University of Waterloo

Faculty of Engineering

SYDE575 Lab 2: Image Enhancement

Prepared By
Sarthak Tamboli | 20665466
Sheen Thusoo | 20728766

Date: 1 October, 2021
Prepared for: Professor David A. Clausi

1. Introduction

The purpose of this lab was to study noise reduction and get hands-on experience with image enhancement and restoration concepts in the spatial domain. Specifically, it focused on noise reduction and image sharpening. First, the convolution in image processing was investigated by convolving three different impulse functions with the Lena image. Next, we contaminated images with three different noise models to test the effectiveness of image processing algorithms by studying their performance under different noise levels. Further, we studied different noise reduction techniques based on spatial filtering and evaluated their performance. Lastly, we investigated noise sharpening techniques in the spatial domain as well as the effect of sharpening filter parameters on image quality. All of the code for this lab was written in MATLAB and can be found in Appendix A.

2. Discrete Convolution for Image Processing

In Section 2, convolution in image processing was studied. The Lena image was loaded, converted to grayscale, and normalized. Then three impulse functions, *h1*, *h2*, and *h3*, were constructed as shown in Equation (1), (2) and (3) below.

$$h1 = \frac{1}{6} * ones(1, 6); \quad (1)$$

$$h2 = h1'; \quad (2)$$

$$h3 = [-\ 1\ 1]; \quad (3)$$

Then, using the *conv2* function, the Lena image was convolved with each of these impulse functions. The original grayscale image can be seen in Figure 1. The images that are results of the convolutions from impulse functions *h1*, *h2*, and *h3* are seen in Figure 2, 3, and 4, respectively. The code for this section can be seen in Appendix A, Figure A.1.

Figure 1: Original Grayscale Lena Image Normalized

Figure 2: Lena Convolved with *h1*

Figure 3: Lena Convolved with *h2*

Figure 4: Lena Convolved with *h3*

1. Convolving the image with *h1* results in a more smooth/blurred image. The impulse function is an averaging filter that averages pixel intensities along the horizontal axis. Thus, the horizontal lines in the image appear to be more smoothed. Essentially, a low pass filter was applied in the horizontal direction.

2. When the image was convolved with the *h2* function it was smoothed/blurred along the lines in the vertical axis spatially. Essentially a low pass filter was applied in the vertical direction. The impulse function is the transpose of the *h1* function, it has a column array of ⅙, thus it makes sense that the lines in the vertical axis of the image are blurred as the filter averages the intensity of the image vertically.

3. When the image was convolved with *h3*, the image became mostly black with the edges of the image (outline of the hat, nose, eyes etc.) being white. Looking at the impulse function, it performs edge detection. Pixel intensities change at an edge and a convolution operation is performed, however when there is no edge the pixel intensities remain the same and the convolution cancels out. Thus, the filter responds strongly in areas where pixel intensity changes i.e. an edge. This filter is horizontal so it will respond stronger to horizontal edges.

4. Convolution can perform the role of image enhancement through removing noise and edge detection. The images convolved in this section show that depending on the impulse function, the images can be smoothed or averaged to allow for higher image quality by reducing noise. It can also be seen that these impulse functions can perform edge detections. Thus, it can be used to extract spatial information of the image. When blurring, the convolution function highlights the low frequency components of the image whereas when detecting edges, the convolution function highlights the high frequency components.

3. Noise Generation

In Section 3, the performance of image processing algorithms under different noise levels was evaluated. The following noise models were applied to a synthetic toy image: an additive zero-mean Gaussian with variance of 0.01, Salt and Pepper with noise density of 0.05, and multiplicative speckle noise with variance of 0.04. The images generated from these models and their associated histograms are shown in the figures below. The code for this section can be seen in Appendix A, Figure A.2.

Figure 5: Original Toy Image

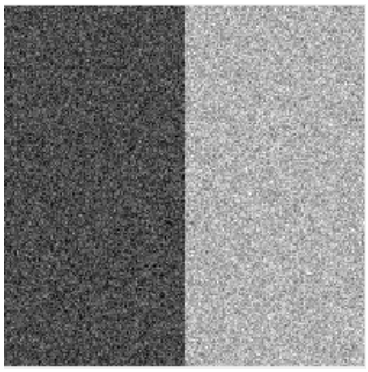

Figure 6: Gaussian Model Toy Image

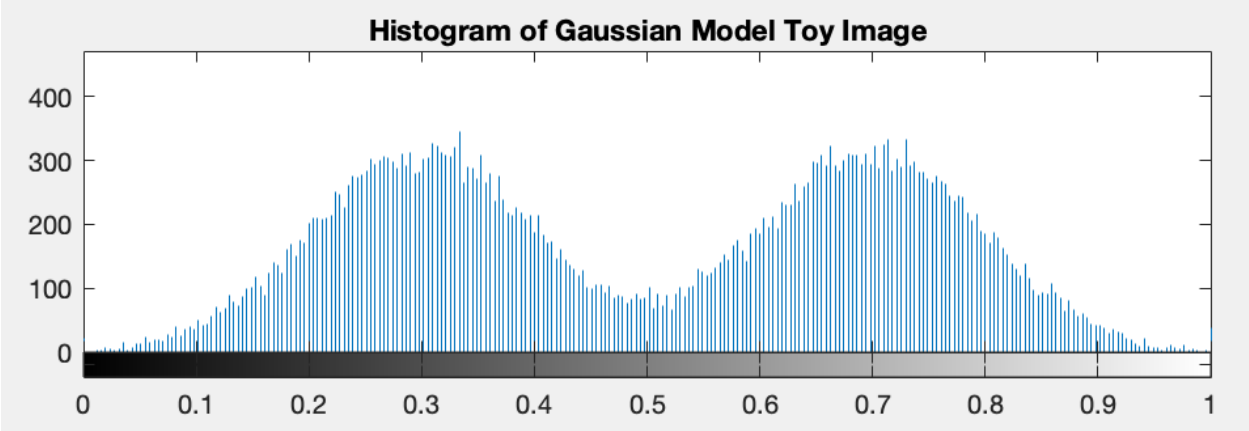

Figure 7: Histogram of Gaussian Model Toy Image

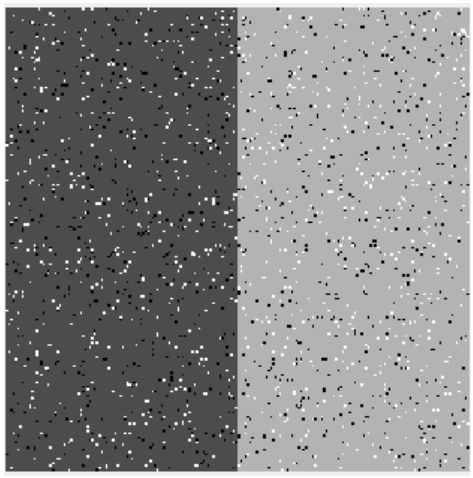

Figure 8: Salt & Pepper Model Toy Image

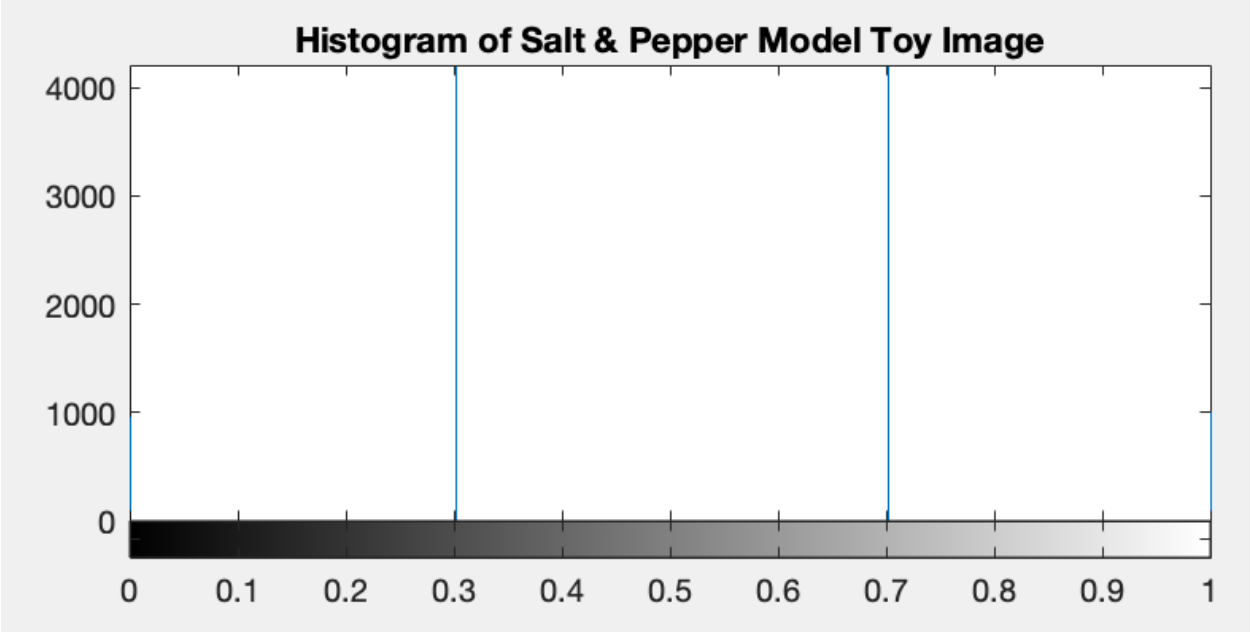

Figure 9: Histogram of Salt & Pepper Model Toy Image

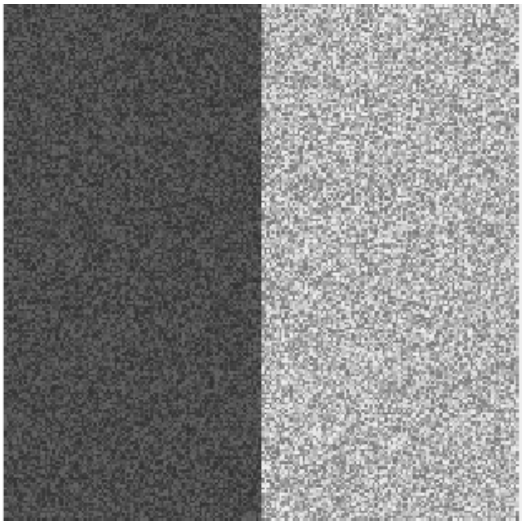

Figure 10: Speckle Model Toy Image

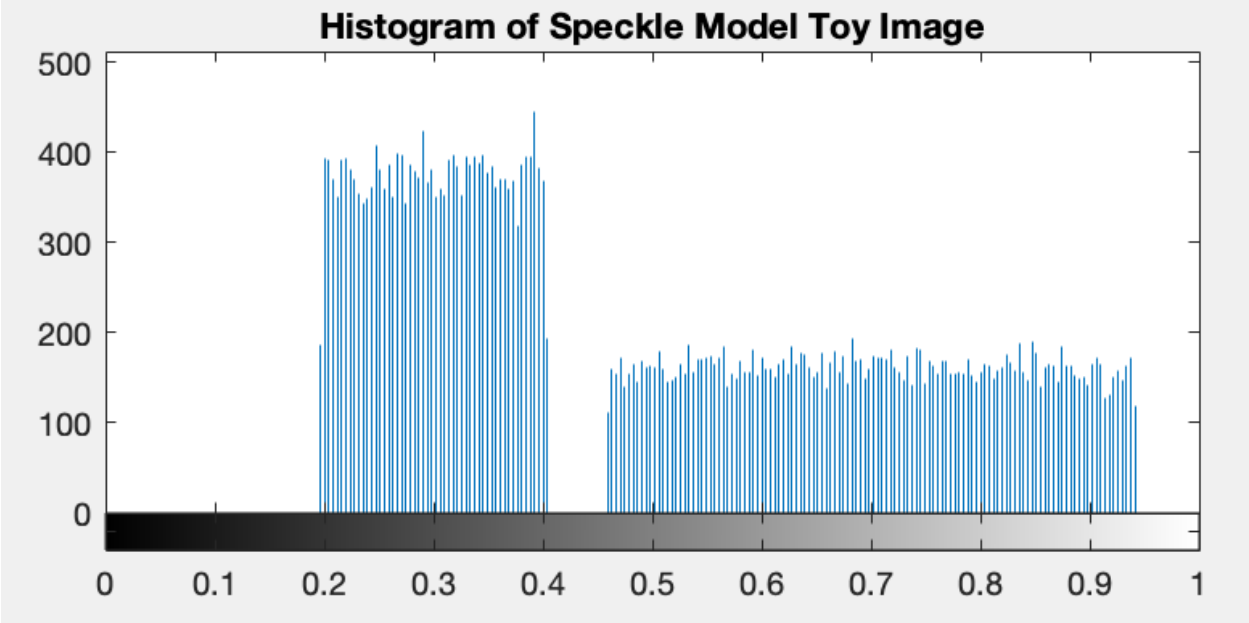

Figure 11: Histogram of Speckle Model Toy Image

5. The histogram created by the Gaussian model has two Gaussian distributions centered around the two pixel intensities in the original image. It appears this way because the model performs additive noise, thus each pixel in the original image has a value added to it. This value is taken from the Gaussian distribution which has a mean of 0 and a variance of 0.01.

   The histogram created by the Salt & Pepper noise model shows that there are only four levels of intensity in the image, that being 0, 0.3, 0.7, and 1. The model adds black (normalized value of 1) and white (normalized value of 0) pixels to the image and reflects the histogram which only has four levels of intensity, that being the two original intensities, 0.3 and 0.7, and the black and white pixel intensities, 0 and 1.

   Lastly, the histogram created by the speckle model has two uniform distributions centered around the two pixel intensities in the original image. This model adds multiplicative noise (with a variance of 0.04) with a uniform distribution. This is reflected in the histogram, as the lower and higher intensities are uniformly distributed (two rectangle shapes). Due to the multiplicative nature of this model, there is more variance around the original higher intensity pixel. This is because these higher values have a higher absolute variance.

6. There are visual differences between the noise contaminated images. The Gaussian noise image looks very noisy; similar to the noise or static that is sometimes seen on a TV. It has many pixels of varying intensities. This is because the additive Gaussian function is unbounded and so its resulting pixels can vary greatly from their original intensities.

   The image with the Salt and Pepper noise model looks as if the noise is sprinkled on top of the image. Both halves of the image retain their original pixel intensities, however there are some added pixels of intensity 0 and 1. There seems as if there are a smaller number of pixel

intensities. This is because the Speckle noise model is bounded and multiplicative. Thus, the original pixel intensities have only slightly been changed.

The image with the Speckle model looks like the Gaussian model but with fewer intensities. The left half is mostly pixels of dark to medium grey intensities, the right half is mostly pixels of lighter grey to white intensities. This is because the model is bounded and multiplicative.

7. In this speckle noise case, the underlying distribution is a bounded uniform distribution with a variance of 0.04. This can be seen in the histogram as there are two distinct uniform distributions rather than a smooth curve like the Gaussian model. In the Speckle model, a value is taken as a sample from the uniform distribution and multiplied by the original pixel intensity. The histogram shows that the intensities that are near or the same as the intensities in the original image are distributed similarly. The noisy pixels are not far from the normal intensities unlike gaussian noise.

8. The noise in the Speckle noise case is multiplicative, thus, the local grey levels will have different levels of spread. For instance, as seen in Figure 11, the lower grey level intensities (from about 0.2 to 0.4) have a lower spread of values than the higher grey level intensities (from about 0.45 to 0.95) which have a wider range or variance of values. The noise variance will be proportional to the intensities of the pixel so a higher pixel intensity will have a wider spread.

4. Noise Reduction in the Spatial Domain

In Section 4, different noise reduction techniques, based on spatial filtering with differing filter parameters, were explored. The Lena image contaminated with zero-mean Gaussian noise with a variance of 0.002 was used as the input. Initially, two different types of averaging filter kernels were tested, a 3×3 and 7×7. Next a 7×7 Gaussian filter kernel with a standard deviation of 1 was used. A new noisy image was then created by applying the salt and pepper noise model, with 0.05 density, and the 7×7 averaging and Gaussian filters were applied separately. Last, the median filter was applied to the Salt and Pepper noisy image. All output images were compared with their respective noisy images and their histograms were analyzed. Table 1 contains all of the PSNR values computed for this section. The code for this section can be seen in Appendix A Figures A.3.1, A.3.2, A.3.3, A.3.4, and A.3.5.

Figure 12: Grayscale and Normalized Lena Image

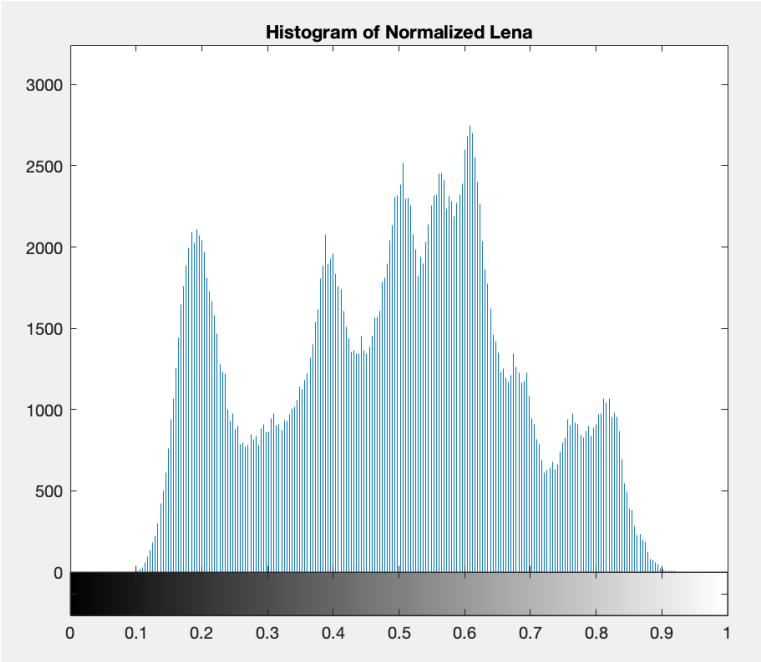

Figure 13: Histogram of Grayscale and Normalized Lena Image

Figure 14: Noisy Lena Image with a zero-mean Gaussian and 0.002 variance

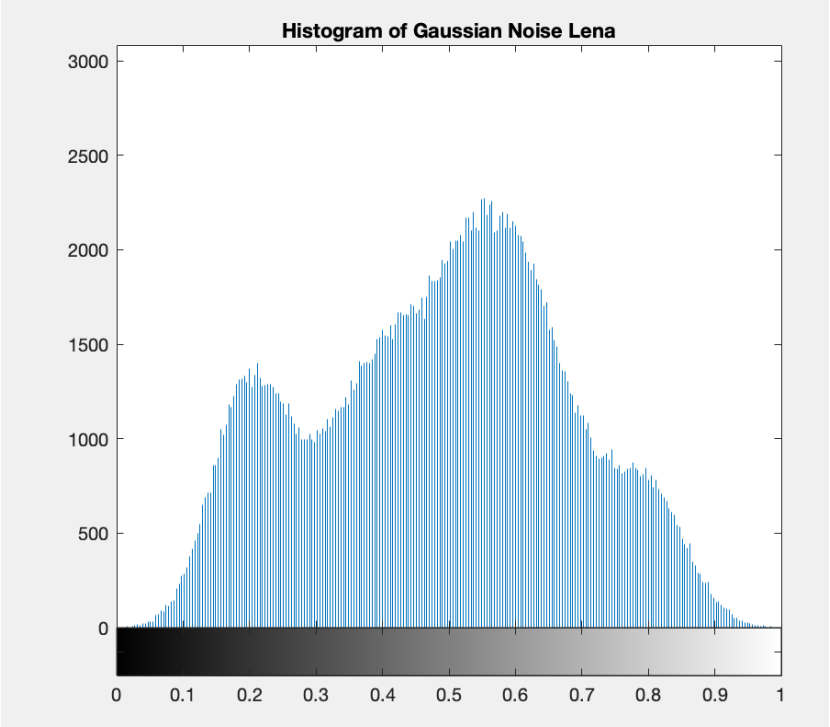

Figure 15: Histogram of Gaussian Noise Lena

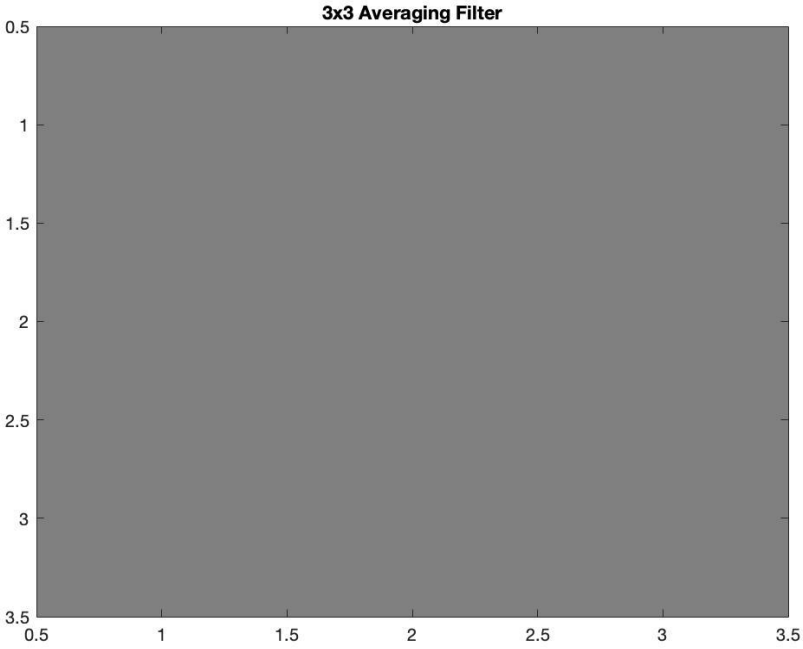

Figure 16: 3 × 3 Averaging Filter Kernel

Figure 17: Denoised Gaussian Lena with 3 × 3 Averaging Filter Kernel

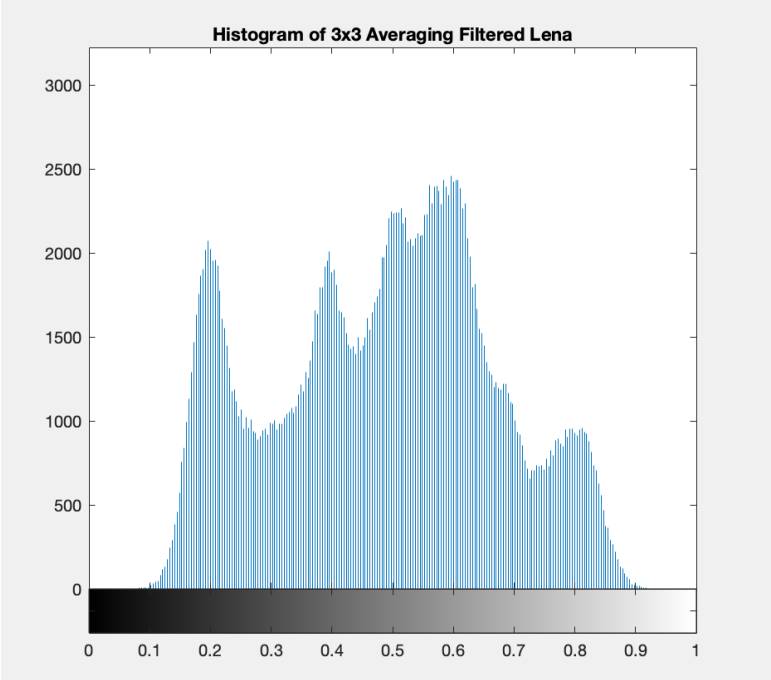

Figure 18: Histogram of Denoised Gaussian Lena with 3×3 Averaging Filter Kernel

Figure 19: Denoised Gaussian Lena with 7×7 Averaging Filter Kernel

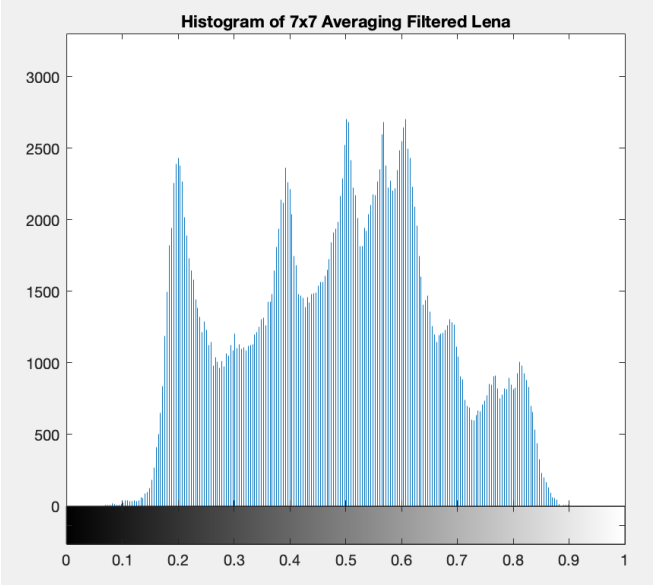

Figure 20: Histogram of Denoised Gaussian Lena with 7 × 7 Averaging Filter Kernel

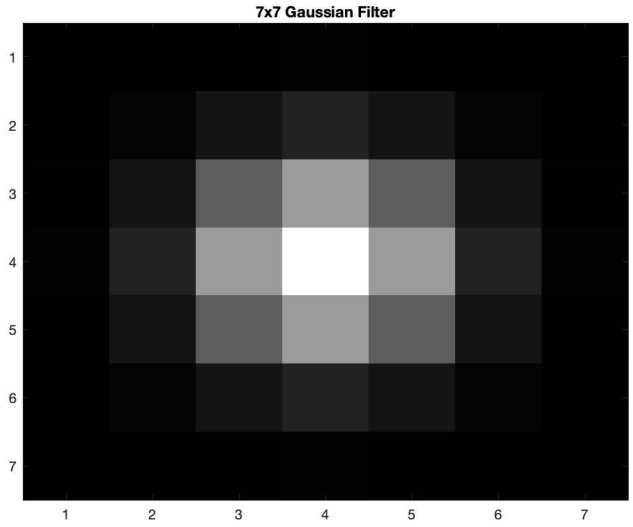

Figure 21: 7 × 7 Gaussian Filter Kernel with Standard Deviation of 1

Figure 22: Denoised Gaussian Lena with 7 × 7 Gaussian Filter Kernel

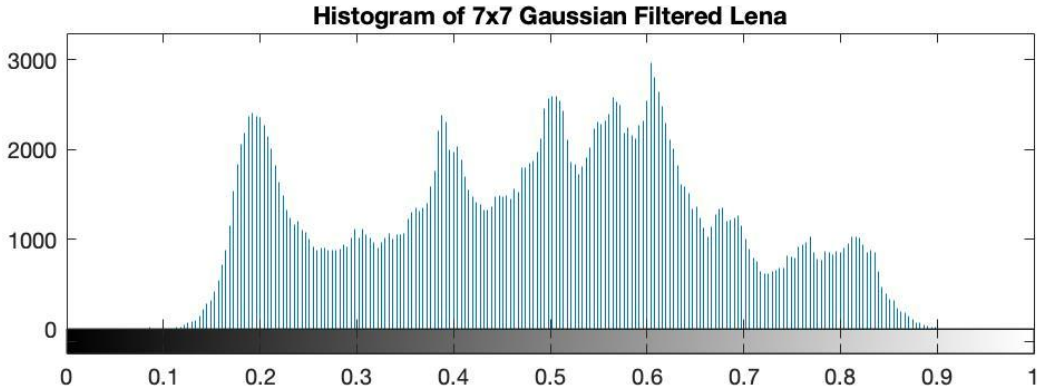

Figure 23: Histogram of Denoised Gaussian Lena with 7 × 7 Gaussian Filter Kernel

Figure 24: Noisy Lena with 0.05 density Salt and Pepper noise added

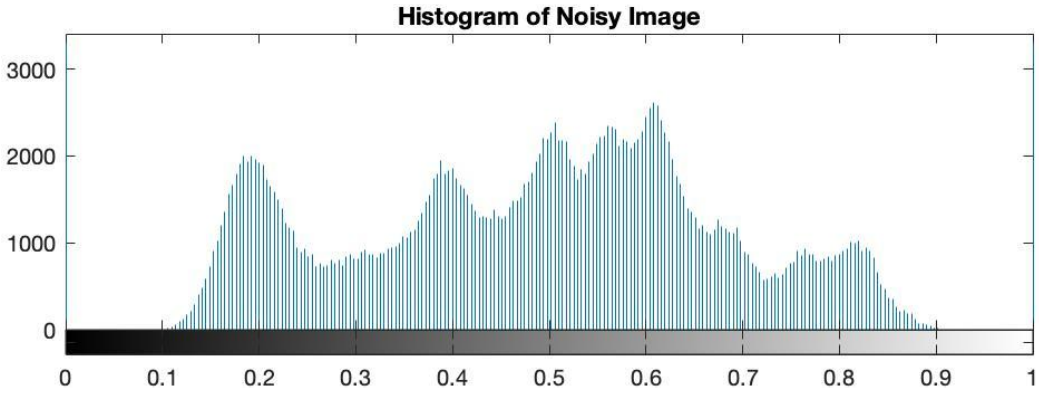

Figure 25: Histogram of Denoised Salt and Pepper Lena

Figure 26: Denoised Salt and Pepper Lena with 7 × 7 Averaging Filter Kernel

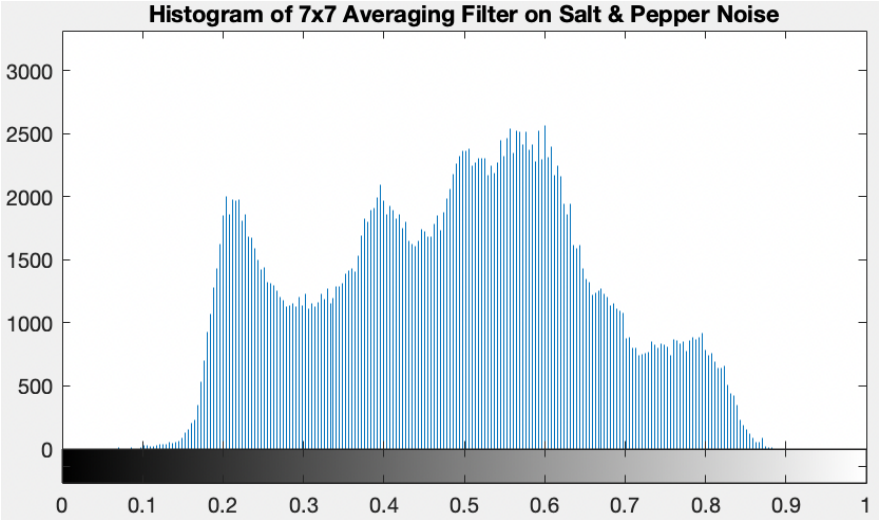

Figure 27: Histogram of Denoised Salt and Pepper Lena with 7 × 7 Averaging Filter Kernel

Figure 28: Denoised Salt and Pepper Lena with 7 × 7 Gaussian Filter Kernel

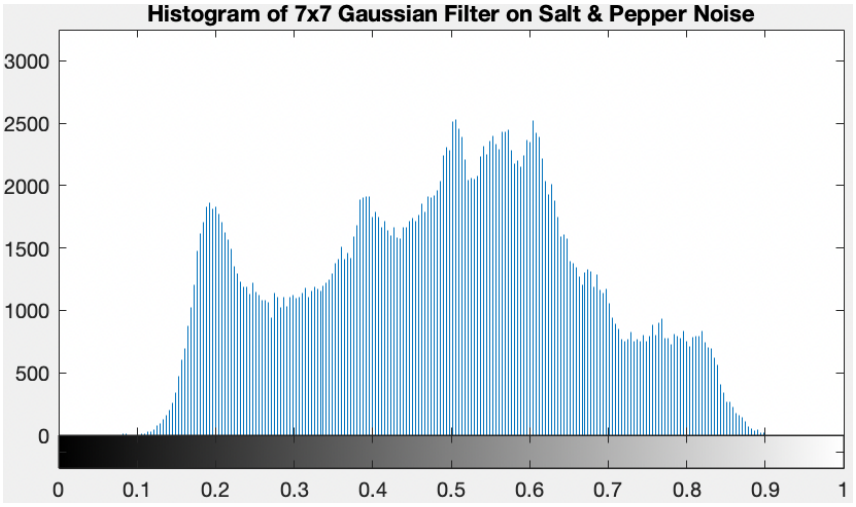

Figure 29: Histogram of Denoised Salt and Pepper Lena with 7 × 7 Gaussian Filter Kernel

Figure 30: Denoised Salt and Pepper Lena with Median Filter

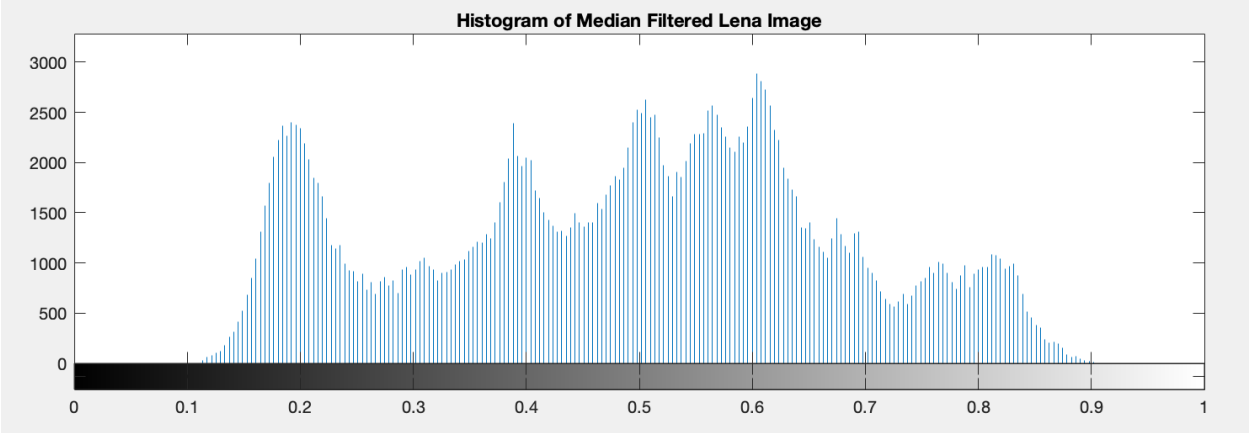

Figure 31: Histogram of Denoised Salt & Pepper with Median Filter (Lena)

Table 1: PSNR Values Computed on Noised/Denoised Image and Original Image.

Denoised/Noised	**Filter Applied**	**PSNR Value**
Noised	Gaussian (0 mean, 0.002 variance)	26.9982
Noised	Salt & Pepper (density 0.05)	18.4386
Denoised	3 × 3 Averaging Kernel	30.6500
Denoised	7 × 7 Averaging Kernel	26.2490
Denoised	7 × 7 Gaussian Kernel (standard deviation of 1)	31.7881
Denoised	7 × 7 Averaging Kernel on Salt & Pepper Noised Image	25.5094
Denoised	7 × 7 Gaussian Kernel on Salt & Pepper Noised Image	27.0839
Denoised	Median on Salt & Pepper Noised Image	35.4426

9. Comparing Figure 14 (noisy) and Figure 17 (denoised), the denoised image is much smoother/blurrier. The denoised image has a slight blurring effect on it as some of the detail from the noisy image was lost in the filtration. The averaging did not have much of an effect on the contrast or varying intensities levels from the noisy image. It worked well to reduce the noise as the noisy image is grain-like, and post-image processing, the resulting image is smoother.

   The PSNR value increased as compared to the noisy image. The PSNR value of the noisy image was 26.9982 whereas the denoised image had a value of 30.6500. There is roughly a 3 unit increase in PSNR value. The increase suggests that the denoised image is of better quality and that is visually evident upon the inspection of both noisy and denoised images as explored initially. The 3×3 averaging filter kernel seems to have slightly reduced the MSE resulting in a better PSNR. The histogram of the denoised image also seems more similar to that of the original image.

10. The histogram, Figure 13, of the noise-free image loosely follows a normal distribution with peaks at both low and high ends of intensities. The low intensity peak is higher than the high intensity peak. This indicates there is a somewhat balanced distribution of all intensities across the pixels with a focus on the low and mid gray levels.

    The noisy image histogram, Figure 15, has transformed to a shape closer to a normal (Gaussian) distribution with an added small peak at a lower intensity. The distribution of the pixel intensities are more evenly distributed than before. This happened because the noise model that was introduced to the image is of a zero-mean Gaussian model, indicating the transformed intensities would be within the middle range of gray levels. The additive nature of this model works to change the original pixel intensity into a range of pixel intensities.

    The denoised image histogram, Figure 18, has almost nullified the effects of the Gaussian noise introduced and the shape of the histogram is similar to the noise-free one in Figure 13. The 3×3 Averaging Filter applied caused the local pixel intensity to be affected by the neighbouring pixel intensities which helped restore the values closer to the noise-free image. There are more distinct peaks and troughs of intensity values in the low range.

11. Based on the visual quality of the denoised image, the benefits of the averaging filter are that they work well to reduce noise. Comparing Figure 17 (denoised) with Figure 14 (noise), the small white pixels covering the noisy image have been removed by the averaging filter. This is one of the benefits of using this filter as the pixel intensity was corrected by its neighboring values from a higher intensity to closer to its original intensity. The resulting image was smoother with minimal discoloration spots. Other benefits include a higher PSNR value which indicates that the image quality has been improved. The drawbacks of the filter include the fact that the image, visually, is not as sharp as the noisy image. As well, the edges are not as well-defined (for example, Lena's hair). The averaging filter affects the edges and other high frequency components of the image. There seems to be a tradeoff between smoothness and

sharpness when using the averaging filter. Overall, the denoised image is of higher quality than the noisy image.

12. The 7×7 filtered denoised image, Figure 19, looks much smoother/blurrier as compared to the 3×3 filtered denoised image, Figure 17, upon initial visual inspection. The different gray levels are represented more evenly throughout the image and there is a greater reduction in white spots and noise in the 7×7 averaging versus the 3×3. However, the 7×7 the image is blurrier and although the different sections of gray levels are portrayed better, the edges are not as sharp or refined. The sharpness and edges in the foreground, background and the subject (Lena) are reduced in 7×7 averaging as the detailing is lost due to increased averaging to correct the intensity levels which causes an increased blur effect. The shadows, contrast, and contours of the image have been negatively impacted by the 7×7 averaging filter. The blurring is more prominent in the 7×7 filter because it averages over a wider area of the image.

    The PSNR value of Figure 19, the 7×7 averaging filtered denoised image, is 26.2490 whereas the PSNR value of Figure 17, the 3×3 filtered denoised image is 30.6500. The PSNR value decreased by approximately 4.4 units from the 3×3 to the 7×7 image which implies that there was a degradation in image quality. This is indicative in the increased blurring effect that caused the image to lose details. In question 11, a tradeoff between smoothness and blurring was noted and it appears that the increase in smoothness does not outweigh the increased MSE the image pixels undergo due to the increase in blurriness. The blurriness reduced the image quality to an extent where the additional smoothing did not have a positive effect on the PSNR value. The 7×7 averaging filter technique's PSNR value is actually less than the denoised image, Figure 14, which is 26.9982. The 7×7 image has a lower PSNR value as the averaging window is larger and the higher frequency details (i.e. feathers in Lena's hat) of the image are blurred.

13. Both histograms, Figure 18 and 20, follow a similar shape and pattern with one major difference. In Figure 20, 7×7 filtered histogram, there are greater intensity values towards the middle gray levels. This shift is noticeable in the frequency reduction of non-peak intensity value ranges. The peaks in the 7×7 histogram are higher and sharper than the 3×3 histogram's which is a direct result of the increased averaging that occurred; it is also more similar to the histogram of the original image (without noise). This is due to the fact that it has a larger averaging window; it filters extreme values. In the original image, areas with a uniform distribution, and with a spike in the histogram, appear the same as the image and histogram of the denoised image. The pixel intensities are more spread out in the 3×3 histogram and as a result have more differing gray level components in the image with better edge definition and overall image quality. These differences occurred due to the increased kernel size. With a greater kernel size, the average value of each transformed pixel will be dependent on its neighbours and in averaging filters, they tend to eliminate pixel intensities that do not represent their local surroundings. This resulted in lower high frequency detail.

14. A larger window size makes the image look smoother/blurrier, the noise has been reduced. Each different section of the image with a different gray level is more normalized and that section will look more uniform after the transformation, similar to the original noiseless image. The drawbacks include increased blurriness and attenuated high frequency detailing. In addition, at edges, the neighbouring pixels can get corrupted and lose their original values as the gray levels from either side distort the clear edge boundary.

15. Figure 22 shows the denoised image from applying the Gaussian filter kernel. The Gaussian denoised image is less blurry and has preserved fine detailing as compared to the two averaging filter kernels. In Gaussian filtering, the neighboring pixels that are further away from the centre (spot) pixel have a lower weight (less of an effect) on the output value of the transformed pixel than the closer ones. In averaging filter kernels, the weights are uniform so the bias introduced from neighboring pixels that aren't representative of the pixel in large kernel window sizes is increased. This also distorts and blurs the edges. Thus, the Gaussian weighted filter preserves edges better.

    The PSNR value of the Gaussian filter is greater than both of the averaging filters. The value is 31.7881 for the Gaussian, 26.2490 and 30.6500 for the 7×7 and 3×3 averaging filters, respectively. For the 3×3averaging filter, it was able to reduce the noise by trying to smooth out each pixel by averaging it with the neighboring pixel intensities. This process produced an increased PSNR value from the original contaminated image but when the window size increased the PSNR value decreased from the contaminated image. The effect of increasing the window size to the point where the unrepresentative pixel intensities caused an extreme shift in the pixel value and resulted in a lesser image quality. Since the Gaussian filter is weighted with respect to the distance from the center pixel the transformations applied resulted in a more clear and smooth image.

16. Since the Gaussian filter does not smooth out extreme values, the high peaks from the averaging filter kernels are not as prominent. In Gaussian filtering, if a spot pixel is at an extreme value (either higher or lower) compared to its neighbours, it will be more biased towards this extreme value as it will be weighted the highest. This non-uniformity is what helps to preserve the edges and the detailing in the images (high frequency components). Figure 23, displays a great balance between the smoothness and edge sharpness evident in the distribution of the pixels where it is more balanced in certain areas while still maintaining all various peaks. The overall shape of the histogram has remained similar.

17. The drawback of using a Gaussian kernel over an averaging kernel is that it performs poorly in reducing severe noise (extreme pixel intensities) as well as an uniform kernel of any size.
    The benefit is that it will produce a much less blurry image with greater sharpness and detailing in the output image. The HVS (Human Visual System) has a primitive inclination to choose

sharpness over smoothness, regardless of the PSNR values, so a Gaussian kernel can be deemed as the better choice amongst the two filters in most cases. The quality and detailing is preserved with Gaussian better.

18. Both filtering options shown in Figure 26, averaging, and Figure 28, Gaussian, performed mediocrely in terms of noise reduction for the Salt and Pepper noise image shown in Figure 24. The original Salt & Pepper image is at a PSNR level of 18.4386 (calculated for reference). The averaging kernel has a PSNR value of 25.5094 and the Gaussian is at 27.0839 points. Although both methods have increased the PSNR value, implying improved image quality, the resulting images still have their downsides. The PSNR values are increased from the Salt and Pepper noise because after the transformations are applied the 0 and 1 pixel intensities that were applied throughout the image are greatly reduced. So the MSE is reduced with more normalized grey levels.

    The averaging kernel removed a lot of the noise introduced by the Salt and Pepper model however, the blurring of the image increased greatly. The white and black spots have also blended in with the surrounding gray levels making it a smoother, less sharp image.

    The Gaussian kernel performed worse at removing the noise from the Salt and Pepper model but maintained the high frequency detailing. The reduction in noise suffered because the noise caused by the Salt and Pepper was preserved as the weighting of those pixels near the centre were higher.

19. Figure 27, averaging histogram, and Figure 29, Gaussian histogram, are both similar in pixel intensity distribution and overall shape. The Gaussian histogram has sharper and more peaks than the averaging histogram. The averaging histogram has more normalized and fewer peaks. Figure 25, is of the noisy Salt and Pepper image, and the peaks are less defined. All histograms have peaks that are more flat and normalized as compared to the original histogram in Figure 13. Due to the introduction of the 0 and 1 intensity pixel values, the filters caused the pixels around the noisy pixels to distort to the noise and become more uniform.

20. As compared to the denoised image using the averaging and Gaussian filters, the median filter produced an image, Figure 30, with much higher quality. The noise is almost completely eliminated and its PSNR value is also much higher with a value of 35.4426. It seems that this filter is much better at reducing extreme values that are present in noisy images produced by the Salt and Pepper model. This is because the filter only selects the median value from the kernel, and thus does not take into account any extreme values. The effect of the Salt and Pepper model is greatly reduced as the noise introduces pixels at extreme ends of the intensity (0, 1) which would be generally ignored.

5. Sharpening in the Spatial Domain

In Section 5, sharpening techniques based on spatial filtering were studied. First, the cameraman image was loaded and normalized. Then, a 7×7 Gaussian filter was applied to the image. This filtered image was then subtracted from the original image. These two images are shown in Figures 32 and 33. The code for this section can be seen in Appendix A, Figure A.4.

Figure 32: Gaussian Filtered Image

Figure 33: Subtracted Image of Cameraman

Next, the subtracted image was added to the original image. The resultant image is shown in Figure 34.

Figure 34: Sharpened Image of Cameraman (Coefficient 1)

Lastly, the subtracted image was multiplied by 0.5 and then added to the original image. The resultant image is shown in Figure 35.

Figure 35: Sharpened Image of Cameraman (Coefficient 0.5)

21. The subtracted image, Figure 33, looks mostly black with white outlines at the edges of the objects in the image (coat, camera, tripod etc.). The high frequency components of the original images that are preserved in the subtracted image are the areas where there is contrast (edges). This is because the Gaussian smoothing filter smooths edges by sampling neighbouring pixels. In the filtered image, regions with edges will change to be more smooth looking whereas regions with no edges will remain mostly unchanged. Hence, when subtracting the Gaussian filtered image from the original image, the high contrast regions (edges) will be preserved whereas smooth regions will be closer to zero as the area has not changed much from the original to the filtered version of the image.

22. The resulting image, Figure 34, appears to have high sharpness as compared to the original. The edges and areas of contrast are highlighted more due to the high sharpening (high-boost filtering). This is because the value of pixels in the edges (coat, camera etc.) are higher (since the high frequency components have been added again) and the value of the pixels in the smooth areas of the image remain the same, that is areas where the subtracted image had an intensity of zero. Thus, due to the increase of intensity of the edge pixels, the image appears sharper than the original.

23. The image, Figure 35, differs as it is less sharp than the previous. This is because the pixel intensities of the subtracted image were divided in half, reducing them. Hence, when the images are added, the pixel intensities are lower. This means that the edge pixel intensities (and other high frequency components) will also be lower, resulting in a less sharp image.

24. Multiplying the subtracted image by a factor less than one reduces sharpening whereas multiplying the same by a factor of greater than one increases sharpening. This factor is key in controlling the level of sharpening of an image. However, it should be noted that a factor of less than one reduces sharpening of high frequency components like edges but also noise (which is good). As well, a factor of greater than one can cause image artifacts to be over-sharpened which may not look good visually.

6. Conclusion

In Section 2, the effects of different impulse functions convolved with the Lena image were investigated. Specifically, it was seen that *h1* performed as a horizontal filter that blurred/smoothed the lines of the image along the horizontal axis, the *h2* function acted as a vertical filter that blurred/smooth lines of the image along the vertical axis, and *h3* acted as an edge detector as it highlighted the edges or high frequency components of the image with a higher intensity.

In Section 3, three different noise models and their effects on a synthetic toy image were studied. First, the additive zero-mean Gaussian model (variance of 0.01) modified the image by adding noise with a normal distribution around the original two images. Thus, the noisy pixels highly varied in intensity. Next the Salt and Pepper model (noise density of 0.05) affected the image by adding intensities of value 0 and 1 (black and white). Lastly, the Speckle noise model (variance of 0.04) affected the image by adding noise with a uniform distribution around the original two intensities. Thus, the noisy pixels had more similar intensities to that of the original image.

In Section 4, different noise reduction techniques, based on spatial filtering with differing filter parameters, were explored. The original grayscale and normalized Lena was first distorted with a zero-mean Gaussian noise model. Then two averaging filters were applied, a 3×3 and a 7×7 kernel sizes were used. The smoothing effect of the averaging filter was demonstrated along with the increase in blur was also observed. As the size of the kernel is increased, the smoothness and blurriness also increases; it is a trade-off. Then the Gaussian filter was applied and it was observed that the blurriness from the averaging filter output was rectified, as the weights depend on how close the pixel is to the centre. This retains the sharpness of the image however, may not remove large dense amounts of noise as it would preserve some values. Next a Salt & Pepper noise model was introduced to the original Lena image and both averaging and Gaussian filters were applied. Both filters improved the image quality through increased PSNR values however, both images still had the same downsides of their respective technique as aforementioned. The median filter was then applied to the Salt and Pepper noisy image and that resulted in the highest image quality from a visual perspective and the PSNR value.

In Section 5, we investigated the effects of the high boost filter. Here, it was found that the subtracted image appeared to highlight high frequency components of the image. When this subtracted image was added to the original, the resultant image was much sharper due to high boost filtering. Lastly, when the subtracted image was multiplied by 0.5 and then added to the original, it was less sharp then the previous since the intensity values had been lowered.

Appendix A

```
%% Section 2 – Discrete Convolution for Image Proccessing %%

% Load the Lena image and convert it to a grayscale image using the rgb2gray function
lena_image = imread("lena.tiff");
lena_grayscale = rgb2gray(lena_image);

% To get intensity of the image within the range of 0 to 1, use the double function on the image
% and then divide it by 255
lena_normalized = double(lena_grayscale)/255;

% Construct three impulse functions h1,h2,h3
h1 = (1/6)*ones(1,6);
h2 = h1';
h3 = [-1,1];

%  Use  the conv2 function  to  convolve  the  Lena  image  with  the
%  impulse  functions h1,h2,h3 separately
h1_conv = conv2(lena_normalized, h1);
h2_conv = conv2(lena_normalized, h2);
h3_conv = conv2(lena_normalized, h3);

% Plot the original image as well as the convolved images
figure;
subplot(2,2,1), imshow(lena_normalized);
title('Image 1: Lena – Normalized');
subplot(2,2,2), imshow(h1_conv);
title('Image 2: Convolution with h1');
subplot(2,2,3), imshow(h2_conv);
title('Image 3: Convolution with h2');
subplot(2,2,4), imshow(h3_conv);
title('Image 4: Convolution with h3');
```

Figure A.1: Section 2 Code

```
%% Section 3 – Noise Generation %%

lena_image = imread("lena.tiff");
cameraman_image = imread("cameraman.tif");

f = [0.3*ones(200,100) 0.7*ones(200,100)];
figure;
imshow(f);

% Noise models
f1 = imnoise(f, 'gaussian', 0, 0.01);
f2 = imnoise(f, 'salt & pepper', 0.05);
f3 = imnoise(f, 'speckle', 0.04);

% Plot the noise-contaminated images and the corresponding histograms for
% each of the noise models
figure;
subplot(3,2,1), imshow(f1);
title('Gaussian Model Toy Image');
subplot(3,2,2), imhist(f1);
title('Histogram of Gaussian Model Toy Image');
subplot(3,2,3), imshow(f2);
title('Salt & Pepper Model Toy Image');
subplot(3,2,4), imhist(f2);
title('Histogram of Salt & Pepper Model Toy Image');
subplot(3,2,5), imshow(f3);
title('Speckle Model Toy Image');
subplot(3,2,6), imhist(f3);
title('Histogram of Speckle Model Toy Image');
```

Figure A.2: Section 3 Code

```
%% Section 4 – Noise Reduction in the Spatial Domain %%

% Load the Lena image and convert it to a grayscale image using the rgb2gray function
lena_image = imread("lena.tiff");
lena_grayscale = rgb2gray(lena_image);

% To get the intensity of the image within the range of 0 to 1, use the
% double function on the image and then divide it by 255 (alternatively use im2double function)
lena_normalized = im2double(lena_grayscale);

figure;
subplot(1,2,1), imshow(lena_normalized);
title('Normalized Lena');
subplot(1,2,2), imhist(lena_normalized);
title('Histogram of Normalized Lena');

% Contaminate the Lena image with zero-mean Gaussian noise with a variance of 0.002
lena_contaminated = imnoise(lena_normalized, 'gaussian', 0, 0.002);

% Plot the noisy image and the corresponding histogram and PSNR between the
% noisy image and the original noise-free image
figure;
subplot(1,2,1), imshow(lena_contaminated);
title('Gaussian Noise Lena');
subplot(1,2,2), imhist(lena_contaminated);
title('Histogram of Gaussian Noise Lena');

lena_contaminated_psnr = 10*log10(1/mean2((lena_normalized-lena_contaminated).^2));

```

Figure A.3.1: Section 4 Code

```
% Create a 3x3 averaging filter kernel using the fspecial function. Plot this filter using imagesc and colormap (gray)
averaging_filter = fspecial('average');

figure;
imagesc(averaging_filter);
colormap('gray');
title('3x3 Averaging Filter');

% Apply the averaging filter to the noisy image using the imfilter function
lena_filter = imfilter(lena_contaminated, averaging_filter);

% Plot the noisy image and the corresponding histogram and PSNR between the
% noisy image and the original noise-free image
figure;
subplot(1,2,1), imshow(lena_filter);
title('3x3 Averaging Filtered Lena');
subplot(1,2,2), imhist(lena_filter);
title('Histogram of 3x3 Averaging Filtered Lena');

lena_filter_psnr = 10*log10(1/mean2((lena_normalized-lena_filter).^2));

% Now create a 7x7 averaging filter kernel and apply it to the noisy image.
averaging_filter_2 = fspecial('average', 7);
lena_filter_2 = imfilter(lena_contaminated, averaging_filter_2);

```

Figure A.3.2: Section 4 Code (cont'd)

```
% Plot the denoised image and the corresponding histogram.
figure;
subplot(1,2,1), imshow(lena_filter_2);
title('7x7 Averaging Filtered Lena');
subplot(1,2,2), imhist(lena_filter_2);
title('Histogram of 7x7 Averaging Filtered Lena');

% Also, compute the PSNR between the denoised image and the original noise-free image.
lena_filter_psnr_2 = 10*log10(1/mean2((lena_normalized-lena_filter_2).^2));
%
% Create a 7x7 Gaussian filter kernel with a standard deviation of 1. Plot the filter.
averaging_filter_3 = fspecial('gaussian', 7, 1);

figure;
imagesc(averaging_filter_3);
colormap('gray');
title('7x7 Gaussian Filter');

% Plot the denoised image and the corresponding histogram.
lena_filter_3 = imfilter(lena_normalized, averaging_filter_3);

figure;
subplot(2,1,1), imshow(lena_filter_3);
title('7x7 Gaussian Filtered Lena');
subplot(2,1,2), imhist(lena_filter_3);
title('Histogram of 7x7 Gaussian Filtered Lena');

```

Figure A.3.3: Section 4 Code (cont'd)

```
% Compute the PSNR between the denoised imageand the original noise-free
% image
lena_filter_psnr_3 = 10*log10(1/mean2((lena_normalized-lena_filter_3).^2));
%
% Create a new noisy image by adding salt and pepper noise (density 0.05) to the image.
lena_noise = imnoise(lena_normalized, 'salt & pepper', 0.05);
lena_salt_pepper_psnr = 10*log10(1/mean2((lena_normalized-lena_noise).^2)); % PSNR Value for reference

% Apply the 7x7 averaging filter and the Gaussian filter to the noisy image separately.
lena_filter_4 = imfilter(lena_noise, averaging_filter_2);
lena_filter_5 = imfilter(lena_noise, averaging_filter_3);

% Plot the noisy image, the denoised images using each method, and the corresponding histograms.
figure;
subplot(2,1,1), imshow(lena_noise);
title('Noisy Image');
subplot(2,1,2), imhist(lena_noise);
title('Histogram of Noisy Image');

% Denoised Images
figure;
subplot(2,2,1), imshow(lena_filter_4);
title('7x7 Averaging Filter on Salt & Pepper Noise');
subplot(2,2,2), imhist(lena_filter_4);
title('Histogram of 7x7 Averaging Filter on Salt & Pepper Noise');

```

Figure A.3.4: Section 4 Code (cont'd)

```
subplot(2,2,3), imshow(lena_filter_5);
title('7x7 Gaussian Filter on Salt & Pepper Noise');
subplot(2,2,4), imhist(lena_filter_5);
title('Histogram of 7x7 Gaussian Filter on Salt & Pepper Noise');

% Also, compute the PSNR between the denoised images and the original noise-free image
lena_filter_psnr_4 = 10*log10(1/mean2((lena_normalized-lena_filter_4).^2));
lena_filter_psnr_5 = 10*log10(1/mean2((lena_normalized-lena_filter_5).^2));

% Apply the median filter on the noisy image. The medfilt2 function will come in handy for this.
lena_filter_6 = medfilt2(lena_normalized);

% Plot the denoised image and the corresponding histogram.
figure;
subplot(2,1,1), imshow(lena_filter_6);
title('Median Filtered Lena Image');
subplot(2,1,2), imhist(lena_filter_6);
title('Histogram of Median Filtered Lena Image');

% Also, compute the PSNR between the denoisedimage and the original
% noise-free image
lena_filter_psnr_6 = 10*log10(1/mean2((lena_normalized-lena_filter_6).^2));

```

Figure A.3.5: Section 4 Code (final part)

```
%% Section 5 - Sharpening in the Spatial Domain %%
% Load the Cameraman image and get intensity of the image within the range of 0 to 1
cameraman_image = imread("cameraman.tif");
cameraman_normalized = im2double(cameraman_image);

% Apply the 7x7 Gaussian filter on Cameraman image & subtract Gaussian-filtered image from the original Cameraman image
averaging_filter_2 = fspecial('average', 7);
cameraman_filered = imfilter(cameraman_normalized, averaging_filter_2);

subtracted_image = cameraman_normalized - cameraman_filered;

% Plot both the Gaussian-filtered image and the subtracted image
figure;
subplot(1,2,1), imshow(cameraman_filered);
title('7x7 Averaging Filtered Cameraman');
subplot(1,2,2), imshow(subtracted_image);
title('Subtracted Image');

% Add the subtracted image to the original image. Plot the resulting image
sharpened_image = cameraman_normalized + subtracted_image;
figure;
imshow(sharpened_image);
title('Sharpened Image of Cameraman');

% Multiply the subtracted image by 0.5 and then add it to the original image
sharpened_image_2 = cameraman_normalized + subtracted_image*0.5;

% Plot the resulting image
figure;
imshow(sharpened_image_2);
title('Sharpened Image of Cameraman (Subtracted image multiplied by 0.5)');
```

Figure A.4: Section 5 Code