AMATH 482 Homework 2 Report

Gg Tran

January 29, 2019

Abstract

We were presented with two assignments:

For the first assignment, we were given an audio file from the music piece Handel's Messiah to analyz. We used Gabor transform to obtain the audio's frequency over time. We adjusted the parameters of our Gabor window to see how those changes would affect our obtained results, which were displayed via spectrograms. Furthermore, we compared how different filters (Gaussians, Shannon step function) affected our results.

For the second assignment, we had two audio files which were identical segments of the song "Mary had a little lamb". One file was the piano version of the music, and the other was the recorder version. Using Gabor transform, we attempted to reproduce the music scores of both files. We managed to obtain reasonable frequency ranges of the instruments (200 - 350 for the piano and 800 - 1000 for the recorder, in Hertz).

I. Introduction and Overview

Limitation of the Fourier transform

While the Fourier Transform is efficient for signal processing, it has one major disadvantage: It can only transform stationary signal - signal whose average Fourier frequency value does not change over time. Although one can use the Fourier transform on non-stationary signal by getting repeated measurements of one signal and get the average of all the measurements' Fourier values, for non-stationary signal whose average Fourier frequency value changes over time (e.g a song), the Fourier transform fails. [1]

Windowed Fourier Transform as a modification

The Windowed Fourier Transform is a simple way to fix the Fourier transform's shortcoming with time resolution. In the Fourier analysis, one takes the whole signal and performs Fourier transform at once. As a result, one completely loses the signal's time resolution. On the other hand, the Windowed Fourier Transform takes each segments of a signal over a time interval, then performs Fourier transform on that particular segment. This process is then repeated on the entire signal. By doing so, the Windowed Fourier Transform informs how the signal's frequencies change over time.

Tradeoff between time & frequency resolution

While the Windowed Fourier transform captures both time and frequency, there is a tradeoff between the two: If one wants better time resolution, one will get less frequency resolution, and vice versa. This is due to the Heisenberg Uncertainty Principle.

II. Theoretical Background

Gabor Transform (short-timed Fourier Transform)

The Gabor Transform is essentially a function with two variable t (time) and ω (angular frequency), with the definition

$$G[f](t,\omega) = \int_{-\infty}^{\infty} g(\tau - t)e^{-i\omega\tau} f(\tau)d\tau^1$$

with τ: the center of the Gabor time-filtering window. The kernel $g(\tau - t)$ induces the time localization, i.e it "slides" the Gabor window down the signal's time interval. The Gabor window centers at each t overlaps one another to produce good time-frequency resolution [1].

During computation, we discretize both the time and the frequency domains of the Gabor window[1]. Thus, the Gabor window is discretized over a lattice/sample points which have the parameter t_0 (times) and ω_0 (frequencies), with $t_0, \omega_0 > 0$. If $0 < t_0, \omega_0 < 1$, we will oversample the signal. Such oversampling yields great time-frequency resolution at the expense of computer memory. If $t_0, w_0 > 1$, the Gabor windows will not achieve the time-frequency resolution to reproduce the signal. [1]

III. Algorithm Implementation and Development

Part 1: Gabor transform on the musical piece "Handel's Messiah"

We applied the Gabor window with the Gaussian filter over segments of the music piece. The Gabor window slided over a the time period equaled to the length of the music piece. We manipulated all properties of the Gabor window to see how different parameters and filters would affect our spectrogram. The manipulated properties were:

- Window width *a*.
- Window translation *tslide*. To achieve undersampling, we put Very large time step for *tslide* (1.5). To achieve oversampling, we put Very small time step for *tslide* (0.05).
- The type of filters. Aside from the Gaussian filter, we used the were Mexican Hat and Shannon step function.

Gaussian: exp(-a * (t - tslide(j)).^2)

Mexican hat: (1 - ((t-tslide(j))/(a/2)).^2) .* exp(-(((t-tslide(j))/($a^{2)/2)}$·2))

Shannon: abs(t-tslide(j)) <= a/2

Part 2: Produce the music scores of the song "Mary had a little lamb"

We applied the Gabor window with the Gaussian filter to filter out the overtones of the piano signal and the recorder signal, with **a = 30**, **tslide = 0:0.5:length of respective audio signal**. At each Gabor window, we obtained the max frequency value. All max values were put in the vectors **max_freq_rec** and **max_freq_piano**. Since we defined our Fourier domain as a vector of wavenumber **k**, we divided all max values by 2π to get the frequency in Hertz. We then found the musical notes that matched with our frequencies to produce the music scores.

IV. Computational Results

Part 1: Handel's Messiah

Gaussian filter

For our Gaussian filter, we broadened our filter window by increasing the parameter a, with $a = 20$ gave the widest window, and $a = 2000$ gaved the narrowest window. For $a = 20$ (the lowest value), the window width was the widest, yielded *better frequency resolution*, but *poorer time resolution*. As we increased the paramter a a to 200 and then 2000, the window width became narrower, yielded *better time resolution*, but *poorer frequency resolution*. The Mexican Hat filter seemed to get more frequency components at each time slide than the Shannon filter.

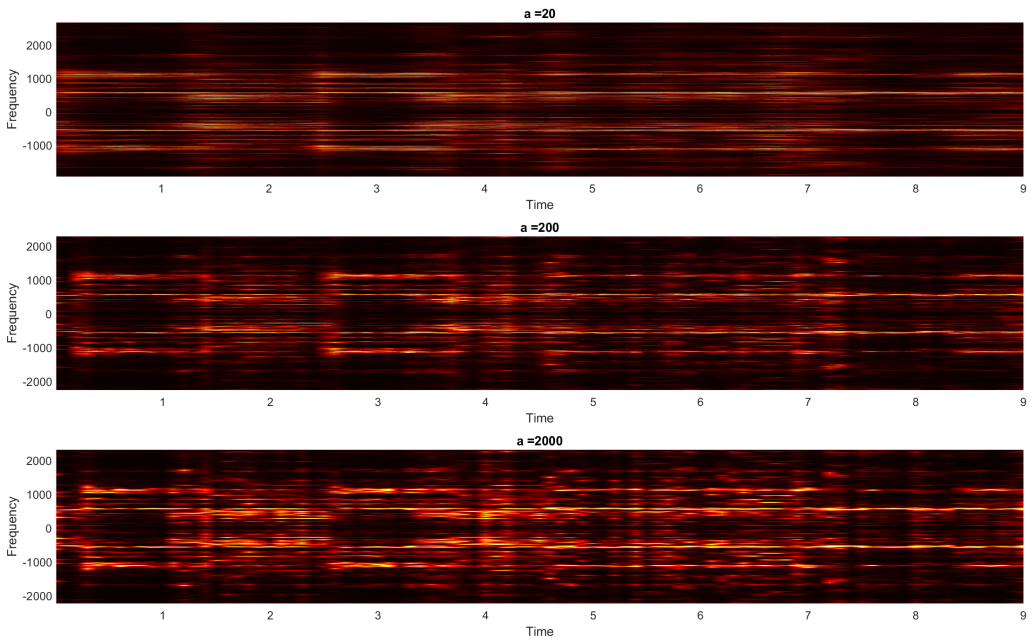

Figure 1: Spectrograms of the music with different width paramter a. The translation parameter was hold constant across all values of a. Gaussian window was widest at the topmost spectrogram, and narrowest at the bottom spectrogram. Narrower window width made the spectrogram gain better time resolution, but poorer frequency resolution.

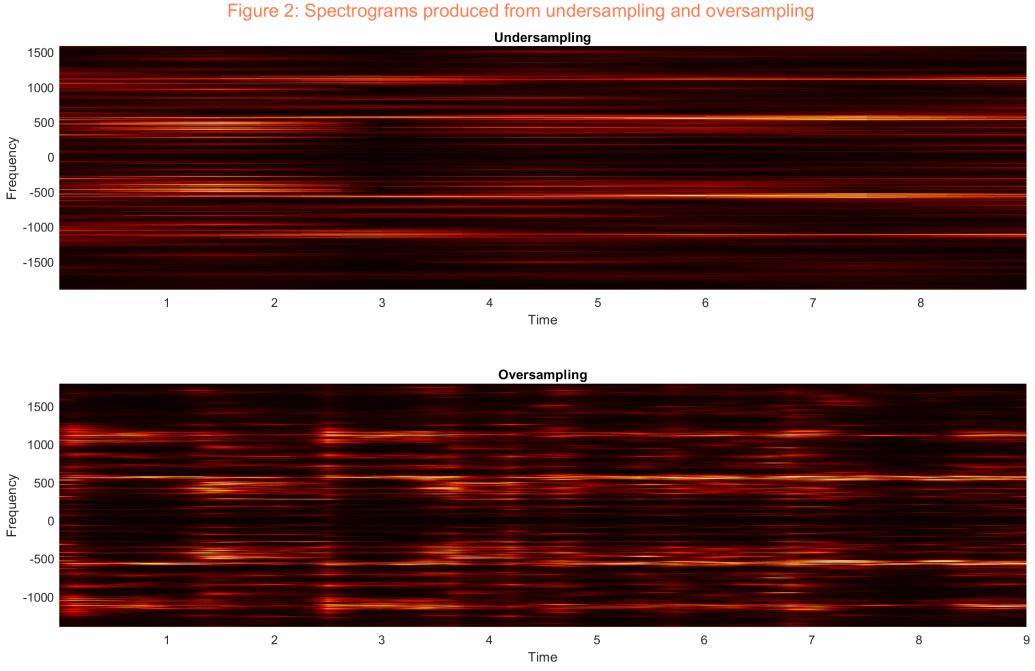

Figure 2: Spectrograms of the music when undersampling (very large window translation) versus when oversampling (very small window translation)

Figure 3: Spectrograms produced from filters other than the Gaussian

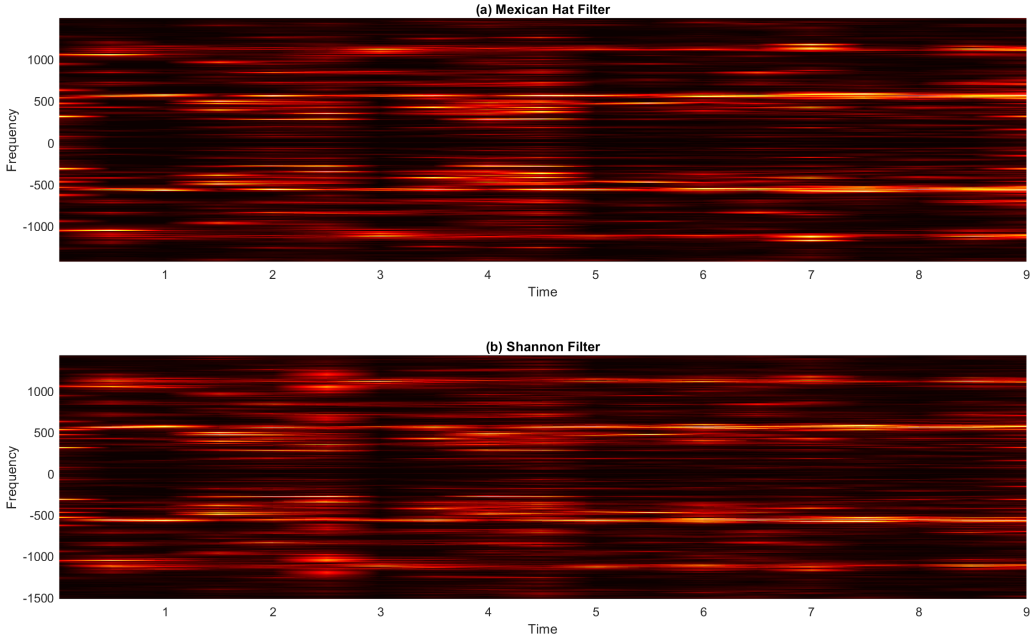

Figure 3: Spectrograms produced when using different filters. a) Windowed transform using Mexican hat filter and b) Windowed transform using Shannon step function. For both (a) and (b), a = 0.2 and `tslide = 0:0.5:9`

Part 2: Mary had a little lamb

After doing the Gabor transform on both the piano and the recorder versions, we found that the note frequencies of the piano ranged from 250 - 320 Hz, whereas those of the recorder ranged from 800 - 1040 Hz.

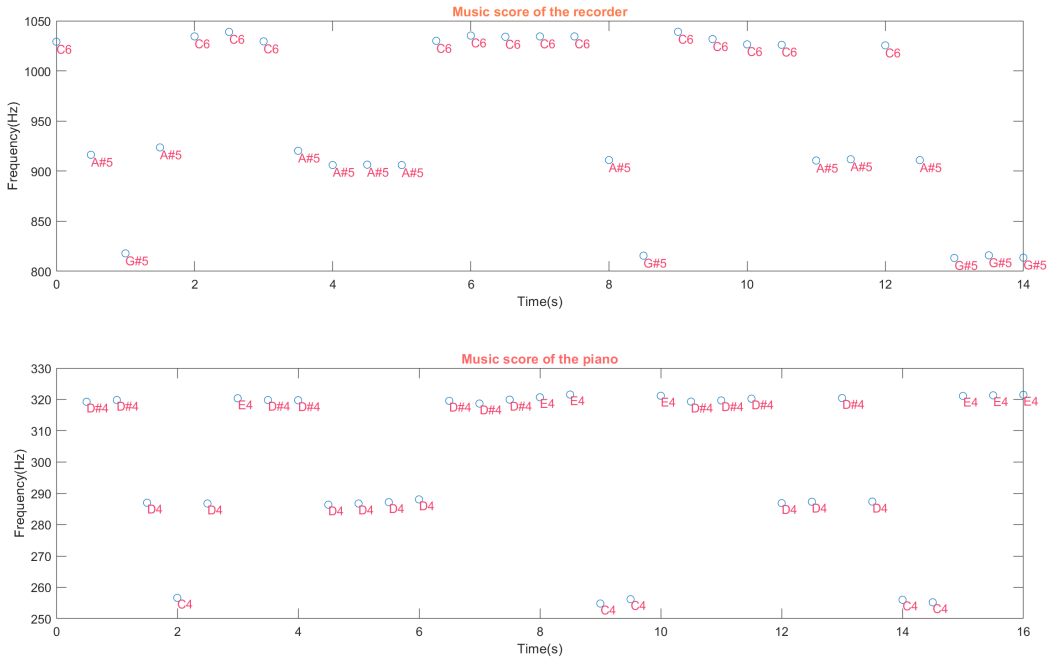

Figure 4: Plots of the frequency range of the musical notes, as played via the recorder and the piano. Notes played through the recorder had much larger frequencies.

V. Summary and Conclusions

The width of the Gabor window affected the time-frequency resolution. Narrower window lost frequency resolution, whereas bigger window lost time resolution. Undersampling and oversampling also affected the time-frequency resolution, with oversampling yielded better resolution in both domains. Out of the three filters used, the Mexican Hat filter seemed to be poorer at filtering out the noises.

As for part 2, the musical notes played through the recorder had higher frequency range than those played through the piano. Perhaps this is due to the different *timbres* that different musical instruments have. A timbre is a characteristic sound made up of harmonics (frequecies that are integer multiples of a fundamental frequency, e.g fundamental frequency of a music note). Timbre makes a musical sound sound different when played by different instruments, since each instruments produce different frequency spectrum. The frequency components generated by the piano and the recorder are very different from each other.

Appendix A: MATLAB functions

fft: Performs the Fast Fourier Transform on the data.

fftshift: Shifts the Fourier form of the function back to its mathematically correct position.

pcolor: Used to draw the manually-made spectrogram.

Appendix B: MATLAB codes

Part 1

```
clear all; close all; clc
load handel
```

```
v = y'/2;
plot((1:length(v))/Fs,v);
xlabel('Time [sec]');
ylabel('Amplitude');
title('Signal of Interest, v(n)');

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

L = 9;
n = length(v);
t2=linspace(0, L, n+1); t=t2(1:n);
k=(2*pi/L)*[0:n/2 -n/2:-1]/(2 * pi);
ks=fftshift(k);

a = [20 200 2000];
tslide = 0:0.1:9;
vft_spec=[];

for index = 1:length(a)
    for j = 1:length(tslide)
       filter = exp(-a(index) * (t - tslide(j)).^2);
       vf = v .* filter;
       vft = fft(vf);
       vft = fftshift(vft);
       vft_spec = [vft_spec; abs((vft))/max(abs(vft))];
    end
    figure(1)
    subplot(length(a), 1, index), pcolor(tslide, ks, vft_spec.'), shading interp
    colormap(hot)
    title(strcat('a = ', num2str(a(index))));
    xlabel('Time'), ylabel('Frequency');
    vft_spec=[];
end
sgtitle('Figure 1: Spectrograms produced from increasingly narrower Gaussian filter window',...
    'Color', [255, 121, 77]/255)



%Undersampling | Oversampling

time_step = [1.5 0.05]
a = 20;
vft_spec_sam=[];

for index = 1:length(time_step)
    tslide_samp = 0:time_step(index):9;
    for j = 1:length(tslide_samp)
       filter = exp(-a * (t - tslide_samp(j)).^2);
       vf = v .* filter;
       vft = fft(vf);
       vft = fftshift(vft);
```

```
       vft_spec_sam = [vft_spec_sam; abs((vft))/max(abs(vft))];
    end
    figure(2)
    subplot(2,1,index), pcolor(tslide_samp, ks, vft_spec_sam.'), shading interp
    colormap(hot)
    xlabel('Time'), ylabel('Frequency');

    title('Undersampling')
    vft_spec_sam=[];
end
title('Oversampling')
sgtitle('Figure 2: Spectrograms produced from undersampling and oversampling',...
    'Color', [255, 121, 77]/255)

% MEXICAN HAT WAVELET
spec_mex = [];
a = 0.2;
tslide = 0:0.5:9;
for j = 1:length(tslide)
       filter = (1 - ((t-tslide(j))/(a/2)).^2) .* exp(-(((t-tslide(j))/(a^2)/2).^2));
       vf = v .* filter;
       vft = fft(vf);
       vft = fftshift(vft);
       spec_mex = [spec_mex; abs((vft))/max(abs(vft))];

end
figure(3)
subplot(2, 1, 1),pcolor(tslide, ks, spec_mex.'), shading interp
colormap(hot)
xlabel('Time'), ylabel('Frequency');
title('(a) Mexican Hat Filter')


%SHANNON
spec_shan = [];
a = 0.2;
tslide = 0:0.5:9;
for j = 1:length(tslide)
        filter = abs(t-tslide(j)) <= a/2;
       vf = v .* filter;
       vft = fft(vf);
       vft = fftshift(vft);
       spec_shan = [spec_shan; abs((vft))/max(abs(vft))];
end
figure(3)
subplot(2, 1, 2),pcolor(tslide, ks, spec_shan.'), shading interp
 colormap(hot)
  xlabel('Time'), ylabel('Frequency');
   title('(b) Shannon Filter')
sgtitle('Figure 3: Spectrograms produced from filters other than the Gaussian',...
    'Color', [255, 121, 77]/255)
```

Part 2

```
clear all; close all; clc
tr_piano=16; % record time in seconds
 y2=audioread('music1.wav'); Fs2=length(y2)/tr_piano;
% plot((1:length(y2))/Fs2,y2);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title('Mary had a little lamb (piano)'); drawnow
% p8 = audioplayer(y2,Fs); playblocking(p8);
% figure(2)
tr_rec=14; % record time in seconds
y1=audioread('music2.wav'); Fs1=length(y1)/tr_rec;
% plot((1:length(y1))/Fs1,y1);
% xlabel('Time [sec]'); ylabel('Amplitude');
% title('Mary had a little lamb (recorder)');
% p8 = audioplayer(y1,Fs1); playblocking(p8);

L = tr_rec;
n = length(y1);
t2=linspace(0, L, n+1); t=t2(1:n);
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);
tslide = 0:0.5:tr_rec;

spec = [];

a = [30 300 3000];
tslide = 0:0.5:tr_rec;

spec = [];
for index = 1:length(a)
    for j = 1:length(tslide)
       filter = exp(-a(index) * (t - tslide(j)).^2).';
       yf = y1 .* filter;
       yft = fft(yf);
       [M, I] = max(abs(yft));
       yft = fftshift(yft);
       max_freq_rec(index,j) = k(I)/(2*pi);
       spec = [spec, abs(yft)/max(abs(yft))];
    %    subplot(3,1,1), plot(t, y1,'k', t, filter,'r')
    %    subplot(3,1,2), plot(t, yf,'k')
    %    subplot(3,1,3), plot(ks, abs(fftshift(yft))/max(abs(yft)))
    %    drawnow
    %    pause(0.0001)
    end
%     figure(1)
%      subplot(length(a), 1, index), pcolor(tslide, ks , spec), shading interp
%      colormap(hot)
%      title(strcat('recorder, a=', num2str(a(index))))
%      figure(2)
%      subplot(length(a), 1, index), plot(tslide, max_freq_rec(index, 1:end))
%      spec = [];

end
subplot(2, 1, 1)
```

```
notes_rec = {'C6'   'A#5'   'G#5'   'A#5'   'C6'    'C6'    'C6'    'A#5'...
    'A#5'   'A#5'   'A#5'   'C6'    'C6'    'C6'    'C6'    'C6'    'A#5'...
    'G#5'   'C6'    'C6'    'C6'    'C6'    'A#5'   'A#5'   'C6'    'A#5'...
    'G#5'   'G#5'   'G#5'};
plot(tslide, abs(max_freq_rec(1, 1:end)),'o')
xlabel('Time(s)'), ylabel('Frequency(Hz)')
title('Music score of the recorder', 'color', [255, 121, 77]/255)
d = 0.5;
text(tslide, abs(max_freq_rec(1, 1:end)) + d, notes_rec,...
    'VerticalAlignment', 'top', 'HorizontalAlignment', 'left', ...
    'color',[244, 66, 110]/255)



%PIANO
L = tr_piano;
n = length(y2);
t2=linspace(0, L, n+1); t=t2(1:n);
k=(2*pi/L)*[0:n/2-1 -n/2:-1];
ks=fftshift(k);
a = [30 300 3000];
tslide = 0:0.5:tr_piano;

spec = [];
for index =1:length(a)
    for j = 1:length(tslide)
       filter = exp(-a(index) * (t - tslide(j)).^2).';
       yf = y2 .* filter;
       yft = fft(yf);
       [M, I] = max(abs(yft));
       yft = fftshift(yft);
       max_freq_piano(index,j) = k(I)/(2*pi);
       spec = [spec, abs(yft)/max(abs(yft))];
    %    subplot(3,1,1), plot(t, y1,'k', t, filter,'r')
    %    subplot(3,1,2), plot(t, yf,'k')
    %    subplot(3,1,3), plot(ks, abs(fftshift(yft))/max(abs(yft)))
    %    drawnow
    %    pause(0.0001)
    end
%     figure(3)
%      subplot(length(a), 1, index), pcolor(tslide, ks , spec), shading interp
%      colormap(hot)
%      title(strcat('piano, a=', num2str(a(index))))
%      figure(4)
%      subplot(length(a), 1, index), plot(tslide(3:end), max_freq_piano(index, 3:end))
%      spec = [];

end
subplot(2,1, 2)
notes_piano =  {'D#4' 'D#4 '    'D4'    'C4'    'D4'    'E4'    'D#4'   'D#4'   'D4'    ...
    'D4 '   'D4 '   'D4 '   'D#4 '  'D#4'   'D#4'   'E4'    'E4'    'C4'...
    'C4'    'E4'    'D#4'   'D#4'   'D#4'   'D4'    'D4'    'D#4'   'D4'    'C4'    ...
```

```
    'C4'    'E4'    'E4'    'E4'};
plot(tslide(2:end), abs(max_freq_piano(1, 2:end)),'o')
xlabel('Time(s)'), ylabel('Frequency(Hz)')
title('Music score of the piano', 'color', [255, 102, 102]/255)
d = 0.5;
text(tslide(2:end), abs(max_freq_piano(1, 2:end)) + d, notes_piano,...
    'VerticalAlignment', 'top', 'HorizontalAlignment', 'left', ...
    'color',[244, 66, 110]/255)
```

Reference

1. Kutz, Nathan J. "Computational Methods for Data Analysis." *In Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data.* Oxford University Press, 2013.