University of Waterloo

Faculty of Engineering

SYDE575 Lab 4: Restoration

Prepared By
Sarthak Tamboli | 20665466
Sheen Thusoo | 20728766

Date: 12 November, 2021
Prepared for: Professor David A. Clausi

1. Introduction

The purpose of this lab is to study image restoration concepts in the frequency domain. In Section 2, we study two frequency domain restoration methods: inverse filtering and Wiener filtering. We perform this on the cameraman image. In Section 3, we study noise reduction using the Lee filter. Here, we use the degraded mage. We estimate the noise variance using an empirical method - capture a "flat region of the image and calculate the variance of the intensities. The code for this lab was written in MATLAB and is shown in Appendix A.

2. Image Restoration in the Frequency Domain

In Section 2, we investigated inverse filtering and Wiener filtering. First, the cameraman image was loaded and adjusted to intensities from 0 to 1. Then, a disk blur function with a radius of 4 was created and applied to the image in the frequency domain. The plot of the original image can be seen in Figure 1 and the blurred image can be seen in Figure 2 and its PSNR value (around 21.096) can be seen in Table 1.

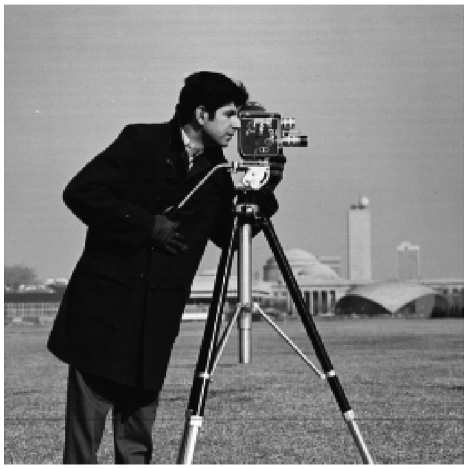


Figure 1: Original Image

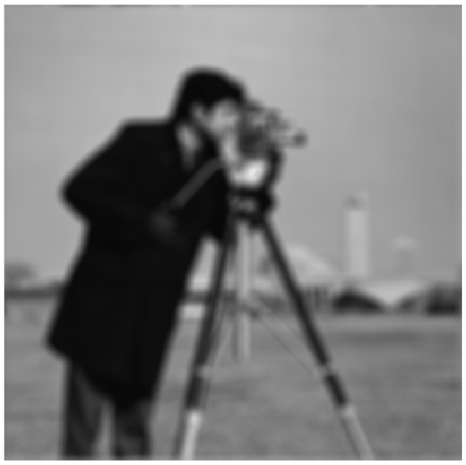


Figure 2: Blurred Image

Table 1: PSNR Values for Section 2

Image	PSNR Value
Blurred Cameraman Image	21.095667966759297
Inverse Filtered Blurred Cameraman Image	253.1469753975226
Inverse Filtered Noisy Cameraman Image	-40.944339389744070
Wiener Filtered Noisy Cameraman Image	20.396116511383504

Next, the inverse filter was applied by dividing the image by the blurring function. Its result can be seen in Figure 3.

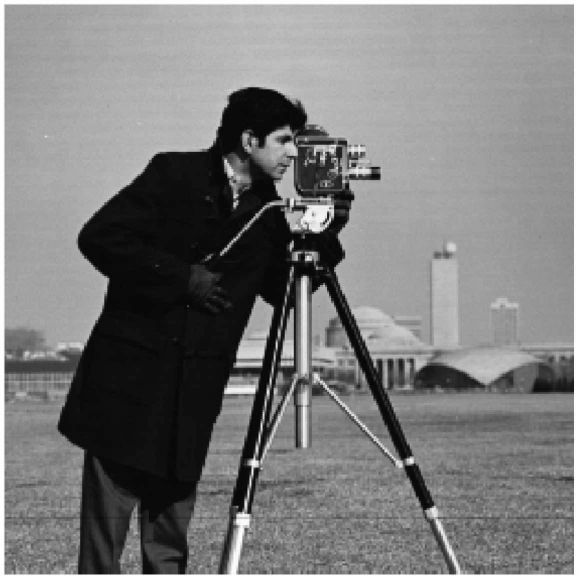


Figure 3: Restored Image - Inverse Filter

1. The restored image is much sharper than the blurred image so it has a higher image quality. The blurred image is completely smoothed so it is difficult to see the fine details in the image; blurring acts like a low-pass filter and attenuates higher spatial frequencies (fine details). The restored image is much sharper so it is easier to discern the details (i.e. edges) of the image. Compared to the original image, the restored image looks very similar but has further enhanced the edges in the image. It has successfully eliminated the smoothening in the blurred image. The PSNR value of the blurred image was around 21.096 whereas the PSNR value of the restored image was around 253.147. The restored image had a much greater PSNR value indicating a much higher image quality; this is reflected in what we see visually from the figures. In the spatial-frequency domain, when we divide the blurred image by the blurring function we get the inverse filter. It then makes sense that the inverse filter, which is high-pass, undos the smoothening done by the blurring function, which is low-pass. Hence, we get a similar image to our original with the blurring removed.

Then, zero-mean Gaussian noise with a variance of 0.002 was added to the blurred image and inverse filtering was applied to the noisy blurred image. The noisy cameraman image can be seen in Figure 4 and the result of the inverse filter can be seen in Figure 5. Its PSNR can be seen in Table 1 - that being around -40.944. The inverse filter is good at de-blurring images.

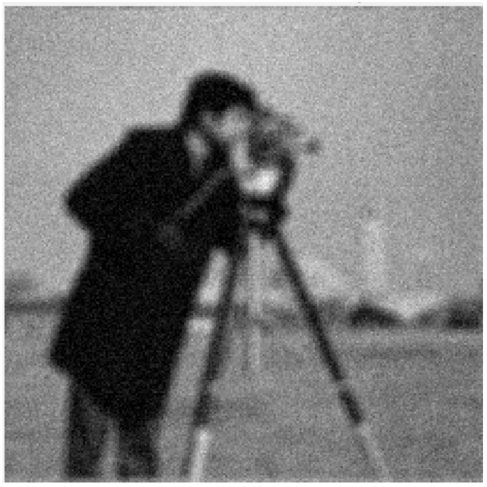


Figure 4: Blurred Cameraman Image with Gaussian Noise

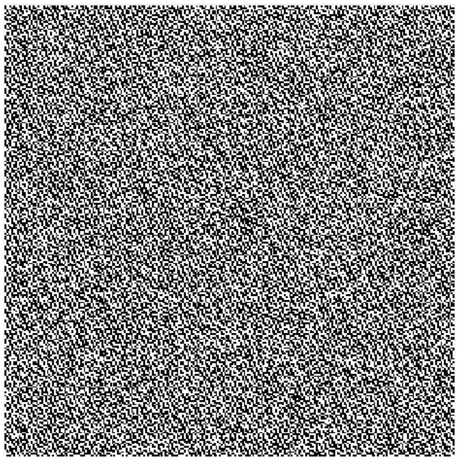


Figure 5: Gaussian Noise Image - Inverse Filter

2. Compared to the restored image in the previous step, this restored image does not resemble the original image at all. This restored image seems like it does not retain the structure of the original image, it only retains the noise added to the blurred image. The restored image in the previous step reduced the blurring and retained the details in the original image; it has much better image quality. The PSNR value of the restored image in the previous step was 253.147 which is much higher than the PSNR value of this restored image which is -40.944. This indicates that the image quality of the restored image in this step is much worse; this is reflected in what can be seen visually from the images. Since the PSNR value is negative, it reflects that the noise is greater than the actual signal in this image. This restored image is much worse because in this case, the image was blurred and a high-frequency Gaussian noise was added to it. Since the inverse filter is a high-pass filter, it will only attenuate low spatial frequencies, and retain high spatial frequencies. This causes the filter to amplify the high-frequency Gaussian noise. Therefore, since the distortion noise, H, is smaller, the noise N predominates in the restored image. In the spatial domain, there is noise at every spatial frequency.

3. It can be said that when inverse filtering is applied to a noise degraded image it does not perform well in reducing the noise. In fact, the inverse filter will amplify this noise in the image and the output will be poor. This is because noise is a high-frequency component and the inverse filter is a high pass filter, so it will not attenuate this noise rather it will amplify noise. This is also shown from the equations below where $F$is the original image, $\widehat{F}$is the restored image, $G$is the blurred image, $H$is the distortion and $N$is the noise. We can see in the final equation for the restored image that if $H$is small, Npredominates the output.

$$\widehat{F} = \frac{G}{H}$$

$$\text{But } G = FH + N$$

$$\text{Thus, } \widehat{F} = \frac{FH + N}{N} = F + \frac{N}{H}$$

Lastly, we applied the Wiener filter to the noisy blurred image using the `deconvwnr` function. The disk function was passed in as the PSF (point spread function). The noise-to-signal value was approximated using the power ratio which is defined as

$$NSR = \frac{S_N(u)}{S_f(u)}$$

Where $S_f(u)$is the power spectrum of the image and $S_N(u)$is the power spectrum of the noise. Since the Gaussian noise and the noisy image signal are both zero-mean, their power spectra are just their variances so

$$S_N(u) = \sigma_N^2 \qquad S_f(u) = \sigma_f^2$$

Thus, the NSR was found to be

$$NSR = \frac{0.002}{0.0508} = 0.0394$$

The restored image can be seen in Figure 6 and its PSNR value is in Table 1, 20.396.

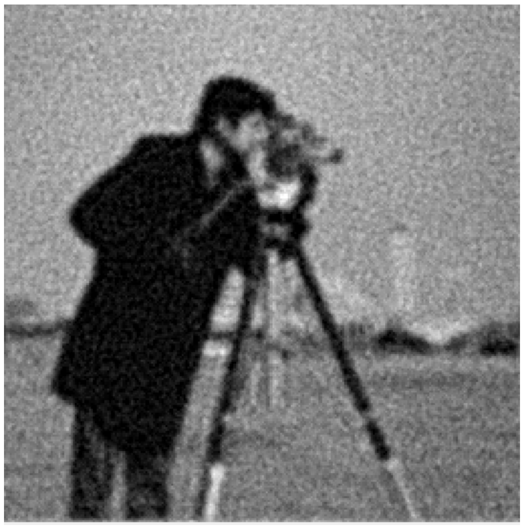


Figure 6: Gaussian Noise Image - Wiener Filter

4. Compared to the restored image from the previous step, this restored image looks more similar to the original cameraman image. The main details of the image, such as the camera and the man, can be distinguished in this restored image. However, it does still have noise present. The PSNR value of the previous restored image was -40.944 whereas the PSNR value of this restored image is 20.396. This higher PSNR value indicates that this restored image has a higher image quality than the previous; this is also reflected when visually inspecting the images. This restored image is better because the Wiener filter aims to minimise the MSE (mean squared image) of the restored image and the original image for an image with additive noise and blurring. Thus, it is able to reduce additive noise (low pass filter) while also unblurring the image (high pass filter). So, its restored image quality is better than inverse filtering for images affected by distortion and noise.

5. When a Wiener filter is applied to noise degraded images, it is able to reduce the additive noise effectively unlike the inverse filter. Because it minimizes the MSE, it optimizes the tradeoff between noise smoothing and inverse filtering. It is able to remove additive noise while unblurring the image as well. This filter performs deconvolution by inverse filtering (high pass filter) and removes additive noise with low pass filtering. Using a higher SNR value leads to the Wiener filter to act similar to an inverse filter whereas using a lower SNR value leads to the filter minimizing the noise in the image.

3. Adaptive Filtering

In Section 3, the Lee Filter, an adaptive filter, was applied to a degraded cameraman image with varying levels of noise variance and filter size. In addition, it was also compared against a Gaussian filter with a standard deviation of 30 in the frequency domain. The corresponding PSNR values are shown below in Table 2. The relevant code for this section can be found in Appendix A, Figure A.2.1 to Figure A.2.5, and Figure A.3.1.

Table 2: PSNR Values for Section 3

Filter Applied	PSNR Value
Lee Filter (5x5)	19.813391412696525
Gaussian Low Pass Filter w/ stddev 30 (in frequency domain)	20.833676470613334
Lee Filter (5x5), High (0.02) Noise Variance	13.706692356357223
Lee Filter (5x5), Low/No (0) Noise Variance	Inf
Lee Filter (4x4), Smaller Size Filter	19.376867193131304
Lee Filter (10x10), Larger Size Filter	20.707564143804788

6.

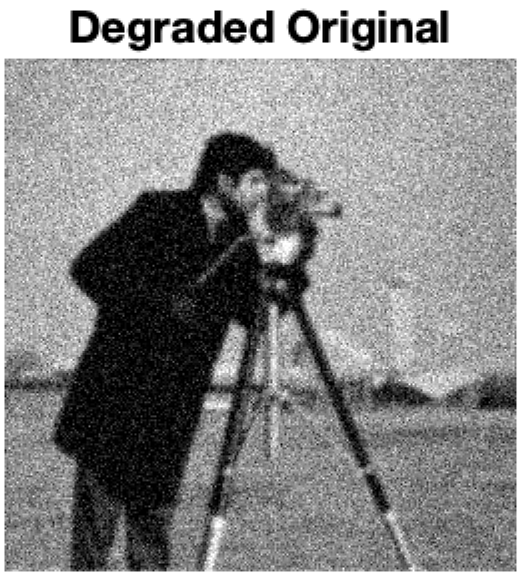


Figure 7: Original Degraded; used to select flat region from top left of image

A flat region in the form of a rectangle from the upper left corner of the original degraded image, Figure 7, was used. The features of the rectangle are shown below in Table 3.

Table 3: Flat Region Rectangle Features from getrect function in MATLAB

xmin	ymin	width	height
4	6	46	22

The noise variance was calculated to be 0.009901364370309.

The top left region was chosen in particular as it represents a region that is roughly uniform in grey level intensity while having little to no edges. Thus, the selection is valid for an adequate flat region due to it having minimal quality image data.

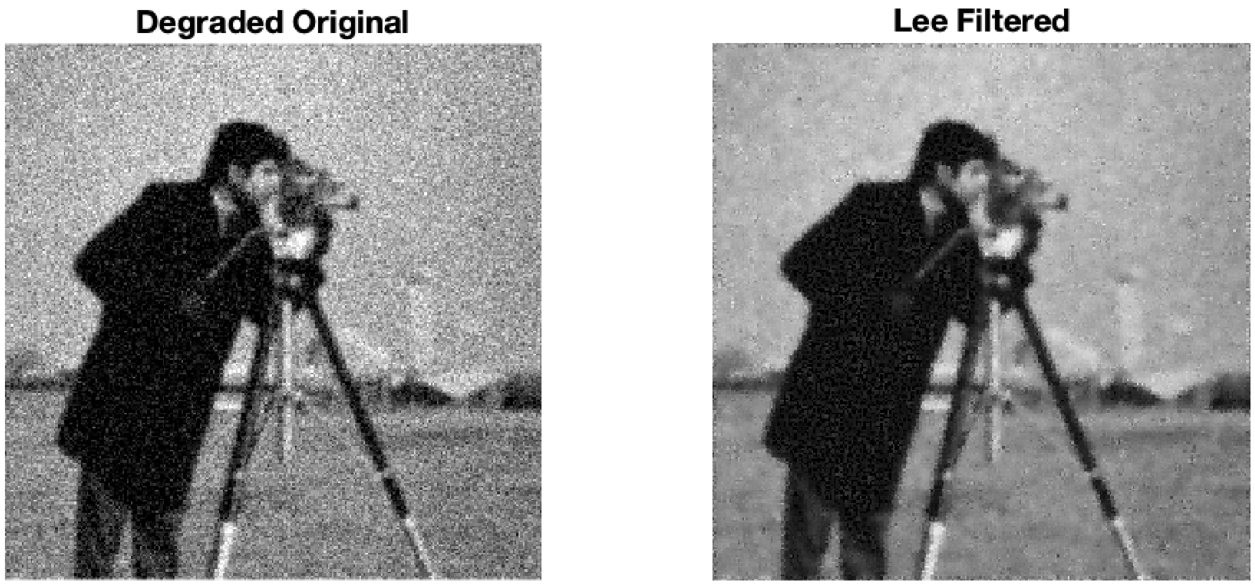


Figure 8: Original Degraded vs 5x5 Lee Filtered

The effect of applying a 5x5 neighborhood Lee Filter to the original degraded image is shown above in Figure 8.

It can be observed that a distinct blur exists overtop the filtered image with a reduction in the "shot noise" effect of the noise in the image. The image is definitely smoother and less granular with increased uniformity in the background and changing grey gradients.

The PSNR value of the Lee Filtered image was 19.813391412696525.

7.

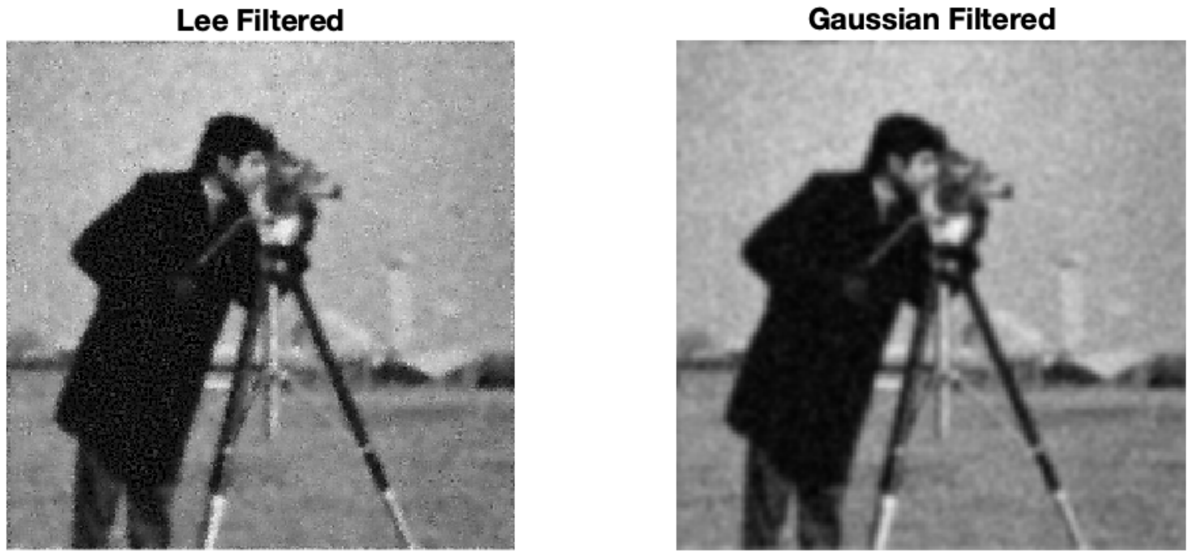


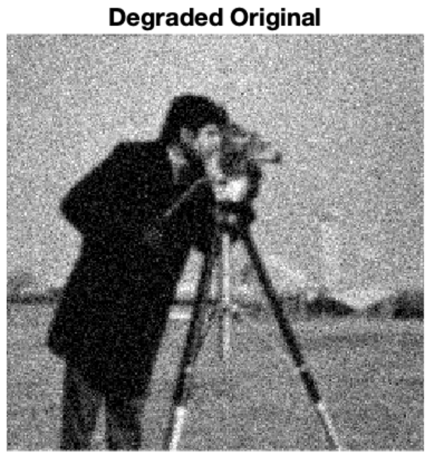
Figure 9: 5x5 Lee Filtered vs Gaussian w/ std dev of 30 in frequency domain

In general it can be said that, in Figure 9, the Gaussian filter smoothes the degraded image better than the Lee filter.

Comparing both low and high frequency areas, it is noted that in the low frequency components of the image the performance is roughly the same. The smoothness/blurring applied to the degraded image in the background aspects reduces the granularity (shot noise-esque disturbance) with a similar fitness. In regards to the high frequency detailing, a difference in performance is evident. The Lee filter does a relatively better job at maintaining the edges and detailing of the cameraman. The edges of the Gaussian filter are uniformly smoothed out due to the nature of the Gaussian filter model; it does not take image details into consideration. The Lee filter is an average of the neighborhood region, in low regions, so when this is applied in areas where high frequency details exists (ie. edges), the original image details are retained. It is easy to see the difference in the center main focal points of the image (cameraman, tripod, camera).

The PSNR value of the Gaussian filtered image is 20.833676470613334 whereas the PSNR value for the Lee filtered image is 19.813391412696525. Despite the PSNR of the Gaussian image being higher, the Lee image is still better for the HVS, due to edges. It is pertinent to understand that the PSNR value is with reference to the degraded image, not to the original image, which implies the Lee filtered image is not as similar to the degraded image as the Gaussian filter is.

8.



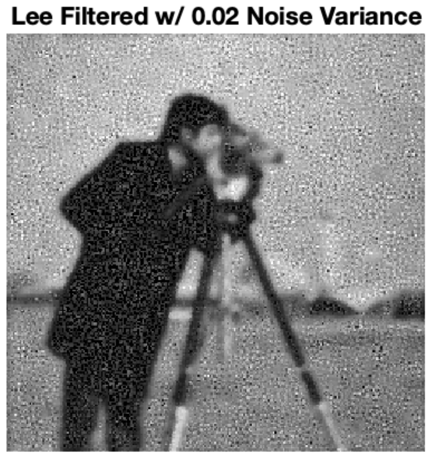


Figure 10: Original Degraded vs 5x5 Lee Filtered w/ 0.02 Noise Variance

The original noise variance of the flat region used was approximately 0.01. In Figure 10, the result from applying the Lee filter with a doubled noise variance, 0.02, is compared against the original degraded image.

The Lee filtered image with a high variance is smoother than both the original degraded and original Lee filtered image with the normal noise variance. The result is due to the derivation of the Lee filter shown in the lab4.pdf document under Section 3. From the equation it is ascertained that increasing the variance, or using high noise variances, will cause the filter to not retain the original image detailing; instead, an averaging is applied. The averaging increases the smoothness as the high frequency details in the images lose their sharpness.

The PSNR value of the high noise variance image is 13.706692356357223, which is lower than 19.813391412696525, PSNR of original Lee image, by a decent margin. The drop in PSNR value can be attributed to the noise variance not being reflective of the image. The original variance calculated was from the image and increasing it was just an experiment to explore the effects of the noise variance in Lee filters. The original variance would produce a better image evident both visually to the HVS and mathematically through PSNR values.

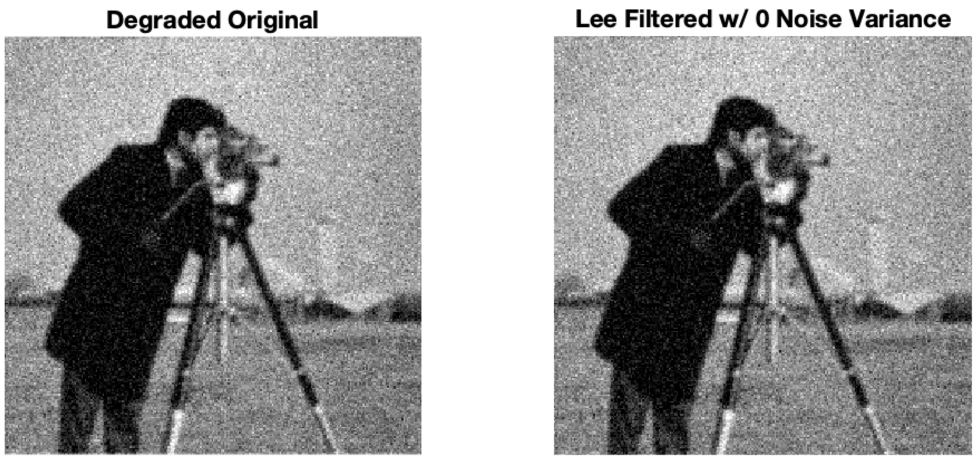


Figure 11: Original Degraded vs 5x5 Lee Filtered w/ 0 Noise Variance

Figure 11, compares the original degraded image and the Lee filtered image with no variance. By visual inspection, it appears that both images are highly similar, in fact look like identical copies.

When choosing a variance lower than the original value, a noise variance of 0 was chosen for simplicity and to explore an extreme effect of noise variance on Lee filters. It turns out that the K value in the derivation evaluates to 1, completely eliminating the effects of the mean and variance of the pixel intensity levels in the original degraded image. It results in the pixel intensities of the original degraded image being multiplied by 1, resulting in no change and retaining the exact original image.

The PSNR value with no noise variance resulted in an infinity value in MATLAB, as both images are identical copies.

9.

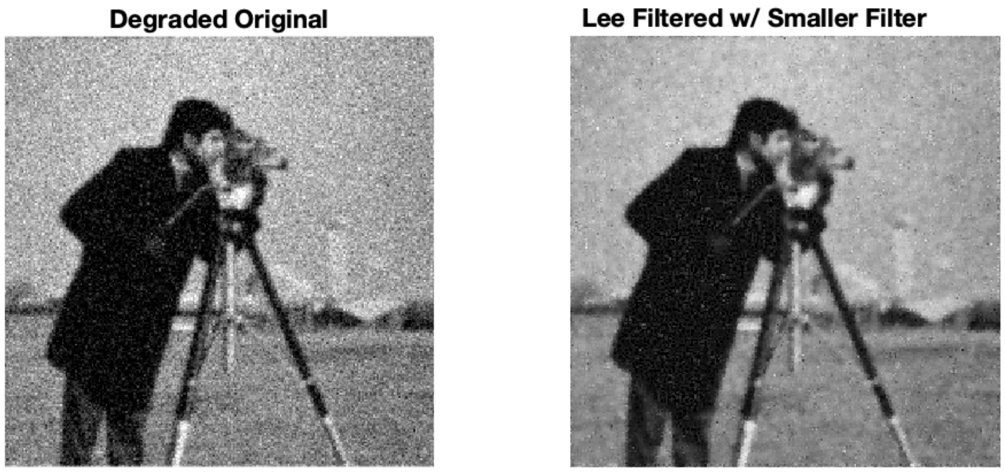


Figure 12: Original Degraded vs 4x4 Lee Filtered w/ Original Noise Variance

In Figure 12, a smaller 4x4 filter size was used to apply the Lee filter and it was compared against the original degraded. The smaller filter size was not as smooth as the original Lee filtered image nor the Lee filtered image with the larger filter size explored in Figure 13. This is due to the smaller window size being less prone to deviations and noise in the image. Especially in areas with coarse, low frequency, details the image was not as smooth. The window size was reduced and with fewer local regional data points, the general smoothness was reduced; however, the edges are better preserved and the noise is also reduced there. The better edge preservation and detailing is noted in the cameraman and the camera stand and main center focal points of the image.

The resulting PSNR value compared to the original degraded image is 19.376867193131304, which is slightly lower than the original Lee filter image by approximately 0.43 units.

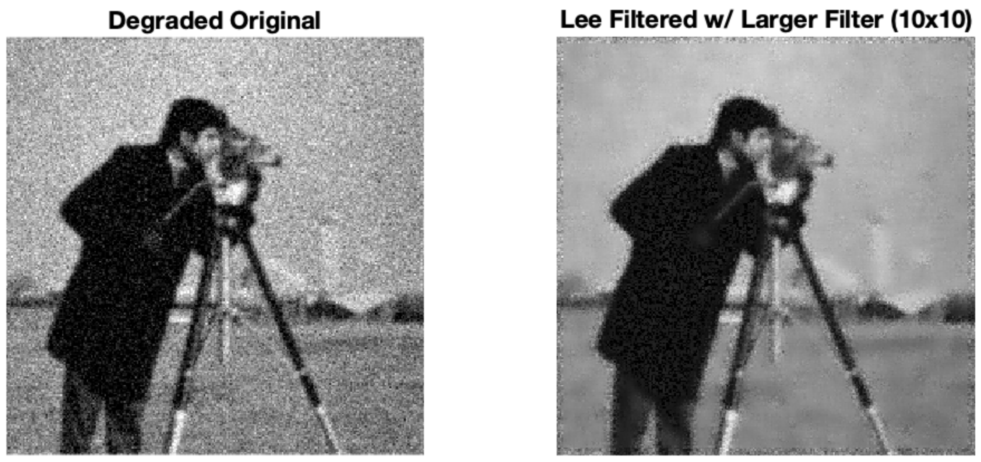


Figure 13: Original Degraded vs 10x10 Lee Filtered w/ Original Noise Variance

In Figure 13 above, a larger sized filter was used. A 10x10 neighborhood size filter to be exact. This increase in filter size, greatly increases the number of data points that are used to develop the new image. It resulted in a much smoother image overall in low frequency areas with minimal levels of detail. The background of the image (ie., skies, ground, building in the back) is relatively smoother compared to the smaller filter size image. However, with the increased smoothness, it also introduced more noise near the edges of the cameraman and camera stand. There is increased static noise surrounding the edges from the large filter size compared to the original or smaller filter size Lee filtered images. This effect is caused by overlapping which occurs due to the large window size and results in inaccurate values for the mean and variance for the window regions near edges and high detailing.

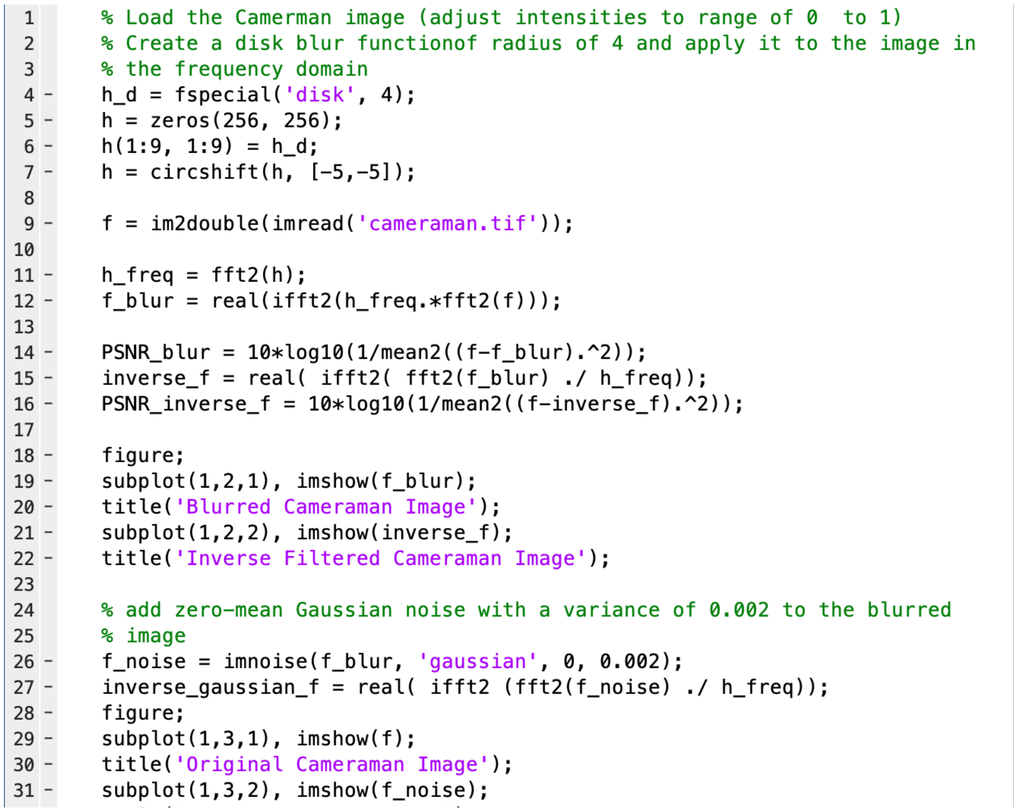
The PSNR value for the larger window size Lee filtered image is 20.707564143804788, which is higher than both the smaller window size Lee filtered image and the original Lee filtered image. Therefore, between the two differing window size filters, the larger filter size is deemed better.

4. Conclusion

In conclusion, the goal of this lab was to explore image restoration techniques. In Section 2, we investigated inverse and Wiener filtering. We first applied a disk blur function to the spatial-frequency representation of the image and applied to it the inverse filter. It was found that the resulting image was of a higher image quality. This was because the blurred image was low frequency and the inverse filter is a high pass filter; thus, the inverse filter attenuated the lower frequency, undoing the effects of blurring. As a result, we achieve a sharper image. We also added additive Gaussian noise to the blurred image and applied the inverse and Wiener filter to it. The inverse filter did not perform well as the high frequency noise (Gaussian noise) was amplified; this is due to the filter being high pass. The Wiener filter performed better as it minimizes the MSE so it optimizes the tradeoff between noise smoothing and inverse filtering. It is able to remove the additive noise while unblurring the image as well.

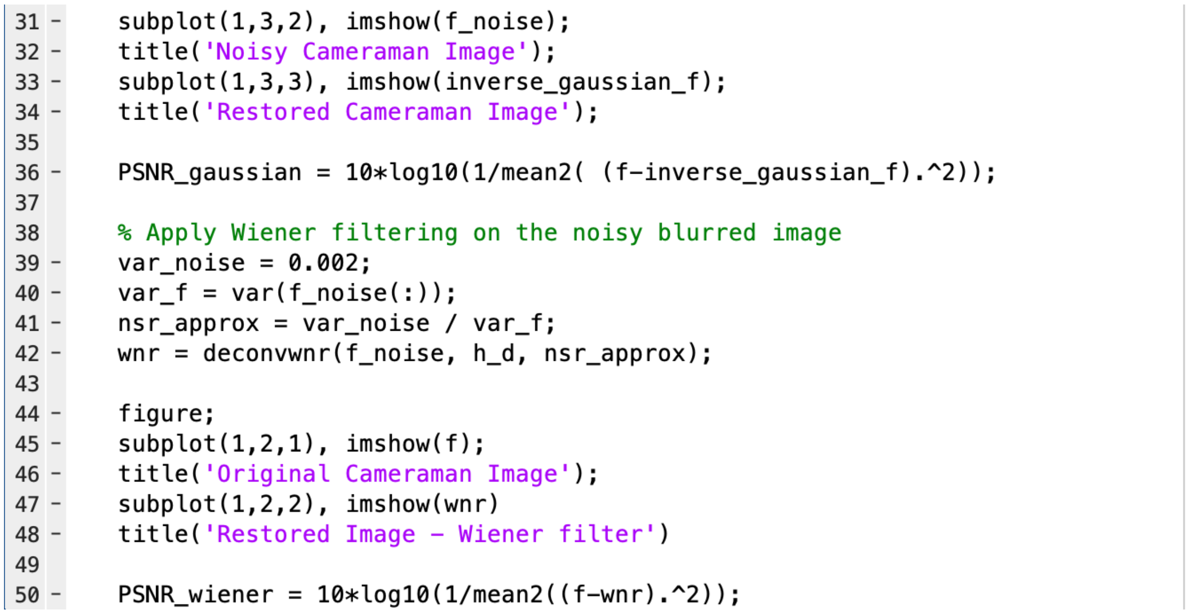
In Section 3, six different filters were applied to the degraded image to explore the effects of the noise variance and window size on adaptive filters, specifically the Lee filter. A Gaussian filter with a standard deviation of 30 in the frequency domain was also explored for comparison against the Lee filter. A flat region was initially chosen and the noise variance was calculated and with a 5x5 window size, the original Lee filtered image was reached with a PSNR value of 19.813391412696525. All PSNR values can be found in Table 2 for this section. The Gaussian filter was compared next and although it had a better PSNR than the original Lee image, it removed detailing from the edges due to its unbiased uniform smoothing. The Lee filtered images with an increased noise variance, resulted in a lower PSNR, compared to the original Lee filtered image, as instead of retaining the original edge and high detailing info, it was averaged. The 0 noise variance model produced an identical image as the Lee formulation led to the scalar K being valued at 1 and no effect of mean nor variance was introduced. The PSNR for this was infinity, or approaching it, as the two copies were identical. Last, in Section 3 the window sizes of 4x4 and 10x10 were applied. The smaller window size preserved details better than the larger window size and the larger window size resulted in an overall smoother image. The PSNR value of the l0x10 filter size image was higher than the smaller 4x4 window size at 20.707564143804788 and 19.376867193131304, respectively.

Appendix A



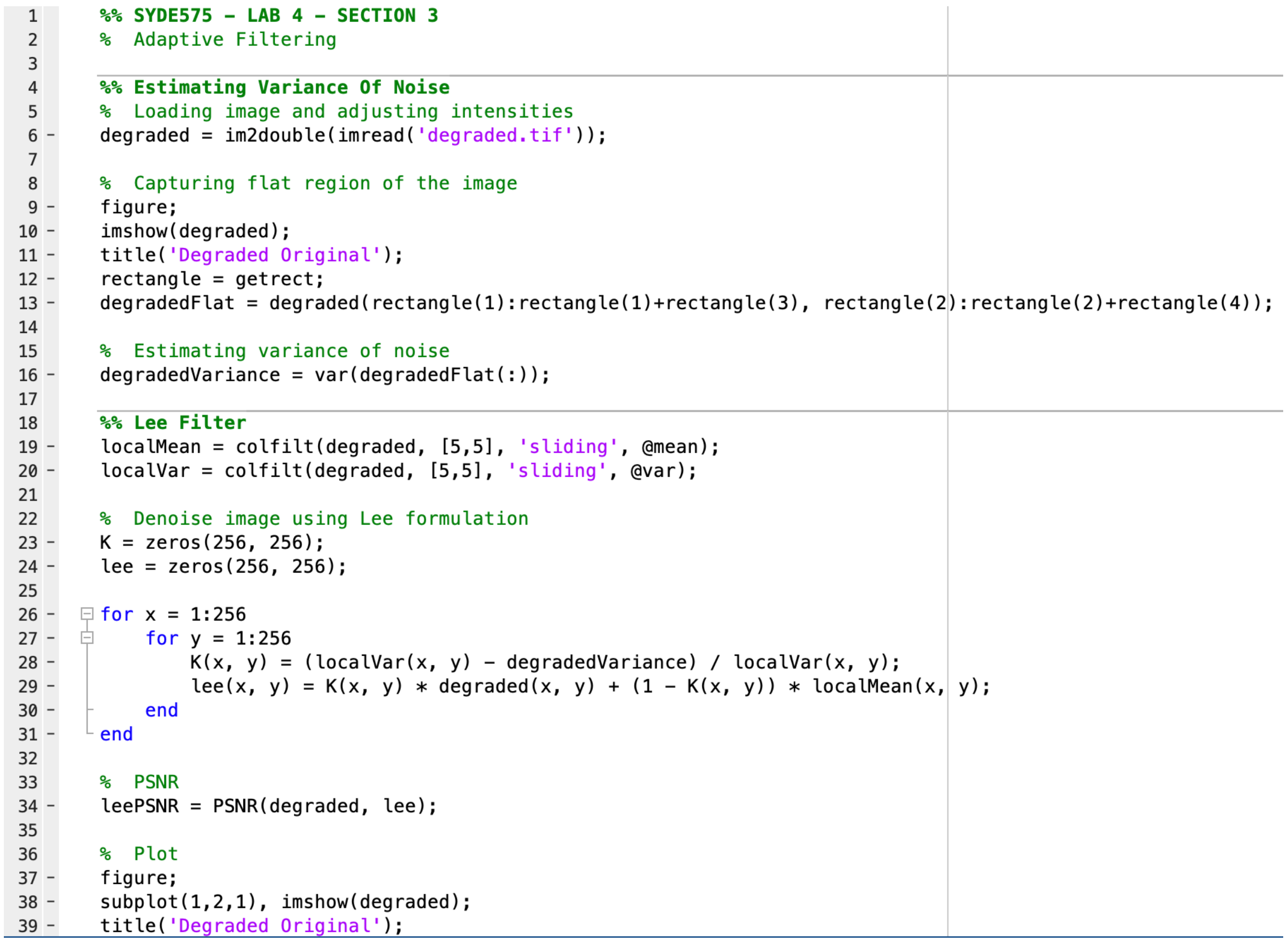
```
% Load the Camerman image (adjust intensities to range of 0  to 1)
% Create a disk blur functionof radius of 4 and apply it to the image in
% the frequency domain
h_d = fspecial('disk', 4);
h = zeros(256, 256);
h(1:9, 1:9) = h_d;
h = circshift(h, [-5,-5]);

f = im2double(imread('cameraman.tif'));

h_freq = fft2(h);
f_blur = real(ifft2(h_freq.*fft2(f)));

PSNR_blur = 10*log10(1/mean2((f-f_blur).^2));
inverse_f = real( ifft2( fft2(f_blur) ./ h_freq));
PSNR_inverse_f = 10*log10(1/mean2((f-inverse_f).^2));

figure;
subplot(1,2,1), imshow(f_blur);
title('Blurred Cameraman Image');
subplot(1,2,2), imshow(inverse_f);
title('Inverse Filtered Cameraman Image');

% add zero-mean Gaussian noise with a variance of 0.002 to the blurred
% image
f_noise = imnoise(f_blur, 'gaussian', 0, 0.002);
inverse_gaussian_f = real( ifft2 (fft2(f_noise) ./ h_freq));
figure;
subplot(1,3,1), imshow(f);
title('Original Cameraman Image');
subplot(1,3,2), imshow(f_noise);
```

Figure A.1.1: Section 2 code



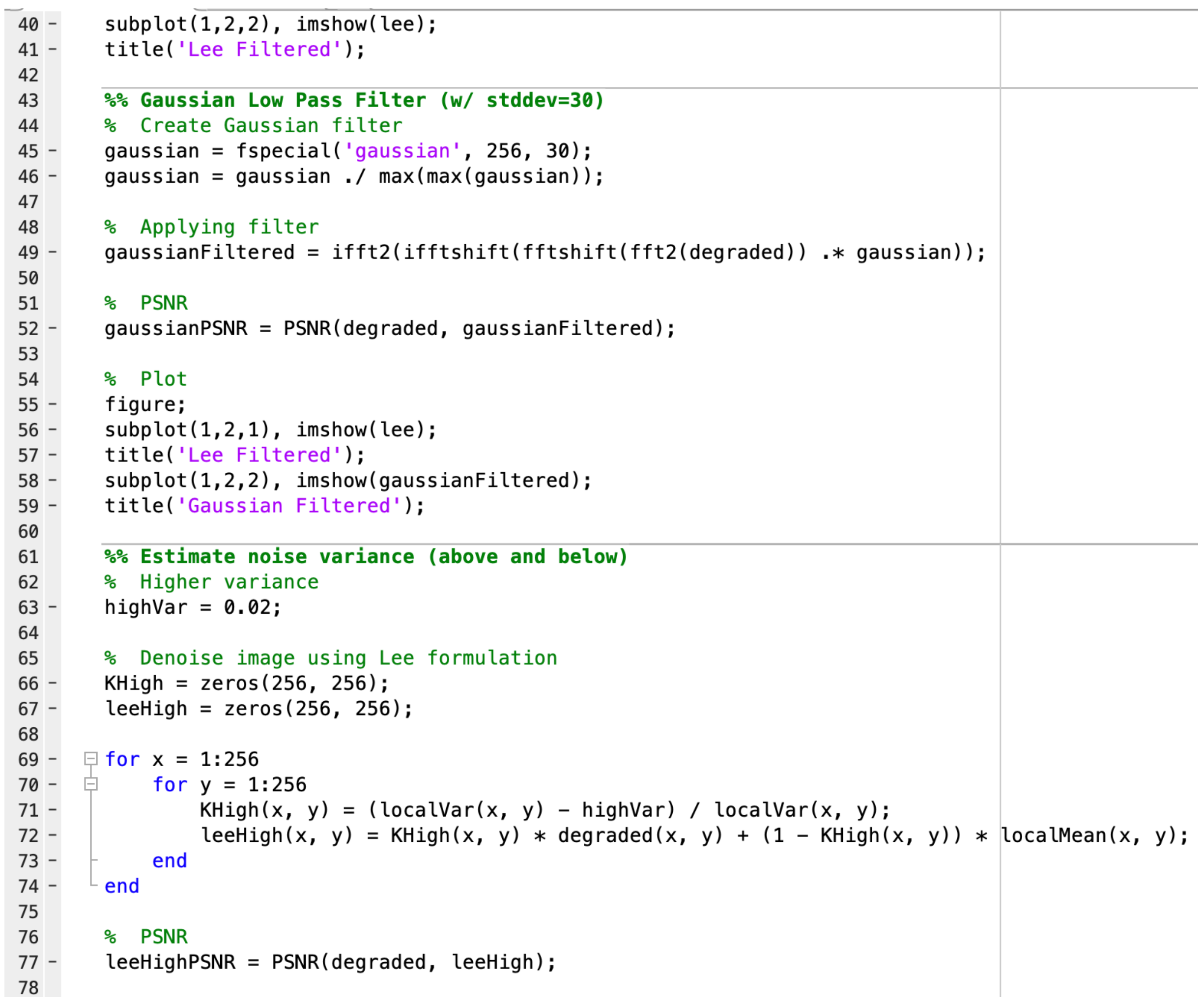
```
subplot(1,3,2), imshow(f_noise);
title('Noisy Cameraman Image');
subplot(1,3,3), imshow(inverse_gaussian_f);
title('Restored Cameraman Image');

PSNR_gaussian = 10*log10(1/mean2( (f-inverse_gaussian_f).^2));

% Apply Wiener filtering on the noisy blurred image
var_noise = 0.002;
var_f = var(f_noise(:));
nsr_approx = var_noise / var_f;
wnr = deconvwnr(f_noise, h_d, nsr_approx);

figure;
subplot(1,2,1), imshow(f);
title('Original Cameraman Image');
subplot(1,2,2), imshow(wnr)
title('Restored Image - Wiener filter')

PSNR_wiener = 10*log10(1/mean2((f-wnr).^2));
```

Figure A.1.2: Section 2 code (continued)



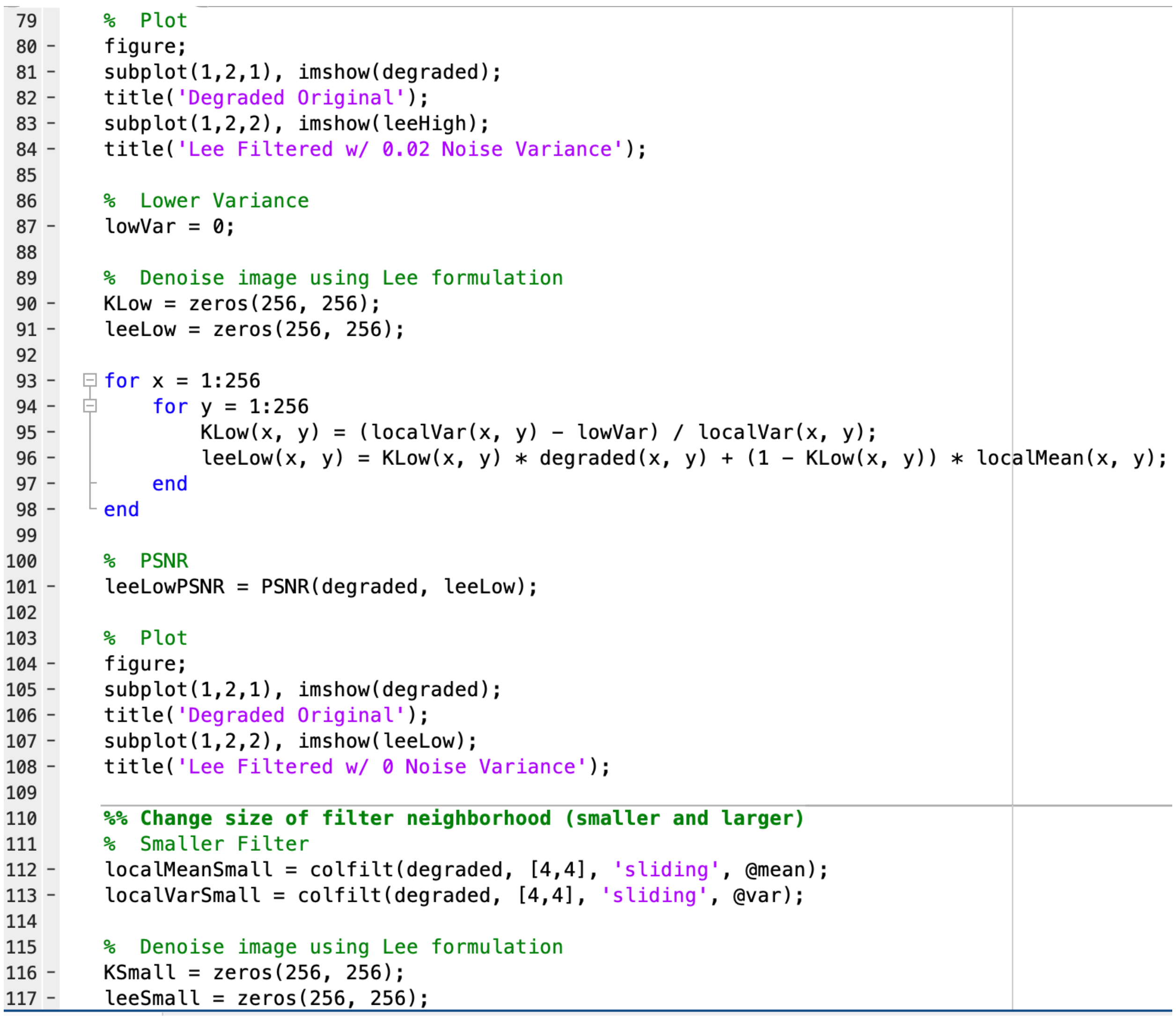
```matlab
%% SYDE575 - LAB 4 - SECTION 3
%  Adaptive Filtering

%% Estimating Variance Of Noise
%  Loading image and adjusting intensities
degraded = im2double(imread('degraded.tif'));

%  Capturing flat region of the image
figure;
imshow(degraded);
title('Degraded Original');
rectangle = getrect;
degradedFlat = degraded(rectangle(1):rectangle(1)+rectangle(3), rectangle(2):rectangle(2)+rectangle(4));

%  Estimating variance of noise
degradedVariance = var(degradedFlat(:));

%% Lee Filter
localMean = colfilt(degraded, [5,5], 'sliding', @mean);
localVar = colfilt(degraded, [5,5], 'sliding', @var);

%  Denoise image using Lee formulation
K = zeros(256, 256);
lee = zeros(256, 256);

for x = 1:256
    for y = 1:256
        K(x, y) = (localVar(x, y) - degradedVariance) / localVar(x, y);
        lee(x, y) = K(x, y) * degraded(x, y) + (1 - K(x, y)) * localMean(x, y);
    end
end

%  PSNR
leePSNR = PSNR(degraded, lee);

%  Plot
figure;
subplot(1,2,1), imshow(degraded);
title('Degraded Original');
```

Figure A.2.1: Section 3 Code



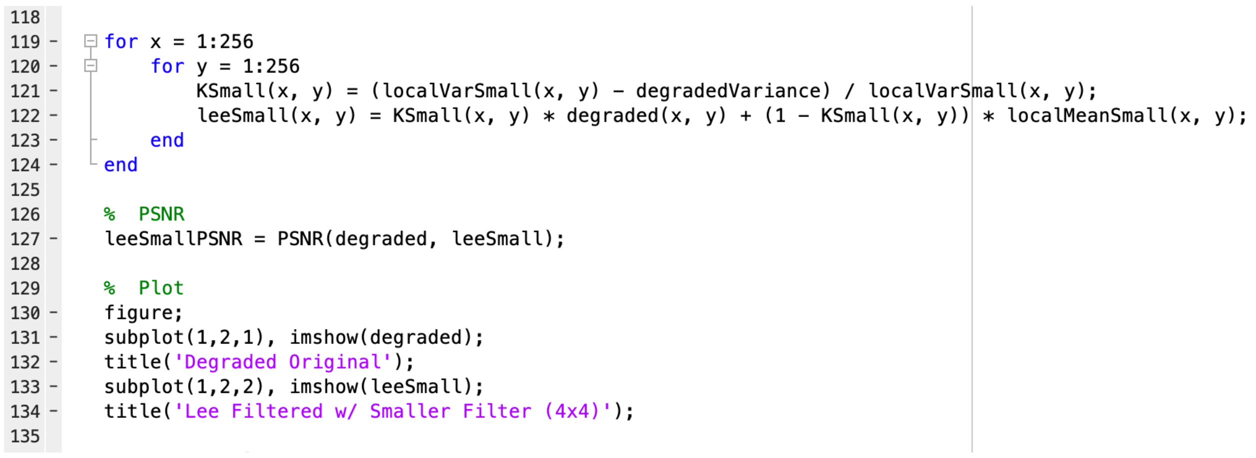
```
subplot(1,2,2), imshow(lee);
title('Lee Filtered');

%% Gaussian Low Pass Filter (w/ stddev=30)
%  Create Gaussian filter
gaussian = fspecial('gaussian', 256, 30);
gaussian = gaussian ./ max(max(gaussian));

%  Applying filter
gaussianFiltered = ifft2(ifftshift(fftshift(fft2(degraded)) .* gaussian));

%  PSNR
gaussianPSNR = PSNR(degraded, gaussianFiltered);

%  Plot
figure;
subplot(1,2,1), imshow(lee);
title('Lee Filtered');
subplot(1,2,2), imshow(gaussianFiltered);
title('Gaussian Filtered');

%% Estimate noise variance (above and below)
%  Higher variance
highVar = 0.02;

%  Denoise image using Lee formulation
KHigh = zeros(256, 256);
leeHigh = zeros(256, 256);

for x = 1:256
    for y = 1:256
        KHigh(x, y) = (localVar(x, y) - highVar) / localVar(x, y);
        leeHigh(x, y) = KHigh(x, y) * degraded(x, y) + (1 - KHigh(x, y)) * localMean(x, y);
    end
end

%  PSNR
leeHighPSNR = PSNR(degraded, leeHigh);
```

Figure A.2.2: Section 3 Code (continued)



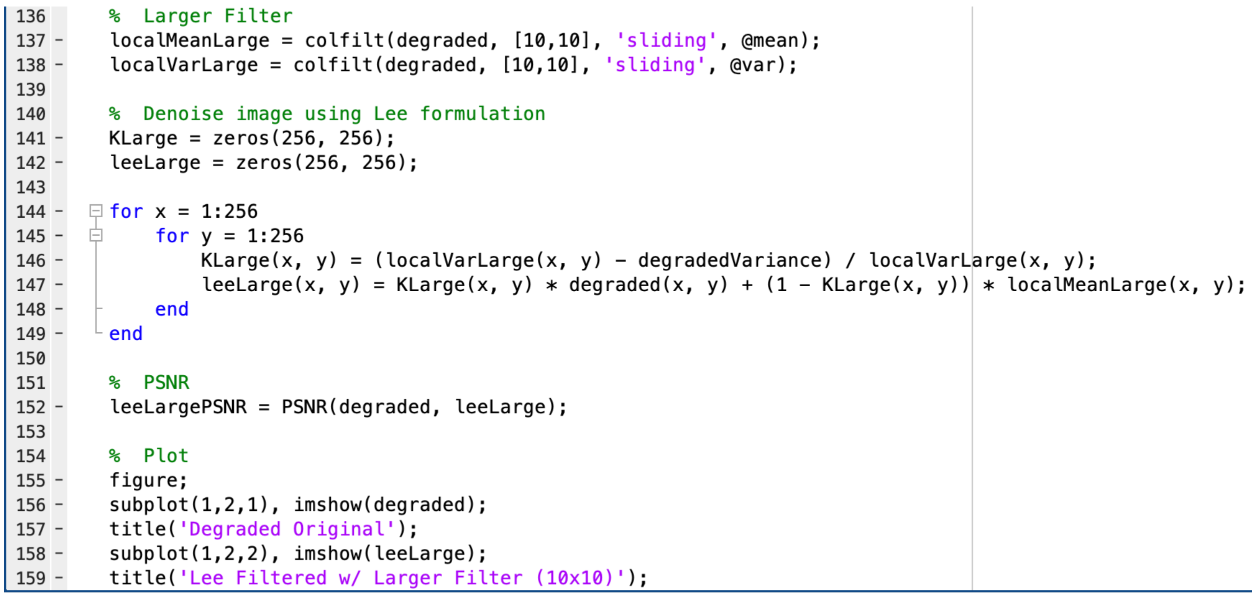
```matlab
%  Plot
figure;
subplot(1,2,1), imshow(degraded);
title('Degraded Original');
subplot(1,2,2), imshow(leeHigh);
title('Lee Filtered w/ 0.02 Noise Variance');

%  Lower Variance
lowVar = 0;

%  Denoise image using Lee formulation
KLow = zeros(256, 256);
leeLow = zeros(256, 256);

for x = 1:256
    for y = 1:256
        KLow(x, y) = (localVar(x, y) - lowVar) / localVar(x, y);
        leeLow(x, y) = KLow(x, y) * degraded(x, y) + (1 - KLow(x, y)) * localMean(x, y);
    end
end

%  PSNR
leeLowPSNR = PSNR(degraded, leeLow);

%  Plot
figure;
subplot(1,2,1), imshow(degraded);
title('Degraded Original');
subplot(1,2,2), imshow(leeLow);
title('Lee Filtered w/ 0 Noise Variance');

%% Change size of filter neighborhood (smaller and larger)
%  Smaller Filter
localMeanSmall = colfilt(degraded, [4,4], 'sliding', @mean);
localVarSmall = colfilt(degraded, [4,4], 'sliding', @var);

%  Denoise image using Lee formulation
KSmall = zeros(256, 256);
leeSmall = zeros(256, 256);
```

Figure A.2.3: Section 3 Code (continued)



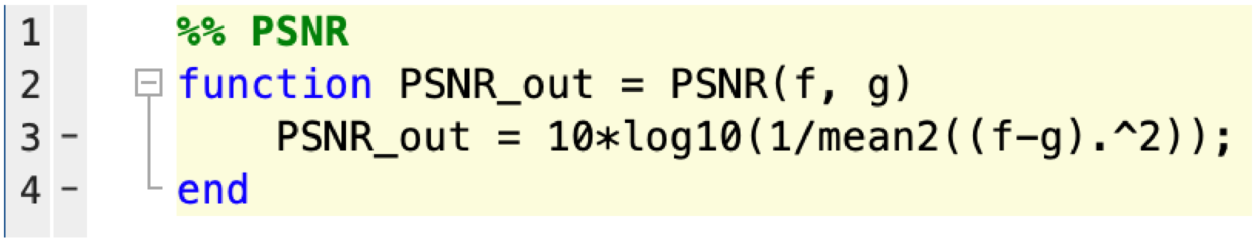
```matlab

for x = 1:256
    for y = 1:256
        KSmall(x, y) = (localVarSmall(x, y) - degradedVariance) / localVarSmall(x, y);
        leeSmall(x, y) = KSmall(x, y) * degraded(x, y) + (1 - KSmall(x, y)) * localMeanSmall(x, y);
    end
end

%  PSNR
leeSmallPSNR = PSNR(degraded, leeSmall);

%  Plot
figure;
subplot(1,2,1), imshow(degraded);
title('Degraded Original');
subplot(1,2,2), imshow(leeSmall);
title('Lee Filtered w/ Smaller Filter (4x4)');

```

Figure A.2.4: Section 3 Code (continued)



```
%  Larger Filter
localMeanLarge = colfilt(degraded, [10,10], 'sliding', @mean);
localVarLarge = colfilt(degraded, [10,10], 'sliding', @var);

%  Denoise image using Lee formulation
KLarge = zeros(256, 256);
leeLarge = zeros(256, 256);

for x = 1:256
    for y = 1:256
        KLarge(x, y) = (localVarLarge(x, y) - degradedVariance) / localVarLarge(x, y);
        leeLarge(x, y) = KLarge(x, y) * degraded(x, y) + (1 - KLarge(x, y)) * localMeanLarge(x, y);
    end
end

%  PSNR
leeLargePSNR = PSNR(degraded, leeLarge);

%  Plot
figure;
subplot(1,2,1), imshow(degraded);
title('Degraded Original');
subplot(1,2,2), imshow(leeLarge);
title('Lee Filtered w/ Larger Filter (10x10)');
```

Figure A.2.5: Section 3 Code (continued)



```
%% PSNR
function PSNR_out = PSNR(f, g)
    PSNR_out = 10*log10(1/mean2((f-g).^2));
end
```

Figure A.3.1: PSNR Code