COL 783

Assignment 2

Rajbir Malik
2017CS10416

September 25, 2019

Denoising and Deblurring

Overview

In this assignment, we were asked to understand and try various operations to recover an image after having been corrupted by nose and blurring.
We progressively managed to get better restoration by first trying fixing with basic **mean** and **median filters**, which are innately unaware of the features in the image.
Then we moved onto intelligent restorations process, in this case **edge preserving denoisers**.
I chose to implement **Total variation denoising (Rudin et al., 1992)**.
Then we referred to *Weiner Filtering* for *deblurring* of various images. We tried with *different noise levels* and *different original image estimates*.
Finally, we put things to practice, working with real life images and trying to bring them to closest possible restoration.
The assignment was divided in 4 different sections, each of which are discussed separately, as followed.

Basic Denoising

Here we were asked to add noise to the images provided and then try to denoise with various filters. `Gaussian` and `Salt/Pepper` noises were used and restoration filters in check were the **mean** and the **median filter**.

Following images were provided.

Figure 1: Barbara

Figure 2: Boat

Figure 3: Sandiego

Figure 4: Shepplogan

Barbara (Gaussian)

Figure 5: Gaussian Noise

Figure 6: Best PSNR image (Mean)

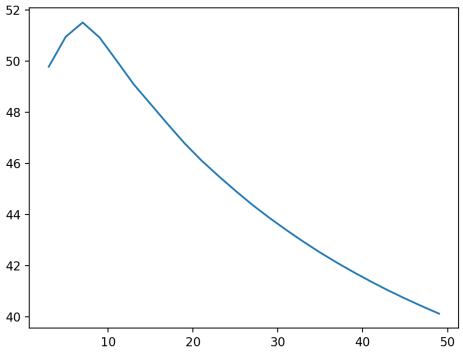

Figure 7: PSNR vs filter size (Mean)

Figure 8: Best PSNR image (Median)

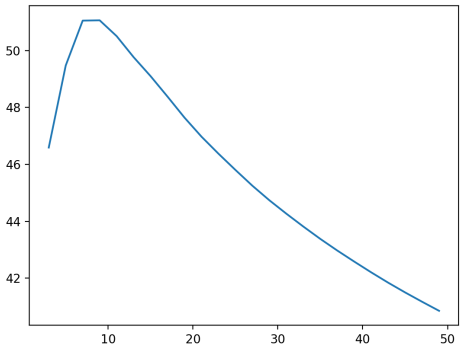

Figure 9: PSNR vs filter size (Median)

Barbara (Salt/Pepper)

Figure 10: Salt/Pepper Noise

Figure 11: Best PSNR image (Mean)

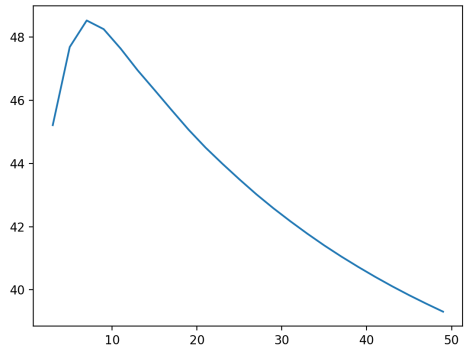

Figure 12: PSNR vs filter size (Mean)

Figure 13: Best PSNR image (Median)

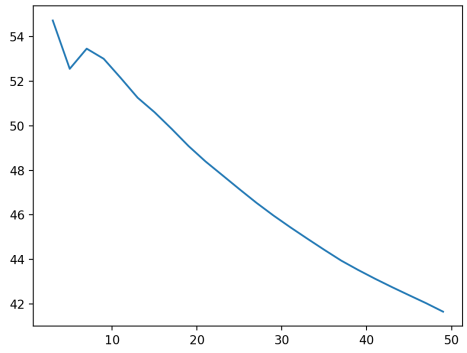

Figure 14: PSNR vs filter size (Median)

Boat (Gaussian)

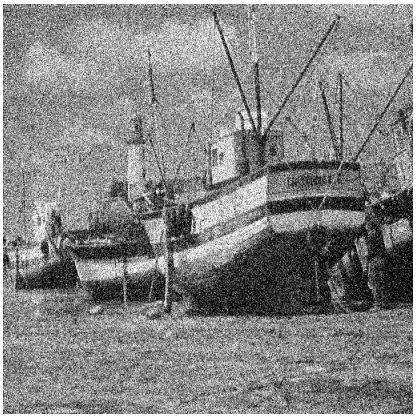

Figure 15: Gaussian Noise

Figure 16: Best PSNR image (Mean)

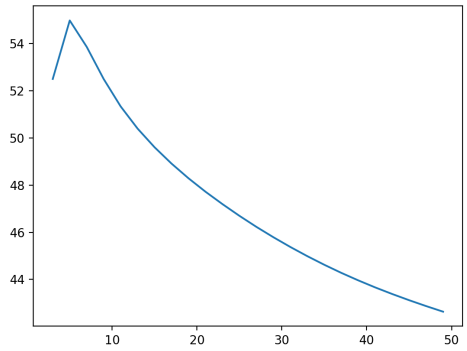

Figure 17: PSNR vs filter size (Mean)

Figure 18: Best PSNR image (Median)

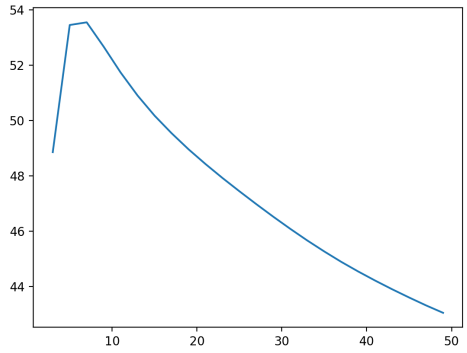

Figure 19: PSNR vs filter size (Median)

Boat (Salt/Pepper)

Figure 20: Salt/Pepper Noise

Figure 21: Best PSNR image (Mean)

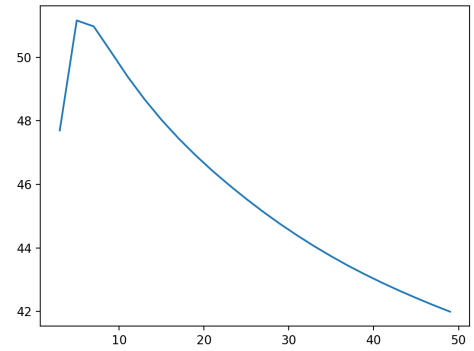

Figure 22: PSNR vs filter size (Mean)

Figure 23: Best PSNR image (Median)

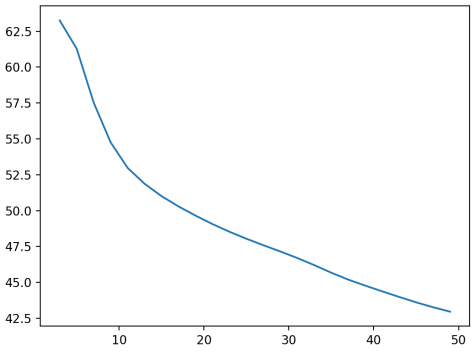

Figure 24: PSNR vs filter size (Median)

Sandiego (Gaussian)

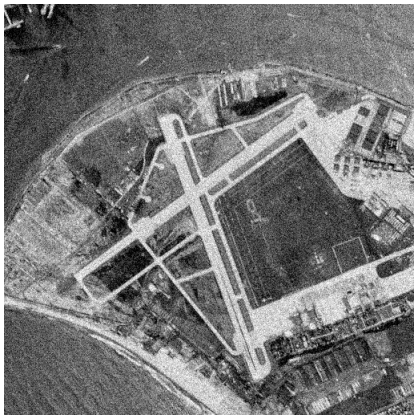

Figure 25: Gaussian Noise

Figure 26: Best PSNR image (Mean)

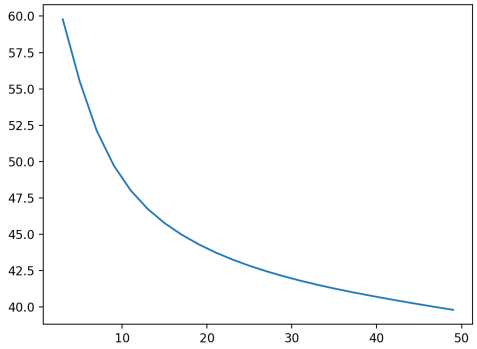

Figure 27: PSNR vs filter size (Mean)

Figure 28: Best PSNR image (Median)

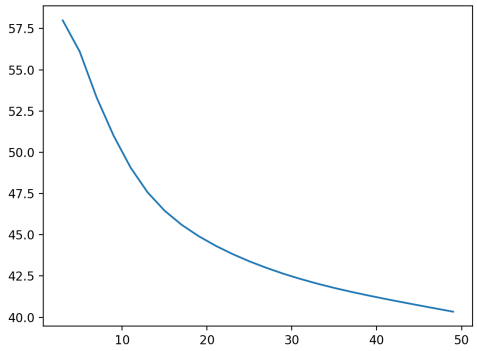

Figure 29: PSNR vs filter size (Median)

Sandiego (Salt/Pepper)

Figure 30: Salt/Pepper Noise

Figure 31: Best PSNR image (Mean)

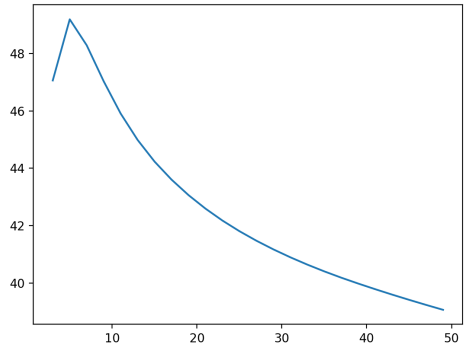

Figure 32: PSNR vs filter size (Mean)

Figure 33: Best PSNR image (Median)

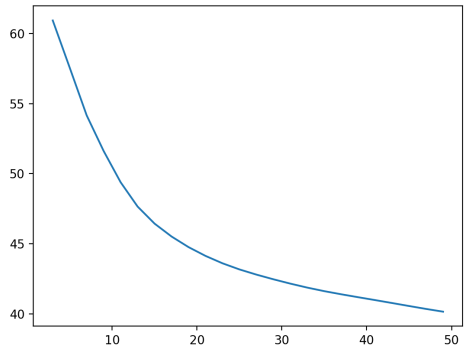

Figure 34: PSNR vs filter size (Median)

Shepplogan (Gaussian)

Figure 35: Gaussian Noise

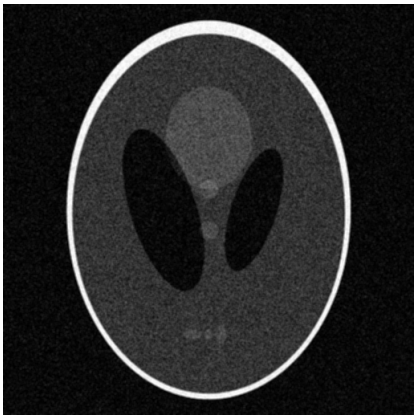

Figure 36: Best PSNR image (Mean)

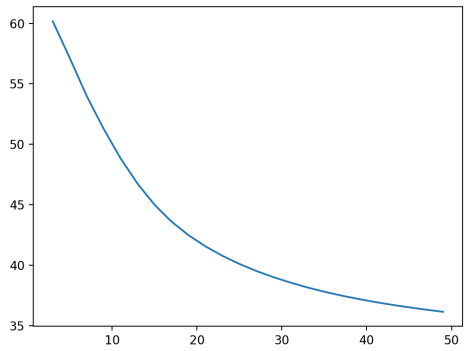

Figure 37: PSNR vs filter size (Mean)

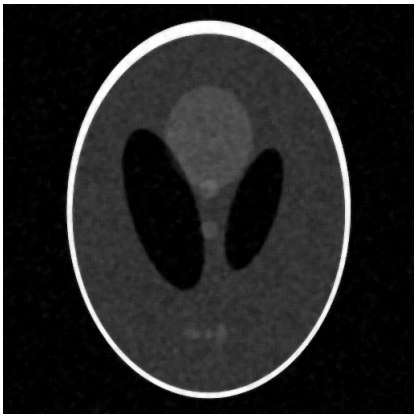

Figure 38: Best PSNR image (Median)

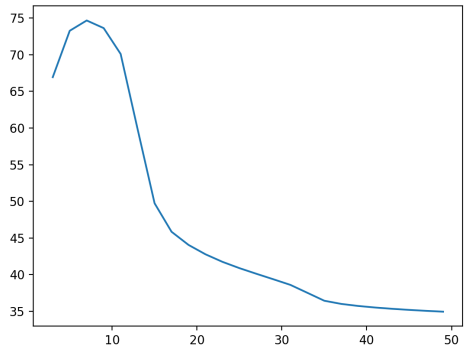

Figure 39: PSNR vs filter size (Median)

Shepplogan (Salt/Pepper)

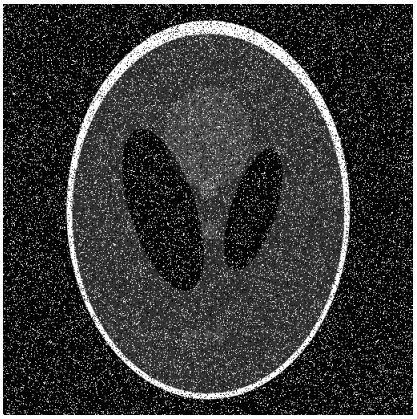

Figure 40: Salt/Pepper Noise

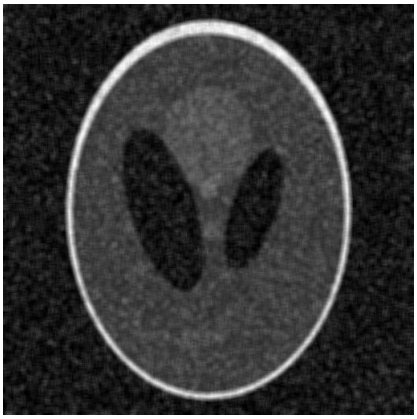

Figure 41: Best PSNR image (Mean)

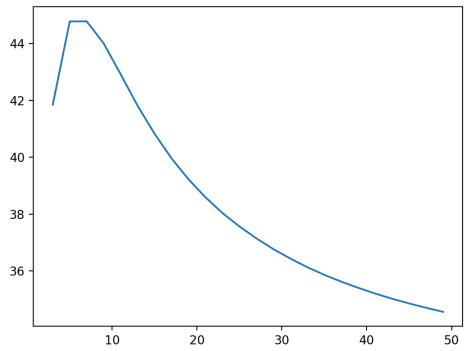

Figure 42: PSNR vs filter size (Mean)

Figure 43: Best PSNR image (Median)

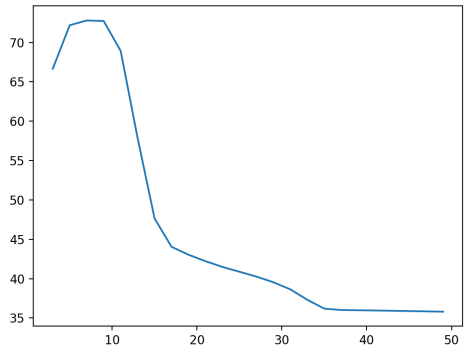

Figure 44: PSNR vs filter size (Median)

Edge-Perserving Smoothing

I implemented the *total variation denoising* and used it on the images from previous discussion. Also, followed are results from an example implementation of total variation.

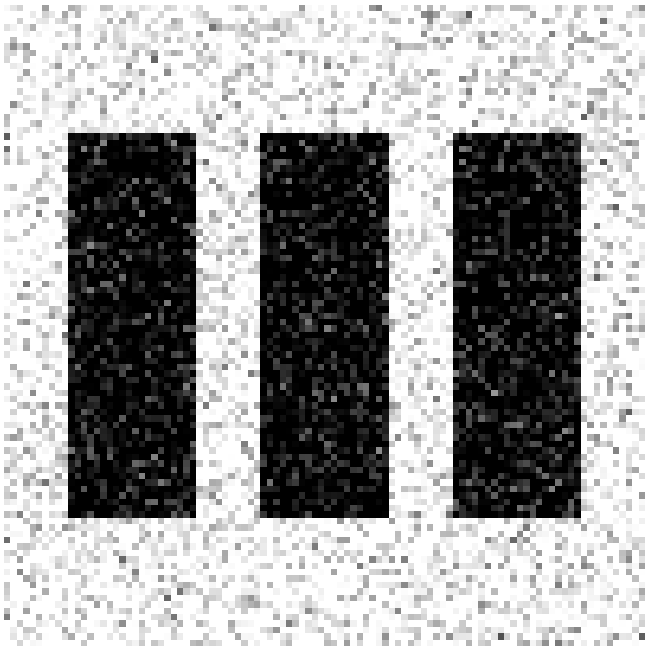

Figure 45: Noisy Image

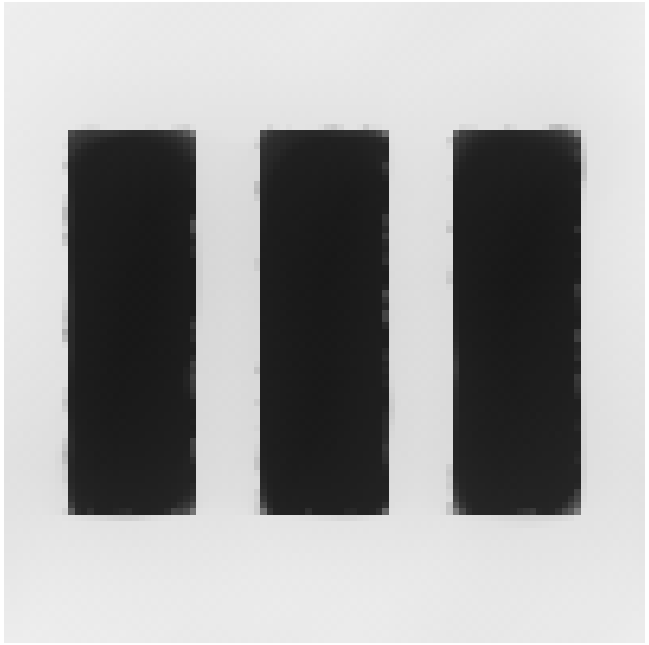

Figure 46: Recovered Image

Gaussian Noise Images

Here are the results of application of *total variation filtering* on *gaussian noise images* from before.

Figure 47: Barbara

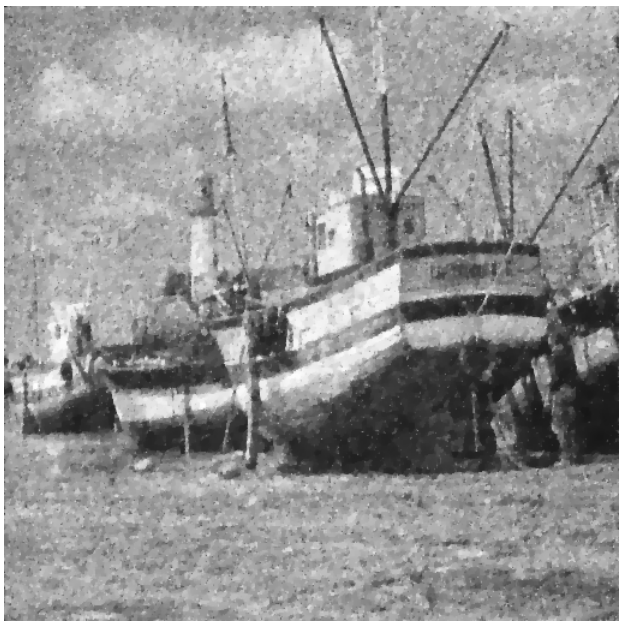

Figure 48: Boat

Figure 49: Sandiego

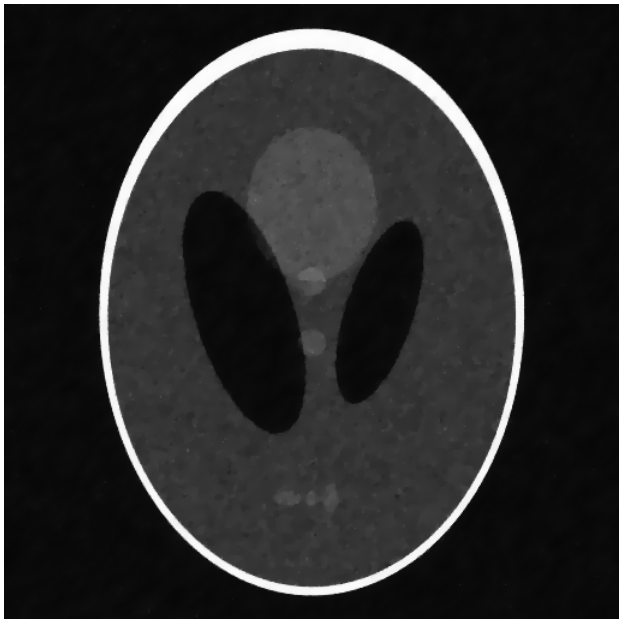

Figure 50: Shepplogan

Salt/Pepper Noise Images

Here are the results of application of *total variation filtering* on *salt/pepper noise images* from before.

Figure 51: Barbara

Figure 52: Boat

Figure 53: Sandiego

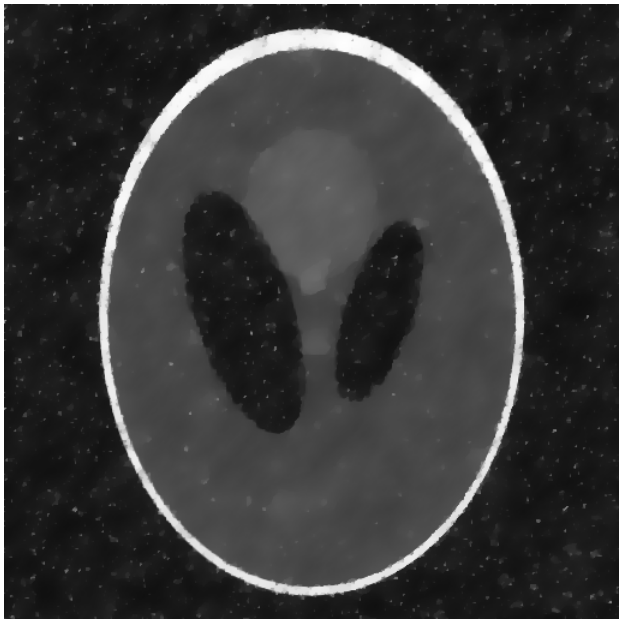

Figure 54: Shepplogan

Deblurring

Variation in Noise

In this section, I focused on implementing the **Wiener Filtering Procedure** and applied on a set of *blurred images*, with *different noise variations*. The results are presented below.

Figure 55: σ = 0.0

Figure 56: Weiner Filtering

Figure 57: Inverse Filtering

Figure 58: σ = 0.001

Figure 59: Weiner Filtering

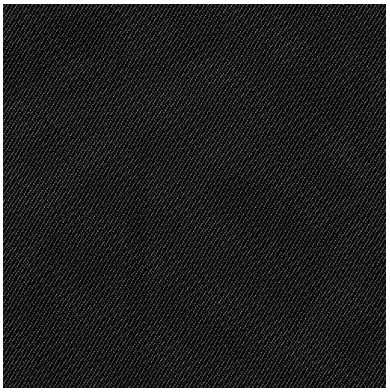

Figure 60: Inverse Filtering

Variation in Noise

With increase in noise, some more observations were made (listed below).

Figure 61: $\sigma = 0.01$

Figure 62: Weiner Filtering

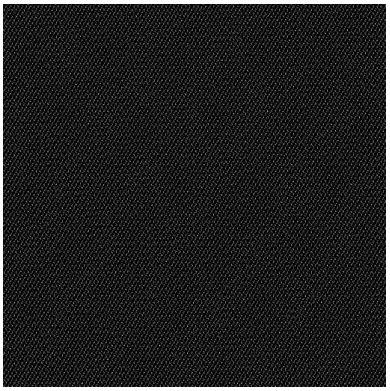

Figure 63: Inverse Filtering

Figure 64: $\sigma = 0.1$

Figure 65: Weiner Filtering

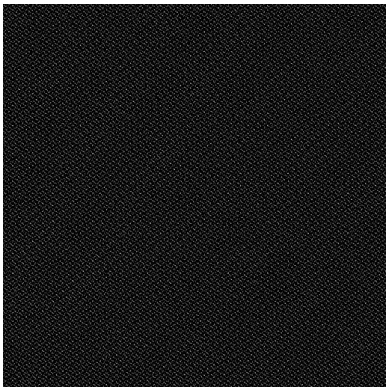

Figure 66: Inverse Filtering

- Inverse Filtering gives invalid results in case there's noise.
- The results obtained from Weiner Filtering get coarser as the noise increases.

Variation in Estimate

In the previous sections we used the original image spectrum $F(u, v)$ as the *estimate parameter* in *weiner filtering*. Now in this section I tried some generic estimates for the image. The following results were observed.

Figure 67: Noisy Image

Figure 68: Real Image Estimate

Figure 69: Constant Image Estimate

Figure 70: Power Estimate $\alpha = 0.53$

Real World Images

In this section we were supposed to use whatever we have learned so far, in trying to recover images from real world scenarios of blurring/noising.

Motion Blur I

The image provided was motion-blurred.

Figure 71: Uniform Patch ($\sigma = 0$)

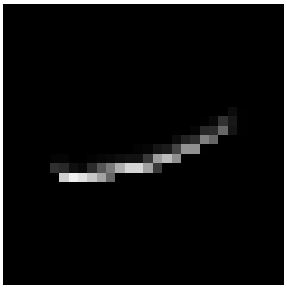

Figure 72: PSF Estimate

Figure 73: Blurred Image

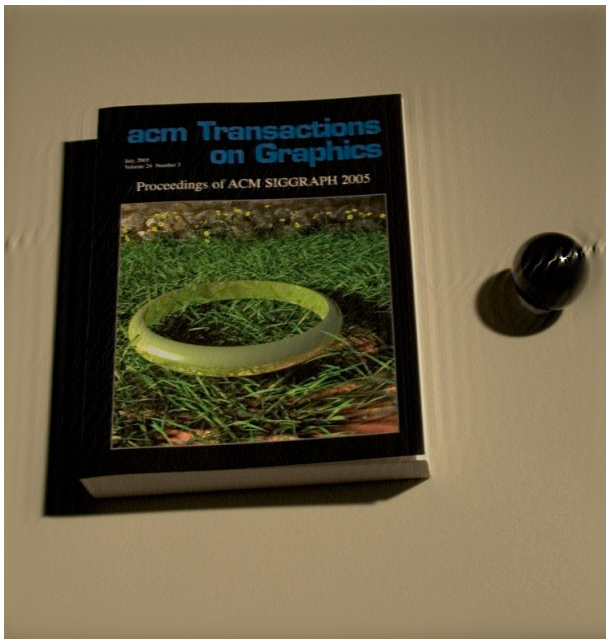

Figure 74: Recovered Image

Defocus Blur I

The image provided was focus-blurred. The image was analysed in various ways to get the idea of H Spectrum and N Spectrum.

For getting the idea of noise, a uniform patch in the image (sky) was clipped and then analysed, which indicated the noise was as good as void.

Then a (partial) edge in the image was analysed, where the gradient of values gave the idea of spread of blurring. A good approximate was a disc shaped spatial filter

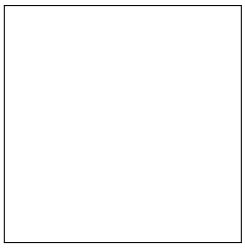

Figure 75: Uniform Patch ($\sigma = 0$)

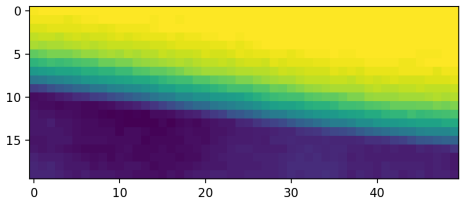

Figure 76: Gradient

Figure 77: Blurred Image

Figure 78: Recovered Image

Defocus Blur II

Working out this image involved the same steps as above, except for that the best approximation for the filter was a disc based function with radius 8.

Figure 79: Blurred Image

Figure 80: Recovered Image

Note

In the case of defocus blur, we observe that ringing is prominent. This may arise from the fact that since the disc function have abrupt cuts, which leads to ringing in the fourier domain, and thus edges resonates rings.

Also to eliminate unnecessary ringing from considering the periodic image (i.e. totally different pixel values once the image is repeated), I blurred the image borders with the wrapped around images. This caused significant decrease in ringing in recovered images.

Observations

Following were the major observations for sub-parts.

- **Mean filter** works better on *gaussian noise* and **Median filter** works best on *S/P* noise.
- **Total Variation** works amazingly well on *gaussian noise*, but fails to give promising results in *S/P* noise. This is perhaps due to consideration of $Laplacian$ operator which in case of S/P isn't of much use.
- **Weiner filtering** works best with *original image* as the estimate. Also, restoration is much better when the noise is low.

Summary

All required by the assignment was implemented and a lot of new things were learned/understood. The codes are written in Python. All the functions were implemented on own (except for the FFT, which was allowed to be used).
The code was patterned as a library of functions, which can be used on any Python interpreter in a series of steps to give the desired solution. (for assignment)
The assignment was intriguing and I learnt a lot of new ideas. Thanks and Regards.

*Bonus Section

I eventually came across a way to design an edge preserving filter. The results were impressive, so I tested it with a challenging image, shown below (Zoomed Versions).

Figure 81: Original Image

Figure 82: Edge Preserved

Figure 83: Weiner

A simple introduction of penalty in terms of **Total Variation Norm** led to edge being preserved. Not much changes were made, since I had implemented total variation in the denoising part :)

Figure 84: Original Image

Figure 85: Blurred/Noisy Image

Figure 86: Edge Preserving Filter

Figure 87: Weiner Recovered