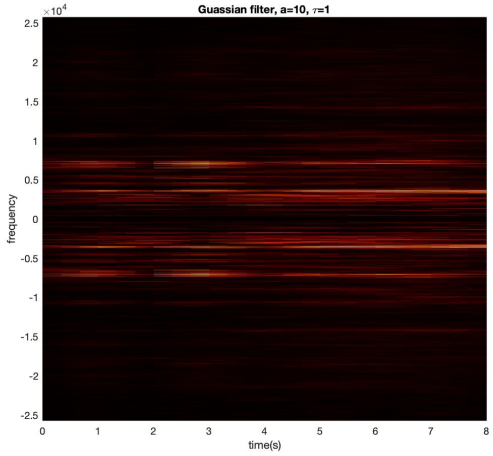
2. Oversampling and undersampling: When $a = 10$, as the τ decreases from 1 to 0.01 (Figure 1c to 1d), it becomes easier to identify the time between frequency changes. So there are more time resolution using smaller τ. Smaller τ means we are sampling more windows, which is oversampling. Therefore, oversampling will give a better time resolution, but this comes at the cost of a longer run time. On the other hand, using larger τ, i.e. undersampling, will give a worse time resolution.



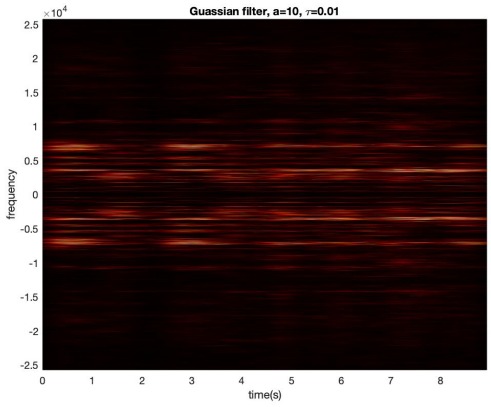
(a) Guassian Filter, a = 1, $\tau = 0.1$



(b) Guassian Filter, a = 10, $\tau = 0.1$



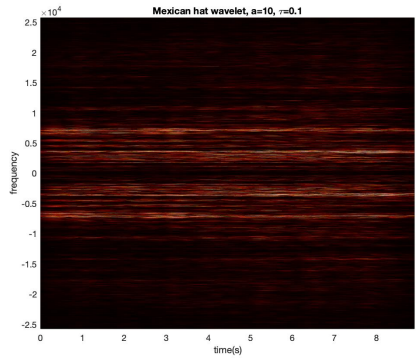
(c) Guassian Filter, a = 10, $\tau = 1$



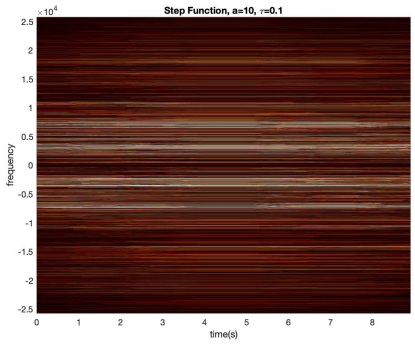
(d) Guassian Filter, a = 10, $\tau = 0.01$

Figure 1: Spectrogram of Handel's Messiah by different window width and translations

3. Different filter functions: When $a = 10, \tau = 0.1$, I examine the performances of different filter functions from Gaussian, Mexican Hat to Step function (Figure 1a, Figure 2a, and Figure 2b respectively). As the spectrogram shows, the Step function gives the best frequency resolution in this example. The Gaussian filter also gives a better frequency resolution compared to the Mexican Hat. However, it seems like the Mexican Hat gives a better time resolution compared with others. Different window functions will have different performance at the spectrogram. But there always will be a trade off between frequency content and time information. We need to carefully balance and choose appropriate functions and parameters to accomplish our goal.



(a) Mexican Hat window function



(b) Step function

Figure 2: Spectrogram of Handel's Messiah by different window function, width a = 10, τ = 0.1

Part II

Figure 3: Piano and Recorder Spectrogram, by Gaussian, width $a = 100$, $\tau = 0.1$

Comparing the spectrogram of music produced by piano with the music scale, I obtain the music score for the piano piece as:

$$E_4, D_4, C_4, D_4, E_4, E_4, E_4, D_4, D_4, D_4, E_4, E_4, E_4, E_4, D_4, C_4, D_4, E_4, E_4, E_4, E_4, D_4, D_4, E_4, D_4, C_4$$

Comparing the spectrogram of music produced by recorder with the music scale, I obtain the music score for the recorder piece as:

$$C_6, B_6, A_6, B_6, C_6, C_6, C_6, B_6, B_6, B_6, C_6, C_6, C_6, C_6, B_6, A_6, B_6, C_6, C_6, C_6, C_6, B_6, B_6, C_6, B_6, A_6$$

According to the definition of overtone, since there are more shadows spread from the lightest point in piano spectrogram (i.e. there are overtones at $2\omega_0, 3\omega_0, ..$) compared to the recorder spectrogram, the piano seems to produce more overtone compared to the recorder. The frequency of recorder is relatively higher than the frequency of the piano.

V. Summary and Conclusions

Over the course of part 1, I use Gábor transform to locate both frequency and time content for a piece of music and see there is always a tradeoff between frequency resolution and time resolution on the spectrogram. Specifically, narrower window width will provide a better frequency resolution but worse time resolution. Oversampling will probably give good time resolution but is inefficient to run. Different window filter functions will perform differently according to the property of the given signal. Overall, we need to thoughtfully balance the tradeoff and choose proper parameters and filter functions to meet our expectation. From the part II, I reproduce the music score from different instruments. From their spectrograms, the piano seems to have higher frequency and more overtones compared to the recorder.

Appendix A MATLAB functions used and brief implementation explanation

`fft:fft(Sg)` fast Fourier transform on Sg

`pcolor: pcolor(tslide, ks, Sgtspec)` creates a pseudocolor plot of Sgtspec with tslide on x-axis, ks on y-axis

`fftshift: fftshift(Sg)` rearranges a Fourier transform Sg by shifting the zero-frequency component to the center

`shading interp` varies the color in each line segment by interpolating the true color value across the line.

Appendix B MATLAB codes

```
%% part1
load handel
v = y';
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');

%p8 = audioplayer(v,Fs);
%playblocking(p8);

v = v(1:length(v)-1);
n = length(v);
L = length(v)/Fs;

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

% Calculate Gabor transform and plot spectrogram
a = 10;
tau = 0.1;
```

```
tslide=0:tau:L;
Sgt_spec = zeros(length(tslide),n);
Sgt_spec = [];
t = (1:length(v))/Fs;
for j=1:length(tslide)
    gau = exp(-a*(t-tslide(j)).^2);
    mex = (1-a*(t-tslide(j)).^2).*exp(-a*(t-tslide(j)).^2/2);
    step = abs(t-tslide(j)) <= a / 2;
    Sg=mex.*v;
    Sgt=fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end
pcolor(tslide,ks,Sgt_spec.'),
shading interp
xlabel('time(s)')
ylabel('frequency')
title('Step Function, a=10, \tau=0.1')
colormap(hot)

%% part 2 piano
[y,Fs] = audioread('music1.wav');
piano = y';  % transpose
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)');
% p8 = audioplayer(y,Fs); playblocking(p8);
L_piano = length(y)/Fs;
n = length(y);
k=(2*pi/(L_piano))*[0:(n/2-1) -n/2:-1];
ks=fftshift(k);
% Calculate Gabor transform and plot spectrogram
a = 100;
tau = 0.1;
tslide=0:tau:L_piano;
```

```
Sgt_spec = zeros(length(tslide),n);
Sgt_spec = [];
t = (1:length(piano))/Fs;
for j=1:length(tslide)
    gau = exp(-a*(t-tslide(j)).^2);
    Sg=gau.*piano;
    Sgt=fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end
pcolor(tslide,ks/(2*pi),Sgt_spec.'),
shading interp
xlabel('time(s)')
ylabel('frequency(hz)')
title('Piano Spectrogram, Gaussian, a=100, \tau=0.1')
set(gca,'Ylim',[200 400])
colormap(hot)


%% part 2 recorder
[y,Fs] = audioread('music2.wav');
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)');
% p8 = audioplayer(y,Fs); playblocking(p8);
L_recorder = length(y)/Fs;
recorder = y';
n = length(y);
k=(2*pi/(L_recorder))*[0:(n/2-1) -n/2:-1];
ks=fftshift(k);
% Calculate Gabor transform and plot spectrogram
a = 100;
tau = 0.1;
tslide=0:tau:L_recorder;
Sgt_spec = zeros(length(tslide),n);
Sgt_spec = [];
```

```
t = (1:length(recorder))/Fs;
for j=1:length(tslide)
    gau = exp(-a*(t-tslide(j)).^2);
    Sg=gau.*recorder;
    Sgt=fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end
pcolor(tslide,ks/(2*pi),Sgt_spec.'),
shading interp
xlabel('time(s)')
ylabel('frequency(hz)')
title('Recorder Spectrogram, Gaussian, a=100, \tau=0.1')
set(gca,'Ylim',[600 1200])
colormap(hot)
```

Reference

MathWorks: https://www.mathworks.com/help/matlab/ref/

J. Nathan Kutz. (2013) Data-Driven Modeling and Scientific Computation: Methods for Complex Systems and Big Data. Oxford University Press. ISBN:978-0-19-966034-6