Time - Frequency Analysis

Yinxi Pan

AMATH482, Homework2, Winter2020

February 8, 2020

Abstract

The first part analysis a music in MATLAB. Produce the spectrogram of the music by Gabor transform, Mexican Hat wavelet, and Shannon wavelet. Examine effect of the width of the Gabor filter on the music data. Explore the idea of oversampling and undersampling. The second part reproduces music scores of a song in two distinct versions. Filter the data by Gabor transform and plot them in spectrogram. Discover the difference between the two recordings.

1 Introduction and Overview

In this report, we deal with the time-frequency analysis. This is a modification of Fourier transform as we discussed in the first report. Fourier transform gives us sufficient information about frequency but we don't know when the frequency happens among time. Time-frequency analysis is also called the Windowed Fourier Transforms. Based on the name of it, this analysis moves the Fourier transform over a time period. One example is the Gabor transform which allows us to extract data through different time intervals. In this way, we could get both the time and frequency information. In later section, we first apply the idea of Gabor transform, evaluate the time-frequency information on given audio files, and compare how different types of transforms affect the solution. Then, we apply the Gabor transform to a song in two distinct versions and reproduce the music score[1] for it.

2 Theoretical Background

1. **Gabor Transform**[2] : Improved from Fourier transform.

[1] Music score is a written form of a musical composition; parts for different instruments appear on separate staves on large pages("Musical Score - Dictionary Definition." Vocabulary.com, www.vocabulary.com/dictionary/musical score.)

[2] Conceptual definitions are all from: Kutz, J. Nathan. Data-Driven Modeling Scientific Computation: Methods for Complex Systems Big Data, Chap13.4-5 OUP Oxford, 2013.

We introduce Fourier transform(FT) in the first report. However, FT has severe limitations since it only captures the signal during the whole time but fail to get the moment in time when the frequency happens. The Gabor Transform, also called the short-time Fourier transform (STFT), is defined as in Equation(1),

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

where the bar denotes the complex conjugate of the function. This transform localizes both time and frequency such that $g(\tau - t)$ acts as a time filter to localize both signal over a window of time . In practice, we use Gabor transform by discretizing the time and frequency domain. Thus, the discrete version of Gabor transform is defined in equation(2)

$$\hat{f}(m,n) = \int_{-\infty}^{\infty} f(t)\bar{g}_{m,n}(t)dt, \text{s.t. } g_{m,n}(t) = e^{i2\pi m\omega_0 t}g(t - nt_0) \tag{2}$$

However, Gabor transform does not ensure perfect time-frequency resolution since it's impossible to both the time and frequency perfectly by *Heisenberg Principle*. So we have to play around with the parameters and see how results change.

2. **Wavelet**[3]: Wavelet is a modification of Gabor transform. It includes translate and scales. This report also includes the Gaussian wavelet but it's the same as the Gaussian filter we discussed in the first report.

   (a) **Mexican hat wavelet**

   The Mexican hat wavelet is essentially a second moment of a Gaussian in the frequency domain and it has excellent localization properties in both time and frequency due to the minimal time-bandwidth product of the Gaussian function.The wavelet and it's transformation are defined as in equation(3)

$$\phi(t) = \frac{2}{\sqrt{3\sigma}\cdot\pi^{\frac{1}{4}}}(1 - (\frac{t}{\sigma})^2)e^{-\frac{t^2}{2\cdot\sigma^2}} \tag{3}$$

   (b) **Shannon wavelet**

   The Shannon filter is simply a step function with value of unity within the transmitted band and zero outside of it. The filter is defined as in equation(4)

$$\phi(x) = \begin{cases} 1, & |x| < \frac{1}{2} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

[3]Conceptual definitions are all from: Kutz, J. Nathan. Data-Driven Modeling Scientific Computation: Methods for Complex Systems Big Data, Chap13.4-5 OUP Oxford, 2013.

3 Algorithm Implementation and Development

1. The first part of the code explores the time frequency signature of **handel** in Matlab.[4]

   (a) Load the data, present the audio file as a vector and create the frequency axis based on the vector.Un-comment the code to see how the original audio file looks like.

      i. The length of the audio is determined by the length of the vector and the sample rate of the audio file.

      ii. Since we are going to use fft() later, we need to discritize the range of x. Also, we need to rescale all frequencies by 2π since the fft() assumes 2π periodic signals. Moreover, we need to use fftshift() making 0 at the center of the axis to avoid aliasing phenomena when plotting.

   (b) Create time slides to compare how the width of a Gaussian filter affects the spectrogram. The method of applying Gaussian filter is the same as in the first report. The only difference is since we are doing time-frequency analysis the original t in the function should be $(t - tslide)$ in order to move the filter.

   (c) Use the same time slides and try Mexican Hat wavelet and Shannon Wavelet on the audio. The idea of applying them is the same as Gaussian filter. Formula these wavelets are obtained from Section2.

   (d) Create oversampling and undersampling Gaussian filter and apply to the original vector. The difference between oversampling and undersampling is the length of time slides. Oversampling is defined as a longer time slide vector and undersampling is defined as a shorter time slide vector.

   (e) Use pcolor() to plot the spectrogram in frequency domain. When plotting, we have to divide the frequency axis by 2π since the unit for audio files is Hz.

2. The second part of the code evaluates the music scores of two different versions of *Mary had a little lamb*. The implementation are the same for two pieces of music. We only need to change the read-in audio file.[5]

   (a) Load the data, present the audio file as a vector and create the frequency domain axis based on the vector. Un-comment the code to see how the original song looks like. We apply the same method as in the first part.

   (b) Create time slides and loop through it. Inside the loop, we apply the Gaussian filter to transform the data with the same method and idea as in the first report. The only difference is since we are doing time-frequency analysis the original t in the function should be $(t - tslide)$ in order to move the filter.

   (c) Use pcolor() to plot the spectrogram. When plotting, we have to divide the frequency axis by 2π since the unit for audio files is Hz.

[4] Matlab code check Appendix.B with Part1

[5] Matlab code check Appendix.B with Part2

4 Computational Results

Part1: Analysis of handel

Figure7 shows the spectrogram of the piece of work by Gabor filtering(implemented with Gaussian filter) with $a = 0.1, 10, 100, 1000$ where a indicates the decreasing rate in Gaussian filter. The larger a implies narrower window, and smaller a implies wider window. Thus, for $a = 0.1$, we get a bunch of frequency information, but lack of time information. As a grows, the spectrogram looks better and it's approximately the same when $a = 100$ and $a = 1000$. Figure1 shows the spectrogram produced by the Mexican Hat wavelet. Figure2 shows the spectrogram produced by the Shannon wavelet. They basically look the same as the Gaussian filter, but Shannon wavelet looks a bit better than Mexican hat. This is because Shannon wavelet is a step function and it captures more resolution in frequency than Mexican Hat wavelet which is a continuous function. Figure3 and Figure4 shows two Gabor transform with Gaussian filter in different time slides. Figure3 is the undersampling example and Figure4 is the oversampling example. By saying undersampling, it means the windows don't cover much information while oversampling means that we have too many windows to capture the information. From the figures, we could see that oversampling does a better job than undersampling. Although oversampling gives a good spectrogram, it takes a long time to run the code. That's why we need to find appropriate parameter to transform data.

Part2: Analysis of Mary has a little lamb

Figure5 and Figure6 show the spectrogram of the music score of *Mary has a little lamb* in piano and recorder. We could read off approximate music scores from the spectrogram. For piano, we have music scores to be (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) where C = 261Hz, D = 293Hz, E = 329Hz approximately. For recorder, we have music scores to be (C A G A C C C A A A C C C C A G A C C C C A A C A G) where C = 1046.5Hz, A = 880Hz, G = 783.99Hz. The music scores(in Hz) are higher in recorder than in piano. Also, we could see that recorder has a clearer spectrogram than piano, though they look similar. In general, different instruments have different timber which relate to the overtones. From the spectrogram, piano seems to have more overtones than recorder. But the relative difference between then are approximately the same.

5 Summary and Conclusions

In the first part, we analysis the time frequency information for an audio file in Matlab. We apply the Gabor transform to visualize the file in spectrogram(Figure(7)). We also try Mexican hat wavelet and Shannon wavelet to compare with the Gabor filtering(Figure(1&2)) which look the same. We also try change the parameters in the filter function(Figure(7)) and realize that larger a makes the spectrogram looks better. We also explore the visualization between oversampling and undersampling(Figure(3&4)). In the second part, we analysis a song played in piano and recorder. We apply the Gabor transform, filter out the overtone and reproduce the music score presented in the spectrogram(Figure(5&6)).For piano, we have music scores to be:

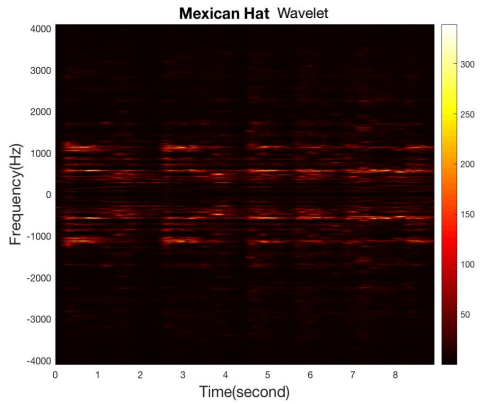

Figure 1: Spectrogram for Mexican Hat wavelet with $\sigma = 0.05$ where σ is the parameter for Mexican Hat wavelet as defined in equation(3).

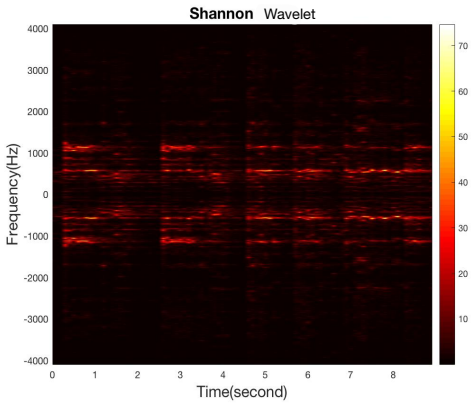

Figure 2: Spectrogram for Shannon wavelet with a width 0.025.

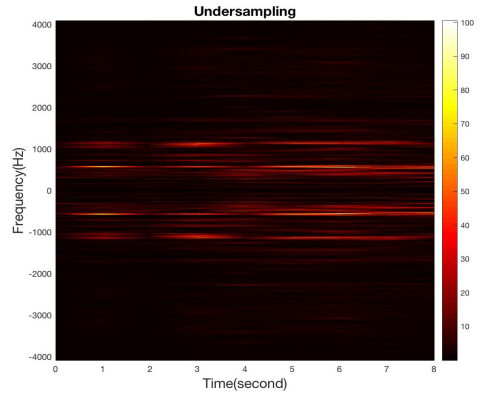

Figure 3: Spectrogram for undersampling case. Time slides increment with 1.With $a = 100$ in the Gaussian filter.

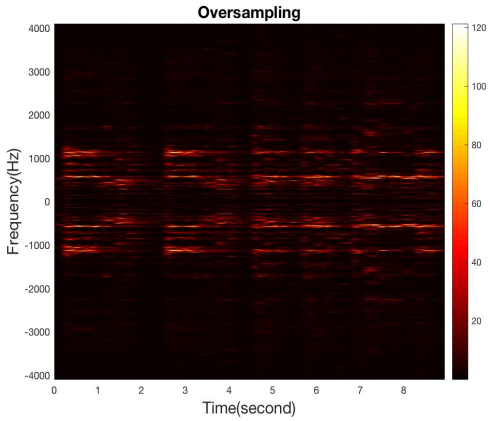

Figure 4: Spectrogram for oversampling case.Time slides increment with 0.01. With $a = 100$ in the Gaussian filter.

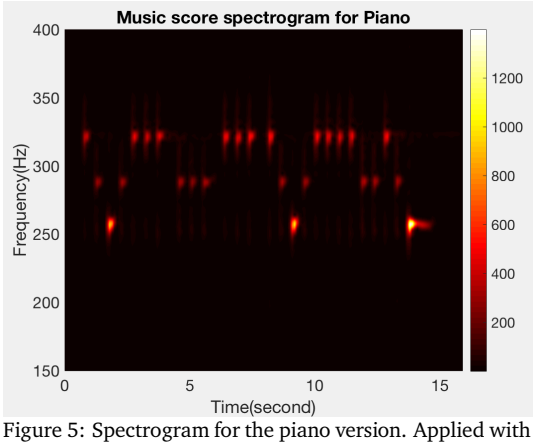

Figure 5: Spectrogram for the piano version. Applied with Gaussian filter.

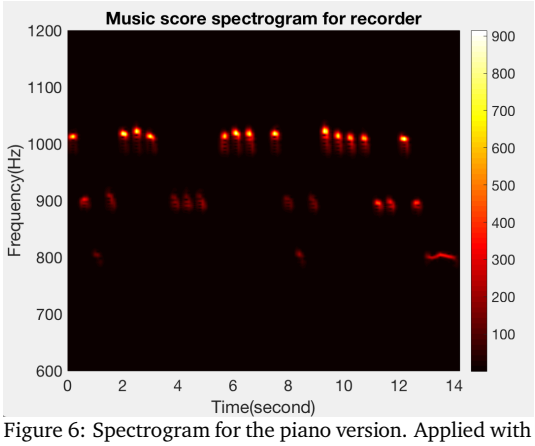

Figure 6: Spectrogram for the piano version. Applied with Gaussian filter.

(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) where C = 261Hz, D = 293Hz, E = 329Hz approximately. For recorder, we have music scores to be:

(C A G A C C C A A A C C C C A G A C C C C A A C A G) where C = 1046.5Hz, A = 880Hz, G = 783.99Hz approximately.

Appendix A.MATLAB functions used[6]

1. audioread('filename'): reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.
2. audioplayer(): creates an audioplayer object for signal Y, using sample rate Fs. The function returns the audio player object, player.
3. colormap(hot): Set background color to be red.
4. colorbar: displays a vertical colorbar to the right of the current axes or chart.
5. 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. For example, if X and Y define an m-by-n grid, then C must be an m-by-n matrix.

[6]Other functions applied in this report can be found in the first report. The full description of each function below can be foun on : "MATLAB." MATLAB Documentation, www.mathworks.com/help/matlab/.

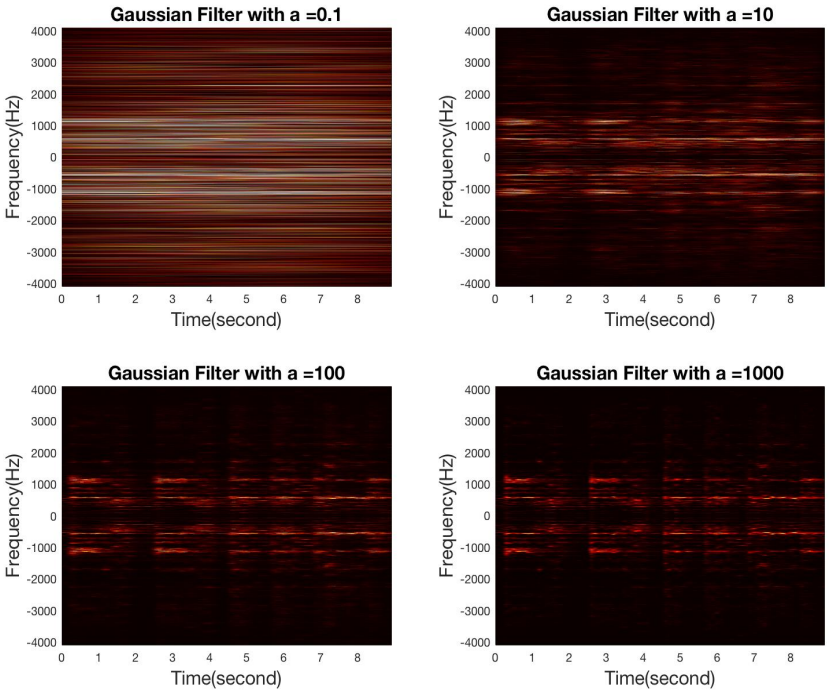

Figure 7: Spectrogram for handle in Matlab. Each applies with different value of a.

Appendix B. MATLAB codes

```
% Part1
clear; close all; clc;
load handel
v = y';
% plot((1:length(v))/Fs,v);
% xlabel('Time [sec]');
% ylabel('Amplitude');
% title('Signal of Interest, v(n)');

% play the music
p8 = audioplayer(v,Fs);
% playblocking(p8);

L=length(y)/Fs; % length of the music
n=length(v);
t2=linspace(0,L,n+1);
t=t2(1:n);
k=(2*pi/L) * [0:(n-1)/2 -(n-1)/2:-1];
ks=fftshift(k);

%% produce spectrograms of the piece of work.
tslide=0:0.1:L;
Vgt_spec = zeros(length(tslide),n);
Vmht_spec = zeros(length(tslide),n);
Vshnt_spec = zeros(length(tslide),n);

% Gaussian
a_vec = [0.1 10 100 1000];
for jj = 1:length(a_vec)
    a = a_vec(jj);

    for j=1:length(tslide)

    g=exp(-a*(t-tslide(j)).^2);
    Vg=g.*v;
    Vgt=fft(Vg);
    Vgt_spec(j,:) = fftshift(abs(Vgt)); % We don't want to scale it

    end

    subplot(2,2,jj)
    pcolor(tslide, ks/(2*pi), Vgt_spec.')
    shading interp
    xlabel('Time(second)','Fontsize',16)
    ylabel('Frequency(Hz)','Fontsize',16)
    title(['Gaussian Filter with a =', num2str(a)],'Fontsize',16)
    colormap(hot)
end


a = 100;
for j=1:length(tslide)
    % Gaussian
```

```
    g=exp(-a*(t-tslide(j)).^2);
    Vg=g.*v;
    Vgt=fft(Vg);
    Vgt_spec(j,:) = fftshift(abs(Vgt)); % We don't want to scale it
    % hat
    sigma = 0.05;
    constant = 2 / (sqrt(3*sigma)* (pi^(1/4)));
    mh = constant .* (1-((t-tslide(j))./sigma).^2) .* exp(-(((t-tslide(j)).^2) ./ ...
        (2 * sigma.^2)));
    Vmh = mh.*v;
    Vmht = fft(Vmh);
    Vmht_spec(j,:) = fftshift(abs(Vmht)); % We don't want to scale it
    % Shannon
    shn = abs(t - tslide(j)) <= 0.05/2;
    Vshn = shn .* v;
    Vshnt = fft(Vshn);
    Vshnt_spec(j,:) = fftshift(abs(Vshnt)); % We don't want to scale it
end


%% oversample and undersample
tslideover=0 : 0.01 : L;
over = zeros(length(tslideover),n);
for j = 1:length(tslideover)
    filterover = exp(-a*(t-tslideover(j)).^2);
    vfo = filterover .* v;

    vfto = fft(vfo);
    over(j,:) = fftshift(abs(vfto));
end


tslideunder=0 : 1 : L;
under = zeros(length(tslideunder),n);
for j = 1:length(tslideunder)
    filterunder = exp(-a*(t-tslideunder(j)).^2);
    vfu = filterunder .* v;

    vftu = fft(vfu);
    under(j,:) = fftshift(abs(vftu));
end


%% Figures
figure(1)
pcolor(tslide, ks/(2*pi), Vgt_spec.')
shading interp
xlabel('Time(second)','Fontsize',16)
ylabel('Frequency(Hz)','Fontsize',16)
title('Gaussian Filter','Fontsize',16)
colormap(hot)
colorbar

figure(2)
pcolor(tslide, ks/(2*pi), Vmht_spec.')
shading interp
xlabel('Time(second)','Fontsize',16)
ylabel('Frequency(Hz)','Fontsize',16)
```

```
title('Mexican Hat Filter','Fontsize',16)
colormap(hot)
colorbar

figure(3)
pcolor(tslide, ks/(2*pi), Vshnt_spec.')
shading interp
xlabel('Time(second)','Fontsize',16)
ylabel('Frequency(Hz)','Fontsize',16)
title('Shannon Filter','Fontsize',16)
colormap(hot)
colorbar

figure(4)
pcolor(tslideover, ks/(2*pi), over.')
shading interp
xlabel('Time(second)','Fontsize',16)
ylabel('Frequency(Hz)','Fontsize',16)
title('Oversampling','Fontsize',16)
colormap(hot)
colorbar

figure(5)
pcolor(tslideunder, ks/(2*pi), under.')
shading interp
xlabel('Time(second)','Fontsize',16)
ylabel('Frequency(Hz)','Fontsize',16)
title('Undersampling','Fontsize',16)
colormap(hot)
colorbar


%% PART2 piano
clear; close all; clc;
% Piano audio
[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)');
p8 = audioplayer(y,Fs);
% playblocking(p8);

L = tr_piano;
n = length(y);
t2 = linspace(0,L,n+1);
t = t2(1:n);
k = (2 * pi /L) * [0 : (n / 2 - 1) (-n/2): -1];
ks = fftshift(k);
v = y';

a = 100;
tslide = 0:0.1:L;
Vgt_spec = zeros(length(tslide),n);

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

```
169
170     Vg = g .* v;
171     Vgt = fft(Vg);
172
173     Vgt_spec(j,:) = fftshift(abs(Vgt)); % We don't want to scale it
174 end
175
176 % recorder audio
177 [yr,Fsr] = audioread('music2.wav');
178 tr_r=length(yr)/Fsr; % record time in seconds
179 % plot((1:length(yr))/Fsr,yr);
180 % xlabel('Time [sec]');
181 % ylabel('Amplitude');
182 % title('Mary had a little lamb (recorder)');
183 p8 = audioplayer(yr,Fsr);
184 % playblocking(p8);
185
186 Lr = tr_r;
187 nr = length(yr);
188 t2r = linspace(0,Lr,nr+1);
189 tr = t2r(1:nr);
190 kr = (2 * pi /Lr) * [0 : (nr / 2 - 1) (-nr/2): -1];
191 krs = fftshift(kr);
192 vr = yr';
193
194 a = 100;
195 tslider = 0:0.1:Lr;
196 Vrgt_spec = zeros(length(tslider),nr);
197
198 for j=1:length(tslider)
199     gr = exp(-a * (tr - tslider(j)) .^ 2);
200     Vrg = gr .* vr;
201     Vrgt = fft(Vrg);
202
203     Vrgt_spec(j,:) = fftshift(abs(Vrgt)); % We don't want to scale it
204 end
205
206 %% spectrograms
207 figure(1)
208 subplot(2,1,1)
209 pcolor(tslide,ks/(2*pi),Vgt_spec.'),
210 shading interp
211 xlabel('Time(second)')
212 ylabel('Frequency(Hz)')
213 title('Music score spectrogram for Piano','Fontsize',16)
214 set(gca,'Ylim',[0 400],'Fontsize',16)
215 colorbar
216 colormap(hot)
217
218 subplot(2,1,2)
219 pcolor(tslider,krs/(2*pi),Vrgt_spec.'),
220 shading interp
221 title('Music score spectrogram for recorder','Fontsize',16)
222 xlabel('Time(second)')
223 ylabel('Frequency(Hz)')
224 set(gca,'Ylim',[600 1200],'Fontsize',16)
225 colorbar
226 colormap(hot)
```