Homework 2 Report

Hooley Cheng
Department of Mathematics
University of Washington
Seattle, WA, 98105, U.S.

February 7, 2020

Abstract

This project intends to investigate various properties of Gabor Filters by serveral differernt examples. We will discuss how the bandwidth affects on the result of Fast-Fourier Transformation of these examples and their spectrograms. Also, we will apply the knowledge into a specific project.

1 Introduction and Overview

We are going to use a portion of Handel's Messiah as the first example to apply different Gabor Filters: Gaussian Window, Mexican hat wavelet and Shannon Window. During our discussion, we found out a interesting phenomenon that relates our time domain and frequency domain. Later on, after comparing several results, we finally found out that this phenomenon is due to Heisenberg Uncertainty. After that, we take a look into one application example of using the technique to help us reproduce music score.

2 Theoretical Background

As discussed before, Gaussian Filter is a powerful tool to help us analysis signals when we know the target frequency. However, in some situation, we might want to extract both time and frequency information from our data.

Then, directly applying Gaussian Filter and Fourier Transform might not be a good choice. Therefore, we need to establish a new approach that help us to deal with this problem, and we found out that a simple modification of the original filter could help:

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

where the new term to the Fourier kernel $g(\tau - t)$ was introduced with the aim of localizing both time and frequency. The Gabor transform, also known as the short-time Fourier transform, is t hen defined as the following:

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

Thus, the function $g(\tau - t)$ acts as a time filter for localizing the signal over a specific window of time, and we can sliding this window alone the time domain. Also, we found other kernels can also function in a very similar behaviour:

$$M(t) = (1 - t^2)e^{-t^2/2}$$

which is known as Mexican hat wavelet, and:

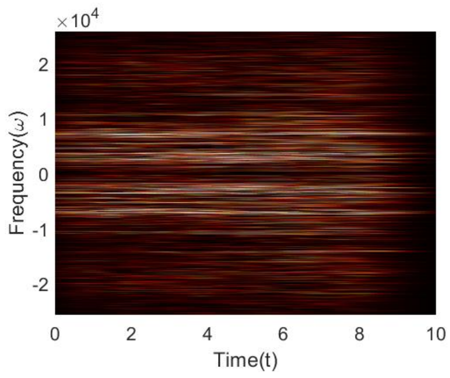
$$S(t) = \begin{cases} 1 & |t - t0| \leq bandwidth \\ 0 & otherwise \end{cases}$$

which is known as Shannon window.

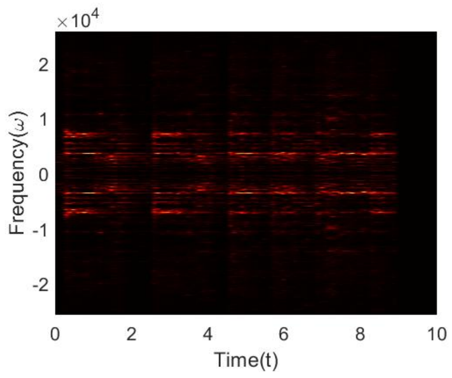
3 Algorithm Implementation and Development

3.1 Relation between bandwidth and spectrogram

First, we are going to investigate how the bandwidth affects spectrogram using Gaussian window:

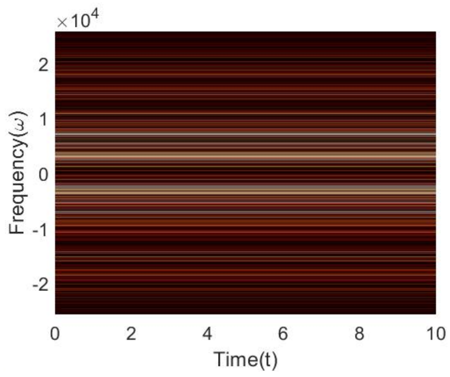


$a = 1$

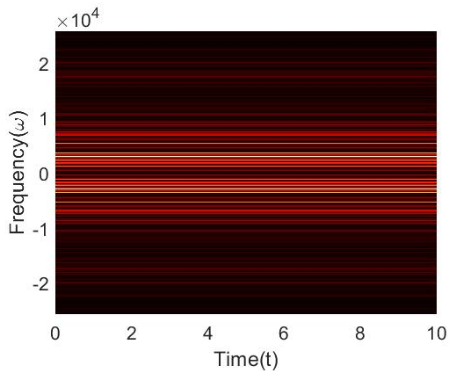


$a = 1000$

Then, we choose another kernel, the Mexican hat wavelet, to compare:

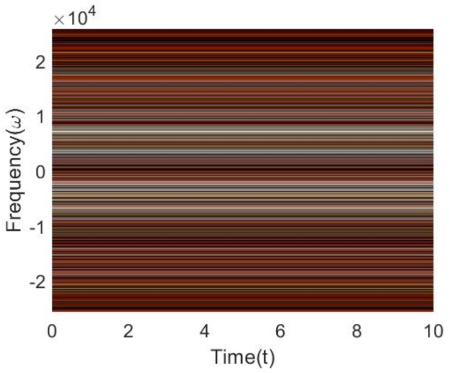


$a = 1$

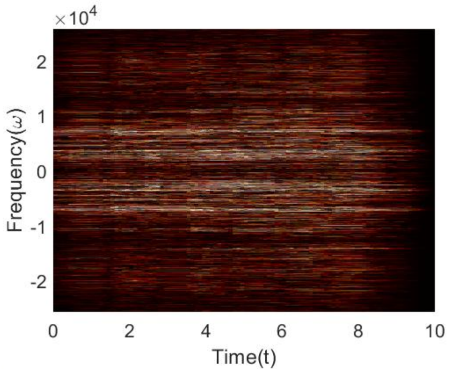


$a = 1000$

and also, we need to try Shannon widow:



$1/a \propto 1$

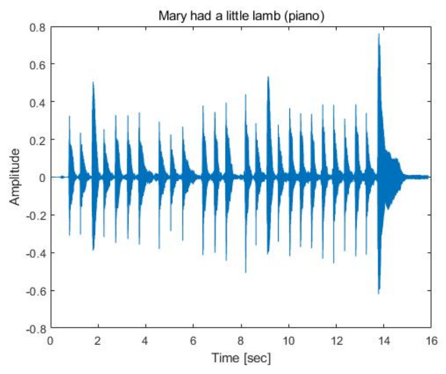


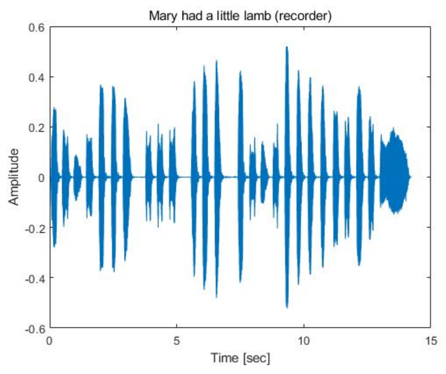
$1/a \propto 20$

We can clearly see the trend from these three examples that as the bandwidth decreasing, the time resolution increases and the frequency resouliton decreases. However, the Mexican hat wavelet seems to have a special property that even though we have narrowed down its band into a very small number, its time resolution is still worse than any other examples.

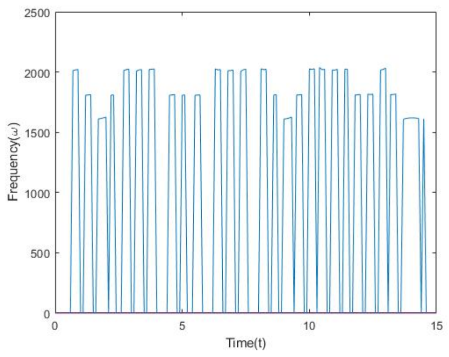
3.2 Application of Gabor filtering

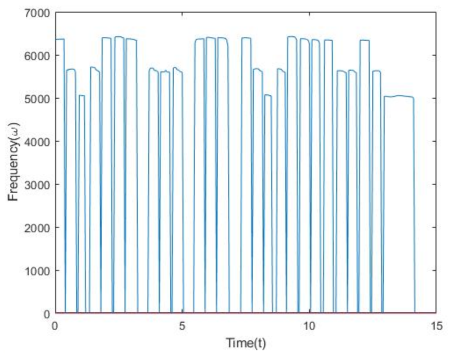
In this section, we going to use Gabor filtering to reproduce the music score of two simple pieces. First, let's take a look at the visualization of the these two pieces:



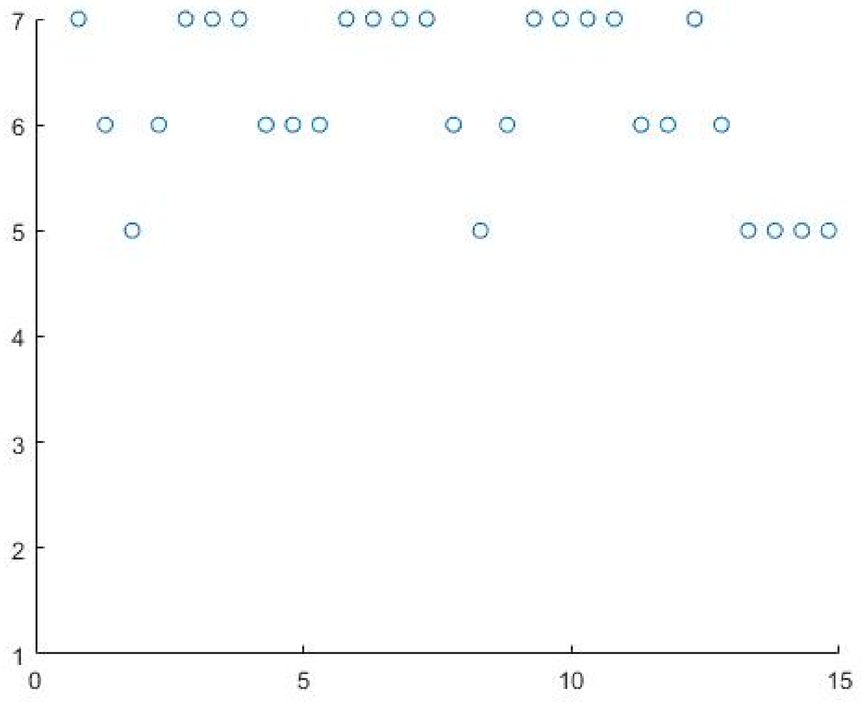


From these two graphs, we can see that there are clear gaps between each node. Therefore, if we use proper bandwidth, we can easily seperate them into single node. By setting $a = 100$, we can easily extract frequency-time graph of these two pieces:





After manually checking the frequency of each node at each time, we can then create a music score graph:



4 Computational Results

Here is the time-frequency table of the pieces:

time(sec)	piano(Hz)	recorder(Hz)	time(sec)	piano(Hz)	recorder(Hz)
0	2015	6362	7.5	1614	5064
0.5	1809	5668	8	1811	5620
1	1616	5054	8.5	2022	6422
1.5	1809	5663	9	2021	6399
2	2017	6397	9.5	2019	6349
2.5	2015	6421	10	2021	6344
3	2023	6369	10.5	1812	5623
3.5	1810	5636	11	1812	5629
4	1808	5599	11.5	2019	6339
4.5	1809	5614	12	1811	5630
5	2019	6368	12.5	1611	5033
5.5	2015	6395	13	1620	5034
6	2021	6392	13.5	1613	5046
6.5	2019	6396	14	1619	5023
7	1810	5676			

5 Summary and Conclusions

Due to the Heisenberg Uncertainty, It is impossible to get accurate information on both time domain and frequency domain, and that is the reason behinds the phenomenon that we found out during 3.1 section. As getting more accurate in time, we increased our time resolution while decreasing the frequency resolution.

6 Appendix A

Y = fftn(X) returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm. The N-D transform is equivalent to computing the 1-D transform along each dimension of X. The output Y is the same size as X.

X = ifftn(Y) returns the multidimensional discrete inverse Fourier transform of an N-D array using a fast Fourier transform algorithm. The N-D inverse transform is equivalent to computing the 1-D inverse transform along each dimension of Y. The output X is the same size as Y.

Y = fftshift(X) rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array. If X is a vector, then fftshift swaps the left and right halves of X. If X is a matrix, then fftshift swaps the first quadrant of X with the third, and the second quadrant with the fourth. If X is a multidimensional array, then fftshift swaps half-spaces of X along each dimension.

X = ifftshift(Y) rearranges a zero-frequency-shifted Fourier transform Y back to the original transform output. In other words, ifftshift undoes the result of fftshift. If Y is a vector, then ifftshift swaps the left and right halves of Y. If Y is a matrix, then ifftshift swaps the first quadrant of Y with the third, and the second quadrant with the fourth. If Y is a multidimensional array, then ifftshift swaps half-spaces of Y along each dimension.

7 Appendix B: Matlab Source Code

Listing 1: Generating Spectrogram of Gaussian Kernel

```
load handel
S = y';
n = length(S);
t2 = (1:length(S))/Fs;
t = t2(1:n);
L = t(end);
k=(2*pi/L)*[0:n/2 -n/2:-1];
ks=fftshift(k);
tau = 3;
a = 1000;
figure(2)
```

```
tslide=0:0.1:10;
Sgt_spec = zeros(length(tslide),n);
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    Sg=g.*S;
    Sgt=fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end
pcolor(tslide,ks,Sgt_spec.'),
shading interp
set(gca,'Fontsize',16)
colormap(hot)
xlabel('Time(t)')
ylabel('Frequency(\omega)')
```

Listing 2: Generating Spectrogram of Mexican Hat Wavelet

```
a=10;
figure(2)
tslide=0:0.1:10;
Sgt_spec = zeros(length(tslide),n);
for j=1:length(tslide)
    mex = (1-a.*(t-t0).^2).*exp(-a.*(t-t0).^2);
    Sg=mex.*S;
    Sgt=fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end
pcolor(tslide,ks,Sgt_spec.'),
shading interp
set(gca,'Fontsize',16)
colormap(hot)
xlabel('Time(t)')
ylabel('Frequency(\omega)')
```

Listing 3: Generating Spectrogram of Shannon Window

```
a=1;
figure(2)
tslide=0:0.1:10;
```

```
Sgt_spec = zeros(length(tslide),n);
for j=1:length(tslide)
    t0 = tslide(j);
    y = abs(t-tslide(j))<=a;
    Sg=y.*S;
    Sgt=fft(Sg);
    Sgt_spec(j,:) = fftshift(abs(Sgt));
end
pcolor(tslide,ks,Sgt_spec.'),
shading interp
set(gca,'Fontsize',16)
colormap(hot)
xlabel('Time(t)')
ylabel('Frequency(\omega)')
```

Listing 4: Generating Frequency-Time graph of music piece

```
figure(9)
[y,Fs] = audioread('music2.wav');
tr_rec=length(y)/Fs; % record time in seconds
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)');

a = 100;
tslide=0:0.05:15;
S = y';
n = length(S);
t2 = (1:length(S))/Fs;
t = t2(1:n);
L = t(end);
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);

score = zeros(length(tslide));
for j=1:length(tslide)
    t0 = tslide(j);
    g=exp(-a*(t-tslide(j)).^2);
```

```
      Sg=g.*S;
      Sgt=fft(Sg);
      y = Sg(find(Sg==max(Sg)));
      if y >=0.05
            x = k(find(Sgt==max(Sgt)));
            score(j) = abs(x);
      end

end
figure(9)
plot(tslide, score);
xlabel('Time(t)')
ylabel('Frequency(\omega)')
```