Gabor Transforms & Signal Processing

W. Hoyt Thomas

02/07/2020

I. Abstract

The time-frequency analysis of three different musical pieces is performed for two different goals. First a portion of Handel's messiah is analyzed to explore how the Gabor Transform works. Secondly, the Gabor Transformed is used to create the musical score of recordings of '*Mary had a little lamb*' on piano and recorder.

II. Introduction

Time-frequency analysis is a tool employed to understand the frequency signatures of signals and where they occur in time; one of the most common signals used for analysis is musical recordings. To start, since a portion of Handel's legendary *Messiah* is conveniently pre-loaded into Matlab, I will use it to demonstrate the uses and limitations of a specific type time-frequency analysis: Gabor Transforms.

Then I will demonstrate how to produce the musical scores when provided with recordings. In this case, I used recordings of the classic tune, '*Mary had a little lamb*,' played on two different instruments: the piano and recorder.

III. Theoretical Background

The Gabor Transform or Short Time Fourier Transform (STFT) provided the theoretical background of the analysis performed in the code. The Fast Fourier Transform (FFT), while powerful, is limited when used the deconstruct non-stationary signals (average of signal changes with time). The FFT will not provide any information for the location of the frequencies in time. The STFT remedied this by creating 'windows' in the time domain, where only the signal inside is considered. Within each window the FFT is applied, then the window is slid across the signal to capture the entire range of frequencies in the signal. The STFT is defined as:

$$\tilde{f}(\tau,\omega) = \int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-i\omega t}dt \text{ (EQ. 1)}$$

where f(t) is the signal, g(t-τ) is the filter function, and $e^{-i\omega t}$ is the Fourier transform. The filter function creates the window for the FFT to be performed, a real and symmetric function is needed for the filter. The most common filter used is a Gaussian filter:

$$g(t-\tau) = e^{-a(t-\tau)} \text{ (EQ. 2)}$$

where *a* controls the width of the gaussian window and τ controls where the center of gaussian window is in time. Unlike spectral filtering, this filter is applied to the signal in the time the STFT filters the signal in the time domain.

IV. Algorithm Development

<u>Part I</u>

A portion of Handel's Messiah is pre-loaded into Matlab, so the signal (vector) and sampling frequency (variable) were extracted from it. The signal [73113x1] had to be transposed into a row vector to match up with the time vector. A frequency vector [1x73113] was also created to match the dimensions of the signal, scaled by (1/L) so the graphed frequencies are in units of Hertz. Then the signal was plotted to visualize the signal graph the signal and played to hear the signal.

Before creating the spectrogram, the width (a) of the gaussian filter (EQ1) had the be set. To shift the center gaussian filter, a vector ('tslide') for tau is needed from zero to the length of the score with the length of the shift (delta tau) also defined. An empty matrix(vgt_spec) was also needed to pass each iteration of FFT to, this was created with zeros() to save space and time. The spectrogram was created by first, defining the gaussian filter multiply it by the signal, then passing it on to the FFT command and storing the inverse of it in the empty matrix. This was repeated for as many shifts as defined earlier. The full spectrogram was plotted with pcolor(), using the frequencies and tslide as the domain and range. To demonstrate the effect of the filter width, spectrograms were created at three different widths (Fig. 1). To demonstrate effect of shift length, spectrograms were created at three different shift lengths (Fig. 2).

Part II

The first step was to load the recordings of '*Mary had a little lamb,*' then extract the signal and sampling frequency. As in part 1, the signal had to be transposed and corresponding time and frequency vectors had to be created. The signal was plotted and played to understand how many notes should appear in the final spectrogram. A spectrogram of the recording was created using the same method as in part 1.

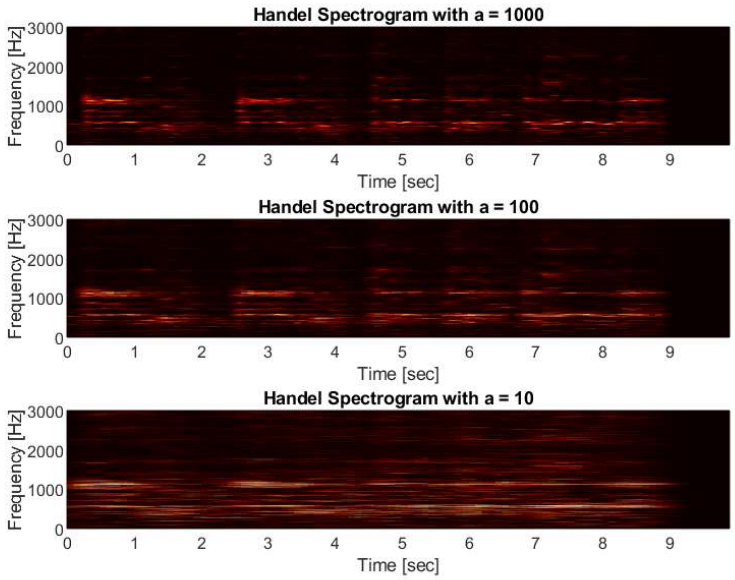

Figure 1: Spectrogram of *Messiah* with varying widths of gaussian filter

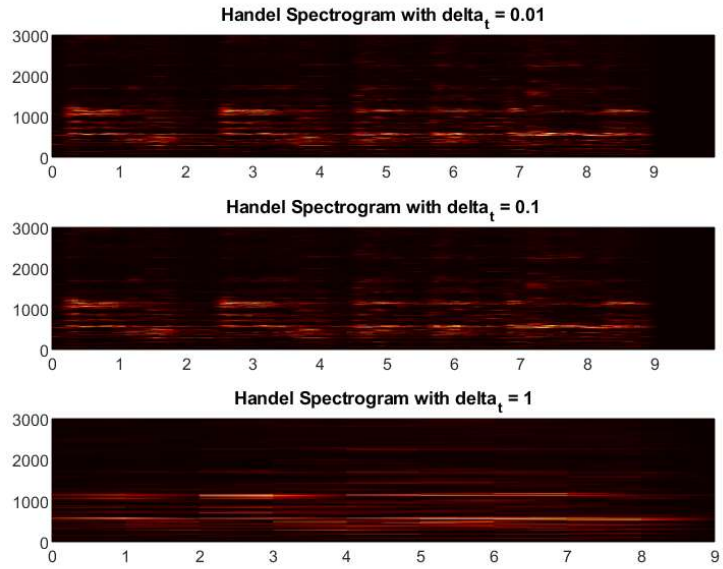

Figure 2: Spectrogram of Messiah with varying shift lengths (delta tau)

Different widths and shift length were tested until it was clear what frequency notes were and when they were being played. To show the score of the music, zooming in to the fundamental frequency was required, in addition, horizontal lines drawn through the approximate frequency of the note. This process was done for the piano (Fig. 3) and recorder (Fig. 4).

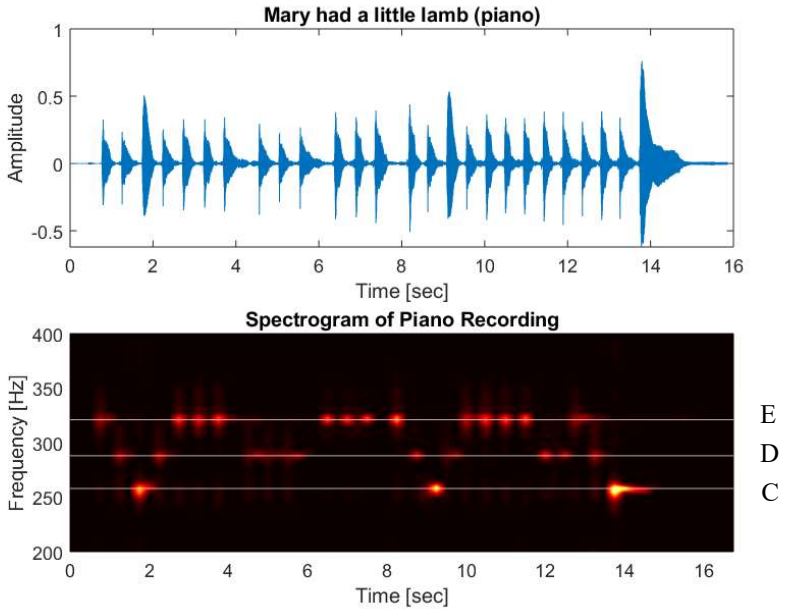

Figure 3: Piano score of Mary had a little lamb with frequencies of notes. Approximate values are C = 258 Hz, D = 288 Hz, E = 321 Hz

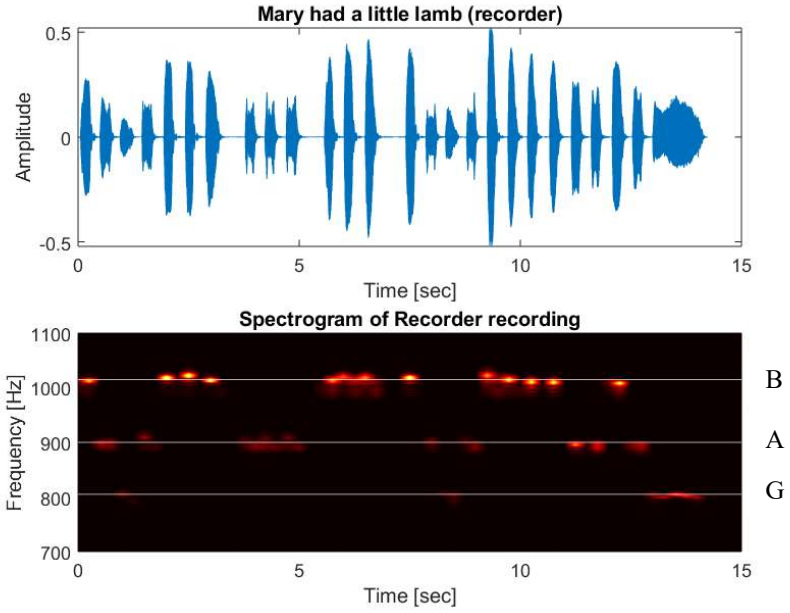

Figure 4: Recorder score of Mary had a little lamb with frequencies of notes. Approximate values are G = 805 Hz, A = 900 Hz, B = 1015 Hz

V. Computational Results

Part I

Figure 1 shows the effect the width has on the spectrograms created by the STFT, using a larger value for 'a' will cause the gaussian filter to be narrower. An infinitely thin gaussian filter would only convey information in the temporal domain since no periodic behavior could be observed in the theoretically thin window. Whereas an increasingly wide filter will convey more information on the frequencies in the signal, but at the cost of temporal definition. When the width equals the length of the signal, the result is a regular FFT and no information of the frequencies in time can be decerned. The too narrower filter is demonstrated in Fig. 1a, the frequency is well defined in time, but getting stretched in the vertical directions. The too wide filter is demonstrated in Fig. 1c, the frequency values are easier to determine, but when the occur gets much harder to tell.

Figure 2 shows the effect shifting delta tau values. By increasing it, the center of the gaussian window is moved further and there is less resulting over-lap. This is known as under-sampling, since some information will be lost in between the windows. Decreasing the delta-tau value will increase the amount of overlap of each gaussian window. As the overlap gets increase, computing the final spectrogram will get harder since more FFTs are being performed. This is known as oversampling and is reflected in Fig. 2a which captured high-frequency portions but the computing time took the longest. Under-sampling is reflected in Fig. 2c, the principle frequencies in each window are stretched over the entire window, then abruptly is cut off at the window's edge.

Part II

Figure 3 shows the score of the piano recording, with the signal plotted above the spectrogram to show the length of each of the notes.

Figure 4 shows the score of the recorder recording, with the signal plotted above the spectrogram to show the length of each note.

VI. Conclusion

When using the STFT, a balance must be achieved between the width and overlap of the window. It is best to test different values for each to see what frequency values are and when they occur.

The scores of the music were made more readable by zooming in on the fundamental frequency of the signal, but for both the piano and the recorder overtones of the fundamental frequencies occurred at integer multiples. However the piano produced much stronger overtones than the recorder, there was greater spectral density at the piano overtones than the recorder overtones.

Appendix A - Matlab Functions

pcolor(X,Y,C) – displays matrix data as an array of colored cells on an x-y plane

input arguments- C – contains the indices of the colormap X,Y – coordinates that match the dimensions of C

Appendix B – Matlab Code

```
%Hoyt Thomas
%AMATH 482 HW 2
%2/07/2020
%Part I
clear all; close all; clc
load handel
v = y';
n = 73113; L = length(v)/Fs;
k=(1/L)*[0:(n-1)/2 -(n-1)/2:-1];
ks=fftshift(k);
t2=linspace(0,L,n+1); t=t2(1:n);

%plot signal
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');

playback commands
p8 = audioplayer(v,Fs);
playblocking(p8);
%% Intial spectrogram
```

```
a = 100;
tslide=0:0.1:L+1;
vgt_spec = zeros(length(tslide),n);
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    vg=g.*v;
    vgt=fftn(vg);
    vgt_spec(j,:) = fftshift(abs(vgt));
end


pcolor(tslide,ks,vgt_spec.'),
shading interp
set(gca,'Ylim',[0 3000])
colormap(hot)


%% Width Comparsion
a_vec = [1000 100 10];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tslide=0:0.1:L+1;
    vgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        vg=g.*v;
        Sgt=fft(vg);
        vgt_spec(j,:) = fftshift(abs(Sgt));
    end

    subplot(3,1,jj)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['Handel Spectrogram with a = ',num2str(a)],'Fontsize',8)
    set(gca,'Ylim',[0 3000],'Fontsize',8)
    colormap(hot)
    xlabel('Time [sec]');
    ylabel('Frequency [Hz]');
end
%% delta t Comparsion
t_vec = [0.01 0.1 1.0];
for jj = 1:length(t_vec)
    a = 100;
    tslide=0:t_vec(jj):L+1;
    vgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        vg=g.*v;
        Sgt=fft(vg);
        vgt_spec(j,:) = fftshift(abs(Sgt));
    end
```

```
    subplot(3,1,jj)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['Handel Spectrogram with delta_t =
',num2str(t_vec(jj))],'Fontsize',8)
    set(gca,'Ylim',[0 3000],'Fontsize',8)
    colormap(hot)
end


%Part II
close all; clear all; clc;
%Piano
[y,Fs] = audioread('music1.wav');
v = y';
tr_piano=length(y)/Fs; % record time in seconds
L=tr_piano; n=length(y);
t2=linspace(0,L,n+1); t=t2(1:n);
k=(1/L)*[0:n/2-1 -n/2:-1]; %(1/L) because we want Hz
ks=fftshift(k);

plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (piano)');
p8 = audioplayer(y,Fs); playblocking(p8);
%% Spectrogram
a = 100;
tslide=0:0.25:L+1;
vgt_spec = zeros(length(tslide),n);
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    vg=v.*g;
    vgt=fft(vg);
    vgt_spec(j,:) = fftshift(abs(vgt)); % We don't want to scale it
end

subplot(2,1,1)
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (piano)');
subplot(2,1,2)
pcolor(tslide,ks,vgt_spec.');
shading interp;
colormap(hot);
set(gca,'Ylim', [200 400]); %200 400
xlabel('Time [sec]'); ylabel('Frequency [Hz]');
title('Spectrogram of Piano Recording');
yline(321,'w'); %E note
yline(288,'w'); %D note
yline(258, 'w'); %C note
%% recorder
clear all; close all; clc;
```

```
[y,Fs] = audioread('music2.wav');
tr_rec=length(y)/Fs; % record time in seconds
u = y';
L=tr_rec; n=length(y);
t2=linspace(0,L,n+1); t=t2(1:n);
k=(1/L)*[0:n/2-1 -n/2:-1]; %(1/L) because we want hz
ks=fftshift(k);

plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)');
p8 = audioplayer(y,Fs); playblocking(p8);

%% Spectrogram
a = 100;
tslide=0:0.25:L+1;
ugt_spec = zeros(length(tslide),n);
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    ug=u.*g;
    ugt=fft(ug);
    ugt_spec(j,:) = fftshift(abs(ugt)); % We don't want to scale it
end

subplot(2,1,1);
plot((1:length(y))/Fs,y);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)');
subplot(2,1,2);
pcolor(tslide,ks,ugt_spec.');
shading interp;
colormap(hot);
set(gca,'Ylim', [700 1100]);
xlabel('Time [sec]'); ylabel('Frequency [Hz]');
title('Spectrogram of Recorder recording');
yline(1015,'w'); %B note
yline(900,'w'); %A note
yline(805, 'w'); % G note
```