Homework 2: G$\acute{a}$bor Transforms

Yilin Li

February 07 2020

Abstract

This project firstly analyzes a portion of Handel's Messiah with time-frequency analysis by using three types of wavelets. Additionally, this report applies a particular G$\acute{a}$bor Transform to two recordings of *Mary had a little lamb* on different instruments and reconstruct their musical scores using spectrograms.

Sec. I. Introduction and Overview

This project has two parts to explore the characteristics and application of the G$\acute{a}$bor Transform. In part I, we explore the time-frequency signature of a 9 second piece of Handel's *Messiah*. We produce spectrograms of it by using the G$\acute{a}$bor Transform and explore how the window width and translation will affect the results. Three types of wavelets are used in this project, Gaussian window, Mexican hat wavelet, and a step-function (Shannon) window.

In part II, we reproduce the music score for piece of *Mary had a little lamb* on both the piano and recorder by using the G$\acute{a}$bor filtering. We compare and contrast their spectrograms to see the difference between a recorder and piano.

The Theoretical Background section introduces the theorems used in the project.

The Algorithm Implementation and Development section lists the algorithms that were implemented in this project and how they were developed.

The Computational Results section shows the plots and analyzes the computational results of MATLAB.

The Summary and Conclusion section summarizes the project and makes a conclusion.

The MATLAB functions and implementations used in this project were attached in Appendix A. The MATLAB codes were attached in Appendix B.

Sec. II. Theoretical Background

G$\acute{a}$bor Transform

The Fourier Transform is define as:

$$\hat{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ikx} dx \tag{1}$$

However, the Fourier Transform cannot localize the various frequencies in time domain, the Gábor transform is developed to localize various frequencies in both time and frequency domain. The Gábor Transform is a special case of the short-time Fourier Transform. As a signal changes over time, the Gábor transform is used to determine the sinusoidal frequency and phase content of its local sections, and it is defined as:

$$\tilde{f}(\tau, \omega) = \int_{-\infty}^{\infty} f(t) g(t-\tau) e^{-i\omega t} dt \tag{2}$$

The time window is centered on τ with the width α, which controls the translation and the scaling of the window. The information outside the time window is severely damped. To sum up, the Gábor transform gains the signal information in both time and frequency domain. The target function is firstly multiplied by a Gaussian function, a window function, and the resulting function is then transformed with a Fourier transform to derive the time-frequency analysis. The following are the three types of wavelets that are used in this project:

Gaussian window

$$g(t-\tau) = e^{-\alpha(t-\tau)^2} \tag{3}$$

$\alpha-$ the width of window, $\tau-$ the center of the window.
Due to the negative sign before α, the larger α relates to the narrower window width.

Mexican hat wavelet

$$\psi(t-\tau) = (1 - (\frac{t-\tau}{\alpha})^2) e^{-\frac{(t-\tau)^2}{2\alpha^2}} \tag{4}$$

$\alpha-$ the width of window, $\tau-$ the center of the window.
The smaller α relates to the narrower window width.

Step-function (Shannon) window

$$g(t-\tau) = \begin{cases} 0 & |t-\tau| > \frac{1}{2}\alpha & (5) \\ 1 & |t-\tau| \le \frac{1}{2}\alpha & (6) \end{cases}$$

$\alpha-$ the width of window, $\tau-$ the center of the window.
A step function with value of unity within the transmitted band and 0 outside of it, so it simply suppresses all frequencies outside of a given filter box.

Spectrogram

A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time, and it is usually depicted as a heat map, i.e., as an image with the intensity shown by varying the colour or brightness. Therefore, it shows what frequency is prominent at the given time span.

Sec. III. Algorithm Implementation and Development

Part I

- First of all, load handle, which is a portion of Handel's Messiah;
- Transpose the audio vector since it is in a column;
- Define the length of the piece L, and the Fourier modes n. Since the audio file produces a vector with an odd number of entries, in order to produce the right number of frequency components, we do not take the last value of the vector, and we can do this because it is periodic, the last value is just as same as the first entry which is zero;
- Define the domain discretization and grid vector;
  - The grid vectors for the frequency domain must be multiplied by $\frac{2\pi}{L}$ because the FFT algorithm assumes 2π period signals;
  - The grid vectors k must be shifted by $fftshift$ function because the FFT algorithm shifts x $\in [-L, 0] \rightarrow [0, L]$ and x $\in [0, L] \rightarrow [-L, 0]$.
- Set the window width and the time sliding for the sliding window;
- Create a spectrogram matrix,vgt_spec, to store the frequency information for the spectrogram to be constructed later;
- Create a $for\ loop$ to allow the wavelet to extract the frequencies information centered at time stops, in the $for\ loop$:
  - Define the function for different translating windows g (Comment others when using one of them);
  - Apply the filter by multiplying the audio signal by the translating window function;
  - Take fft of filtered data;
  - Take the absolute and fftshifted value of the resulting vector and add it to vgt_spec;
- Plot the process of the accomplishment of the G*á*bor transform
- Plot the spectrogram using the *pcolor* function resulting from vgt_spec that was created before with time on x-axis and frequencies on the y-axis

Part II

- Read in the audio files;
- Record time in seconds;
- Transpose the audio vector since it is in a column;
- The rest steps are the same as in Part I, except that in Part II we only use the standard Gaussian G*á*bor Window.

Sec. IV. Computational Results

Part I

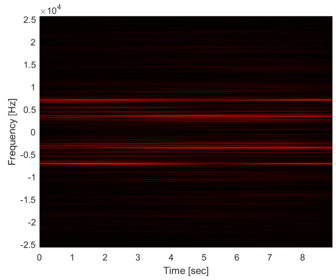

Figure 1: Spectrogram under Gaussian filter with $\alpha = 0.2$

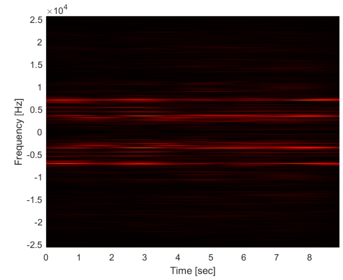

Figure 2: Spectrogram under Gaussian filter with $\alpha = 1$

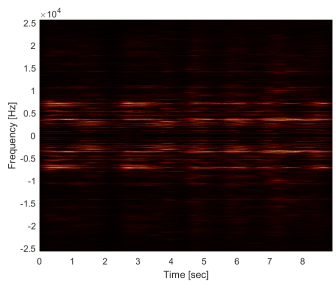

Figure 3: Spectrogram under Gaussian filter with $\alpha = 20$

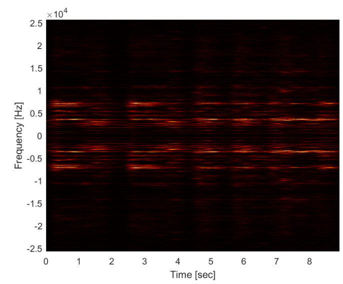

Figure 4: Spectrogram under Gaussian filter with $\alpha = 50$

Start with the Gaussian filter and width $\alpha = 1$ and translation $dt = 0.1$ (Figure 2). To compare, I generated the spectrogram with $\alpha = 0.2, 1, 20$, and 50. The Figure 1 gives us pretty good resolution in frequency but almost no information in time. The Figure 4 has good resolution in time but trade off the frequency information. Over all, $\alpha = 20$ looks like a more resonable parameter for width of window with translation 0.1.

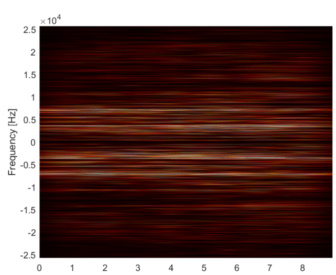

Figure 5: Spectrogram under Gaussian filter with $dt = 0.01$

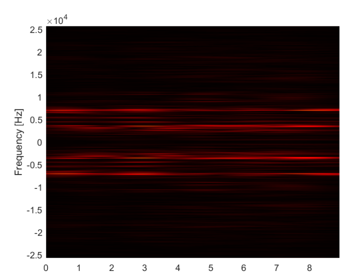

Figure 6: Spectrogram under Gaussian filter with $dt = 0.1$

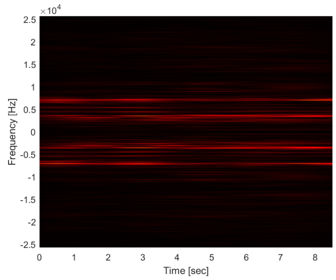

Figure 7: Spectrogram under Gaussian filter with $dt = 0.5$

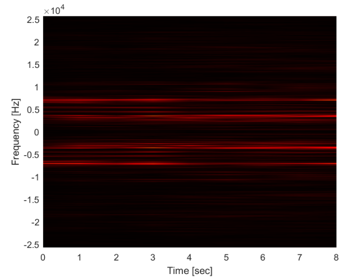

Figure 8: Spectrogram under Gaussian filter with $dt = 1$

To compare the translation, I generated the spectrogram with $dt = 0.01, 0.1, 0.5$, and 1 with $\alpha = 1$. In Figure 5, $dt = 0.01$, this is the case called "oversampling", and it gives us pretty good resolution in frequency but almost no information in time. In Figure 8, $dt = 1$, this is the case called "undersampling", it gives us bad resolution on both time and frequency because compare to the width of window $\alpha = 1$, the translation is too large.

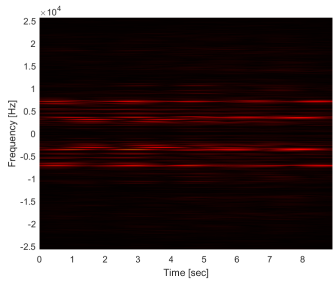

Figure 9: Spectrogram under Mexican Hat Wavelet

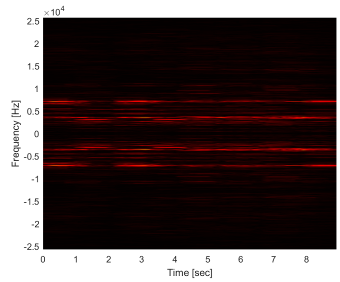

Figure 10: Spectrogram under Shannon Wavelet

Figure 9 is the Spectrogram under Mexican Hat Wavelet. Figure 10 is it under Shannon Wavelet. With the translation $dt = 0.1$, as the width α decreases, the resolution in frequency become better and better, which is better than Gaussian filter. Under the same width and translation, Shannon Wavelet seems also better than Gaussian filter.

Part II

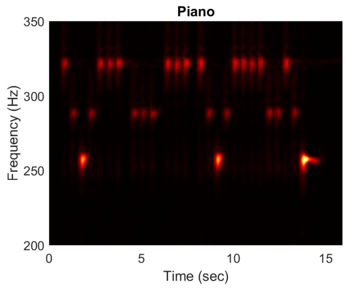

Figure 11: Portion of Spectrogram for the music on piano

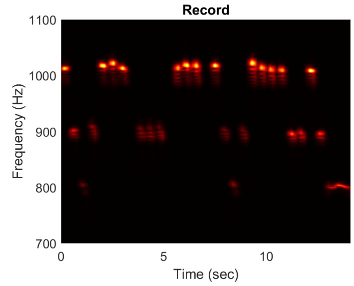

Figure 12: Portion of Spectrogram for the music on recorder

The Figure 11 and Figure 12 gives portion of the Spectrogram for the music on piano and recorder with width $\alpha = 40$ and translation $dt = 0.15$ (The whole spectrogram is in Appendix B). From two figures, we can see that the fundamental tone are around 200 - 350 for piano and 700 - 1100 for recorder.

The notes for piano are:
$320Hz, 280Hz, 260Hz, 280Hz, 320Hz, 320Hz, 320Hz, 280Hz,$
$280Hz, 280Hz, 320Hz, 320Hz, 320Hz, 320Hz, 280Hz, 260Hz, 280Hz, 320Hz, 320Hz,$
$320Hz, 320Hz, 280Hz, 280Hz, 320Hz, 280Hz, 260Hz$

Compare to the music scale in Herz, we get the music score for piano: 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

For the recorder: 1030Hz, 920Hz, 820Hz, 920Hz,1030Hz,1030Hz,1030Hz,920Hz,920Hz, 920Hz, 1030Hz,1030Hz,1030Hz,1030Hz,920Hz,820Hz,920Hz,1030Hz,1030Hz,1030Hz,1030Hz, 920Hz,920Hz,1030Hz,920Hz,820Hz

The music score for recorder: 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

Sec. V. Summary and Conclusions

To sum up, the reasonable translation and width of window based on the data length can give us a better resolution of frequency and time. Additionally, different types of wavelets are also worth to consider.

From the plots, we can see that the difference between different instruments embodies on the intensity of overtones (the brightness on that frequency). At the same frequency with same volume, different instruments have different intensity of overtones.

Appendix A.

MATLAB functions and implementation

- **plot(X,Y)** creates a 2-D line plot of the data in Y versus the corresponding values in X.

- **audioplayer(Y,Fs)** creates an audioplayer object for signal Y, using sample rate Fs. The function returns the audio player object, player.
- **audioread(filename)** reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.
- **playblocking(playerObj)** plays the audio associated with audioplayer object playerObj from beginning to end. playblocking does not return control until playback completes.
- **fft(X)** computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm.
- **fftshift(X)** rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.
- **abs(X)** returns the absolute value of each element in array X.
- **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.
- **colormap(map)** sets the colormap for the current figure to the colormap specified by map.

Appendix B. MATLAB codes

Part I

```
% Handel's Messiah
clear; close all; clc;

load handel
v = y';

%% Plot the portion of music I will analyze.
% figure(1)
% plot((1:length(v))/Fs,v);
% xlabel('Time [sec]');
% ylabel('Amplitude');
% title('Signal of Interest, v(n)');

%% To play this back in MATLAB:
% p8 = audioplayer(v,Fs);
% playblocking(p8);

```

```
L = (length(v)-1) / Fs; % Length of the piece
% Identify first and last points of v
v = v(1:end-1); % periodic
n = length(v);
t = (1:n)/ Fs;
k = (2*pi/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

%% Plot the portion of music in freq domain
% figure(2)
% vt = fft(v);
% vt_shift = fftshift(vt);
% plot(ks,abs(vt_shift)/max(abs(vt)));
% xlabel('Freq [\omega]');
% ylabel('Amplitude');
% title('Signal of Interest in Freq Domain');

% Use Gabor filter produce spectrograms of the piece of work
% Set window width
a = 1;
% a = 0.2;
% a = 20;
% a = 50;

% Different translations
dt = 0.1;
% dt = 0.01;
% dt = 0.5;
% dt = 1;
tslide = 0:dt:L; % Set time sliding
vgt_spec = [];
for j = 1:length(tslide)
    %% Gaussian Window
    % g = exp(-a*(t-tslide(j)).^2);
    %% Mexican Hat Wavelet
    % g = (1-((t-tslide(j))/a).^2).*exp(-(((t-tslide(j))/a).^2)/2);
    %% Step-function (Shannon) Window
    g = (abs(t-tslide(j)) < a/2);
    vg = g.*v;  % Apply filter
    vgt = fft(vg);  % Take fft of filtered data
    vgt_spec(j,:) = fftshift(abs(vgt)); % Store fft in spectrogram

    % Plot the process of the accomplishment of the Gabor transform
```

```
%      subplot(3,1,1), plot(t,v,'k',t,g,'r')
%      xlabel('Time (sec)'), ylabel('Amplitude')
%      title('Gabor Filter and Signal')
%      axis([-50 50 0 1])
%      subplot(3,1,2), plot(t,vg,'k')
%      xlabel('Time (sec)'), ylabel('Amplitude')
%      title('Gabor Filter * Signal')
%      axis([-50 50 0 1])
%      subplot(3,1,3), plot(t,abs(fftshift(vgt))/max(abs(vgt)),'k')
%      xlabel('Time (sec)'), ylabel('Amplitude')
%      title('Gabor Transform of Signal')
%      axis([-50 50 0 1])
%     drawnow
%      pause(0.1)
end
vgt_spec;
figure()
pcolor(tslide,ks,vgt_spec.'),shading interp
colormap(hot)
xlabel('Time [sec]'),ylabel('Frequency [Hz]')
```

Part II

```
% Mary had a little lamb - Piano
clear; close all; clc;

% [y,Fs] = audioread('music1.wav');
[y,Fs] = audioread('music2.wav');


tr_piano = length(y)/Fs;  % record time in seconds
% figure()
% plot((1:length(y))/Fs,y);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title('Mary had a little lamb (piano)');
%% title('Mary had a little lamb (recorder)');
% p8 = audioplayer(y,Fs); playblocking(p8);

y = y.';
L = tr_piano;
n = length(y);
t = (1:n)/Fs;
k = (2*pi/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

```

```matlab

%% Plot the portion of music in freq domain
% figure()
% yt = fft(y);
% yt_shift = fftshift(yt);
% plot(ks,abs(yt_shift)/max(abs(yt)));
% xlabel('Freq [\omega]');
% ylabel('Amplitude');
% title('Mary had a little lamb (piano) in Freq Domain');
%% title('Mary had a little lamb (recorder) in Freq Domain');

% Use Gabor filter produce spectrograms of the piece of work
a = 40; % Set window width
tslide = 0:0.15:L; % Set time sliding
ygt_spec = [];
for j = 1:length(tslide)
    g = exp(-a*(t-tslide(j)).^2);
    yg = g.*y;  % Apply filter
    ygt = fft(yg);  % Take fft of filtered data
    ygt_spec(j,:) = fftshift(abs(ygt)); % Store fft in spectrogram

    % Plot the process of the accomplishment of the Gabor
      transform
%     subplot(3,1,1), plot(t,v,'k',t,g,'r')
%     xlabel('Time (sec)'), ylabel('Amplitude')
%     title('Gabor Filter and Signal')
%     axis([-50 50 0 1])
%     subplot(3,1,2), plot(t,vg,'k')
%     xlabel('Time (sec)'), ylabel('Amplitude')
%     title('Gabor Filter * Signal')
%     axis([-50 50 0 1])
%     subplot(3,1,3), plot(t,abs(fftshift(vgt))/max(abs(vgt)),'k')
%     xlabel('Time (sec)'), ylabel('Amplitude')
%     title('Gabor Transform of Signal')
%     axis([-50 50 0 1])
%     drawnow
%     pause(0.1)
end
ygt_spec;

% Plot portion of spectrogram
figure()
pcolor(tslide,ks/(2*pi),ygt_spec.'),shading interp
% set(gca,'Ylim',[200 350],'Fontsize',[14])
set(gca,'Ylim',[700,1100],'Fontsize',[14])
```

```
xlabel('Time (sec)'), ylabel('Frequency (Hz)');
% title('Piano');
title('Record');
colormap(hot)
%
%% Plot full of spectrogram
% figure()
% pcolor(tslide,ks/(2*pi),ygt_spec.'),shading interp
% set(gca,'Ylim',[200 1500],'Fontsize',[14])
% xlabel('Time (sec)'), ylabel('Frequency (Hz)');
% colormap(hot)
```

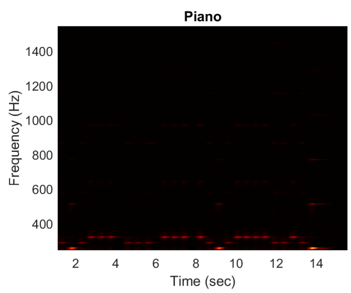

Figure 13: Full Spectrogram for the music on piano

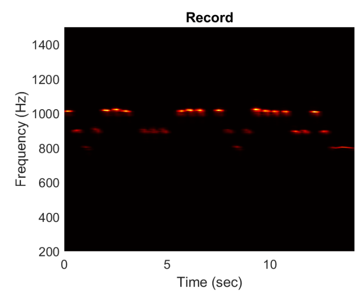

Figure 14: Full Spectrogram for the music on recorder