**Lab 4: Restoration**

****

Jonathan Tjong 20723414

William Li 20720929

**SYDE 575 - University of Waterloo**

Submitted To:Prof David Clausi

Due Date:Nov 12, 2021

# Introduction

In Lab 4, we explored some more about image restoration in the frequency domain, and then we got hands on experience with adaptive filtering using a Lee filter. Both these parts were done using MATLAB.

Firstly, we blurred the cameraman.tif image using a blur function in the frequency domain. We then experimented with restoring the image using inverse filtering, both when the blurred image had noise or no noise. Afterwards, we applied Wiener filtering to the same noisy blurred image and observed the difference in the two restoration techniques.

Next, we applied a Lee filter to a degraded image. In order to do this, we had to determine an estimate of the noise variance in the image, and we also had to calculate local means and variances (using pixel neighborhoods) throughout the image. A Lee filter is an adaptive filter, which means that it adapts to local detail in an image and changes its degree of smoothing in different parts of the image. Studying the results from the Lee filtering gave us more insight on how adaptive filtering works and operates.

# 

# Image Restoration in the Frequency Domain

| **Blurred image:**    PSNR: 21.0957 | **Restored image using inverse filter:**    PSNR: 251.8595 |
| --- | --- |

**1. Compare the restored image with the original image and the blurred image. How does the restored image and the PSNR differ from the blurred image? Is it better or worse? Why?**

The restored image is identical to the original image indicated by the comparatively high value of 251 compared to the PSNR of the blurred image. The result is perfect due to the use of an inverse filter with no additive noise present. Due to convolution theorem the blurred image was simply the product of the original image and the blurring filter. The inverse filter takes the blurred image and divides the blurring filter, resulting in the original image.

| **Blurred image with Gaussian noise** | **Blurred image with Gaussian noise, restored using inverse filtering:**    PSNR: -38.1974 |
| --- | --- |

**2. Compare the restored image with the restored image from the previous step. How does the restored image and the PSNR differ from the previous restored image? Is it better or worse? Why?**

Now the restored image is completely unrecognizable. The PSNR is much worse with a score of -38.1974. Noise predominates the entire image due to the use of an inverse filter with additive noise. The noise component is also divided by the filter, which when small, can amplify the noise at every spatial location.

**3. Can you draw any conclusions about inverse filtering when applied to noise degraded image?**

When applying the inverse filter to a noise degraded image the additive noise can actually be magnified. Due to the additive noise being divided by the filter, if the filter is small, the noise will be magnified at every location in the spatial domain. Thus, it can be concluded that inverse filtering is not practical when applied to noise degraded images.

**Blurred image with Gaussian noise, restored using Wiener filtering:**

****

PSNR\_wiener\_rest\_w\_noise : 21.5221

**4. Compare the restored image with the restored image from the previous step. How does the restored image and the PSNR differ from the previous restored image? Is it better or worse? Why? Explain it in context with the concept behind Wiener filtering.**

The Wiener filtered restored image is much better compared to the inverse filtered gaussian noised image. The PSNR of 21.5221 is higher than the original blur and it remains recognizable. The Wiener filter is suited and designed to invert frequency blurring as well as smooth additive noise. Using knowledge of the blurring filter and the noise-to-signal power ratio of the additive noise, the Wiener filter will unblur the frequency blurring as well as smooth the additive noise.

**5. Can you draw any conclusions about Wiener filtering when applied to noise degraded image?**

Since the Wiener filter is designed to invert frequency blurring as well as smooth additive noise, it is well suited to handling noise degraded images. The Wiener filter minimizes the overall mean-squared-error of the image and produces a linear estimation of the original image. The Wiener filter essentially acts as a bandpass filter, where the highpass is the inverse filter and the lowpass filter is due to the noise-to-signal power ratio. If the nsr is set to 0, the Wiener filter acts the same as an inverse filter.

# 

# Adaptive Filtering

| **Original cameraman image:** | **Degraded image:** |
| --- | --- |

**6. Use a flat region of degraded.tif and estimate the variance of the noise (getrect may be helpful here).**

The flat region of the image we used to estimate noise variance was the cameraman’s dark coat. Capturing a small rectangle of the coat using getrect(), the resulting noise variance using matlab’s var() function was 0.0044.

| **Denoised image using Lee filter (using noise variance of 0.0044):**    PSNR: 20.9003 | **Denoised image using Gaussian low pass filter (std. dev. of 30):**    PSNR: 22.1124 |
| --- | --- |

**7. Compare this result to that using a Gaussian low pass filter with standard deviation of 30 (in the frequency domain). Note the performance in areas of high and low detail.**

Comparing the result of the Lee Filter and the Gaussian low pass filter, it looks like the result of the Lee filter overall has less contrast and is more faded. However, it also looks a little sharper in areas of high detail. For example, the creases and edges at the cameraman’s legs and right arm look sharper, as well as the camera tripod legs. This is because for a Lee filter, as local image variance increases and becomes larger relative to noise variance, less smoothing is performed and detail is preserved.

On the other hand, the grass seems to be smoothed out a lot more by the Lee filter in comparison to the Gaussian filter. It’s likely because this area has relatively low local variance. Obviously, the Gaussian filter is not adaptive and applies its smoothing throughout all areas of the image. Overall, the Gaussian filter does a good job of denoising the image and achieves a higher PSNR.

| **Lee filter using noise variance of 0.002:**    PSNR: 19.8909 | **Lee filter using noise variance of 0.01:**    PSNR: 21.0031 |
| --- | --- |

**8. Try varying your estimate of the noise variance both above and below the value you got from your flat region. How does this change the filter’s results? Why?**

When using a low estimate of the noise variance, we see that local variance throughout the image is considered higher, resulting in less smoothing overall. Edges and finer detail are preserved more, and we get a sharper image with more contrast. However, a downside to this is that the noise also gets preserved more. The image is visibly noisier when using a low estimate.

When using a high estimate of the noise variance, the opposite happens. The image is faded and has much less contrast, with more smoothing overall. As a result, the noise is less visible. When a higher estimate of the noise variance is used, local image variance becomes relatively lower, resulting in more smoothing being applied by the Lee filter.

| **Lee filter using neighbourhood of 3 x 3:**    PSNR: NaN | **Lee filter using neighbourhood of 10 x 10:**    PSNR: 20.7206 | **Lee filter using neighbourhood of 20 x 20:**    PSNR: 20.3372 |
| --- | --- | --- |

Note: These three images were denoised using a noise variance of 0.0044 for the Lee filter.

**9. Try changing the size of the filter neighborhood to be smaller and larger than 5x5. How does this change the results? Why?**

When a smaller filter neighborhood is used, less contrast is seen in the image. Overall, the image appears to be of lesser quality and smoother. I think this is because when using such a small window for the filter, the local variance and mean are not accurately capturing the variance and mean of a region of an image. For example, an area of the image could contain lots of detail, such as the camera, but when we zoom too far in and look at too miniscule of a neighborhood the detail is not captured. Thinking about how small a 3 x 3 pixel neighborhood is, it could likely be just the same value in all elements of the neighborhood.

When a larger filter neighborhood is used, the image appears to be sharper and have more contrast. I think this is because now we have a large enough neighborhood to accurately capture the true variance and mean of a region of the image, i.e. the grassy field with its shadows. This means that the Lee filter can do a better job of adapting to different areas of the image and has better judgement of when to smooth the image and when to preserve the image. Overall, the image is of much better quality in comparison to when a very small filter neighborhood is used.

# 

# Conclusion

In conclusion, after experimenting with blurring filters and applying inverse and Wiener filters, we supported our understanding of what each filter does. Inverse filters were adept at undoing blurs and recovering images but performed poorly in the presence of additive noise. Wiener filters were the best of both worlds and were able to invert blurring and smooth additive noise.

Afterwards, we explored and learned about adaptive filtering. Using MATLAB we coded a Lee filter, a form of adaptive filter. This required calculating an estimate of the noise variance in the image, which was done by computing the variance of a “flat” portion of the image. We also had to calculate local means and variances using pixel neighborhoods throughout the image. Looking at the results of filtering, we were able to study and observe how the Lee filter is able to adapt to the detail of an image, and apply varying degrees of smoothing over different parts of the image. When local image variance is high relative to noise variance, the image is preserved and little smoothing is applied. When local image variance is low relative to noise variance, the local mean is used to apply smoothing.

We also observed that when a low noise variance estimate is used for the Lee filter, edges and finer detail are preserved more, and we get a sharper image with more contrast. When a high noise variance estimate is used, we see that more smoothing overall happens and there is less contrast in the image. As well, when a very small neighborhood is used for local variance and local mean for the Lee filter, the image quality decreases and is over smoothed. On the other hand, when a larger neighborhood is used, the image is sharper, has more detail, and is of better quality overall. This is likely because the neighbourhood is large enough to accurately capture the true variance and mean of a region of the image.

# 

# Appendix

**Part 2 Code:**

close all;

h\_d = fspecial('disk', 4);

h = zeros(256,256);

h(1:9,1:9) = h\_d;

h = circshift(h, [-5,-5]);

% figure

% imshow(h, []);

f = im2double(imread('cameraman.tif'));

h\_freq = fft2(h);

figure

imshow((abs(h\_freq)), []);

f\_blur = real(ifft2(h\_freq.\*fft2(f)));

figure

imshow(f\_blur, []);

psnr\_blurred = psnr(f\_blur, f);

% Now apply inverse filtering to it by dividing the image by the blurring function hfreq

f\_restored = ifft2(fft2(f\_blur)./h\_freq);

figure

imshow(f\_restored, []);

psnr\_restored = psnr(f\_restored, f);

% Now add zero-mean Gaussian noise with a variance of 0.002 to the blurred image.

% Apply inverse filteringto the noisy blurred image. Plot the restored image and the PSNR

gaussian\_filter = fspecial('gaussian', 256, 0.002);

f\_blur\_with\_noise = imnoise(f\_blur, 'gaussian', 0, 0.002);

figure

imshow(f\_blur\_with\_noise, []);

f\_restored\_with\_noise = ifft2(fft2(f\_blur\_with\_noise)./h\_freq);

figure

imshow(f\_restored\_with\_noise, []);

psnr\_restored\_with\_noise = psnr(f\_restored\_with\_noise, f);

% Wiener filter

wiener\_filtered = ifftshift(deconvwnr(f\_blur\_with\_noise, h, 0.002/var(f(:))));

figure

imshow(wiener\_filtered, []);

psnr\_restored\_wiener\_filter = psnr(wiener\_filtered, f);

**Part 3 Code:**

close all;

image = im2double(imread('degraded.tif'));

figure

imshow(image, []);

% 6. getrect, calculating noise\_var

% rect = getrect;

% xmin = rect(1);

% ymin = rect(2);

% xmax = rect(3) + xmin;

% ymax = rect(4) + ymin;

% subimage = image(ymin:ymax, xmin:xmax);

% noise\_var = var(subimage(:)); %noise\_var = 0.0044

noise\_var = 0.0044;

% Lee filter

original\_image = im2double(imread('cameraman.tif'));

figure

imshow(original\_image, []);

% Change for question 9, default should be [5, 5]

neighbourhood = [10 10];

local\_mean = colfilt(image, neighbourhood, 'sliding', @mean);

local\_var = colfilt(image, neighbourhood, 'sliding', @var);

K = (local\_var-noise\_var)./local\_var;

Lee\_filtered = K.\*image + (1-K).\*local\_mean;

figure

imshow(Lee\_filtered, []);

psnr\_lee\_filtered = psnr(Lee\_filtered, original\_image);

% 7. comparison to Gaussian filter

g = fspecial('gaussian', 256, 30);

m = max(g(:));

g = g ./ max(g(:));

%imshow(g, []);

degraded\_spectra = fftshift(fft2(image));

gaussian\_filtered = ifft2(ifftshift(degraded\_spectra.\*g));

figure

imshow(abs(gaussian\_filtered), []);

psnr\_gaussian\_filtered = psnr(abs(gaussian\_filtered), original\_image);