SYDE 575 Lab 2
Image Enhancement

Group 23: Laura Chambers, ID: [redacted]

1 Introduction

The purpose of this lab is to gain experience with image enhancement and restoration techniques including noise reduction and image sharpening in the spatial domain.

Noise reduction is used to remove contaminated pixels from an image by estimating the correct pixel values based on neighbouring pixels. This estimation can be done using filters such as an averaging, weighted average, or median filter.

Image sharpening is used to enhance the appearance of edges within an image. Edges are easily recognized by the human visual system. Sharpening edges can make an image look more appealing and make objects and details more discernable.

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

2 Discrete Convolution for Image Processing

Figure 2-1 Discrete convolution of Lena image with impulse functions

1. Convolving the image with h1 smoothed the image, making it less sharp. This is because h1 is a horizontal 1-D spatial smoothing filter, which blurs edges. The filter is an averaging mask, which weights neighbouring pixels equally at 1/6. All values in h1 are greater than zero and the DC gain is equal to 1. Convolving the image with h2 averages neighbouring pixels, which decreases the difference between adjacent pixels. The softens the sharpness of edges.

2. Convolving the image with h2 smoothed the image, making it less sharp. This is because h2 is a vertical 1-D spatial smoothing filter, which blurs edges in an image. The filter is an averaging mask, which weights neighbouring pixels equally at 1/6. All values in h2 are greater than zero and the DC gain is equal to 1. Convolving the image with h2 averages neighbouring pixels, which decreases the difference between adjacent pixels. The softens the sharpness of edges.

3. Convolving the image with h3 highlights only the edges of the image and leaves the rest of the image dark. This is because h3 is an edge detection filter. The filter is a first difference operator that includes negative weights and has a DC gain of zero. Convolving the image with h3 subtracts the

values of adjacent pixels. The resulting pixels appear dark if the difference in intensity was small, or bright if the difference in intensity was large, as is the case with edges.

4. In the context of image processing, convolution can either smooth an image or detect edges within an image.

3 Noise Generation

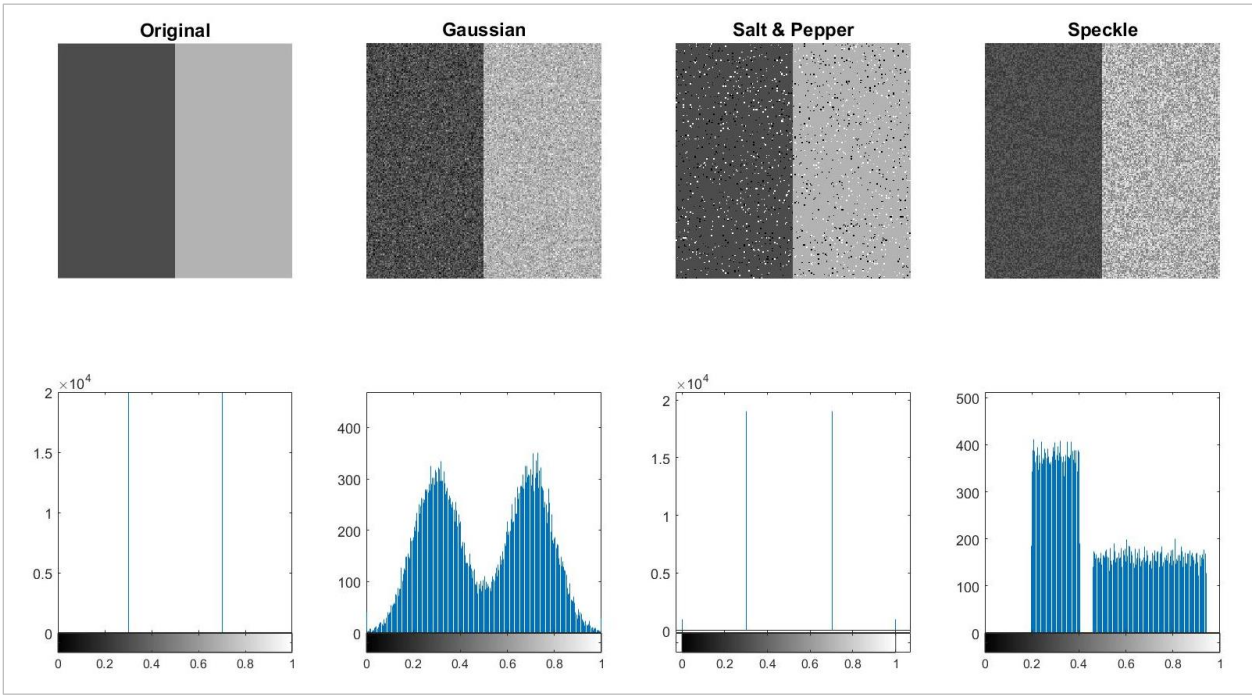

Figure 3-1 Noise generation test images and histograms

5. The test image consists of two rectangles. The intensities of the left and right rectangles are 0.3 and 0.7, respectively. The original histogram shows that the pixels are distributed equally between these two intensities.

   First, Gaussian noise was added to the test image. A zero-mean Gaussian distribution is shaped like a bell curve. The histogram shows two bell curves with means at the original intensities of 0.3 and 0.7.

   Next, salt and pepper noise was added to the test image. Salt and pepper noise is composed of scattered black and white pixels. This can be observed in the histogram, in which a portion of the original 0.3 and 0.7 intensity pixels have been redistributed to 0 and 1 intensities.

   Finally, speckle noise was added to the test image. Speckle appears as grainy noise, with some pixels being lighter and darker than in the original image. In the histogram, the original 0.3 and 0.7 intensities have become more widely distributed around each value. The lighter distribution shows more variability than the dark.

6. The noise is clearly visible in each contaminated image. The Gaussian and speckle noise look similar, but the Gaussian noise is more subtle. This is because the majority of pixels are distributed

closer to their original values of 0.3 and 0.7. In the speckle noise, the intensities are more evenly distributed, resulting in more contrast with each rectangle of the test image. The salt and pepper noise is distinct from the other two. The noisy pixels are more discernable, since they are either black (0) or white (1). The white pixels within the left, dark rectangle and the black pixels within the right, light rectangle are obvious.

7. Based on the histogram, the underlying distribution is a uniform distribution. The histogram appears to have two rectangular parts to the distribution, centred on the original values of 0.3 and 0.7. The rectangular shapes suggest a uniform distribution.

8. Speckle noise is multiplicative, meaning it depends on the local gray level of the pixels. The PDF of uniform noise is:

$$p(z) = \begin{cases} \frac{1}{b-a} & a \le z \le b \\ 0 & \text{otherwise} \end{cases}$$

This means that the probability for a value within the range of a and b is inversely proportional to that spread. The speckle histogram pattern reflects this. The lighter part of the distribution shows relatively more spread and a lower peak than the darker part.

4 Noise Reduction in the Spatial Domain

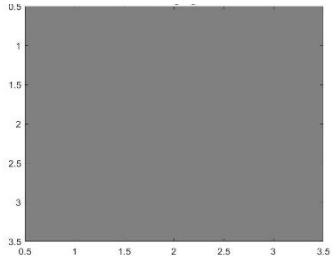

Figure 4-1 3x3 averaging filter

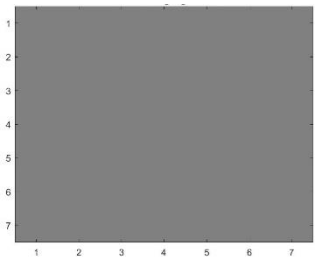

Figure 4-2 7x7 averaging filter

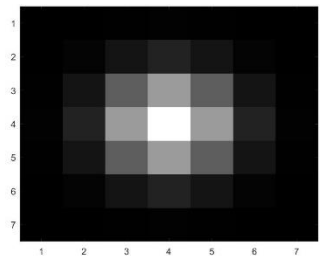

Figure 4-3 7x7 Gaussian filter

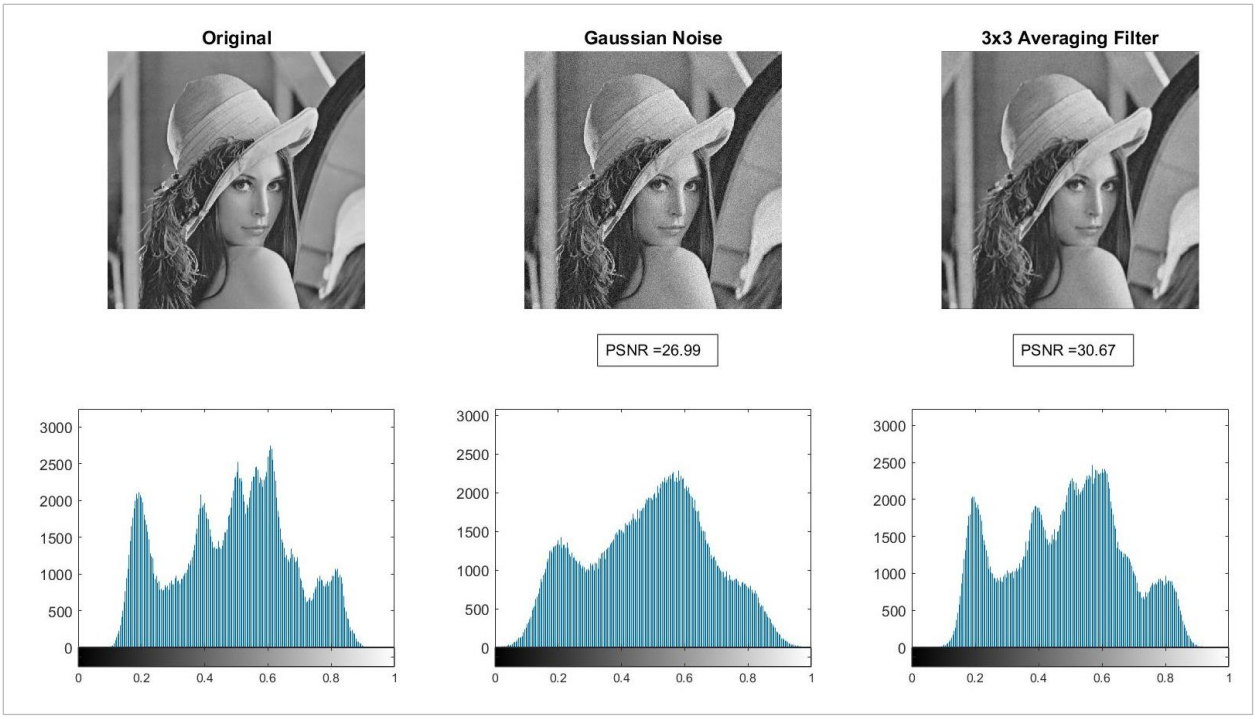

Figure 4-4 Gaussian noise reduction via 3x3 averaging filter

9. Visually, the de-noised image looks less grainy but somewhat blurrier than the noisy image. The averaging filter averaged-out the contaminated pixels based on neighbouring pixels in the noisy image, but also made edges less sharp in the process. PSNR did not decrease, but rather increased, indicating that the image is, in fact, less noisy.

10. The histogram of the de-noised image resembles the histogram of the original image. The Gaussian noise in the contaminated image smoothed out the peaks and valleys in the original histogram. The de-noised image re-gained the original histogram distribution by averaging the values of neighbouring pixels. This replaces the noisy pixels with values that more closely matched their original pixel values.

11. The benefit of the averaging filter is that it effectively removes noise. The drawback is that it over-smooths the image, making it looked blurred.

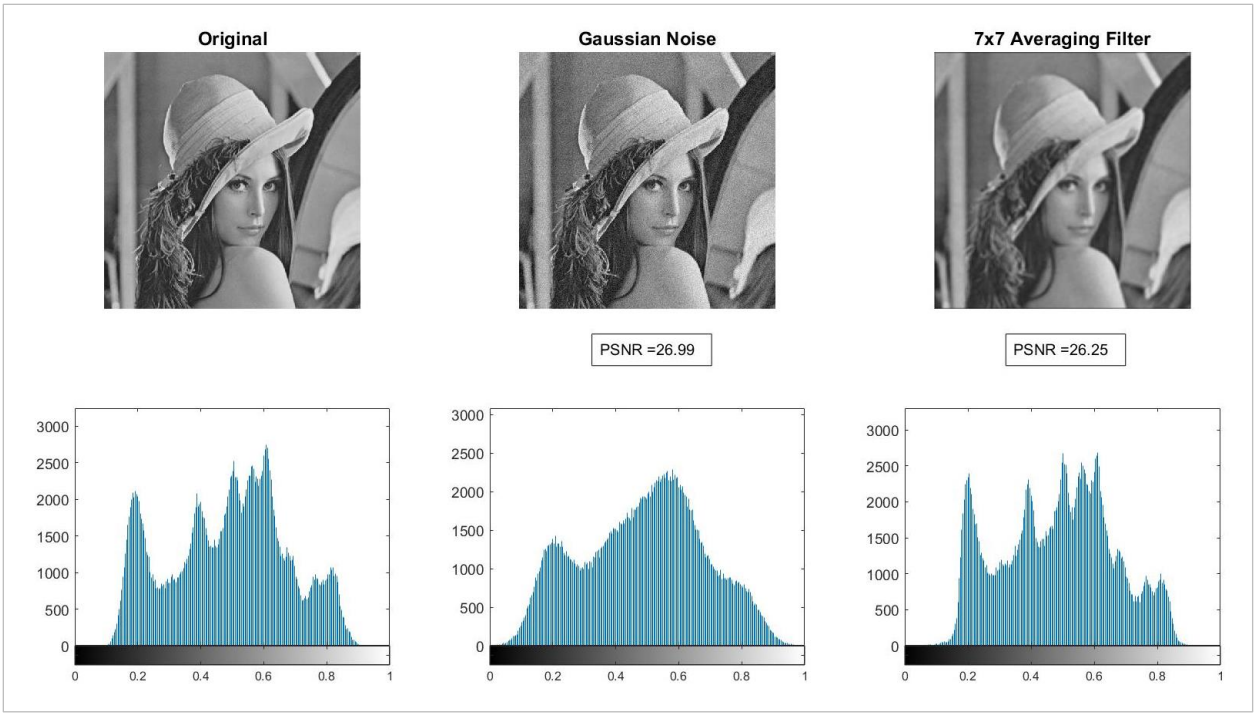

Figure 4-5 Gaussian noise reduction via 7x7 averaging filter

12. The image from the 7x7 kernel looks blurrier than the image from the 3x3 kernel. The 7x7 mask averages more neighbouring pixels to determine the filtered pixel values. This decreases the effect of the darkest and brightest pixels in the image since they move toward the average. For this reason, the PSNR decreased compared to both the 3x3 kernel and noisy images.

13. Both the 7x7 and 3x3 kernel histograms recovered much of the shape of the original histogram with some differences. The 7x7 kernel histogram shows reduced counts of the darkest and brightest pixel values. This can be observed by the skinnier tails at either end of the 7x7 kernel histogram compared to the fatter tails in the original histogram. However, the 7x7 kernel histogram more closely matches the original histogram in the mid-range of pixel values. For example, the 3x3 kernel histogram did not regain the shape of the original histogram as well as the 7x7 kernel histogram between 0.5 and 0.7.

14. The benefit of a larger window size is that it effectively filters noise from mid-range pixels values. The drawback is that it is less effective at filtering noise from the darkest and brightest pixels in an image, consequently making the image appear blurrier and reducing PSNR compared to a smaller window size.

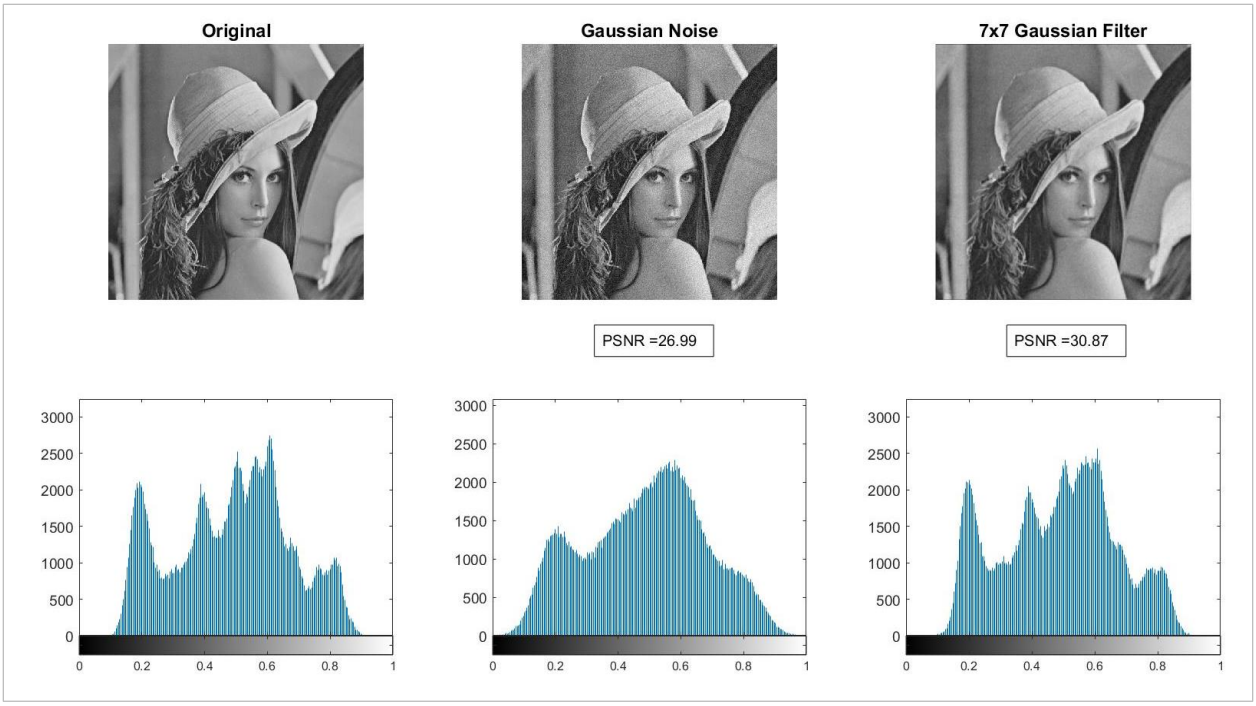

Figure 4-6 Gaussian noise reduction via 7x7 Gaussian filter

15. The 7x7 Gaussian kernel image appears less blurry than the 7x7 averaging kernel image and comparable to the 3x3 averaging kernel image. Similarly, its PSNR is higher than the 7x7 averaging kernel PSNR, and very close in value to the 3x3 averaging kernel image. This is because the 7x7 Gaussian kernel uses a weighted average. Pixels that are closer to the input pixel are assigned higher weights than those farther away. As a result, the weighted average combined with a larger window size appears to have a similar filtering affect as a smaller window averaging filter.

16. The 7x7 Gaussian kernel histogram appears similar to the 3x3 averaging kernel histogram. It better recovers the darkest and lightest pixel values from the original histogram compared to the 7x7 averaging kernel, and less so the mid-range pixel values.

17. The benefit of a Gaussian filter is that it removes noise while causing less blur than an averaging filter of the same window size. The drawback is that is still causes some blurring of edges.

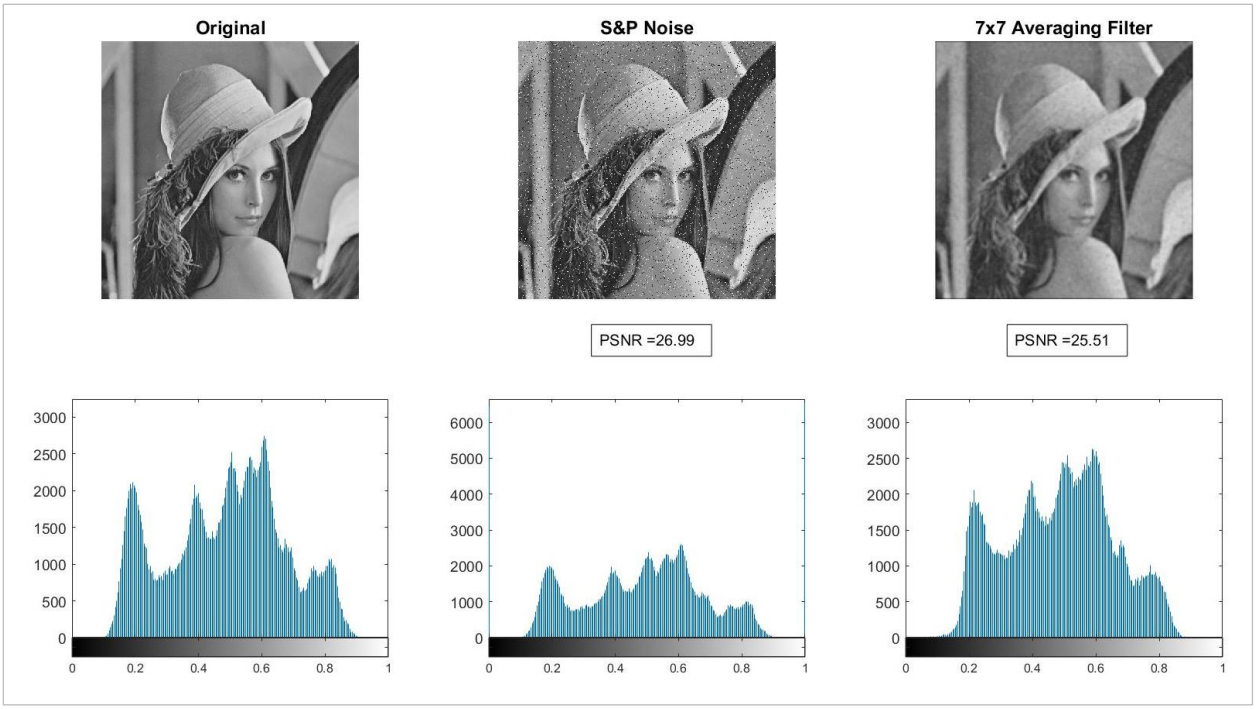

Figure 4-7 Salt and pepper noise reduction via 7x7 averaging filter

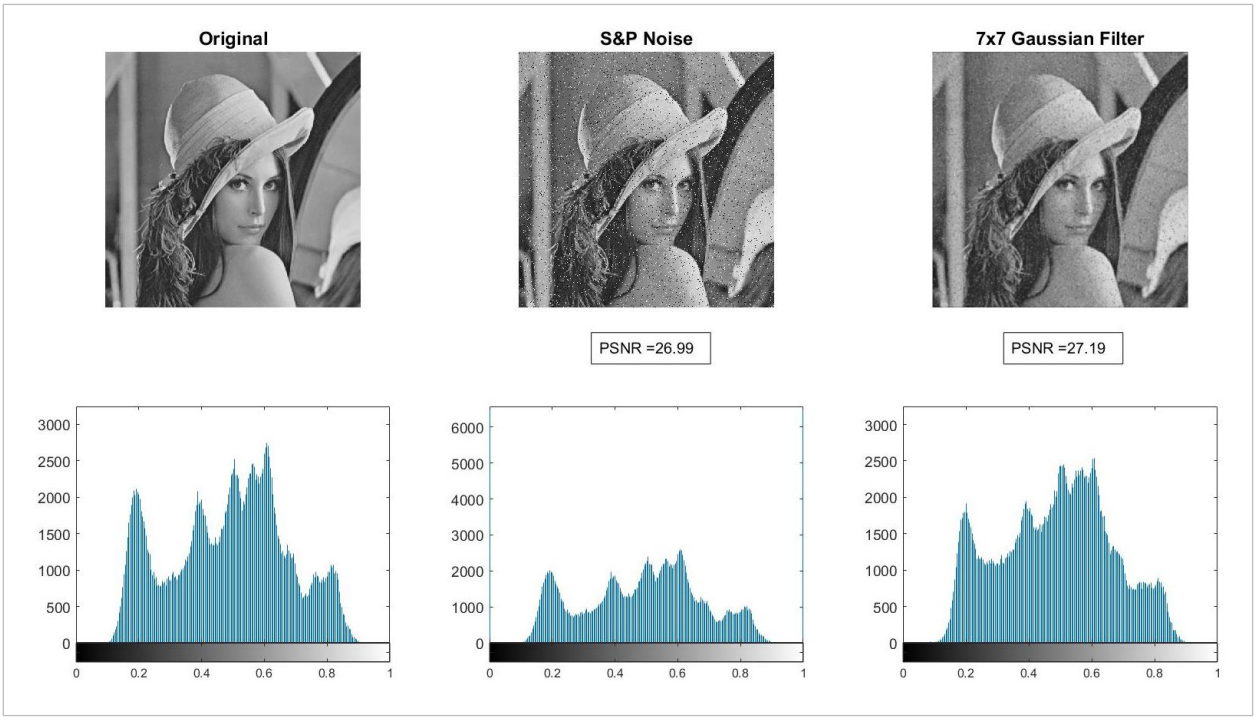

Figure 4-8 Salt and pepper noise reduction via 7x7 Gaussian filter

18. The averaging and Gaussian filtering methods do not perform very well on the salt and pepper image in terms of noise reduction. The averaging filter PSNR is slightly lower and the Gaussian filter PSNR is only slightly higher than the PSNR for the noisy image. This is because the salt and pepper noise adds black and white pixels to the image, which act as outliers in the averaging process.

19. The histogram for the noisy image shows salt and pepper pixels at either extreme of 0 and 1. The histograms for the noisy and denoised images show lower counts toward the darkest and brightest pixel values compared to the original histogram. This is due to the effect of averaging as well as salt and pepper pixels skewing the averages. For example, a black pixel in an area that is otherwise bright causes the averaged output pixel to skew toward a middle pixel value.

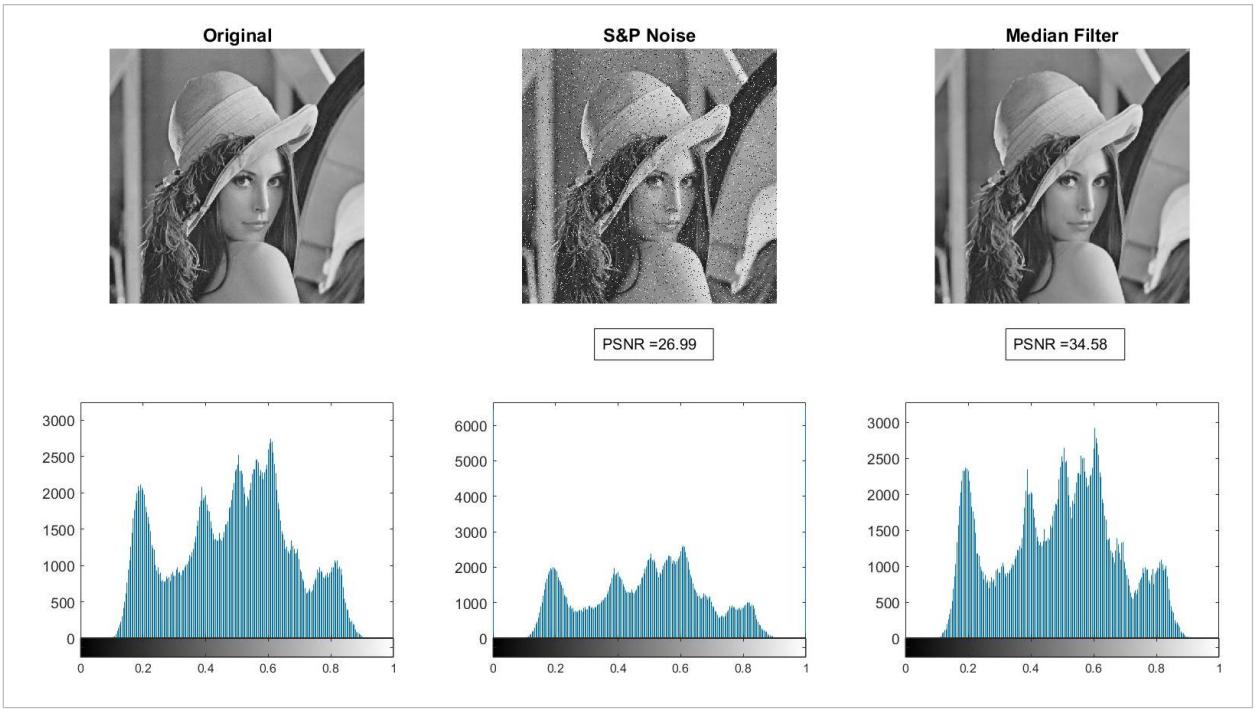

Figure 4-9 Salt and pepper noise reduction via median filter

20. The image produced by the median filter looks noticeably better than the imaged produced by the averaging and Gaussian filtering method. Noise is removed from the image while maintaining edges. The PSNR reflects this as well, with the median PSNR exceeding PSNR values for all other tested filters. The median filter sorts neighbouring pixels in order and takes the median value. This avoids the effect of outlier values, like salt and pepper pixels, from skewing the result.

5 Sharpening in the Spatial Domain

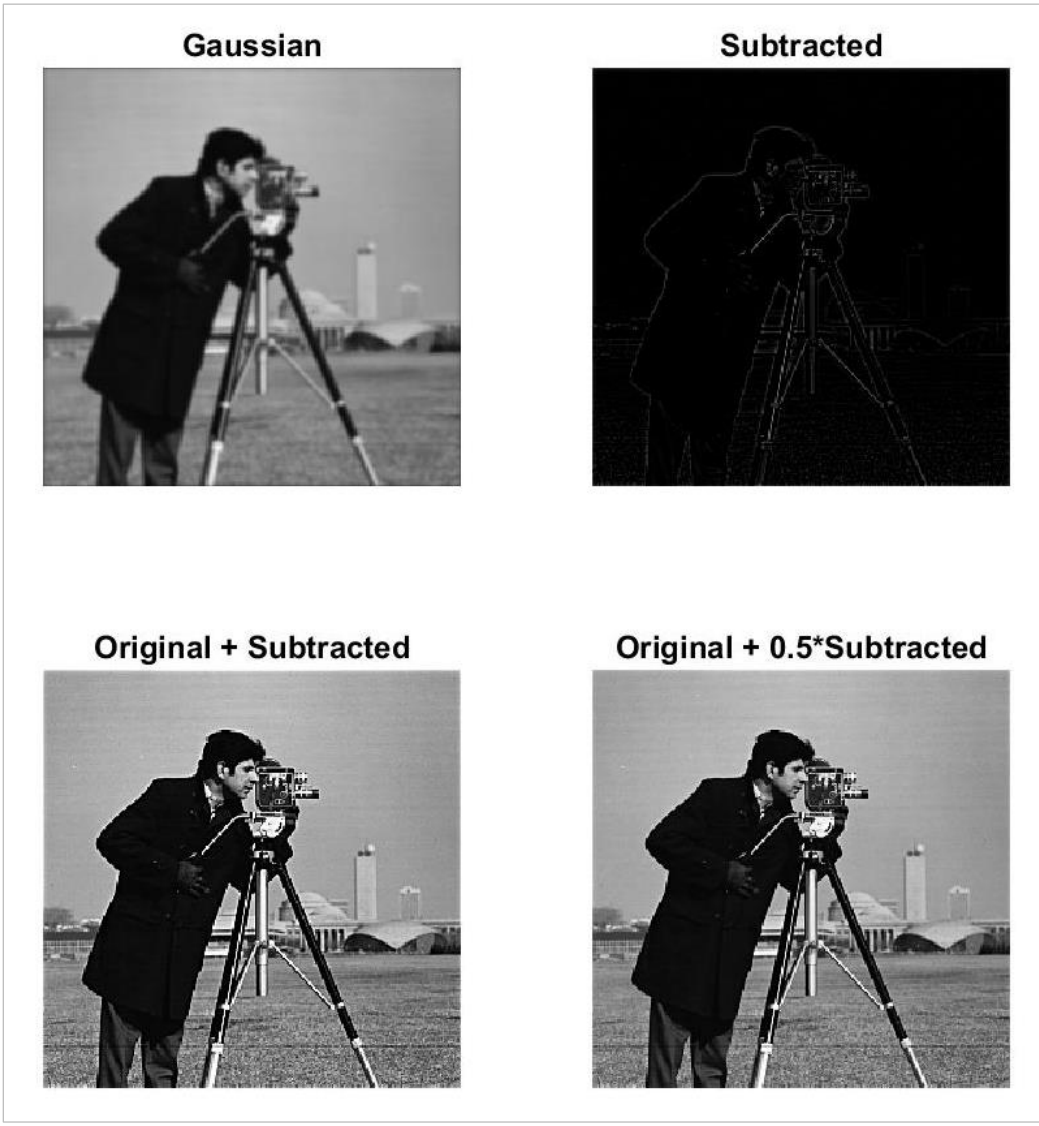

Figure 5-1 Results of sharpening techniques

21. The subtracted image looks dark with a light outline of the edges in the cameraman image. High spatial frequencies are preserved in the subtracted image. The Gaussian filter slightly blurs edges in the image, changing the pixel values near edges compared to the original. When the filtered image is subtracted from the original, the difference between the pixel values is greatest near the edges. This causes them to appear relatively bright in the subtracted image.

22. When the subtracted image is added to the original, the resulting image appears sharper. This is because the higher pixel values along the edges have been added to the edges in the original image, making them brighter. The brighter edges increase local contrast with neighbouring dark pixels, which our eyes perceive as sharper edges.

23. When the subtracted image is halved and then added to the original image, the resulting image appears slightly less sharp compared to when the full subtracted image is added. This is because the edge pixel values that are added to the original edges are lower. The resulting edges have higher contrast than the original image but are not as sharp as the addition of the full subtracted image.

24. Multiplying the subtracted image by a factor less than one decreases the sharpening effect of the subtracted image by decreasing the brightness and width of the edges in the subtracted image. Conversely, multiplying the subtracted by a factor greater than one increases the sharpening effect of the image by increasing the brightness and width of the edges in the subtracted image. This can be seen in Figure 5-2 where the subtracted image is multiplied by a factor of 10.

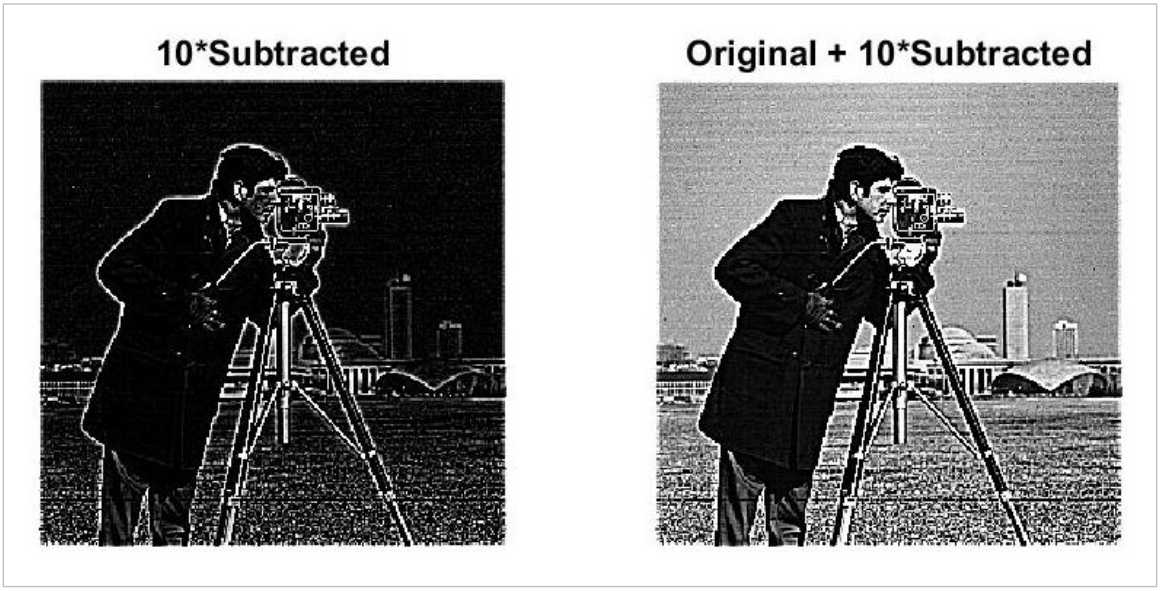

Figure 5-2 Multiplying the subtracted image by a factor of 10

6 Conclusions

This lab provided hands-on experience with fundamental image enhancement techniques, including noise reduction and image sharpening, and practice using image processing tools in MATLAB.

From this lab, it is evident the performance of noise reduction techniques depends on the estimation method and window size. Averaging filters reduce noise but introduce blur, while median filters effectively reduce noise without adding blur.

Image sharpening techniques function by detecting differences in neighbouring pixels. Edges are characterized by discontinuities in grey levels. A subtracted image can be added to an original image in order to sharpen edges. The subtracted image can be multiplied to change the strength of the sharpening effect.

Appendix – MATLAB Code

File: lab2_prt2_discrete_convolution.m

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

% Load images
lena = imread('lena1.tiff');

% Convert images to grayscale and normalize intensities from 0 to 1
lena_gray = convert2grayscale(lena);
lena_gray_norm = double(lena_gray)/255;

% Impulse functions
h1 = (1/6)*ones(1,6)
h2 = h1'
h3 = [-1 1]

% Convolve Lena and impulse functions
lena1 = conv2(lena_gray_norm, h1);
lena2 = conv2(lena_gray_norm, h2);
lena3 = conv2(lena_gray_norm, h3);

% Plot images
figure;
set(gcf, 'Position', get(0,'Screensize'));
subplot(2, 2, 1);
imshow(lena_gray_norm);
title('Original', 'FontSize', fontSize);
subplot(2, 2, 2);
imshow(lena1);
title('Impulse function h1', 'FontSize', fontSize);
subplot(2, 2, 3);
imshow(lena2);
title('Impulse function h2', 'FontSize', fontSize);
subplot(2, 2, 4);
imshow(lena3);
title('Impulse function h3', 'FontSize', fontSize);
```

File: lab2_prt3_noise_generation.m

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

% Generate test image
f = [0.3*ones(200,100) 0.7*ones(200,100)];

% Add noise to test image
f_gauss = imnoise(f,'gaussian',0,0.01);
f_salt_pepper = imnoise(f,'salt & pepper',0.05);
f_speckle = imnoise(f,'speckle',0.04);

% Plot images
figure;
set(gcf, 'Position', get(0,'Screensize'));
subplot(2, 4, 1);
imshow(f);
```

```
title('Original', 'FontSize', fontSize);
subplot(2, 4, 2);
imshow(f_gauss);
title('Gaussian', 'FontSize', fontSize);
subplot(2, 4, 3);
imshow(f_salt_pepper);
title('Salt & Pepper', 'FontSize', fontSize);
subplot(2, 4, 4);
imshow(f_speckle);
title('Speckle', 'FontSize', fontSize);

% Plot histograms
subplot(2, 4, 5);
imhist(f);
subplot(2, 4, 6);
imhist(f_gauss);
subplot(2, 4, 7);
imhist(f_salt_pepper);
subplot(2, 4, 8);
imhist(f_speckle);
```

File: lab2_prt4_noise_reduction_spatial_domain.m

```
clc;
close all;
clear;

% Load images, convert to grayscale, and normalize intensities
lena = imread('lena1.tiff');
lena_gray = convert2grayscale(lena);
lena_gray_norm = double(lena_gray)/255;

% Add noise to lena_gray_norm
lena_gauss = imnoise(lena_gray_norm,'gaussian',0,0.002);
lena_salt_pepper = imnoise(lena_gray_norm,'salt & pepper',0.05);

% Create filters and plot
H_avg3 = fspecial('average', [3 3]);
H_avg7 = fspecial('average', [7 7]);
H_gauss = fspecial('gaussian', [7 7], 1);
plotFilter(H_avg3, '3x3 Averaging Filter');
plotFilter(H_avg7, '7x7 Averaging Filter');
plotFilter(H_gauss, '7x7 Gaussian Filter');

% Filter images
lena_filter1 = imfilter(lena_gauss, H_avg3);
lena_filter2 = imfilter(lena_gauss, H_avg7);
lena_filter3 = imfilter(lena_gauss, H_gauss);
lena_filter4 = imfilter(lena_salt_pepper, H_gauss);
lena_filter5 = imfilter(lena_salt_pepper, H_avg7);
lena_filter6 = medfilt2(lena_salt_pepper);

% Calculate PSNR
psnr_gauss = PSNR_norm(lena_gray_norm, lena_gauss)
psnr_salt_pepper = PSNR_norm(lena_gray_norm, lena_salt_pepper)
psnr_filter1 = PSNR_norm(lena_gray_norm, lena_filter1)
psnr_filter2 = PSNR_norm(lena_gray_norm, lena_filter2)
psnr_filter3 = PSNR_norm(lena_gray_norm, lena_filter3)
psnr_filter4 = PSNR_norm(lena_gray_norm, lena_filter4)
psnr_filter5 = PSNR_norm(lena_gray_norm, lena_filter5)
psnr_filter6 = PSNR_norm(lena_gray_norm, lena_filter6)
```

```
% Plot images
plot3Images(lena_gray_norm, lena_gauss, lena_filter1, 'Gaussian Noise', ...
    '3x3 Averaging Filter', psnr_gauss, psnr_filter1);
plot3Images(lena_gray_norm, lena_gauss, lena_filter2, 'Gaussian Noise', ...
    '7x7 Averaging Filter', psnr_gauss, psnr_filter2);
plot3Images(lena_gray_norm, lena_gauss, lena_filter3, 'Gaussian Noise', ...
    '7x7 Gaussian Filter', psnr_gauss, psnr_filter3);
plot3Images(lena_gray_norm, lena_salt_pepper, lena_filter4, 'S&P Noise', ...
    '7x7 Gaussian Filter', psnr_gauss, psnr_filter4);
plot3Images(lena_gray_norm, lena_salt_pepper, lena_filter5, 'S&P Noise', ...
    '7x7 Averaging Filter', psnr_gauss, psnr_filter5);
plot3Images(lena_gray_norm, lena_salt_pepper, lena_filter6, 'S&P Noise', ...
    'Median Filter', psnr_gauss, psnr_filter6);
```

File: lab2_prt5_sharpening_spatial_domain.m

```
clc;
close all;
clear;

% Load images, convert to grayscale, and normalize intensities
cameraman = imread('cameraman1.tif');
cameraman_gray = convert2grayscale(cameraman);
cameraman_gray_norm = double(cameraman_gray)/255;

% Apply the Gaussian filter
H_gauss = fspecial('gaussian', [7 7], 1);
camerman_gauss = imfilter(cameraman_gray_norm, H_gauss);

% Manipulate the original and subtracted image
camerman_subtract = cameraman_gray_norm - camerman_gauss;
cameraman_add = cameraman_gray_norm + camerman_subtract;
cameraman_mult_add = cameraman_gray_norm + (camerman_subtract.*0.5);

% Plot images
fontSize = 12;
figure;
set(gcf, 'Position', get(0,'Screensize'));
subplot(2, 2, 1);
imshow(camerman_gauss);
title('Gaussian', 'FontSize', fontSize);
subplot(2, 2, 2);
imshow(camerman_subtract);
title('Subtracted', 'FontSize', fontSize);
subplot(2, 2, 3);
imshow(cameraman_add);
title('Original + Subtracted', 'FontSize', fontSize);
subplot(2, 2, 4);
imshow(cameraman_mult_add);
title('Original + 0.5*Subtracted', 'FontSize', fontSize);
```

File: PSNR_norm.m

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

File: plot3Images.m

```
function [] = plot3Images(f,g,h, title1, title2, psnr1, psnr2)
```

```
    fontSize = 12;
    figure;
    set(gcf, 'Position', get(0,'Screensize'));
    subplot(2, 3, 1);
    imshow(f);
    title('Original', 'FontSize', fontSize);
    subplot(2, 3, 4);
    imhist(f);
    subplot(2, 3, 2);
    imshow(g);
    title(title1, 'FontSize', fontSize);
    subplot(2, 3, 5);
    imhist(g);
    subplot(2, 3, 3);
    imshow(h);
    title(title2, 'FontSize', fontSize);
    subplot(2, 3, 6);
    imhist(h);
    txt = strcat('PSNR = ', sprintf('%.2f',psnr1));
    annotation('textbox', [0.48, 0.45, 0.1, 0.1], 'string', txt);
    txt = strcat('PSNR = ', sprintf('%.2f',psnr2));
    annotation('textbox', [0.76, 0.45, 0.1, 0.1], 'string', txt);
end
```

File: plotFilter.m

```
function [] = plotFilter(filter, type)
    fontSize = 12;
    figure;
    colormap(gray);
    imagesc(filter);
    title(type, 'FontSize', fontSize);
end
```