Gabor transform

Andrei Karavanov

Friday, Feb. 15, 2019

Abstract

In this paper we will perform a time-frequency analisis using Gabor window filtering. In part one, we will explore three different filters and experiment with different widths and sampling rates. In part two, we will use Gabor window filtering to turn two audio files into music scores.

Introduction and Overview

Part One

In this part of the homework we will use time-frequency analysis to analyze a portion of Handels Messiah. More specifically, we will use various Gabor transformations to plot a spectogram of this piece of art. Later, we will discuss how each parameter affects the quality of the spectogram.

Part Two

In this part of the homework we will apply our knowledge of the Gabor filtering to turn two different recording of the "Mary had a little lamb", which were played on different instruments, into music scales.

Theoretical Background

Since in this homework we want to learn more about frequency of the signal through the time, a Fast Fourier Transform will not help us, since in the process of moving data into Fourier domain all information about time gets lost. For that purpose we will use Gabor window filtering. This appoach will allow us to have information about frequency of the signal throughout whole time period. The way Gabor transformation works is that we have to divide whole time length on n different time intervals. And then at each interval apply some sort of filter to get signal information only at that specific time interval. Then we will apply a Fast Fourier Transformation to that filtered part of the signal to move it to the Fourier domain. That will allow us to get all of the frequencies that one can find at that specific time interval. We will use the following formula to move our filtered signal to the Fourier domain:

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

After doing so we will store this information in a data structure, and repeat the same step for all of the time intervals that we have created. One thing to mention, due to Heisenberg uncertainty principle we can not expect to have both good resolution in time domain and good resolution in frequency domain. For that reason, in Part One we will experiment with different widths of filters. We will approach Part Two with already great understanding of the time-frequency analysis. For this part we will come up with a specific parameters of our filter that will both encapsulate the important information and get the right amount of it. We will use the same Gabor window filtering method, which we already discussed above. However, in this part we find the maximum/central frequency at each time step. Those frequencies will be our note frequencies that we will use for recreating the note score.

Algorithm Implementation and Development

Part One

In this part of assignment we will use the following three filters:

1. Gaussian filter of the form $\exp -a * (t - \tau)^2$, where a is the width of the filter and τ is current time position.
2. Ricker/Mexican filter of the form $\frac{2}{\sqrt{3a} * \pi^{1/4}}(1 - a(t - \tau)^2)(e^{-\frac{a}{2}(t\tau)^2})$, where a is the width of the filter and τ is current time position.
3. Shannon filter of the form $|t - \tau| \leq \frac{1}{a}$, where a is the width of the filter and τ is current time position. Note that this is just a step-function window.

Then we will perform Gabor window filtering for different filters, which were described above, different parameters of width - a and different parameters of sampling - tslide. Each transform will follow the following logic:

For each time step in the time interval:
    Initialize a filter around that time step
    Filter our signal using created above filter
    Take the fast Fourier transform of the filtered data
    Shift the filtered data that is already in the Fourier space
    Store the absolute value of the shifted signal

Part Two

In this part of assignment we will use the following filter:

1. Gaussian filter of the form $\exp -a*(t-\tau)^2$, where a is the width of the filter and τ is current time position.

Then for each recording we will perform Gabor Gaussian window filtering and find the central frequency that we will later use to reconstruct our music sheet. The whole algorithm is described above:

For each time step in the time interval:
    Initialize a filter around that time step
    Filter our signal using created above filter
    Take the fast Fourier transform of the filtered data
    Shift the filtered data that is already in the Fourier space
    Find the value and the index of the maximum of the absolute value of data
    Find and transform correspoding wave number into Hz
    Store data in array

Computational Results

Part One

The following figure 1 shows how Gabor window filterings were performed using three different filters: Gaussian - red, Shannon - green, Mexican - blue. Next to it on figure 2, presented spectograms produced by mentioned above windows.

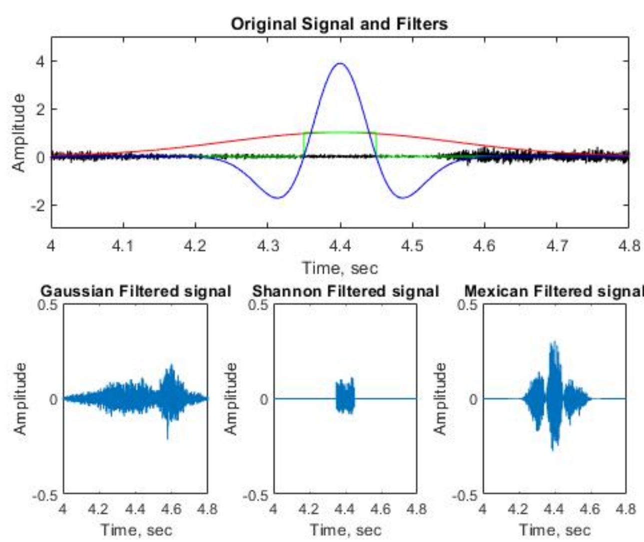


Figure 1: Process of filtering signals

Then we proceed to explore Shannon spectogram. We started with changing its width - a, from 5 to 20 to 200. The following figure depicts our findings.

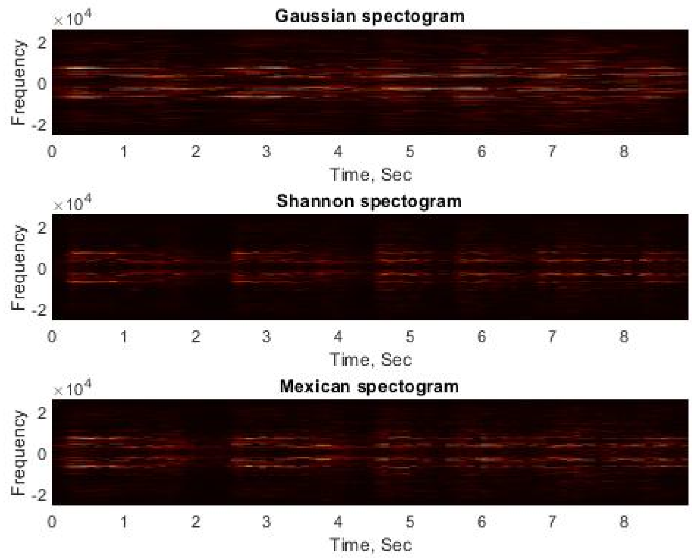


Figure 2: Spectograms produced by different filtering windows.

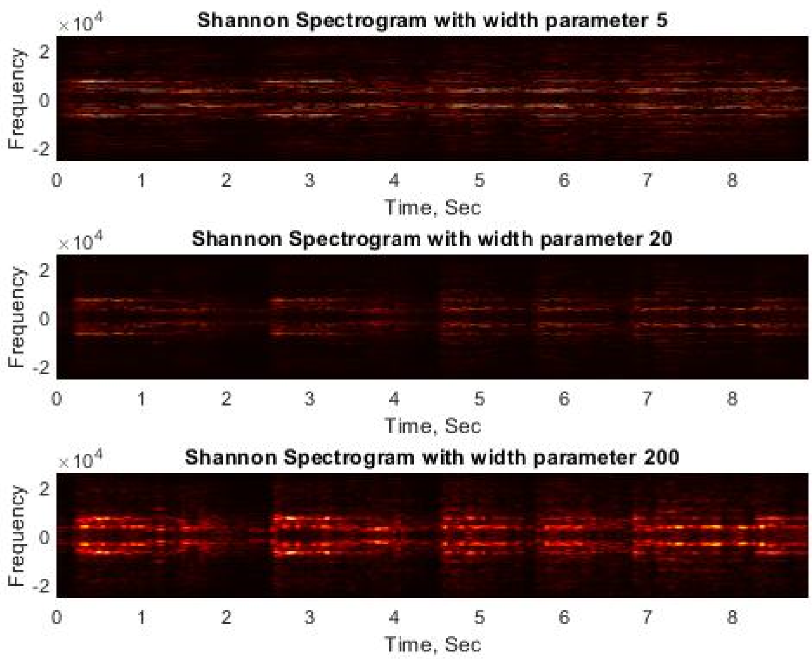


Figure 3: Variations of filtering with different window widths.

From the figure 3, we can see trade offs that occur between time and frequency resolutions. High detalization in frequency domain leads to uncertainty in time domain (top spectogram). While high precision in time domain leads to poor/blurry detalization in frequency domain (bottom spectogram).

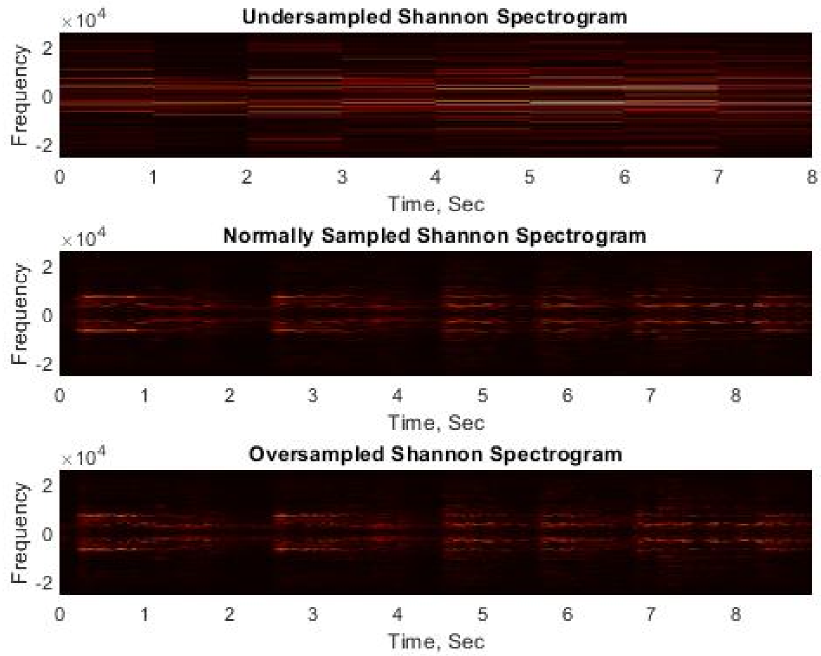


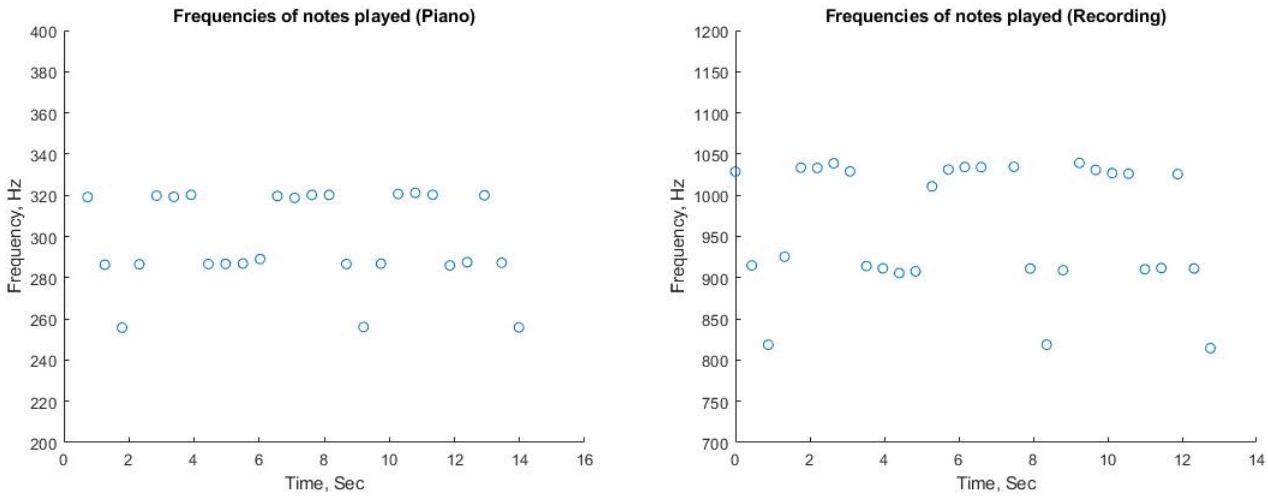
Figure 4: Variations of filtering with different sampling.

From the figure 4 we can see that different sampling rates lead to different quality of the spectogram. Under sampling leads to the loss of important information. Over sampling, on the other hand, while presenting slightly better quality of the spectogram, takes a lot of memory, since it stores more information.

Lastly looking at figure 1 we can see some differences between each window. Mexican window for instance "chops" signal into three sub-parts, which can potentially lead to the loss of information at those nodes. Shannon window is a step-function window, which means that potentially information can be lost on the boundaries. Lastly, since Gaussian window makes values bigger the closer they are to the center of the filter, potentially that can bring slight resolution imprecision.

Part Two

After performing Gabor Gaussian window transformation using Gaussian filter with the widths $a = 0.53$ for the piano and $a = 0.44$ for the recording, the following figure 5 represents music scales of the piano and the recording:



(a) Frequencies of notes played on piano

(b) Frequencies of notes played on recording

Figure 5: Frequencies of notes played on different instruments

Now, if we will translate those frequencies into notes we will get the following scales:
Piano :
[E4,D4,C4,D4,E4,E4,E4,D4,D4,D4,D4,E4,E4,E4,E4,D4,C4,D4,E4,E4,E4,D4,D4,E4,D4,C4]

Recording :
[C6,B5,A#5,B5,C6,C6,C6,C6,B5,B5,B5,B5,C6,C6,C6,C6,B5,A#5,B5,C6,C6,C6,C6,B5,B5,C6,B5,A#5,B5]

Summary and Conclusions

Part One

This portion of the homework gave us a chance to apply our knowledge of time-frequency analysis to analyze a portion of Handels Messiah. In particular, we used Gabor filtering to produce spectrograms of the given audio-file. While doing so we have experimented with the width of the filter, the speed at which it slid across the signal, and different variations of Gabor window. We have started with the Gaussian window, and then looked at the Mexican hat window and a step-function (Shannon) window. Then, we explored effects of different window width and different sampling rates on the quality of the spectogram. Altogether, our work allowed us to gain a better understanding of the Gabor filtering wavelets.

Part Two

This portion of the homework gave us a chance to apply our knowledge of time-frequency analysis to produce a music score of recordings using Gabor Gaussian window filtering. If we will compare the recorder frequencies with the piano frequencies, we can see that the first ones are higher pitched than the other ones. Also, during construction of the note scores we noticed the presence of the overtones in the piano file and lack of those in the recorder file. That resulted in piano sound being more "full" and recording sound more "flat". The overtones, can be seen on the spectogram, which due to a spacial constraint I was not able to attach.

Appendices

MATLAB commands

linspace() : Used to generate linearly spaced vector.
ind2sub() : Used to find subscripts from linear index.
fft() : Used to move data into Fourier space.
fftshift() : Used to shift zero-frequency component to center of spectrum.
ifftshift() : Used to zero-frequency shift.
ifft(): Used to move data back from Fourier space.
pcolor(): Used to create spectograms.
audioread(): Used to read input audio file and turn it into a vector.
scatter(): Used to plot a music scale.

MATLAB code

```
% HW1 - Gabor transforms
%-----------------------------------------------%
% Part 1

clear all; close all; clc
load handel

v = y'/2;

n = length(v);
L = n/Fs;
vf = v(1:end - 1);
k=(2*pi/L)*[0:n/2-1 -n/2:-1]; ks=fftshift(k);

t2 = (1:length(v))/Fs;
t = t2(1:n-1);

width = [5, 20, 300];
tslide_one = 0:1:t(end);
```

```matlab
tslide_two = 0:0.1:t(end);
tslide_three = 0:0.05:t(end);

Vst_spec_one = [];
Vst_spec_two = [];
Vst_spec_three = [];
tslide = tslide_three;
for j=1:length(tslide)
    sOne = (abs(t - tslide(j)) <= 1/width(1)); % Shannon
    sTwo = (abs(t - tslide(j)) <= 1/width(2)); % Shannon
    sThree = (abs(t - tslide(j)) <= 1/width(3)); % Shannon
    VsOne = sOne.*vf; VstOne = fft(VsOne);
    VsTwo = sTwo.*vf; VstTwo = fft(VsTwo);
    VsThree = sThree.*vf; VstThree = fft(VsThree);
    Vst_spec_one = [Vst_spec_one; abs(fftshift(VstOne))];
    Vst_spec_two = [Vst_spec_two; abs(fftshift(VstTwo))];
    Vst_spec_three = [Vst_spec_three; abs(fftshift(VstThree))];
end

Vst_spec_two_slide_one = [];
tslide = tslide_one;
for j=1:length(tslide)
    s = (abs(t - tslide(j)) <= 1/width(1)); % Shannon
    Vs = s.*vf; Vst = fft(Vs);
    Vst_spec_two_slide_one = [Vst_spec_two_slide_one; abs(fftshift(Vst))];
end

Vst_spec_two_slide_three = [];
tslide = tslide_three;
for j=1:length(tslide)
    s = (abs(t - tslide(j)) <= 1/width(3)); % Shannon
    Vs = s.*vf; Vgt = fft(Vs);
    Vst_spec_two_slide_three = [Vst_spec_two_slide_three; abs(fftshift(Vst))];
end

tslide = tslide_two;
Vgt_spec_two = [];
Vst_spec_two_slide_two = [];
Vmt_spec_two = [];
for j=1:length(tslide)
    g = exp(-width(2)*(t - tslide(j)).^2); % Gaussian
    s = (abs(t - tslide(j)) <= 1/width(2)); % Shannon
    m = 2.*(1 - ((t-tslide(j))/width(2).^-1).^2) ...
```

```
        .*exp(-((t-tslide(j)).^2)/...
        (2.*width(2).^-2))/(sqrt(3.*width(2).^-1)...
        .*pi^(1/4)); % Ricker
    Vg = g.*vf; Vgt = fft(Vg);
    Vs = s.*vf; Vst = fft(Vs);
    Vm = m.*vf; Vmt = fft(Vm);
    if (j == length(tslide)/2)
        figure
        subplot(2,3,1:3)
        plot(t,vf,'k',t,g,'r',t,s,'g',t,m,'b')
        axis([4 4.8 -3 5])
        xlabel('Time, sec');
        ylabel('Amplitude');
        title('Original Signal and Filters');
        subplot(2,3,4)
        plot(t,Vg)
        axis([4 4.8 -0.5 0.5])
        xlabel('Time, sec');
        ylabel('Amplitude');
        title('Gaussian Filtered signal');
        subplot(2,3,5)
        plot(t,Vs)
        axis([4 4.8 -0.5 0.5])
        xlabel('Time, sec');
        ylabel('Amplitude');
        title('Shannon Filtered signal');
        subplot(2,3,6)
        plot(t,Vm)
        axis([4 4.8 -0.5 0.5])
        xlabel('Time, sec');
        ylabel('Amplitude');
        title('Mexican Filtered signal');
    end
    Vgt_spec_two = [Vgt_spec_two; abs(fftshift(Vgt))];
    Vst_spec_two_slide_two = [Vst_spec_two_slide_two; abs(fftshift
       (Vst))];
    Vmt_spec_two = [Vmt_spec_two; abs(fftshift(Vmt))];
end
%%
close all;
figure
subplot(3,1,1)
pcolor(tslide_three,ks,Vst_spec_one.'), ...
    shading interp, colormap(hot)
str = sprintf('Shannon Spectrogram with width parameter 5');
```

```
title(str)
xlabel('Time, Sec')
ylabel('Frequency')

subplot(3,1,2)
pcolor(tslide_three,ks,Vst_spec_two.'), shading interp,...
    colormap(hot)
str = sprintf('Shannon Spectrogram with width parameter 20');
title(str)
xlabel('Time, Sec')
ylabel('Frequency')

subplot(3,1,3)
pcolor(tslide_three,ks,Vst_spec_three.'), shading interp,...
    colormap(hot)
str = sprintf('Shannon Spectrogram with width parameter 200');
title(str)
xlabel('Time, Sec')
ylabel('Frequency')

figure
subplot(3,1,1)
pcolor(tslide_one,ks,Vst_spec_two_slide_one.'), ...
    shading interp, colormap(hot)
str = sprintf('Undersampled Shannon Spectrogram');
title(str)
xlabel('Time, Sec')
ylabel('Frequency')

subplot(3,1,2)
pcolor(tslide_two,ks,Vst_spec_two_slide_two.'), shading interp,...
    colormap(hot)
str = sprintf('Normally Sampled Shannon Spectrogram');
title(str)
xlabel('Time, Sec')
ylabel('Frequency')

subplot(3,1,3)
pcolor(tslide_three,ks,Vst_spec_two.'), shading interp,...
    colormap(hot)
str = sprintf('Oversampled Shannon Spectrogram');
title(str)
xlabel('Time, Sec')
ylabel('Frequency')
```

```
figure
subplot(3,1,1)
pcolor(tslide_two,ks,Vgt_spec_two.'), shading interp,...
    colormap(hot)
str = sprintf('Gaussian spectogram');
title(str)
xlabel('Time, Sec')
ylabel('Frequency')
subplot(3,1,2)
pcolor(tslide_two,ks,Vst_spec_two_slide_two.'), shading interp,...
    colormap(hot)
str = sprintf('Shannon spectogram');
title(str)
xlabel('Time, Sec')
ylabel('Frequency')
subplot(3,1,3)
pcolor(tslide_two,ks,Vmt_spec_two.'), shading interp,...
    colormap(hot)
str = sprintf('Mexican spectogram');
title(str)
xlabel('Time, Sec')
ylabel('Frequency')

%% Part 2

clear all; close all; clc

L=16; % record time in seconds
y=audioread('music1.wav');
Fs=length(y)/L;
v = y'/2;
t = (1:length(v))/Fs;
n=length(v);
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);

width = [10000000];
tslide_one = 0.2:0.53:14;

for i = 1:length(width)
    figure(2*i - 1)
    title('Piano recording')

    Sgt_spec = [];
    frequencies = [];
```

```
    for j=1:length(tslide_one)
        g = exp(-width(i)*(t - tslide_one(j)).^10);
        Sg = g.*v; %filtered with gaussian
        Sgt = fft(Sg); %fft gaussian
        [val, index] = max(abs(fftshift(Sgt)));
        frequency = ks(index)/(2*pi);
        frequencies = [frequencies, frequency];
        Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];
    end
    frequencies = abs(frequencies);

    figure
    scatter(tslide_one(1:length(tslide_one)),frequencies(1:length(tslide_one)));
    xlabel('Time, Sec');
    ylabel('Frequency, Hz');
    title('Frequencies of notes played (Piano)');
    axis([0 L 200 400])
    drawnow
end
%%
clear all; close all; clc

L=14; % record time in seconds
y=audioread('music2.wav');
Fs=length(y)/L;
v = y'/2;
t = (1:length(v))/Fs;
n=length(v);
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);

width = [10000000];
tslide = 0:0.44:13;

for i = 1:length(width)
    figure(2*i - 1)
    title('Recording')

    Sgt_spec = [];
    frequencies = [];
    for j=1:length(tslide)
        g = exp(-width(i)*(t - tslide(j)).^10);
        Sg = g.*v; %filtered with gaussian
        Sgt = fft(Sg); %fft gaussian
```

```
        [val, index] = max(abs(fftshift(Sgt)));
        frequency = ks(index)/(2*pi);
        frequencies = [frequencies, frequency];
        Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];
    end
    frequencies = abs(frequencies);

    figure
    scatter(tslide(1:length(tslide)),frequencies(1:length(tslide))
       );
    xlabel('Time, Sec');
    ylabel('Frequency, Hz');
    title('Frequencies of notes played (Recording)');
    axis([0 L 700 1200])
    drawnow
end
```