SYDE 575 Image Processing
Lab 4: Restoration

Group 23: Laura Chambers, ID: [redacted]

1 Introduction

The purpose of this lab is to gain an understanding of image restoration in the frequency domain, including inverse, Wiener, and Lee filtering. These techniques can be used to correct for image distortion and additive noise.

Inverse and Wiener filters are global filters that apply the same process to the whole image. In contrast, the Lee filter is an adaptive filter that responds to the underlying characteristics of the image based on local statistical measures.

The MATLAB code for this lab can be found in the Appendix.

2 Image Restoration in the Frequency Domain

Figure 2-1 Original, blurred, and inverse filtered images and PSNR values

1. The inverse filtered image has a very high PSNR value, indicating that the error between the original image and the inverse filtered image is essentially zero. The inverse filtered image is equivalent to the original image. In comparison, the blurred image has a low PSNR of 16.90. The inverse filter effectively reversed the blur function that was applied to the image. Direct inverse filtering is quick and effective when the blur model is accurately known.

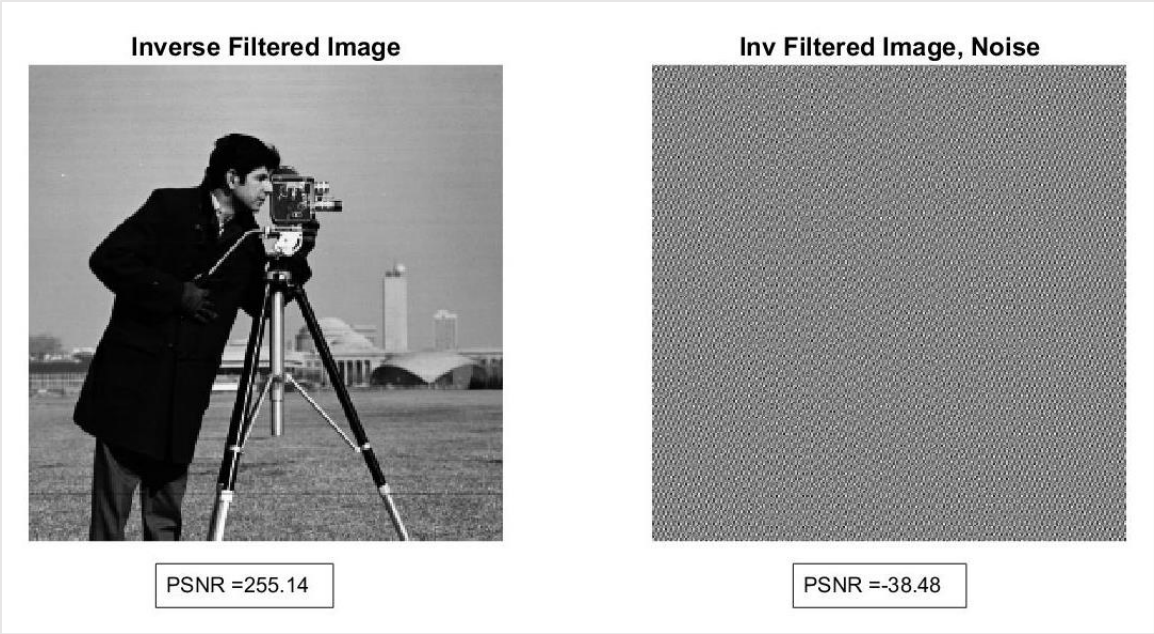

Figure 2-2 Comparison of inverse filtering of original and noisy blurred images

2. When there is additive noise in the image, direct inverse filtering is ineffective. As can be observed in Figure 2-2, the inverse filtered image of the noisy blurred image is unusable. The filtered image does not resemble the original image at all and has a negative PSNR. The blurring function acts as a low pass filter; therefore, its inverse acts as a high pass filter. Noise is high frequency, so it remains and dominates the inverse filtered image.

3. Based on these results, it is evident that inverse filtering is effective for restoring blurred images, especially when the blur model is well known. However, inverse filtering is very ineffective for restoring noise degraded images.

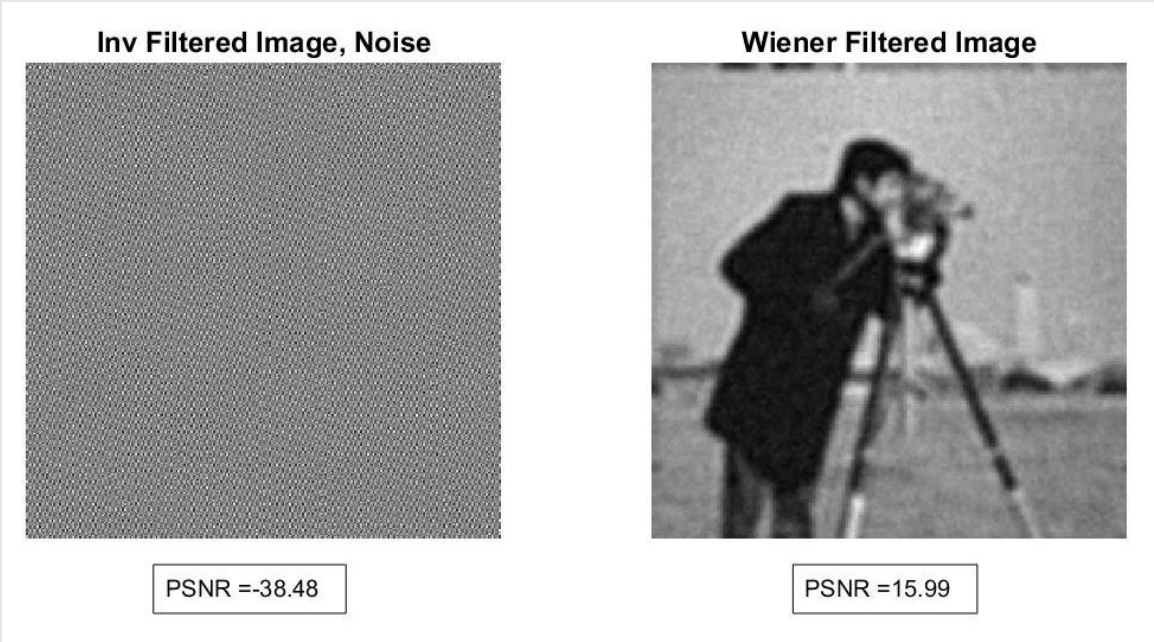

Figure 2-3 Comparison of inverse filter and Wiener filter for noisy blurred image

4. The Wiener filtered produced a far better restored image compared to the inverse filter. The Wiener filtered image still looks quite blurry, but it is recognizable. While the inverse filtered image had a negative PSNR value, the Wiener filtered image has a positive PSNR value that is comparable to that of the blurred image in Figure 2-1.

   The Wiener filter aims to minimize the mean squared error between the original and degraded images. It assesses the signal to noise ratio (SNR) of the spectrum and filters accordingly. If SNR is high, the Wiener filter retains the signal. If SNR is low, the Wiener filter attenuates the signal. This means that in a noise-degraded image, SNR is low for high frequencies, so they are attenuated.

5. Based on these results, Wiener filtering is much more effective than inverse filtering at reducing additive noise in degraded images.

3 Adaptive Filtering

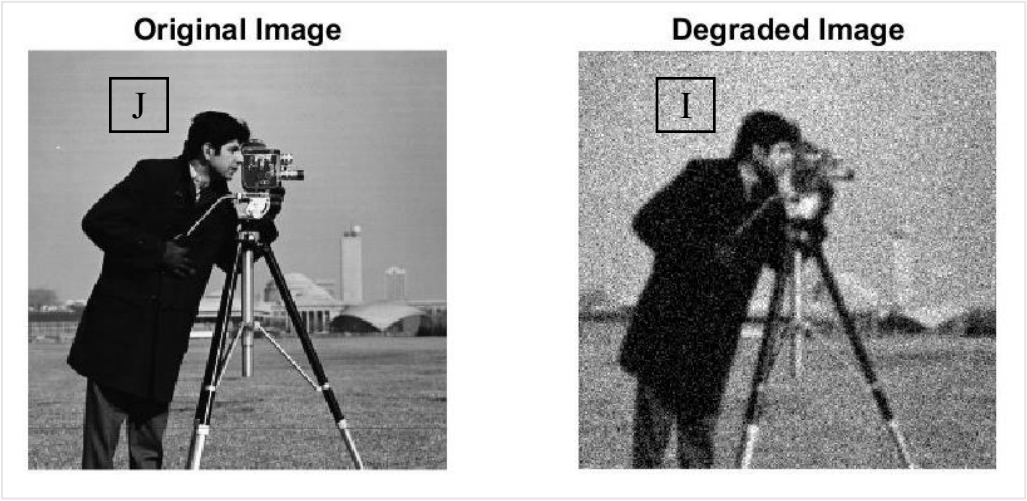

Figure 3-1 Flat sections used in noise variance estimate

6. To estimate noise variance, a flat section (I) was taken from the top left of the degraded image. The same portion (J) of the original cameraman image was also isolated. I was then subtracted from J to produce a noise-only section from which the variance was taken. Relevant MATLAB code is shown below from file lab4_prt3_adaptive_filtering.

```
% Find variance of noise in degraded image
f = im2double(imread('cameraman.tif');
I = f(25:65, 1:40); %flat region of degraded image
J = g(25:65, 1:40); %same region of undegraded image
noise_only = J - I
noise_var = var(J(:));
```

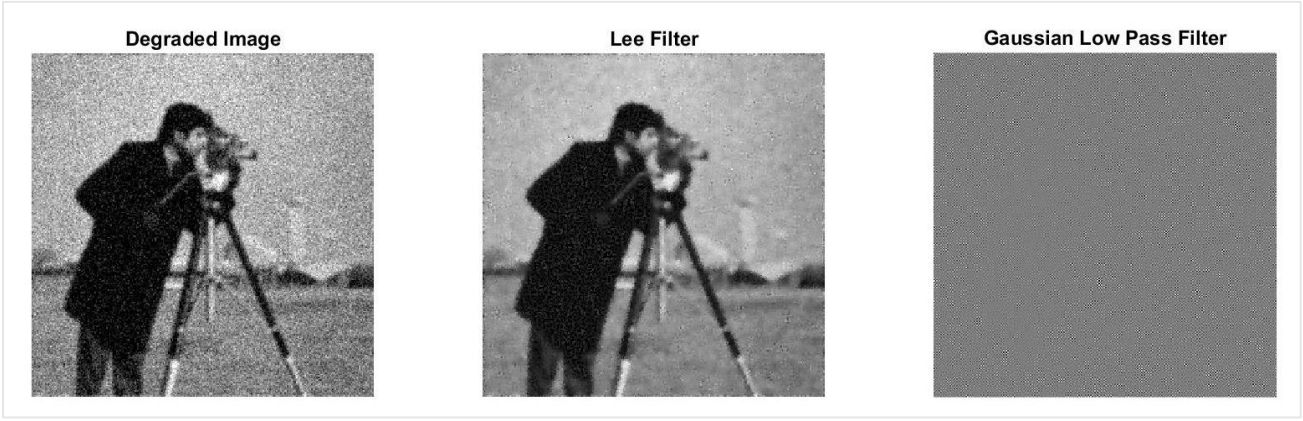

Figure 3-2 Results of Lee filter and Gaussian low pass filter

7. The result of the Lee filter is much better than the result of the Gaussian low pass filter. The result of the low pass filter does not resemble the original image. When the standard deviation of the Gaussian

low pass filter was increased for experimentation, the shapes in the image became visible. This indicates that the low pass filter with standard deviation of 30 has attenuated high frequencies that contribute noise, edges, and detail to the image. As a result, the image is unrecognizable.

In comparison, the result of the Lee filter is recognizable and less speckled than the degraded image. Areas like the sky have been smoothed out while edges have been retained.

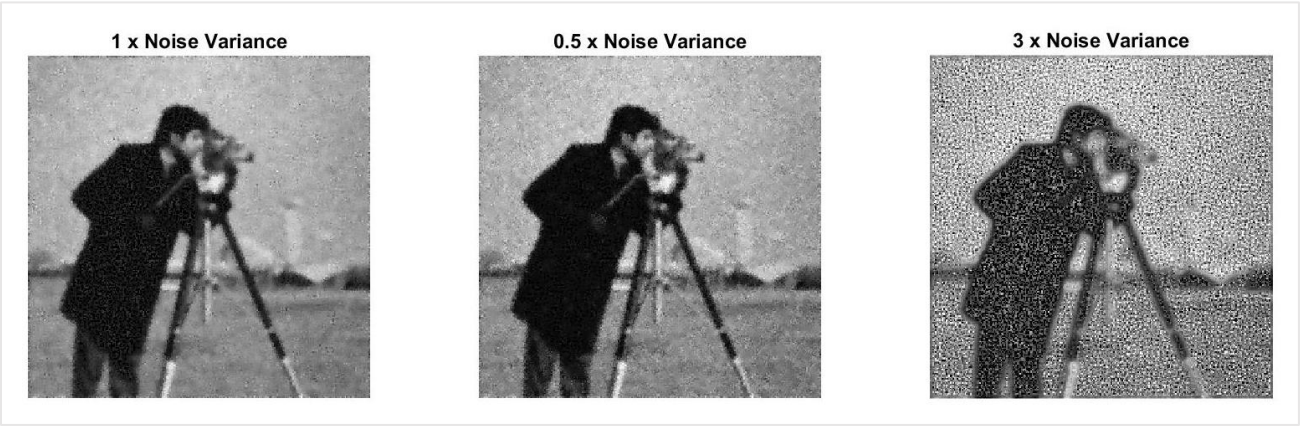

Figure 3-3 Effect of noise variance estimate

8. The noise variance estimate term, σ_n^2, appears in the numerator of the equation for K in the Lee filter.

$$K = \frac{\sigma_g^2(i,i) - \sigma_n^2}{\sigma_g^2(i,j)} \quad ; \quad \hat{f} = K(i,j)g(i,j) + (1-K)\overline{g(i,j)}$$

If σ_n^2 increases, K decreases, and the local mean signal is weighted more heavily in $\hat{f}$. In general, σ_n^2 is expected to be less than the local variance in the image. If σ_n^2 increases to the point that is exceeds $\sigma_g^2(i,i)$, K becomes negative. In this case, the original signal is negatively weighted, and the weight of the local average is greater than one. In Figure 3-3 3 x Noise Variance this seems to be the case. The image looks noisier -- it appears that more pixels have become dark because due to negative values.

If σ_n^2 decreases, K increases, and the original signal is weighted more heavily in $\hat{f}$. This can be seen in Figure 3-3 0.5 x Noise Variance where more of the original speckle is visible compared to the 1 x Noise Variance.

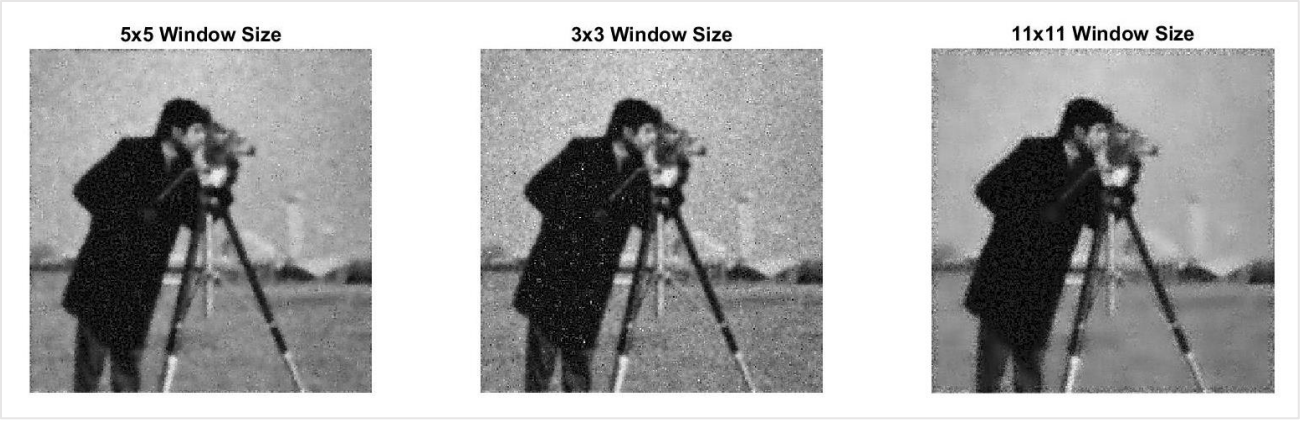

Figure 3-4 Effect of the size of the filter neighbourhood

9. As the window size decreases, there are fewer samples included in in the estimate of local mean and variance. This means that the estimate of statistics may not be very good; however, it decreases the likelihood of outliers skewing the estimate. In Figure 3-4, the 3x3 Window image appears more speckled compared to the images with larger window sizes.

   As the window size increases, there are more samples included in the estimate of local mean and variance, which improves the estimate. However, a larger window size increases the risk of having outliers within the window that skew estimates. In Figure 3-4, the 11x11 Window image appears less speckled compared to the images with smaller window sizes.

4 Conclusions

This lab provided hand-on experience with image restoration techniques in the frequency domain and practice using image processing tools in MATLAB.

From this lab, it is evident that direct inverse filtering is effective when the blur model of the image is well known and ineffective for restoring additive noise. The Wiener filter is far superior than inverse filtering when there is noise in the image.

The Lee filter adapts to detail in an image such that edges and details are preserved while noise is averaged out. The results of the Lee filter depend on the noise variance estimate, which affects how the image signal and local mean are weighted (K). Results also depend on the size of the filter neighbourhood. A smaller size means less reliable statistics but lower risk of outliers, while the opposite if true of a larger size.

Appendix – MATLAB Code

File: lab4_prt2_restoration_freq_domain.m

```
clear all; close all; clc;
fontSize = 12;

% Load image
f = im2double(imread('cameraman.tif'));
```

```
% Create blurred image
h = fspecial('disk',4);
h_freq = fft2(h, size(f,1), size(f,2)); %pad h
f_blur = real(ifft2(h_freq.*fft2(f)));
psnr1 = PSNR_norm(f, f_blur);

% Apply inverse filtering
f_blur_inv = real(ifft2(fft2(f_blur)./h_freq));
psnr2 = PSNR_norm(f, f_blur_inv);

% Add zero-mean Gaussian noise to the blurred image
f_blur_n = imnoise(f_blur,'gaussian',0,0.002);
f_blur_n_inv = real(ifft2(fft2(f_blur_n)./h_freq));
psnr3 = PSNR_norm(f, f_blur_n_inv);

% Apply Wiener filter to the noisy blurred image
f_wiener = deconvwnr(f_blur_n, h, 0.15);
psnr4 = PSNR_norm(f, f_wiener);

% Plot images - question 1
figure(1);
subplot(1, 3, 1);
imshow(f);
title('Original Image', 'FontSize', fontSize);
subplot(1, 3, 2);
imshow(f_blur, []);
title('Blurred Image', 'FontSize', fontSize);
txt = strcat('PSNR = ', sprintf('%.2f',psnr1));
annotation('textbox', [0.47, 0.1, 0.1, 0.1], 'string', txt);
subplot(1, 3, 3);
imshow(f_blur_inv, []);
title('Inverse Filtered Image', 'FontSize', fontSize);
txt = strcat('PSNR = ', sprintf('%.2f',psnr2));
annotation('textbox', [0.75, 0.1, 0.1, 0.1], 'string', txt);

% Plot images - questions 2, 3
figure(2);
subplot(1, 2, 1);
imshow(f_blur_inv, []);
title('Inverse Filtered Image', 'FontSize', fontSize);
txt = strcat('PSNR = ', sprintf('%.2f',psnr2));
annotation('textbox', [0.22, 0.1, 0.1, 0.1], 'string', txt);
subplot(1, 2, 2);
imshow(f_blur_n_inv, []);
title('Inv Filtered Image, Noise', 'FontSize', fontSize);
txt = strcat('PSNR = ', sprintf('%.2f',psnr3));
annotation('textbox', [0.67, 0.1, 0.1, 0.1], 'string', txt);

% Plot images - questions 4, 5
figure(3);
subplot(1, 2, 1);
imshow(f_blur_n_inv, []);
title('Inv Filtered Image, Noise', 'FontSize', fontSize);
txt = strcat('PSNR = ', sprintf('%.2f',psnr3));
annotation('textbox', [0.22, 0.1, 0.1, 0.1], 'string', txt);
subplot(1, 2, 2);
imshow(f_wiener, []);
title('Wiener Filtered Image', 'FontSize', fontSize);
txt = strcat('PSNR = ', sprintf('%.2f',psnr4));
annotation('textbox', [0.67, 0.1, 0.1, 0.1], 'string', txt);
```

File: lab4_prt3_adaptive_filtering.m

```
clear all; close all; clc;
fontSize = 12;

% Load degraded image
g = im2double(imread('degraded.tif'));

% Find variance of noise in degraded image
f = im2double(imread('cameraman.tif'));
I = f(25:65, 1:40); %flat region of degraded image
J = g(25:65, 1:40); %same region of undegraded image
noise_only = J - I
noise_var = var(J(:));

% Construct and apply Lee filter with different noise variances
% and window sizes
f_hat_1 = lee_filter(g, noise_var, 5);
f_hat_2 = lee_filter(g, 0.5*noise_var, 5);
f_hat_3 = lee_filter(g, 3*noise_var, 5);
f_hat_4 = lee_filter(g, noise_var, 3);
f_hat_5 = lee_filter(g, noise_var, 11);

% Generate and apply Gaussian filter
[M N] = size(g);
H_gauss = double(fspecial('gaussian', [M N], 120));
H = H_gauss./max(H_gauss,[],'all');
g_gauss = real(ifft2(H_gauss.*fft2(g)));

% Plot images
% Question 6
figure(1);
subplot(1,2,1);
imshow(f);
title('Original Image', 'FontSize', fontSize);
subplot(1,2,2);
imshow(g);
title('Degraded Image', 'FontSize', fontSize);

% Question 7
figure(2);
subplot(1,3,1);
imshow(g);
title('Degraded Image', 'FontSize', fontSize);
subplot(1,3,2);
imshow(f_hat_1);
title('Lee Filter', 'FontSize', fontSize);
subplot(1,3,3);
imshow(g_gauss, []);
title('Gaussian Low Pass Filter', 'FontSize', fontSize);

% Question 8
figure(3);
subplot(1,3,1);
imshow(f_hat_1);
title('1 x Noise Variance', 'FontSize', fontSize);
subplot(1,3,2);
imshow(f_hat_2);
title('0.5 x Noise Variance', 'FontSize', fontSize);
subplot(1,3,3);
imshow(f_hat_3);
title('3 x Noise Variance', 'FontSize', fontSize);

% Question 9
figure(4);
```

```
subplot(1,3,1);
imshow(f_hat_1);
title('5x5 Window Size', 'FontSize', fontSize);
subplot(1,3,2);
imshow(f_hat_4);
title('3x3 Window Size', 'FontSize', fontSize);
subplot(1,3,3);
imshow(f_hat_5);
title('11x11 Window Size', 'FontSize', fontSize);
```

File: lee_filter.m

```
function [f_hat] = lee_filter(g, noise_var, window)
    local_mean = colfilt(g, [window,window], 'sliding', @mean);
    local_var = colfilt(g, [window,window], 'sliding', @var);
    K = (local_var - noise_var)./local_var;
    f_hat = (K.*g)+((1-K).*local_mean);
end
```

File: PSNR_norm.m

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