

**CS 474 - Image Processing and Interpretation**

**Fall 2021 - Dr. George Bebis**

**Programming Assignment 4 Report**

**Written by Joseph Trierweiler and Logan Leavitt**

**Joseph was responsible for Experiment 2 and Experiment 4**

**Logan was responsible for Experiment 1 and Experiment 3**

# Experiment 1

## Theory

In this experiment, it is demonstrated that filtering can be effectively done in the frequency domain. For noise which is periodic in nature, frequency domain filtering is often better than filtering in the spatial domain. Filtering in the frequency domain can be done via multiplication. That is, given a spectra  $F(u, v)$ , we can obtain  $\hat{F}(u, v)$  via  $\hat{F}(u, v) = H(u, v)F(u, v)$ .

The filters tested in this experiment are band-reject and notch filters. Specifically, these are implemented as Butterworth filters in this experiment. The general form of the band-reject filter is given by,

$$H_{BR}(u, v) = \frac{1}{1 + \left[ \frac{DW}{D^2 - D_0^2} \right]^{2n}}$$

Where  $D_0$  is the cutoff frequency,  $W$  is the width,  $n$  is the order of the filter, and  $D = \sqrt{u^2 + v^2}$ . The Butterworth notch filter, on the other hand, can be constructed as a product of high-pass filters that have been shifted to the specified “notches”. Each individual high-pass filter is of the form,

$$H_{HP}(u, v) = \frac{1}{1 + \left[ \frac{D_0}{D} \right]^{2n}}$$

Where  $D_0$ ,  $D$ , and  $n$  are defined the same. A notch filter with two notches at  $(u_0, v_0)$ ,  $(-u_0, -v_0)$  would then be of the form,

$$H_N(u, v) = H_{HP}(u - u_0, v - v_0)H_{HP}(u + u_0, v + v_0)$$

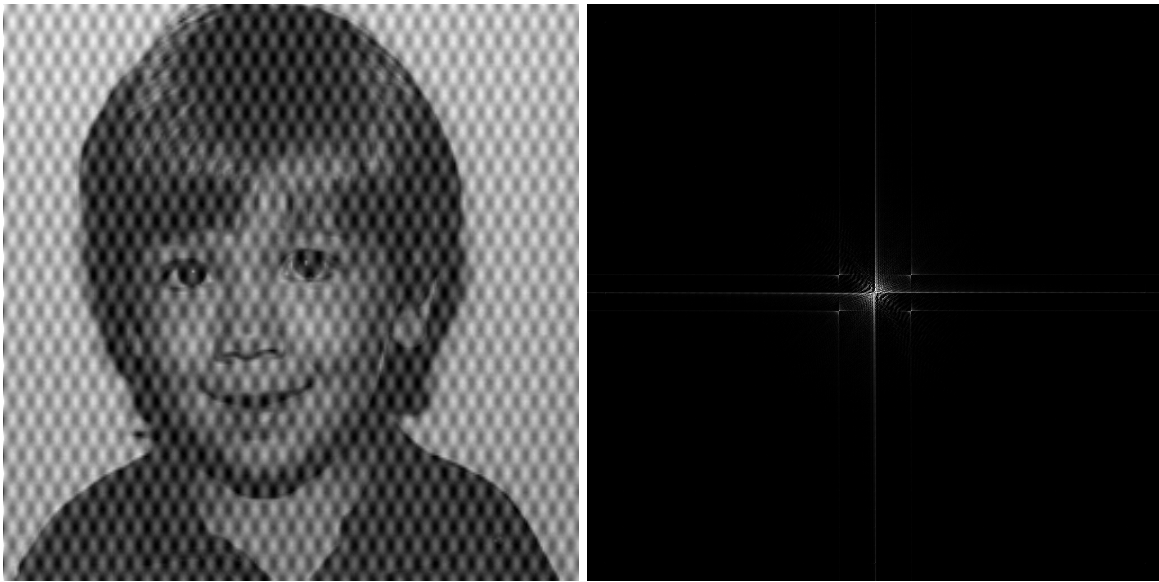
## Implementation

To implement frequency domain filtering, the 2D FFT code from project 3 is utilized. The general procedure to filter the image is: pad the image with zeroes, apply the 2D fft, multiply by  $H(u, v)$ , and take

the inverse 2D FFT. In addition, the spectra is centered by multiplying by  $(-1)^{i+j}$  in the spatial domain. Gaussian smoothing was also required to be used for this project. Code from an earlier project was utilized to obtain those results. The code for the band-reject filter can be found in `q1/band.cpp`, for the notch filter in `q1/notch.cpp`, for Gaussian smoothing in `q1/gaussian.cpp`, and for extracting the noise component in `q1/noise.cpp`.

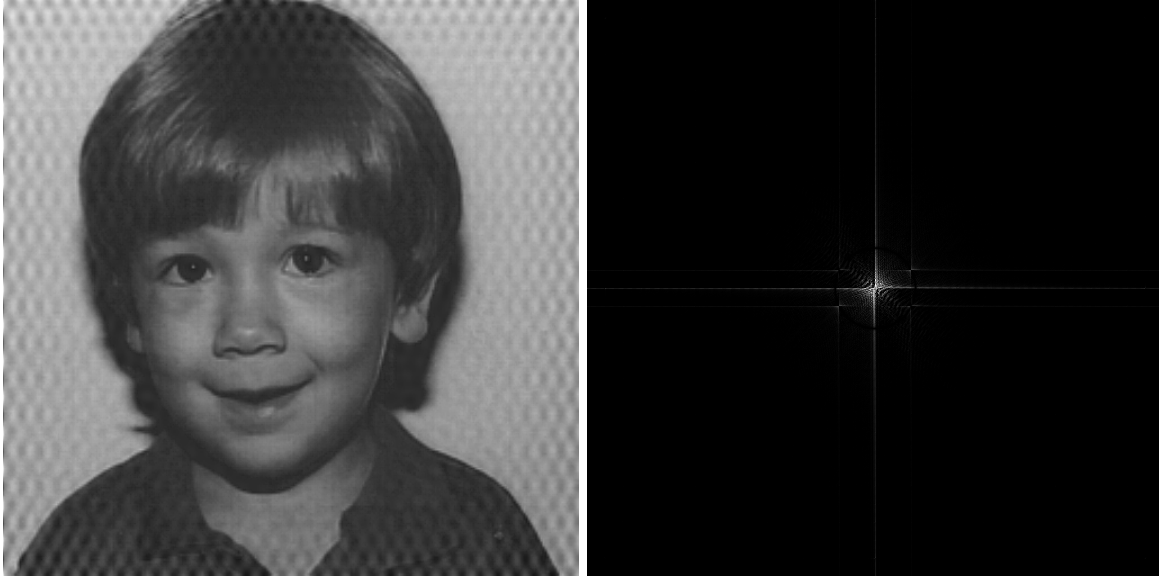
### *Results and Discussion*

Shown in Figure 1 is the noisy image used for this experiment, and its corresponding spectra. By analyzing the image, we can see spikes in the spectra at  $(32, 64)$ ,  $(-32, 64)$ ,  $(32, -64)$ , and  $(-32, -64)$  (relative to the center of the image). Using this information, we can hypothesize that removing these spikes will eliminate the noise from the image.



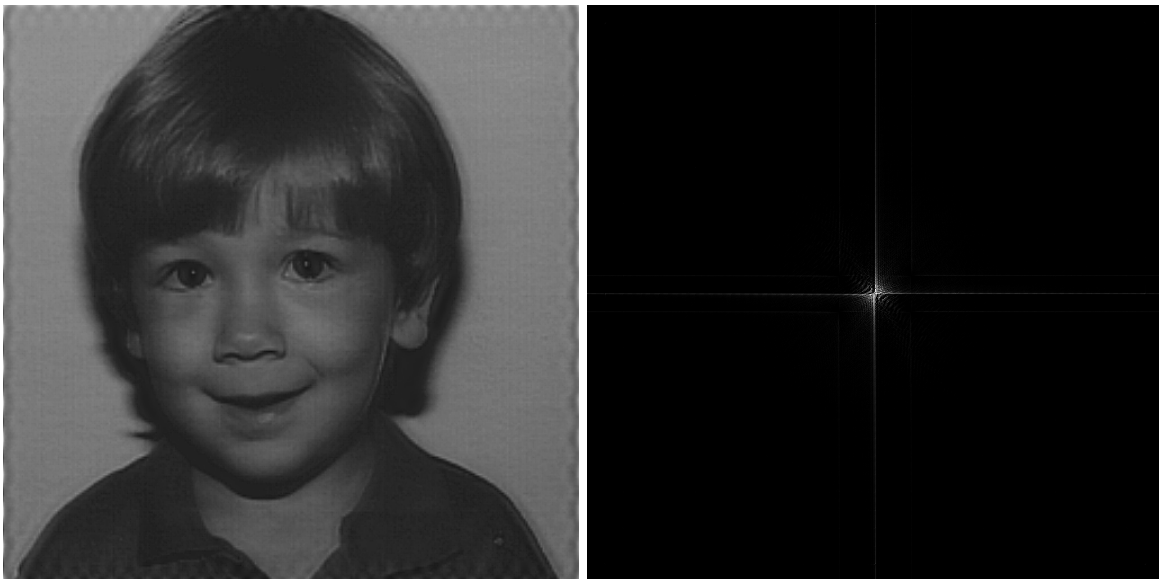
**Figure 1: boy\_noise.pgm and its corresponding spectra.**

To eliminate the noise, both a band-reject and notch-reject filter was tested. Figure 2 showcases the results of a Butterworth band-reject filter. As can be seen, this band-reject filter removes noise from the center of the image, but leaves a significant amount of noise around the edges.



**Figure 2: Results of filtering boy\_noisy.pgm with a Butterworth band-reject filter. The order of the filter is 1, and the width of the band is 5. Using an order greater than 1 produced a ringing effect.**

Figure 3 shows the results of a Butterworth notch-reject filter. The results are an improvement from the band-reject filter.



**Figure 3: Results of filtering boy\_noisy.pgm with a Butterworth notch-reject filter. The order of the filter is 1, and  $D_0$  was chosen to be 20. Using an order greater than 1 produced a ringing effect.**

How does this compare with Gaussian smoothing? The results of Gaussian smoothing with a 7x7 and 15x15 filter are shown in Figure 4. Gaussian smoothing produces a better image of the boy's face, but cannot totally remove noise without significantly blurring the image.



**Figure 4: Gaussian smoothing applied to boy\_noise.pgm with a 7x7 (left) and a 15x15 (right) filter.**

To instead extract noise from the image, the notch filter used in Figure 3 was adapted to function as a notch-pass filter, allowing only the noise to pass through. That is, if  $H(u, v)$  is the notch-reject filter, we can multiply in the frequency domain by  $1 - H(u, v)$  to extract the noise. The results of this are shown in Figure 5.

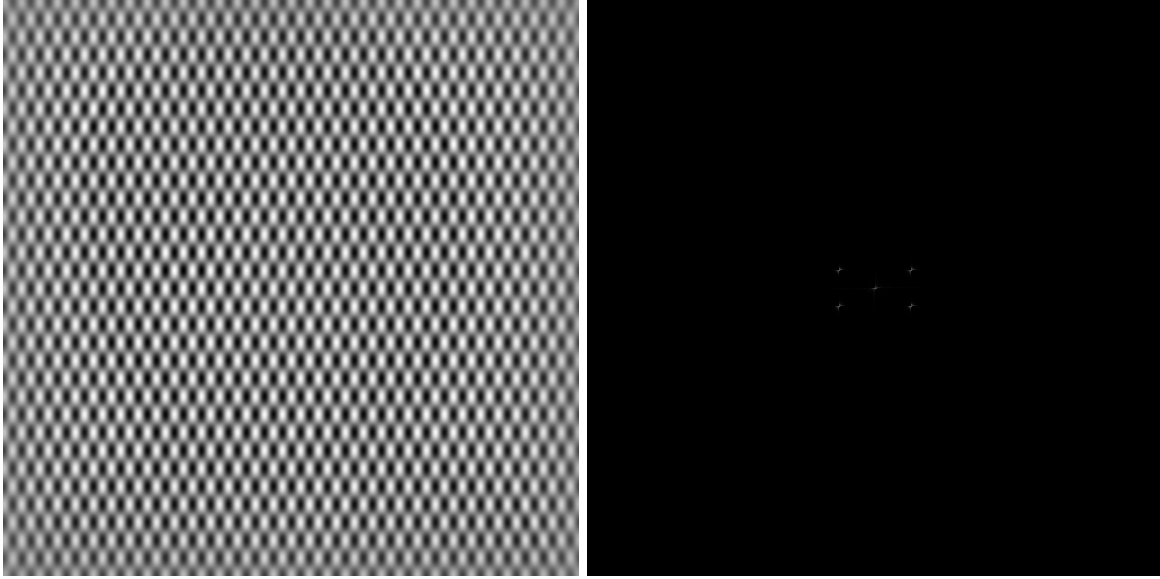


Figure 5: Noise extracted using a Butterworth notch-pass filter. (Noise image is rescaled to [0,255] for visibility)

## Experiment 2

### *Theory*

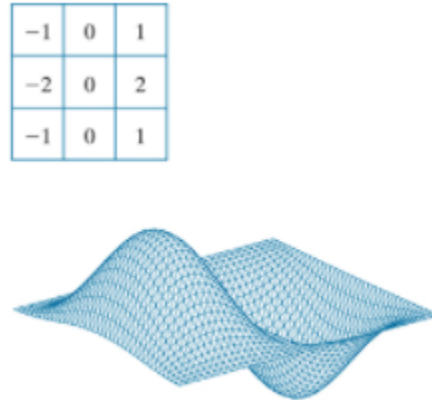
The convolution theorem states that if two functions are convoluted in the spatial domain, then they are equal to the same two functions being convoluted in the frequency domain. A mathematical expression of the theorem can be found below:

$$f(x, y) * h(x, y) \Leftrightarrow F(u, v)H(u, v)$$

This is the case for sampled functions and discrete convolutions, however in the discrete case sequences must be padded with zeros to avoid overlap. What this experiment intends to prove is that it's possible to obtain the same results regardless of what method one chooses.

### *Implementation*

The convolution theorem can be proven via convoluting lenna.jpg with spatial filtering, then with frequency filtering, and then comparing results. Spatial filtering can be accomplished with the Sobel mask, shown below:



**Figure 6: A Sobel Mask represented in both the spatial domain (top) and the frequency domain (bottom)**

For spatial filtering, we re-used our implementation from Project two, where we applied the same sobel filter in the spatial domain. For frequency filtering we will transform the sobel mask into a mask of the same size as the original image, taking care to preserve its original center. Afterwards we will perform a forward fourier transform on both the mask and the image, and proceed to pointwise multiply the two frequencies. When the image is returned to the spatial domain it should have almost identical results as the transformation applied in our spatial filtering experiment.

### *Results and Discussion*

Unfortunately I was not able to get the same results as the textbook. In figure 7 I show both the results of my attempt at applying the transformation, as well as what the Sobel mask looks like when applied in the spatial domain. The results are supposed to look similar, so I assume that I have implemented one of the instructed steps incorrectly.



Figure 7: Our results applying a Sobel mask in the Frequency domain (left) versus applying a Sobel mask in the Spatial Domain (right)

### Experiment 3

#### *Theory*

This experiment assumes the following model for image degradation.

$$g(x, y) = H[f(x, y)] + \eta(x, y)$$

Where  $H$  models the degradation process, and  $\eta(x, y)$  is the additive noise. If we assume that  $H$  is linear and shift-invariant, then this simplifies to,

$$g(x, y) = f(x, y) * h(x, y) + \eta(x, y)$$

Where  $h(x, y) = H[\delta(x, y)]$  is the impulse response. We can then consider this equation in the frequency domain.

$$G(u, v) = H(u, v)F(u, v) + N(u, v)$$

If we consider the case when there is no noise, we can easily obtain  $F(u, v)$  from  $G(u, v)$  by dividing by  $H(u, v)$ . This is simply Inverse filtering. However, since Inverse filtering ignores the noise component,



dividing by  $H(u, v)$  can cause the noise component to dominate the image when  $H(u, v)$  is small. To get around this, we can only apply Inverse filtering in a certain radius.

Wiener filtering, on the other hand, improves upon Inverse filtering by approximating the noise component. Wiener filtering is given by the formula,

$$F(u, v) = \frac{|H(u, v)|^2}{|H(u, v)|^2 + K} \cdot \frac{G(u, v)}{H(u, v)}$$

To prevent division by zero, we can also note that  $H(u, v)H^*(u, v) = |H(u, v)|^2$ , and rewrite the equation as,

$$F(u, v) = \frac{H^*(u, v)}{|H(u, v)|^2 + K} \cdot G(u, v)$$

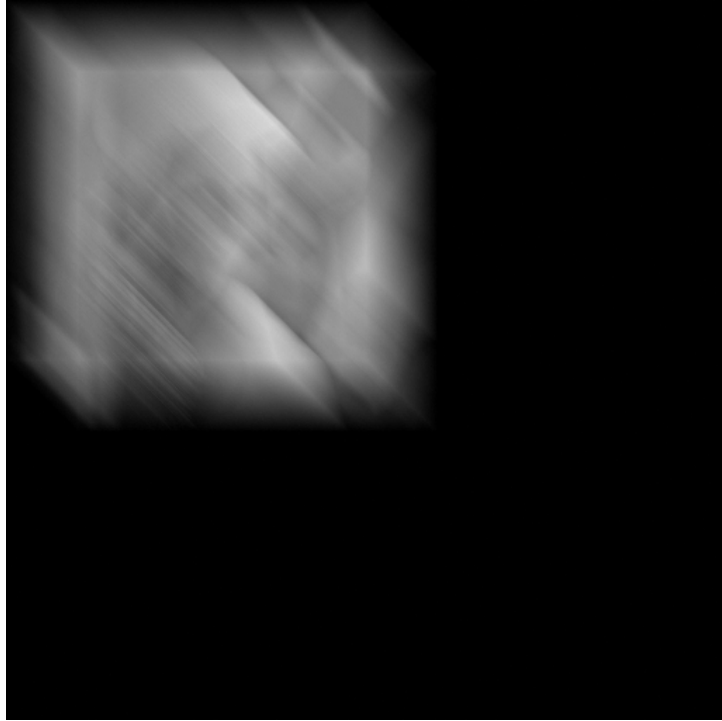
Where  $H^*(u, v)$  is the conjugate.

### *Implementation*

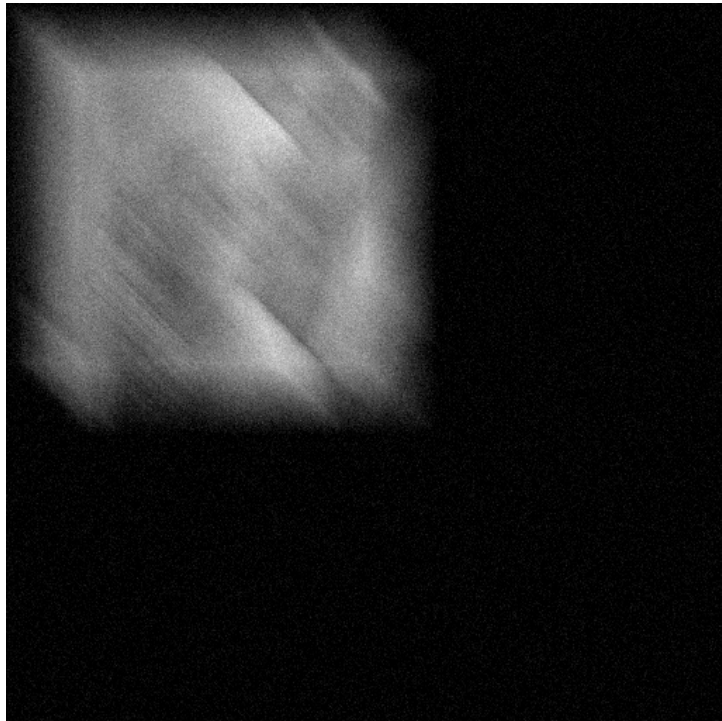
The code for this experiment is very similar to the code used for Experiment 1. The only differences are in how the image is filtered in the frequency domain. For Inverse filtering and Wiener filtering, the image is filtering using the equations described in the *Theory* section. However, the image was also modified in the spatial domain by adding a noise component. As specified, the noise was generated using `boxmuller.c`. The code for degrading the image can be found in `q3/degrade.cpp`, for Inverse filtering in `q3/inverse.cpp`, and for Wiener filtering in `q3/wiener.cpp`.

### *Results and Discussion*

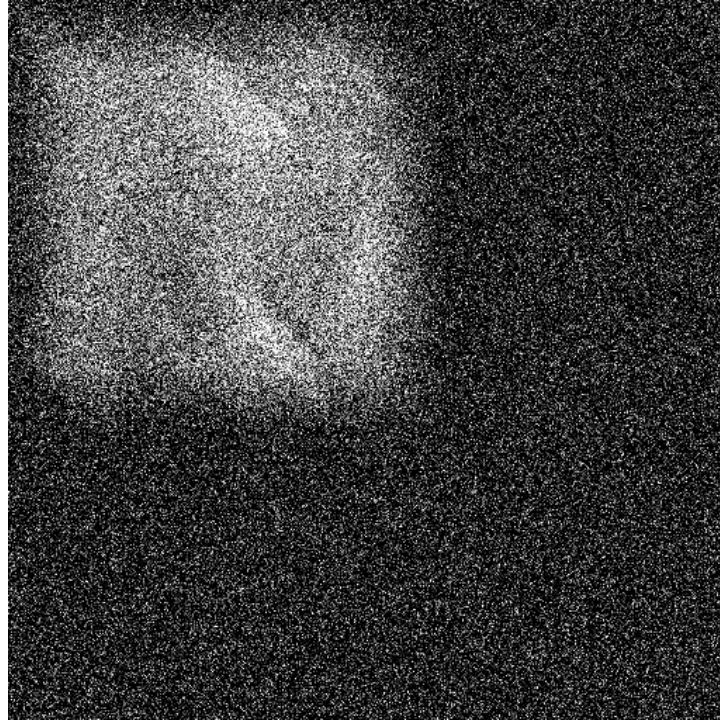
The filter  $H(u, v)$  was applied to the image, and Gaussian noise was added using the Box-Muller code with  $(\mu, \sigma) = (0, 1), (0, 10), (0, 100)$ .



**Figure 8: Degraded image with  $\sigma = 1$**

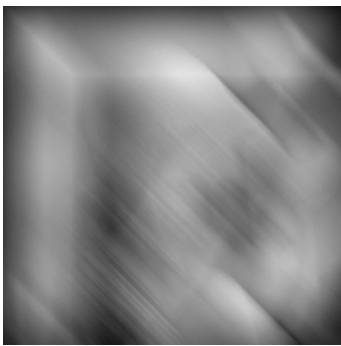
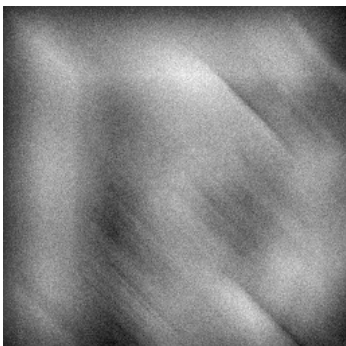
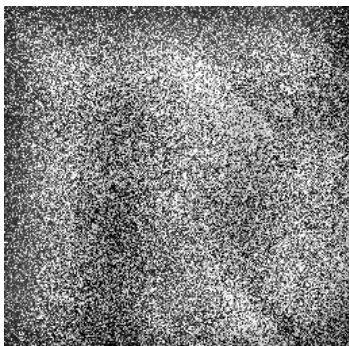


**Figure 9: Degraded image with  $\sigma = 10$**



**Figure 10: Degraded image with  $\sigma = 100$**

Inverse filtering was then applied to each of these images. However, since the specified filter  $H(u, v)$  has several zeroes, dividing by  $H(u, v)$  quickly causes the values to blow up. See the results in Figure ? when inverse filtering is applied on a radius (R) of 6 and 10.

	$\sigma = 1$	$\sigma = 10$	$\sigma = 100$
R=6			

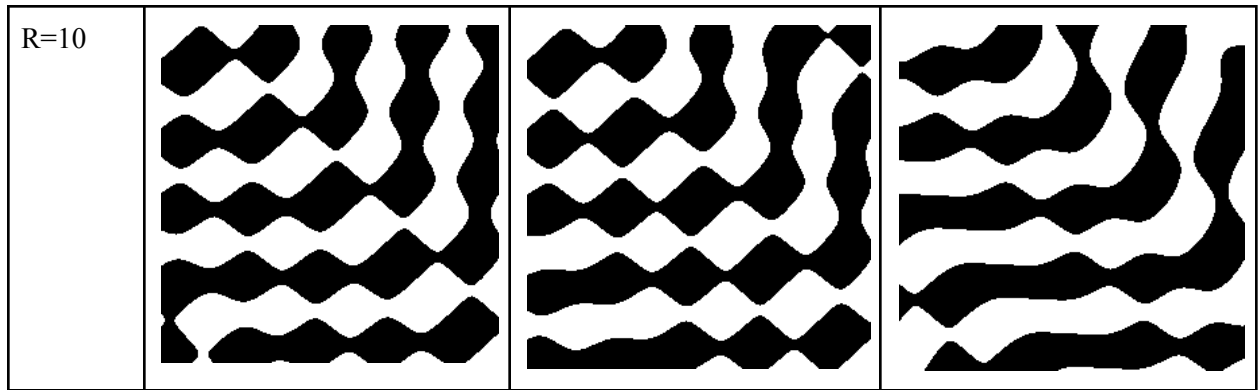


Figure 11: Results of directly applying Inverse Filtering

When  $R=10$ , we can see that some zeroes of  $H(u, v)$  are included, and division by 0 causes the values of the image to blow up. However, we can remedy this by effectively ignoring the case when  $H(u, v) = 0$ . To account for floating point precision, we consider the case when  $|H(u, v)| < \epsilon$ , for a small  $\epsilon$ . In that case, we simply don't divide by  $H(u, v)$ . In this experiment,  $\epsilon$  was chosen to be  $10^{-7}$ . Ignoring the zeroes of  $H(u, v)$  allows us to significantly increase the radius used, improving results.

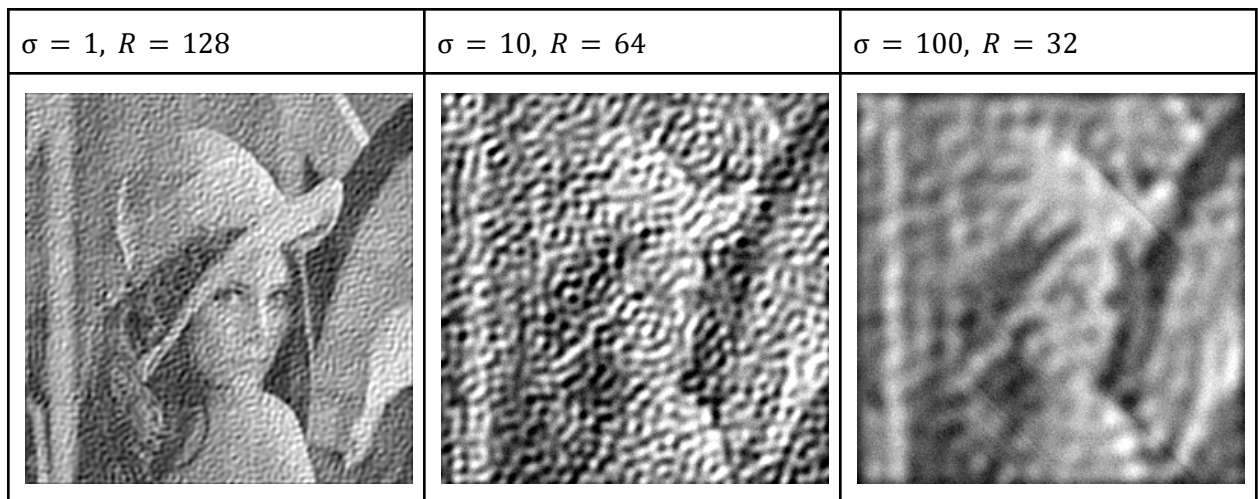

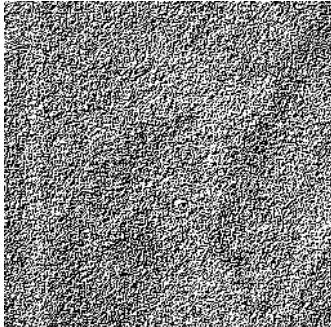
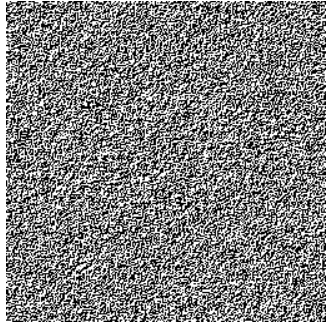


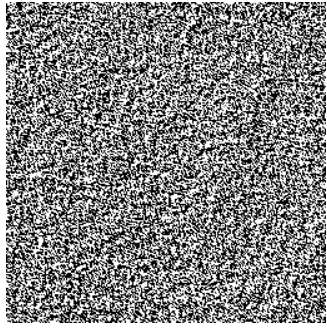


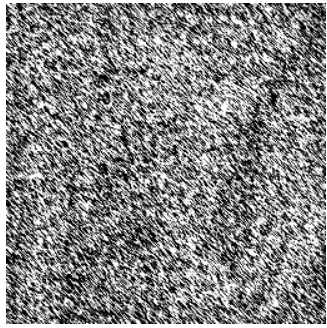
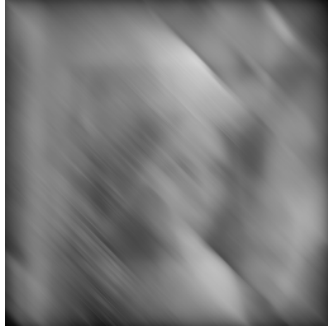

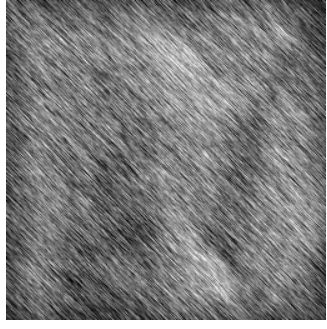


Figure 12: Results of Inverse filtering when the zeroes of  $H(u, v)$  are ignored. Radius is adjusted for each individual case.

One can note that there is an inverse relationship between the optimal radius, and the size of the noise added to the image. As the amount of noise increases, the radius needs to be smaller to prevent the noise from dominating the image.

On the other hand, Wiener filtering did not require ignoring the zeroes of  $H(u, v)$  to get good results. See Figure 13 for the results of Wiener filtering for various  $K$ .



	$\sigma = 1$	$\sigma = 10$	$\sigma = 100$
$K = 0.0001$			
$K = 0.001$			
$K = 0.01$			

$K = 0.1$			
-----------	---	--	---

**Figure 13: Results of Wiener filtering for various K**

Interestingly, the optimal K looks to be proportional to the amount of noise in the image. This is expected, since K is an approximation of the noise component in Wiener filtering.

The optimal results of Wiener Filtering are compared with the optimal results of Inverse Filtering in Figure 14. As expected, Wiener filtering produces results which are less affected by noise than Inverse filtering.

	Inverse filtering	Wiener filtering
$\sigma = 1$		

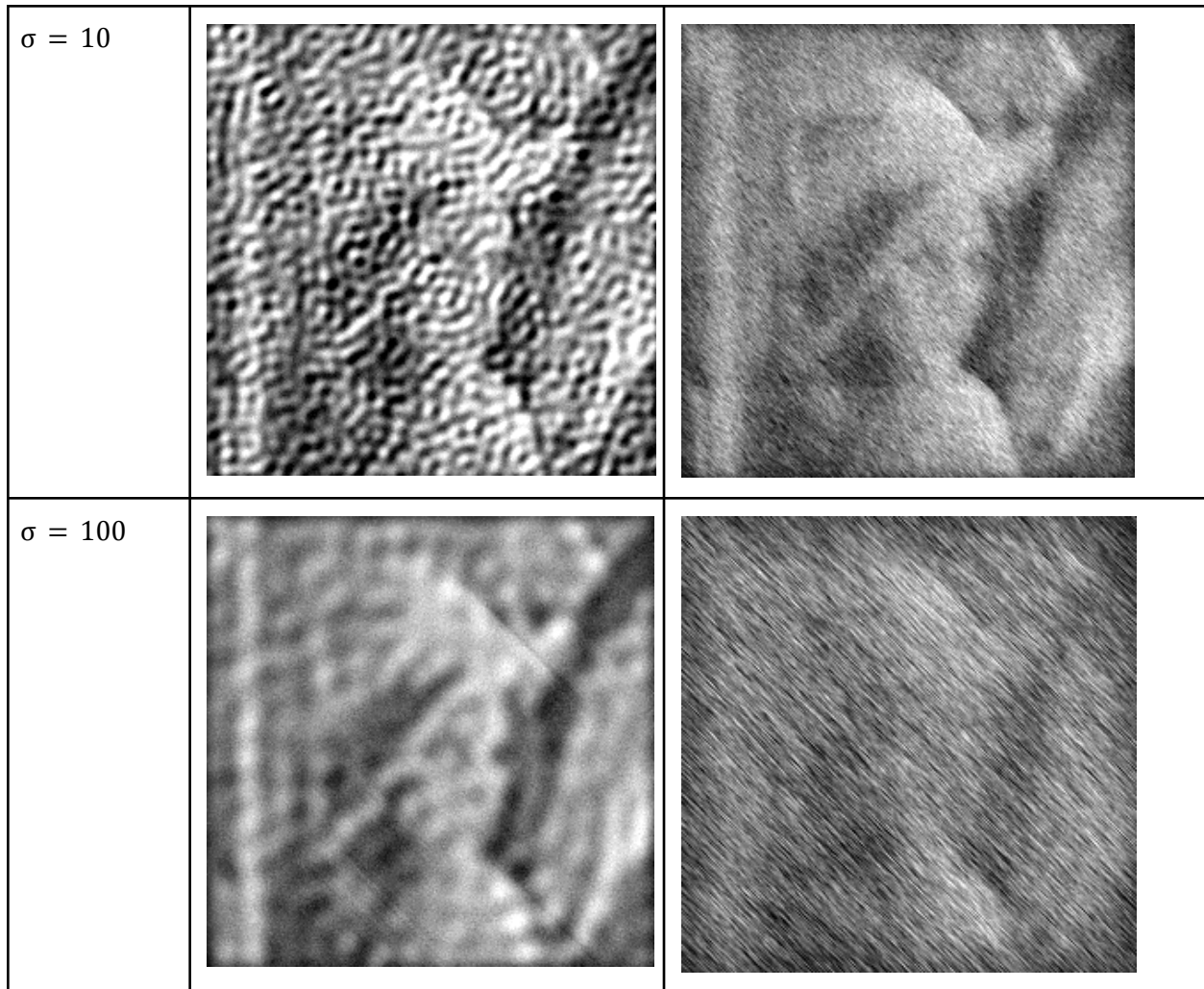


Figure 14: Comparison of the results of Inverse and Wiener filtering

## Experiment 4

### *Theory*

Homomorphic filtering is a frequency filtering method that excels at removing shadows due to uneven illumination. It can both enhance high frequency wavelengths, while still preserving fine detail with low frequencies. It does this through an illumination component and a reflection component from an image, where illumination relates mostly to low frequencies and reflection to high frequencies. This formation

can be represented mathematically below, where ‘i’ is the illumination component and ‘r’ is the reflection component:

$$f(x, y) = i(x, y) r(x, y) \quad F(u, v) = I(u, v) * R(u, v)$$

**In the Spatial Domain**

**In the Frequency Domain**

In order to manipulate the illumination and reflection values found in the frequency domain, they first must be de-convoluted. This can be done by taking the log of the image’s values in the spatial domain. Afterwards we can apply a specific filter to attenuate the lighting of an image as we see fit. For this experiment we will be demonstrating the effects of a high emphasis filter shown below:

$$H(u, v) = (\gamma_H - \gamma_L)[1 - e^{-c([D(u, v)]^2 / D_0^2)}] + \gamma_L$$

where  $D_0$  is the cutoff frequency of the filter and  $\gamma_L$  and  $\gamma_H$  are the gains for the low and high frequencies correspondingly and  $c = 1$ . The high emphasis filter is a high pass filter, meaning it will emphasize high frequencies and attenuate lower ones, preserving fine detail at the same time.

### *Implementation*

In order to accomplish homomorphic filtering we must first apply a logarithmic transformation on the input image. Then after performing a forward fourier transform, we must take care to center the image in the frequency domain. We can do that with the frequency-shift transformation, and then we can apply our high emphasis filter. For this experiment we will test different combinations of  $\gamma_H$  and  $\gamma_L$  as discussed in our theory section. The ranges tested were  $\gamma_H = [1.25-1.75]$  and  $\gamma_L = [0.25-0.75]$  The cutoff frequency we chose for all combinations was a constant 1.8.



### *Results and Discussion*

Below are the results of our testing. It seems as though for an overexposed image, such as the original image, a low Gamma H and a higher Gamma L value, such as the combination found in Figure 7(c) worked best. It seems as Gamma H decreases, the overexposure found in the original image is diminished. It also seems as Gamma L increases, the contrast of the resulting image increases.



**Original Image**



**(a) Gamma H = 1.25 Gamma L = 0.25 (b) Gamma H = 1.25 Gamma L = 0.5 (c) Gamma H = 1.25 Gamma L = 0.75**



(d) Gamma H = 1.5 Gamma L = 0.25 (e) Gamma H = 1.5 Gamma L = 0.5 (f) Gamma H = 1.5 Gamma L = 0.75



(g) Gamma H = 1.75 Gamma L = 0.25 (h) Gamma H = 1.75 Gamma L = 0.5 (i) Gamma H = 1.75 Gamma L = 0.75

Figure 15 shows all the configurations of the high emphasis filter tested in this experiment