Gábor Transforms and Spectrograms:
Now That Rings a Bell (Curve)

Tim Ferrin

February 7, 2020

Abstract

This report involves analyzing three different recordings of music. The first is a recording of a snippet of George Frideric Handel's *Messiah*. The second and third are personal recordings of *Mary Had a Little Lamb* on a piano and recorder. *Messiah* is analyzed with a variety of filters and alterations on the filters in order to explore the different effects these combinations can have when analyzing a complex piece of music. The recordings of *Mary Had a Little Lamb* are analyzed and filtered for overtones in order to produce a sort of musical score for the tune.

I Introduction and Overview

The previous report concerned a specific application of the Fourier Transform and filtering out noise. This report explores the weaknesses of that data analysis tool and some alternatives to supplement those weaknesses. An in-depth coverage of the differences between the two methods of this report and the last report are covered in the Theoretical Background section. Essentially, there are many trade-offs when using Fourier analysis which must be delicately balanced. An extension on the Fourier transform is explored in this report and visualized through the use of spectrograms (sort of waterfall plots that show the strengths of frequencies in a signal over time).

Two different pieces of music will be analyzed, George Frideric Handel's *Messiah* and *Mary Had a Little Lamb*. A small section of *Messiah* is analyzed using different size and shape windows and a recording of *Mary Had a Little Lamb* on the piano is compared to a recording of it on the recorder (recorder^2!). The signal data with Gábor windows (explained in Theoretical Background) shown in Figure 1).

As a disclaimer, my computer was not able to produce many of the plots or if it could would take tens of minutes or crash my computer. As a result I could not complete several plots I hd coded.

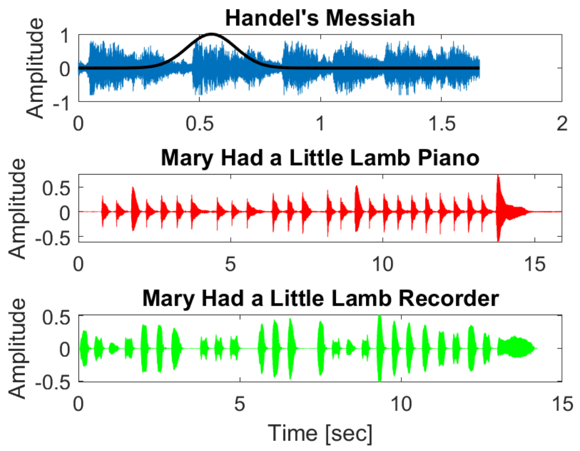

Figure 1: The three vectors of data analyzed in this report plotted vs time. An example Gábor window is shown on the Messiah graph, see Theoretical Background for its use.

II Theoretical Background

This report examines the pros and cons of using different methods for analyzing time-frequency data. It builds on the background of Fourier Analysis discussed in the corresponding section of the previous report, "Fourier Analysis of Ultrasound Data: It's a Dog-Eat-Marble World". Please refer to that report for any questions regarding the basics of the Fourier transform, Fourier series, and data filtering using the two aforementioned concepts.

As mentioned before, taking the fast Fourier transform of a data set takes the data from the strength of a signal in the spatial domain to the strength of the frequencies of sine and cosine waves in that signal. This sort of analysis is great for what are known as stationary signals, or signals that do not change over time. Other data sets, such as recordings of songs, however, consist of signals whose frequencies change over time. While this method reveals all the frequency information you could want, it tells you absolutely nothing about where each frequency occurs in the time period covered by the data. If we want to know what frequencies (notes) are played at different points in a song, we will have to change how we approach the problem.

For this problem, we will have to use what is referred to as the short time Fourier transform. The basic concept is that in order to receive some time information from the signal while also retaining frequency information, we will use a "window" function to only examine a certain time interval of the signal. The Fourier transform of this windowed data is then computed and stored. The window is then shifted to the right slightly and the frequency information for this new time interval is obtained. Each set of data is then plotted side by side in order to compare the relative strengths of the different frequencies as the signal progresses. This window function can take the form of many different shapes, although a common window is a standard Gaussian curve. Other curves exist, and common choices have the properties (1) $\|g\|_2 = 1$, and (2) g is real and symmetric, where g is the window curve. Three potential windows are shown in Figure (1)

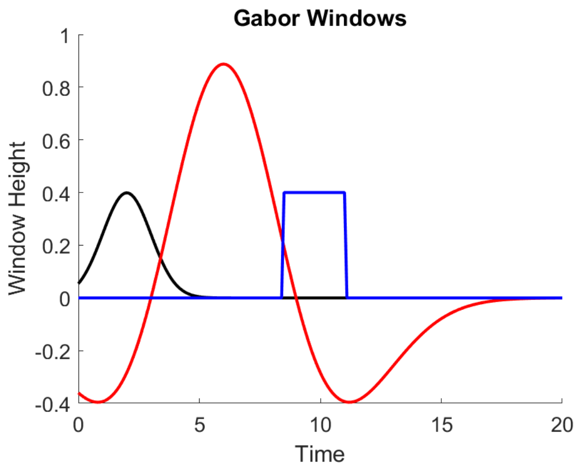

Figure 2: Some potential options for choice of Gábor window. Different windows produce different effects, so the optimal one varies from application to application. Different width and translation parameters are used.

The short time Fourier transform is also referred to as the Gábor transform and, it should be noted, is very similar to the regular Fourier transform and is also linear. Recall that the equation for the (continuous) Fourier transform for a signal $f(x)$ is

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

The Gábor transform and inverse Gábor transform formula is now

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

$$f(t) = \frac{1}{2\pi} \cdot \frac{1}{\|g\|_2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \tilde{f}(\tau, \omega) g(t-\tau) e^{i\omega\tau} d\omega d\tau \tag{3}$$

Where

$$\|g\|_2 = \sqrt{\int_{-\infty}^{\infty} |g|^2 dt} \tag{4}$$

and τ is simply a shift parameter from the center frequency.

One drawback of using this strategy is that a window can only detect frequencies whose wavelengths are less than or equal to its width. So as we decrease the width of our window we get more information about where the frequencies occur in time, but less information about what those frequencies actually are. This is a result/alternate statement of Heisenberg's Uncertainty Principle, which in simplified terms establishes a hard limit on the precision of the prediction of a particle's position and momentum, and other pairs of complimentary variables. In the realm of data analysis, this means that the more we know about time, the less we know about frequency, and vice versa.

Another drawback to using the Gábor Transform is that since we are dealing with real world data, we cannot use the continuous form of the transform and must instead use the discrete form. A consequence of this is that we cannot shift our window an infinitesimal amount, we must use some finite $\Delta\tau$. If we use a large $\Delta\tau$ relative to the window width, we are missing information in between windows–this is called undersampling. On the other side, if we use a small $\Delta\tau$ relative to the window width, we are receiving redundant information in the overlap of the windows–this is called oversampling. Neither option is particularly desirable so a happy medium is sought.

III Algorithm Implementation and Development

Part 1:

The first half of the report concerns analyzing Handel's *Messiah*. Three different effects are studied: The width of the Gábor window, the $\Delta\tau$ or the amount of overlap between shifts, and finally the different shape of the Gábor window function. In order to produce the spectrogram for the given data set, the setup is the same as with the Fourier analysis and the window is applied in the same way as the filter was in the last report. Except now since we are discussing music, the factor of 2π is taken away from the k vectors to give us frequencies in Hz.

To produce differing window sizes, a vector of desired sizes is established and a for loop works through each size window. For each iteration through, a window size is selected and the window is translated horizontally by each individual time in tslide (as seen in Appendix B). Then the data is multiplied by the filter, the FFT is taken, and finally all the data is plotted side by side in a pcolor map, showing the relative strenghts of the frequencies over time. This process can be repeated with different window function shapes, and a shape of Mexican Hat and Step filter was used and compared in the Computational Results section.

To produce oversampling or undersampling, the only thing that changes in the code is the middle number in the tslide specification. This is essentially the $\Delta\tau$ that discretizes time into manageable chunks. As discussed in the Theoretical Background, the smaller the $\Delta\tau$ is relative to the window width, the more overlap and redundant information the process is obtaining. Alternatively, the greater the $\Delta\tau$ is relative to the window width, the less overlap and more information could potentially be lost.

Part 2:

The setup for the analysis of the recordings of *Mary Had a Little Lamb* is the exact same as in Part 1. An analysis of the FFT of the data (See Figure 1) and listening to the actual song, the reader/listener will see/hear that there are only 3 notes ever played. We would like to find out what those notes are. In order to do so, we use the FFT to approximate the ranges where the three spikes occur. We then select the transformed data on those three intervals, find the max the absolute value of the FFT on each interval, and index that max to find the actual frequencies.

After that, we produce spectrograms for the data to see a type of score, where we can clearly see in Figure 4 that there are again only 3 notes played throughout the piece. The noise in the spectrogram above the data corresponds to what are called overtones. These are scalar multiples of the the resonant frequencies at which the instrument plays. These are also known as octaves. Each different instrument produces a unique signature of overtones and thus sound different even when playing the same frequency note.

IV Computational Results

Part 1:

The most computational evidence we can point to in this case is really just comparing the graphs of our spectrograms. As this recording encompasses a large band with many instruments and singers, we do not expect to see the clean sort of notes that we will see in Part 2. However, a comparison of different window sizes for the Gaussian and Mexican Hat filters can be seen in Figures 3 and 4.

Part 2:

For this section we were supposed to reproduce a score for each of the recordings of *Mary Had a Little Lamb*. The three notes printed out by the code in Appendix B for the piano are about 261, 293, and 329 Hz, which correspond to middle C and the D and E above it. For the recorder they are Images of their scores can be seen below in Figures something and something.

V Summary and Conclusions

As this report was mostly exploratory and not set to compute some sort of end goal, there is not much of a conclusion to make, other than the fact that the tools we use are extremely powerful. Through this report we have analyzed two different pieces of music and discussed the pros and cons of both the original Fourier transform and alterations/extensions of it. Exploring the effects that minute changes can have on a data set is extremely interesting and is even more applicable. Even though these methods have been known for many years, they still provide an important foundation for data analysis in our world today.

Appendix A: MATLAB Functions Used and Brief Implementation Explanation

fft(data) – Performs the fast Fourier transform, transforming data from the time domain to the frequency domain.

pcolor(x,y,data) – Produces a color plot of the data at the x and y positions. Used for "homemade" spectrograms.

spectrogram(x) – Plots the short-time Fourier transform of the input signal, x.

Appendix B: MATLAB Codes

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

%p8 = audioplayer(v,Fs);
%playblocking(p8);

L = length(S)/Fs; n = length(S);
t2=linspace(0,L,n+1);t=t2(1:n);
k=(1/L)*[0:ceil(n/2-1) ceil(-n/2):-1];
ks=fftshift(k);
St = fft(S);

figure(1)
subplot(2,1,1)
plot(t,S,'k','Linewidth',2)
set(gca,'Fontsize',16), xlabel('Time (t)'), ylabel('S(t)')

subplot(2,1,2)
plot(ks,abs(fftshift(St))/max(abs(St)),'r','Linewidth',2);
set(gca,'Fontsize',16)
set(gca,'Fontsize',16)
xlabel('frequency (Hz)'), ylabel('FFT(S)')


%% Spectrograms for varying window sizes (Gaussian)
```

```
figure(7)

a_vec = [100 80 60 10];
for jj = 1:length(a_vec)
    a = a_vec(jj); %a_vec(jj);
    tslide=0:0.05:L;
    Sgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        Sg=g.*S;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end

    subplot(2,2,jj)
    pcolor(tslide,ks,Sgt_spec.'),
    shading interp
    title(['a = ',num2str(a)],'Fontsize',16)
    xlabel('Time (sec)')
    ylabel('Frequency (Hz)')
    set(gca,'Fontsize',16)
    colormap(hot)
    colorbar
end

sgtitle('Gaussian Gabor Transform Spectrograms for Different
   Widths')

%% Spectrograms for varying window sizes (Mexican Hat)

a_vec = [100 80 60 10];
for jj = 1:length(a_vec)
    a = a_vec(jj); %a_vec(jj);
    tslide=0:.01:L;
    Sgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        g=(1-a*t.^2).*exp(-a*(.5*t.^2));
        Sg=g.*S;
        Sgt=fft(Sg);
        Sgt_spec(j,:) = fftshift(abs(Sgt));
    end

    subplot(2,2,jj)
    pcolor(tslide,ks,Sgt_spec.'),
    shading interp
    title(['a = ',num2str(a)],'Fontsize',16)
    xlabel('Time (sec)')
    ylabel('Frequency (Hz)')
    set(gca,'ylim',[-500 500],'Fontsize',16)
    colormap(hot)
    colorbar
end

sgtitle('Mexican Hat Gabor Transform Spectrograms for Different
   Widths')

%%

tslide=0:0.1:10;

for j=1:length(tslide)
    g=(1-a*t.^2).*exp(-a*(.5*t.^2));
    Sg=g.*S;
    Sgt=fft(Sg);

    subplot(3,1,1)
    plot(t,S,'k','Linewidth',2)
    hold on
    plot(t,g,'m','Linewidth',2)
    hold off
    set(gca, 'Fontsize', 16), xlabel('Time (t)'), ylabel('S(t)')

    subplot(3,1,2)
    plot(t,Sg,'k','Linewidth',2)
    set(gca,'Fontsize',16), xlabel('Time (t)'), ylabel('Sg(t)')

    subplot(3,1,3)
    plot(ks,abs(fftshift(Sgt))/max(abs(Sgt)),'r','Linewidth',2);
    %axis([-50 50 0 1])
    set(gca,'Fontsize',16)
    xlabel('frequency (\omega)'), ylabel('FFT(Sg)')
    drawnow
    pause(0.1)
end
```

```
% Part 2

clear; close all; clc
[y,Fs] = audioread('music1.wav');
y=y';
```

```
tr_piano=length(y)/Fs;  % record time in seconds

% Piano Stuff
L = tr_piano; n = length(y);
t2=linspace(0,L,n+1); t=t2(1:n);
k=(1/L)*[0:ceil(n/2-1) ceil(-n/2):-1];
ks=fftshift(k);
yt = fft(y);

%p8 = audioplayer(y,Fs); playblocking(p8);

%% Plot the signal and its fft

figure(1)
subplot(2,1,1)
hold on
plot(t,y,'k','Linewidth',2)
set(gca,'Fontsize',16), xlabel('Time (t)'), ylabel('S(t)')

subplot(2,1,2)
plot(ks,abs(fftshift(yt))/max(abs(yt)),'r','Linewidth',2);
set(gca,'xlim',[-600 600],'Fontsize',16)
xlabel('frequency (Hz)'), ylabel('FFT(Sp)')
%% Find the main frequencies

firstLoc = k(abs(yt) == max(abs(yt((238<k)&(k<271)))));
firstNote = firstLoc(1)
secondLoc = k(abs(yt) == max(abs(yt((278<k)&(k<306)))));
secondNote = secondLoc(1)
thirdLoc = k(abs(yt) == max(abs(yt((306<k)&(k<342)))));
thirdNote = thirdLoc(1)

%% Spectrogram of unfiltered frequency signal
    a=60;
    sigma=1;
    tslide=0:0.1:L; % 4 SHOULD BE L, BUT IT'S 4 RN TO SAVE
       COMPUTATION TIME
    yht_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        Sg=g.*y;
        ygt=fft(Sg);
        ygt_spec(j,:) = fftshift(abs(ygt)); % We don't want to
           scale it
    end
```

```matlab

    figure(3)
    pcolor(tslide,ks,ygt_spec.')
    shading interp
    set(gca,'Ylim',[0 600],'Fontsize',16)
    xlabel('Time (sec)')
    ylabel('Frequency (Hz)')
    title('Music Score Piano (Spectrogram)')
    colormap(hot)

    print(gcf,'ScorePiano.png','-dpng')
```

```matlab
%% Plot for Gaussian, Mexican Hat, and Shannon Gabor Windows
clear; close all; clc
t=0:.1:20;

a=1;
tau=02;
gauss=(1/(a*sqrt(2*pi)))*exp((-(t-tau).^2)/(2*a^2));

sigma=3;
tau=6;
mexhat=(2/((3*sigma)^.5)*(pi^.25)).*(1-((t-tau)/sigma).^2).*exp
    ((-(t-tau).^2)/(2*sigma^2));

width=2.5;
shift=8.5;
step=((0<=(t-shift))&((t-shift)<=width))/width;

figure(1)
hold on
plot(t,gauss,'k','Linewidth',2);
plot(t,mexhat,'r','Linewidth',2);
plot(t,step,'b','Linewidth',2);
set(gca,'Fontsize',14)
xlabel('Time')
ylabel('Window Height')
title('Gabor Windows')
print(gcf,'GaborWindows.png','-dpng')

%% Plots of Signal Data
load handel
[y2,Fs] = audioread('music1.wav');
[y3,Fs] = audioread('music2.wav');
y1=y';
```

```
y2=y2';
y3=y3';

figure(2)
subplot(3,1,1)
plot((1:length(y1))/Fs,y1)
hold on
plot((1:length(y1))/Fs,exp(-50*((1:length(y1))/Fs-.55).^2),'k','
    Linewidth',2)
ylabel('Amplitude')
title('Handel''s Messiah')
set(gca,'Fontsize',14)
subplot(3,1,2)
plot((1:length(y2))/Fs,y2,'r')
ylabel('Amplitude')
title('Mary Had a Little Lamb Piano')
set(gca,'Fontsize',14)
subplot(3,1,3)
plot((1:length(y3))/Fs,y3,'g')
xlabel('Time [sec]')
ylabel('Amplitude')
title('Mary Had a Little Lamb Recorder')
set(gca,'Fontsize',14)

print(gcf,'MusicPlots.png','-dpng')
```