SYDE 575
Lab 4 - Report

Prepared by:
Timothy Tsang 20556306
Jonathan Wen 20515413

Professor David A. Clausi

Wednesday November 7th, 2018

1. Introduction

This lab aims to introduce techniques related to image restoration and removing noise within images using various filters, all within the frequency domain. First, image restoration techniques related to inverse filtering and Wiener filtering were investigated and compared upon various types of noise. Lastly, the structure of the image was used to determine what values the filter would have through the use of the Lee filter.

2. Image Restoration in the Frequency Domain

First, a disk blur function was created and used to create a blurring function. The blurring function was multiplied in frequency domain (convolved in time) to the Cameraman image used to create a blurred, smoothed image. Calculating it's PSNR relative to the original Cameraman image, the value was 16.8964. Below displays the blurred Cameraman image.

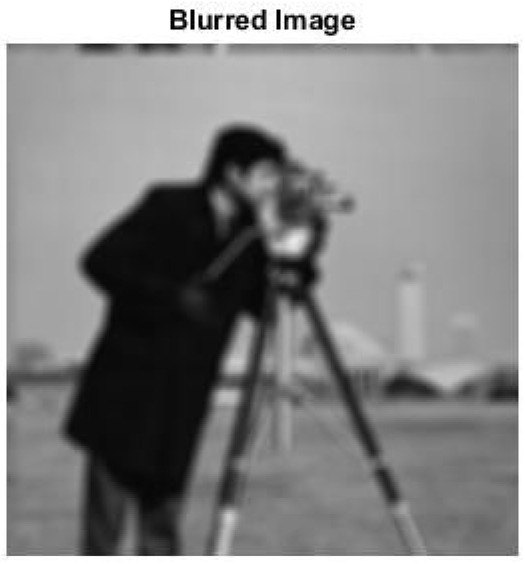

Figure #1 - Blurred Cameraman image

Next, the use of inverse filtering was done to attempt to restore the image to its original form. The blurred image was divided by the fourier transformed blurred image with the blurring filter function. Converting back into the time domain through inverse fourier transform, the restored image was plotted and it's PSNR relative to the original Cameraman image was 255.1425. Clearly this restoration technique produced a result that was very similar to the original image. Below displays the restored image via inverse filtering.

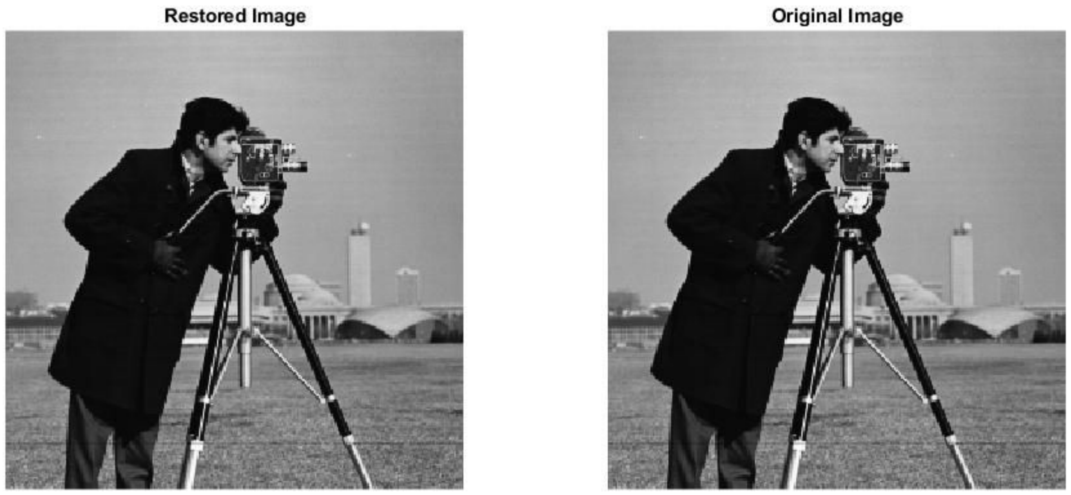

Figure #2 - Restored image via Inverse Filtering

1. The restored image is very close to the original image, with inverse filtering effectively removing the smoothing and blurring. In fact, the restored image has it's edges enhanced; the edges of the image has even more clearer edges than the original image! Quantitatively, the PSNR of the restored image was 255.1425, which is significantly higher than the blurred images PSNR, which was 16.8964. In every way, the restored image is much more closer to the original image. We know that the inverse filter is a high pass filter, and with the blurred image having a lot more high intensity pixels, with the use of the filter, the restored image will retain most of the pixels and display similar to the original image. We also predict that if this inverse filter is used for images with noise, it would not produce a good result since HPF would perform poorly with high intensity noise.

A Gaussian noise was added into the blurred Cameraman image, with a mean of zero and a variance of 0.002. Calculating it's PSNR relative to the original Cameraman image, the value was found to be 16.5100. This is lower than the PSNR of the blurred image, since more noise was added onto the image, making it further away from the original image.

In order to restore the image to its original form, the use of inverse filtering was done on the blurred noise-induced image. Calculating the PSNR, the value was -40.9833, which was due to the result looking like static noise, being very far away from the original image. Below displays the "restored" image done via inverse filtering on the noisy blurred image.

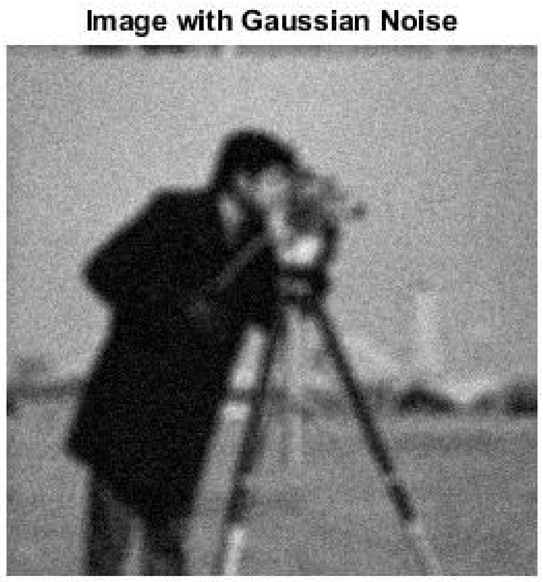

Figure #3 - Image with Gaussian noise

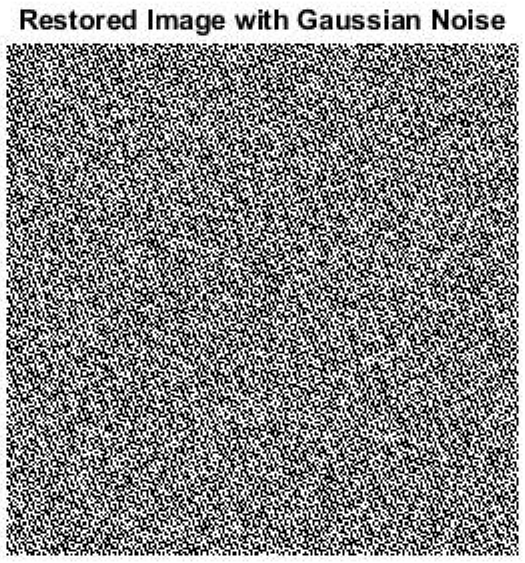

Figure #4 - Restored image with Gaussian noise

1. The restored image with the Gaussian-noisy blurred image was very poor, since the noise that was introduced in the image was high intensities. Since the inverse filter is a high pass filter, it will amplify high frequency noise, which includes the noise. In doing so, the restored image using the inverse filter is very poor - in fact, the noise is enhanced the most while the original image is retained less. The PSNR of the gaussian-noisy blurred image was -40.9833, which means that there are a lot of errors and noise within the restored image.

2. Noise is typically high frequency and thus for noise degraded images, inverse filtering would perform very poorly, since it is a high pass filter, which would amplify the noise rather than removing the noise.

Next, the Wiener filter was used to restore the Gaussian-noisy blurred Cameraman image. In order to use the Wiener filter, the built in deconvwnr function was used. The point spread function used the original disk blur function (h). Lastly, the noise-to-signal power ratio of the additive noise was calculated by dividing the variance of the noise with the variance of the noisy image. This was done because as shown in class, the NSR can be found as $NSR = \frac{P_{noise}}{P_{signal}}$. Since we can calculate the variance of both the signal (noisy image) and the noise (which was given in the lab manual), and since the signal and noise are both zero-mean, we are able to calculate the NSR, which is simply the quotient of $NSR = \frac{\sigma^2_{noise}}{\sigma^2_{signal}}$. With the variance of the noise as 0.002 and the variance of the noisy image as 0.0508, the NSR was calculated as 0.0394.

The Wiener filter was used upon the Gaussian-noisy blurred image, which resulted in a PSNR of 15.8790, relative to the original Cameraman image. Below illustrates the result:

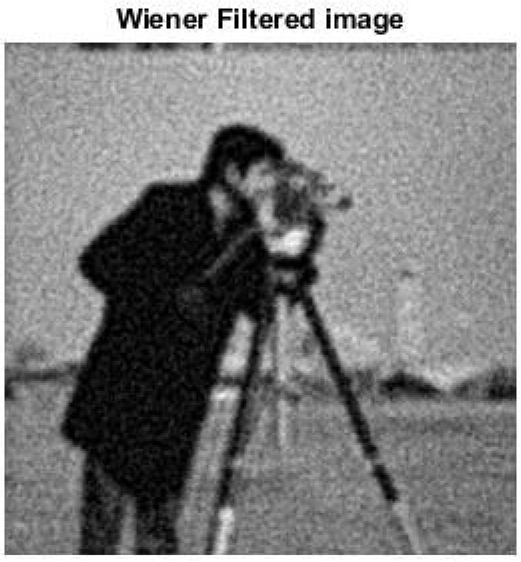

Figure #5 - Restored Image using the Wiener Filter

1. Strictly looking qualitatively at the two photos (Figure 3 and Figure 5), the restored image clearly does remove some noise and is more closer to the original Cameraman image compared to the one just with Gaussian noise.
   Quantitatively, when comparing the PSNR, the PSNR of the wiener filtered image relative to the original Cameraman image was 15.8790, while the PSNR of the

gaussian-noisy image was 16.5100 relative to the original Cameraman image. This means that the filtering did not restore the image closer to the actual original image - rather, it made it worse! Here, the Noise-to-Signal ratio used was 0.0394, as stated above. Through trial and error, increasing the NSR to around 0.09 gave us the best result and PSNR, resulting in a PSNR of 16.1855. However, that is still worse than the gaussian-noisy image!
By definition, the Wiener filter attempts to minimize the MSE (mean squared error) for an image with additive noise.

2. When Wiener filtering is applied to a noise degraded image, it effectively removes the unwanted additive noise, as well as keeps the overall integrity of the image.

3. Adaptive Filtering

In this section, the structure of the image will be taken into account - specifically, the degree of smoothing of different sections of the image. Here, the degraded image was used as the test image, which is shown below:

Figure #6 - Degraded Image

Here, the use of a "flat" region of the image was used to calculate the variance of the intensity levels. In doing so, the MATLAB built-in function getrect was used to determine the range of the a rectangle subsection on the image. The use of the background, specifically the top right background was used. In order to calculate the variance of the noise, the variance of the rectangle subsection was calculated. Specifically, in the lab, the rectangle results were:

- Xmin = 204
- Ymin = 47
- Width = 19
- Height = 22

In doing so, the variance of the noise was calculated to be 0.0071.

Next, the local mean and variance of the image was calculated using the colfilt function. Those two arrays will be used to calculate the K matrix and the Lee formation filter respectively. Iterating through the size of the image (256x256), the K and Lee values were calculated. The PSNR of the Lee filtered image was found to be 22.7112 relative to the original image. Below displays the original and denoised image using the Lee filter:

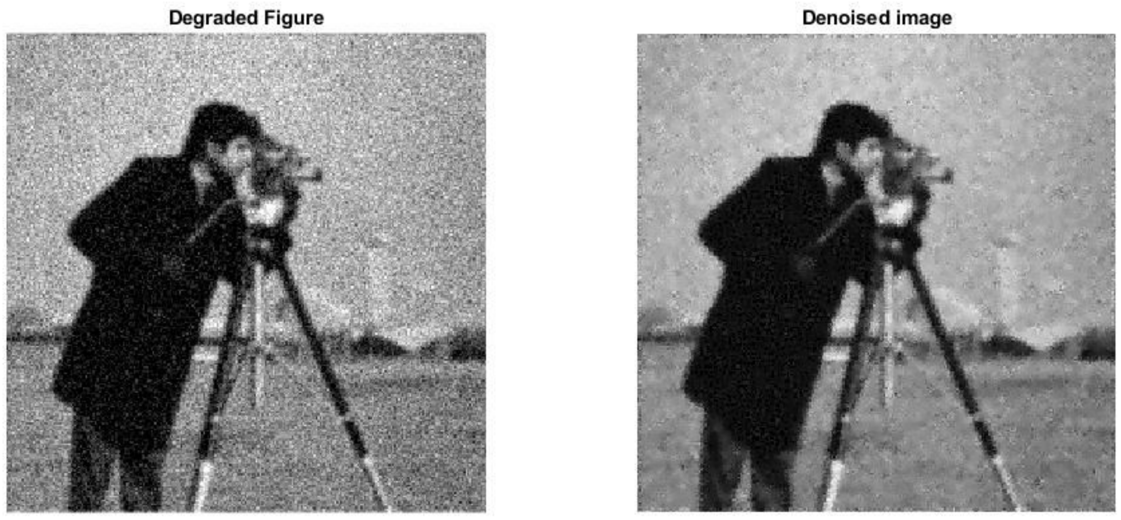

Figure #7 - Denoised Image via the Lee filter

A Gaussian low pass filter with a standard deviation of 30 was also used to filter the degraded image. The result was compared with the denoised image using the Lee filter, shown below:

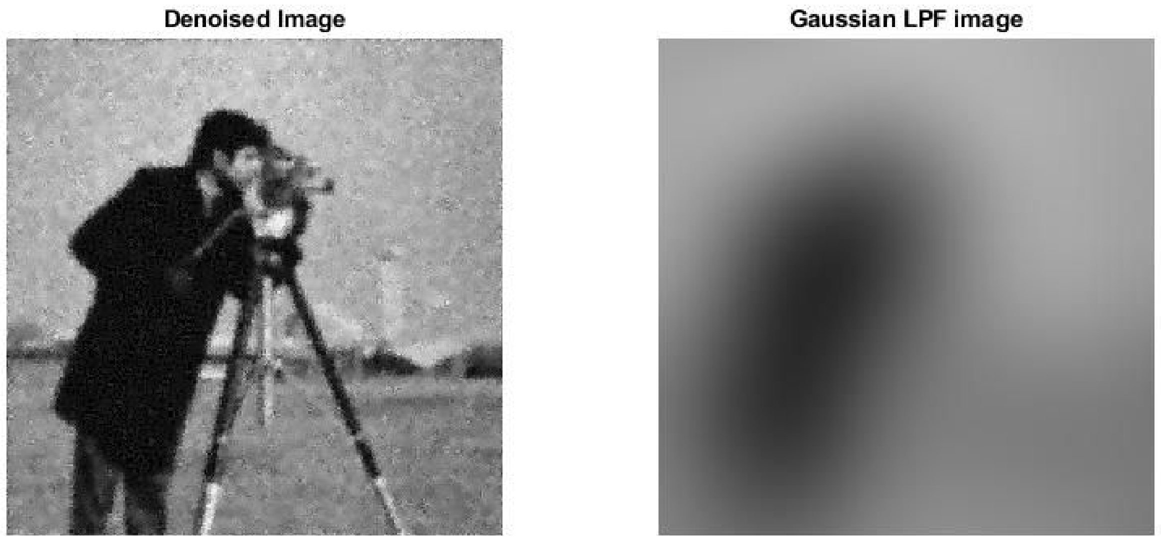

Figure #8 - Lee Filter result vs Gaussian LPF result

1. Using a Gaussian LPF, the result image is an extremely blurred image of the original. This is because the standard deviation (sigma=30) is extremely high, which will maximally smooth the image. When a sigma value of 3 is used instead, the result looks like this:

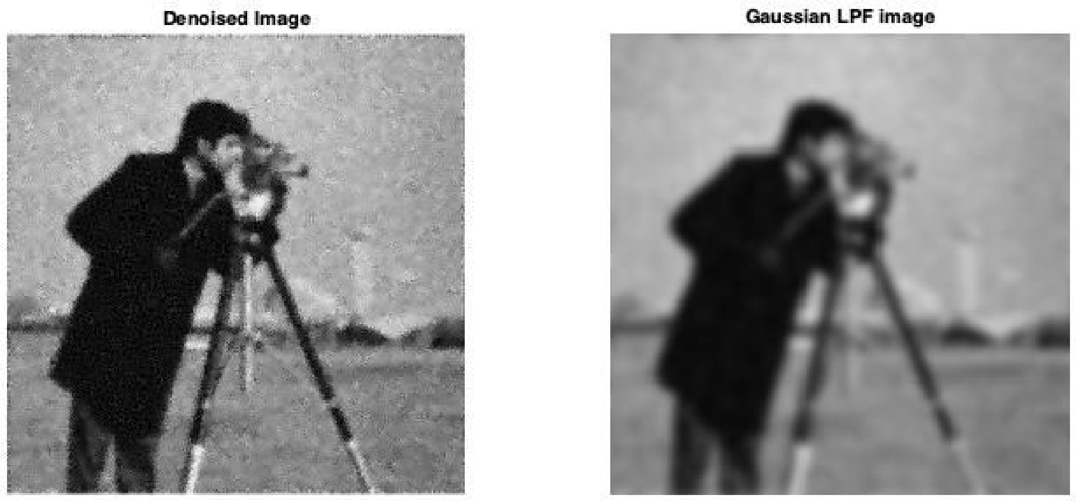

Figure #9 - Lee Filter result vs Gaussian LPF with sigma=3

Evidently, the image is still blurred. The regions of the image where there is low detail / contrast are preserved the same), whereas the edges with high contrast are smoothed.

2. By changing the estimate of the noise variance, the resulting image is also greatly affected. Currently, the variance is set to 0.007. If the variance of the noise is set to 0, this assumes that there will be no noise. Therefore, the resulting image will look identical to the original. If the variance is set to 0.02, the result is flatter, and the noise is not eliminated. This is because as the variance of the noise increases, the SNR decreases and approaches to 0, and the Lee filter approaches to the mean of g(i,j).

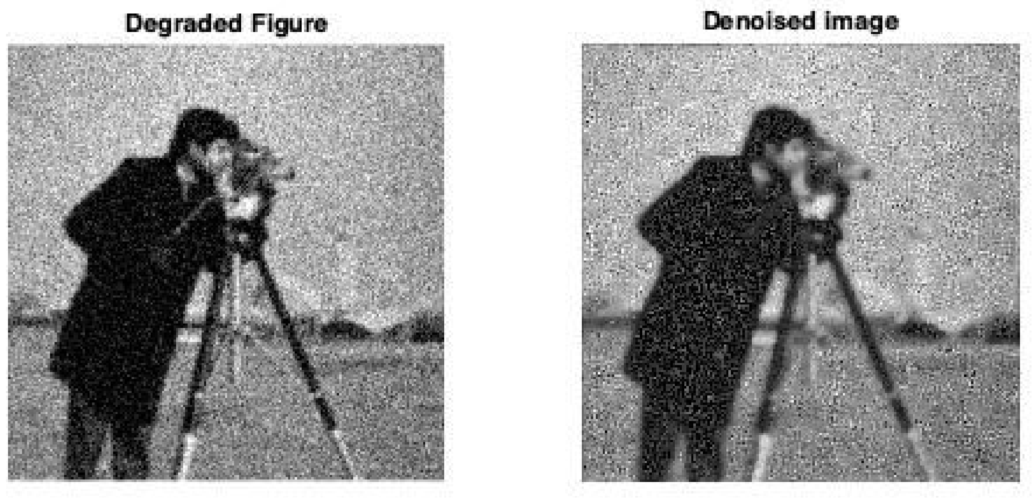

Figure #10 - Lee Filter result when noise variance = 0.02

3. The size of the filter neighbourhood that was originally taken was 19 pixels by 22 pixels - it was a pretty small window. Increasing the size of the filter from the same region (top right area), say 60 pixels by 60 pixels, we get a denoised image shown below, with a PSNR of 5.8347.

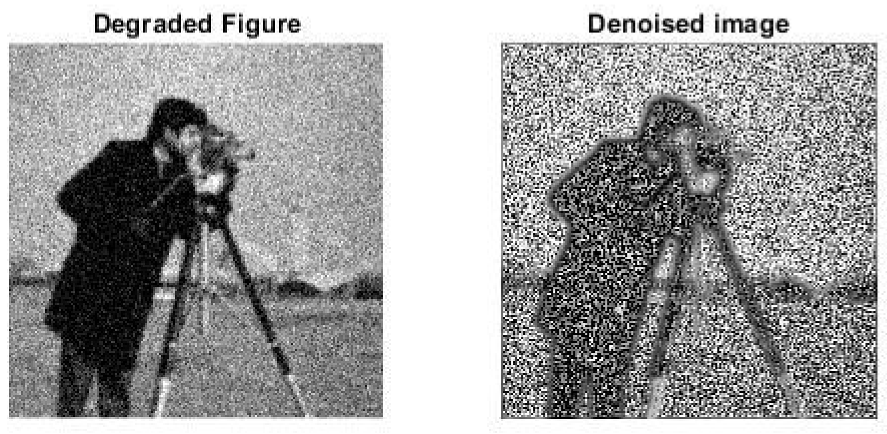

Figure #11: Lee Filtered Image with a 60x60 window

Decreasing the size of the filter, say 10 pixels by 10 pixels within the same background region, we get a denoised image shown below, with a PSNR of 18.5536.

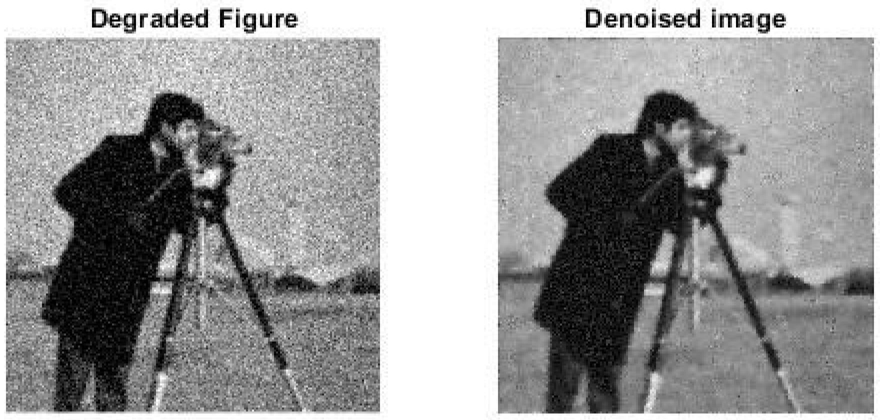

Figure #12: Lee Filtered Image with a 10x10 window

The size of the window will affect the estimate of the variance of the noise. If the window is too small, there will not be enough pixels to estimate accurately the variance. The value can change dramatically depending on where the window is placed, even though the region may be considered 'flat'. However, on the other hand, if the window size if too big, the region may overlap with features, which will also affect the estimate of the variance. The window area will likely not be 'flat'. Therefore, there needs to be a balance of the window size.

4. Conclusion

In conclusion, this lab was able to introduce image restoration and restoration techniques that can be used on images with a variety of noise. While the technique of inverse filtering is useful, it is not great for noisy images, especially high frequency noise. The Wiener filter is also a good technique for images with additive noise, as the technique attempts to optimize the mean squared error. Additionally, adaptive filtering was used to remove Gaussian noise from an image; depending on the window size and variance chosen, the image was accurately be restored and effectively remove noise.

5. Appendix

```
%% Section 1
degraded = imread('degraded.tif');
cameraman = imread('cameraman.tif');

%% Section 2 - Image Restoration in the Frequency Domain

h = fspecial('disk',4);
f = im2double(imread('cameraman.tif'));
h_freq = fft2(h, size(f,1), size(f,2)); % pad h
f_blur = real(ifft2(h_freq.*fft2(f)));

figure;
imshow(f_blur);
title('Blurred Image');

% PSNR of blurred image
f_blur_psnr = 10*log10(1/mean2((f-f_blur).^2));

% Inverse Filter
inverseFilterImage = real(ifft2((fft2(f_blur)./h_freq)));
figure;
subplot(1,2,1);
imshow(inverseFilterImage);
title('Restored Image');

subplot(1,2,2);
imshow(cameraman);
title('Original Image');

% PSNR of Inverse Filtered image
f_inverse_psnr = 10*log10(1/mean2((f-inverseFilterImage).^2));


% Image with Gaussian Noise
% Add Gaussian Noise
noiseyImage = imnoise(f_blur,'gaussian',0,0.002);
inverseFilterNoisyImage = real(ifft2((fft2(noiseyImage)./h_freq)));

figure;
subplot(1,2,1);
imshow(inverseFilterNoisyImage);
title('Restored Image with Gaussian Noise');

subplot(1,2,2);
imshow(noiseyImage);
title('Image with Gaussian Noise');
```

```
% PSNR of Noisy image and Restored image
f_noisy_image_psnr = 10*log10(1/mean2((f-noiseyImage).^2));
f_noisy_image_restored_psnr =
10*log10(1/mean2((f-inverseFilterNoisyImage).^2));


% Apply Wiener Function
var_noise = 0.002;
var_f = var(noiseyImage(:));
nsr = var_noise/var_f;
wienerFilter = deconvwnr(noiseyImage, h, nsr);

figure;
imshow(wienerFilter);
title('Wiener Filtered image');

% PSNR of Wiener Filtered Image
wiener_psnr = 10*log10(1/mean2((f-wienerFilter).^2));


%% Section 3 - Adaptive Filtering

d = im2double(imread('degraded.tif'));
figure;
imshow(d);
title('Degraded Figure');
% rect = getrect; % [204,47,19,22]
rect = [204,47,19,22];
d_rect = d(rect(1):rect(1)+rect(3), rect(2):rect(2)+rect(4));
var_rect = var(d_rect(:));
var_rect = 0.02;

local_mean = colfilt(d, [5,5], 'sliding', @mean);
local_var = colfilt(d, [5,5], 'sliding', @var);

K = zeros(256,256);
Lee = zeros(256,256);

for i = 1:256
    for j = 1:256
        K(i,j) = (local_var(i,j)-var_rect)/local_var(i,j);
        Lee(i,j) = K(i,j)*d(i,j) + (1-K(i,j))*local_mean(i,j);
    end
end

% Plot
figure;
```

```
subplot(1,2,1);
imshow(d);
title('Degraded Figure');

subplot(1,2,2);
imshow(Lee);
title('Denoised image');

% PSNR
lee_psnr = 10*log10(1/mean2((d-Lee).^2));

% Low pass filter
gaussianFilterImage = imgaussfilt(d, 3);

% Plot
figure;
subplot(1,2,1);
imshow(Lee);
title('Denoised Image');

subplot(1,2,2);
imshow(gaussianFilterImage);
title('Gaussian LPF image');
```