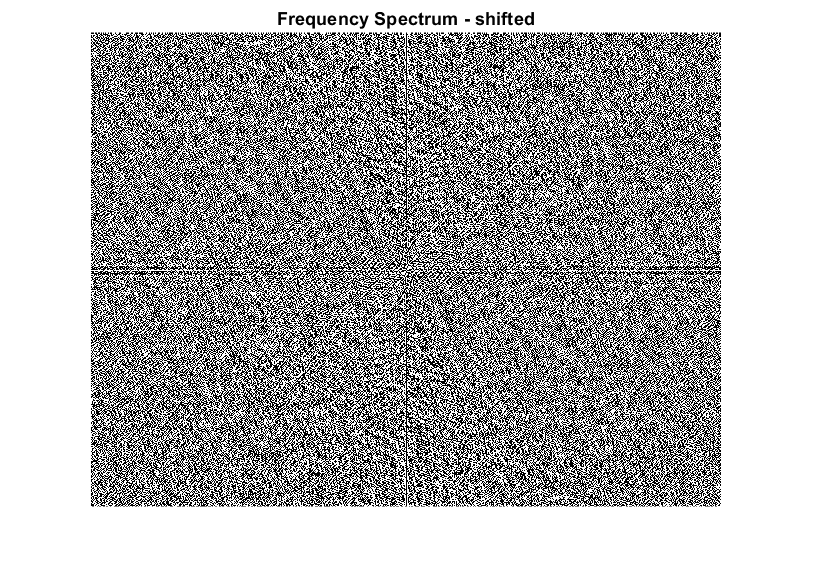
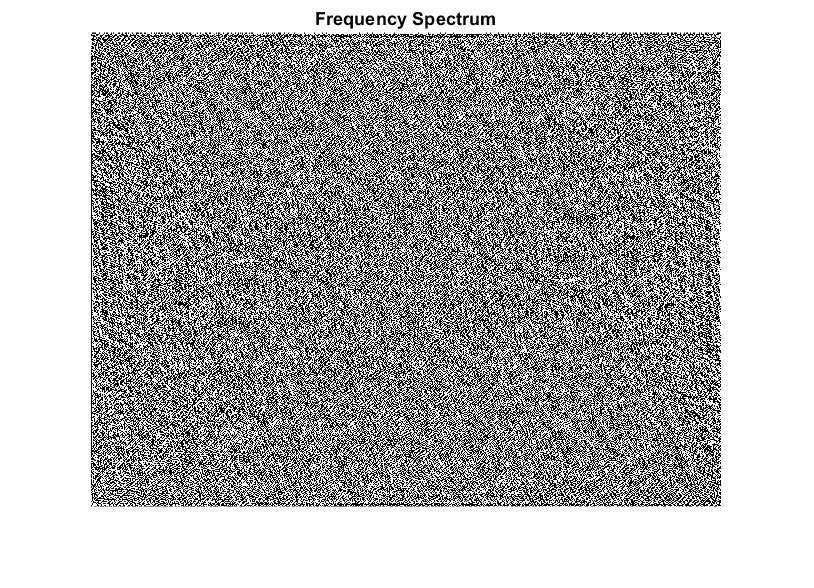
**Image Processing**

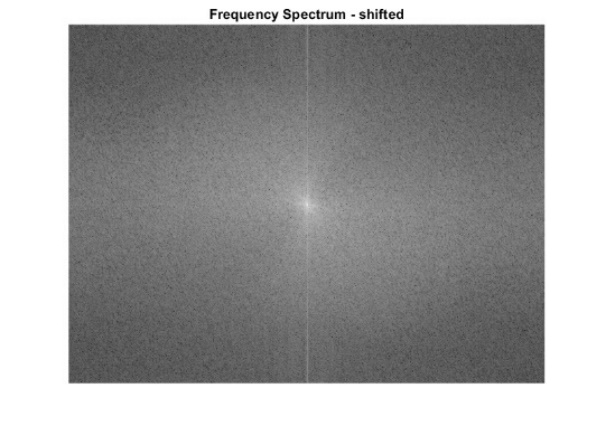
Mohammad Sorkhian

In this assignment, we get familiar with frequency and spatial domains and the conversion between these two. Also, learn how to filter periodic noise in the frequency domain with band-reject filters. All the codes have been included at the end of the report.

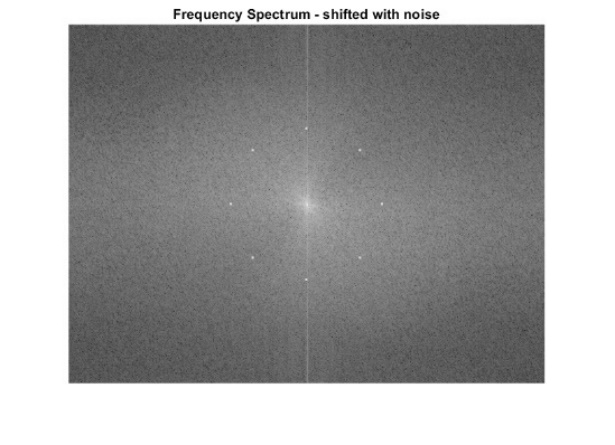
1. I converted the image to the frequency domain and centered it with and functions, respectively. After that, calculated which is 23,229,905.



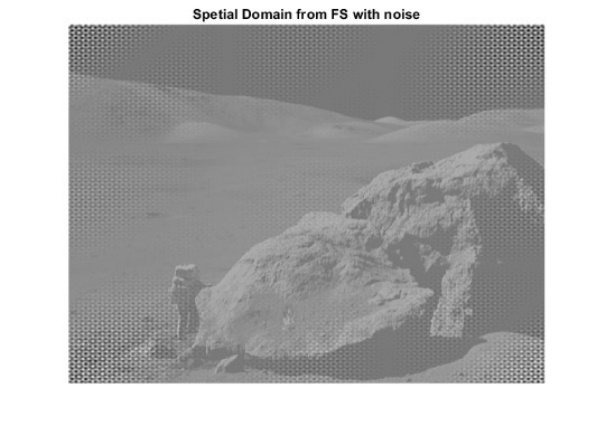
1. I benefited from the logarithmic function for displaying the frequency spectrum.



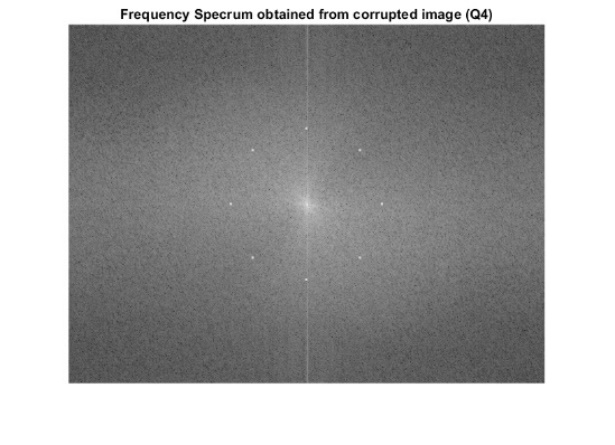
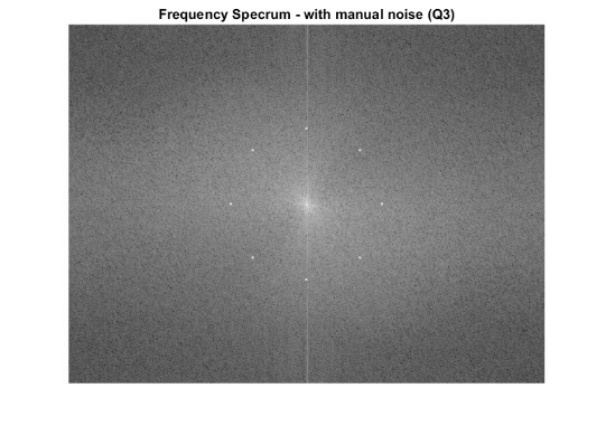
1. In this section, I manually introduced noise on centered frequency spectrum at mentioned points with value of , and represented it with the benefit of logarithmic function.



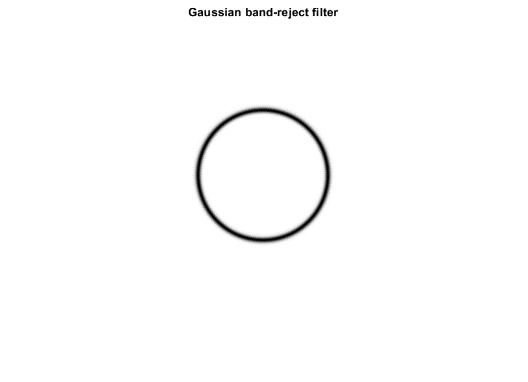
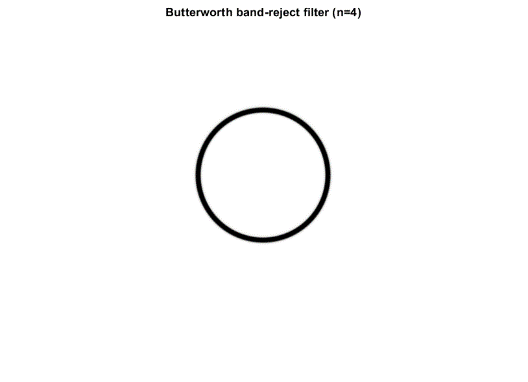
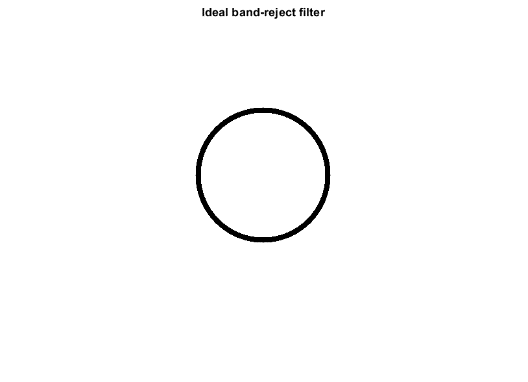
1. I converted original and noisy frequency spectrums back to the spatial domain. It can be seen in converted original frequency domain to the spatial domain we obtain the same image as the original one. But for the corrupted frequency spectrum with noise, we have periodic noise in the image. Also, the shape of noise in the picture relatively represents the shape of noise in the frequency spectrum image. Interestingly, with representing relatively small noise, we could lose a significant amount of details.



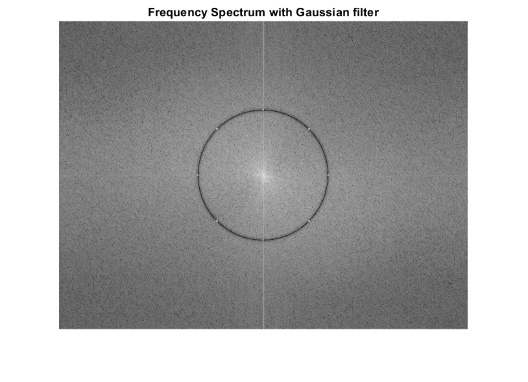
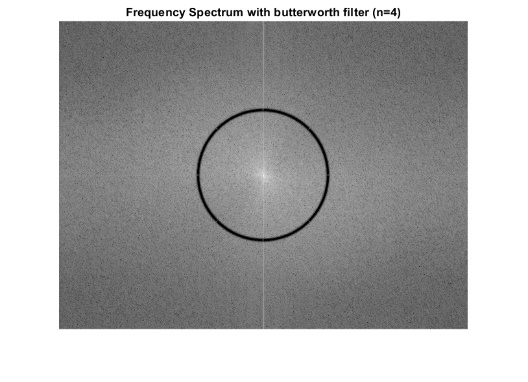
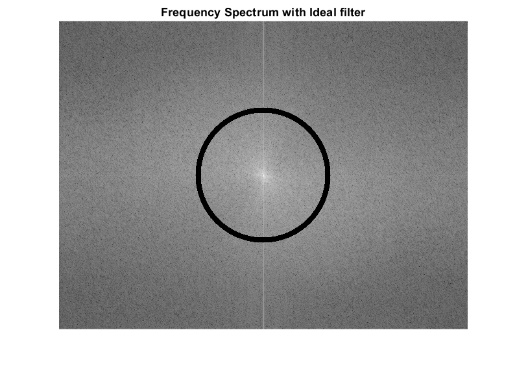
1. In this part, I obtained the FS of the corrupted image with noise from Q4 and compared it with FS with manual noise. I saw that these two images are similar to each other.



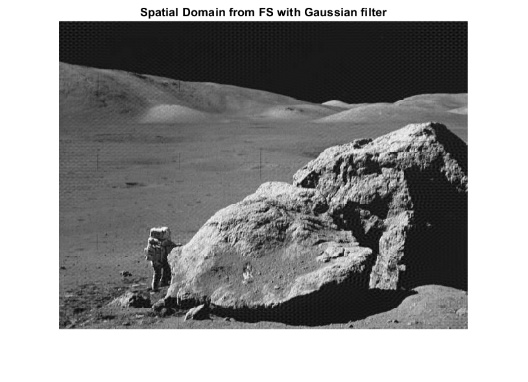
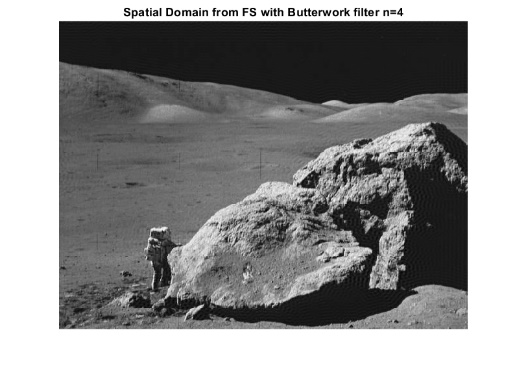
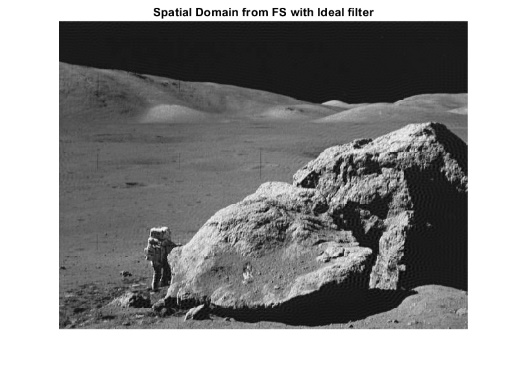
1. I designed three band-reject filters (Ideal, Butterworth, and Gaussian) based on given formulas.



1. I applied the designed filters on FS with noise.



1. In the following, I have converted band-rejected frequency spectrums to the spatial domain.
2. All three filters were able to remove noises very well. We can see many more details in the filtered image compared to the noisy image. Among these three filters the best result in reference to repetitive noise reduction is from the ideal filter and after that Butterwork.
3. Compared to the original image, we can see the ringing effect in the Ideal and Butterwork filter, but in the image that was filtered with the Ideal filter we have more probability of ringing effect because of sharp transition in its cutoff frequency. There is some remained repetitive noise in the Gaussian filter. The reason for that is because the Gaussian filter has a very smooth cutoff frequency region and has not been able to eliminate the introduced noise completely.



1. As we can see Butterwork filter with n=4 is representing behavior relatively similar to Ideal filter, and in both, we can see the ringing effect, but as we decrease n, this effect will become less in the Butterwork filter. This ringing effect is because of the sharp transition between passed frequencies and filtered frequencies. Although we have not ringing effect in the Gaussian filter, we can still see repetitive noise since it was not able to eliminate introduced noise in the frequency spectrum completely because, in this filter, the transition between passed and filtered frequencies is very smooth.

**Matlab Code:**

clear all

img =double(imread('moonlanding.png')); % Read the image

%%%%% part 01 %%%%%

fr = fft2(img); % Convert the image to frequency domain

fr\_sh = fftshift(fr); % Center the spectrom

FS\_max = max(fr\_sh(:)); % Calculate the maximum value of the frequency spectrum

% figure, imshow(fr), title("Frequency Spectrum");

% figure, imshow(fr\_sh), title("Frequency Spectrum - shifted");

%%%%% part 02 %%%%%

fr\_sh\_dis = log(1 + abs(fr\_sh)); % Convert to log for displaying the result

% figure, imshow(fr\_sh\_dis,[]), title("Frequency Spectrum - shifted") ;

%%%%% part 03 %%%%%

% Introdce noise to the FS

[hight,width] = size(fr\_sh);

Center = [hight/2 width/2];

D = 100;

deg = pi/4;

fs\_max = FS\_max/10;

N = [Center(1)-D Center(2)];

E = [Center(1) Center(2)+D];

W = [Center(1) Center(2)-D];

S = [Center(1)+D Center(2)];

NE = [round(Center(1)-(sin(deg)\*D)), round(Center(2)+(cos(deg)\*D))];

NW = [round(Center(1)-(sin(deg)\*D)), round(Center(2)-(cos(deg)\*D))];

SE = [round(Center(1)+(sin(deg)\*D)), round(Center(2)+(cos(deg)\*D))];

SW = [round(Center(1)+(sin(deg)\*D)), round(Center(2)-(cos(deg)\*D))];

fr\_sh\_noise = fr\_sh;

for x = (1:8)

points = [N; E; W; S; NE; NW; SE; SW];

P = points(x,:);

fr\_sh\_noise = noise\_m(fr\_sh\_noise, P, fs\_max);

end

fr\_sh\_noise\_dis = log(1 + abs(fr\_sh\_noise)); % Convert to log for displaying the result

% figure, imshow(fr\_sh\_noise\_dis, []), title("Frequency Spectrum - shifted with noise");

%%%%% part 04 %%%%%

fr\_sh\_sh = fftshift(fr\_sh); % Shift the centered FS back

fr\_sh\_sh\_noise = fftshift(fr\_sh\_noise); % Shift the centered FS with noise back

img\_fr\_sh\_sh = ifft2(fr\_sh\_sh); % Convert FS to Spatial Domain

img\_fr\_sh\_sh\_noise = ifft2(fr\_sh\_sh\_noise); % Convert FS with noise to Spatial Domain

% figure, imshow(img\_fr\_sh\_sh,[]), title('Spetial Domain from FS');

% figure, imshow(img\_fr\_sh\_sh\_noise,[]), title('Spetial Domain from FS with noise');

%%%% part 05 %%%%%

fr\_4 = fft2(img\_fr\_sh\_sh\_noise); % Convert corrupted image in q4 to FS

fr\_4\_sh = fftshift(fr\_4); % Center the FS

fr\_4\_sh\_dis = log(1 + abs(fr\_4\_sh)); % Convert to log for displaying the result

% figure, imshow(fr\_sh\_noise\_dis, []), title("Frequency Specrum - with manual noise (Q3)");

% figure, imshow(fr\_4\_sh\_dis,[]), title("Frequency Specrum obtained from corrupted image (Q4)");

%%%% part 06 %%%%%

D0 = 100;

W = 8;

[hight,width]=size(img);

filter\_ideal = ones(hight,width);

x0 = round(width/2);

y0 = round(hight/2);

for j = 1:hight % Design Ideal-band-reject filter

for i = 1: width

ii = i-x0;

jj = j-y0;

D = sqrt(ii^2+jj^2);

if (D0-W/2) <= D && D <= (D0+W/2)

filter\_ideal(j,i) = 0;

end

end

end

filter\_btw = ones(hight,width);

n = 4;

for j = 1:hight % Design Butterworth-band-reject filter

for i = 1: width

ii = i-x0;

jj = j-y0;

D = sqrt(ii^2+jj^2);

filter\_btw(j,i) = 1/(1+(((D\*W)/(D^2-D0^2))^(2\*n)));

end

end

filter\_gaussian = ones(hight,width);

for j = 1:hight % Design Gaussian-band-reject filter

for i = 1: width

ii = i-x0;

jj = j-y0;

D = sqrt(ii^2+jj^2);

filter\_gaussian(j,i) = 1-exp(-(((D^2-D0^2)/(D\*W))^2));

end

end

% figure, imshow(filter\_ideal), title("Ideal band-reject filter");

% figure, imshow(filter\_btw), title('Butterworth band-reject filter (n=4)');

% figure, imshow(filter\_gaussian), title('Gaussian band-reject filter');

%%%% part 07 %%%%%

% Applying designed filters on the FS with noise

fr\_sh\_rm\_noise\_ideal = fr\_sh\_noise.\*filter\_ideal;

fr\_sh\_rm\_noise\_btw = fr\_sh\_noise.\*filter\_btw;

fr\_sh\_rm\_noise\_gaussian = fr\_sh\_noise.\*filter\_gaussian;

% figure, imshow(log(1 + abs(fr\_sh\_rm\_noise\_ideal)),[]), title('Frequency Spectrum with Ideal filter');

% figure, imshow(log(1 + abs(fr\_sh\_rm\_noise\_btw)),[]), title('Frequency Spectrum with butterworth filter (n=4)');

% figure, imshow(log(1 + abs(fr\_sh\_rm\_noise\_gaussian)),[]), title('Frequency Spectrum with Gaussian filter');

%%%% part 08 %%%%%

% Convert filtered FSs to the spatial domain

fr\_rm\_noise\_ideal = fftshift(fr\_sh\_rm\_noise\_ideal);

fr\_rm\_noise\_btw = fftshift(fr\_sh\_rm\_noise\_btw);

fr\_rm\_noise\_gaussian = fftshift(fr\_sh\_rm\_noise\_gaussian);

img\_rm\_noise\_ideal = ifft2(fr\_rm\_noise\_ideal);

img\_rm\_noise\_btw = ifft2(fr\_rm\_noise\_btw);

img\_rm\_noise\_gaussian = ifft2(fr\_rm\_noise\_gaussian);

% figure, imshow(img\_rm\_noise\_ideal,[]), title('Spatial Domain from FS with Ideal filter');

% figure, imshow(img\_rm\_noise\_btw,[]), title('Spatial Domain from FS with Butterwork filter n=4');

% figure, imshow(img\_rm\_noise\_gaussian,[]), title('Spatial Domain from FS with Gaussian filter');

function [result] = noise\_m(img, point, value)

% This function gets an image and desired point,

% then apply noise(value) on the 3\*3 neighboring pixels

for j = -1:1

for i =-1:1

img((point(1)+i), (point(2)+j)) = value;

end

end

result = img;

end