Time Frequency Analysis Applied to Music

Jeffrey M Abraham

ABSTRACT

The purpose of this report is to explore the Gábor transform as a method of time frequency analysis with different window functions, and widths. The Gábor method is also used to create a musical score for a short audio recording of a song on multiple instruments.

Keywords: Gábor Transform, Musical Score

I INTRODUCTION AND OVERVIEW

This analysis is being done to record the frequency of notes and the time they were played with enough precision to recreate a musical score. Initially some experimentation with different window types, width, and number of windows used for filters was needed. To ensure that the signal is neither over or under filtered a time frequency analysis of Handel's Messiah was performed. Using what was learned from the exploratory analysis, an analysis of two audio recordings of "Mary Had a Little Lamb" performed on a piano and recorder was performed. The overtones were filtered out, and a musical score recreated.

II THEORETICAL BACKGROUND

Gábor Transform

The Gábor transform utilizes a filtering function applied successively across the time domain to analyze the frequency information of a file in sections to preserve time information. Gábor is traditionally performed by sliding a Gaussian function throughout the signal and a Fast Fourier Transform performed on each filtered signal. An example of this can be seen in **Figure 1** and Equation 1, in practice the window size pictured would be much too wide. This window size was chosen to visualize the process. τ is the sliding variable, so by varying τ one can slide the function around to analyze different portions of the material.

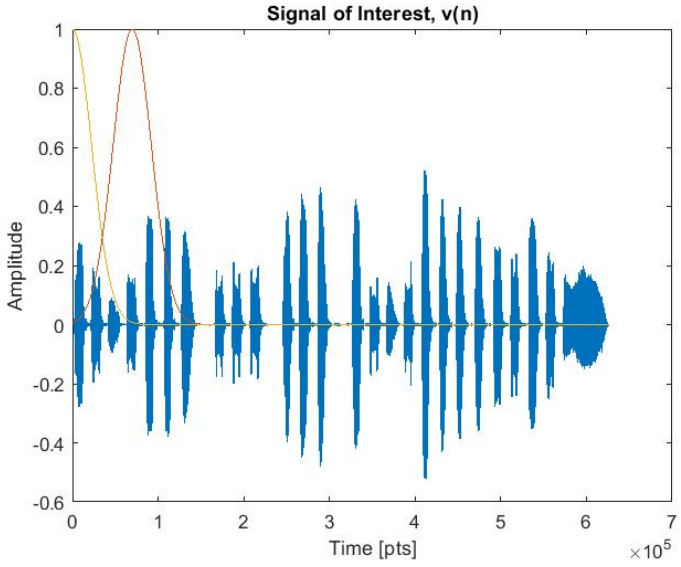

Figure 1. An example of Gábor filtering with Gaussian Functions

$$G(t-\tau) = e^{-a(t-\tau)^2} \tag{1}$$

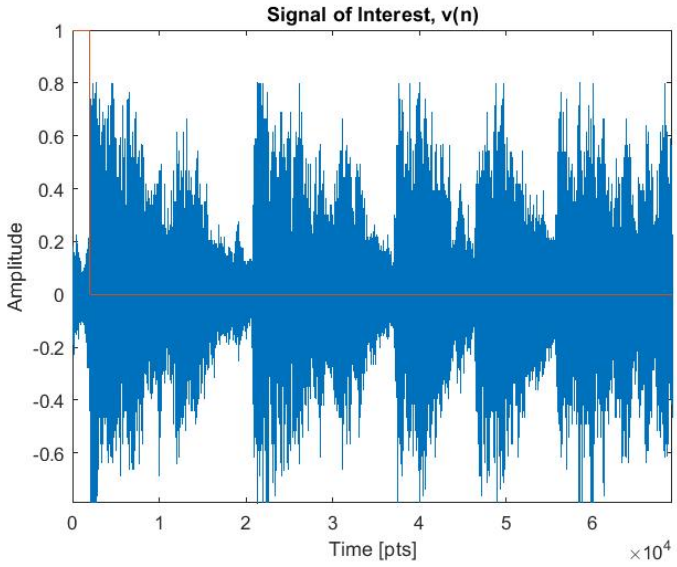

Figure 2. An example of Gábor filtering with Shannon Function

Shannon Window

The Shannon window is another filtering function that is just a step function that has no scaling factor. An example of such function can be seen in **Figure 2** and Equation 2. Since there is no scaling, overlap of functions is unnecessary so the next window begins at the end of the previous window.

$$\begin{cases} 1, & 0 < t < a \\ 0, & a < t < length \end{cases} \quad t \in [0, length] \tag{2}$$

Mexican Hat Wavelet

The Mexican Hat Wavelet is found by taking the second derivative of a negative Gaussian function and normalizing it. The formula is shown in Equation 3. This wavelet got its name because when plotted as a surface in 3 dimensions it looks like a sombrero.

$$H(t) = \left(1 - \frac{(t-\tau)^2}{\sigma^2}\right) e^{-\frac{(t-\tau)^2}{2\sigma^2}} \tag{3}$$

Overtones

Overtones are the properties of instruments that give them their unique sound. If an instrument is playing a note at ω, its overtones will be sounds at integer multiples of ω. In order to only analyze the base notes these overtones need to be filtered out. This can be done by applying a filter function in frequency space centered around the base frequencies of a song.

III ALGORITHM IMPLEMENTATION AND DEVELOPMENT

Gábor Transform Experimentation

I designed three Gábor transform window functions, and tested different window widths to find out an appropriate amount of overlap, and number of windows to give a reasonable amount of resolution in both time and frequency space. The code for this experimentation can be seen in the first matlab file in Appendix B. Each type of window was tested at a wide variety of widths and sample frequencies to create examples of over-filtering, under-filtering, and a good middle ground. Spectrum's made from these Transforms can be seen in **Figures 3**, and **4**. The under filtered transforms give poor resolution in the time domain while over filtered transforms give poor resolution in the frequency domain. This can be seen by the horizontal and vertical stretching for under and over filtered figures respectively.

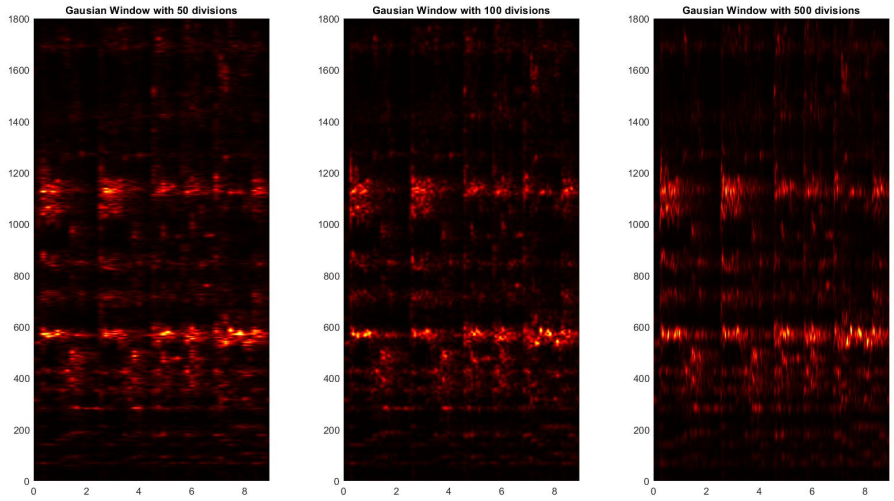

Figure 3. Under-Sampled(Left) & Over-Sampled(Right)

Recording Analyses

After experimentation I decided to use a Gaussian window for my time frequency analysis. I could use the same number of divisions as before as the recordings are of similar length in time. The width of each function had to be changed because the sampling frequency on each recording is different. To have the same level of overlap I set my scaling constant to have a value of 35% of the total amplitude at a distance of half of the time shift so the following filter would intersect the previous function at 0.35. For the piano recording, 80 windows were taken and had a length of 701440 points, so each window was 8768 points or 0.199 seconds. Calculations for the piano case shown in Equation 4 and the recorder in Equation 5

$$0.35 = e^{-a(4384)^2} \quad \frac{\ln 0.35}{-4384^2} = a = 5.46x10^{-8} \tag{4}$$

$$0.35 = e^{-a(3923.2)^2} \quad \frac{\ln 0.35}{-3923.2^2} = a = 6.82x10^{-8} \tag{5}$$

After performing a Gábor transforms with the equations using the above constants, the spectrograms produced needed to have the overtones filtered out. This was done by centering a Gaussian filter around the base frequencies played. This was around 300Hz for the piano and 900Hz for the recorder. The width was set to have a value of 0.5 around 115 Hz in either direction. This song consisted only of three notes so this width covers the notes easily, but an Shannon window could be used for a more abrupt cut-off.

IV COMPUTATIONAL RESULTS

The spectrogram for the Piano gave nice results with 80 divisions. The notes played were not exactly on key, but were the closest to C_4 D_4 and E_4. Lines were added to the spectrogram at those frequencies as shown in **Figure 5**. The notes show up slightly below the note frequencies, this is likely due to imperfect tuning.

The recorder spectrogram gave less pronounced results with 80 divisions, the notes appeared to be made up of a harmony of a few notes around the note frequencies. The notes were closest to G_5 A_5 and B_5. The recorder notes show up above the note frequencies as seen in **Figure 6**, with a recorder the pressure you blow with has a large impact on the way the notes sound, so the player may not have played with the exact pressure required to achieve perfect notes.

Both spectrograms are scaled to have 450Hz represented, but higher in the musical scale each note is further apart as ω_0 is the same note as $2\omega_0$ just one octave apart.

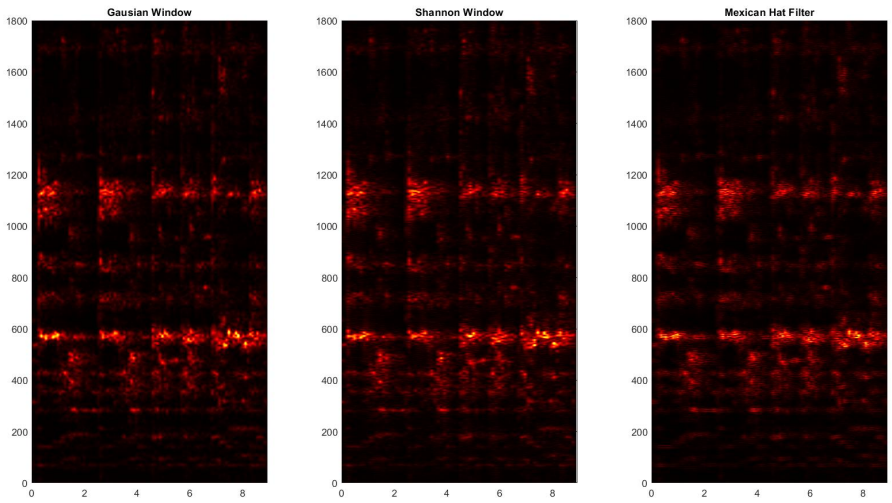

Figure 4. This figure shows spectrograms from three different window shapes

V SUMMARY AND CONCONCLUSIONS

Through exploration of the Gábor transform, I learned how to better tune mathematical methods to their application. The algorithm developed is able to create a musical score for any single instrument recording, and with further testing could work with multiple instruments playing at different center frequencies. The difference between instruments can be seen easily by the overtones represented. The piano's overtones show up with similar intensities at $2\omega_0$ and $3\omega_0$ while the recorder has stronger overtones at $3\omega_0$ than at $2\omega_0$. The recorder was played at higher frequencies and in a different key, but just from the overtones I could tell the difference between the same note played on either instrument just by looking at the overtones on the spectrograms.

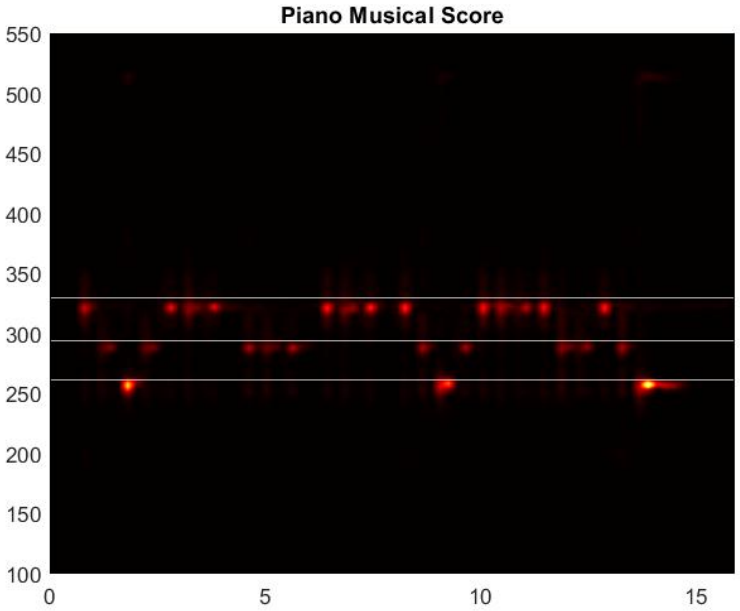

Figure 5. C_4, D_4 and E_4 represented by lines from bottom to top

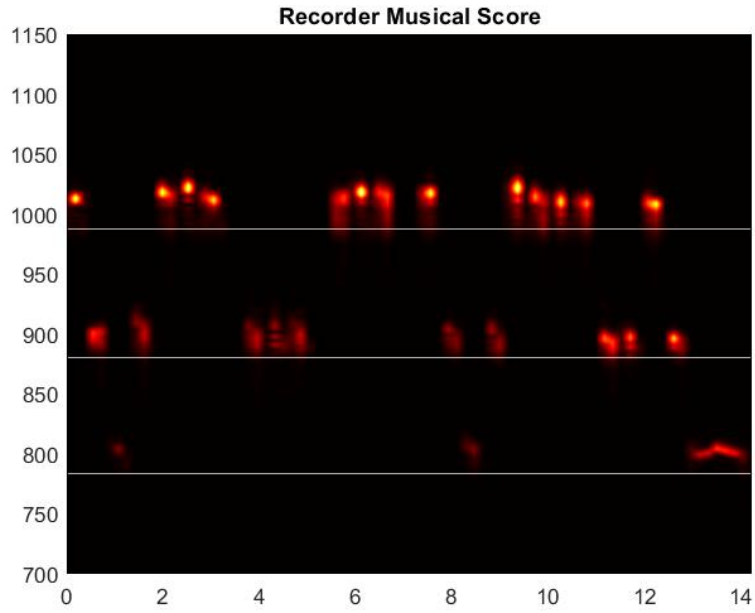

Figure 6. G_5 A_5 and B_5 represented by lines from bottom to top

APPENDIX A.

Matlab Functions

[y,Fs] = audioread('Filename.wav')
Loads an audio file into y and the sampling frequency in Fs

fft(X)
1 dimensional fast Fourier transform.

fftshift(X)
Shifts frequencies to have zero frequencies centered

ifft(Y)
1 dimensional inverse fast Fourier transform.

pcolor(X,Y,Z)
Plots X, and Y with colors determined by the intensities in Z for each point

APPENDIX B.

Matlab Part 1

```
clc; clear all; close all;
%%
load handel
v = y';
L = length(v);
figure(1)
plot(1:L,v);
xlabel('Time [pts]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');
t = 1:length(v);
k = (2*pi*Fs/length(v))*[0:(length(v)-1)/2 -(length(v)-1)/2:-1];
ks = fftshift(k);
%%
%Gausian Window Gabor
figure(2)
res = [50 100 300 500];
% 300:1.8e-5 500:5e-5
al = [5e-7 2e-6 1.8e-5 5e-5];
for jj = 1:4
    tslide = linspace(0,length(v),res(jj));
    a = al(jj);
    spec = zeros(length(tslide),length(v));
    for j = 1:length(tslide)
        g = exp(-a*(t-tslide(j)).^2);
%         plot(t,g)
%         hold on
%         pause(0.1)
        vg = v.*g;
        vgt = fft(vg);
        spec(j,:) = fftshift(abs(vgt));
    end
    subplot(1,4,jj)
    pcolor(tslide/Fs,ks/(2*pi),spec'/max(spec,[],'all'))
    shading interp
    title(['Gausian Window with ', int2str(res(jj)),' divisions'])
    axis([0 length(v)/Fs -1800 1800])
% subplot(1,4,jj)
% plot(t,exp(-a*(t-tslide(j-2)).^2))
% hold on
% plot(t,exp(-a*(t-tslide(j-3)).^2))
end
%%
%Shannon Window Gabor
figure(3)
width = [2000 1000 500];
for kk = 1:3
    shslide = 1:width(kk):length(v);
    spec2 = zeros(length(shslide),length(v));
    for k = 1:length(shslide)
        s = zeros(1,length(v));
        s(1,shslide(k):shslide(k) + width(kk)) = ones(1,width(kk)+1);
```

```
        s = s(1,1:length(v));
        vs = v.*s;
        vst = fft(vs);
        spec2(k,:) = fftshift(abs(vst));
    end
    subplot(1,3,kk)
    pcolor(shslide/Fs,ks/(2*pi),spec2'/max(spec2,[],'all'))
    shading interp
    title(['Shannon Window with a width of ', int2str(width(kk)),'
        samples'])
    axis([0 length(v)/Fs 0 1000])
end
%%
%Mexican Hat Filter
figure(4)
shifts = [50 100 300];
for ll = 1:3
    hslide = linspace(1,length(v),shifts(ll));
    st = 700;
    spec3 =zeros(length(hslide),length(v));
    for l = 1:length(hslide)
        mh = (2/(pi^(1/4)*sqrt(3*st))).*(1-((t-hslide(l))/st).^2).*
            exp(-(((t-hslide(l))).^2)/(2*st^2));
        vm = v.*mh;
        vmt = fft(vm);
        spec3(l,:) = fftshift(abs(vmt));
    %      plot(t,mh)
    %      hold on;
    end
    subplot(1,3,ll)
    pcolor(hslide/Fs,ks/(2*pi),spec3'/max(spec3,[],'all'))
    shading interp
    title(['Mexican Hat Filter with ', int2str(shifts(ll)),'
        divisions'])
    axis([0 length(v)/Fs 0 1000])
%      subplot(1,4,ll)
%      plot(t,(2/(pi^(1/4)*sqrt(3*st))).*(1-((t-hslide(3))/st).^2).*
    exp(-(((t-hslide(3))).^2)/(2*st^2)))
%      hold on
%      plot(t,(2/(pi^(1/4)*sqrt(3*st))).*(1-((t-hslide(4))/st).^2).*
    exp(-(((t-hslide(4))).^2)/(2*st^2)))
end
```

Matlab Piano

```
clc; clear all; close all;
%%
%Load Piano
[y,Fs] = audioread('music1.wav');
v = y';
% v = v(1:length(v)/10);
tr_piano=length(v)/Fs;  % record time in seconds
% plot((1:length(y))/Fs,y);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title('Mary had a little lamb (piano)');
L = length(v);
figure(1)
```

```
plot(1:L,v);
xlabel('Time [pts]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');
t = 1:length(v);
k = (2*pi/tr_piano)*[0:(length(v)/2)-1 -(length(v)/2)-1:-1];
ks = fftshift(k);
ks = ks(1:L);


%%

res = 80;
a = 5.46e-8;
b = 6.5e-7;
tslide = linspace(0,length(v),res);
spec = zeros(length(tslide),length(v));

for j = 1:length(tslide)
    g = exp(-a*(t-tslide(j)).^2);
    %Overtone Filter
    o = exp(-b*(ks-300*2*pi).^2);
%     plot(t,g)
%     hold on;
    vg = v.*g;
    vgt = fftshift(fft(vg)).*o;
    spec(j,:) = abs(vgt);
end

pcolor(tslide/Fs,ks/(2*pi),spec')
shading interp
colormap hot
hold on;
% contour(tslide/Fs,ks/(2*pi),spec'/max(spec,[],'all'),6)
title('Piano Musical Score')
axis([0 length(v)/Fs 100 550])

yline(329.63,'w','Linewidth',1);
yline(293.66,'w','Linewidth',1);
yline(261.63,'w','Linewidth',1);
```

Matlab Recorder

```
clc; clear all; close all;

%Load Recorder
[y,Fs] = audioread('music2.wav');
v = y';
% v = v(1:length(v)/10);
tr_recorder=length(v)/Fs;  % record time in seconds
% plot((1:length(y))/Fs,y);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title('Mary had a little lamb (piano)');
L = length(v);
figure(1)
plot(1:L,v);
xlabel('Time [pts]');
```

```
ylabel('Amplitude');
title('Signal of Interest, v(n)');
t = 1:length(v);
k = (2*pi/tr_recorder)*[0:(length(v)/2)-1 -(length(v)/2)-1:-1];
ks = fftshift(k);
ks = ks(1:L);
%%

res = 80;
a = 6.82e-8;
b = 6.5e-7;
tslide = linspace(0,length(v),res);
spec = zeros(length(tslide),length(v));

for j = 1:length(tslide)
    g = exp(-a*(t-tslide(j)).^2);
    %Overtone Filter
    o = exp(-b*(ks-900*2*pi).^2);
%     plot(t,g)
%     hold on;
    vg = v.*g;
    vgt = fftshift(fft(vg)).*o;
    spec(j,:) = abs(vgt);
end

pcolor(tslide/Fs,ks/(2*pi),spec')
shading interp
colormap hot
hold on;
% contour(tslide/Fs,ks/(2*pi),spec'/max(spec,[],'all'),6)
title('Recorder Musical Score')
axis([0 length(v)/Fs 700 1150])

yline(880,'w','Linewidth',1);
yline(987.77,'w','Linewidth',1);
yline(783.99,'w','Linewidth',1);
```