Gabor transforms

Author : Richy Yu

Abstract

This paper deals with one application of Gabor Transform. Given a music recording, I would use Gabor Transform to create a spectrogram for analyzation. I will also explore how the window width, oversampling, and types of function window affect the spectrogram. The programming software used is MATLAB.

Introduction and Overview:

Part one :

In part one, I will analyze a portion of Handel's Messiah with time-frequency analysis. More specifically, I will explore the time- frequency signature of this 9 second piece of classic work. The following will be the things I do : Firstly, through use of the Gabor filtering we used in class, produce spectrograms of the piece of work. Secondly, explore the window width of the Gabor transform and how it effects the spectrogram. Thirdly, explore the spectrogram and the idea of oversampling (i.e. using very small translations of the Gabor window) versus potential under sampling (i.e. using very course/large translations of the Gabor window). Fourthly, use different Gabor windows. Perhaps you can start with the Gaussian window, and look to see how the results are affected with the Mexican hat wavelet and a step-function (Shannon) window.

Part two :

In part two, I will analyze music recordings for the song *Mary had a little lamb* one both the recorder and piano. Through use of the Gabor filtering we used in class, I will reproduce the music score for this simple piece. I will also explore the difference between a recorder and piano.

Theoretical Background

The mathematical concept that is used in this paper is Gabor Transform. Gabor Transform is a modification of Fourier Transform. The drawback of Fourier Transform is that when the original signal is transformed into frequency domain, we lose the information about time. In Gabor Transform, we will divide the original data into different windows and do the Fourier Transform in each window, so there will be a time associated with each window. In this way, we will be able to preserve the information about the time. Using this technique, I will be able to plot the frequency versus time, and such plot is called spectrogram. Gabor Transform is mathematically defined as

$$[f](t,\omega) = \int_{-\infty}^{\infty} f(\tau)\bar{g}(\tau - t)\, e^{-i\omega\tau} d\tau = (f, \bar{g}_{t,\omega})\ . \qquad \text{eq 1}$$

However, I will not be directly using this formula. What I will use is the MATLAB function fft. I will first use window function to filter the data, and then use fft to transform the data inside the window. Three specific filter functions will be used. They are gaussian filter, Mexican hat wavelet, and a step-function (Shannon) window. Gaussian filter is defined as

$$g(t) = \ e^{-a(t-b)^2}. \qquad \text{eq 2}$$

Mexican hat wavelet is defined as

$$\varphi(t) = \frac{2}{\sqrt{3\sigma}\pi^{1/4}}(1\text{-}\left(\frac{t}{\sigma}\right)^2)\mathrm{e}^{-\frac{t^2}{2\sigma^2}}, \qquad \text{eq 3}$$

A step-function (Shannon) window is defined as

$$Y(t) = \begin{cases} 1 \text{ if } a < t < b \\ 0, \text{otherwise} \end{cases}. \qquad \text{eq 4}$$

Algorithm Implementation and Development

Part one :

Firstly, I define n as the length of our array, which is the number of data points we have. L is the length of the recording, which I got from length of the array divided by Fs, which is the sampling frequency in HZ. I partition the range [0, L] by n + 1 points, and throw away the last point and keep the first n points, because the last point has the value of L. Since n is an odd number, the frequency k is defined differently. I scale the k by 1/L because the length of our data is L. I want to explore three different widths of the window by changing the parameter's value. More specifically three parameter values are 1, 100, 1000, respectively.

I partition the range [0, L], with increment 0.1. I loop through each point in the tslide and define three gaussian functions with those three values mentioned above. I multiply this function with the data and take the Fourier Transform to represent the data in the frequency domain. Then I take the absolute value and apply fftshift on the result. Then I am able to plot the spectrogram with different window width.

I also explore the different number of points in the time domain by partitioning the range [0, L] with different increments. I tried 1 and 0.01 as the increments. All of the rest procedures are the same as before.

I also use different types of window functions, and they are gaussian, Mexican hat Wavelet, and step function. All of the rest procedures are the same as before.

Part two :

In this part, the purpose is to find the music score. I choose the increment of the tslide to be 0.2, and parameter to be 100. Only gaussian function will be used as a filter. Also, each time through the loop, I use the MAX function to find the strongest frequency at that time. The strongest frequency is taken as the music score at that time. All of the rest procedures are the same as part one. I repeat the same procedure for both piano and recorder.

After I get the music scores, I will plot them as comparison of piano and recording.

Computational Results

Part one :

The three graphs in figure 1 are spectrograms with a = 100, 1000, and 1 respectively. As we can see, with smaller window width, the resolution in the time domain is better, but the resolution in the frequency domain is bad, corresponding to the middle graph in figure 1. With larger window width, the resolution in the time domain is bad, but the resolution in the frequency domain is better, corresponding to the right graph in figure 1. This is a tradeoff between frequency and time. The left graph in figure 1 is moderate window width.

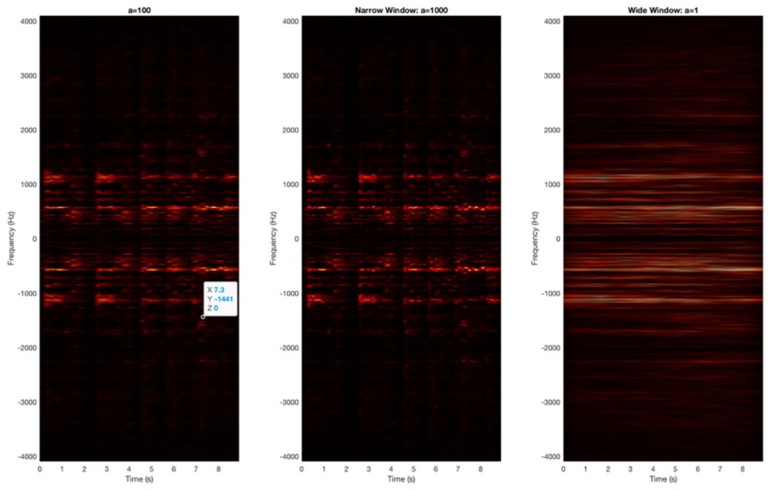

Figure 1 : comparison of window width

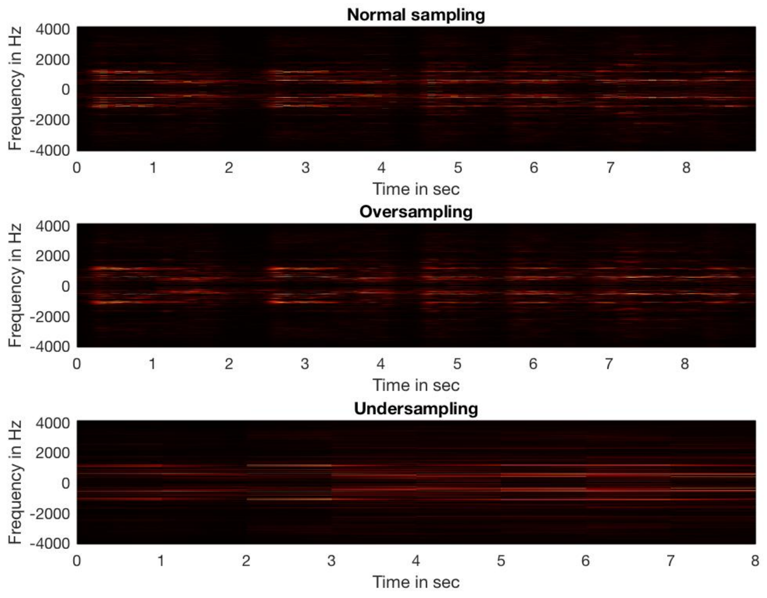

Figure 2 : comparison of oversampling and under sampling

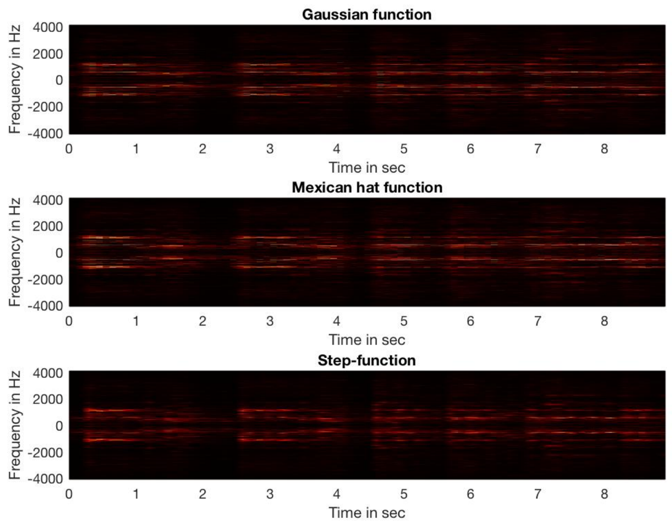

Figure 3 : comparison of different filter functions

In figure two, we can see that the graph with under sampling is not very clear, because information is not enough, and the graphs with oversampling and normal are almost identical, but it takes a long time to run the code implemented with oversampling. This may imply that over sampling does not grant many benefits, and under sampling is not a good approach.

As we can see in figure 3, the graphs produced by three different filter functions look almost the same. Therefore, I believe the type of functions used does not significantly influence the spectrogram. They are equally good in this particular case.

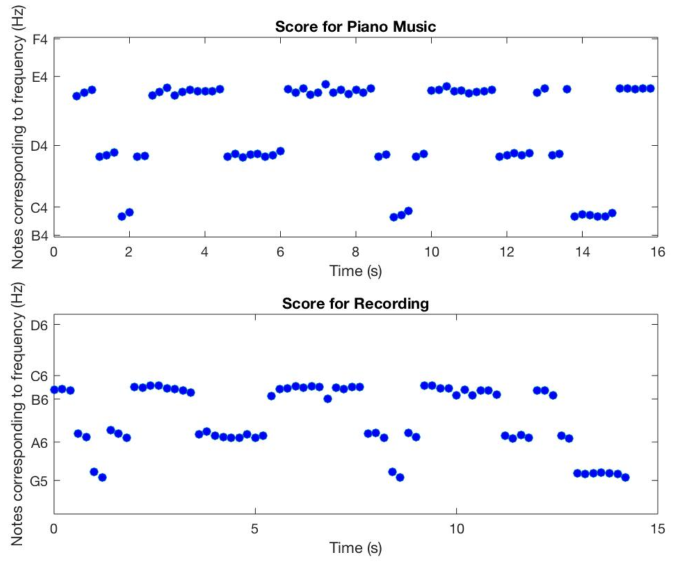

Figure 4 : music scores for piano and recordings

Part 2 :

Figure 4 is the music score for both piano and recorder. I label the graph according to frequency. As we can tell from the graph, the difference is that recorder has high frequency.

Summary and Conclusion :

This paper involves using Gabor transform to do the time frequency analysis for a given music recording. In part one, I begin by exploring how the window size, oversampling or under sampling, types of window function can influence the spectrogram. In part two, I use the plausible values I found out in part one to analyze a piece of music in piano and recording. I compare the music score of the two and find out that recording has high frequency.

Appendix A (MATLAB functions used) :

fft : gives the Fourier Transform to the given input. Dimension is 2.
fftshift : shifts the input, so that the order matches the original order.
pcolor : used to plot the spectrogram. Creates a pseudocolor plot using the values in the arguments
colormap : used to plot the spectrogram. Set the colormap for the current figure.

Appendix B(matlab code)

```
%% Part 1
clear; close all; clc
load handel  % Fs = the sampling frequency in Hz
             % y = the audio signal amplitude as a single column vector
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);

%% different window widths
L = length(v)/Fs;
n = length(v);
t2=linspace(0,L,n+1);
t=t2(1:n);
k=(1/L)*[0:(n-1)/2 -(n-1)/2:-1];
ks=fftshift(k);

a = [1 100 1000];
tslide = 0:0.1:L;
```

```
vgt_spec = zeros(length(tslide), n);
vgt_spec1 = zeros(length(tslide), n);
vglt_spec2 = zeros(length(tslide), n);
for j = 1:length(tslide)
    g = exp(-a(2)*(t-tslide(j)).^2);
    g_narroww = exp(-a(3)*(t-tslide(j)).^2);
    g_widew = exp(-a(1)*(t-tslide(j)).^2);

    vg = g.* v;
    vgt = fft(vg);
    vg1 = g_narroww .* v;
    vgt1 = fft(vg1);
    vg2 = g_widew .* v;
    vgt2 = fft(vg2);

    vgt_spec(j,:) = fftshift(abs(vgt));
    vgt_spec1(j,:) = fftshift(abs(vgt1));
    vglt_spec2(j,:) = fftshift(abs(vgt2));
end
figure(1)
subplot(1, 3, 1);
pcolor(tslide,ks,vgt_spec.'),
shading interp
title('a=100')
xlabel('Time in sec'), ylabel('Frequency in Hz')
colormap(hot)

subplot(1, 3, 2);
pcolor(tslide,ks,vgt_spec1.'),
shading interp
title('Narrow Window: a=1000')
xlabel('Time in sec)'), ylabel('Frequency in Hz')
colormap(hot)

subplot(1, 3, 3);
pcolor(tslide,ks,vglt_spec2.'),
shading interp
title('Wide Window: a=1')
xlabel('Time in sec'), ylabel('Frequency in Hz')
colormap(hot)

%% Oversampling and Undersampling
tslide1 = 0:0.01:L;  % increment  = 0.01
vgtspec1 = zeros(length(tslide1), n);
for j = 1:length(tslide1)
    g = exp(-100*(t-tslide1(j)).^2);
    vg = g.* v;
    vgt = fft(vg);
    vgtspec1(j,:) = fftshift(abs(vgt));
end
tslide2 = 0:1:L;  %  increment = 1
vgtspec2 = zeros(length(tslide2), n);
for j = 1:length(tslide2)
    g = exp(-100*(t-tslide2(j)).^2);
    vg = g.* v;
    vgt = fft(vg);
    vgtspec2(j,:) = fftshift(abs(vgt));
```

```
end
figure(2)
subplot(3,1,1)
pcolor(tslide,ks,vgt_spec.'),
shading interp
title('Normal sampling')
xlabel('Time in sec'), ylabel('Frequency in Hz')
colormap(hot)

subplot(3,1,2)
pcolor(tslide1,ks,vgtspec1.'),
shading interp
title('Oversampling')
xlabel('Time in sec'), ylabel('Frequency in Hz')
colormap(hot)

subplot(3,1,3)
pcolor(tslide2,ks,vgtspec2.'),
shading interp
title('Undersampling')
xlabel('Time in sec'), ylabel('Frequency in Hz')
colormap(hot)


%% Part I Using Different Gabor windows

vmt_spec = zeros(length(tslide), n);
vst_spec = zeros(length(tslide), n);
for j = 1:length(tslide)
    %  Mexican hat wavelet
    sigma = 0.05;
    mex_hat = (2/(sqrt(3*sigma)*(pi^0.25))).*(1-((t-tslide(j))/sigma).^2)...
        .* exp(-((t-tslide(j)).^2)/(2*sigma^2));

    % Step-function (Shannon) window
    width = 0.05;
    shn = abs(t - tslide(j)) <= width /2;

    vmh = mex_hat .* v;
    vmht = fft(vmh);

    vsh = shn .* v;
    vsht = fft(vsh);

    vmt_spec(j,:) = fftshift(abs(vmht));
    vst_spec(j,:) = fftshift(abs(vsht));
end

figure(3)
subplot(3,1,1)
pcolor(tslide,ks,vgt_spec.'),
shading interp
title('Gaussian function')
xlabel('Time in sec'), ylabel('Frequency in Hz')
colormap(hot)
```

```
subplot(3,1,2)
pcolor(tslide,ks,vmt_spec.'),
shading interp
title('Mexican hat function')
xlabel('Time in sec'), ylabel('Frequency in Hz')
colormap(hot)

subplot(3,1,3)
pcolor(tslide,ks,vst_spec.'),
shading interp
title('Step-function')
xlabel('Time in sec'), ylabel('Frequency in Hz')
colormap(hot)


%% Part II - Starter Code
clear; close all; clc

[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);


%% Part II - 1
% piano
v = y';
n = length(v);
t2 = linspace(0,tr_piano,n+1);
t_p = t2(1:n);
k_p = (2*pi/tr_piano)*[0:n/2-1 -n/2:-1];
ks_p = fftshift(k_p);

tslide_p = 0:0.2:tr_piano;
spectrogram_p = zeros(length(tslide_p), n);
notes_piano = zeros(1, length(tslide_p));
for j = 1:length(tslide_p)
    f = exp(-100*(t_p-tslide_p(j)).^2);
    window = f .* v;
    windowfft = fft(window);
    [M, I] = max(windowfft); %Find max frequency
    notes_piano(1,j) = abs(k_p(I))/(2*pi);
    spectrogram_p(j,:) = fftshift(abs(windowfft));
end

% recorder
[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)');
% p8 = audioplayer(y,Fs); playblocking(p8);

v = y';
```

```matlab
n = length(v);
t2 = linspace(0,tr_rec,n+1);
t_rec = t2(1:n);
k_rec = (2*pi/tr_rec)*[0:n/2-1 -n/2:-1];
ks_rec = fftshift(k_rec);

tslide_rec = 0:0.2:tr_rec;
spectrogram_rec = zeros(length(tslide_rec), n);
notes_rec = zeros(1, length(tslide_rec));
for j = 1:length(tslide_rec)
    f = exp(-100*(t_rec-tslide_rec(j)).^2);

    window = f .* v;
    windowfft = fft(window);
    [M, I] = max(windowfft);  %Find max frequency
    notes_rec(1,j) = abs(k_rec(I))/(2*pi);
    spectrogram_rec(j,:) = fftshift(abs(windowfft));
end


figure(4)
subplot(2,1,1)
pcolor (tslide_p,(ks_p/(2*pi)), spectrogram_p .'),
shading interp
title ("Piano")
xlabel('Time in sec'), ylabel('Frequency in Hz')
ylim ([0 400])
colormap(hot)

subplot(2,1,2)
pcolor (tslide_rec,(ks_rec/(2*pi)), spectrogram_rec .'),
shading interp
title ("Recorder")
xlabel('Time in sec'), ylabel('Frequency in Hz')
ylim ([0 1400])
colormap(hot)

%%
% Reproduce the music score/note on the piano and recorder
figure(5)
subplot(2, 1, 1)
plot(tslide_p, notes_piano,'o','MarkerFaceColor', 'b');
yticks([246.9417,261.6256,293.6648,329.6276,349.2282]);
yticklabels({'B4','C4','D4','E4','F4'});
ylim ([246 350])
title("Score for Piano Music");
xlabel("Time (s)"); ylabel("Notes corresponding to frequency (Hz)");

subplot(2, 1, 2)
plot(tslide_rec, notes_rec,'o','MarkerFaceColor', 'b');
yticks([783.99, 880, 987.77, 1046.5, 1174.7, ]);
yticklabels({'G5','A6','B6','C6','D6'});
ylim ([700 1200])
title("Score for Recording");
xlabel("Time (s)"); ylabel("Notes corresponding to frequency (Hz)");
```