AMATH 482 Project 2 : Gabor Transform

Yixuan Liu

February 2020

1 Abstract

The power of the Gabor transform is that it can give us information about both the time and frequency domain of the data. By using the Gabor transform, we can visualize data in the spectrogram. In this project, we use three different windows for Gabor transform to visualize *Handel's Messiah* in the spectrogram and adjust some parameters to see some changes in the spectrogram. In the end, we also use Gabor transform to reproduce the music score of *Mary had a little lamb* and distinguish the difference of music produced by the piano and by the recorder.

2 Introduction and Overview

In this project, we have two parts of analyzing data:

Part 1: First, we are given a short chunk of music and try to produce the spectrograms of this piece of work. After that, we change the width of the Gabor window to see the change of resolution in the spectrogram. Moreover, we change the translations of the Gabor window to see the result of oversampling and undersampling. Finally, we explore three types of Gabor windows to differentiate some characteristics of each window (Gaussian window, Mexican hat wavelet, and step-function).

Part 2: We look at the music produced by piano and recorder, doing Gabor transform with the Gaussian window on these data to create a spectrogram. Then, we find the maximum frequency in each time-step to filter out overtones and reproduce the music scores. In the end, we compare the data of piano and recorder to see some similarities.

3 Theoretical Background

3.1 Gabor Transform

Basic idea of Gabor transform is that taking a signal and breaking it down into different frequencies with corresponding time,which is an improvement of Fourier transforms. Here is

the Gabor transform defined as:

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

Where the bar denotes the complex conjugate of the function. For convenience we will consider g be real and symmetric with $\|g(t)\| = \|g(\tau - t)\| = 1$. Thus, we have:

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

Where $g(\tau - t)$ inducing localization of the Fourier integral around t = τ. In other words, we can do Fourier transform around τ and obtain it's corresponding frequency. If we increase the value of τ by constant rate, we can slide the window through the whole time domain. In the meantime, the parameter a in the function g can control the width of the window, determining how long the time interval should be for each slide. Also, With the data in both the time interval and frequency domain, we can construct a spectrogram to show their relationships. However, the drawback of the Gabor transform is that we cannot capture high resolution in both time and frequency domain due to Heisenberg's uncertainty principle.

3.2 Wavelet

With simple modification to the Gabor Transform, we can increase the resolution in time successively: First, we extract low frequencies with wider window with low resolution in time. Then, we increase the width of window to extract high frequency and high resolution in time, which is the fundamental principle of wavelet theory. Here is the mother wavelet:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a}) \tag{3}$$

where $a \neq 0$ and b is real constant. The parameter a controls the width of window and b controls the translation.

1. **Gaussian window**

   The Gaussian window is a modification of normal distribution, which eliminates anything far from mean(target time) and enlarge the part around target time. We should notice that in the normal distribution, the variance is on the denominator, which means as we increase variance, the parameter in the exponential part becomes smaller. However, in the Gaussian filter, the parameter is on the numerator, which means as we increase the parameter a, we decrease variance(width). Here is comparsion between normal distribution and Gaussian filter:

$$g(t - \tau) = e^{-a(t-\tau)^2} \tag{4}$$

$$N(\mu, \sigma^2) \sim e^{-(\frac{x-\mu}{2\sigma})^2} \tag{5}$$

2. **Mexican hat wavelet**

The Mexican hat wavelet has excellent localization properties in both time and frequency due to the minimal time-bandwidth product of the Gaussian function:

$$\psi(t) = \frac{2}{\sqrt{3a}\pi^{1/4}}(1 - (\frac{t-\tau}{a})^2)e^{-\frac{(t-\tau)^2}{2\sigma^2}} \tag{6}$$

which is essentially a second moment of a Gaussian in the frequency domain.

3. **Step function**

   The step function is just a function with 0 outside the [$\tau-$width $\tau+$width] and 1 inside the [$\tau-$width $\tau+$width].

4 Algorithm Implementation and Development

4.1 Loading data(Part one)

First, we load the data handel and get Fs and y. Then, we define the number of points(73113) in the data as n. In addition, we define t as (1-length(v))/Fs and the length of time domain(L) is from 0 to length(v)/Fs. Because we have an odd number of data points, we define k from -(n-1)/2 to (n-1)/2, which will give us odd numbers of k. Also, we do the same re-scaling by multiplying (2pi/L) with k. Finally, we do fftshift on k to make it easy to plot.

4.2 Spectrograms with varying parameters(Part one)

First, we create 'a' vector to contain 10, 1, 0.1. And then, we determine the change of slide of window is 0.1 from 0 to 9(tslide vector). And then we create Vgtspec vectors(length(slide),n) to store all the output, where we have length(slide) times of output and each one is length of n. After that, we create an outer for-loop to loop over vector 'a' and an inner for-loop to loop over tslide. In the inner for-loop, at each iteration, we define a new gaussian window which is g=exp(-a*(t-tslide)^2) where a is current 'a' vector in outer loop and tslide is current tslide vector in inner loop. Then, we multiply our signal data v with our filter and then take Fourier transform on it. To store our fourier transform data, we need shift this abs(data) to match up with shifted k. Finally, we plot a spectrogram for each 'a'.
Then, we fix a = 1 and vary the dt in the tslide. Without outerloop, we only loop over dt = 0.01(oversampling) and 1(undersampling). We do same thing like before to plot spectrogram for each dt.

4.3 Spectrograms with different windows(Part one)

For our convenience, we put the Mexican hat wavelet and step function both in the for loop where we vary parameter a(We do not need to create another for-loop). We define m as Mexican hat wavelet(same form in the background part), where τ is current tslide in the tslide vector in the inner loop and a is the value in the outer loop. For step function, we follow with it's definition: we create a zero vector with length n as v and define all the index in t vector

where tsilde(τ) - a $< t <$ tsilde(τ) + a, as temp. Finally, replace the corresponding index in the zero vector(v(temp)) with one, which will give us the step function. Doing this process, we create a step function to filter out data around [$\tau-$width $\tau+$width]. After we define these functions, we do same procedures before to plot their corresponding spectrograms.

4.4 Loading data(part two)

First, we load piano audio and get y and Fs. Then, we define the length of y as n(number of points) and t,L as same as before. Here we have even n, then we define k from -n/2 to n/2 and rescaling it,then do fftshift on it to get kx. Same procedure for recorder data.

4.5 filtering out overtones and plotting spectrograms

First, we define the a = 20 and dt=0.1 for the Gaussian window. Also, we create a music score vector to store the filtered frequency and vgt spec to store all fft values at each time step. Then, we loop over tslide(0:dt:L), multiply v with Gaussian window, take fft on the result, and store iffshift(abs(result)) to the vge spec. At the end of the loop, we just plot its spectrograms(we plot ks/2pi rather than ks because we want our unit to be Hz). Those steps are as same as before. To filter out overtone, at each step, we find the maximum in the fft(v*g), which tells us the most intense frequency around tslide(j) and plug the index in k(not ks because we do not shift the fft(v*g)). This k gives us the most intense frequency around tsilde(j). To store this k(k/2pi because we need a unit in Hz) value to the music score each time, in the end, we can get a decent reproduction of music score. And we do the same procedure for recorder data.

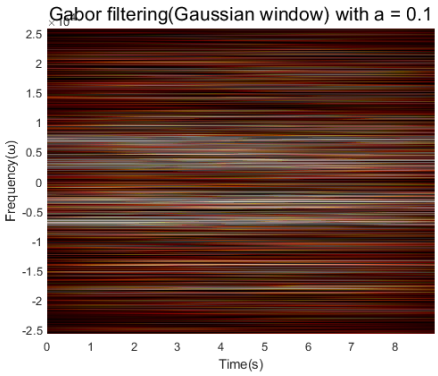
5 Computational Results

5.1 part one 1.2

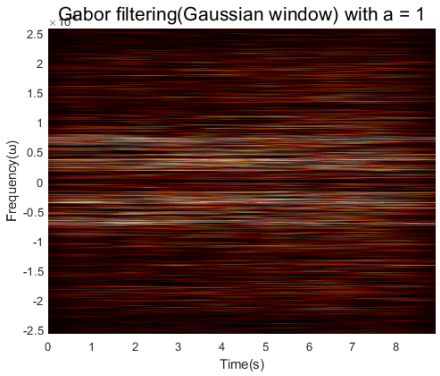
First, we start with Gaussian window. Figure 2 shows that if we start with parameter a = 1 and dt = 0.1, we can get a normal picture: some information in time and some information in frequency. However, when we vary the window size(a), things change a lot. When a is small(0.1), which means large window size, we got no information in the time domain and all we have are some variations of frequencies(should be same to spectrogram created by no window). When a is large(10), which means window size is small, we can easily distinguish some time chunks. However, we cannot find low frequencies in the spectrogram. It is a trade-off to determine whether we want more information in frequency domain or in the time domain.

5.2 part one 3

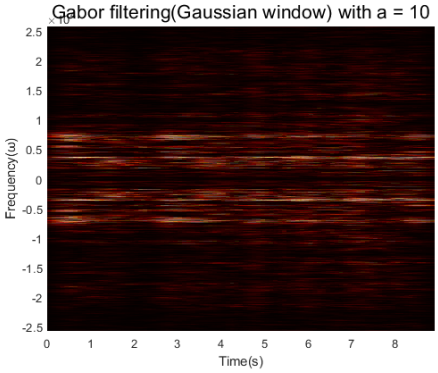
When we fix a = 1, we vary dt to see some influnences. First, looking at figure 5, we get really bad resolution both in time and frequency domain. This is because we lose data in two slide. We call this undersampling. Looking at figure 4, there is not much difference between



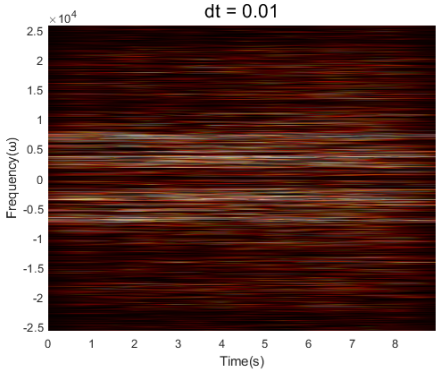
(a) Figure 1 : spectrogram with Gaussian filter where a = 0.1



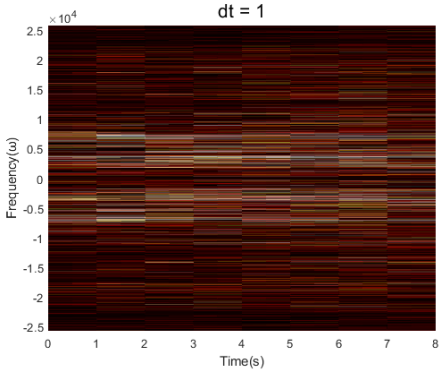
(b) Figure 2 : spectrogram with Gaussian filter where a = 1



(c) Figure 3 : spectrogram with Gaussian filter where a = 10

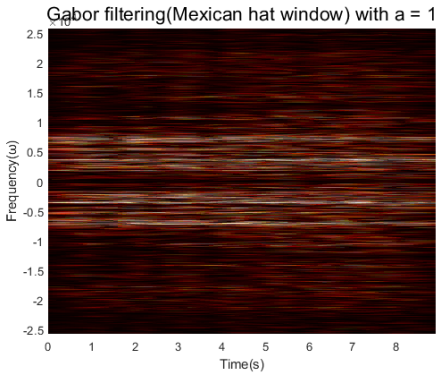


(a) Figure 4 : spectrogram with Gaussian filter where dt=0.01

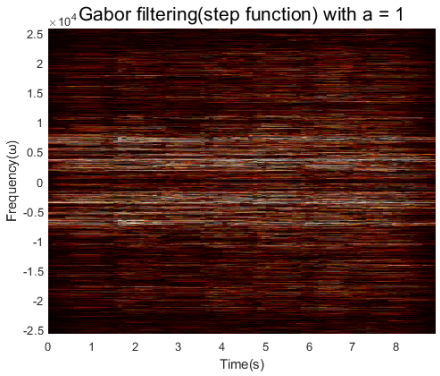


(b) Figure 5 : spectrogram with Gaussian filter where dt = 1

this figure and figure 2(dt=0.1). However, to generate this figure, we iterate 10 more times than generating figure 2, which is a huge cost.



(a) Figure 6 : spectrogram with Mexican hat wavelet where a=1



(b) Figure 7 : spectrogram with Step function where a = 1

5.3 part one 4

Comparing figure 6,7,2 we can tell that step function gives us the worst resolution of spectrogram because step function does not enlarge the target range. And Gaussian window looks a little better than Mexican hat wavelet.

5.4 part two

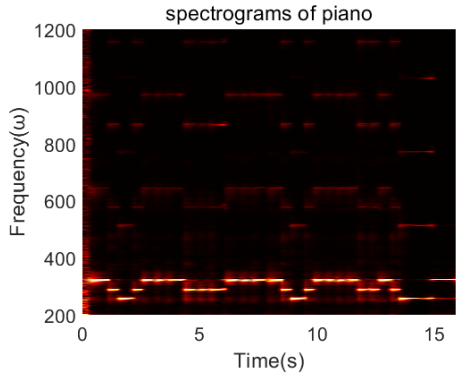
To produce spectrograms for these two audio, we take a = 20 and dt = 0.1. Looking at figure 10 and 11, we can tell that the fundamental frequency for piano is around 200 to 350 Hz and for recorder is from 750 to 1050 Hz. To figure out how many nodes of music score,we zoom in the figure 8 and figure 9 to see the separations in times. And the music score for piano can be count by looking at Figure 12:**EDCDEEEDDDEEE EDCDEEEEDEDC**. And the music score for recorder can be count by looking at Figure 13:**BAGABBBAAABBB BAGABBBBAABAG**. Also, looking at figure 8 and figure 9, we can say that recorder has fewer overtones than pianos. Also, the frequency of recorder is as same patteren as the frequency of piano.

6 Conclusion

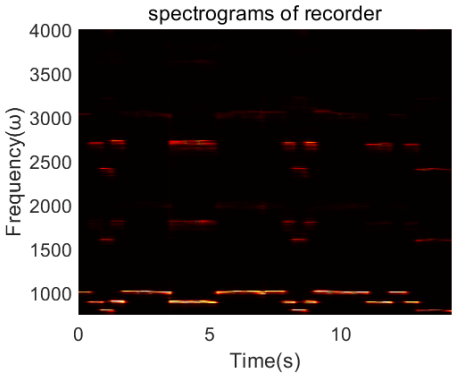
In this project, we use Gabor Transform with different windows to plot spectrograms for audio data. Also, we adjust the parameters of Gaussian window to see the different pattern of different parameter. Later, we also reduce the overtones in the music to reproduce the music score.

7 Appendix A: MATLAB functions used and brief implementation explanation

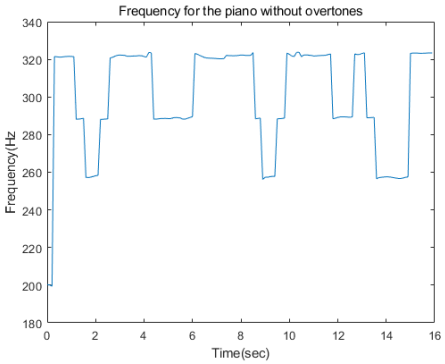
- pcolor(x,y,z): draw color to represent the intense of z at grid(x,y).



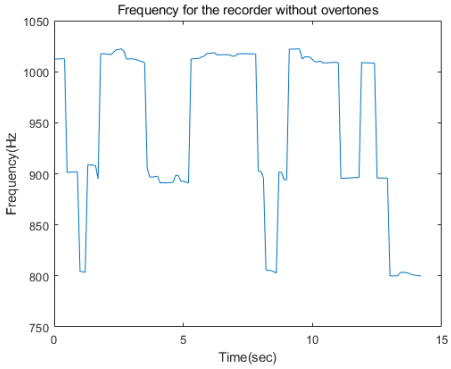
(a) Figure 8 : spectrogram for piano



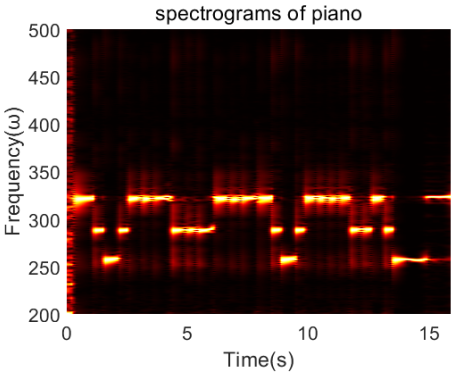
(b) Figure 9 : spectrogram for recorder



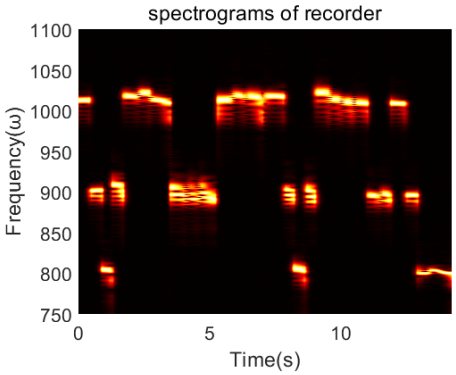
(c) Figure 10 : frequency for piano without overtones



(d) Figure 11 : frequency for recorder without overtones



(e) Figure 12 : zoom in spectrogram for piano



(f) Figure 13 :zoom in spectrogram for recorder

- ffshift(k): Shift zero-frequency component to center of spectrum. we use it to make graph easier to read.
- max(frq(:)): get coordinate of maximum value.

8 Appendix B: MATLAB codes

```
clc;close all;clear all
%part 1
load handel
v=y';
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('signal of Interest, v(n)')

n=length(v); %how many points do we have
t = (1:length(v))/Fs; % follow what instruction did to define t
L=length(v)/Fs; %around 9 seconds
k = (2*pi/L)*[0:(n-1)/2 -(n-1)/2:-1]; % rescale to L domain
ks = fftshift(k); %easy for ploting.


%% Spectrograms for varying window sizes(5,1,0.2 with dt = 0.1)
figure(2)

a_vec = [10 1 0.1];
for i = 1:length(a_vec)
    a = a_vec(i);
    tslide=0:0.1:L;
    Vgt_spec = zeros(length(tslide),n);%gaussian window
    Vmgt_spec = zeros(length(tslide),n);%mexican hat wavelet
    Vsgt_spec = zeros(length(tslide),n);%step function
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        m =
            2/(pi^1/4*sqrt(3*a))*(1-((t-tslide(j))/a).^2).*exp(-((t-tslide(j))/a).^2);
        s = zeros(1,length(v));
        temp = find(tslide(j)-a<t & t< tslide(j)+a);
        s(1,temp(1):temp(end)) = ones(1,length(temp));
        Vg=g.*v;
        Vmg=m.*v;
        Vsg=s.*v;
        Vgt=fft(Vg);
        Vmgt=fft(Vmg);
        Vsgt=fft(Vsg);
        Vgt_spec(j,:) = fftshift(abs(Vgt)); % match with ks
        Vmgt_spec(j,:) = fftshift(abs(Vmgt));
        Vsgt_spec(j,:) = fftshift(abs(Vsgt));
    end

    figure(i)
    pcolor(tslide,ks,Vgt_spec.'),
    shading interp
```

```
    title(['Gabor filtering(Gaussian window) with a = ',num2str(a)],'Fontsize',16)
    xlabel('Time(s)')
    ylabel('Frequency()')
    colormap(hot)

    figure(i+3)
    pcolor(tslide,ks,Vmgt_spec.'),
    shading interp
    title(['Gabor filtering(Mexican hat window) with a =
        ',num2str(a)],'Fontsize',16)
    xlabel('Time(s)')
    ylabel('Frequency()')
    colormap(hot)

    figure(i+6)
    pcolor(tslide,ks,Vsgt_spec.'),
    shading interp
    title(['Gabor filtering(step function) with a = ',num2str(a)],'Fontsize',16)
    xlabel('Time(s)')
    ylabel('Frequency()')
    colormap(hot)

end

%% Spectrograms for varying dt[0.01 1] with a =1
a=1;
dt_vec = [0.01 1];
for jj = 1:length(dt_vec)
    dt = dt_vec(jj);
    tslide=0:dt:L;
    Vgt_spec = zeros(length(tslide),n);
    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()
    pcolor(tslide,ks,Vgt_spec.'),
    shading interp
    title(['dt = ',num2str(dt)],'Fontsize',16)
    xlabel('Time(s)')
    ylabel('Frequency()')
    colormap(hot)
end
```

```
%% part 2
%% piano
[y,Fs] = audioread('music1.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)');

n=length(y); %how many points do we have
t = (1:length(y))/Fs; % follow what instruction did to define t
L=tr_piano; %around 16 seconds
k = (2*pi/L)*[0:(n-1)/2 -n/2:-1]; % rescale to L domain
ks = fftshift(k); %easy for ploting
v = y';

a=20;
dt=0.1;
tslide=0:dt:L;
music_score = zeros(1,length(tslide));
Vgt_spec = zeros(length(tslide),n);%gaussian window
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        Vg=g.*v;
        Vgt=fft(Vg);
        Vgt_spec(j,:) = fftshift(abs(Vgt))/max(abs(Vgt)); % match with ks
        [mxv,idx] = max(abs(Vgt));
        music_score(j) = k(idx)/(2*pi); % no shifting of k and convert it to Hz
    end
plot(tslide,music_score)
title('Frequency for the piano without overtones')
xlabel('Time(sec)')
ylabel('Frequency(Hz')
 figure()
    pcolor(tslide,ks/(2*pi),Vgt_spec.'),
    shading interp
    title('spectrograms of piano','Fontsize',16)
    xlabel('Time(s)')
    ylabel('Frequency()')
    set(gca,'Ylim',[200 1200],'Fontsize',16) %[200 450] to zoom in
    colormap(hot)

%% recorder
figure(2)
[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)');
n=length(y); %how many points do we have
t = (1:length(y))/Fs; % follow what instruction did to define t
L=tr_rec; %around 14 seconds
k = (2*pi/L)*[0:(n-1)/2 -n/2:-1]; % rescale to L domain
ks = fftshift(k); %easy for ploting
v = y';

a=20;
dt=0.1;
tslide=0:dt:L;
music_score = zeros(1,length(tslide));
Vgt_spec = zeros(length(tslide),n);%gaussian window
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        Vg=g.*v;
        Vgt=fft(Vg);
        Vgt_spec(j,:) = fftshift(abs(Vgt))/max(abs(Vgt)); % match with ks
        [mxv,idx] = max(abs(Vgt));
        music_score(j) = k(idx)/(2*pi); % no shifting of k and convert it to Hz
    end
plot(tslide,music_score)
title('Frequency for the recorder without overtones')
xlabel('Time(sec)')
ylabel('Frequency(Hz')
 figure()
    pcolor(tslide,ks/(2*pi),Vgt_spec.'),
    shading interp
    title('spectrograms of recorder','Fontsize',16)
    xlabel('Time(s)')
    ylabel('Frequency()')
    set(gca,'Ylim',[750 4000],'Fontsize',16) %[750 1100] to zoom in

    colormap(hot)
```