Gabor transforms and music analyze

Chenghao Chen
Amath 482

1. Abstract

In homework 2, we want to analyze a portion of Handel's Messiah in time and frequency domain. In order to solve the problem, we are required to use Gabor transform to filter the music and show the spectrogram of the work. Then, we can find the relationship between the window width and the spectrogram. In part two, we use Gabor filter to reproduce the music score for two music pieces and compare the difference between a recorder and piano.

2. Introduction

In the problem, we have two part. The first part is to analyze a portion of Handel's messiah using Gabor filtering. By changing different window width and translation, we are able to determine the influence of these parameters on the spectrogram. Finally, we want to switch different Gabor window, such as Mexican hat and step function, to see how the spectrograms look like. The second part is to analyze the difference between a recorder and piano version of the same music piece. We reproduce the music scores using Gabor filtering and compare the difference.

3. Theoretical Background

Gabor Transform

the definition of the Gabor transform, or STFT, is modified to

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

where the bar denotes the complex conjugate of the function. Thus the function $g(\tau - t)$ acts as a time filter for localizing the signal over a specific window of time. The integration over the parameter τ slides the time-filtering window down the entire signal in order to pick out the frequency information at each instant of time. The definition of the Gabor transform captures the entire time-frequency content of the signal. Indeed, the Gabor transform is a function of the two variables t and ω. Note that if $0 < t0$, $\omega 0 < 1$, then the signal is over-sampled and time frames exist which yield excellent localization of the signal in both time and frequency. If $\omega 0, t0 > 1$, the signal is under-sampled and the Gabor lattice is incapable of reproducing the signal. [1]

Benefit: Gabor Transform is able to capture the *moment in time* and analyze non-stationary signal occurred. We can use it to localize both time and frequency properties in a given signal.

Drawback: the method is limited by the time filtering itself. Since the Heinsenberg relationship must hold, the shorter the time filtering window, the less information there is concerning the frequency content. In contrast, longer windows retain more frequency components, but this comes at the expense of losing the time resolution of the signal.

Wavelets

Wavelet analysis begins with the consideration of a function known as the mother wavelet:

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

where a≠ 0 and b are real constants. The parameter a is the scaling parameter illustrated in Fig. 115 whereas the parameter b now denotes the translation parameter. Unlike Fourier analysis, and very much like Gabor transforms, there are a vast variety of mother wavelets that can be constructed. In principle, the mother wavelet is designed to have certain properties that are somehow beneficial for a given problem. Thus, depending upon the application, different mother wavelets may be selected. Three windows are selected in this homework. [1]
The Gaussian window

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

The Mexican hat wavelet

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

The Shannon window

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

4. Algorithm Implementation

In the first part, we need to compare different window width and translation in the code. Therefore, I decide to create two functions to reduce the redundancy.

The first function is to find frequency and time. First, I load the music and import data "v" and "Fs". For Fourier mode, I need to remove the last point in v. Then, I define time "t". The wavenumber "k" is rescaled by (2pi/L) and set the frequency from 0 to (n/2 -1) and (-n/2 to -1) since FFT shifts the domain, Then, I use fftshift to get "ks". In the function, I set window width as "a" and translation as "dt". I make ts "0:dt:9" to represent the translation of time. Then, I use a for loop to translate the Gaussian window. I use "g" to represent gaussian filter and multiply v with the filter "g". Then, I get the Fourier transform of the filtered music and fftshift the absolute value, which help us shift back the data. At last, music3 is the data I want to plot the spectrogram.

The second function is to plot the spectrogram, which takes ts and music3 as variables. I use pcolor and colormap(hot) to plot the spectrogram.

With these two functions, I am able to set different window width and different translation time every time and plot the spectrogram. First, I decide to compare different window width with fixed translation time dt=0.1. I set window width as 0.1, 10 and 50 and plot the spectrogram. Then, I decide to compare different translation time with fixed window width a = 10. I set translation time 0.1, 0.5 and 1. Finally, I change the Gaussian window to Mexican hat and step function, which I just replace the filter to desired equation. I set window width a = 0.1 and dt = 0.1 and plot the spectrograms.

In the second part, instead of load handel, I load music1 and music2 in the code. Just like what I did in the first part, I construct the model and set a = 10 and ts = 0:0.1: 16 or 14. Then, I still use a for loop to filter the data and plot the spectrogram of the two music signals. However, I use max to find the maximum index of the data. In the second part, the frequency needs to be divided by 2pi because we want to see the score in Hertz

5. Computational Results

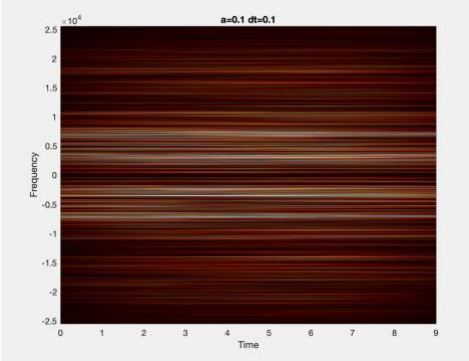


Figure (1) : spectrogram with large window width.

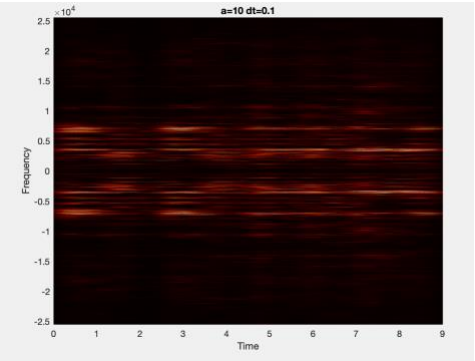


Figure (2) : spectrogram with medium window width

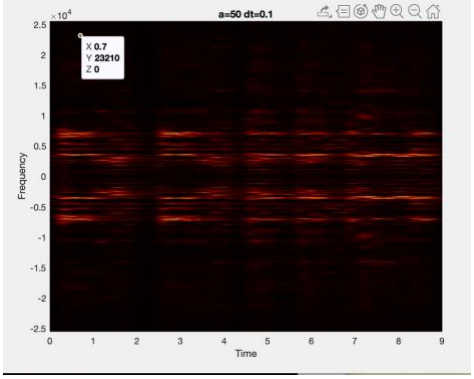


Figure (3) : spectrogram with small window width

As we can see from the spectrogram, the larger the window width is, the more accurate information about frequency domain. The small window width will help us locate the signal in time domain.

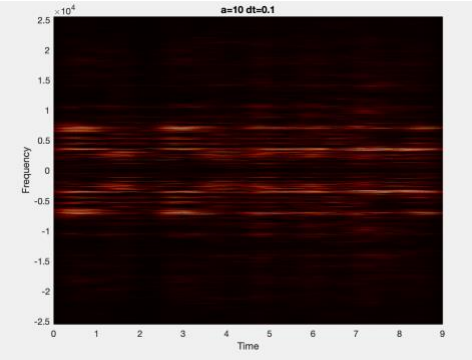


Figure (4) : spectrogram with large sampling

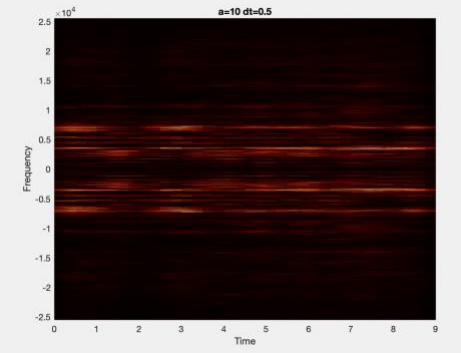
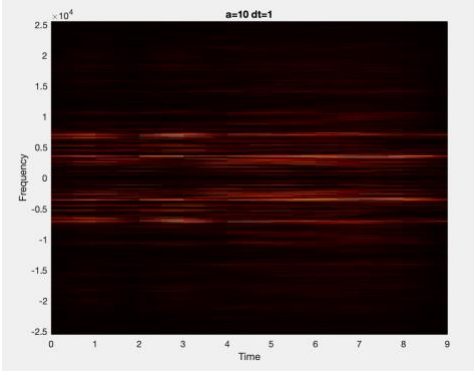


figure (5) : spectrogram with medium sampling



Figure(6) : spectrogram with small sampling

As we can see from the spectrograms, if we have a small sampling, which means that we don't have enough data, the resolution is not good in the plot. If we have a large sampling, which means that we might have duplicate data, the spectrogram might be same.

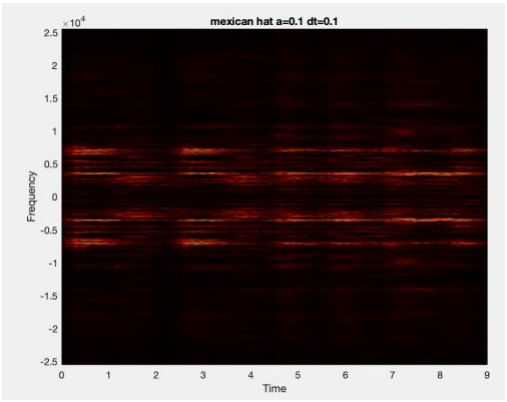


Figure (7) : spectrogram with Mexican hat

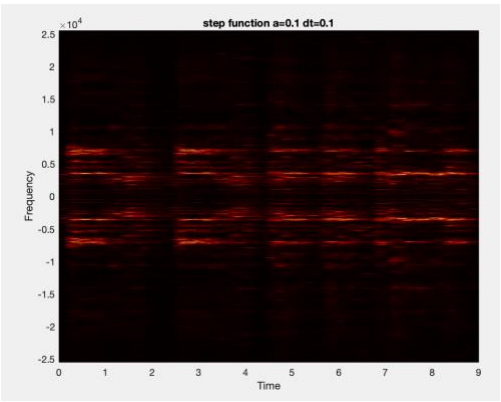


figure (8) : spectrogram with step function

As we can see from the spectrograms, step function has the worst resolution both in time and frequency domain. Gaussian has the best resolution among three spectrograms.

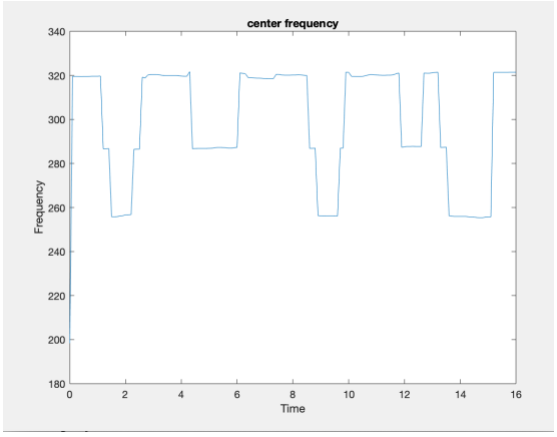
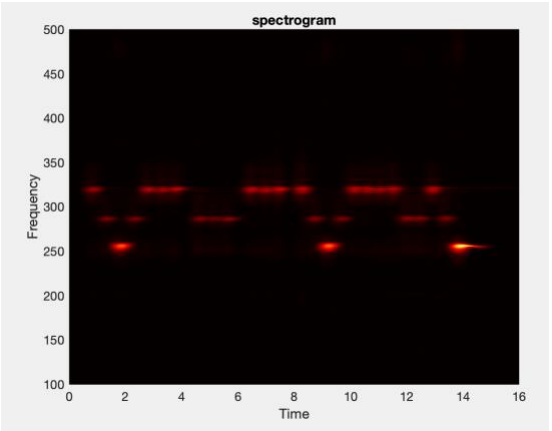


Figure (9) : Music score for piano



figure(10) : spectrogram of the piano

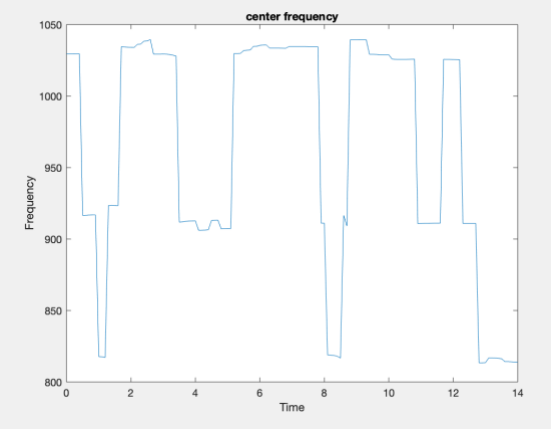


Figure (11) : music score for recorder

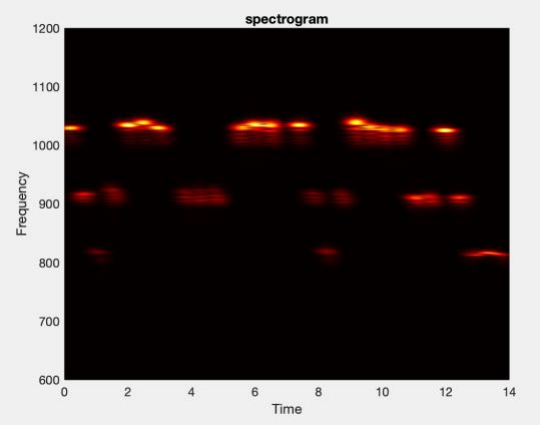


figure (12) : spectrogram of the recorder

As we can see from figure 9 and figure 11, the music scale for piano 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 music scale for recorder 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

Therefore, we can conclude that piano version and recorder version all follow the same pattern except that the recorder frequency is higher than piano version. Also, piano has more overtones and recorder has more undertones.

6. Conclusion

In this homework, I learn how to use Gabor filtering to show the spectrogram of the music signal and find the relationship between window width and the signal. Also, the translation time is another factor that effect the signal. Moreover, we discover different types of wavelets that can affect the signal as well, such as Mexican hat and step function. In part two, I understand how to tell the music scale and difference between piano version and recorder version.

7. Citation

[1] J. N. Kutz, Data-driven modeling & scientific computation: methods for complex systems & big data. Oxford: Oxford University Press, 2013.

[2] “Makers of MATLAB and Simulink,” MathWorks. [Online]. Available: https://www.mathworks.com/. [Accessed: 25-Jan-2020].

8. Appendix A

fftn(N) : This MATLAB function returns the multidimensional Fourier transform of an N-D array using a fast Fourier transform algorithm. [2]

fftshift(N) : This MATLAB function rearranges a Fourier transform X by shifting the zerofrequency component to the center of the array. [2]

max(N) : This MATLAB function returns the maximum elements of an array. [2]

[y,Fs] = audioread(filename) reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs. [2]

colormap map sets the colormap for the current figure to one of the predefined colormaps. [2]

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. [2]

Appendix B

```
clear all; close all; clc;
%part 1
load handel
v = y'/2; L = 9;
v = v(1:(length(v) - 1)); n = length(v); t = (1:n)/Fs;
k = (2*pi/L)*[0:(n/2-1) -n/2:-1]; ks = fftshift(k);

%Gussian window a = 0.1 translation  0.1
[music3,ts] = gussian(0.1,0.1);
      figure(1)
      plotfigure(ts,music3);
      title ('a=0.1 dt=0.1')

      %Gussian window a = 10 translation  0.1
      [music3,ts] = gussian(10,0.1);
      figure(2)
      plotfigure(ts,music3);
      title('a=10 dt=0.1')

      %Gussian window a = 50 translation  0.1
      [music3,ts] = gussian(50,0.1);
      figure(3)
      plotfigure(ts,music3);
      title('a=50 dt=0.1')

      %Gussian window a = 10 translation 0.5
      [music3,ts] = gussian(10,0.5);
      figure(4)
      plotfigure(ts,music3);
      title('a=10 dt=0.5')

      %Gussian window a = 10 translation  1
      [music3,ts] = gussian(10,1);
      figure(5)
      plotfigure(ts,music3);
      title('a=10 dt=1')

      %Mexican hat a = 0.1, dt = 0.1
      a=0.1;
      ts=0:0.1:9;
      music3 = [];
      for j=1:length(ts)
          g=(2/(sqrt(3*a)*(pi)^(1/4)))*(1-((t-ts(j))/a).^2).*exp(-(t-
  ts(j)).^2/(2*a^2));
          music1=g.*v;
          music2=fft(music1);
          music3(j,:) = abs(fftshift(music2));
      end
      figure(6)
      plotfigure(ts,music3);
      title('mexican hat a=0.1 dt=0.1')

      %step function a = 0.1, dt = 0.1
      a=0.1;
```

```
    ts=0:0.1:9;
    music3 = [];
    for j=1:length(ts)
        g=(abs(t-ts(j))<a);
        music1=g.*v;
        music2=fft(music1);
        music3(j,:) = abs(fftshift(music2));
    end
    figure(7)
    plotfigure(ts,music3);
    title('step function a=0.1 dt=0.1') clear all; close all; clc;


    %part 2
    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 = 10;
    ts = 0:0.1:16;
    music3 = [];
    ff = [];
    for j = 1 : length(ts)
        g = exp(-a*(t-ts(j)).^2);
        music1 = g.*y;
        music2 = fft(music1);
        music3(j,:) = abs(fftshift(music2));
        [M, I] = max(abs(music2));
        ff = [ff; abs(k(I))/(2*pi)];
    end
    figure(1)
    plot(ts, ff)
    xlabel('Time')
    ylabel('Frequency')
    title('center frequency')

    figure(2)
    pcolor(ts, ks/(2*pi) ,music3.'), shading interp
    set(gca,'Ylim', [100 500],'Fontsize', [12])
    xlabel('Time')
    ylabel('Frequency')
    colormap(hot)
    title('spectrogram')

    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 = 10;
    ts = 0:0.1:14;
    music3 = [];
    ff = [];
```

```
    for j = 1 : length(ts)
        g = exp(-a*(t-ts(j)).^2);
        music1 = g.*y;
        music2 = fft(music1);
        music3(j,:) = abs(fftshift(music2));
        [M, I] = max(abs(music2));
        ff = [ff; abs(k(I))/(2*pi)];
    end
    figure(3)
    plot(ts, ff)
    xlabel('Time')
    ylabel('Frequency')
    title('center frequency')

    figure(4)
    pcolor(ts,ks/(2*pi),music3.'), shading interp
    set(gca,'Ylim', [600 1200],'Fontsize', [12])
    xlabel('Time')
    ylabel('Frequency')
    colormap(hot)
    title('spectrogram')


    function [music3,ts] = gussian(a,dt)
        load handel
        v = y'/2; L = 9;
        v = v(1:(length(v) - 1));
        n = length(v); t = (1:n)/Fs;
        k = (2*pi/L)*[0:(n/2-1) -n/2:-1]; ks = fftshift(k);
        music3 = [];
        ts = 0:dt:9;
        for j = 1:length(ts)
        g = exp(-a*(t-ts(j)).^2);
        music1 = v.*g;
        music2 = fft(music1);
        music3(j,:) = abs(fftshift(music2));
        end
    end

    function plotfigure(ts, music3)
        load handel
        v = y'/2; L = 9;
        v = v(1:(length(v) - 1));
        n = length(v); t = (1:n)/Fs;
        k = (2*pi/L)*[0:(n/2-1) -n/2:-1]; ks = fftshift(k);
        pcolor(ts,ks,music3.'), shading interp
        xlabel('Time')
        ylabel('Frequency')
        colormap(hot)
    end


```