**Lab 2: Image Enhancement**

****

Jonathan Tjong 20723414

William Li 20720929

**SYDE 575 - University of Waterloo**

Submitted To:Prof David Clausi

Due Date:Oct 1, 2021

# Introduction

In this second lab for SYDE 575, we explored different image enhancement techniques, such as applying spatial filters and masks. Using matlab and imnoise we learned about different noise generation techniques and used them to create sample noised images. Using different filtering and masking techniques we experimentally denoised the images and compared techniques using PSNR. Finally, we applied sharpening and edge enhancement techniques to an image using unsharp masking.

# Discrete Convolution for Image Processing

| **Original** | **h1 (⅙\*[1 1 1 1 1 1])** |
| --- | --- |
| **h2 (h1 transposed)** | **h3 ([-1 1])** |

**Q1. What did convolving the image with h1 do to the image? Looking at the impulse function, explain why convolving the image with h1 yields such results.**

Convolving the image with h1 smooths out the image horizontally by applying a 1-directional local averaging filter. Applying h1 (⅙\*[1 1 1 1 1 1]) as a filter means that the values of pixels are determined by averaging out the neighbouring pixels in the x direction. This results in blurriness and smoothing of vertical edges. For example, the straight hairs on Lena’s left side look very blurry in comparison to the original image or the image convolved with h2.

**Q2. What did convolving the image with h2 do to the image? Looking at the impulse function, explain why convolving the image with h2 yields such results.**

Convolving the image with h2 smooths out the image in one direction, however this time in the vertical direction. The h2 filter, which is the transpose of the h1 filter, is a 1-direction local averaging filter which determines pixel intensity by averaging out the neighbouring pixels in the y direction only. This results in blurriness and smoothing of horizontal edges. For example, Lena’s horizontal mouth line looks much blurrier in comparison to the original image or the image convolved with h1.

**Q3. What did convolving the image with h3 do to the image? Looking at the impulse function, explain why convolving the image with h3 yields such results.**

The h3 ([-1 1]) filter is a simple 1-directional first difference mask, which is capable of detecting edges across the x-direction (in other words, vertical edges). As a result, after convolving the image with h3, we are left with only white lines in the image, which are the vertical edges detected by the mask.

**Q4. Based on these results, what role can convolution perform in the context of image processing?**

In the context of image processing, convolution allows us to easily apply spatial filters to an input image and generate new transformed images. These filters can have a wide variety of purposes and outcomes, such as image smoothing and edge detecting.

# Noise Generation

Show all noise contaminated images

| Gaussian Noise | Salt & Pepper Noise | Speckle Noise |
| --- | --- | --- |
| Gaussian Histogram | Salt & Pepper Histogram | Speckle Histogram |

**Q5. Describe each of the histograms in the context of the corresponding noise models. Why do they appear that way?**

The gaussian histogram appears as two peaks centered on each of the original grey values. This is due to the fact that the noise generated is using a gaussian distribution with a mean of zero on each gret pixel. Therefore the end histogram result is two gaussian distributions on each original grey value.

The salt and pepper histogram shows undisturbed grey values for the majority of the image, with a few additions of black and white specks. The black and white specks are seen on the histogram as the small bars on the ends of each histogram. They are relatively small as the noise density for salt and pepper was 5%.

The speckle noise is a multiplicative noise based on the original pixel value and thus shows up as two different peaks of varying size.

**Q6. Are there visual differences between the noise contaminated images? What are they? Why?**

There are a few major differences. The gaussian noise looks like a smooth distribution of noise within each grey section of the image. The salt and pepper noise looks more disturbed, with a sprinkling of white and black pixels placed randomly on the image. The speckle noise looks more harsh.

**Q7. In the speckle noise case, what is the underlying distribution used? Can you tell from the histogram? How?**

The underlying distribution is still based on the original pixel value, as the noise is only generated in the surrounding intensities from the original value. The distribution is a randomly generated scalar of the image value added back to the original image.

**Q8. In the speckle noise case, you will notice that the peaks of the histogram are no longer of the same height as they were in the original image. Also, the spread around each of the peaks is also different from each other. Why? Hint: Noise is multiplicative.**

This is due to the fact that the speckle noise is generated using the equation J = I + n\*I, where n is a randomly generated scalar from mean 0 with a specified variance. In this case, the variance was 0.04. For the lower intensity grey, the noise was generated using a 4% deviation from the original grey level. Compared to the higher intensity grey, the low intensity has a smaller range of values and thus the peak is higher and skinnier. The higher intensity grey value has a greater range within the same 4% deviation. The result is a wider but shorter peak of noise generated.

# Noise Reduction in the Spatial Domain

| Zero Mean Gaussian Noise (0.002 Variance) | PSNR = 26.9914 |
| --- | --- |

|  |
| --- |

| Average 3x3 Filter Lena | PSNR = 30.6448 |
| --- | --- |

**Q9. Compare the visual difference between the noisy image and the denoised image. How well did it work? Why? Did the PSNR decrease?**

The 3x3 averaging filter worked reasonably well on the gaussian noised image. Since the image has mostly smooth and uniform sections, the averaging filter is able to remove much of the generated noise on these sections. The 3x3 averaging filter was able to increase the PSNR of the restored image from 26 to 30.

**Q10. Compare the histograms of the noise-free, noisy, and denoised images. What happened? Why?**

From the original image to the noisy image, the histogram showed signs of information loss by blurring and smoothing the individual peaks in the histogram. This occurred when the gaussian noise was applied as it affects each individual pixel according to its original value. The end result is that you’d see a gaussian distribution centered around each value depicted on the original histogram. In this case, it ends up smoothing the entire histogram.

From the noisy image to the denoised image, you can see the denoised histogram more closely relates to the original histogram. The averaging filter was able to recreate most of the original image by averaging values from each pixel’s surrounding neighbors. This allowed the histogram to regain most of its original shape as the image intensity distribution was averaged back to normal.

**Q11. Based on visual quality of the denoised image, what are the benefits and drawbacks associated with the average filter?**

Some benefits are that the averaging filter is less computationally heavy, and is relatively easy to calculate and implement. It is also generally useful in smoothing and reducing noise. However, since it is good at removing noise by averaging the surrounding values, this also means that it blurs the sharp contrasts in the image.

| Average 7x7 Filter Lena | PSNR = 26.2514 |
| --- | --- |

**Q12. Compare the visual difference between the denoised image from the 7x7 filtering kernel and the denoised image from the 3x3 filtering kernel. Are there any differences? Why? Did the PSNR decrease? Why?**

The 7x7 filtered image looks much smoother, as the averaging was computed over a larger mask. However, this also means the image looks much blurrier overall. As the lines and contrasts in the original image were all averaged out. The 3x3 filtered image also retains a similar smoothed quality, but looks less blurry than the 7x7 filter. This was because the 3x3 filter used the same averaging principle but over a much smaller mask about 5 times smaller. In terms of PSNR, the 7x7 filter has a much lower PSNR value compared to the 3x3 filter. The PSNR dropped from 30 to 26. In fact, compared to the noisy image, the PSNR still dropped from 27 to 26. This means the 7x7 averaging mask was too strong and ended up degrading the image further from the original.

**Q13. Compare the histograms of the two denoised images. What are the differences? Why?**

The 7x7 mask has much more pronounced peaks in the histogram compared to the 3x3 mask. Because more averaging was done using the 7x7 mask, the histogram was able to clearly pick out the individual peaks and more closely resemble the original histogram.

**Q14. Based on visual quality of the denoised image, what are the benefits and drawbacks associated with using a larger window size**

The larger window size is better at removing noise, as can be seen in the visual comparison between the 7x7 mask and the 3x3 mask. The 7x7 averaging mask produced cleaner uniform surfaces and completely eliminated all traces of noise. However, the drawback of the larger window size is that the image overall loses many of its edges and contours as the whole image becomes much more blurry.

|  |
| --- |

| Gaussian 7x7 Filter Lena | PSNR = 30.8518 |
| --- | --- |

**Q15. Compare the visual difference between the denoised image from the Gaussian filtering kernel and the denoised images from the averaging filter kernels. Are there any differences? Why? Did the PSNR decrease? Why?**

The gaussian filtering mask performed much better visually compared to the averaging filters. The gaussian filtered image has the same smoothing quality that removes noise, but it does not blur the images to the same degree that the averaging filters do. This is because the gaussian filter is a weighted average with more emphasis on closer neighboring pixels.

**Q16. Compare the histograms of the denoised image using the Gaussian filtering kernel and the denoised images from the averaging filter kernels. What are the differences? Why?**

The gaussian filter is able to recapture most of the original histogram shape with most of the original peaks present in the restored histogram. Compared to the histogram from the averaging filters, the gaussian peaks are less pronounced and it still retains some of the noise in the valleys of the histogram. This could be because the averaging effect of the gaussian is smaller, so it has difficulty removing all the noise in the high contrast areas as it won’t effectively smooth out the noise in those areas.

**Q17. Based on visual quality of the denoised image, what are the benefits and drawbacks associated with using a Gaussian kernel as opposed to an averaging kernel?**

The benefits of using a gaussian filter is that it will preserve edges and high contrast areas of the image more clearly. The gaussian filter will also still be able to smooth out most noise as it still averages the values from the neighboring pixels. Compared to the averaging filter, a drawback of the gaussian filter is that the averaging effect will be lessened over a greater area. The averaging mask will have a stronger averaging effect which could be more beneficial to images with large uniform patches

| Salt & Pepper Noisy Lena | Salt & Pepper Histogram  PSNR: 18.4470 |
| --- | --- |

| Gaussian 7x7 Filter Lena on S&P | PSNR: 27.1533 |
| --- | --- |

| Average 7x7 Filter Lena on S&P | PSNR:25.4949 |
| --- | --- |

**Q18. How does the averaging filter and Gaussian filtering methods perform on the noisy image in terms of noise reduction? Explain in terms of visual quality as well as PSNR. Why do we get such results?**

The averaging filter and gaussian filtering methods produce different results visually. Once again, the gaussian filter seems to preserve the edges and sharp contrast components of the image. However, the salt and pepper noise is still somewhat evident in the image.

On the other hand, the averaging filter produces a more blurry and smoothed out image overall. Although we lose sharpness, the blurriness visually removes the salt and pepper noise to an extent.

In terms of PSNR, we see that the gaussian filter produced a slightly higher quality image with a PSNR of 27.15, while the averaging filter had a PSNR of 25.5. This is because although the spottiness of the salt and pepper noise is a little more visible, the preserved sharpness of the image makes it higher quality.

**Q19. Compare the histograms of the denoised images with that of the noisy image. What characteristics are present in all of the histograms? Why?**

In both the histograms of the denoised images and the histogram of the noisy image, we can see the peaks where light intensity is most frequent remains consistent. In other words, the overall shape of the histogram is the same throughout the histograms (ignoring the noise values at 0 and 255 in the noisy histogram). This is because salt and pepper noise adds extreme intensity values at both ends of the histogram, but does not corrupt the overall shape of the rest of the histogram. The frequency patterns throughout the rest of the image are preserved. After applying filters to the image to remove the noise, we still see the same frequency patterns in the histogram. The spatial smoothing filters are not destructive and for the most part preserve intensity levels throughout the image.

| Median Filter Lena on S&P | PSNR: 34.4212 |
| --- | --- |

**Q20. How does the denoised image produced using the median filter compare with the denoised images produced using averaging filter (7x7) and Gaussian filtering methods? Explain in terms of visual quality as well as PSNR. Why do we get such results with a median filter when compared to the other spatial filtering methods?**

Visually we can observe that the median filter was able to denoise the image very well and performed better than the gaussian filter and the averaging filter. The image looks very similar to the image, with high sharpness and contrast, and with seemingly no trace of the salt and pepper noise. The PSNR was very high with a value of 34.42, significantly higher than the PSNR values using the other two spatial filtering methods.

This high performance is because median filters are extraordinarily effective at denoising outliers, such as the extreme black and white pixels introduced in salt and pepper noise. This is because median filters replace the noisy outlier pixel values with a median value within its neighbours, forcing it to be like one of its neighbours without averaging. This method also produces very little blurring in comparison to averaging filters.

# Sharpening in the Spatial Domain

| **Gaussian filtered Cameraman image** | **Subtracted image (original image - gaussian filtered image)** |
| --- | --- |

**Q21. What does the subtracted image look like? What frequency components from the original image are preserved in the subtracted image? Why?**

The subtracted image looks completely dark, except for the presence of white lines which are the sharp edges of the original image. We can see the outline of the cameraman as well as the outlines of the camera and tripod.

We can say that the high frequency components of the original image are preserved in the subtracted image. High frequency components of an image are the parts of an image that change rapidly from one intensity to another, or in other words, the edges.

The effect that we see here occurs since the gaussian filter smooths out the original image, essentially removing the sharp edges, and then when we subtract the original image with the filtered image we are left with those removed edges only.

| **Addition of original image and subtracted image** |
| --- |

**Q22. What does the resulting image look like? How does it differ from the original image? Explain why it appears this way.**

The resulting image looks very crisp and sharp, with clearer edges and finer detail as a result. In comparison to the original image, the tripod looks shinier and the image looks brighter overall. As well, looking at the back of the man's head and body, we can see a bright white outline, which causes higher contrast at the edge, “boosting” it.

This is because we are using unsharp masking, which is an edge enhancement method. By adding back the subtracted image (an unsharp mask), we are able to create a sharpened image by increasing the contrast of edges even further.

| Addition of original image and 0.5 x subtracted image |
| --- |

**Q23. Compare the results produced by adding the subtracted image to the original image and that produced by adding half of the subtracted image to the original image. How does it differ? Explain why it appears this way.**

The high contrast effect that occurs at the edges in the original image plus the subtracted image is also present when only half the subtracted image is added, but the effect is not seen to quite the same degree. The edges are not quite as sharp and we also don’t see the white outline seen before as much here. This is because the intensity of the subtracted image is cut in half before being added to the original image, attenuating the boosting effect at the edges that unsharp masking produces.

**Q24. What does multiplying the subtracted image by a factor less than one accomplish? What about greater than one?**

Multiplying the subtracted image by a factor less than one is a way to accomplish unsharp masking if not a high degree of edge enhancement is desired. If only subtle edge enhancement is needed, we can tune the unsharp masking by multiplying the subtracted image by a factor less than one.

On the other hand, if we want even stronger edge enhancement and we want to exaggerate its effects, we can multiply the subtracted image by a factor greater than one. This is a method called high boost filtering and can provide even higher contrast at the edges.

# Conclusion

In this lab, we used matlab to explore fundamental image enhancement techniques.

First, we convolved an image with various impulse functions to produce smoothed output images in both x and y directions, as well as to produce an image consisting only of the original image’s vertical edges. We learned that discrete convolution is very powerful and important in the context of image processing.

Next, we generated three different types of noise onto a synthetic toy image. The three noise models were 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). We plotted the resulting noisy images and histograms, and observed their visual differences.

Third, we explored various spatial filtering techniques to reduce noise and enhance images. We used a 3x3 averaging filter, a 7x7 averaging filter, and a 7x7 gaussian filter to denoise an image with gaussian noise. All three methods were effective at restoring the image somewhat and reducing the noise, and varied in performance. The 7x7 averaging filter seemed to over blur the image and produce too much smoothing. The gaussian filter performed better than the basic averaging filters due to its more intelligent weighting distribution.

We also generated salt and pepper noise on the image and tried using an averaging filter, gaussian filter, and a median filter to denoise the image. It was evident that a median filter was best suited for denoising salt and pepper noise due to its capability to deal with outlier pixels.

Finally, we obtained hands-on experience with edge enhancement in an image using unsharp masking. We also saw that we could multiply the subtracted image used by a factor greater or lesser than one to tune the intensity of the edge enhancement.

# 

# Appendix

Part 2 code:

close all

lena = rgb2gray(imread('lena.tiff'));

% Normalize lena

lena = double(lena)/255;

figure

imshow(lena)

h1 = (1/6)\*ones(1,6);

h2 = h1';

h3 = [-1 1];

lena\_h1 = conv2(lena, h1);

figure

imshow(lena\_h1);

lena\_h2 = conv2(lena, h2);

figure

imshow(lena\_h2);

lena\_h3 = conv2(lena, h3);

figure

imshow(lena\_h3);

Part 3 code:

close all

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

figure

imshow(f);

figure

plot(imhist(f));

title('Histogram of base image');

f\_gaussian = imnoise(f, 'gaussian');

figure

imshow(f\_gaussian);

figure

plot(imhist(f\_gaussian));

title('Gaussian noise histogram');

f\_salt\_pepper = imnoise(f, 'salt & pepper');

figure

imshow(f\_salt\_pepper);

figure

plot(imhist(f\_salt\_pepper));

title('Salt & pepper noise histogram');

f\_speckle = imnoise(f, 'speckle', 0.04);

figure

imshow(f\_speckle);

figure

plot(imhist(f\_speckle));

title('Speckle noise histogram');

Part 4 code (was edited and used multiple times for different filters):

close all

lena = rgb2gray(imread('lena.tiff'));

% Normalize lena

lena = double(lena)/255;

% figure

% imshow(lena);

% figure

% plot(imhist(lena));

% title('Original lena histogram');

% Add noise

lena\_noisy = imnoise(lena, 'salt & pepper');

% figure

% imshow(lena\_noisy);

% figure

% plot(imhist(lena\_noisy));

% title('Noisy lena histogram with Salt & Pepper');

% h = fspecial('average', [7 7]);

% filtered\_lena = imfilter(lena\_noisy, h);

% figure

% imshow(filtered\_lena);

% figure

% plot(imhist(filtered\_lena));

% title('Filtered lena histogram using 7x7 Averaging on Salt&Pepper');

%

% psnr\_original\_filtered = 10\*log10(1/mean2((lena-filtered\_lena).^ 2));

filtered\_lena = medfilt2(lena\_noisy);

figure

imshow(filtered\_lena);

figure

plot(imhist(filtered\_lena));

title('Filtered lena histogram using median filter on Salt&Pepper');

psnr\_original\_filtered = 10\*log10(1/mean2((lena-filtered\_lena).^ 2));

Part 5 code:

close all

cam = double(imread('cameraman.tif'))/255;

smoothed\_cam = imfilter(cam, fspecial('gaussian', [7 7], 1));

figure

imshow(smoothed\_cam);

subtracted\_cam = cam - smoothed\_cam;

figure

imshow(subtracted\_cam);

enhanced\_cam = cam + subtracted\_cam;

figure

imshow(enhanced\_cam);

enhanced\_cam\_half\_effect = cam + 0.5 \* subtracted\_cam;

figure

imshow(enhanced\_cam\_half\_effect);