

CSC 391: Spatial and Frequency Filtering

Name: Zander Miller

1) Smoothing and Denoising

The first method of smoothing/ denoising tested was a Gaussian filter, which is a type of linear filter that assigns value to each pixel from the mean of surrounding pixels, with priority based on proximity to the center. The priority is calculated with the function:

$$\frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Where the center of the filter array is (0,0), so each location in the array further from the center would have a greater value for (x^2+y^2) , and therefore a lower overall part of the mean.

The variable σ effects the level of bias towards close to the center pixels in the array. The higher σ , the less effective the bias. The variable k , determines the size of the Gaussian filter, so that the filter is a $k \times k$ square.

Fig. 1, on the next page, provides an illustration of what various combinations of σ and k look like, where the row is σ and the column is k .

It's notable that increases in both of these generally increase the level of blur, and the effect is most noticeable when both are increased at the same time. One of the first things I noticed while experimenting was that while σ was 1, the difference between higher numbers of k was not observable. In **Fig. 1**, it's difficult to tell the difference between the 9x9 filter and the 27x27 filter, and in fact, while $\sigma = 1$, both closely resemble the results of a 101x101 filter. This is because despite the size of the filter, since σ was low, the pixels further away from (0,0) were so unprioritized that they were having no bearing on the filtered image. Increasing σ allows for new levels of blurriness.

The bottom right corner of **Fig. 1** was the blurriest level of Gaussian filter tested, and it's clear that much of the detail, such as the edges, was lost. The static like distortion of the original image is no longer visible, but much of the fidelity of the image was lost, and the progression of **Fig. 1** really illustrates the reason that Gaussian blur isn't a great method of removing distortion. The image goes from an image with distortion, to a blurred image with distortion, to a blurred image. Details like the individual hairs on the dog, the shine in her eyes, and the wetness of the nose are lost, while the effect of the distortion is still kind of noticeable.




















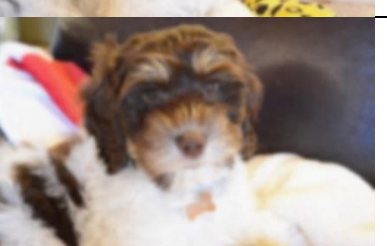





σ	3x3	5x5	7x7	9x9	27x27
1					
2					
3					
4					
5					

Fig. 1: Gaussian Filter Data on 'DSC_9259-0.25.JPG'



Fig. 2: Median Filter on 'DSC_9259-0.25.JPG'

Fig. 2 (Above) shows the progression of different levels of a median filter, which replaces each pixel with the median of it and its surrounding pixels within a $k \times k$ area.

Unlike the Gaussian filter, the median filter tends to preserve the sharpness of the division between objects. For example, even with the blurry 27x27 median filter shown above, the division between the dog and background is still discernable. That said, some of the subtler sharpness, such as the dog's fur or eye, is eventually lost as the filter progresses. When k is increased, the filter also develops a tendency to 'smear' areas of color to incorrect areas, a close inspection of the 27x27 filter reveals that the red area of the Santa hat behind the dog has increased. This is not an issue that the Gaussian filter had, and I think the reason for this is that the Gaussian filter gives priority to closer pixels, while a 27x27 median filter gives an element 12 pixels away the same priority as an element right next to the pixel in question. As a result, the smearing has a similar vertical and horizontal pattern as the box filter blur we observed in class. This is notable since the box filter also gives the same priority to all pixels in its range.

Both the Gaussian and Median blurs come at the cost of fine detail in the image, but the smaller median ones do a better job at preserving some of the sharpness of the image, making the effect subtler than its Gaussian counterpart. Larger median filters, however, pretty much destroy the image.

The median filter also notably had a longer execution time, usually taking around 40 seconds to complete, compared to the less than a second execution times of the Gaussian blur.

Edge Detection

I ran several tests using the Canny edge detection function of cv2, the first experimented with adjusting the low threshold variable of the function. The results are included below:



Fig. 3: Effect of Canny Edge Detector on 'DSC_9259.JPG' with various low thresholds

Increasing the low Threshold had the effect of ignoring more edges, such as some in the fur, but it does not make it clearer at discerning objects. Individual lines get smaller, but it doesn't really change the fact that the dog's snout doesn't have an easily detectable outline above their chest.

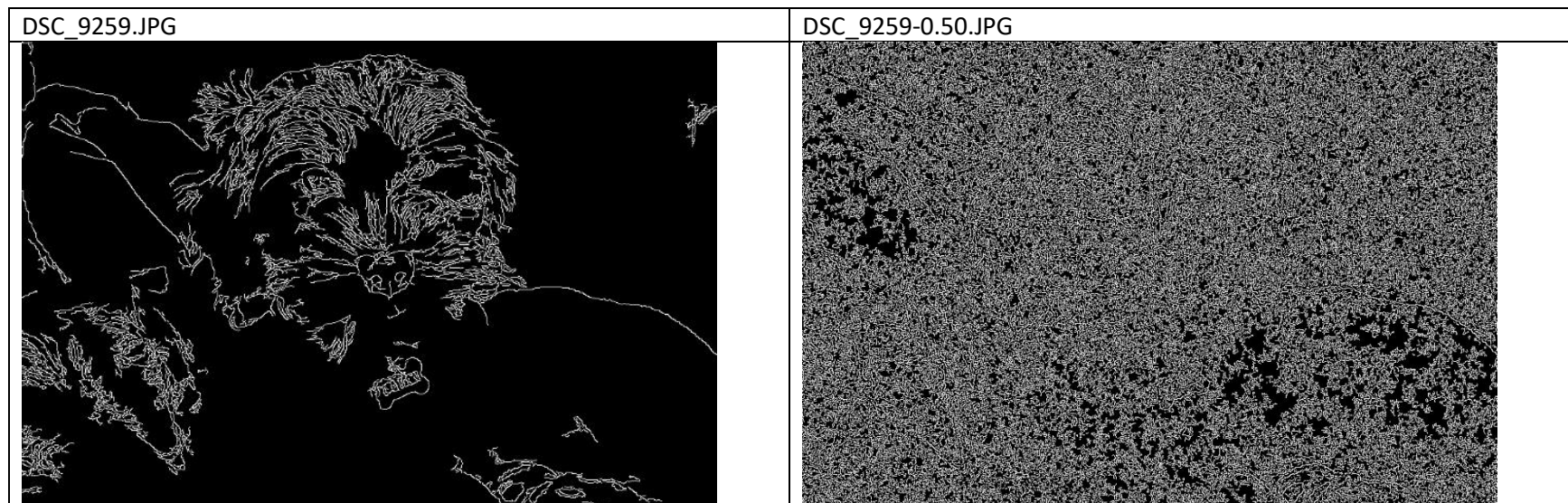


Fig. 4: Effect of Canny Edge Detector on original and 'noisy' images

Using the Canny Edge Detection on the noisy image of the dog however, produced much less useful results. As **Fig. 4** demonstrates the noise interferes with the edge detection to such a degree that it is impossible to discern anything in the output.

One way to get around this could be to use a smoothing algorithm that I tested in the previous part, to see if these could reduce the effects of the noise:

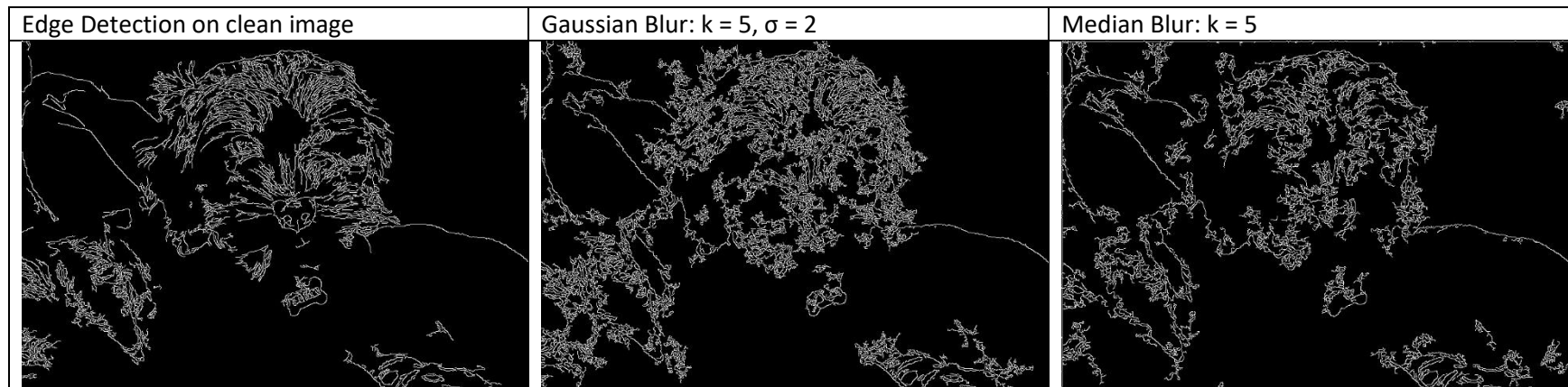


Fig. 5: Impact of using blurring in combination with Canny Edge Detection on 'DSC_9259-0.50.JPG'

Fig. 5 shows the impact of using various blur techniques on the noisy image, and it's clear that both have a significant impact on the success of the edge detection. Both have allowed actual discernable edges to be detected in the image compared to the unfiltered version, but the Median Blur seems to have preserved more detail and has less excess noise related lines compared to the Gaussian blur. This is most prominent on the dog's nose, which is not visible on the Gaussian image, but is on the Median one. It makes sense based on my original analysis of these, where I noted that the median blur preserved more edges than Gaussian.

2) Frequency Analysis

The 'Frequency_Analysis.py' program performs a 2-Dimensional DFT on an image, with functionality that can display the result in either a three-dimensional plot or a two-dimensional image.

It also allows for the display using the log of the magnitude+1, as this allows for much more usable information than the original, mostly black alternative.

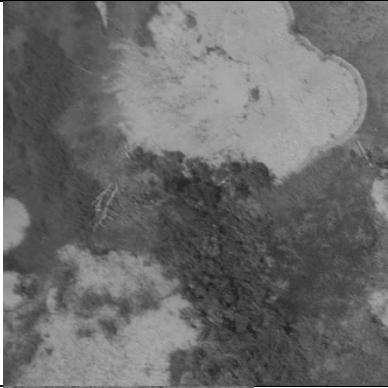


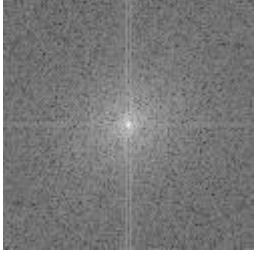
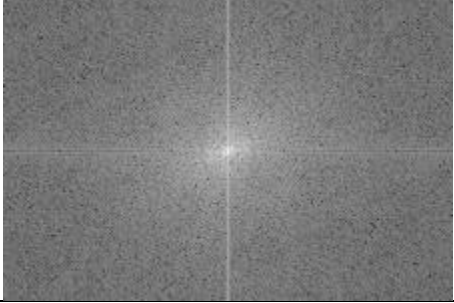
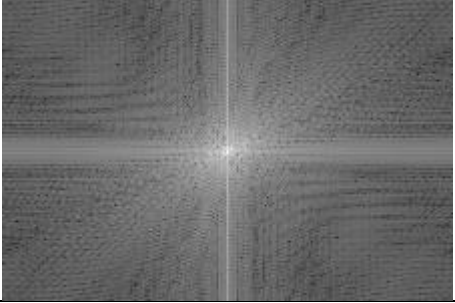
File Name	window-00-00.jpg	DSC_9259.JPG	Denoise_Med_27x27.jpg
Image (in Grayscale)			
Magnitude Plot			
x-axis coefficients	[127 91 116 98 123 111 119 125 123 105 127 94 112 110 122 100 103 83 118 111 124 116 79 109 82 127 127 117 104 94 123 124 118 109 118 120 104 101 111 120 118 120 111 119 110 114 112 120 104 116 97 97 118 97 123 107 122 117 133 98 109 127 115 119 83 115 121 124 118 132 119 113 132 113 118 127 112 94 106 97 119 110 105 102 120 95 119 120 115 110 112 120 114 123 120 112 109 111 106 113 125 114 117 124 119 126 116 94 112 123 121 104 127 98 109 122 101 116 129 101 128 106 90 116 107]	[123 112 118 85 111 109 117 74 120 106 110 110 106 103 110 116 98 105 122 114 120 119 83 104 107 103 126 116 116 106 112 116 118 115 123 92 125 120 119 126 116 118 106 54 123 117 87 118 109 84 119 119 105 102 114 122 120 117 121 122 124 119 108 124 124 122 115 107 103 114 108 124 123 128 133 104 130 125 108 111 128 115 119 116 121 125 122 81 122 119 118 121 112 110 119 124 123 116 109 122 92 115 113 109 113 99 114 109 102 97 114 108 127 118 116 114 118 116 113 112 119 111 115 112 105 114 108 92 109 115 111 123 102 119 118 123 95 95 109 111 124 111 126 111 119 108 117 118 109 75]	[99 85 90 95 92 90 92 91 86 101 91 95 99 88 89 99 81 85 99 72 88 93 76 81 84 63 61 78 90 64 92 86 85 73 101 95 89 84 88 94 101 98 66 88 51 105 87 94 75 86 95 87 93 82 72 99 94 101 96 106 74 104 90 100 79 97 63 109 99 118 116 124 122 122 133 121 133 123 123 123 113 119 97 108 88 105 85 72 88 107 78 97 96 98 96 102 98 93 83 81 101 90 90 69 90 88 81 100 83 93 95 88 84 55 98 93 101 75 65 90 72 87 82 78 83 74 88 90 85 96 86 44 102 92 88 105 96 90 103 96 92 96 82 97 91 84 88 101 85 75]

Fig. 6: Effect of 2D DFT on various images

Fig. 6 (above) displays the results of the use of the 2D DFT on several images, the first two are just grayscale converted versions of otherwise unchanged images, and the third is the output of a 27 x 27 median blur on the puppy image.

Interestingly, the DFT plot of the original puppy image more resembles that of the drone image than the blurred puppy image. Generally, the coefficients of the first two had greater values than the median blur, with the exception of the bright vertical line near the center. The first two also seem to show more variance, which is why their plots look resemble tv static.




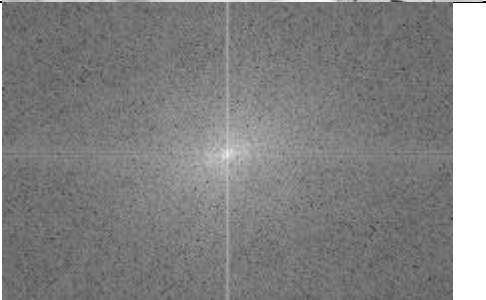

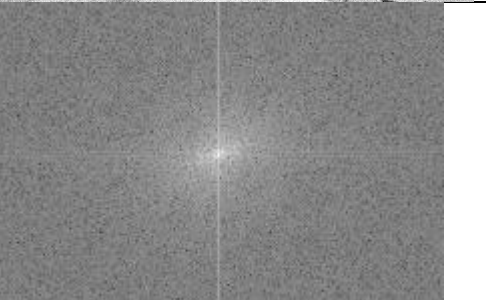
File Name	DSC_9259.JPG	DSC_9259-0.25.JPG	DSC_9259-0.50.JPG
Image (In Grayscale)			
Magnitude Plot			

Fig. 7: DFT for images with noise

Fig. 7 shows the plots for the same image with increasing levels of noise. The increase in the Fourier coefficients' magnitudes with increasing levels of noise is discernable due to how the plots become lighter with more noise. This is because there a much greater frequency in the change in each pixels value when random noise is added.

One of the most consistent similarities visible in all of the images that I tested, are the horizontal and vertical lines that follow the axes of the plots. It is most noticeable with the vertical one, but closer inspection allows one to see it with the horizontal view as well.

In class, we were able to view several examples of what DFT plots would look like on images such as letters. I noticed that lines on the plot were often perpendicular to lines on the image they were derived from. The Letter A, for instance, is made up of two diagonal lines and a horizontal one, has a plot featuring diagonal white lines, and a stronger vertical line. This is because there would be higher contrast detected across a bold black line one a white background.

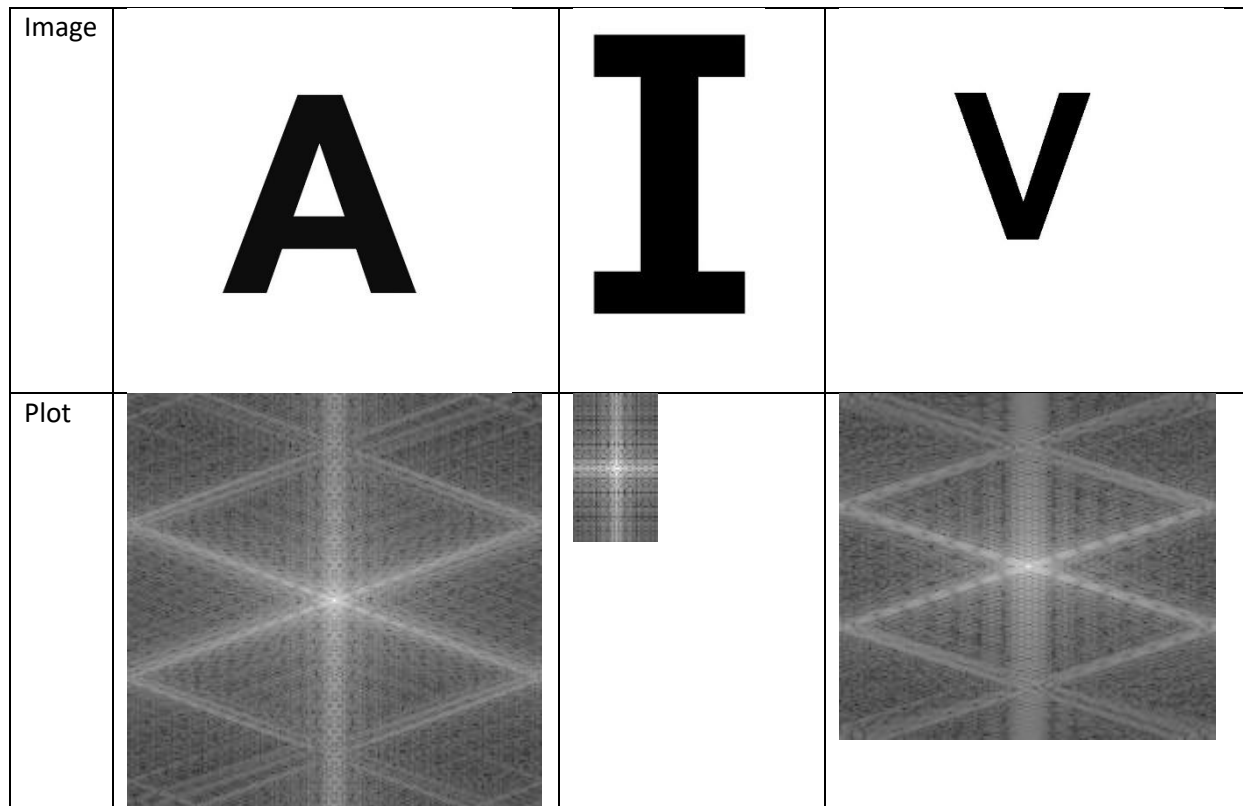


Fig. 8: DFT plots of various letters

These plots help isolate how change in different directions can affect the DFT plot. It is notable that the 'I' plot has white lines across the X and Y axes, and this is because the image only features changes in value in those directions. At first, I was surprised at how 'A' and 'V' had such similar plots, despite 'A' having an additional horizontal line, but then I realized that 'V' had changes in value across a horizontal line at the top and bottom of the letter, leading to the same thing.

Knowing this trend, I can discern that the other images also have strong changes in value across a horizontal surface as well. In the puppy image, for instance, the edge between the puppy and the couch is mostly horizontal, and if I refer to **Fig. 3**, it's apparent that the horizontal edges such as the top of the puppy's leg and head, are more pronounced than the vertical ones, such as the right side of the puppy's head. The right side was not even detected as a continuous line when the low threshold was set to 100. I believe this is why, in **Fig. 7**, the horizontal line is less pronounced than the vertical one.

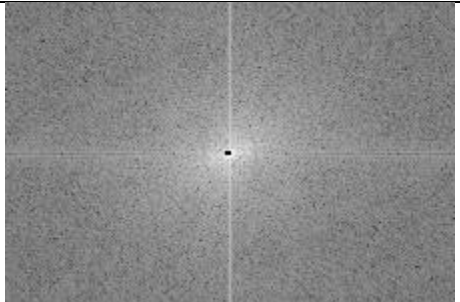
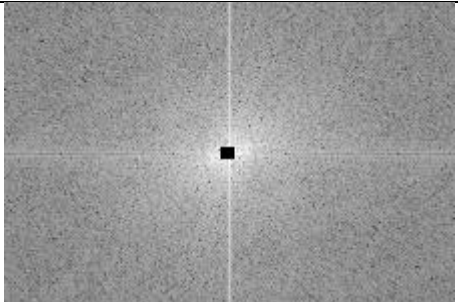
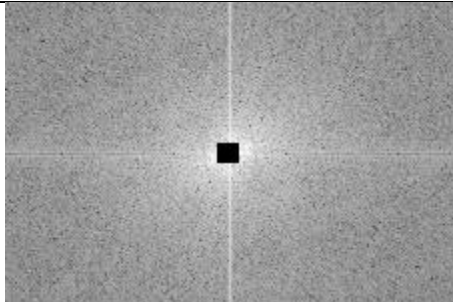



Type of Zeroes	Center, 2x2	Center, 6x6	Center, 10x10
Plot			
New Image			

Fig. 9: Effects of zeroing out a black box in the middle of the DFT plot

Fig. 9 (above) shows the effect of changing the values of a black box in the middle of the image to 0. I was surprised to see that this didn't cause an easy to detect pattern. From what I can tell, increasing the size of the box caused the dark-light balance to change. The smallest box made the lighter areas of the image slightly darker, and the darkest areas incredibly light. The increase in the size of the box seemed to shift the threshold back, where some parts that were formerly slightly dark became incredibly light, and some of the darkest parts of the image became darker.

I believe this is because the box is screwing up the transitions between edges, this is most discernable with the 2x2 box, which only really disrupts the relationship between very-dark to very-light edges, and effectively reversed the relationship.

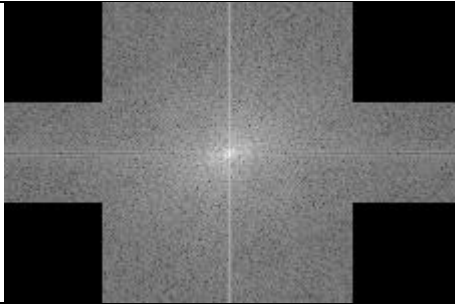
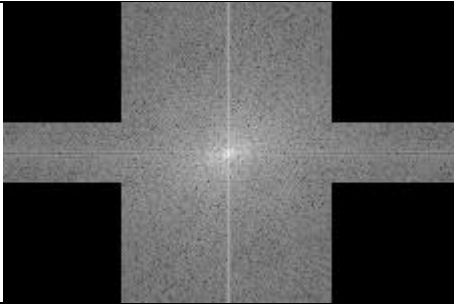
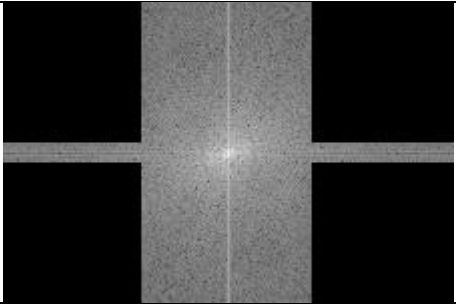



Type of Zeroes	Corners, 50x50	Corners, 60x60	Corners, 70x70
Plot			
New Image			

Fig. 10: Effect of zeroing out black boxes in the corners

Fig. 10 (above) shows the effect of zeroing out black boxes in the corners of the DFT plot. Before I discuss the figures themselves, I feel I should note just how large the boxes had to be before there was a discernable effect on the image. I didn't include several tests with smaller boxes that didn't change anything, but it wasn't until I set the size of the box to 50. I was surprised how little was changed, even with fairly large boxes. Compare this to changing 4 pixels in the middle and reversing the relationship between dark and light.

The first effect I noticed was that some pixels in the white part of the dog's paw changed to black. This generally increased as more pixels closer to the center were changed. I would assume this was related to the phenomenon I noticed earlier, with the disruption of the dark-light balance.

The subtler change was the appearance of vertical and horizontal stripe distortion that only became visible with the 70x70 boxes. I believe these are the result of the loss of diagonal high-frequency information while preserving the horizontal and vertical. This might be leading to an emphasis on those contrasts, and appear as stripes.

I think a more precise decrease in magnitude of the outer edges could be used to create a smooth effect. The outer edges seem to control shifts between similar values, so removing them with a more elegant method than a black box could remove the shift between light noise.

The reduction of the pixels near the center seems to effect the changes at low frequency edges, so it is reasonable that this information could be useful for the detection of prominent edges.

3) Frequency Filtering

The final part of my project involved the employment of Ideal and Butterworth filters on images. An Ideal filter is one that sharply cuts off frequencies across a specific threshold, while a Butterworth filter is one that gradually decreases values across the threshold at a specific rate, and is generated with the formula: $H(u, v) = \frac{1}{1 + (\sqrt{2} - 1) \left(\frac{D(u, v)}{D_0} \right)^{2n}}$. Where n is the order of the filter that determines the rate, and D_0 is the threshold.

Low-Pass Filters

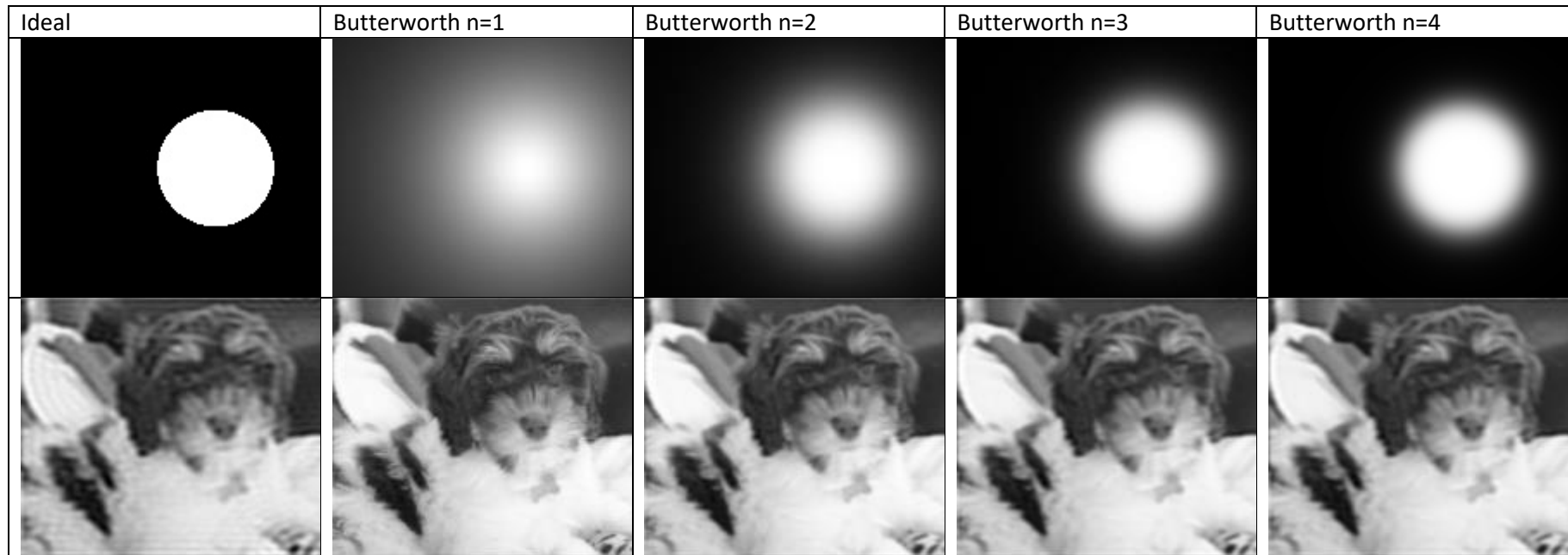


Fig. 11: Effect of Low-Pass filters on 'DSC-9259.JPG'

Fig. 11 (above) demonstrates the effects of zero-ing coefficients using either an ideal filter, or a Butterworth filter. Specifically, this is a low pass filter, so frequencies above a certain level are excluded or lessened.

The first picture demonstrates the effects of an ideal filter on the image. As a result of the filter, there is a definite loss of sharpness, high-frequency information like the fur have been lost. A stranger effect is the striping that is added to the image. It seems to be edge dependent judging by how it adds curved lines that parallel the edges of the white part of the Santa hat.

The Butterworth filters become blurrier as the order n increases, which makes sense considering more high-frequency information is being zeroed out. It also makes sense that the change is barely noticeable for a Butterworth filter with $n=1$, because as the image shows, it's only zeroing out the most high-frequency data, and slightly decreasing high frequency data is difficult to detect.

None of the Butterworth filters show the striping effect present in the ideal filter, which makes me think that the striping effect is the result of the sharp cut-off present in the ideal filter. This is because there is a level of frequency that is kept, and anything above that is completely lost, but since close frequencies are likely near each other on the image, the contrast between visible and lost information would be striking. The gradual change from the Butterworth filters eliminates this issue. I should also not that the striping effect is similar to what I observed when I zeroed out black boxes in the corners of DFTs, which makes since, as I did not gradually zero out there either.

High-Pass Filters

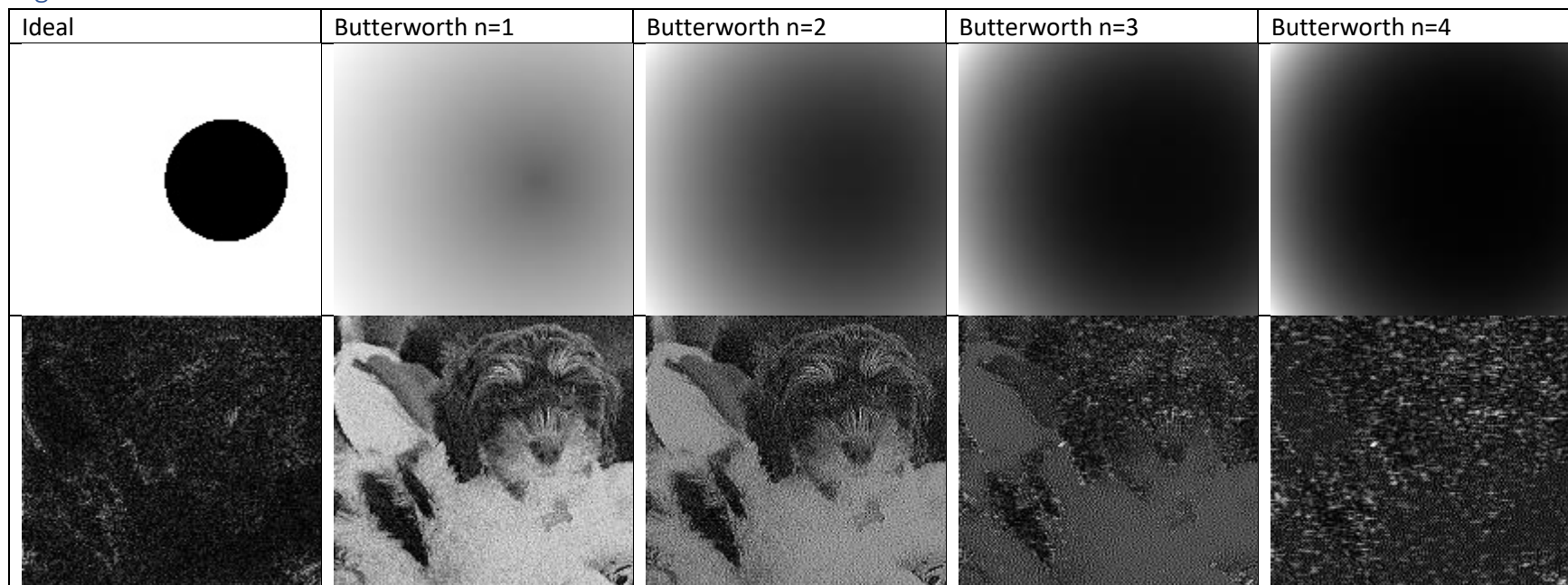


Fig. 12: Effect of High-Pass filters on 'DSC-9259.JPG'

Fig. 12 (Above) shows the effects of zero-ing the low frequency coefficients with both an ideal and Butterworth generated filter. This effect was interestingly very different from the result I got when zero-ing values in a black box.

The first image shows the effect of an ideal filter on the image. This seemed had the effect of turning most of the image black, however, there are still white parts on some edges, I would not consider it an edge detection filter since these edges are all kind of fuzzy, and there's a lot of noise that was not present in the original image. Furthermore, not every major edge was marked, such as the edge between the hat and the couch.

Unlike with the low-pass filtering, the Butterworth filters seem much less useful than the ideal ones. Generally, as the order, n , increased, the level of noise in the image also increased. By the time $n=4$, the image is completely unrecognizable. I suppose it makes sense that if emphasizing the low frequencies blurred the image, which reduced the noise, removing the low frequencies would emphasize the noise. Subtracting these results as an array from the original image as array would result in images similar to the low-pass filter results.