Saeed, extensive results v. good!

18/20

IMAGE PROCESSING COURSE (EE6553)

NOVEMBER 20, 2020

SAEED KAZEMI
Student Number 3713280

DUE: NOV 20, 2020

A. Abstract

In this assignment, I worked on a MATLAB program for implementing a Gaussian lowpass filter and Spatial and frequency operations on an image for finding the restored image from a low pass filter. This project contained two functions and the main m-file that call all functions for solving examples. In section next section, the techniques used for this assignment will be discussed. In section C, the output results are checked based on images that are shown in part D, and finally, in the appendix, my codes are added.

B. Technical discussion

In this section, the techniques were used in this assignment are discussed. For the first part of question one, we made a Gaussian noise and added to the image with the *agn_noise* function. In this function, the *randn* function was used to make a Gaussian noise. This function returns a random number with a Gaussian distribution, zero mean, and unit variance. For changing the mean and variance of this number, we can use the below formula.

```
Gaussian Number(Mean, Variance) = Variance * randn() + Mean
```

This formula changes the scale of the gaussian random number. For example, if the Variance is selected a high amount, the noise scale will increase. Due to the input image is a 2D array, we needed to calculate for each pixel a Gaussian number. Therefore the size of the input image was set as an input argument for the randn function. As a result, the output was a matrix as big as the input image size. The below equation indicates this part of the code. In this project, three noise matrixes with different scale (low, medium, and high variance) were produced in order to investigate the impact of level noise.

```
noise = Variance .* randn(size(img)) + mean;
```

In section 2 of the first question, a Gaussian low pass filter was made. For this type of filter, we needed to implement a Gaussian shape on the frequency domain. So, in the first step, the distance of each pixel to the center of the filter was calculated. We considered that the `fftshift` was used after using `fft`. Consequentially all low frequencies were in the center of the image. Then based on the below equation, a value was assigned to the pixel.

```
filter(i,j)= exp(-(D^2)/(2*Cutoff^2))
```

In this equation, Cutoff and D are the cut-off frequency and the distance between pixel (i, j) and the center of filter respectively. Figure 1 shows this filter in the frequency domain. As can be seen, this filter has a large value around the center (low frequencies) and low values at the corners and edges (high frequencies).

With increasing the cut-off frequency, the filter gets bigger and passes more frequencies and vise versa. Another interesting point about this filter is that in the spatial domain, we

have another Gaussian shape, and if we increase the cut-off frequency, the shape of the filter gets more narrow in the spatial domain. For this project, we filtered the input image with two cut-off frequencies to analyze the effect of the cut-off frequency.

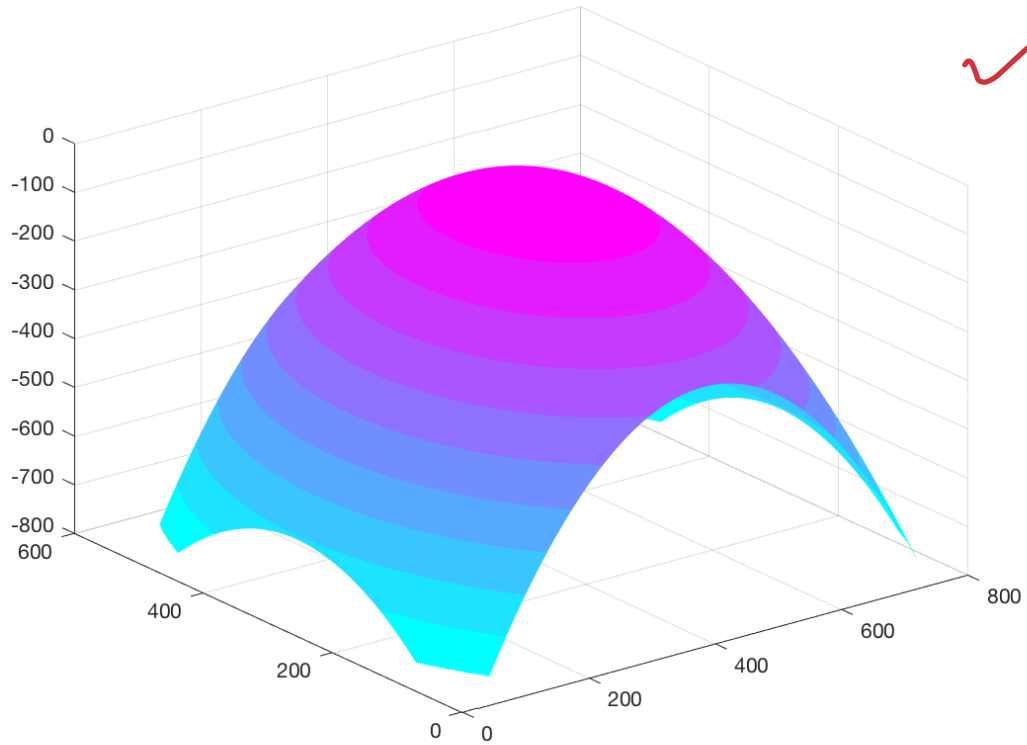


Figure 1) The Gaussian low pass filter.

In the third section of question 1, we filtered the input image with a Gaussian low pass filter in the frequency domain then added noise in the spatial domain. The below diagram depicts different parts of the system.

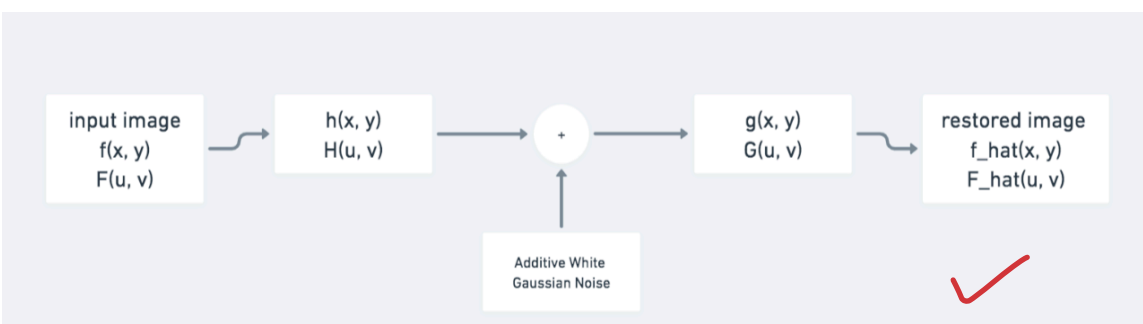


Figure 2) the diagram that used for this assignment

In this section, six images were produced with different noise levels and cut-off frequencies. These images called LL (low cut-off frequency and noise level), LM (low cut-off frequency and medium noise level), LH (low cut-off frequency and high-level noise), HL (high cut-off frequency and low-level noise), HM (high cut-off frequency and medium noise level), and HH (high cut-off frequency and noise level).

In the next section, these six produced images were compared to the input image by the SSIM and MSE index. These two methods are used for measuring the quality of similarity and difference between the two images. The SSIM index is a number between 0 and 1. when two images have the same quality, this index will be 1 and vise versa.
MSE is another measure that indicates the difference between the two images. This measure (due to the square term) is always a non-negative number, and values closer to zero show the quality of the difference is low or the images are more similar. The results of these images will review in the next section of this document.

$$\hat{F}(u,v) = \left[\frac{1}{\tilde{H}(u,v)}\left[\frac{\left|\tilde{H}(u,v)\right|^2}{\left|\tilde{H}(u,v)\right|^2 + K}\right]\right]G(u,v)$$

In the second question, the spectrum of the output images was calculated. Also for intuition, a 1D slice of this spectrum was shown. this image shows the effect of the noise level and the cut-off frequency in the spectrum of the output image.
In the second and third part of this question, we tried to restore the original image from the output images with two types of method, inverse, and Wiener filter. Based on the system, the output image computed

g(x, y) = h(x, y)*f(x, y) + n(x, y)

where * is 2-D convolution, h(x, y) is a Gaussian low pass filter, f(x, y) is the original image, and n(x, y) is AWGN. In the ideal case, we can invert all the elements of H (the Fourier transform of h) to get a high pass filter. However, as noticed, a lot of the elements in H have values either at zero or very close to it. Inverting these elements would give us either infinities or some extremely high values. In other words, the restored image will be exploded. To avoid this effect, there are two solutions, thresholding, and limitation. In thresholding instead of making a full inverse matrix, we inverse the low-frequency part of H. in other words, for values lower than this threshold, we return this threshold. The below equation depicts this method.

$$H(u,v) = \begin{cases} H(u,v) & H(u,v) > Threshold \\ Threshold & else \end{cases}$$

Another interesting thing about this method is that if H is a low pass filter, the inverse H is a high pass filter. Figure 3 shows the inverse filter H.

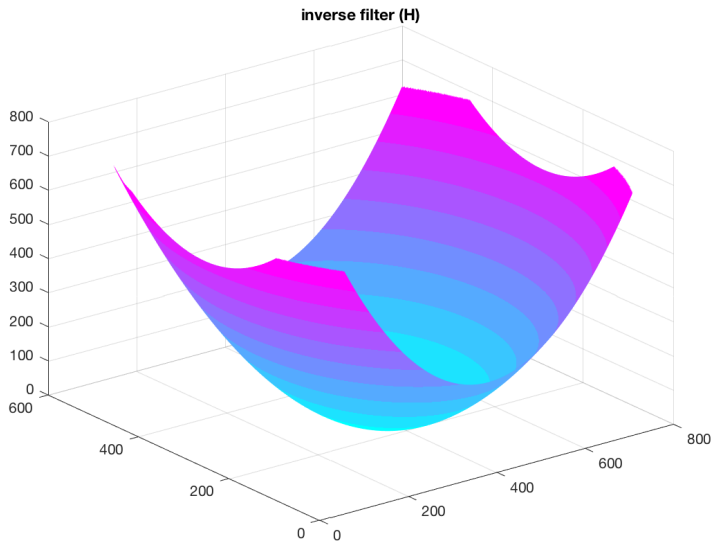


Figure 3) the inverse low pass filter

Although the inverse filter is the simplest solution to the deblurring, it is very sensitive to additive noise. Another method is the Wiener filter. The below equation indicates the formula of this filter. As you can see, there is a k parameter in this equation that takes a positive value. If we choose a small value for this parameter, this filter will turn into an inverse filter. Also when a big number is selected, the restored image will be a zero matrix. This parameter adjusts a state between inverse filtering and noise smoothing. In other words, the Wiener filter removes the AWGN and deblurring simultaneously.

$$\hat{F}(u,v) = \left[\frac{1}{\tilde{H}(u,v)}\left[\frac{\left|\tilde{H}(u,v)\right|^2}{\left|\tilde{H}(u,v)\right|^2 + K}\right]\right]G(u,v)$$

C. Discussion of results

In order to see the effect of the noise in the original image, we implemented three types of Gaussian noise. With increasing the scale, the quality of the output image has been reduced. Figure 10 to Figure 13 show these effects. Figure 14 shows only the medium scale of Gaussian noise. We can show this scale as an SNR parameter. If this parameter is a large value, the effect of the noise on the original image is low and vise versa. Also, these three noisy matrixes had some differences in terms of their element ranges. The high variance had a high range changes for its elements and vise versa.

In the next section, we applied two low pass filters with different cut-off frequencies. As it is obvious, if the filter had a low cut-off frequency, the output image is blurry. Also in the spatial domain, we have a wide Gaussian shape (see Figure 18).
The below image shows the result of the Gaussian filter with a large cut-off frequency. As expected, the output image was less blurry than the low cut-off frequency filter. Also, the filter in the spatial domain was more narrow than the low cut-off frequency filter. Compare Figure 18 and Figure 16. We can interpret when the cut-off frequency is selected a large number, the more frequency could appear at the output and we have more detail.

Figure 4) filtered image with a low pass Gaussian filter. The cut-off frequency was 11.

In the next section, six images were produced based on the noises matrixes and filtered images. Then we compared them with original images. For comparing these images, we used SSIM and MSE indexes. The below table shows two indexes for our images.

good

The SSIM index	Low-level noise	Medium level noise	High-level noise
Cut-off frequency = 2	0.0222	0.0033	0.0007
Cut-off frequency = 11	0.1338	0.0414	0.0011

The MSE index	Low-level noise	Medium level noise	High-level noise
Cut-off frequency = 2	1834.9528	1932.1619	11833.2415
Cut-off frequency = 11	758.6281	856.4187	10743.1971

As you can see, with increasing the variance of the noise, the SSIM index has been decreased. This index shows that the similarity of the noisy image with the original image goes down with increasing the scale of noise. Also with increasing noise, the MSE index that shows the difference had been increased.
Increasing the cut-off frequency has a different effect in comparison with the noise level. When the cut-off frequency increased, the similarity increased and the MSE declined. In summary, if either increasing the scale of noise or decreasing the cut-off frequency could increase the similarity or decrease the difference quality of two images.

In the next question, we desire to restore the input image from the output images. In the first step, we look at their spectrums and a 1D slice of these spectrums. This 1-D slice of the 2-D estimate of the power spectrum could show some information related to the level of noise and the cut-off frequency of the filter. For example, the below images are from high noise level and cut-off frequency (Figure 5) and low noise level and cut-off frequency (Figure 6).

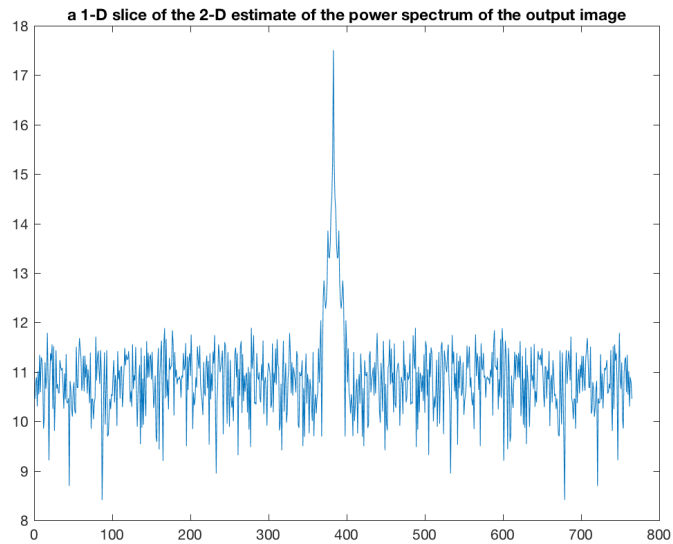


Figure 5) a 1-D slice of the 2-D estimate of the power spectrum of the output image, with a high level of noise and cut-off frequency

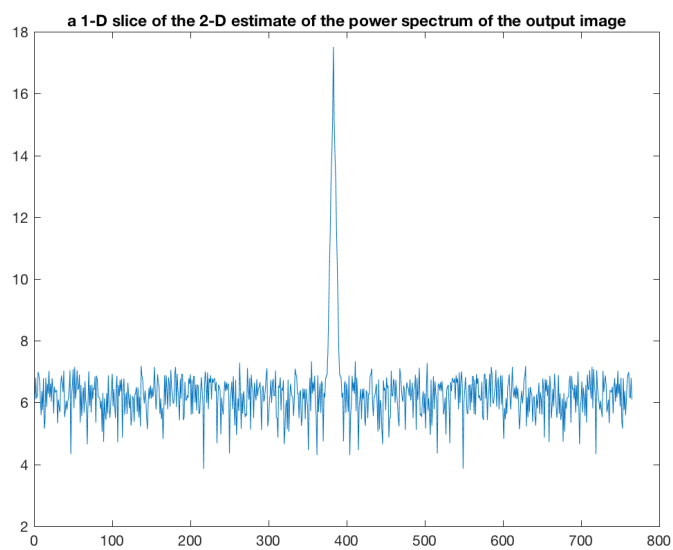


Figure 6) a 1-D slice of the 2-D estimate of the power spectrum of the output image, with a low level of noise and cut-off frequency

As it is obvious, if the level of noise increases, we have a high range of changes in the high-frequency part of this plot (around 0 and 700). Also if the cut-off frequency increases, the

pick of this will get wider. The 2-D estimate of the power spectrums and their slices for six output images are shown in Figure 25 to Figure 36.
For restoring the image with an inverse filter, we encountered an exploded image (Figure 7).

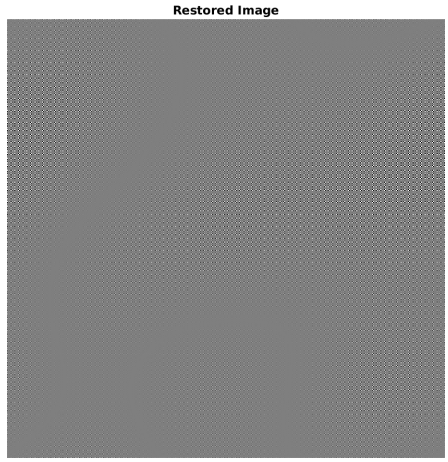


Figure 7) exploded image

Therefore for the rest of the test, we used a threshold. Figure 37 to Figure 42 show the restored images from this method. The SSIM and MSE index for these images and the original image are shown in the below table.

Level of noise	Cut-off frequency	The SSIM index	The MSE index
Low	11	0.1743	625.1319
medium	11	0.1597	634.6517
high	11	0.0576	1692.1836
Low	2	0.0481	1499.7430
medium	2	0.0477	1500.0315
high	2	0.0485	1550.1345

As this table shows, with either increasing the noise or decreasing the cut-off frequency, the SSIM index went down. In other words, the restored image did not have a similar quality to the original image. Also, since some of the frequencies have not been restored (effect of thresholding), there is some distortion in the restored images.

For the Wiener filter, we needed to select k for the filter. One of the methods could be to do a repetition on k, until finding the best result and return the k. For this project, because we knew the noise and image spectrum, we could find the k from the below formula.

$$k = \frac{|N(u,v)|^2}{|F(u,v)|^2}$$

Should this be a number?

Figure 43 to Figure 48 depict the out images for this filter. Also, the below table compares the quality of similarity between the output image and the original. As it is obvious, the Wiener filter could restore the original image better than the inverse filter.

are MSE or SSIM numbers better?

Level of noise	Cut-off frequency	The SSIM index	The MSE index	K
Low	11	0.2085	510.6087	7e-5
medium	11	0.1715	604.4736	0.007
high	11	0.0846	3169.8606	0.7025
Low	2	0.0628	1203.4512	7e-5
medium	2	0.0540	1442.8873	0.0070
high	2	0.0250	4160.5341	0.7025

These two images show the output of inverse (the first image) and Wiener filter (the second image). As expected, the quality of the second image was better than the first one.

Which test case did you use?

Figure 8) the output of the inverse filter

Figure 9) the output of the Wiener filter

D. Image Results

Figure 10) the original image

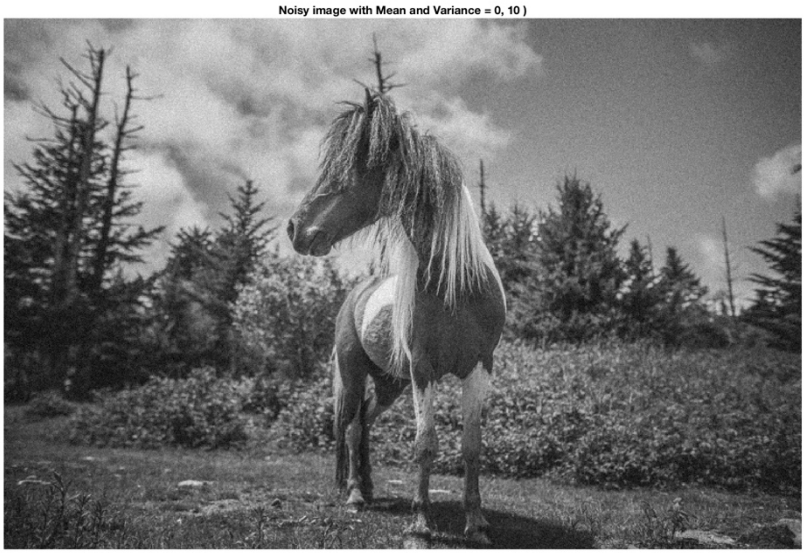


Figure 11) The original image with medium variance noise

Figure 12) The original image with low variance noise

Figure 13) The original image with high variance noise

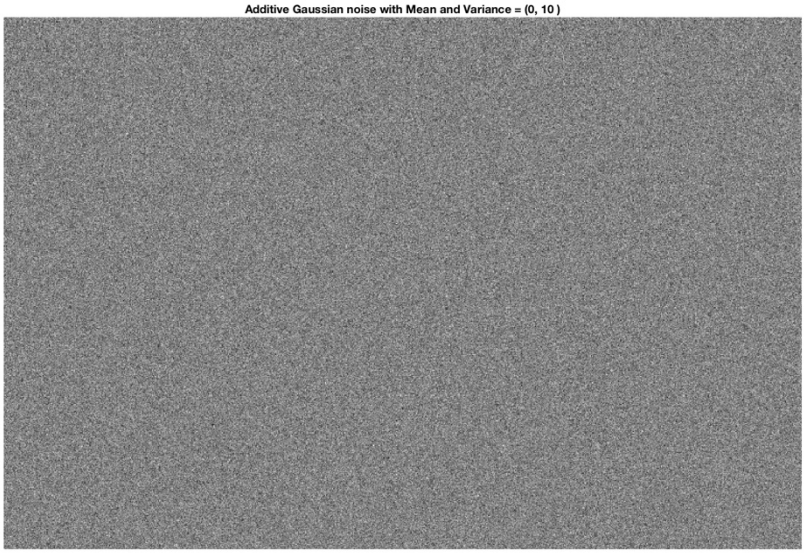
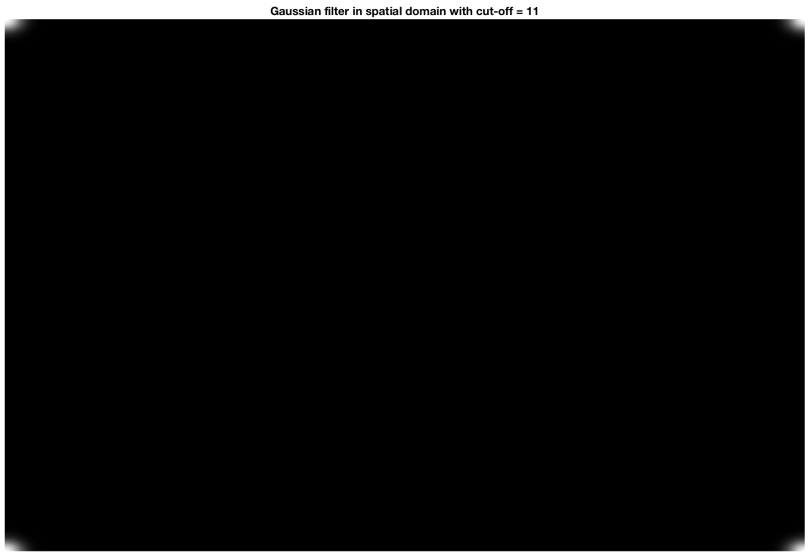


Figure 14) The Gaussian noise with a medium variance

Figure 15) filtered image with a low pass Gaussian filter. The cut-off frequency was 11.



?

Figure 16) the image Gaussian low pass filter in the spatial domain. the cut-off frequency was 11.

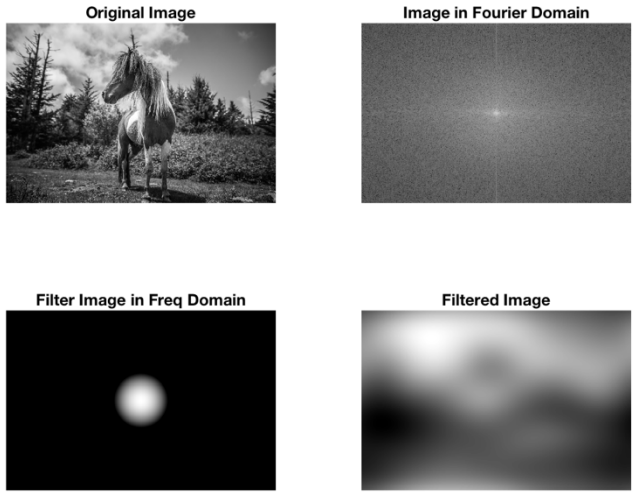
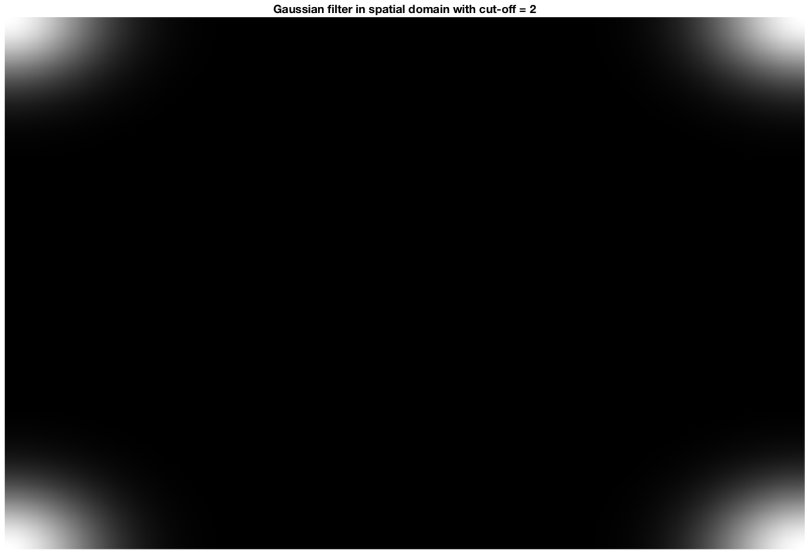
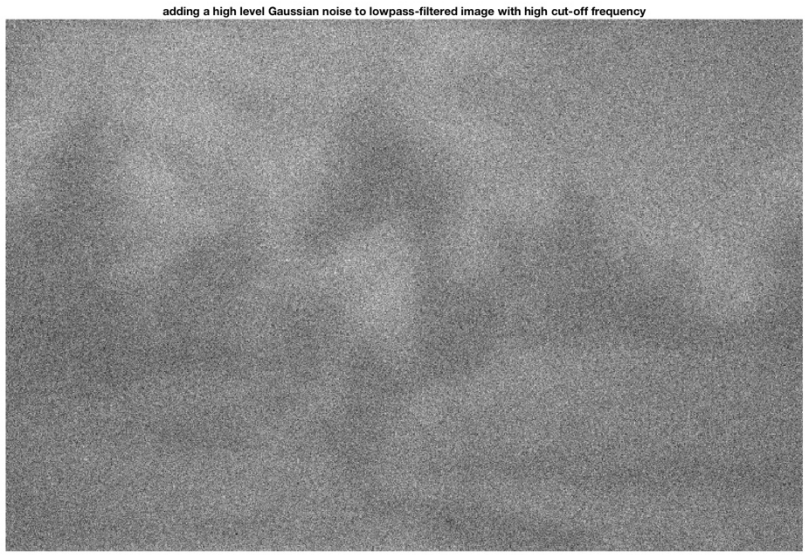


Figure 17) filtered image with a low pass Gaussian filter. The cut-off frequency was 2.



ok

Figure 18) the image Gaussian low pass filter in the spatial domain. the cut-off frequency was 2.



✓

Figure 19) Adding a high-level Gaussian noise to a lowpass-filtered image with a high cut-off frequency

Figure 20) Adding a medium level Gaussian noise to lowpass-filtered image with a high cut-off frequency

Figure 21) Adding a low-level Gaussian noise to a lowpass-filtered image with a high cut-off frequency

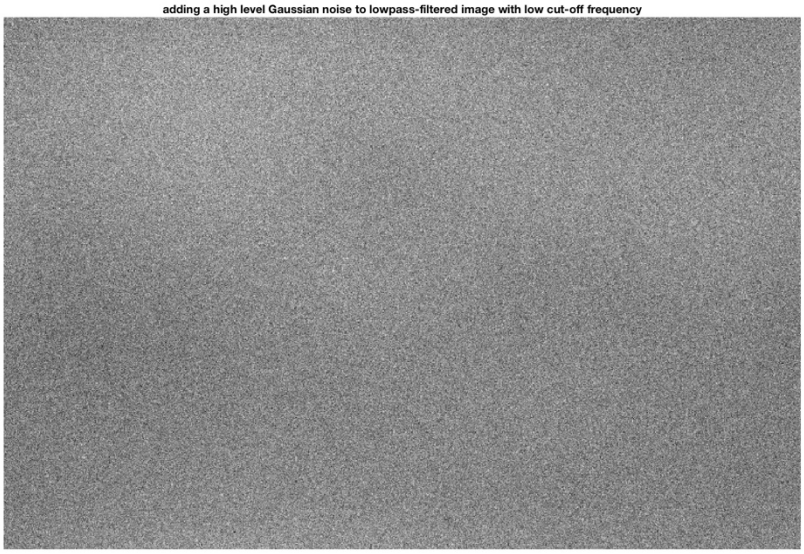


Figure 22) Adding a high-level Gaussian noise to a lowpass-filtered image with a low cut-off frequency

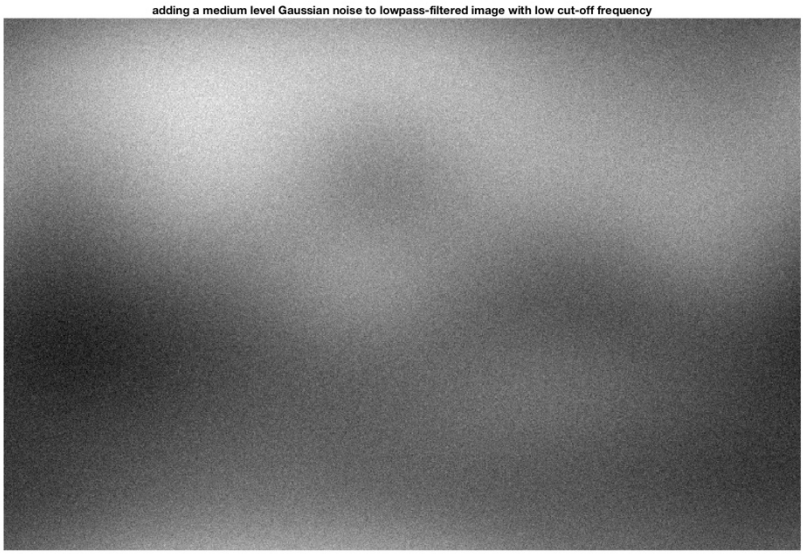


Figure 23) Adding a medium level Gaussian noise to lowpass-filtered image with h low cut-off frequency

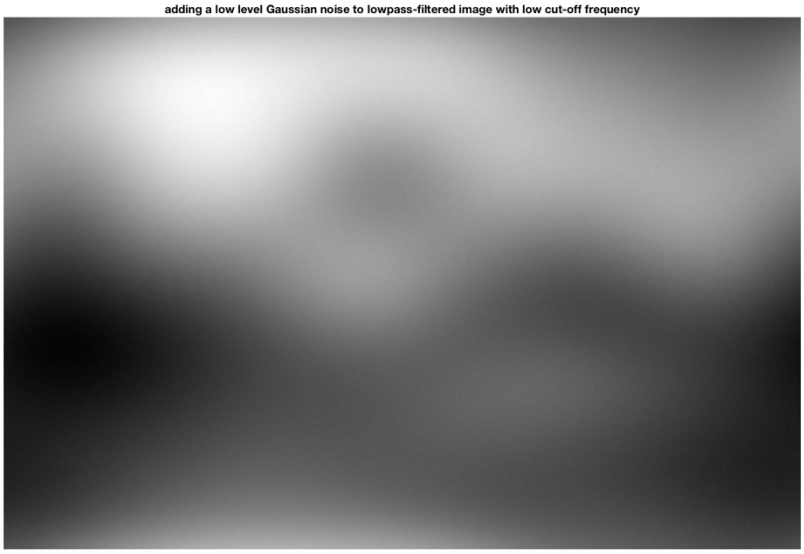


Figure 24) Adding low-level Gaussian noise to lowpass-filtered image with a low cut-off frequency

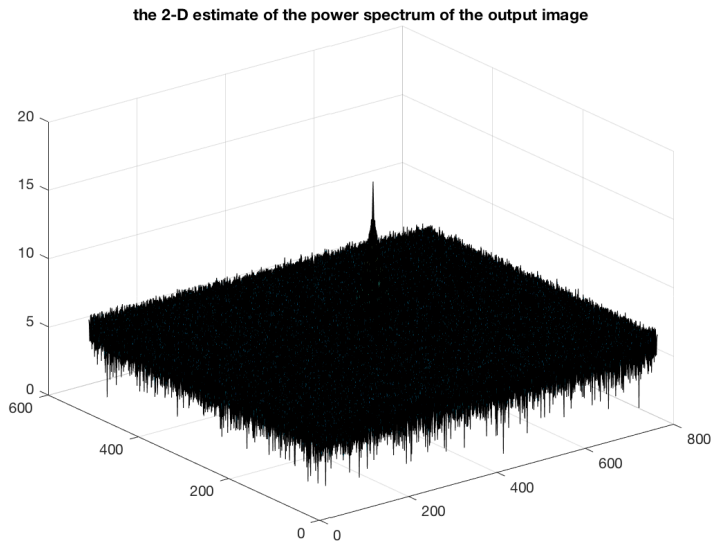


Figure 25) the 2-D estimate of the power spectrum of the output image HL

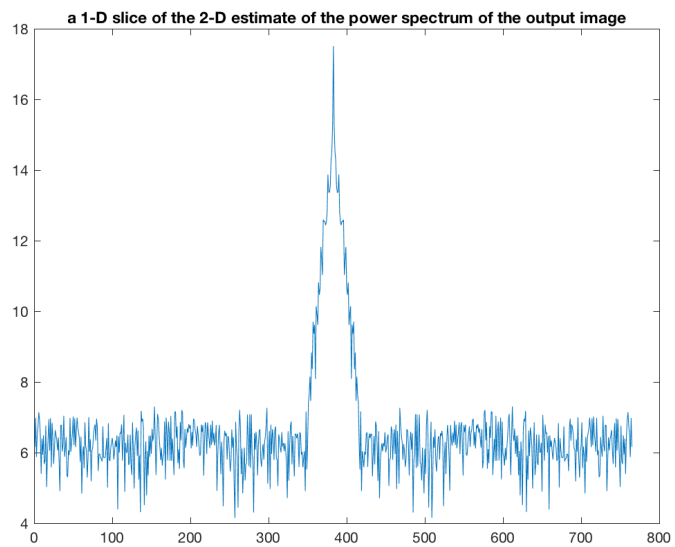


Figure 26) a 1-D slice of the 2-D estimate of the power spectrum of the output image HL

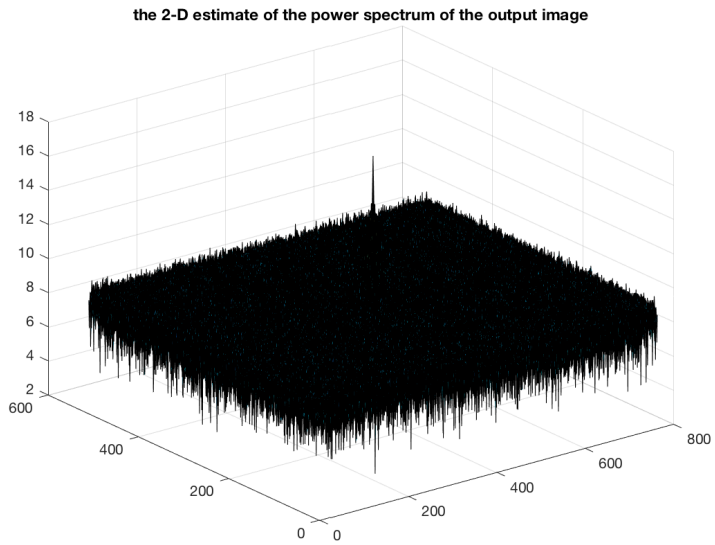


Figure 27) the 2-D estimate of the power spectrum of the output image HM

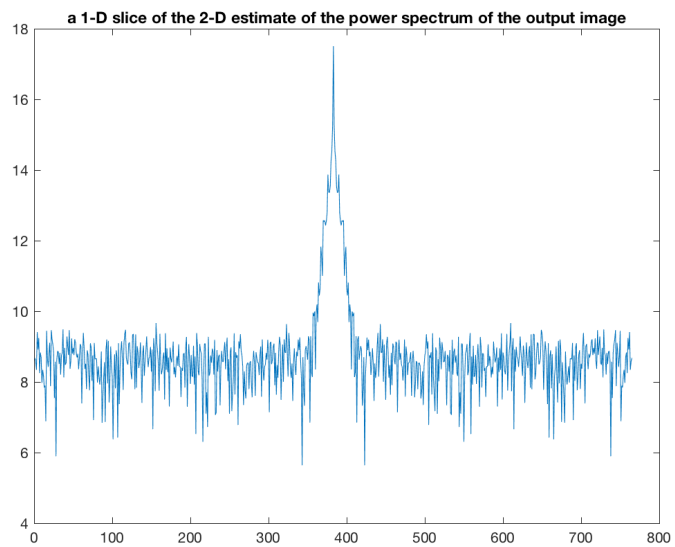


Figure 28) a 1-D slice of the 2-D estimate of the power spectrum of the output image HM

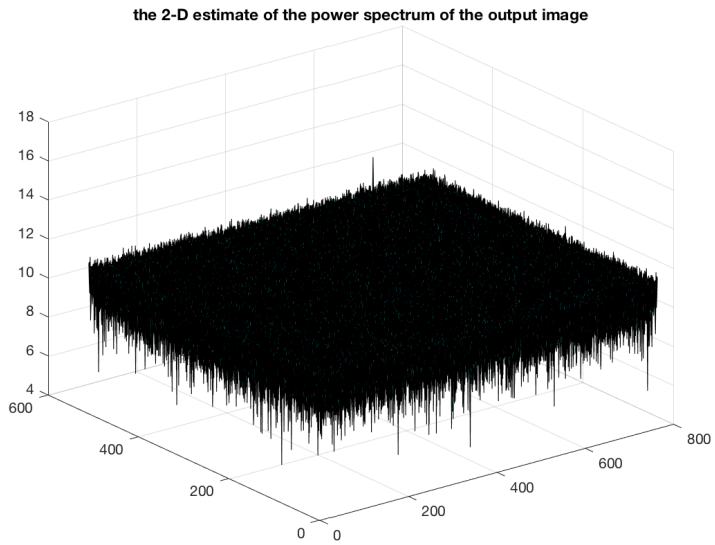


Figure 29) the 2-D estimate of the power spectrum of the output image HH

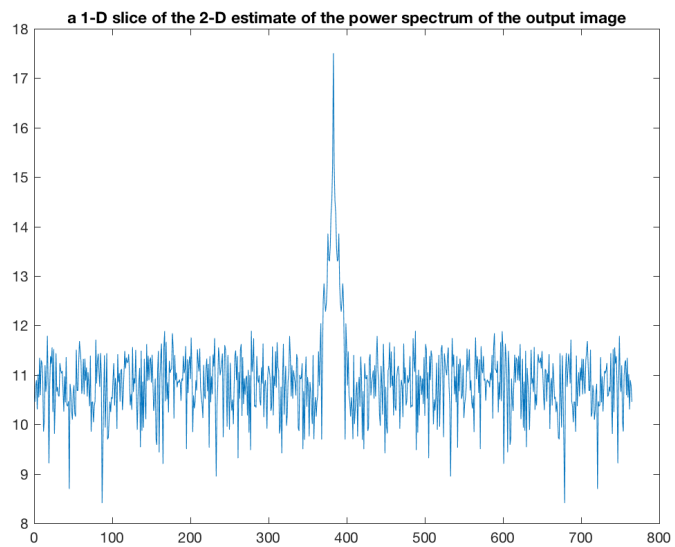


Figure 30) a 1-D slice of the 2-D estimate of the power spectrum of the output image HH

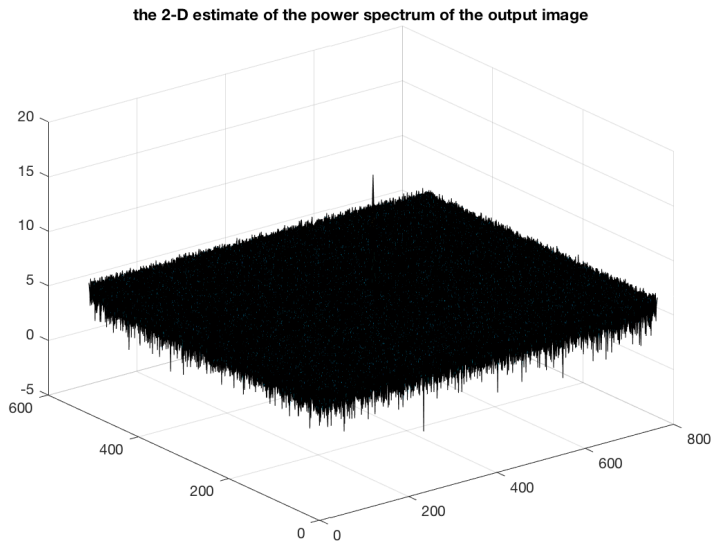


Figure 31) the 2-D estimate of the power spectrum of the output image LL

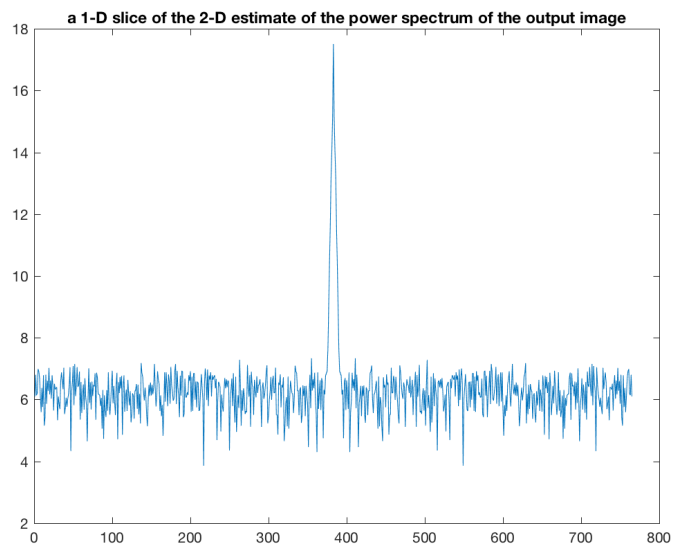


Figure 32) a 1-D slice of the 2-D estimate of the power spectrum of the output image LL

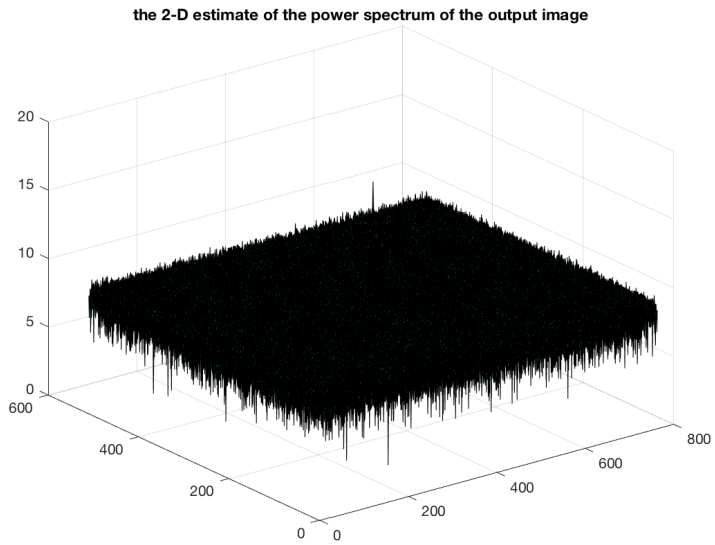


Figure 33) the 2-D estimate of the power spectrum of the output image LM

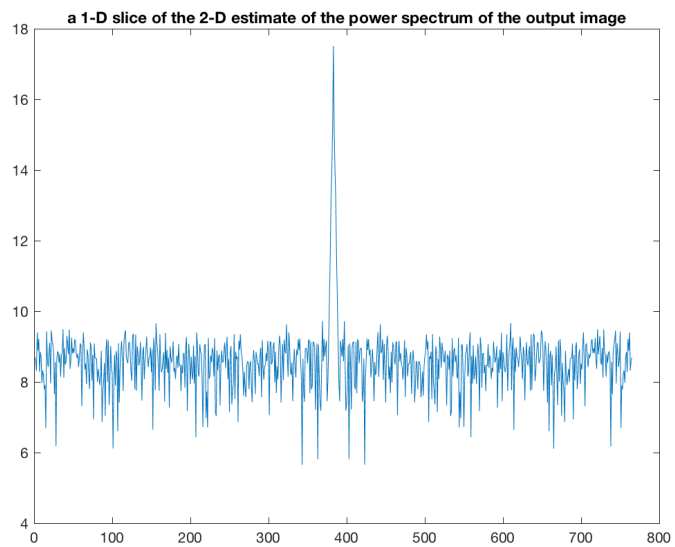


Figure 34) a 1-D slice of the 2-D estimate of the power spectrum of the output image LM

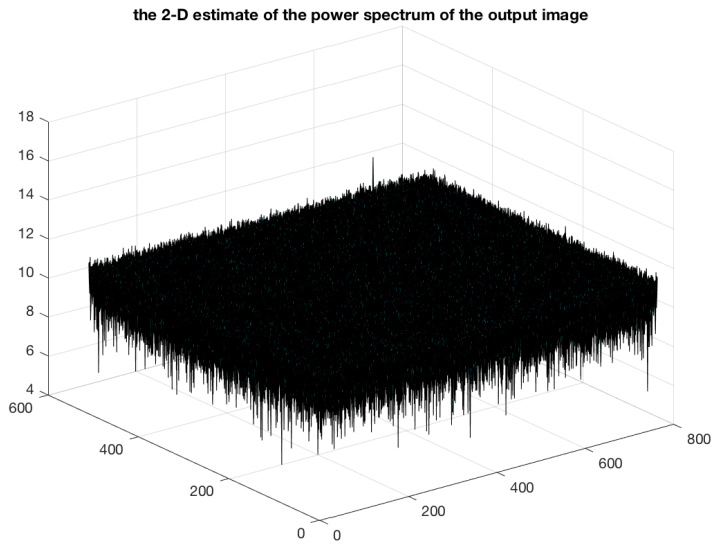


Figure 35) the 2-D estimate of the power spectrum of the output image LH

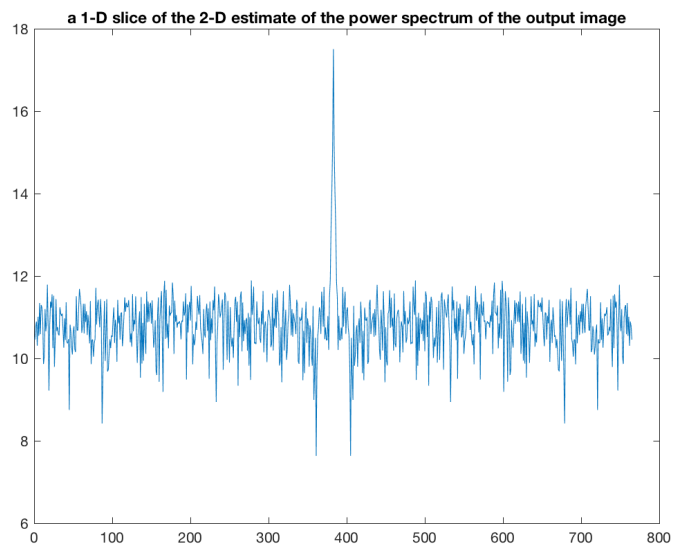


Figure 36) a 1-D slice of the 2-D estimate of the power spectrum of the output image LH

Inv. filtering

Figure 37) restored image from high cut-off frequency and low-level noise

Figure 38) restored image from high cut-off frequency and medium level noise

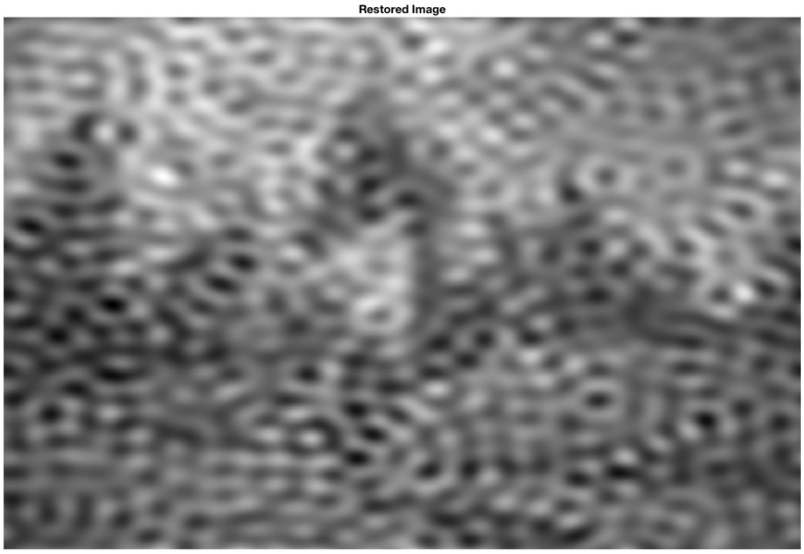


Figure 39) restored image from high cut-off frequency and high-level noise

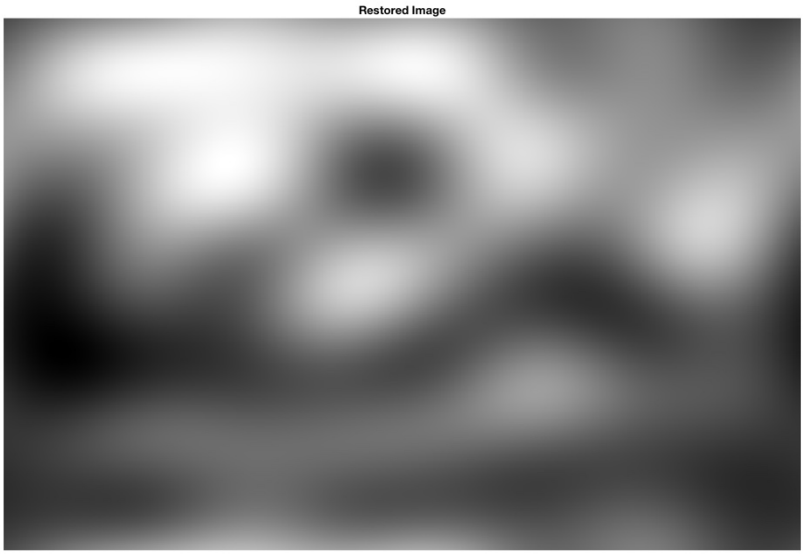


Figure 40) restored image from low cut-off frequency and low-level noise

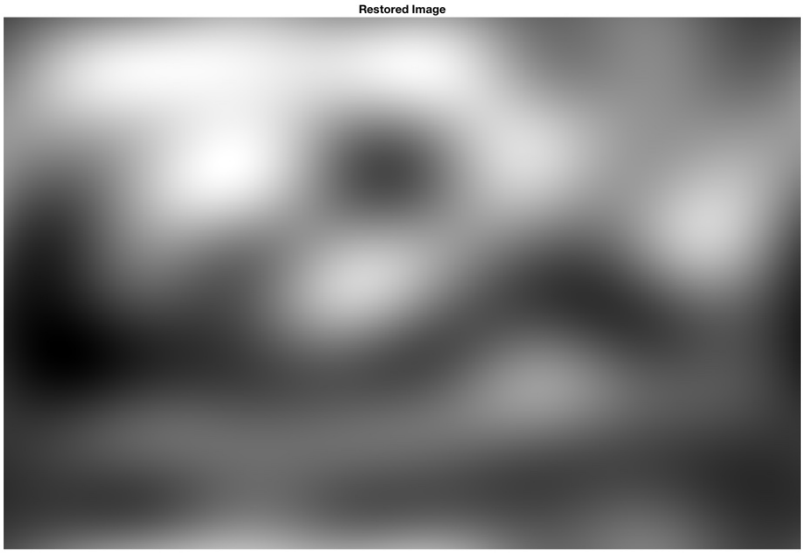


Figure 41) restored image from low cut-off frequency and medium level noise

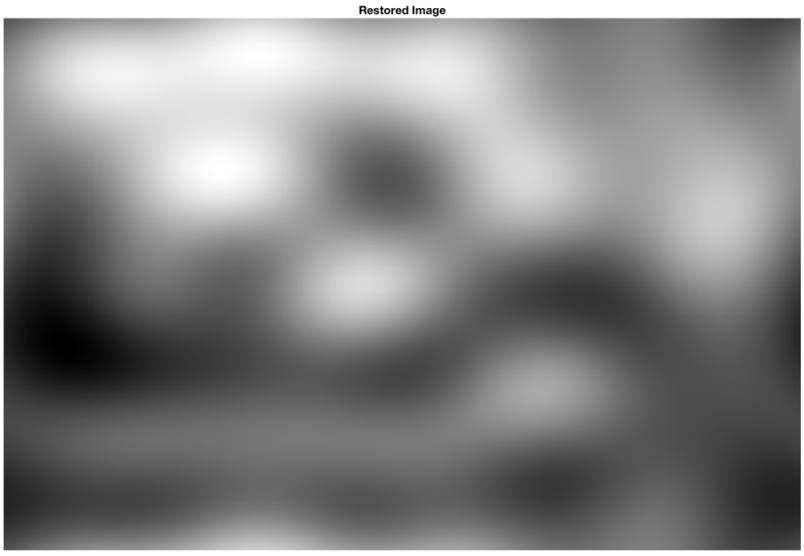


Figure 42) restored image from low cut-off frequency and high-level noise

Figure 43) restored image from high cut-off frequency and low-level noise

Figure 44) restored image from high cut-off frequency and medium level noise

Figure 45) restored image from high cut-off frequency and high-level noise

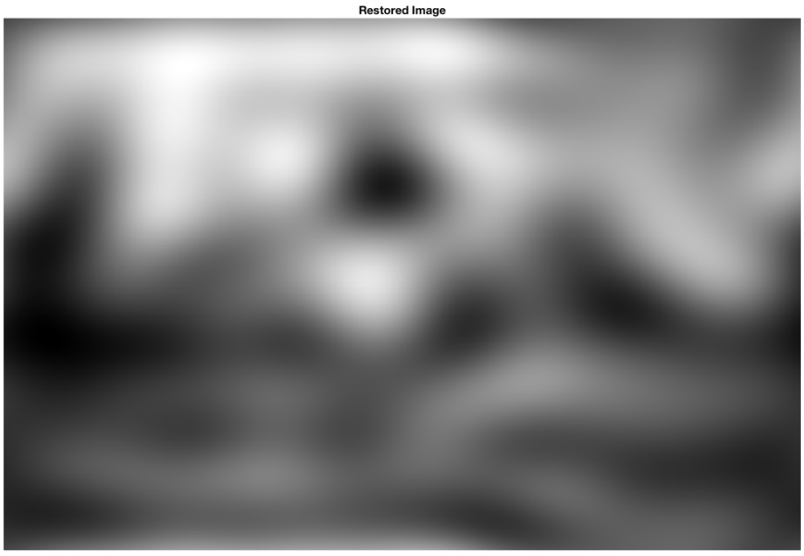


Figure 46) restored image from low cut-off frequency and low-level noise

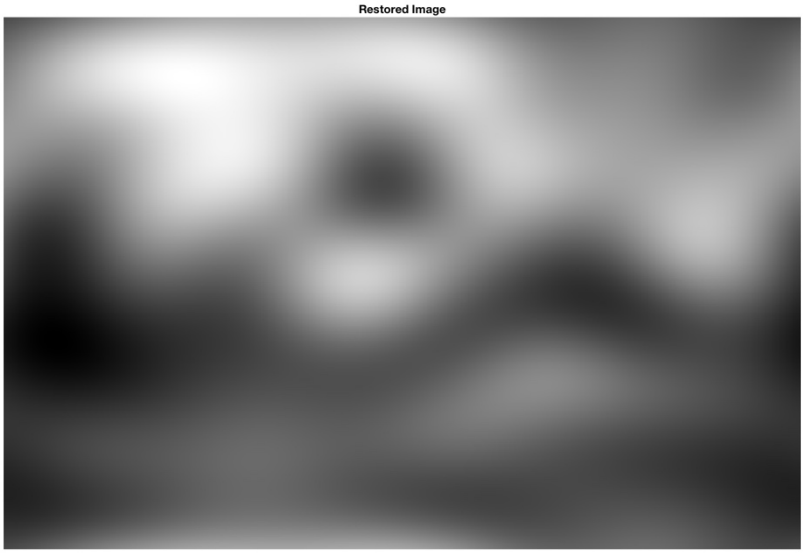


Figure 47) restored image from low cut-off frequency and medium level noise

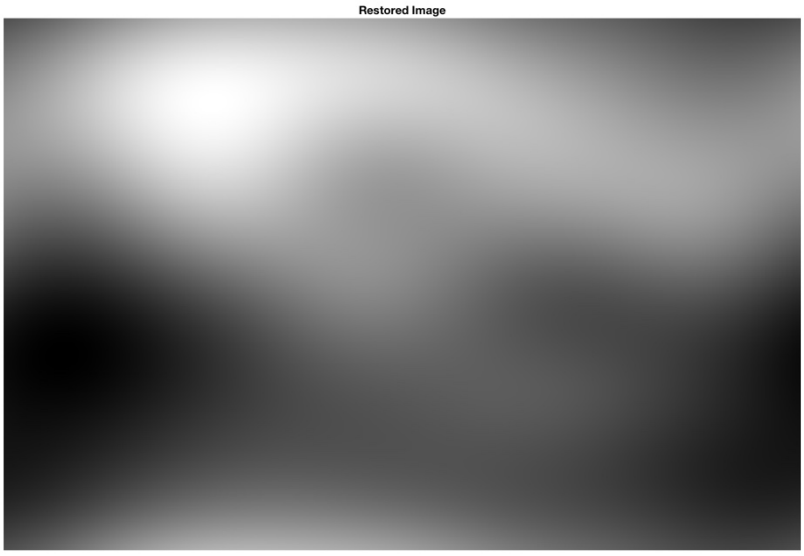


Figure 48) restored image from low cut-off frequency and high-level noise

Figure 49) restored image from high cut-off frequency and low-level noise in reverse filter

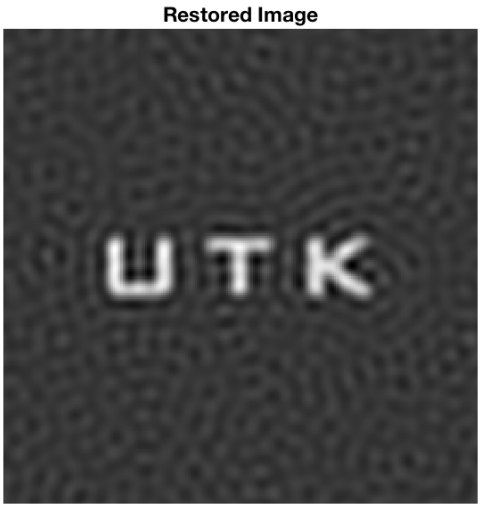


Figure 50) restored image from high cut-off frequency and medium level noise in reverse filter

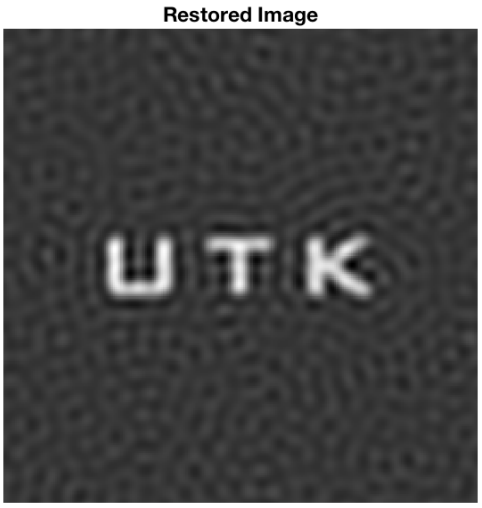


Figure 51) restored image from high cut-off frequency and high-level noise in reverse filter

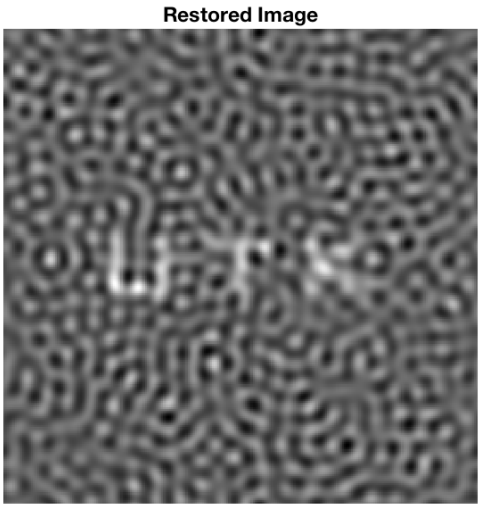


Figure 52) restored image from low cut-off frequency and low-level noise in reverse filter

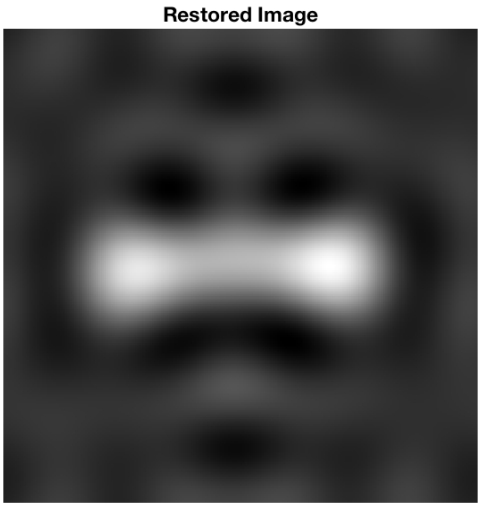


Figure 53) restored image from low cut-off frequency and medium level noise in reverse filter

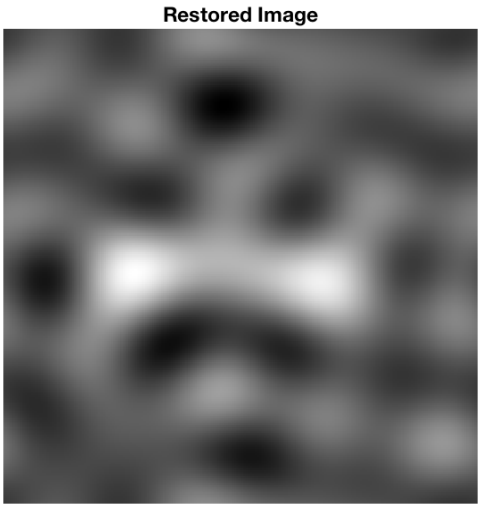


Figure 54) restored image from low cut-off frequency and high-level noise in reverse filter

Figure 55) restored image from high cut-off frequency and low-level noise in the Wiener filter

Figure 56) restored image from high cut-off frequency and medium level noise the in Wiener filter

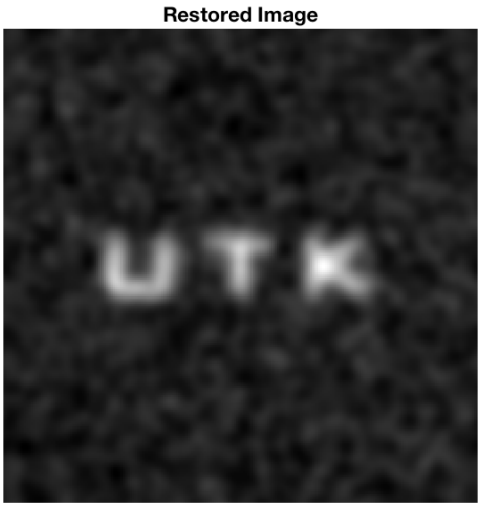


Figure 57) restored image from high cut-off frequency high-level level noise in Wiener filter

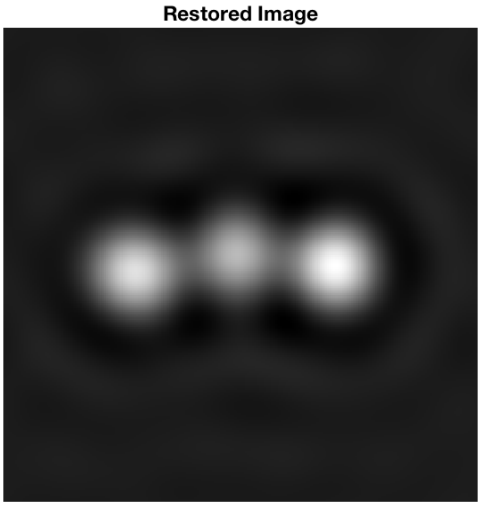
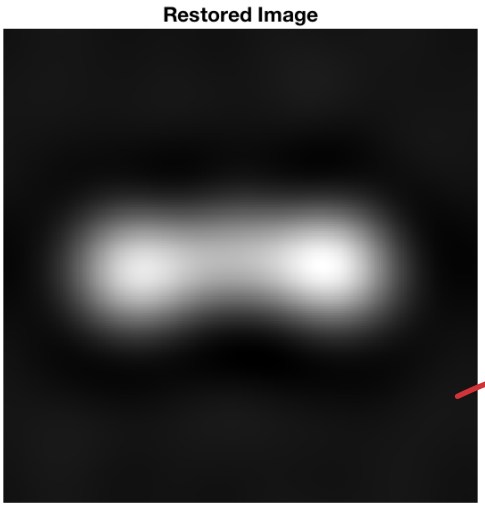


Figure 58) restored image from low cut-off frequency low-level level noise the in Wiener filter



Why do think this did not work?

Figure 59) restored image from low cut-off frequency high-level level noise in Wiener filter

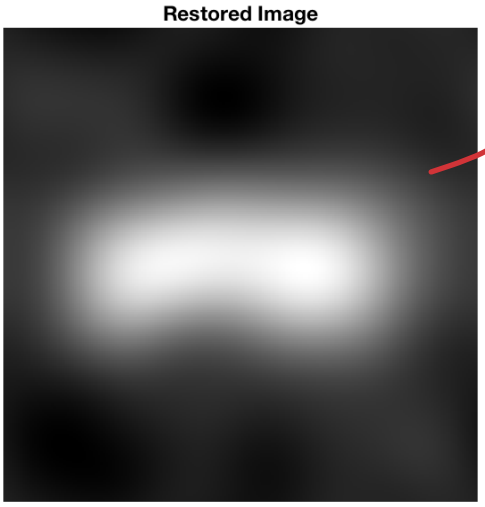


Figure 60) restored image from low cut-off frequency high-level level noise in Wiener filter

E. Appendix (codes)

I. M-file

```
clearvars;
clc
close all

%% 1 a
Variance = 10;
Mean = 0;

img = rgb2gray(imread('Pony.jpg'));
img = double((img(:,1:end-1)));

[imnoise_M, noise_M] = agn_noise(img,Mean,Variance);
[imnoise_L, noise_L] = agn_noise(img,Mean,0.1  * Variance);
[imnoise_H, noise_H] = agn_noise(img,Mean,10   * Variance);

figure;imshow(img, [])
title('Orginal Image')
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-a1','png')
figure;imshow(imnoise_M, [])
title(['Noisy image with Mean and Variance = ' num2str(Mean) ', ' num2str(Variance) ' )' ])
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-a2','png')
figure;imshow(imnoise_L, [])
title(['Noisy image with Mean and Variance = ' num2str(Mean) ', ' num2str(0.1  * Variance) ' )' ])
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-a3','png')
figure;imshow(imnoise_H, [])
title(['Noisy image with Mean and Variance = ' num2str(Mean) ', ' num2str(10 * Variance) ' )' ])
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-a4','png')
figure;imshow(noise_M, [])
title(['Additive Gaussian noise with Mean and Variance = (' num2str(Mean) ', ' num2str(Variance) ' )'
])
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-a5','png')

close all
%% 1 b
%img = imread('image.jpg');
cutoff_H = 11;
figure;
[OutputImage_H, ~, ~, filter_H] = G_lowpass(img,cutoff_H);
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-b1','png')


filter_S = real(ifft2(ifftshift(filter_H)));

figure;imshow(filter_S,[])
title(['Gaussian filter in spatial domain with cut-off = ' num2str(cutoff_H)])
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-b2','png')


cutoff_L = 2;
figure;
[OutputImage_L, ~, ~, filter_L] = G_lowpass(img,cutoff_L);
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-b3','png')


filter_S = real(ifft2(ifftshift(filter_L)));

figure;imshow(filter_S,[])
title(['Gaussian filter in spatial domain with cut-off = ' num2str(cutoff_L)])
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-b4','png')
%% 1 c
OutputImage_H = double(OutputImage_H);
OutputImage_L = double(OutputImage_L);

HH = OutputImage_H + noise_H;
HM = OutputImage_H + noise_M;
HL = OutputImage_H + noise_L;

LH = OutputImage_L + noise_H;
LM = OutputImage_L + noise_M;
LL = OutputImage_L + noise_L;

figure;imshow(HH,[])
title('adding a high level Gaussian noise to lowpass-filtered image with high cut-off frequency')
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-c1','png')
figure;imshow(HM,[])
title('adding a medium level Gaussian noise to lowpass-filtered image with high cut-off frequency')
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-c2','png')
figure;imshow(HL,[])
title('adding a low level Gaussian noise to lowpass-filtered image with high cut-off frequency')
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-c3','png')
```

```
figure;imshow(LH,[])
title('adding a high level Gaussian noise to lowpass-filtered image with low cut-off frequency')
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-c4','png')
figure;imshow(LM,[])
title('adding a medium level Gaussian noise to lowpass-filtered image with low cut-off frequency')
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-c5','png')
figure;imshow(LL,[])
title('adding a low level Gaussian noise to lowpass-filtered image with low cut-off frequency')
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/1-c6','png')

%% 1 d
ssimval_HH = ssim(HH,img);
fprintf('\n The SSIM index error for high level noise and cut of frequency = %0.2d  is %0.4f\n',
cutoff_H, ssimval_HH);
ssimval_HM = ssim(HM,img);
fprintf('\n The SSIM index error for medium level noise and cut of frequency = %0.2d  is %0.4f\n',
cutoff_H, ssimval_HM);
ssimval_HL = ssim(HL,img);
fprintf('\n The SSIM index error for low level noise and cut of frequency = %0.2d  is %0.4f\n',
cutoff_H, ssimval_HL);

ssimval_LH = ssim(LH,img);
fprintf('\n The SSIM index error for high level noise and cut of frequency = %0.2d  is %0.4f\n',
cutoff_L, ssimval_LH);
ssimval_LM = ssim(LM,img);
fprintf('\n The SSIM index error for medium level noise and cut of frequency = %0.2d  is %0.4f\n',
cutoff_L, ssimval_LM);
ssimval_LL = ssim(LL,img);
fprintf('\n The SSIM index error for low level noise and cut of frequency = %0.2d  is %0.4f\n',
cutoff_L, ssimval_LL);



err_HH = immse(HH,img);
fprintf('\n The mean-squared error for high level noise and cut of frequency = %0.2d  is %0.4f\n',
cutoff_H, err_HH);
err_HM = immse(HM,img);
fprintf('\n The mean-squared error for medium level noise and cut of frequency = %0.2d  is %0.4f\n',
cutoff_H, err_HM);
err_HL = immse(HL,img);
fprintf('\n The mean-squared error for low level noise and cut of frequency = %0.2d  is %0.4f\n',
cutoff_H, err_HL);

err_LH = immse(LH,img);
fprintf('\n The mean-squared error for high level noise and cut of frequency = %0.2d  is %0.4f\n',
cutoff_L, err_LH);
err_LM = immse(LM,img);
fprintf('\n The mean-squared error for medium level noise and cut of frequency = %0.2d  is %0.4f\n',
cutoff_L, err_LM);
err_LL = immse(LL,img);
fprintf('\n The mean-squared error for low level noise and cut of frequency = %0.2d  is %0.4f\n',
cutoff_L, err_LL);



%% 2 a
close all
test = {HL,HM,HH,LL,LM,LH};
for i = 1:6
    ss = test{1,i};
    [x, y] = size(ss);
    G{1,i} = fftshift(fft2(ss));
    figure;surf(log(abs(G{1,i})))
    title('the 2-D estimate of the power spectrum of the output image')
    saveas(gcf, ['/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/2-a2-' num2str(i)],'png')
    figure;plot(1:y, log(abs(G{1,i}(round(x/2),:))))
    title('a 1-D slice of the 2-D estimate of the power spectrum of the output image')
    saveas(gcf, ['/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/2-a2a-' num2str(i)],'png')
end

%% 2 b

iH = 1./filter_H;
figure;surf(log(abs(filter_H)))
shading interp;
colormap(cool(10));
title('inverse filter (H)')
saveas(gcf, '/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/2-b3','png')

close all
filter = {filter_H, filter_H, filter_H, filter_L, filter_L, filter_L};
for i = 1:6

    %F_hat = (abs(filter{1,i}) > 0.1) .* G{1,i} ./ filter{1,i};
    F_hat = G{1,i} .* iH;

    F_hat(isnan(F_hat)) = 0;

    f_hat = real(ifft2(ifftshift(F_hat)));
    f_hat(isnan(f_hat)) = 255;
```

```
    figure;imshow(f_hat,[])
    title('Restored Image')
    saveas(gcf, ['/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/2-b2b-' num2str(i)],'png')

    ssimval = ssim(f_hat,img);
    fprintf('\n The SSIM index error for recovered image is %0.4f\n', ssimval);

    err  = immse(f_hat,img);
    fprintf('\n The mean-squared error for recovered image is %0.4f\n', err);
end

%% 2 c


 iH = 1./filter_H;
% figure;imshow(log(abs(1./filter_H)),[])
% title('inverse filter (H)')

k = 1e-4;
W = (iH .* ((abs(filter_H).^2)./((abs(filter_H).^2)+k)));

F_hat = G{1,1} .* W;

f_hat = ifft2(ifftshift(F_hat));
f_hat = real(f_hat);

figure;imshow(f_hat,[])
title('Restored Image')

ssimval = ssim(f_hat,img);
fprintf('\n The SSIM index error for recovered image is %0.4f\n', ssimval);

err  = immse(f_hat,img);
fprintf('\n The mean-squared error for recovered image is %0.4f\n', err);

%%
close all
filter = {filter_H, filter_H, filter_H, filter_L, filter_L, filter_L};
for i = 4

    k = 0.5;

    NF = fft2(noise_L);
    N = sum(sum((abs(NF).^2)));
    SF = fft2(img);
    S = sum(sum((abs(SF).^2)));
    k = N/S

    W = ( (1 ./ filter{1,i}) .*((abs(filter{1,i}).^2)./((abs(filter{1,i}).^2)+k)) );

    %F_hat = (abs(filter{1,i}) > 0.1) .* G{1,i} ./ filter{1,i};
    %F_hat = G{1,i} .* iH;

    F_hat = G{1,i} .* W;
    F_hat(isnan(F_hat)) = 0;

    f_hat = real(ifft2(ifftshift(F_hat)));

    figure;imshow(f_hat,[])
    title('Restored Image')
    saveas(gcf, ['/Users/saeedkazemi/Documents/MATLAB/Lab3 EE/saves/2-c2c-' num2str(i)],'png')

    ssimval = ssim(f_hat,img);
    fprintf('\n The SSIM index error for recovered image is %0.4f\n', ssimval);

    err  = immse(f_hat,img);
    fprintf('\n The mean-squared error for recovered image is %0.4f\n', err);
end
```

II. Gaussian low pass filter

```
function [OutputImage, filtered_image_f, fftInput, filter] = G_lowpass(InputImage,CutFrequency)

    I = (InputImage);

    [x, y] = size(I);
    if rem( x , 2) == 0 && rem( y , 2) == 0
        I = I(1:x-1,1:y-1);
    elseif rem( x , 2) == 0 && rem( y , 2) == 1
        I = I(1:x-1,1:y);
    elseif rem( x , 2) == 1 && rem( y , 2) == 0
        I = I(1:x,1:y-1);
    end

    [x, y] = size(I);
```

```
    fftInput = fft2(I);
    fftInput = fftshift(fftInput);
    subplot(2,2,1)
    imshow(I, []);
    title('Original Image')
    subplot(2,2,2)
    imshow(log(abs(fftInput)),[])
    title('Image in Fourier Domain')

    filter = zeros(x,y);

    for i = 1:x
        for j =1:y
            D = ((i-(x+1)/2)^2 + (j-(y+1)/2)^2)^.5;

            filter(i,j)= exp((-(D^2))/(2*CutFrequency^2));% 1/(1 + (D/CutFrequency)^(2*FilterOrder));

        end
    end

    subplot(2,2,3)
    imshow(log(abs(filter)),[])
    title('Filter Image in Freq Domain')

    filtered_image_f = filter.*fftInput;
    filtered_image = ifft2(ifftshift(filtered_image_f));
    filtered_image = real(filtered_image(1:x,1:y));
    OutputImage = uint8(filtered_image);
    subplot(2,2,4)
    imshow(OutputImage,[])
    title('Filtered Image')

end
```

III. Additive White Gaussian Noise

```
function [imnoise,noise] = agn_noise(img,mean,Variance)

    %computed noise
    noise = Variance .* randn(size(img)) + mean;
    %received signal
    imnoise = img + noise;

end
```