AMATH482 HOMEWORK2

CIHAN MERTOZ

Abstract. Signal processing is one of the most important developments in the recent history of mathematics. In the last homework, the Fast Fourier Transform was used to analyze a data sets. In this Homework, unlike the last one, our signal is not static which means it changes over time. Since the Fourier Transform only gives information about the the frequency domain, we need use another technique to get information about both the time and the frequency of signal while still obeying the Heisenberg Uncertainty Principle.

1. Introduction and Overview

In this assignment we have two seperate sections. In the first one, Handel's Messiah is analyzed using time-frequency analysis methods. In the second part, we are given to seperate data sets containing the Mary had a little lamb song. The first data set has this song played in a piano and the second one has it recorded. We are asked to produce the music score for each recording and compare them.

2. Theoretical Background

2.1. **Gabor Transform.** Fourier Transform(FT) is a very useful signal analysis tool. The main idea of FT is that it decomposes a function of time into its constituent frequencies. However, when FT'ed, the function loses its time components. Since we are dealing with music files, knowing when a certain frequency happens is crucial.Thus we need a new Transform that accounts for the time as well. Gabor transform or short-time Fourier transform, is method developed by the British Physicist Dennis Gabor.The idea of this transform is pretty simple. Fourier transform:

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

It can easily be seen that there is no term that depends on time here. Gabor introduced a time-filter $g(\tau - t)$ where τ is the center of the specific time window we are looking at with the width a. By changing the Fourier Kernel with a new Kernel that acounts for the time as well, the FT takes the form:

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

This is the infamous Gabor Transform. To solve any problem with Gabor Transform computationally, we need to discretize both the frequency and the time domain. We will use this fact in our problem as well. Lastly, although Gabor Transform does a good job of giving information about both the time and the frequency

Date: February 2020.

at the same, due to the Heisenberg Uncertainty Principle, it has its limitations. Namely, the better the time resolution the worse the frequency resolution. We will try to overcome this limitation by working with different time resolutions so that we have a more complete analysis.

2.2. **Wavelets.** Since we are slicing up the signal into different subsections, we need a method that slices and highlights different parts of the signal at hand. There are three main wavelets and depending on the demands of the field, an appropriate filter can be chosen.

2.2.1. *Gaussian Window.* Gaussian Window, creates a Gaussian distribution centered around with width a:

$$g(\tau - t) = e^{-a(\tau - t)^2}$$

2.2.2. *Mexican Hat Wavelet.* As the name suggest, this window looks like a Mexican hat:

$$g(\tau - t) = e^{\frac{(\tau - t)^2}{2a^2}} \frac{2}{\sqrt{3a}\pi^{1/4}}(1 - \frac{(t - \tau)^2}{a^2})$$

2.2.3. *Shanon Wavelet.* Shannon Wavelet is window which has a value of 1 inside the given range and zero outside:

$$g(\tau - t) = |t - \tau| < \sigma$$

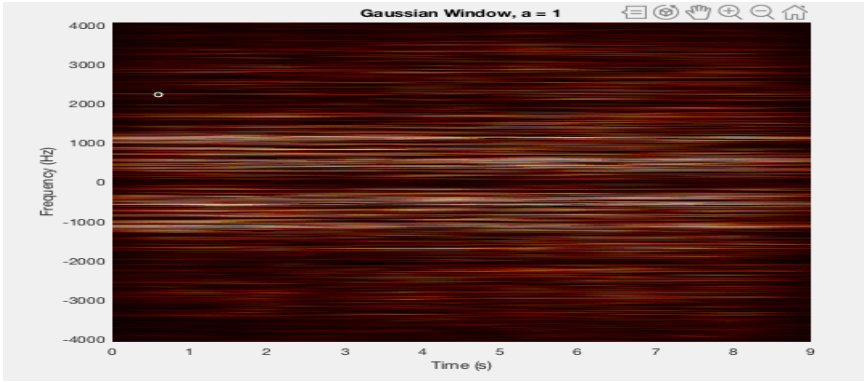
3. Algorithm Implementation and Development

3.1. For the first part of the homework, I played around with the different wavelet types and sizes. Basically, I created a FT with length 9. After the FT is defined I used it to apply different wavelet types. In the first three, I played with the size of the Gaussian filter and tried to see the differences between each plot. Then I created two other plots where I changed the window width of a Mexican Hat Wavelet. Lastly, I played around with sigma value of the Shannon wavelet to see the differences between the plots.

3.2. For the second part of the homework, I started by uploading the music files we are given. Then created a the basis for FT. Again applied a Gaussian window to find the frequencies. Different than Part 1, we also found the maximum frequency in each defined time-step. This gave us the ability to find the score of the files at hand

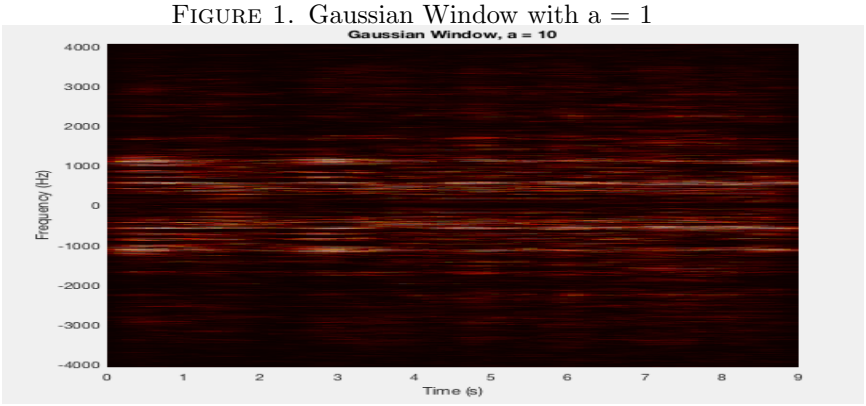
4. Computational Results

4.1. Gaussian Window. Here are the results for the Gaussian Windows with different "a" values.



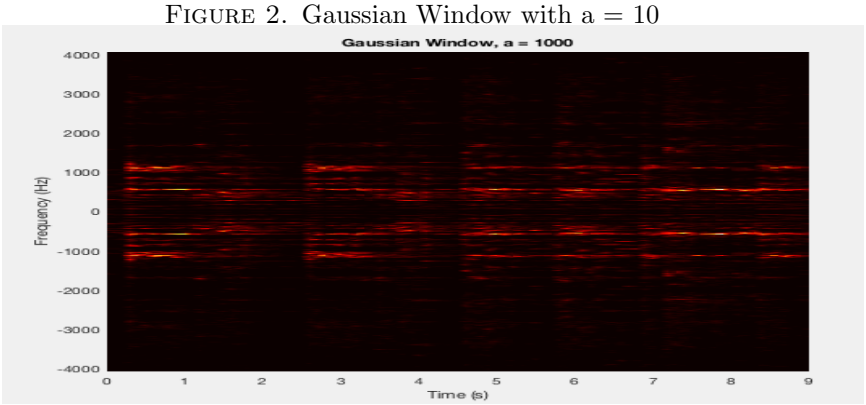
0.5

FIGURE 1. Gaussian Window with a = 1



0.5

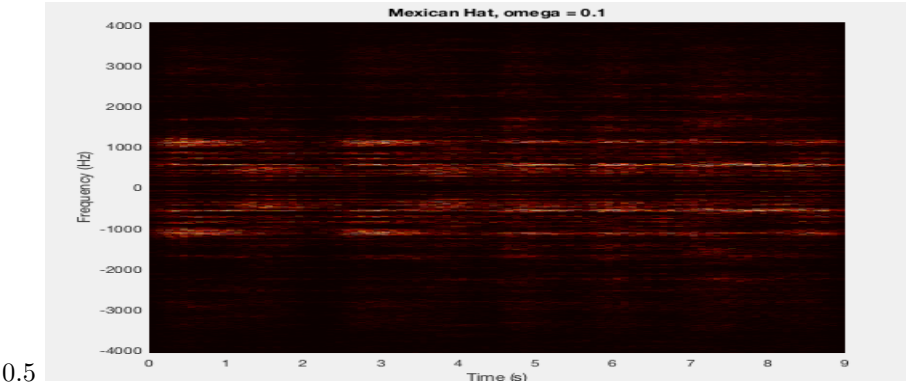
FIGURE 2. Gaussian Window with a = 10



0.5

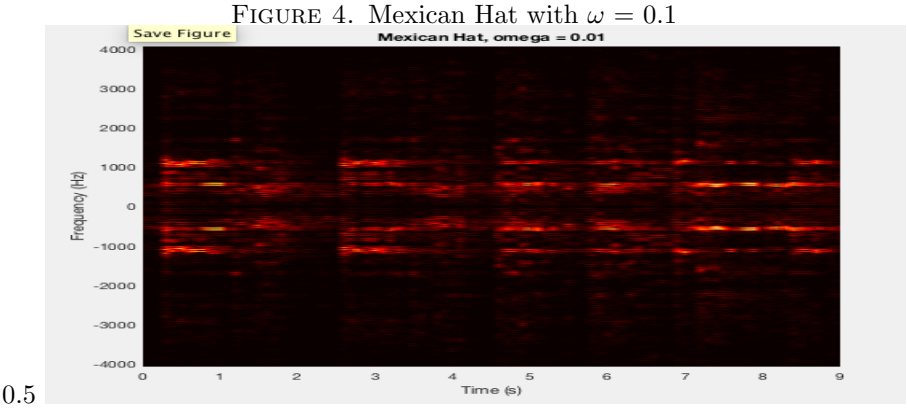
FIGURE 3. Gaussian Window with a = 100

4.2. Mexican Hat Wavelet. Here are the results for the Mexican Hat Wavelets with different ω values.



0.5

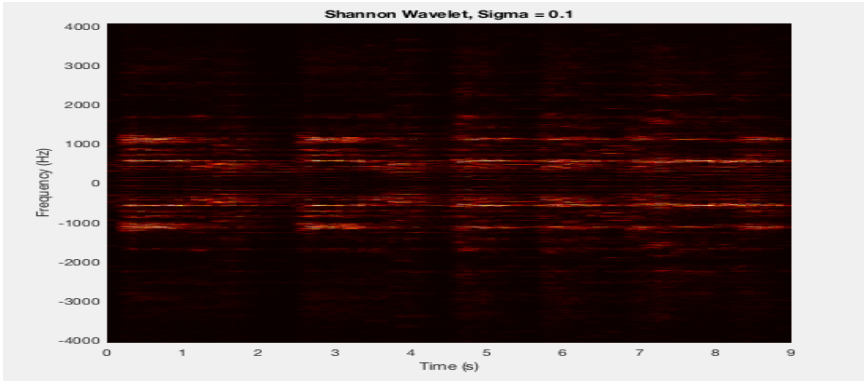
FIGURE 4. Mexican Hat with $\omega = 0.1$



0.5

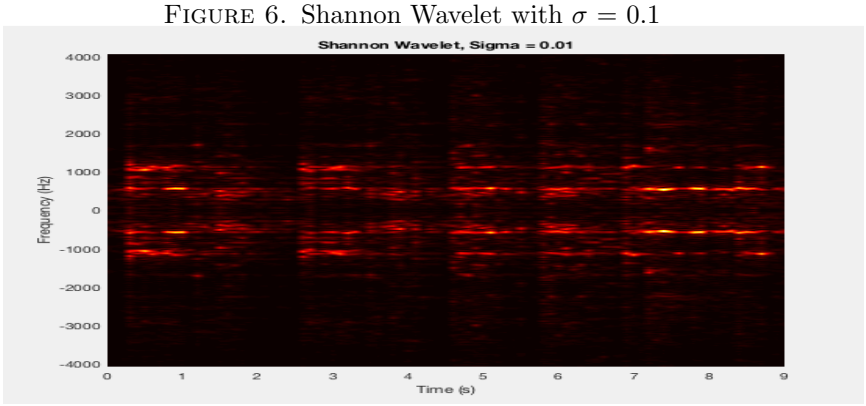
FIGURE 5. Mexican Hat with $\omega = 0.01$

4.3. Shannon Wavelet. Here are the results for the Shannon Wavelet with different σ values.



0.5

FIGURE 6. Shannon Wavelet with $\sigma = 0.1$



0.5

FIGURE 7. Shannon Wavelet with $\sigma = 0.01$

5. CONCLUSION

As a conclusion, we learned that dynamic signals require a different approach than static signals. We need to choose windows for analysis, however due to the Heisenberg Uncertainty principle, we need to pick many different windows for the sake of completeness.

6. Appendix A

```
% Part 1
clear all; close all; clc;

load handel
v = y'/2;
v = v(1:length(v)-1);

% Fourier Transform
L = 9; n = length(v);
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= 0:0.1:9;
spectrum1 = [];
% Gaussian Window, a = 1
for i=1:length(tslide)
    g = exp(-1*(t-tslide(i)).^2);
    filter_g = g.*v;
    fourier_vg = fft(filter_g);
    spectrum1 = [spectrum1; abs(fftshift(fourier_vg))];
end
figure;
pcolor(tslide, ks./(2*pi),spectrum1.'), shading interp, colormap(hot)
xlabel("Time (s)");ylabel("Frequency (Hz)"); title("Gaussian Window, a = 1");


spectrum2 = [];
%Gaussian Window, a = 10
for i=1:length(tslide)
    g = exp(-10*(t-tslide(i)).^2);
    filter_g = g.*v;
    fourier_vg = fft(filter_g);
    spectrum2 = [spectrum2; abs(fftshift(fourier_vg))];
end
figure;
pcolor(tslide, ks./(2*pi),spectrum2.'), shading interp, colormap(hot)
xlabel("Time (s)");ylabel("Frequency (Hz)"); title("Gaussian Window, a = 10")

spectrum3 = [];
%Gaussian Window, a = 1000
for i=1:length(tslide)
    g = exp(-1000*(t-tslide(i)).^2);
```

```
    filter_g = g.*v;
    fourier_vg = fft(filter_g);
    spectrum3 = [spectrum3; abs(fftshift(fourier_vg))];
end
figure;
pcolor(tslide, ks./(2*pi),spectrum3.'), shading interp, colormap(hot)
xlabel("Time (s)");ylabel("Frequency (Hz)"); title("Gaussian Window, a = 1000")

spectrum4 = [];
om = 0.1;
%Mexican Hat Wavelet, omega = 0.1
for i=1:length(tslide)
    g = (2/ (sqrt(3*om) * pi^(0.25))).* (1-((t-tslide(i))/om).^2) .* exp(-((t-tsli
    filter_g = g.*v;
    fourier_vg = fft(filter_g);
    spectrum4 = [spectrum4; abs(fftshift(fourier_vg))];
end
figure;
pcolor(tslide, ks./(2*pi),spectrum4.'), shading interp, colormap(hot)
xlabel("Time (s)");ylabel("Frequency (Hz)"); title("Mexican Hat, omega = 0.1")


spectrum5 = [];
om = 0.01;
%Mexican Hat Wavelet, omega = 0.01
for i=1:length(tslide)
    g = (2/ (sqrt(3*om) * pi^(0.25))).* (1-((t-tslide(i))/om).^2) .* exp(-((t-tsli
    filter_g = g.*v;
    fourier_vg = fft(filter_g);
    spectrum5 = [spectrum5; abs(fftshift(fourier_vg))];
end
figure;
pcolor(tslide, ks./(2*pi),spectrum5.'), shading interp, colormap(hot)
xlabel("Time (s)");ylabel("Frequency (Hz)"); title("Mexican Hat, omega = 0.01")

spectrum6 = [];
sg = 0.1;
%Shannon Wavelett, sigma = 0.1
for i=1:length(tslide)
    g = abs(t-tslide(i)) < sg;
    filter_g = g.*v;
    fourier_vg = fft(filter_g);
    spectrum6 = [spectrum6; abs(fftshift(fourier_vg))];
end
figure;
```

```
pcolor(tslide, ks./(2*pi),spectrum6.'), shading interp, colormap(hot)
xlabel("Time (s)");ylabel("Frequency (Hz)"); title("Shannon Wavelet, Sigma = 0.1"

spectrum7 = [];
sg = 0.01;
%Shannon Wavelett, sigma = 0.01
for i=1:length(tslide)
    g = abs(t-tslide(i)) < sg;
    filter_g = g.*v;
    fourier_vg = fft(filter_g);
    spectrum7 = [spectrum7; abs(fftshift(fourier_vg))];
end
figure;
pcolor(tslide, ks./(2*pi),spectrum7.'), shading interp, colormap(hot)
xlabel("Time (s)");ylabel("Frequency (Hz)"); title("Shannon Wavelet, Sigma = 0.01"

% Part 2.1
close all; clear all; clc;

[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);
%figure(2)

% Fourier Transform
n = length(y);
k = (2*pi/tr_piano)*[0:n/2-1 -n/2:-1]; ks = fftshift(k);
t = (1:n)/Fs;
tslide = 0:0.2:tr_piano;

spectrum = [];
score = [];

% Gaussian Window and Score
for i=1:length(tslide)
    g = exp(-100*(t-tslide(i)).^2);
    filter_g = g.*y';
    fourier_vg = fft(filter_g);
    spectrum = [spectrum; abs(fftshift(fourier_vg))];
    [M,I] = max(abs((fourier_vg)));
    score = [score; abs(k(I))/(2*pi)];
```

```
end

figure()
pcolor(tslide,(ks/(2*pi)),spectrum.'), shading interp
xlabel("Time (s)");ylabel("Frequency (Hz)"); title("Piano");
ylim([100 500])

% Part 2.2
close all; clear all; clc;

[y,Fs] = audioread('music2.wav');
tr_rec=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);
%figure(2)
% Fourier Transform
n = length(y);
k = (2*pi/tr_rec)*[0:n/2-1 -n/2:-1]; ks = fftshift(k);
t = (1:n)/Fs;
tslide = 0:0.2:tr_rec;

spectrum = [];
score = [];

% Gaussian Window and Scor
for i=1:length(tslide)
    g = exp(-100*(t-tslide(i)).^2);
    filter_g = g.*y';
    fourier_vg = fft(filter_g);
    spectrum = [spectrum; abs(fftshift(fourier_vg))];
    [M,I] = max(abs((fourier_vg)));
    score = [score; abs(k(I))/(2*pi)];

end

figure()
pcolor(tslide,(ks/(2*pi)),spectrum.'), shading interp,colormap(hot)
xlabel("Time (s)");ylabel("Frequency (Hz)"); title("Piano");
ylim([600 1300])
```

7. Appendix B,Functions Used

fftn(x) = computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm.

fftshift(X) rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.