UNIVERSITY OF ROCHESTER
CSC 249, MACHINE VISION
SPRING 2017

Project 1

Nathan Conroy

February 28, 2017

1 INTRODUCTION

The purpose of this project was to familiarize us with image processing techniques using the many built-in Matlab capabilities as well as some self-made functions. The assignment involved testing many different types of filters and analyzing how they work. This was done by calculating the filter kernels and plotting their frequency responses as well as applying them to the image via filtering techniques. This allowed me to observe their benefits and weaknesses, such as how they handle certain types of irregularities found in images, such as noise. With this information I was able to propose solutions for how to fix such image processing problems. The project also introduced us to calculating and analyzing Fourier Transforms in images and how features such as brightness affect their result.

2 METHODS

Part 1 of the assignment served as a simple introduction to reading, displaying, and writing an image using basic Matlab commands. I read in an image, "phase1_grayscale.jpg", which was one of the brighter images I took for our first homework assignment. I then displayed the image in two different ways using Matlab's imshow function (Figure 2.1) and Matlab's imagesc function (Figure 2.2).

Figure 2.1: phase1_imshow.jpg

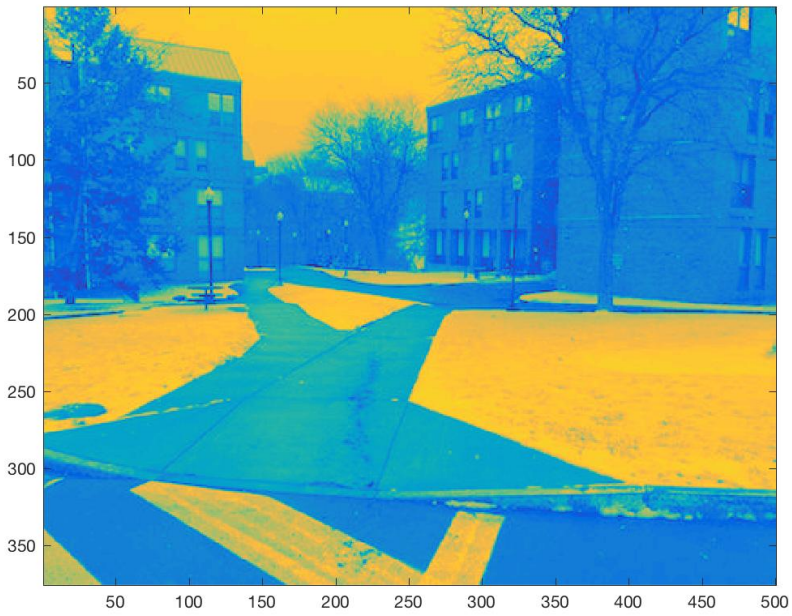

Figure 2.2: phase1_imagesc.jpg

Using Matlab's colormap and colorbar functions, I created and displayed a colormap representing the distribution of triplet color values represented in the image with a colorbar to indicate the mapping of data values into the colormap. The resulting figure is seen in Figure 2.3.

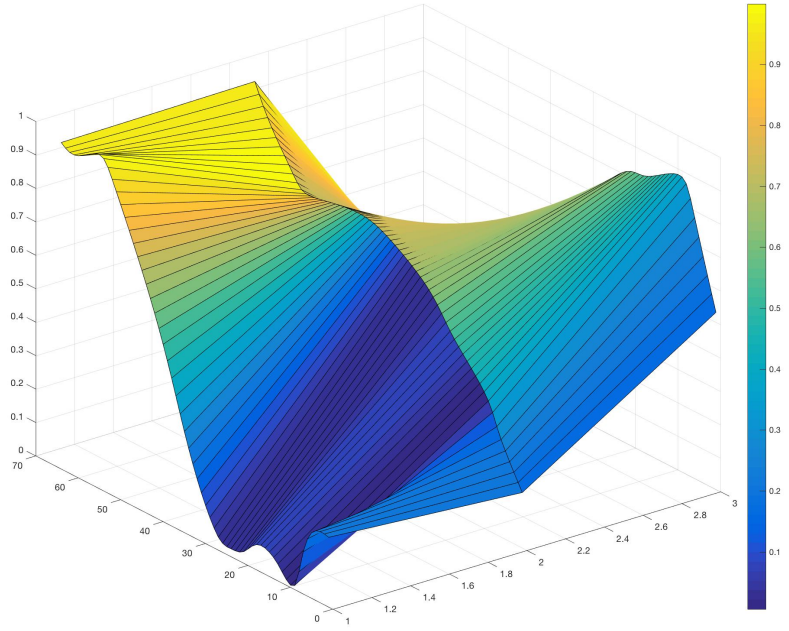

Figure 2.3: colormap.jpg

The second part of the assignment, which dealt with filtering images in the space domain, required that I use a variety of different image filter types on the same image and view how certain changes to the image, such as the addition of noise, affected the results. The first type of filter I applied to the image was a 3x3 Gaussian Low Pass filter. I used the built-in fspecial function to create a 3x3 filter with a standard deviation of 3. I then used the built-in filter2 command to apply the filter to my image, which is displayed in figure 2.4. I then used Matlab's freqz2 command to display the frequency response of the filter, which can be seen in Figure 2.5.

Figure 2.4: gaussian_lp.jpg

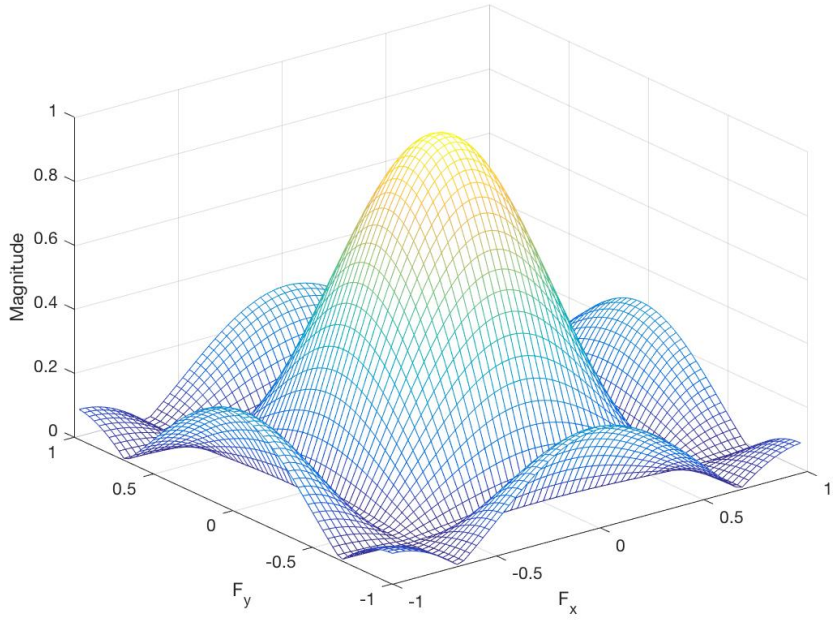

Figure 2.5: freqResponse_lp.jpg

The second type of filter I applied to the image was a 3x3 High Pass FIR filter, however this time instead of using the built-in Matlab commands I created my own function for the purpose. I first had to design a 3x3 kernel to do the filtering, which is shown in Figure 2.6. I then filtered the image by iterating through every single pixel of the original image and performed a 2-D convolution using the kernel and a 3x3 neighborhood in which the pixel in focus was the center of the neighborhood. The resulting image is shown in Figure 2.7. I again used the freqz2 command to display the frequency response of the kernel, which can be seen in Figure 2.8.

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 2.6: kernel used for the high pass filter

Figure 2.7: gaussian_hp.jpg

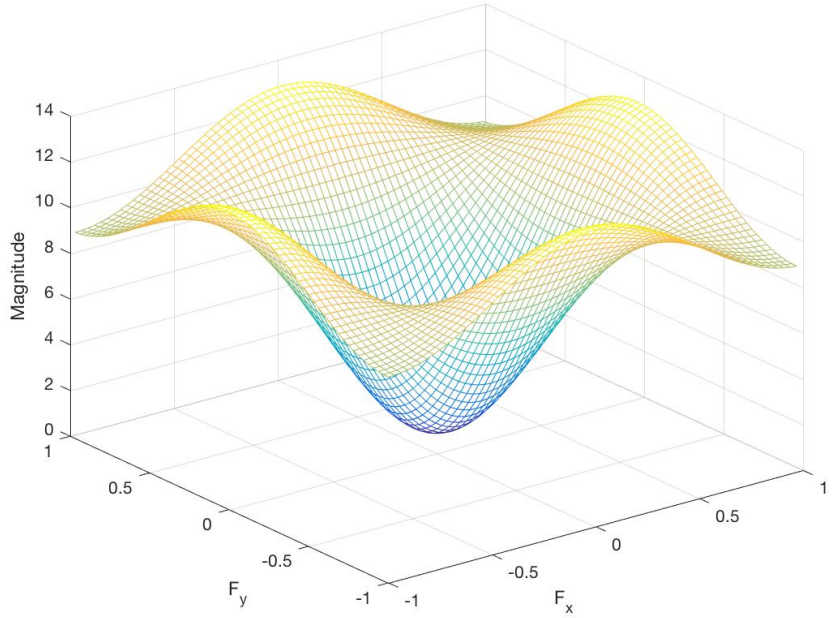

Figure 2.8: freqResponse_hp.jpg

I then applied a 3x3 Median Filter using the built-in medfilt2 command (as seen in Figure 2.9) and a Sobel Edge Filter using fspecial type 'Sobel' to get the filter and filter2 to apply the filter. The result of the Sobel filter is seen in figure 2.10, and its frequency response is shown in Figure 2.11.

Figure 2.9: medfilt.jpg

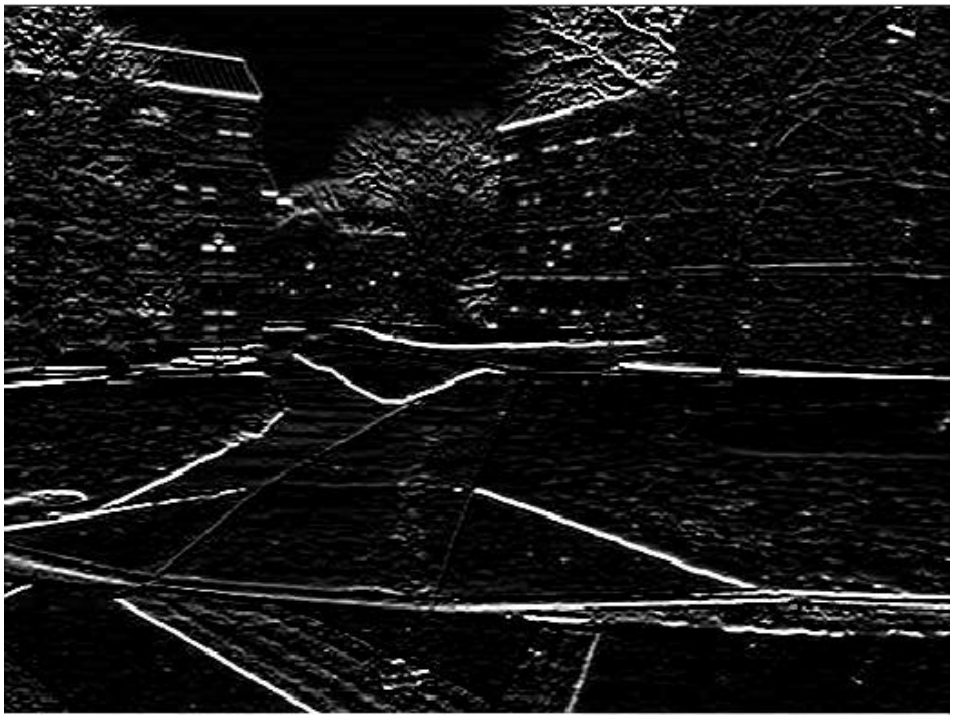

Figure 2.10: sobelFilter.jpg

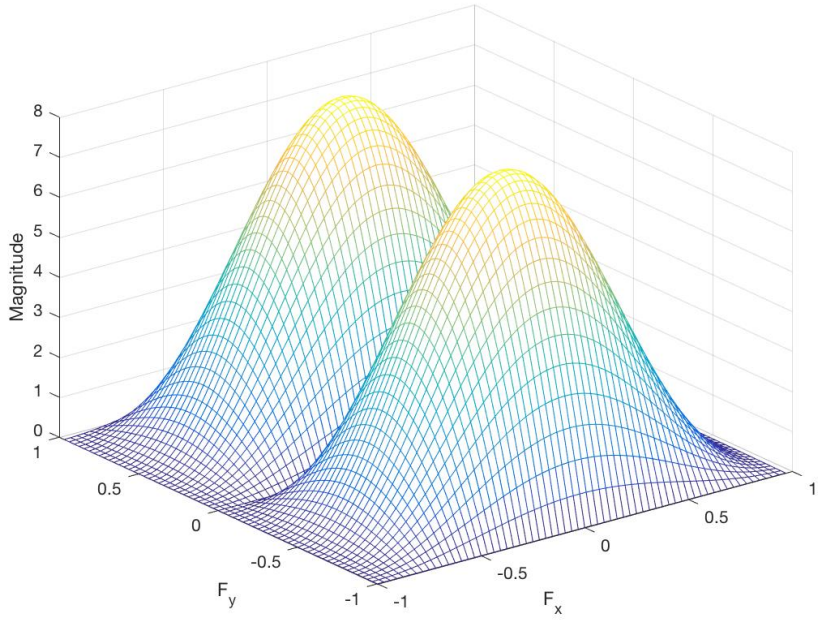

Figure 2.11: freqResponse_sobel.jpg

The next step of the assignment was to add Gaussian noise to the original image and view how it affected the previously performed filters. I used the built-in imnoise command to add noise to the original image of type 'gaussian'. The resulting image is shown in Figure 2.12.

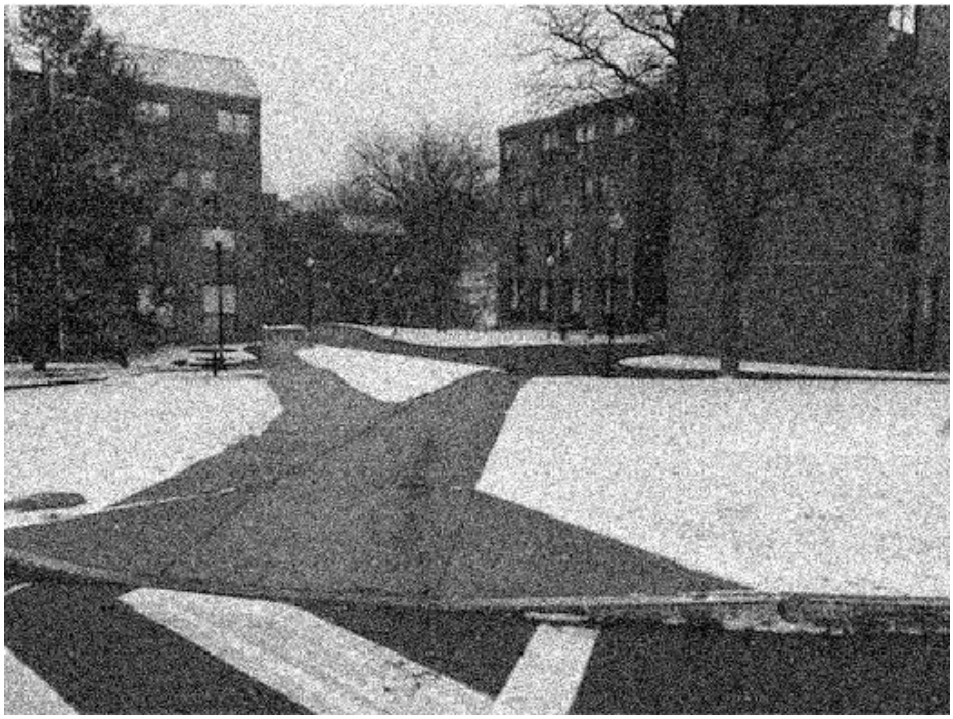

Figure 2.12: original image with Gaussian noise

I then performed all of the previously performed filterings in the exact same manner but with the input now being the image with Gaussian noise. The result of the 3x3 Gaussian Low Pass Filter is shown in Figure 2.13, the result of the 3x3 High Pass FIR Filter is shown in Figure 2.14, the result of the 3x3 Median Filter is shown in Figure 2.15, and the result of the Sobel Edge Filter is shown in 2.16.

Figure 2.13: Gaussian Low Pass Filter on image with Gaussian noise

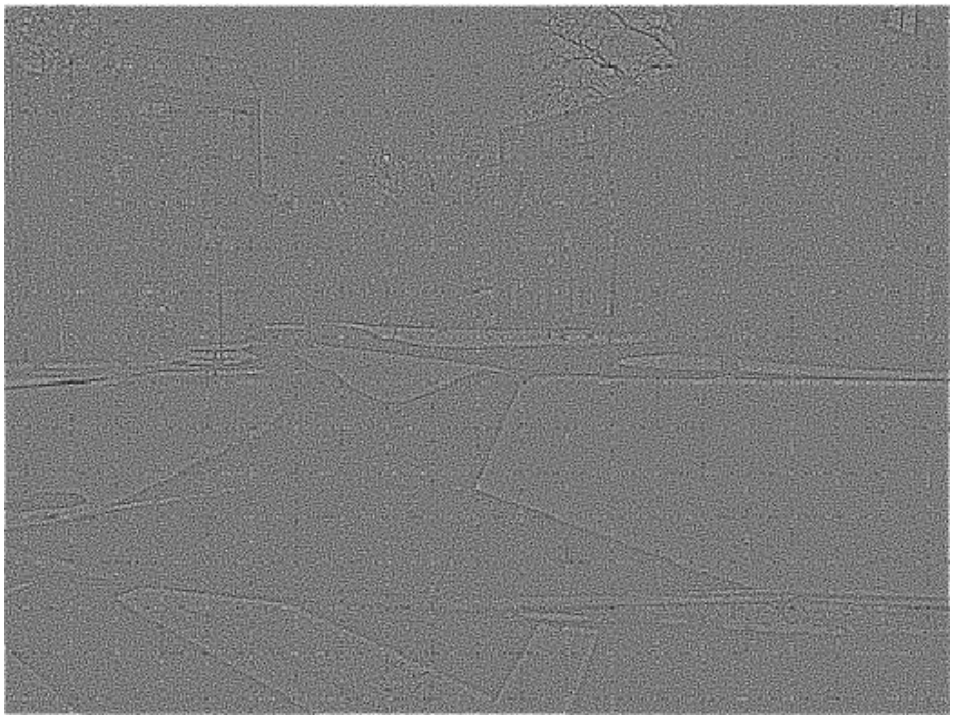

Figure 2.14: High Pass FIR Filter on image with Gaussian noise

Figure 2.15: Median Filter on image with Gaussian noise

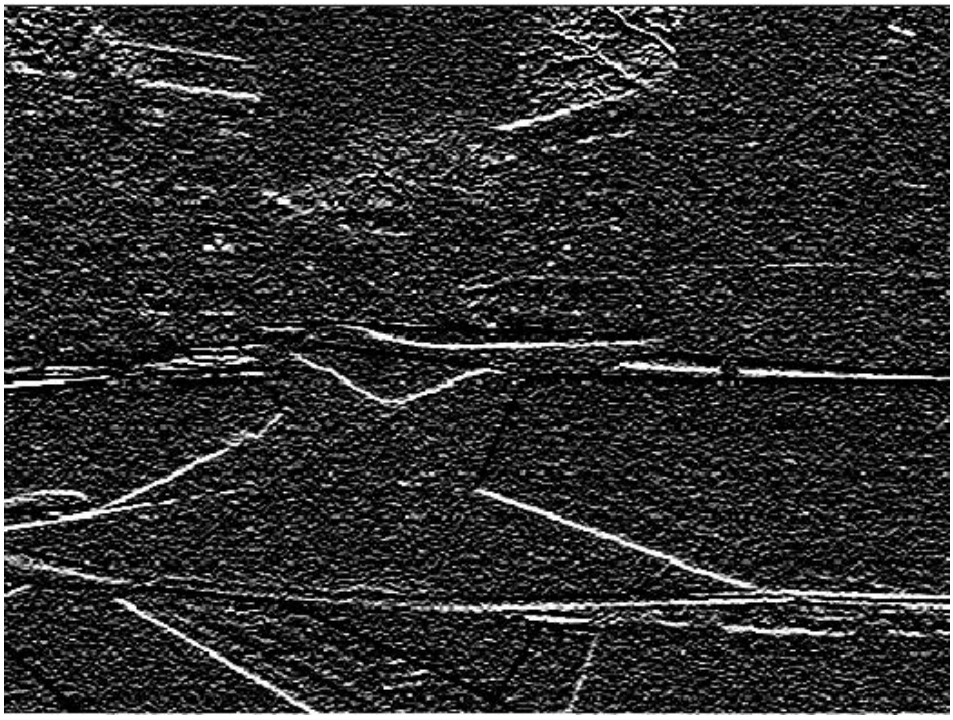

Figure 2.16: Sobel edge filter on image with Gaussian noise

For the last part of section 2 of this assignment, I added impulse noise of the type 'Salt and Pepper' to the original image (as seen in Figure 2.17), and performed the same operations again to view the results of each of the other filters. The result of the 3x3 Gaussian Low Pass Filter is shown in Figure 2.18, the result of the 3x3 High Pass FIR Filter is shown in Figure 2.19, the result of the 3x3 Median Filter is shown in Figure 2.20, and the result of the Sobel Edge Filter is shown in 2.21.

Figure 2.17: original image with impulse noise

Figure 2.18: Gaussian Low Pass Filter on image with impulse noise

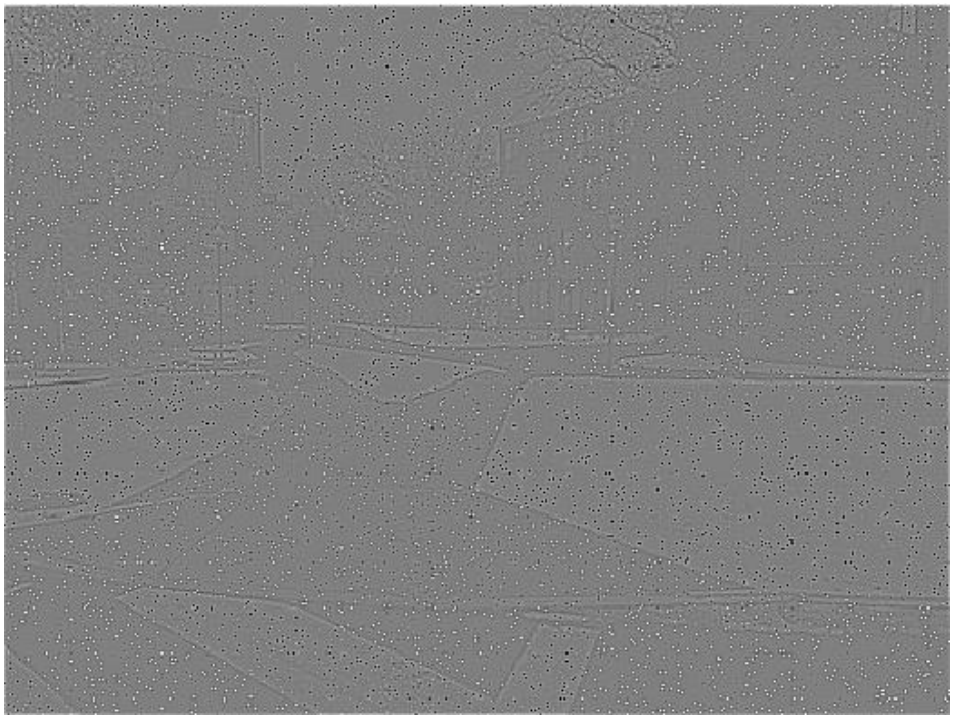

Figure 2.19: High Pass FIR Filter on image with impulse noise

Figure 2.20: Median Filter on image with impulse noise

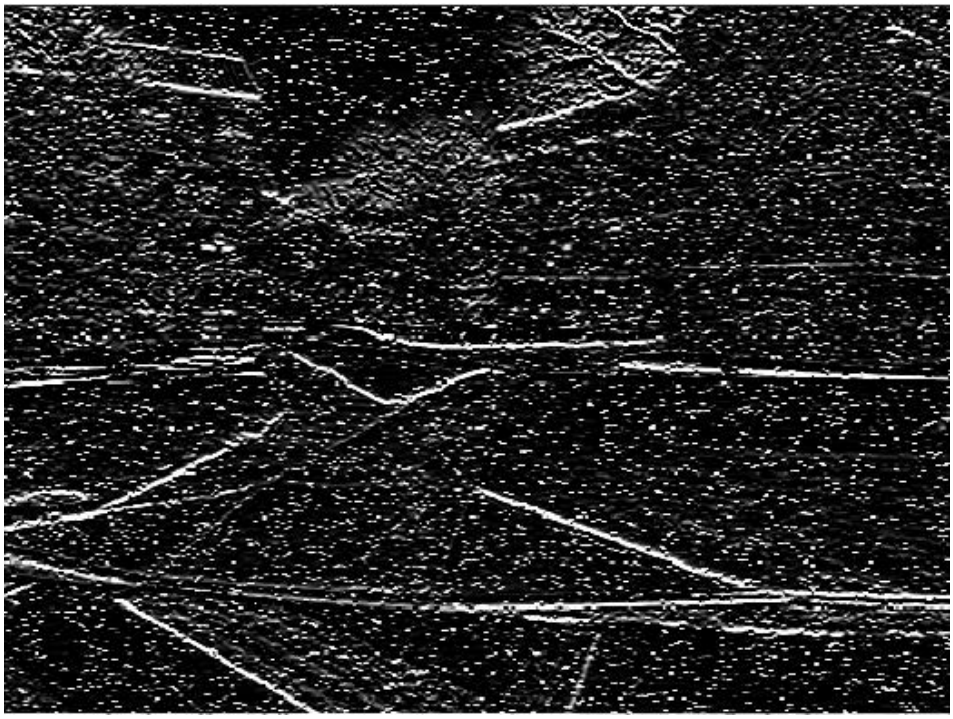

Figure 2.21: Sobel edge filter on image with impulse noise

The third part of the assignment required that I calculate the 2-D Fourier Transform of the original Image and display the results. I used the built-in fft2 function to calculate the FFT and displayed the result with imagesc. The resulting image is seen in Figure 2.22. I then used the fftshift to shift the component to the center of the spectrum, as seen in figure 2.23.

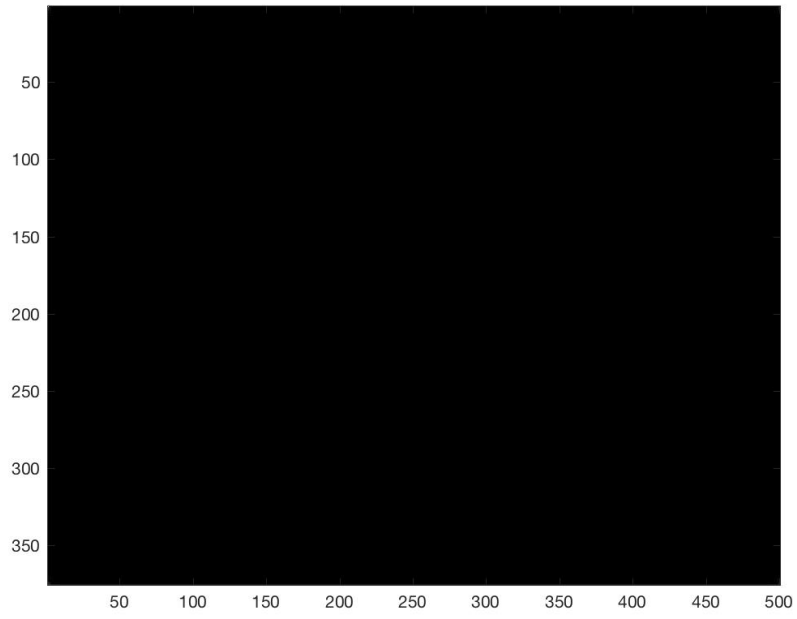

Figure 2.22: 2-D Fourier Transform of image, unedited

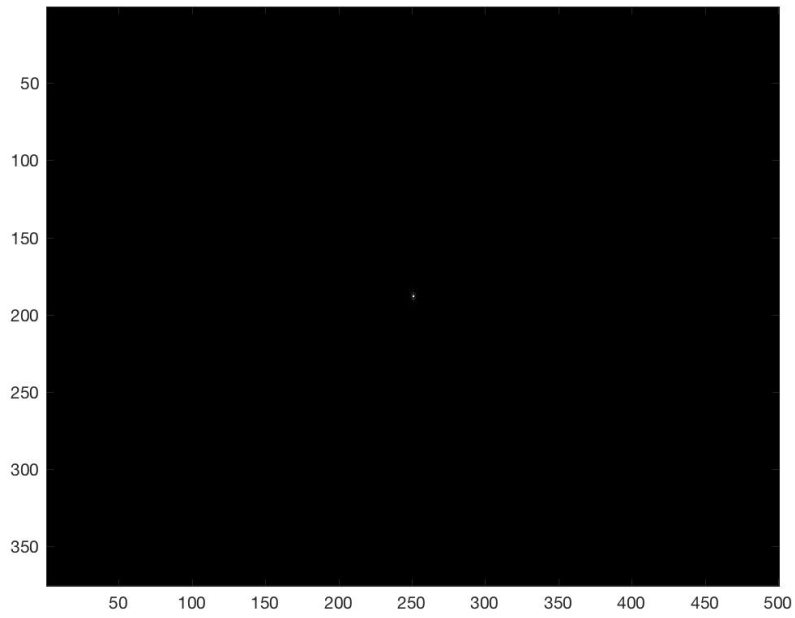

Figure 2.23: 2-D Fourier Transform of image, with fftshift

Lastly, I repeated the previous 2 steps on the image, but first took the log of the FFT magnitude and added a small offset. The resulting images are seen in Figures 2.24 and 2.25.

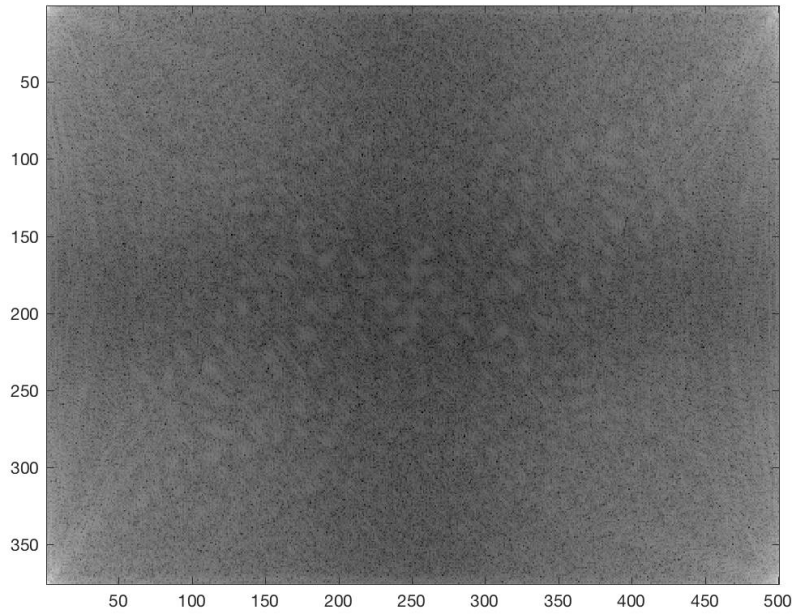

Figure 2.24: 2-D Fourier Transform of image, with log + offset

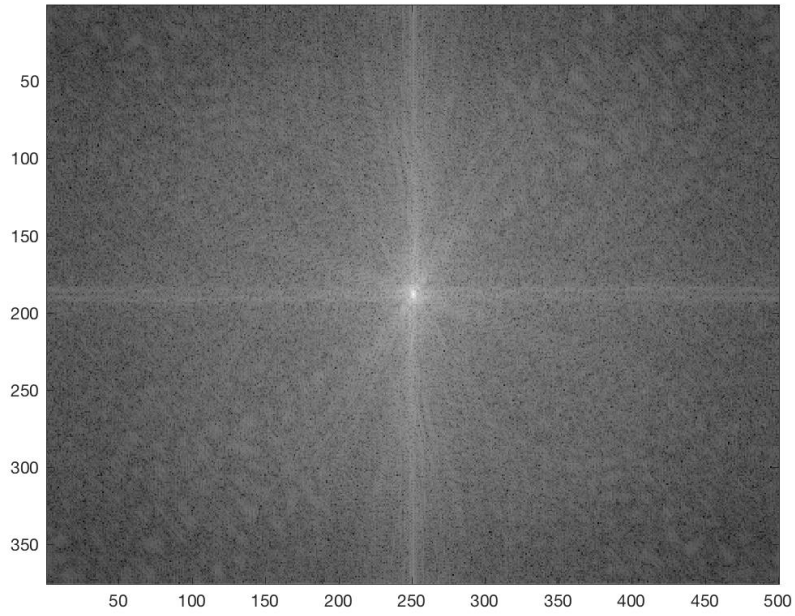

Figure 2.25: 2-D Fourier Transform of image, with fftshift and log + offset

3 Results and Analysis

In part 1 of the assignment, I displayed my image using both imshow (Figure 2.1) and imagesc (Figure 2.2). While the purpose of both commands is to the display the image taken as a a parameter, they both serve different functions. The imshow command simply displays the graphic in its given form. The imagesc command on the other hand displays the graphic with scaled colors representing the full range of colors in the colormap. Pixels with the same value are represented with the same color, and the deviations between the colors are distributed evenly. This feature is very handy for analyzing images that use a small range of pixel values.

The result of my 3x3 Gaussian Low Pass filtered image in part 2 of the assignment looks fairly similar to the original image upon first glance. However if you look closely you notice that the image seems to look a bit blurrier and smoother around the edges. It is specifically more noticeable when looking at the tree branches in the background, which seem to blend together much more than in the original image. This makes sense because Gaussian low pass filtering is also called Gaussian smoothing and Gaussian blur, and is typically used in graphics software to reduce detail and image noise.

The Gaussian 3x3 low pass filter is also separable, meaning that it can be applied to an image as two independent 1-D calculations and give the same result as the 2-D calculation. To demonstrate this, I created a 1x3 filter and a 3x1 filter using the fspecial command, and applied them both to the image separately using the filter2 command, and the resulting image was identical to the result produced by my 3x3 filter.

Figure 3.1: image filtered with Gaussian 3x3 filter

Figure 3.2: image filtered with separate 3x1 and 1x3 Gaussian filters

I created my high pass filter function by simulating the process of the built-in filter2 command. A high pass filter serves the purpose of sharpening an image's features by bringing out the outlines in the image. The kernel which I chose was a good fit for this purpose, and I applied it to the image by coding a process which performs a 2-D convolution of the kernel and the image. I processed pixels at the image border by using the "zero padding" technique.This technique works by filling in missing pixels with zeros in neighborhoods in which the pixel in focus is a border pixel. An example of this is shown in figure 3.3.

Figure 3.3: zero padding technique

I found the results of the filtered images with added noise to be quite interesting. The low pass filter seemed to reduce the noise of the image significantly by smoothing the image for both the Gaussian and the impulse noise. The high pass filter seemed to do the opposite, bringing out the added noise while softening the background for both types of noise. However, it seemed to handle the impulse noise even worse than the Gaussian noise. The median filter seemed to act similarly to the low pass filter, reducing the noise of the image. With Gaussian noise, the median filter seemed to reduce the noise less than the extent of the low pass filter, however it seemed to perform much better for the impulse noise, nearly eliminating the noise altogether. The Sobel edge detector was not immune to noise, and seemed to handle it rather poorly. While prominent edges were able to stand out within the image with thicker white lines, smaller edges resembled the noise picked up by the filter making the image seem very cluttered. A method that I would propose to reduce the effects of noise in a high pass filter is to first perform a Gaussian low pass if the noise is of Gaussian type or perform a median pass filter if the noise is of impulse type, and then perform a high pass filtering of the result.

When computing the FFT of my image, my initial display of the result turned out to be essen-

tially completely black. It wasn't until I shifted the origin to be at the center of the diagram using the fftshift command that I was even able to see the origin, which appeared as a small white speck. When I took the log of the FFT magnitude (plus an offset) before displaying in part 3d, the resulting display became much more detailed and easy to analyze. I preferred this display for this reason.

Figures 3.4 and 3.5 juxtapose the Fourier Transforms of both the bright scene image (my original image) and the dark scene image (the other image from homework 1).

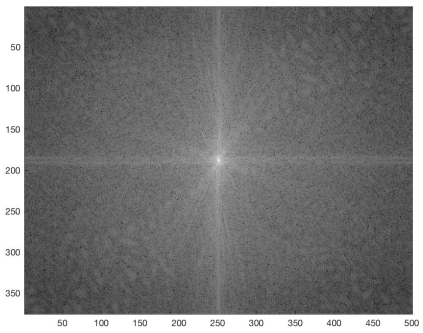

Figure 3.4: Fourier Transform of brighter image

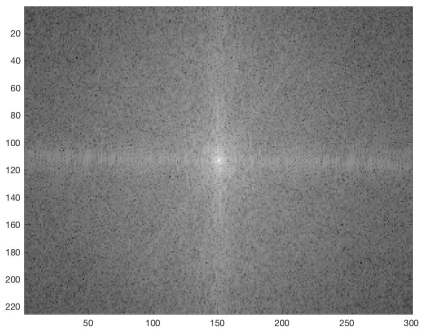

Figure 3.5: Fourier Transform of darker image

As seen, both images seem to have two dominating directions, vertical and horizontal passing through the center. It is definitely apparent that the magnitude of the brighter transform is greater than that of the darker transform, and its stripes are more sharpened. The two transforms also have different origins, the brighter being at about (250, 200) and the darker being at about (150, 110).

4 Appendix

I chose to include my figures within their relevant sections to improve readability.