AMATH 482: Gábor Transformations of Music

Anh-Minh Nguyen

January 13, 2020

Abstract
Expanding our analysis, this report will feature some time analysis as we cover Windowed Fourier Transforms, in the form of Gábor Transforms. We will use these transformations to analyze some pieces of recognizable music, and we will see how the choice of function, width, and translation affects the transformation of signals.

1 Introduction and Overview

In this report, we will expand the scope of our analysis to include the dimension of time with time-frequency analysis instead of pure Fourier analysis. We will analyze a portion of Handel's Messiah and then proceed to dissect two versions of "Mary Had a Little Lamb" using Gábor Transformations. Hopefully, by introducing time as a factor, we can glean more useful information about the signals and their defining characteristics.

2 Theoretical Background

Time is a essential component of music because it determines the arrangement and order of frequencies that we hear. Without time, music would be a cacophony. So, one way to localize both time and frequency is through the use of Windowed Fourier Transforms, which splits the original signal into separate time frames. In this case, we could perform Fourier analysis on each of the frames and gain some information on time. However, we would not be addressing the continuity issues arising from separating the signal while losing the overall context of the original signal as a whole.

To address this, we will introduce the Gábor Transformation, or the short-time Fourier transform, defined below:

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

The $g(\tau - t)$ represents the filter function which transforms the signal. Something you might notice is the t, which helps determine the amount of overlap, addressing the continuity issues around the the frequency of τ with width a. Of course, $f(\tau)$ is our signal.

Applying the Gábor Transformation though, we discretize both time and frequency domains. We consider a lattice of time and frequency:

$$v = m\omega_0 \tag{2}$$

$$\tau = nt_0 \tag{3}$$

We have that $m, n \in \mathbb{Z}$ and $\omega_0, t_0 > 0$ are constants. Our g filter becomes

$$g_{m,n}(t) = e^{i2\pi m\omega_0 t}g(t - nt_0) \tag{4}$$

and the Gábor Transformation *transforms* into

$$\tilde{f}(m,n) = int_{-\inf}^{\inf} f(t)\bar{g}_{m,n}(t)dt \tag{5}$$

So, we over-sample the signal if $0 < t_0, \omega_0 < 1$, achieving good resolution in both domains of time and frequency. But, if $\omega_0, t_0 > 1$, we under-sample the signal and lose much information of the original signal.

A key relationship is that the shorter our window, the more information we have about time but we have less about frequency. The opposite is true with a wider window. Thus, this illustrates the time-frequency trade-off, especially with a fixed window size.

We will stick with the Gaussian filter for the most part, but at the end of our first part, we will experiment with the Mexican Hat wavelet and Shannon window. The equation for the Mexican hat wavelet is:

$$\hat{f}(t) = \frac{2}{\sqrt{\sigma pi^{1/4}}}(1 - (\frac{1}{\sigma})^2)e^{-\dfrac{t^2}{2\sigma^2}} \tag{6}$$

which looks similar to the second moment of the Gaussian function. The equation for the Shannon Wavelet is:

$$\begin{cases} 1 & |t| < a \\ 0 & \text{if otherwise} \end{cases} \tag{7}$$

This is essentially the step function.

As we go on, we will using spectrograms, which are visual representations of the range of frequencies in relation to time.

3 Algorithm Implementation and Development

To begin implementation, many of the steps are same as they are in our first report. We define a length L, our time domain from finding the number of samples and dividing it by the sample rate, and n, the number of points, as the number of samples. We have, v, which is the signal samples in the time domain by transposing y. One interesting change we make is when defining our time-steps t, we divide the steps by the sample rate, Fs. This will scale our time domain into the correct interval of time-steps while having the same number of elements as n. In part 1, we will have an odd number of points, so we augment creation of our frequencies, k, slightly, using $\frac{n-1}{2}$ instead of $\frac{n}{2}$, as the midpoints and then multiplying it by $2\pi/L$, to transform to frequency domain. We then create ks, which we use *fftshit* to shift for plotting purposes. One important note to make, is that our k is at 2π radians per second, something that will useful to know later.

One variable we will be defining over and over is *tslide*. This determines the center frequencies and help in plotting our spectrograms. For the next few sections, we create a vector of window widths, *a vec*. We multiply our signal v by the filter we are using and then Fourier Transform it. We then store the shifted values for plotting. And we keep doing this for different combinations of window widths and translations, and include the Mexican Hat and Shannon Wavelets.

For each combination, we will use nested for loops. In the outside for loop, we define the width and create *tslide* and *vgt spec*, a matrix that will store the shifted values of our Gábor Transformed values to plot after running the inside for loop. Speaking of, the inside for loop will create the filter g for each value of *t slide* and then apply it to v, then we Fourier Transform it and store the shifted absolute frequencies.

For part 2, we begin by following the same procedure, but twice, because we have piano and recorder versions of "Mary Had a Little Lamb". We follow many of the same steps except there will be much more trial and error to find the optimal window width and time-step. The most interesting change we will implement is when we plot the spectrograms, we will convert our radians per seconds to Hertz, so we can determine if we reproduce the songs but in the Hertz range of the instrument. This can be done by dividing our ks frequencies by 2π.

Inside our inside for loop, we will also be grabbing the index of the maximum value of the Fourier Transform of the filtered function because we want to find the central frequencies. Storing them in a vector, *hertz maxes*. We then find the result in frequency domain, by using the indices, and converting to Hertz. Then, we can plot our "clean" signal.

4 Computational Results

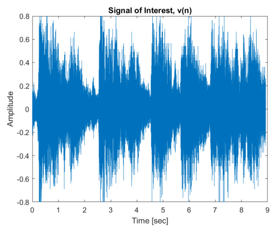

Figure 1: Handel Signal

This is piece of music from Handel that we analyzed in the time domain.

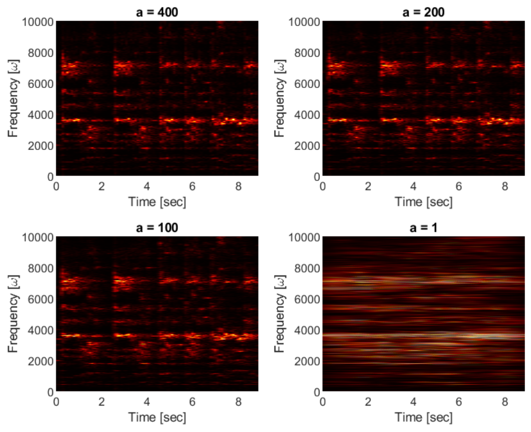

Figure 2: $\Delta t = .1$, Gábor

We began with a translation of $\Delta t = .1$. Here, we can see as we increase the width, we get a better grasp of the frequencies.

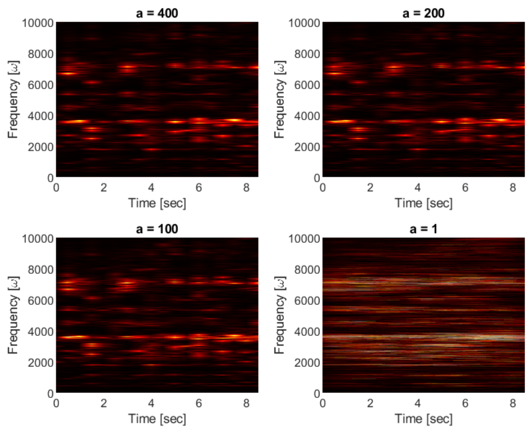

Figure 3: $\Delta t = .0.5$, Gábor

Now, we switch to a translation of $\Delta = 0.5$. This is when we begin to undersample the data, we see the frequencies look "blurrier" compared to when the translation was smaller, losing the context of time. Unfortunately, my computer was unable to go under a translation of 0.1. But, if we did manage to have a smaller translation, the frequencies would appear more "distinct" and defined. One thing we notice is that when the width increases, the more information about time we have.

Let's see what happens when we change the filter functions. We will keep the translation as $\Delta = 0.5$.

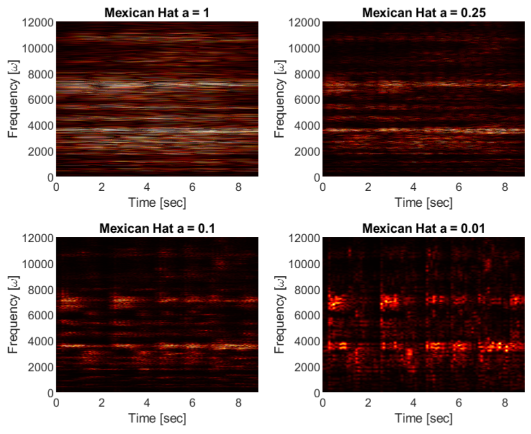

Figure 4: $\Delta t = 0.5$, Mexican Hat Wavelet

Here, we switched to using the Mexican Hat Wavelet as our filter function. First, we see that it is inverse of the Gaussian filter, as the width decreases, the more defined the frequencies become. It does appear that the Mexican Hat provides a better capture of the higher frequencies as well.

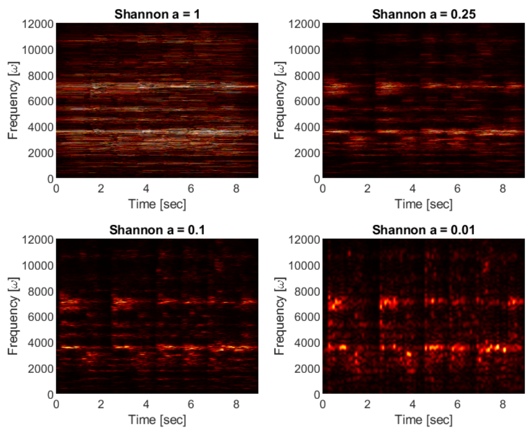

Figure 5: $\Delta t = 0.5$, Shannon Wavelet

The Shannon Wavelet also has the width to time relationship as the Mexican Hat Wavelet, but it seems to take more time to "ramp up". As soon as the width starts getting small, it performs as well as the Mexican Hat Wavelet.

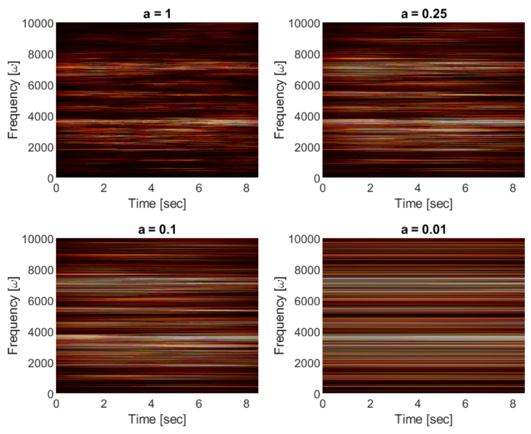

Figure 6: $\Delta t = 0.5$, Gábor Wavelet

This final set of transformations is the Gábor with the same translation as the Mexican Hat and Shannon Wavelets above for comparison sake. In general, we see that it's better to oversample to capture more information about the context of the signal. And most certainly, we have to spend a large amount of time messing with the width and translation size to find the optimal spectrogram.

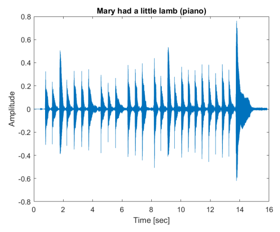

Figure 7: Piano Signal

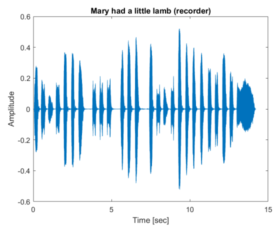

Figure 8: Recorder Signal

For our second part, we observed the signals of "Mary Had a Little Lamb". As we said before, the first version is played on the piano while the second is played on the recorder.

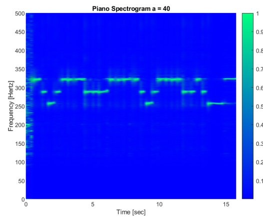

Figure 9: Piano Spectrogram

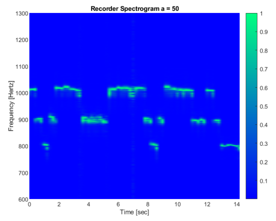

Figure 10: Recorder Spectrogram

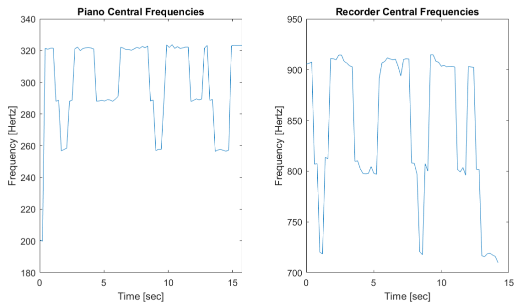

Figure 11: Central Frequencies

After many iterations of testing, I decided to use a width of 40 and 50 and a translation of 0.21 and 0.2 for the piano and recorder. Using the Gaussian window, we managed to filter out the overtones. Observing the piano spectrogram, we see that the range of values are about 250 to 325 Hertz. For the recorder, the range is about 800 to 1020 Hertz.

5 Summary and Conclusions

As we have seen, using Gábor Transformations can give us much more information about the time. Using it, we analyzed the piece of music of Handel and kept time as a factor. We saw how using different filter functions, widths, and translations give us varying information about it. Then, we applied Gábor Transformations to two versions of ”Mary Had a Little Lamb” to filter out the overtones and find the Hertz frequencies.

6 Appendix

A MATLAB Functions

- *fftshift*: Switches opposite sides of a data object, useful for moving an object with center at 0
- *fft*: Fourier Transform of an 1-dimensional object
- *pcolor*: Creates a makeshift color matrix
- *shading*: Translates the color matrix and helps blend the colors
- *colormap*: Essentially paints over the color matrix for labeling purposes

B MATLAB Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% PART 1 %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; close all; clc;
load handel
v = y';
v = v(1:length(v));

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

%% Fourier Transform
L = length(y)/Fs; n = length(v);
t = (1:length(v))/Fs;
k = (2*pi/L)*[0:(n-1)/2 -(n-1)/2:-1];
ks=fftshift(k);
vt = fft(v);
%% FFT
figure(2)
tslide=0:0.1:L;
vgt_spec = repmat(fftshift(abs(vt)),length(tslide),1);
pcolor(tslide,ks,vgt_spec.'),
shading interp
title('fft','Fontsize',8)
xlabel("Time (sec)");
ylabel("Frequency (\omega)");
set(gca,'Fontsize',8)
colormap(hot)
colorbar

%% Gabor Construction t = 0.1%%
figure(3)
a_vec = [400 200 100 1];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tslide=0:0.1:L;
    vgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        vg=g.*v;
        vgt=fft(vg);
        vgt_spec(j,:) = fftshift(abs(vgt));
    end
    subplot(2,2,jj)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['a = ',num2str(a)],'Fontsize',8)
    set(gca,'Fontsize',8, 'Ylim', [0 10000])
    xlabel("Time [sec]");
    ylabel('Frequency [\omega]');
    colormap(hot)
end

%% Gabor Construction t = 0.5 %%
figure(4)
a_vec = [400 200 100 1];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tslide=0:0.5:L;
    vgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        vg=g.*v;
        vgt=fft(vg);
        vgt_spec(j,:) = fftshift(abs(vgt));
    end
    subplot(2,2,jj)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['a = ',num2str(a)],'Fontsize',8)
    set(gca,'Fontsize',8, 'Ylim', [0 10000])
    xlabel("Time [sec]");
    ylabel('Frequency [\omega]');
    colormap(hot)
end

%% Mexican Hat %%
figure(5)
a_vec = [1 .25 .1 .01];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tslide=0:0.1:L;
    vgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        mexicanhat = 2 / (sqrt(3*a)*(pi)^(1/4)) * ...
        (1-((t-tslide(j))/a).^2).* ...
        exp(-(t-tslide(j)).^2 / (2*a^2));
        vg=mexicanhat.*v;
        vgt=fft(vg);
        vgt_spec(j,:) = fftshift(abs(vgt));
    end
    subplot(2,2,jj)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['Mexican Hat a = ',num2str(a)],'Fontsize',8)
    set(gca,'Fontsize',8, 'Ylim', [0 12000])
    xlabel("Time [sec]");
    ylabel('Frequency [\omega]');
    colormap(hot)
end

%% Shannon %%
```

```
figure(6)
a_vec = [1 .25 .1 .01];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tslide=0:0.1:9;
    vgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        shannon = (abs(t-tslide(j)) < a);
        vg=shannon.*v;
        vgt=fft(vg);
        vgt_spec(j,:) = fftshift(abs(vgt));
    end
    subplot(2,2,jj)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['Shannon a = ',num2str(a)],'Fontsize',8)
    set(gca,'Fontsize',8, 'Ylim', [0 12000])
    xlabel("Time [sec]");
    ylabel('Frequency [\omega]');
    colormap(hot)
end

%% Gabor Comparison %%
figure(7)
a_vec = [1 .25 .1 .01];
for jj = 1:length(a_vec)
    a = a_vec(jj);
    tslide=0:0.5:L;
    vgt_spec = zeros(length(tslide),n);
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2);
        vg=g.*v;
        vgt=fft(vg);
        vgt_spec(j,:) = fftshift(abs(vgt));
    end
    subplot(2,2,jj)
    pcolor(tslide,ks,vgt_spec.'),
    shading interp
    title(['a = ',num2str(a)],'Fontsize',8)
    set(gca,'Fontsize',8, 'Ylim', [0 10000])
    xlabel("Time [sec]");
    ylabel('Frequency [\omega]');
    colormap(hot)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% PART 2 %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Piano
close all; clc; clear all;
figure(8)
[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);

%% Recorder
figure(9)
[y2,Fs2] = audioread('music2.wav');
tr_rec=length(y2)/Fs2;  % record time in seconds
plot((1:length(y2))/Fs2,y2);
xlabel('Time [sec]'); ylabel('Amplitude');
title('Mary had a little lamb (recorder)');
% p8 = audioplayer(y2,Fs2); playblocking(p8)

%% Fourier Transform
v = y'; v2 = y2';
n = length(v) ; n2 = length(v2);
t = (1:length(v))/Fs; t2 = (1:length(v2))/Fs2;
k = (2*pi/tr_piano)*[0:n/2-1 -n/2:-1];
k2 = (2*pi/tr_rec)*[0:n2/2-1 -n2/2:-1];
ks=fftshift(k); ks2=fftshift(k2);
%% Gabor Transformation Piano %%
close all;
a = 40;
tslide=0:.21:tr_piano;
hertz_maxes = tslide * 0;
vgt_spec = zeros(length(tslide),n);
for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2);
    vg=g.*v;
    vgt=fft(vg);
    [M, Ind] = max(abs(vgt));
    hertz_maxes(j) = abs(k(Ind)/(2*pi));
    vgt_spec(j,:) = fftshift(abs(vgt))/max(abs(vgt));

end

%% Spectrogram Piano %%
figure(10)
pcolor(tslide, ks/(2*pi),vgt_spec.'),
shading interp
title(['Piano Spectrogram a = ',num2str(a)],'Fontsize',8)
set(gca,'Fontsize',8, 'Ylim', [0 500])
xlabel('Time [sec]');
ylabel('Frequency [Hertz]');
colormap(winter)
colorbar
%% Gabor Transformation Recorder %%
close all;
a = 50;
tslide2=0:.2:tr_rec;
vgt_spec2 = zeros(length(tslide2),n2);
hertz_maxes2 = tslide2*0;
for j=1:length(tslide2)
```

```
    g=exp(-a*(t2-tslide2(j)).^2);
    vg=g.*v2;
    vgt=fft(vg);
    [M, ind] = max(abs(vgt));
    hertz_maxes2(j) = abs(k(ind)/(2*pi));
    vgt_spec2(j,:) = fftshift(abs(vgt))/max(abs(vgt));
end

%% Spectrogram Recorder %%
figure(11)
pcolor(tslide2, ks2/(2*pi), vgt_spec2.'),
shading interp
title(['Recorder Spectrogram a = ',num2str(a)],'Fontsize',8)
set(gca,'Fontsize',8, "Ylim", [600 1300])
xlabel('Time [sec]');
ylabel('Frequency [Hertz]');
colormap(winter)
colorbar

%% Songs
figure(12)
subplot(1, 2, 1)
plot(tslide, hertz_maxes)
title("Piano Central Frequencies")
xlabel("Time [sec]")
ylabel('Frequency [Hertz]')

subplot(1, 2, 2)
plot(tslide2, hertz_maxes2)
title("Recorder Central Frequencies")
xlabel("Time [sec]")
ylabel('Frequency [Hertz]')
```

C References

MATH 482 Course Notes
https://en.wikipedia.org/wiki/Mexican_hat_wavelet
https://en.wikipedia.org/wiki/Shannon_wavelet