SYDE 575

Group #22

Lab 2 Report

Prof David A. Clausi

Abdulhameed Naji - *20642623*

Shilpan Shah - *20603389*

Contents

List of Figures

1 Introduction

In this lab, different techniques were used to study noise generation & reduction, fundamentals of image enhancement, and restoration concepts. Image enhancement and restoration are important aspects of image processing with various application that rely on the quality of the image. Some applications of image enhancement include medical imagining, object detection and tracking, and video editing.

Noise was generated on two sample images; the noise models used include additive zero-mean Gaussian, Salt & Pepper and Multiplicative Speckle. Each noise model produces a varying strength and pattern of granularity that is a function of parameters such as variance or density. Images were then de-noised using spatial filters such as the Average, Gaussian, and Median filter - the resulting images then formed a basis for comparison to identify the most suitable filtering method as a function of the image and noise model.

Finally, unsharp masking was used to perform an edge boost operation on a sample image. The process produces an image with exclusively high frequency components by subtracting low frequency components. The resulting edge mask is then scaled and superimposed on the source image to control the extent of edge sharpening.

2 Noise Generation

The experiment creates an image consisting of 2 vertical solid bars of contrasting intensities; various noise models are then applied to the image. The source image and its associated histogram are shown in Figure 1. The image is then contaminated with Gaussian noise of mean 0 and variance 0.01, Salt & Pepper noise with noise density 0.05, multiplicative Speckle noise with variance 0.04 - the three noisy images and their respective histograms are shown in Figures 2, 3, and 4.

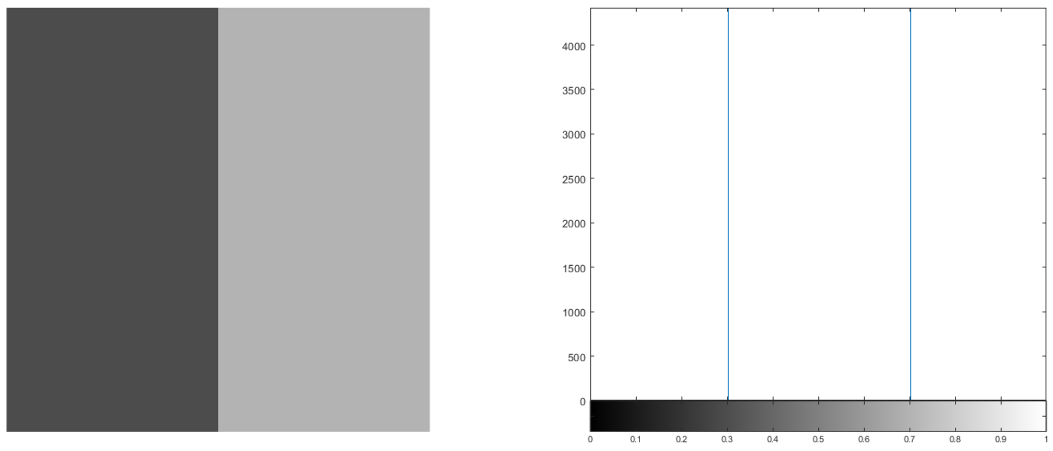


Figure 1: Comparison of source images

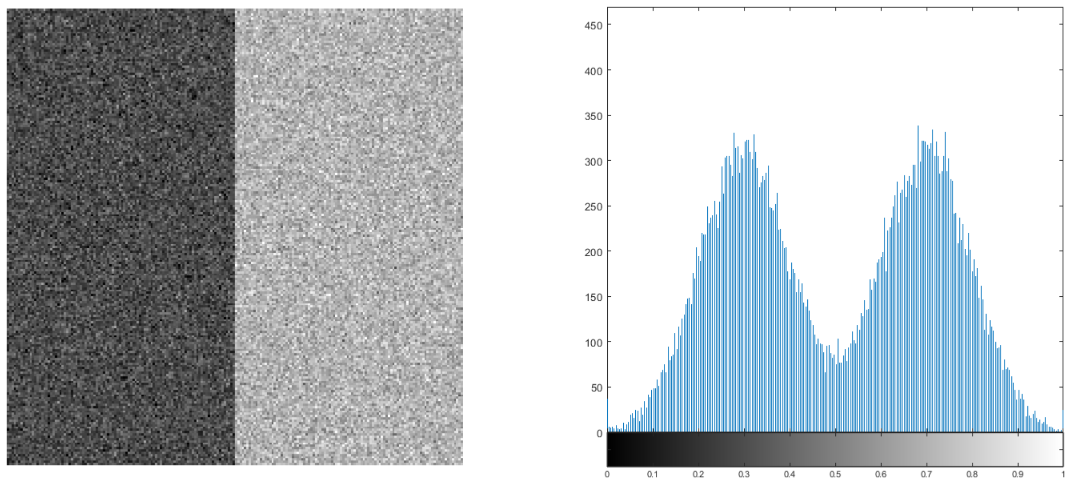


Figure 2: Image contaminated with Gaussian noise

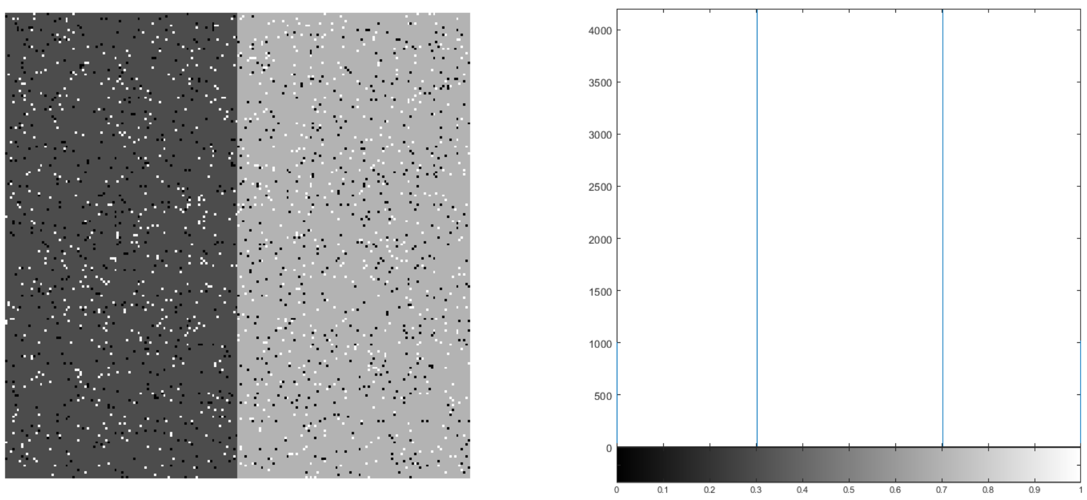


Figure 3: Image contaminated with Salt& Pepper noise

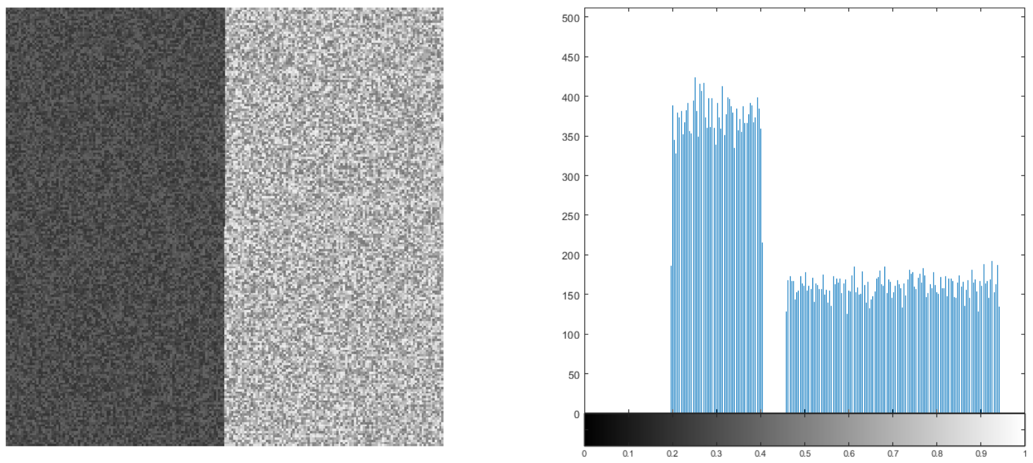


Figure 4: Image contaminated with Multiplicative Speckle noise

2.1 Histogram Interpretation

The Gaussian noise model produces two peaks on each side of the vertical bars. The Gaussian distribution is that of a bell curve - since the noise model is additive, the bell curve is superimposed with the two peak bars shown in the source image in Figure 1. The result is a bell curve distribution localized around a peak formed by the source image histogram. Visually, the peaks correspond to the two distinct intensity boundaries of the solid bars. From there, a Gaussian distribution forms around the peaks in order to produce a range of intensities following the normal distribution.

Salt and Pepper noise is characterized as an impulse noise. Noise added by this model is purely black or white, and occurs due to sharp or sudden disturbances in an image. An example would include a 3D rendered scene that contains invalid or physically inaccurate materials, which in turn causes overexposed or unlit intensities for a pixel (white and black respectively). As such, the *shape* of the histogram formed by the noise model is preserved, but new levels are introduced in intensities 0 and 1 respectively. Since the noise manifests binary intensity values, the 0 and 1 grey levels appear as discrete impulses.

Speckle noise is an additive model that adds a multiplicative value of the source image pixel as noise. The noise uses a uniform distribution with a mean of 0 and variance of 0.04, this means the random multiplicative factors have higher probabilities for selection when within the extents formed by the variance. The histogram of the image gets broader and more uniform. Since the speckle noise uses the source pixel, the change in histogram will be relative and heavily localized - this leads to a flatter histogram. The higher the variance, the more drastic and random the output of the uniform distribution is - this in turn leads to a larger relative change of pixel intensity. In effect, this will broaden the histogram from its original peak value.

2.2 Visual Comparison

Generally speaking, all noise models added a form of granularity to the image. The noise types primarily influence the relative intensities added to the image.

The Gaussian noise uses a normal distribution centered around 0 with a variance of 0.04. This means that the noise is very likely to manifest values within the 0.2 standard deviation range from 0. The farther from the mean, the less likely - visually this translate into smoother, less sudden changes in color. The grains maintain spatial coherency because their values cannot be drastically different from each other.

Conversely, Salt and Pepper noise is a binary value noise - the image is contaminated with purely black and white intensities. The selected Salt and Pepper noise creates less grains (which is a function of noise density), but has more sudden changes in color.

Speckle noise applies completely random variations in color due to its uniformly distributed multiplicative value (recall a uniform distribution is flat); however, the purely random values are applied to the original image's pixel intensities. As a result, speckle noise produces a relative change in intensity whose change is a function of the uniform distribution's variance. More variance correlate to stronger multiplicative values, which in turn strongly offset the intensities of the original image. As a result, the resulting noisy image is sharper and more random than the Gaussian model, but its effect is localized by the source pixel intensity.

2.3 Underlying Distribution in Speckle Noise

The overall shape of the histogram is flat. This is a property of the uniform distribution, where each value has equal probabilities of occurring. The multiplicative speckle adds a noise value that is the source pixel color scaled by a random sample picked from the uniform distribution - in this case, with variance 0.04. The formulation is as follows:

$$g(x, y) = f(x, y) + rand(0, 1) * f(x, y)$$

Based on the formulation, the variance of the uniform distribution controls the width of the histogram bins. As variance grows, a wider selection of samples are available from the uniform distribution, causing the intensities to offset more significantly. This in turn translates into new bins forming further away from the original peak. As a result, the traded pixels will widen the histogram and reduce the extent of the peak.

2.4 Peaks of histogram in Speckle Noise

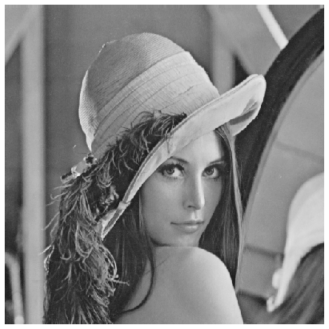
An important observation about this noise model is that the noise strength is relative to the source pixel. If the source pixel is dark, the result of the scaled additive noise will be small, which in turn does not deviate the pixel's intensity significantly. This is observed in both the image and the histogram; the dark half of the image has grain patterns that remain within the darker ranges. The corresponding histogram is narrow - indicating that the noise had lesser impact.

Conversely, a bright color is represented with a larger value. When multiplied by the uniform distribution sample, it creates a noise value that may be of high intensity. Hence, the right half portion of the image noise grains are more pronounced due to the presence of wide deviation in intensities. The effect is further highlighted in the histogram - since the noise effect is stronger due to the high intensity values in the brighter ranges, the new resulting pixels will be superimposed with a large value that will offset their intensities significantly. The histogram consequently is wider than the darker portion of the image.

Since intensities are being shifted relative to the original grey levels, new bins are formed around the former peaks. As more pixels deviate in intensity, the original bin becomes smaller, and the surrounding bins increase to represent the result of the noise model. Since the distribution used is uniform, the bins naturally form a flat histogram in accordance to the effect of the uniform distribution.

3 Noise Reduction

This lab component applies Gaussian and Salt & Pepper noise to the original Lena image shown in Figure 5. The image is then de-noised comparatively using averaging and Gaussian filters. The final PSNR values are tabulated in Figure 6 below, and will be discussed further.



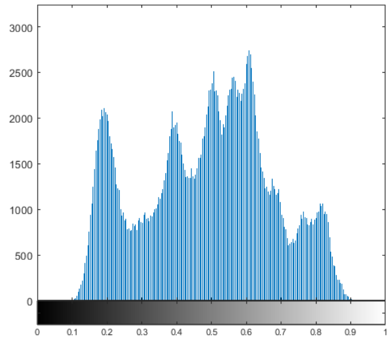


Figure 5: Sample image to be contaminated and de-noised

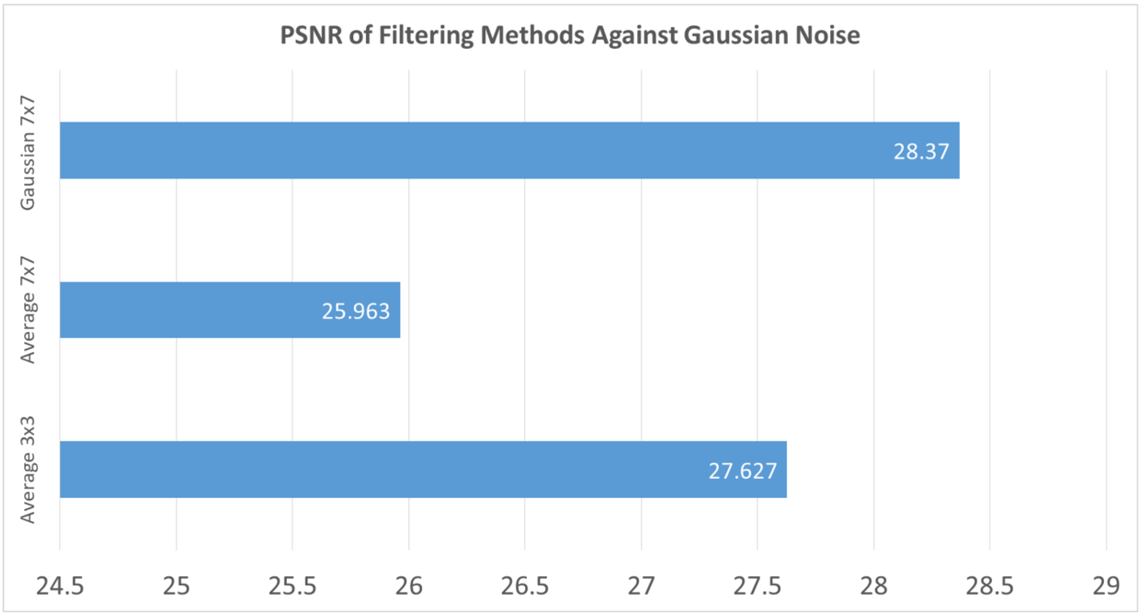


Figure 6: PSNR of filtered images operating on Gaussian noise.

3.1 Gaussian Noise

The first experiment applies a Gaussian noise of variance 0.002. The resulting image is shown in Figure 7 with a corresponding PNSR value of 20.068. As with the noise generation examples, the histogram of the resulting image is a superimposition of the normal distribution to the source histogram. The noisy image contains fictitious intensities introduced by the normal distribution. Note that while the noisy image follows the normal distribution, the peaks and deformities are relative to the peak grey levels present in the original image.

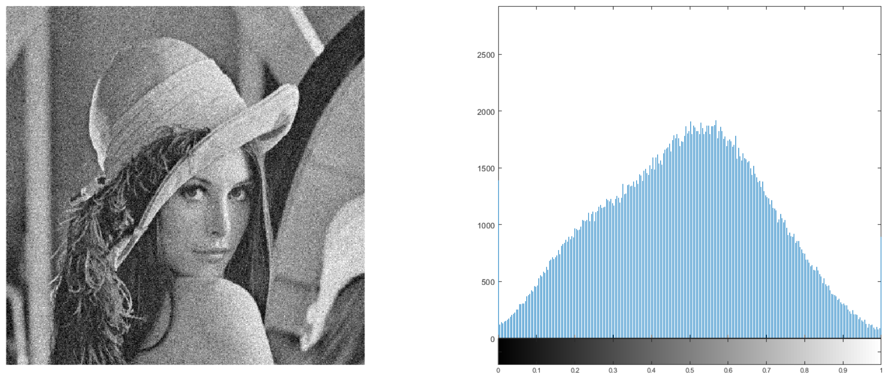


Figure 7: Noisy image caused by Gaussian noise

3.1.1 De-noise using 3x3 Average

The result of the de-noise operation is shown in Figure 8. Visually, the granularity in the image is slightly suppressed at the cost of color contrast and edge details.

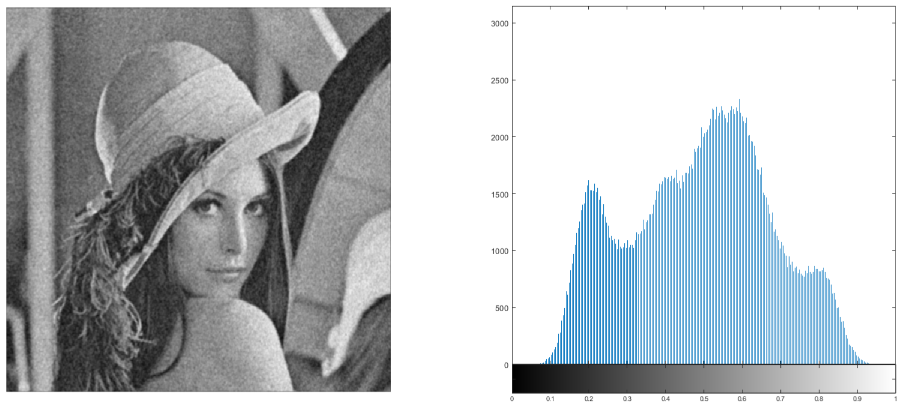


Figure 8: Image denoised using 3x3 Average filter

The color map of the filter is shown in Figure 9. The filter is a solid grey color because it represents an even weight contribution of pixels represented by a 3x3 grid. In this case, in order to maintain a DC gain of 1, each pixel contributes $\frac{1}{9}$th of its intensity to the averaging calculation.

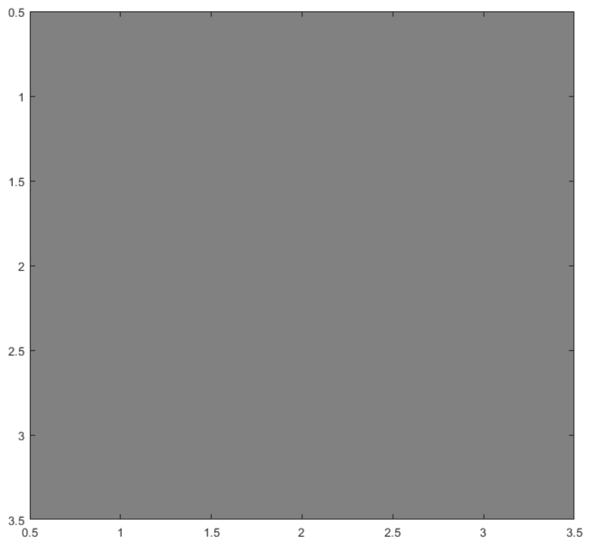


Figure 9: Colormap of 3x3 Average filter

The PSNR increased to a value of **27.627** which indicates the filter successfully reduced intensity error due to the noise.

The normal distribution trend present in the noisy image is effectively reduced; but the grey level peaks are not as refined as the source image. This correlates to the visual interpretation of the extent of blur as not all the normal distribution components induced by the Gaussian noise are rectified. In some ways, the filter did not fully ”invert” the noise values.

Visually, the intensity of the noise is reduced - deviation of noise samples are smaller and hence they appear smoother. The image is improved, but at the cost of blurring and edge detail suppression.

3.1.2 De-noise using 7x7 Average

Increasing the average operation to include more surrounding neighbor is equivalent to increasing the strength of the blur. Figure 10 shows the de-noised image. The image is significantly blurrier since a larger mask correlates to averaging of further away pixels. The PSNR value understandably decreased to **25.963**, which verifies our visual interpretation of a worse representation compared to the 3x3 filter. Although the Gaussian noise is practically non-existent, the same can be said about the original image's fine and edge details.

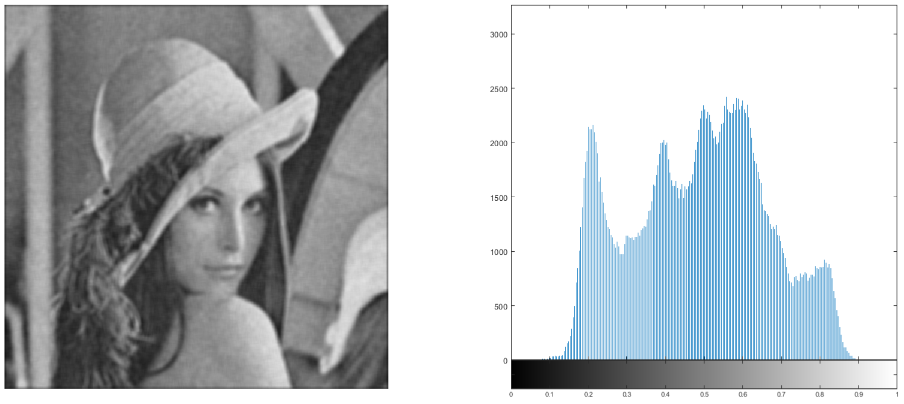


Figure 10: Denoised image using 7x7 Average filter

The histogram appears narrower and sharper. This corresponds to the observed reduction of the Gaussian noise - the broader normal distribution is effectively removed, however the histogram fails to capture some peak details especially around 0.6 to 0.7 intensity ranges. As such, some pixel intensities were blurred and lumped into surrounding bins. By strengthening the blur, intensities are being averaged to approximately similar values - hence, they truncate to the same bin to form narrower peaks. Notice that although the histogram is similar to the source image, the resulting image is not representative of the source image. This highlights the fact that a histogram has no spatial context; hence, very little can be said about how the image actually looks based on histogram alone.

Although the filter is effective at removing noise, its final result is undesirable because it incorporates information from pixels that are less likely spatially related.

3.1.3 De-noise using Gaussian

The average filter is now replaced with a more spatially aware filter. The result of a 7x7 Gaussian filter with standard deviation of 1 is shown in Figure 11. The associated color map is shown in Figure 12.

Upon first inspection, the Gaussian filter performs significantly better than a 7x7 averaging filter in terms of detail preservation. The intensities in the image remain mostly intact, whereas the 7x7 average filter smeared some of the color to darker values. Unlike the averaging filter however, there is a trade-off in terms of its noise reduction. That said, the noise reduction is superior to that of the 3x3 average. In other words, the Gaussian filter offers a middle ground between the 3x3 and 7x7 average filter.

Since the filter worked to remove some of the noise present in the image while also retaining prevalent contrast and color information, the PSNR value is the best of the examined methods with a value of **28.370**.

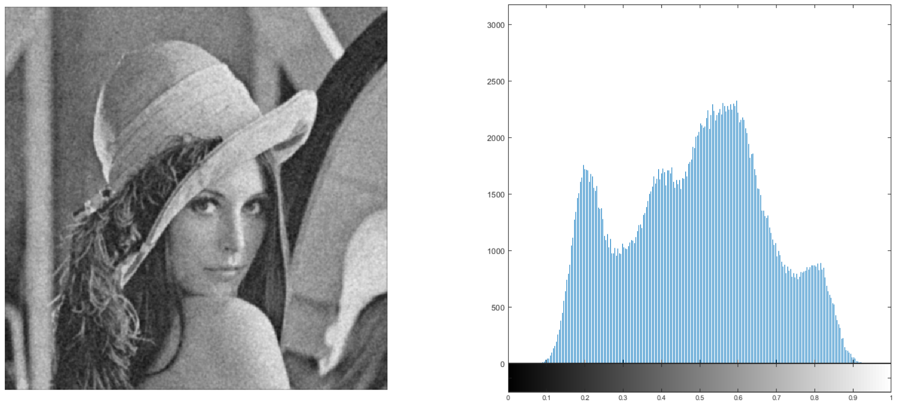


Figure 11: Denoised image using the Gaussian filter

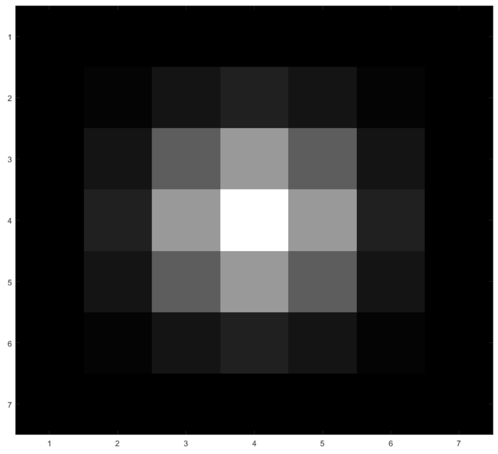


Figure 12: Colormap of Gaussian filter

The histogram is very similar to that of the 3x3 average, but with narrower ranges and peaks. However, it does not match the 7x7 average and source image in terms of smoothness or roundness of the curve - this is mostly due to the remaining imposition of Gaussian noise, forcing the histogram to tend more to a normal distribution.

As seen by the color map of the filter, it weights pixels closer to the target pixel higher when performing its convolution - this in turn makes the filter reject the impact of pixels that are farther away while also retaining a large masking window. As a result, the filter performs enough blur to suppress the intensity of the Gaussian noise while also reducing the effect of intensity smearing due to naive averaging.

3.2 Salt & Pepper Noise

Salt and Pepper noise is applied to the source image, and the 7x7 Gaussian and average filters are applied in order to compare their effectiveness versus the median filter. The result of the contamination is displayed in Figure 13. The PSNRs from each methods are displayed in Figure 14 and will be discussed further.

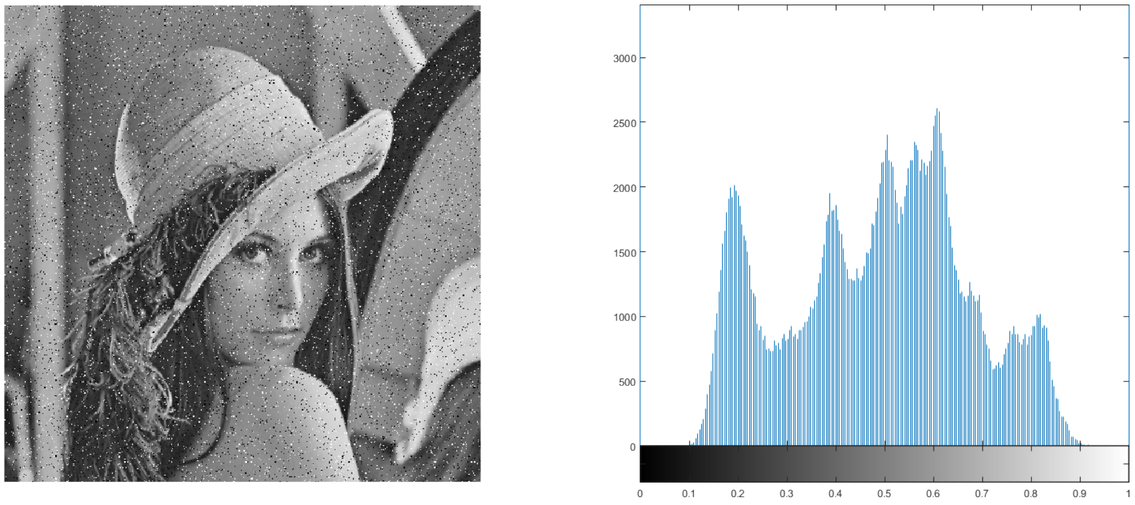


Figure 13: Lena image contaminated with Salt and Pepper noise

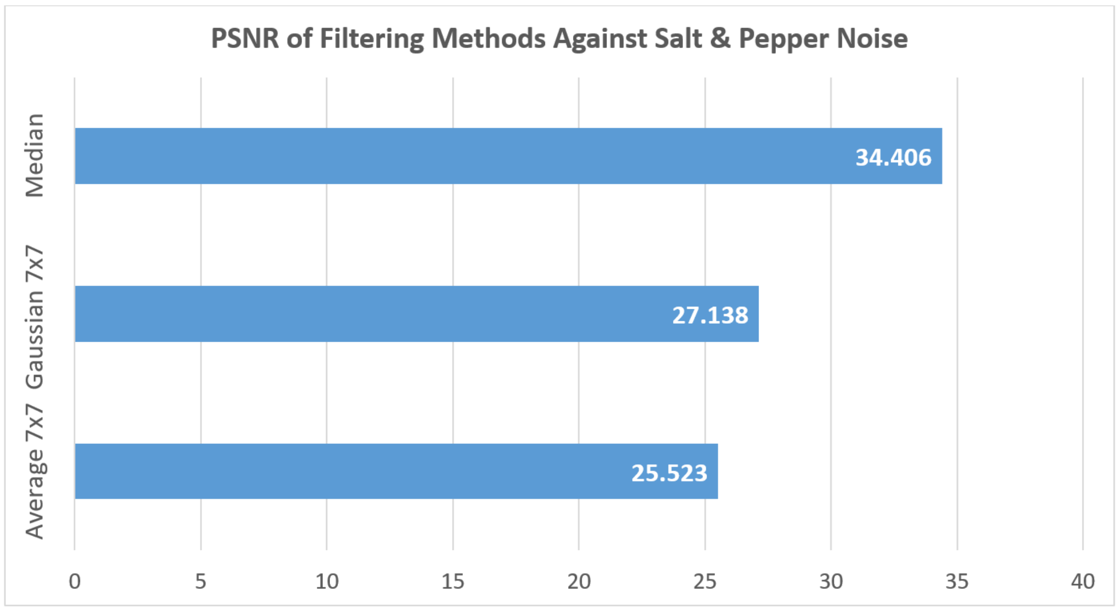


Figure 14: PSNR of filtered images against Salt & Pepper noise

3.2.1 Gaussian & Average Filters

As with the previous image, the Average filter completely removes notable noisy pixels since it completely sacrifices the image integrity. The complete noise rejection and edge loss is shown in Figure 15. In contrast, the Gaussian filter fails to completely negate the noise - this is because the Gaussian filter is simply a weighted average - instead of completely rejecting the faulty noise sample, it averages the pixel with surrounding neighbors. This in effect reduces the extremities of the noise as seen in Figure 16 - the Salt and Pepper noise is blurred into a grey intensity as opposed to being purely black and white.

The relative shape of the histograms remain the same. The presence of the Salt and Pepper noise influences the width of the histogram. When statistically averaged (such as the Gaussian case), the histogram widens. When intensities are heavily blurred together, the histogram narrows due to color lumping as shown in the average filter case.

The PNSRs of the average and Gaussian methods are **25.523** and **27.138** respectively; further emphasizing the detail destruction caused by a large average mask.

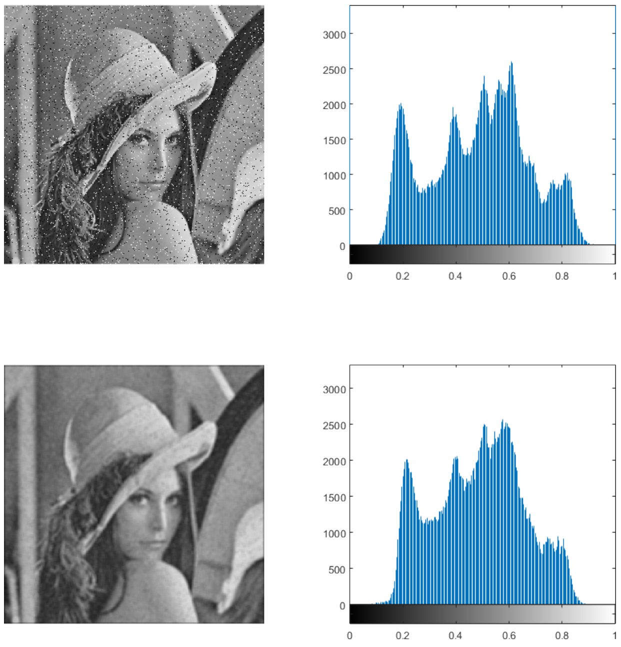


Figure 15: Effect of applying 7x7 average filter on Salt and Pepper noise

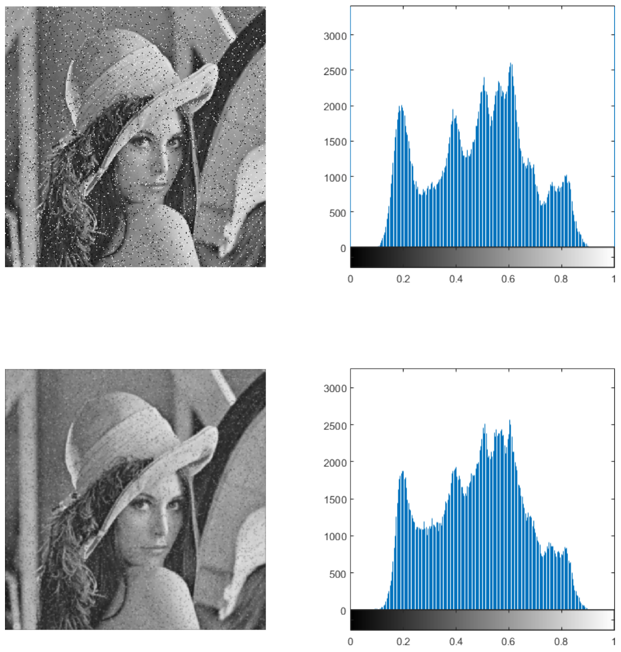


Figure 16: Effect of applying 7x7 Gaussian filter on Salt and Pepper noise

3.2.2 Median Filter

The median filter is especially effective on Salt and Pepper noise as seen in Figure 17. The resulting image is virtually identical to the source image, with rich edge and contrast details and nearly no Salt and Pepper noise present. As a result, the PSNR significantly improved to **34.525**. The median filter is effective in rectifying impulse type noise, this is because it does not assume all pixels in its window are relevant. Instead, by finding the median of the set, it deduces that an outlier intensity such as the one caused by Salt and Pepper noise should not belong in the image. In effect, the median filter rejects extreme changes in intensities - which is exactly what Salt and Pepper noise contributes. As such, it cures the image in an ideal fashion.

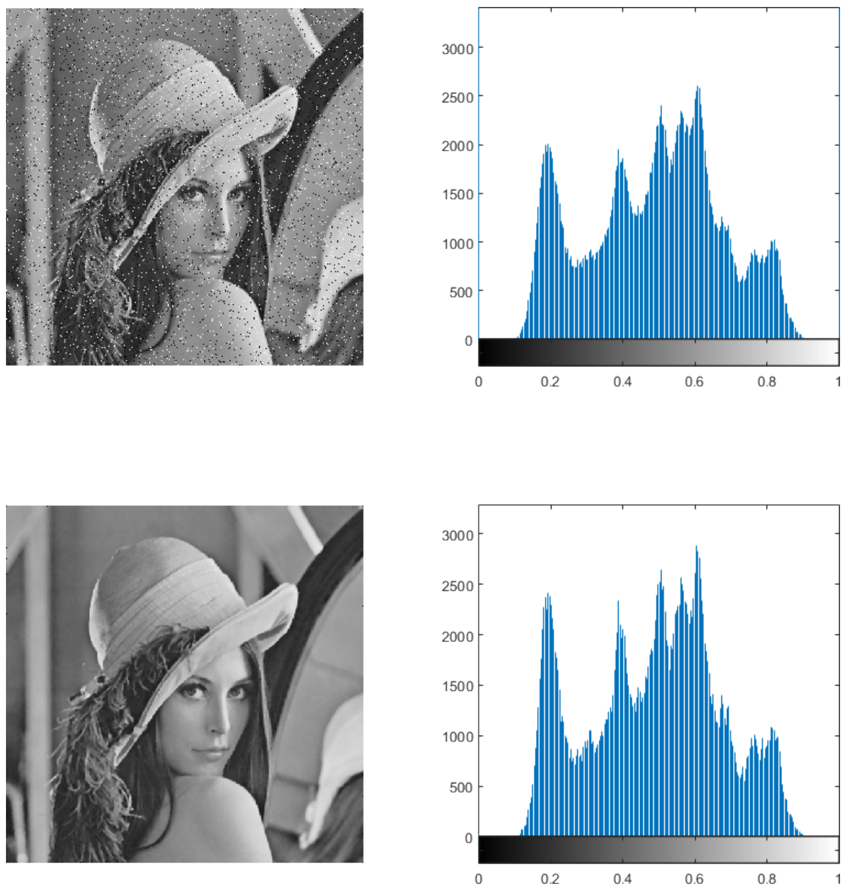


Figure 17: Effect of applying median filter on Salt and Pepper noise

4 Image Sharpening

The high-boost filter is implemented by creating an edge mask formed by means of isolating high frequency components of the image.

4.0.1 Subtracted Image

The subtracted image depicted in Figure 18 is essentially an edge mask. By applying a Gaussian blur, we effectively perform low pass filtering - now high frequency contents (edges) are notable. Subtracting the filtered image from the source will in turn cancel out all low frequency components. The remaining result are only the high frequency contents - which in this image manifests itself as the edges of the cameraman.

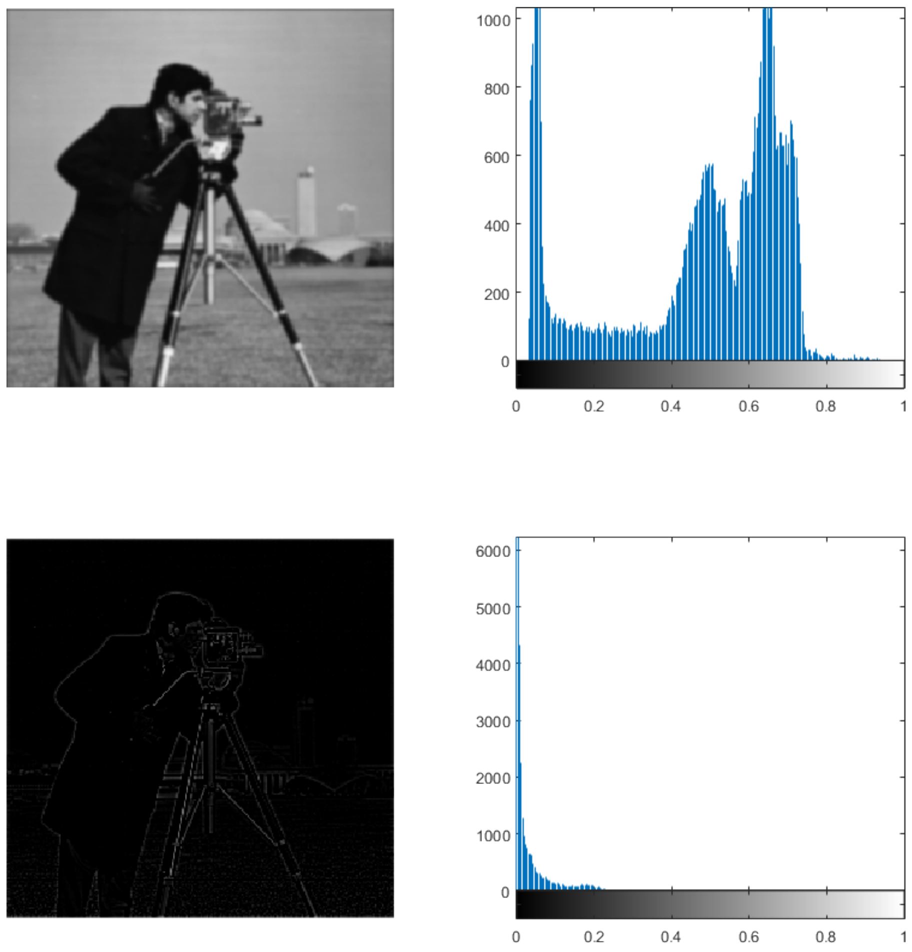


Figure 18: Blurred image(top) and the result of subtraction (bottom)

4.0.2 Added Image

By adding the edge mask to the source image, the high frequency components have their values increased. Visually, the edge mask is added on top of the source image - hence, the cameraman edges have their intensities added and increased. Consequently, the resulting image has sharper edges as shown in Figure 19.

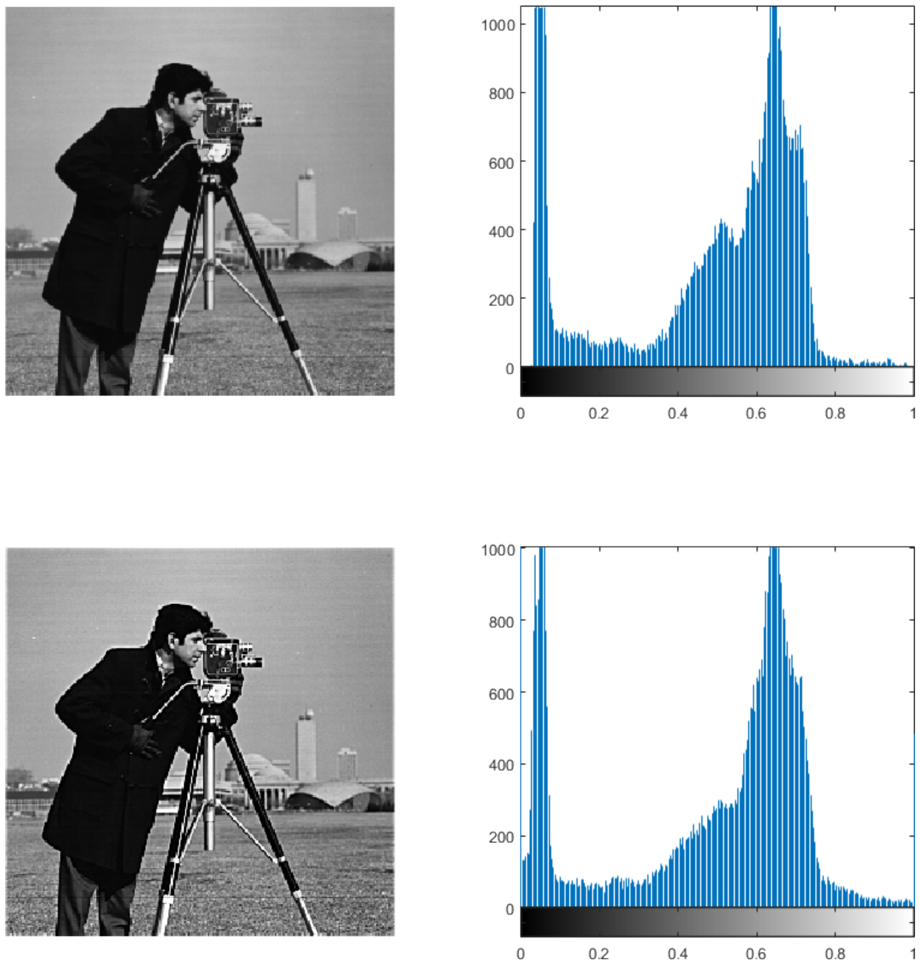


Figure 19: Source image (top) and the sharpened image (bottom)

4.0.3 Reduced Edge Mask

Since the sharpening is a matter of adding the high frequency pixel intensities to the source image, the strength of sharpening can be easily scaled by multiplying the edge mask by a factor. A factor of 0.5 decreases the intensities of the mask by half - as a result, when applied to the image, the effect of sharpening is slightly less pronounced than the case of using the full strength of the edge mask. Figure 20 better visualizes the difference. Less contrast and artifacts are observed around the edges for a reduced sharpening operation.

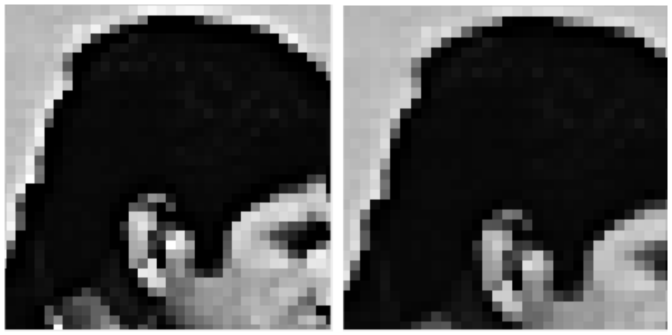


Figure 20: Image with default edge mask (Left) versus image with edge mask reduced by half (Right)

Since the edge mask can be scaled to control the sharpening effect, it can be concluded that using a multiplier greater than 1 would strengthen the sharpening effect. Since the scaling is linear, using a sufficiently strong multiplier can inadvertently saturate the image intensities. Figure 21 shows an example of an edge boost using a factor of 10. The image colors saturate to the maximum brightness values in some portions, and the overall image is brighter. The histogram captures this effect - the previously prevalent grey levels at 0.6-0.8 have been smoothed and shortened. Some of the components of that bin shifted to the brighter end of the histogram. The same can be said for the previously prevalent peak at 0-0.1.

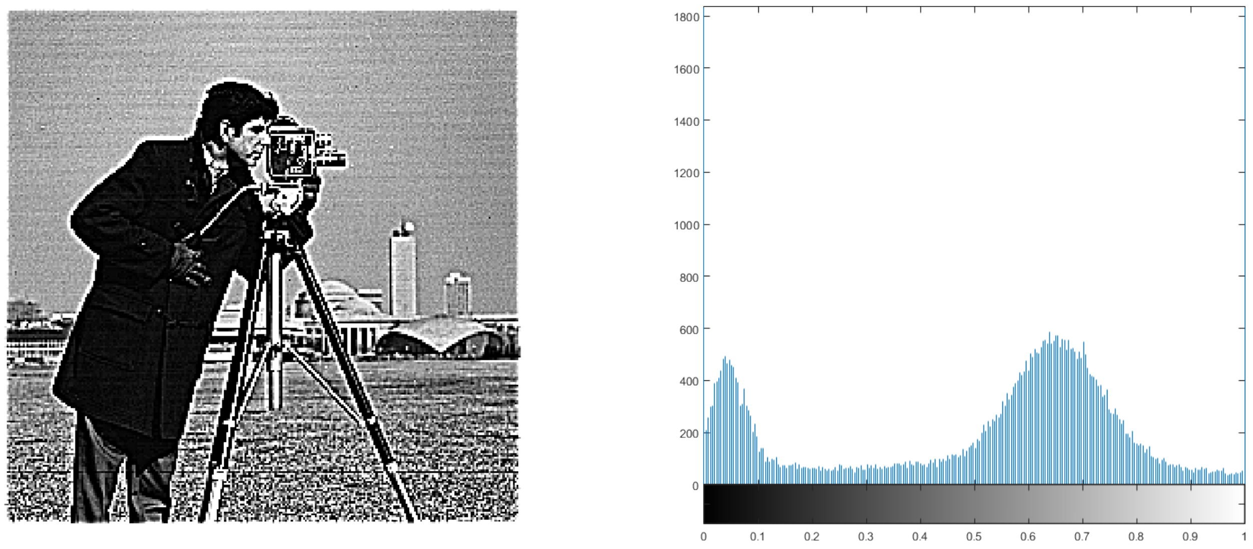


Figure 21: Image edge boosted with a factor of 10

5 Conclusion

As seen from the report, different noise models affect the image in different ways. Gaussian noise creates a histogram that is a bell curve localized around two peaks. This visually creates intensities distributed around the peaks. Salt and Pepper noise is an impulse noise that produces black or white pixel intensity. The histogram for Salt and Pepper model does not change in shape but new level of black and white pixel intensities are introduced. Speckle noise is a flat distribution centred at 0, which produces a multiplicative value with high probability for selection within extents of the variance. As seen in the histogram, there is a more uniform and broader range of intensities.

Furthermore, de-noise techniques were applied in an attempt to recover noise contaminated images using the average and Gaussian filters. The 7x7 average applies a stronger blur than the 3x3 average - which in turn has the effect of eliminating notable noise and undoing fine image details. This is a direct consequence of taking more neighboring pixels into account when producing the average. Consequently, the 7x7 average had lesser PSNR than the 3x3 average. The Gaussian filter is shown to work the best with PSNR value of 28.37; it removes noise in the image while retaining contrast and color. As was depicted in the colormap, the Gaussian filter weights closer pixels higher than farther pixels. This induces a localized blur enough to rectify some of the noise while retaining edge detail and contrast.

The aforementioned methods were applied to a Salt & Pepper contaminated image to try and denoise the image. The average filter once again blurs out the image's desired edge detail. Alternatively, the Gaussian filter fails to completely reject noise; it leaves behind subtle remnants of the averaged impulse value. Applying a median filter to the noisy image is shown to have the best effect; It rectifies impulse type noise by finding the median value in the set of masked pixels. Hence Salt & Pepper noise, which contributes extreme intensities is removed as the median filer would consider that as an outlier value relative to the median.

Image sharpening is done using a high-boost filter, where an edge mask is formed by means of isolating high frequency elements of the image. This is a result of subtracting a Gaussian filtered (low pass filter) image from the source image, which leaves only high frequencies elements notable. Adding the edge mask to the image will increase the intensities of the high frequency components – effectively forming sharper edges. The method is useful due to its scalability; the edge mask can be multiplied by a factor to increase or decrease the sharpening effect. Using a factor greater than 1 increases the strength of sharpness and is known as high-boost filtering.

6 Appendix

6.1 Noise.m

```
clc;
clear;
%% Noise Generation
f = [0.3*ones(200,100) 0.7*ones(200,100)];

f_gaussian = imnoise(f, 'gaussian', 0, 0.01);
f_saltpepper = imnoise(f, 'salt & pepper', 0.05);
f_speckle = imnoise(f, 'speckle', 0.04);

PlotImageHist(f);
PlotImageHist(f_gaussian);
PlotImageHist(f_saltpepper);
PlotImageHist(f_speckle);


%% Noise Reduction
lena = double(rgb2gray(imread('lena.tiff')))./255;
PlotImageHist(lena);
%% Contaminate Image with Gaussian , variance 0.002
lena_gaussian = imnoise(lena, 'gaussian', 0.002);
PlotImageHist2(lena, lena_gaussian);
lena_gaussian_PSNR = imPSNR(lena, lena_gaussian);

% Denoise using averaging filter (3x3)
avg_filter3 = fspecial('average', [3 3]);

% Plot the filter
figure
imagesc(avg_filter3)
colormap(gray)

% Apply the filter
lena_denoised = imfilter(lena_gaussian, avg_filter3);
PlotImageHist2(lena, lena_denoised);
avg3PSNR = imPSNR(lena, lena_denoised);

% Denoise using averaging filter (7x7)
avg_filter7 = fspecial('average', [7 7]);
lena_denoised = imfilter(lena_gaussian, avg_filter7);
PlotImageHist2(lena, lena_denoised);
avg7PSNR = imPSNR(lena, lena_denoised);
```

```
% Denoise using gaussian with standard deviation 1
gaussian_filter = fspecial('gaussian', [7 7], 1);

% Plot the filter
figure
imagesc(gaussian_filter)
colormap(gray)

% Apply filter
lena_denoised = imfilter(lena_gaussian, gaussian_filter);
PlotImageHist2(lena,lena_denoised);
gaussianPSNR = imPSNR(lena, lena_denoised);

%% Contaminate using Salt & Pepper
lena_saltpepper = imnoise(lena, 'salt & pepper');
PlotImageHist(lena_saltpepper);

lena_denoised = imfilter(lena_saltpepper, avg_filter7);
PlotImageHist2(lena_saltpepper, lena_denoised);
avg7_sp_PSNR = imPSNR(lena, lena_denoised);

lena_denoised = imfilter(lena_saltpepper, gaussian_filter);
PlotImageHist2(lena_saltpepper, lena_denoised);
gaussian_sp_PSNR = imPSNR(lena, lena_denoised);

% Apply Median Filter
lena_denoised = medfilt2(lena_saltpepper);
PlotImageHist2(lena_saltpepper, lena_denoised);
median_sp_PSNR = imPSNR(lena, lena_denoised);
```

6.2 Sharpening.m

```
clear;
clc;
cameraman = im2double(imread('cameraman.tiff'));

% Becuase edges are main boundary of difference when applying a gaussian
% blur, subtracting the image leaves us with the edges! Now can simply use
% the edge mask to amplify edge details in the original image.
gaussian_filter = fspecial('gaussian', [7 7], 1);
cameraman_filtered = imfilter(cameraman, gaussian_filter);
cameraman_subtracted = cameraman - cameraman_filtered;
```

```
PlotImageHist2(cameraman_filtered, cameraman_subtracted);
% PlotImageHist(cameraman_filtered);
% PlotImageHist(cameraman_subtracted);

cameraman_sharpened = cameraman + cameraman_subtracted;
cameraman_sharpened5 = cameraman + 0.5 * cameraman_subtracted;

% PlotImageHist(cameraman_sharpened);
% PlotImageHist(cameraman_sharpened5);
PlotImageHist2(cameraman, cameraman_sharpened);
PlotImageHist2(cameraman_sharpened, cameraman_sharpened5);

PlotImageHist(cameraman + 10 * cameraman_subtracted);
```

6.3 PlotImageHist2.m

```
function figHandle = PlotImageHist2( imgA, imgB )
figHandle = figure;
subplot(2,2,1)
    imshow(imgA);
subplot(2,2,2)
    imhist(imgA);
subplot(2,2,3)
    imshow(imgB);
subplot(2,2,4)
    imhist(imgB);
end
```