RBE549 - Midterm Exam

Keith Chester

Due date: October 16, 2022

Problem 1

In this problem we are presented with a Hough transform problem, wherein x, y, b, and m are positive or negative real numbers. 2 points are given in (m, b) space by $P_1 : (m, b) = (0.5, 4)$ and $P_2 : (m, b) = (0.9, 0)$.

A

What line L_1 in (m, b) space through P_1 and P_2?

We can set the line equation ($y = mx + b$) for P_1 and P_2 equal to eachother to solve this.

$$P_1 = P_2 \tag{1}$$

$$0.5x + 4 = 0.9x + 0 \tag{2}$$

$$4 = 0.4x \tag{3}$$

$$10 = x \tag{4}$$

...and then we plug in x to solve for y:

$$0.5(10) + 4 = y \tag{5}$$

$$y = 9 \tag{6}$$

...thus the equation for L_1, a line that passes through points P_1 and P_2, is $b = -10m + 9$.

B

The point P_3 in (x, y) space that corresponds to line L_1 is the point $(10, 9)$, as calculated above in part A. We can also just deduce it from our solved equation mentioned in A - specifically that since $y = mx + b$ for $b = -mx + y$, and we stated that $b = -10m + 9$, we can also deduce that $(x, y) = (10, 9)$.

C

Since it is a horizontal line, which means the slope m must be $m = 0$, we now know that the resulting point, P_4, must fall along the line we calculated earlier, L_1, specifically where $m = 0$, or the vertical axis of (m, b) space. Thus we can solve:

$$b = -10m + 9 \tag{7}$$

$$b = -10 * 0 + 9 \tag{8}$$

$$b = 9 \tag{9}$$

...thus we know that P_4 will fall on $(m, b) = (0, 9)$. This means that the equation for L_2 is $y = 9$.

Problem 2

In this problem we are aiming to discover a structuring element SE such that $II \bigoplus SE = OI$

A

A $3x3$ structing element SE that satisfies the given II and OI would be:

$$\begin{array}{|c|c|c|} \hline 1 & 0 & 1 \\ \hline 0 & \mathbf{0} & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array} \tag{10}$$

...where the central 0 is the point of origin for the SE.

B

Another structuring element SE_2 can satisfy $II \bigoplus SE_2 = II$ but $SE \neq SE_2$, where again the central value (now a 1) is the point of origin::

$$\begin{array}{|c|c|c|} \hline 1 & 0 & 1 \\ \hline 0 & \mathbf{1} & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array} \tag{11}$$

Problem 3

In this problem we are solving equations having to do with image focus and lenses.

A

In this problem, we attempt to prove with similar triangles that:

$$\frac{1}{-z_0} + \frac{1}{z_c} = \frac{1}{f} \tag{12}$$

First, we look at the original diagram provided. I have added markers for the lengths (z_0 and z_c) assigned to their lengths, and marked the two lengths of f to its distance from the center of the lens.

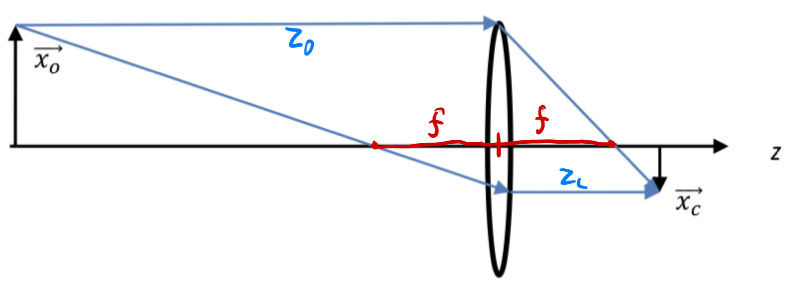

Figure 1: Thin Lens Diagram

We will note that the triangle provided by z_0 and be represented on the other side of the lens by flipping it along the lens as an axis. From there we can begin to mark angles.

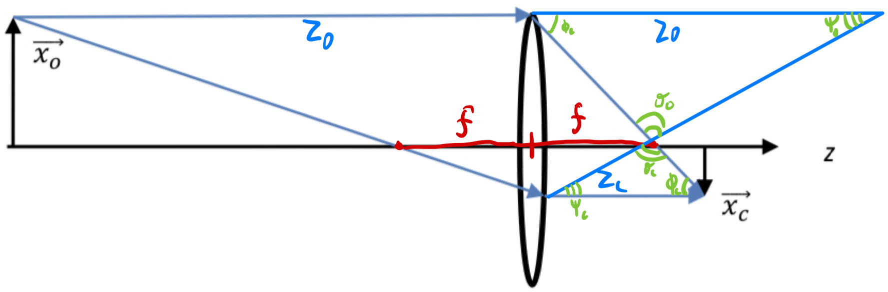

Figure 2: Thin Lens Diagram Expanded

Because the z access, z_c, and z_0 are parallel lines, we can thus know that opposite angles of lines crossing them are equivalent. Thus we see that ϕ_c and ϕ_0 are equivalent and marked as such. Thesame goes for ψ_c and ψ_0, as marked. Since the lines passing through the focal point f is rom parallel lines and continuous, we can also determine that θ_0 and θ_c are also equivalent.

With this, we know that these are similar triangles, and can begin to form ratios to represent the relationships. Thus we have:

$$\frac{-z_0}{z_c} = \frac{f}{z_c - f} \tag{13}$$

From here we can perform algebra to isolate the variables into a manner that is closer to what we sought to originally prove:

$$-z_0 z_c - z_0 f = z_c f \tag{14}$$

$$-z_0 z_c = z_c f - z_0 f \tag{15}$$

$$-z_0 z_c = f(z_c - z_0) \tag{16}$$

$$\frac{-z_0 z_c}{f} = z_c - z_0 \tag{17}$$

$$\frac{1}{f} = \frac{z_c - z_0}{-z_0 z_c} \tag{18}$$

$$\frac{1}{f} = \frac{z_c}{-z_0 z_c} + \frac{z_0}{z_0 z_c} \tag{19}$$

$$\frac{1}{f} = \frac{1}{-z_0} + \frac{1}{z_c} \tag{20}$$

B

In this problem we are tasked with finding the number of receptors on which the image of Mars falls while looking up into the night sky. We say that a human eyeball has a radius of $12mm$ and contains roughly 1.5×10^8 receptors. We assume that the receptors are uniform across a $160°$ partial sphere in the back of our eye. The planet of Mars has a $4 \times 10^3 km$ radius with an average distance of $2.25 \times 10^8 km$. We use an f focal value equal to the eye's diameter ($24mm$).

To solve this, first we use similar triangles using a lens diagram (similar to the one provided for the question in $3A$); this allows us to determine the size of the image of Mars on our eye.

$$\frac{z_0}{z_c} = \frac{h_0}{h_1} \tag{21}$$

In this we have z_0 as the distance from Mars to our eye, and h_0 the width of Mars, and h_i is the size of the image of Mars on our eye. We need to solve for z_c before we can figure out h_1, so to do this we look at the equation utilized earlier:

$$\frac{1}{-z_0} + \frac{1}{z_c} = \frac{1}{f} \tag{22}$$

$$\frac{1}{-2.25 \times 10^8 km} + \frac{1}{z_c} = \frac{1}{24mm} \tag{23}$$

$$\frac{1}{z_c} \approx 83.33 + 4.44 \times 10^{-9} \tag{24}$$

$$z_c \approx 0.024 \approx 24mm \tag{25}$$

Now we can determine the size of the image of Mars on our eye:

$$\frac{z_0}{z_c} = \frac{h_0}{h_1} \tag{26}$$

$$\frac{2.25 \times 10^8 km}{12mm} = \frac{8,000km}{h_1} \tag{27}$$

$$h_1 = 8.53 \times 10^{-7} m = 8.53 \times 10^{-4} \tag{28}$$

...which is an incredibly tiny size. So let's figure out how many receptors we have per square mm of our eye. First we need to find the area of a spherical cap, such that we can determine the average number of receptors per mm in the eye. We know that the area of a spherical cap (S) is:

$$S = 2rh\pi \tag{29}$$

...where h is equal to:

$$h = r - \sqrt{r^2 - a^2} \tag{30}$$

...but because of the partial sphere, a is not straight but rather angled to reach the 160°; thus we have an actual interior angle of 80°, so $a = r * \cos(10°)$

$$h = 12mm - \sqrt{(12mm)^2 - (12mm * \cos(10°))^2} \tag{31}$$

$$h \approx 0.00992 \approx 9.92mm \tag{32}$$

Now that we have h, we can solve for S:

$$S = 2rh\pi = 2 * 12mm * 9.92mm * \pi \approx 747.95mm^2 \tag{33}$$

Since we know that there are approximately 1.5×10^8 receptors, we can use a ratio to get the approximate receptors per mm, or r_{mm}:

$$\frac{1mm^2}{r_{mm}} = \frac{747.95mm^2}{1.5 \times 10^8} \tag{34}$$

$$r_{mm} \approx 200,548 \tag{35}$$

Now that we know the number of receptors per mm, we need to find the area that the Mars image sits on. With an image height of 8.53×10^{-7}, which means a radius of half that, or 4.27×10^{-7}, which is our a. Using the area formulas we used earlier:

$$h = r - \sqrt{r^2 - a^2} = 12mm - \sqrt{(12mm)^2 - (4.27 \times 10^{-4})^2} = 7.60 \times 10^{-9} \tag{36}$$

$$S = 2\pi rh = 2\pi(12mm)7.11 \times 10^{-7} mm = 5.73 \times 10^{-7} mm^2 \tag{37}$$

So this area times our known receptors per mm:

$$x_{receptors} = 200,548 \times 5.73 \times 10^{-7} = 0.115 \tag{38}$$

...and now, we have ≈ 0.115 receptors seeing the image of Mars - which is why you can't see Mars with your naked eye on a clear night when it's at the average distance.

Problem 4

In this problem we have an image which has two background objects and background pixels with brightness values that are distributed according to the Rayleigh distribution with parameters σ_b, σ_{o1}, and σ_{o2} with $0 < \sigma_b < \sigma_{o1} < \sigma_{o2}$. The probability of having brightness k is given by:

$$P_b(k) = \frac{k}{\sigma_b^2} e^{\frac{-k^2}{2\sigma_b^2}}, P_{o1}(k) = \frac{k}{\sigma_{o1}^2} e^{\frac{-k^2}{2\sigma_{o1}^2}}, P_{o2}(k) = \frac{k}{\sigma_{o2}^2} e^{\frac{-k^2}{2\sigma_{o2}^2}} \tag{39}$$

We wish to find the decision rule that maximizes the probability of a correct decision. We'll use two thresholds - specifically first we check if the given pixel is b or $o1$. Then a second threshold, which we'll calculate as well, is $o2$.

$$\frac{1}{\sqrt{2\pi}\sigma_{o1}} e^{\frac{-(x-\mu)^2}{2\sigma_{o1}^2}} > \frac{1}{\sqrt{2\pi}\sigma_b} e^{\frac{-(x-\mu)^2}{2\sigma_b^2}} \tag{40}$$

First we will try to isolate our terms by taking the natural log. Since $ln(ab) = ln(a) + ln(b)$ and $ln(\frac{a}{b}) = ln(a) - ln(b)$...:

$$\ln(1) - \ln(\sqrt{2\pi}\sigma_{o1}) + \frac{-(x-\mu)^2}{2\sigma_{o1}^2} > \ln(1) - \ln(\sqrt{2\pi}\sigma_b) + \frac{-(x-\mu)^2}{2\sigma_b^2} \tag{41}$$

$$\ln(\sigma_b) - \ln(\sigma_{o1}) > \frac{-(x-\mu)^2}{2\sigma_{o1}^2} - \frac{-(x-\mu)^2}{2\sigma_b^2} \tag{42}$$

$$\ln(\sigma_b) - \ln(\sigma_{o1}) > \frac{((x-\mu)^2 2\sigma_b^2 - ((x-\mu)^2 2\sigma_{o1}^2}{2\sigma_{o1}^2 2\sigma_b^2 2} \tag{43}$$

$$(\ln(\sigma_b) - \ln(\sigma_{o1}))(2\sigma_{o1}^2 2\sigma_b^2 2) > ((x-\mu)^2 2\sigma_b^2) - ((x-\mu)^2 2\sigma_{o1}^2) \tag{44}$$

$$\frac{(\ln(\sigma_b) - \ln(\sigma_{o1}))(2\sigma_{o1}^2 2\sigma_b^2 2)}{\sigma_b^2 - \sigma_{o1}^2} > (x-\mu)^2 \tag{45}$$

...and finally we can finalize this result by taking the square root of above, resulting in:

$$\sigma_b\sigma_{o1}\sqrt{2\frac{\ln(\sigma_b) - \ln(\sigma_{o1})}{\sigma_b^2 - \sigma_{o1}^2}} > |x-\mu| \tag{46}$$

We will redo this process again for the $o2 > o1$ threshold.

$$\frac{1}{\sqrt{2\pi}\sigma_{o2}} e^{\frac{-(x-\mu)^2}{2\sigma_{o2}^2}} > \frac{1}{\sqrt{2\pi}\sigma_{o1}} e^{\frac{-(x-\mu)^2}{2\sigma_{o1}^2}} \tag{47}$$

$$\ln(1) - \ln(\sqrt{2\pi}\sigma_{o2}) + \frac{-(x-\mu)^2}{2\sigma_{o2}^2} > \ln(1) - \ln(\sqrt{2\pi}\sigma_{o1}) + \frac{-(x-\mu)^2}{2\sigma_{o1}^2} \tag{48}$$

$$\ln(\sigma_{o1}) - \ln(\sigma_{o2}) > \frac{-(x-\mu)^2}{2\sigma_{o2}^2} - \frac{-(x-\mu)^2}{2\sigma_{o1}^2} \tag{49}$$

$$\ln(\sigma_{o1}) - \ln(\sigma_{o2}) > \frac{((x-\mu)^2 2\sigma_{o1}^2 - ((x-\mu)^2 2\sigma_{o2}^2}{2\sigma_{o2}^2 2\sigma_{o1}^2 2} \tag{50}$$

$$(\ln(\sigma_{o1}) - \ln(\sigma_{o2}))(2\sigma_{o2}^2 2\sigma_{o1}^2 2) > ((x-\mu)^2 2\sigma_{o1}^2) - ((x-\mu)^2 2\sigma_{o2}^2) \tag{51}$$

$$\frac{(\ln(\sigma_{o1}) - \ln(\sigma_{o2}))(2\sigma_{o2}^2 2\sigma_{o1}^2 2)}{\sigma_{o1}^2 - \sigma_{o2}^2} > (x - \mu)^2 \tag{52}$$

...and finally we can finalize this result by taking the square root of above, resulting in:

$$\sigma_b \sigma_{o2} \sqrt{2\frac{\ln(\sigma_{o1}) - \ln(\sigma_{o2})}{\sigma_{o1}^2 - \sigma_{o2}^2}} > |x - \mu| \tag{53}$$

Problem 5

In this problem we are looking to perform a number of transforms on the following image:

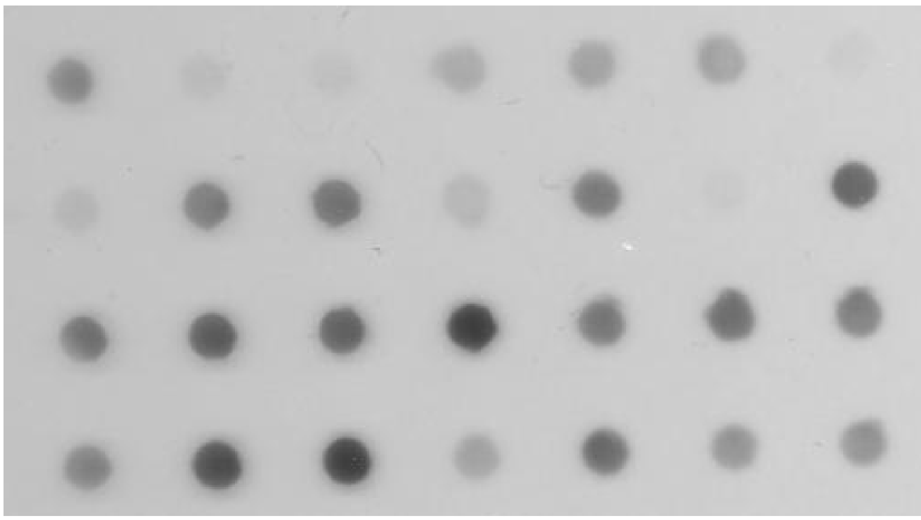

Figure 3: Original Image

All code for these exercises are at the end of this problem.

D

First we compete and find the grayscale image's Sobel edges. The results of this are below:

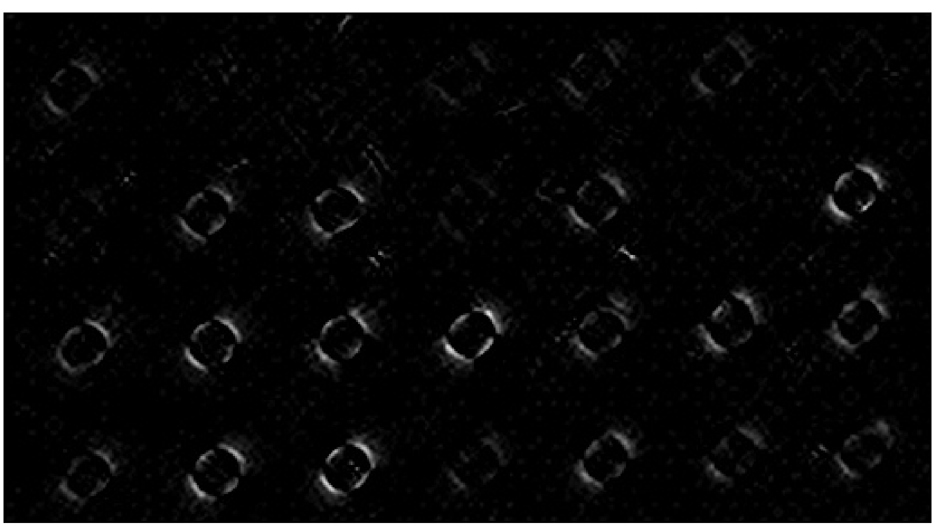

Figure 4: Sobel Edges

E

We are the nasked to find the Marr-Hildreth edges for various values of σ - specifically $[1, 2, 4, 8, 16]$. We get closed contours as the σ raises.

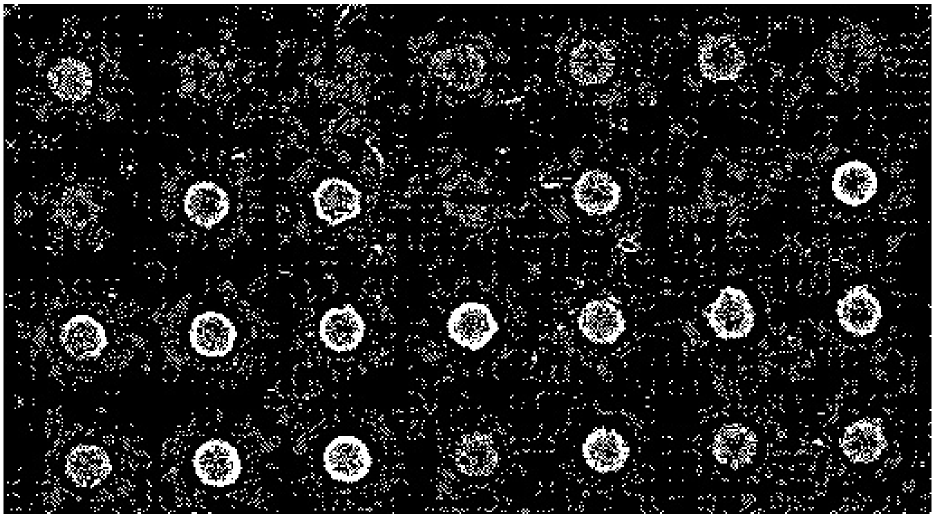

Figure 5: $\sigma = 1$

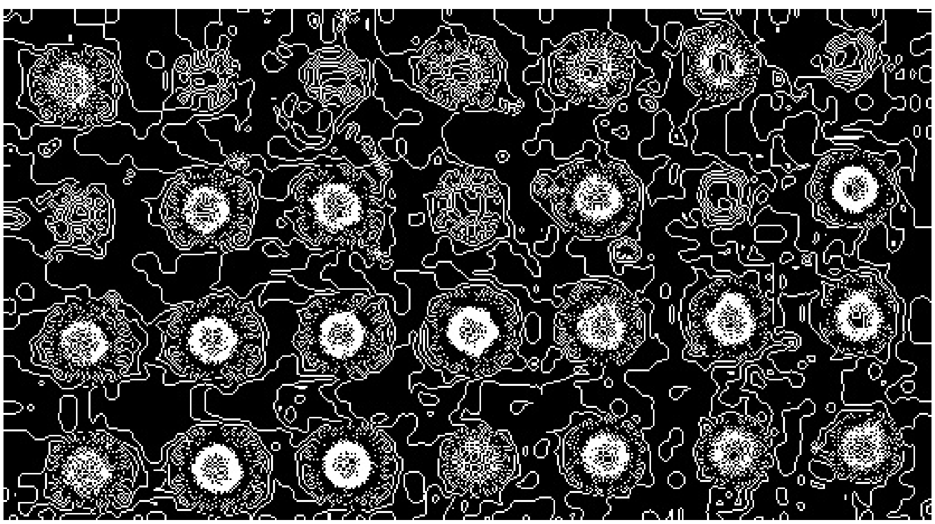

Figure 6: $\sigma = 2$

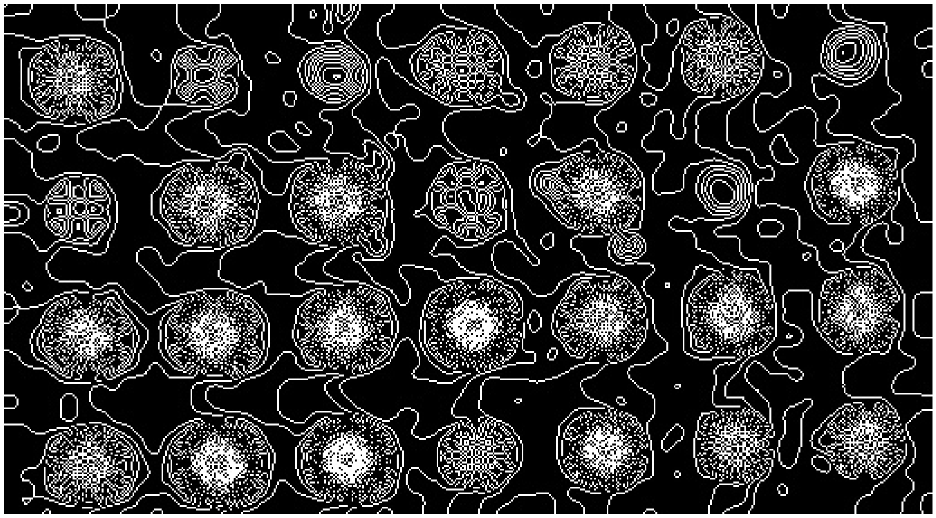

Figure 7: $\sigma = 4$

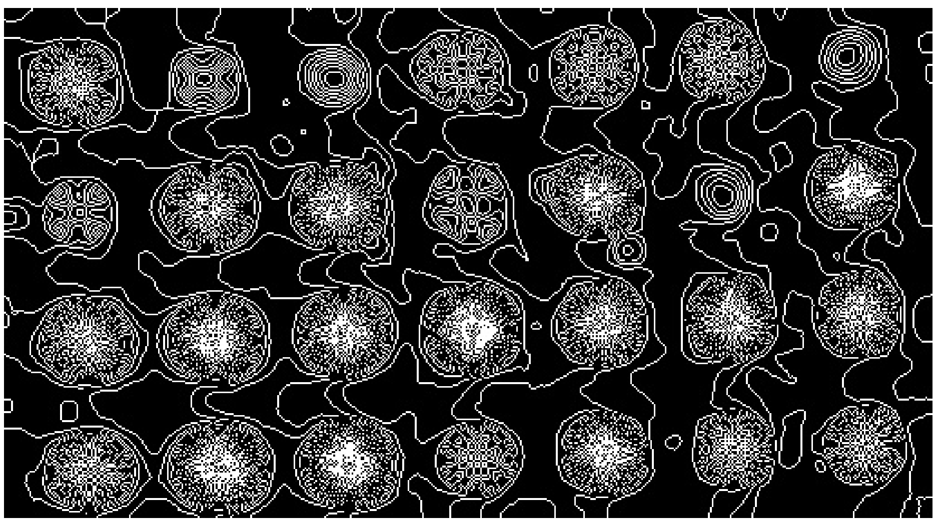

Figure 8: $\sigma = 8$

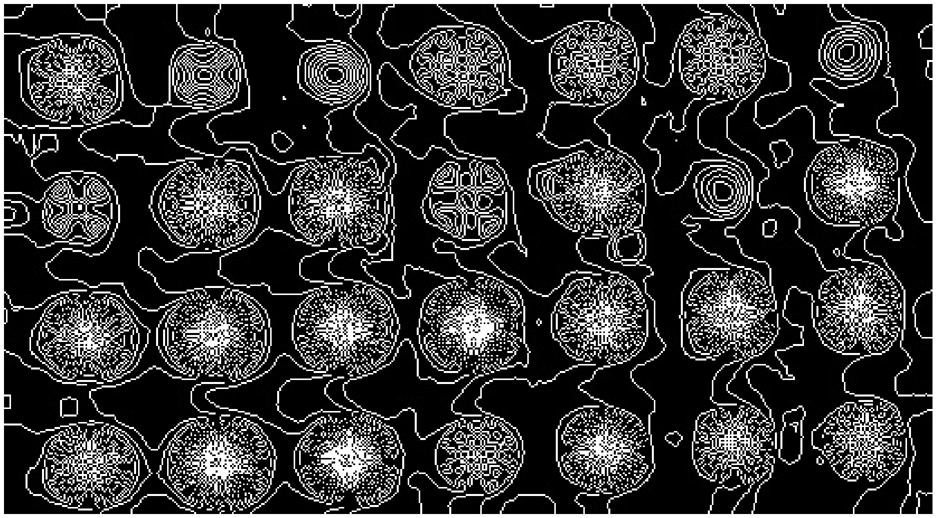

Figure 9: $\sigma = 16$

F

Now we are tasked with designing a filter (I chose a high-pass filter) in the Fourier domain to perform th efunction of edge detection.

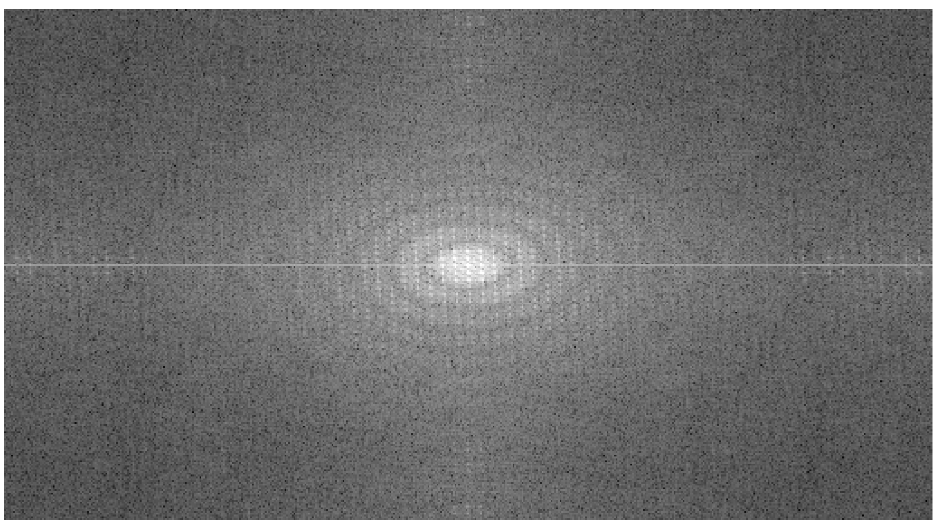

Figure 10: Fourier Transform of our Original Image

We create the high-pass filter by occluding via a mask. I used a circle mask to prevent issues with ringing which a square would have.

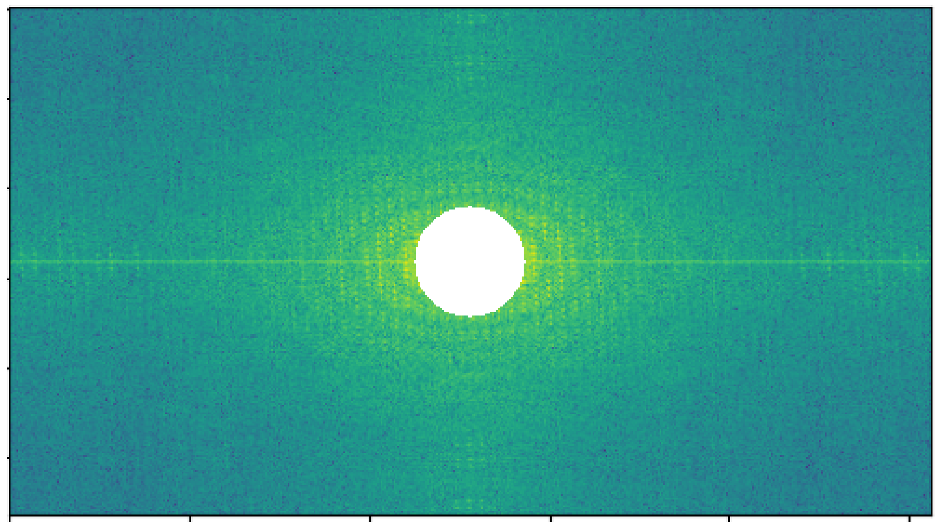

Figure 11: Fourier Transform w/ Mask

The end result is an edge detection of the image:

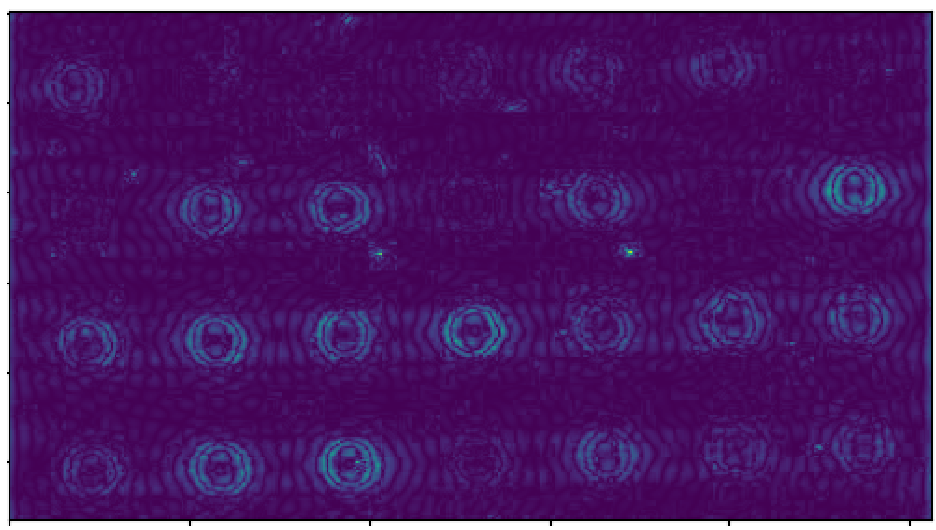

Figure 12: Post Fourier High Pass

Problem 5 Code

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# First load the image
img = cv2.imread("./imgs/dot-blot.jpg", cv2.IMREAD_COLOR)

# # We need the image in grayscale for sobel
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imwrite("./imgs/gray.jpg", img)

# Then it's generally accepted that blurring helps edge
# detection techniques
blurred = cv2.GaussianBlur(img, (3,3), 0)

# Now let's generate the sobel edge detection
sobel = cv2.Sobel(blurred, cv2.CV_64F, dx=1, dy=1, ksize=5)
```

```python
cv2.imwrite("./imgs/sobel.jpg", sobel)

# Now let's show our results
cv2.imshow("Original", img)
cv2.waitKey(0)
cv2.imshow("Sobel", sobel)
cv2.waitKey(0)

# Now let's do Marr-Hildreth edges, which is the Laplace
# of the Gaussian
for sigma in [1, 2, 4, 8, 16]:
    blurred = cv2.GaussianBlur(img, (15,15), sigma)
    mh = cv2.Laplacian(blurred, cv2.CV_8UC1)
    _, threshold = cv2.threshold(mh, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    # Show our result
    cv2.imshow(f"MH @ Sigma = {sigma}", threshold)
    cv2.waitKey(0)
    cv2.imwrite(f"./imgs/mh-{sigma}.jpg", threshold)

# Now let's do the fourier transform filter
dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
dft_shifted = np.fft.fftshift(dft)

magnitude_spectrum = 20 * np.log((cv2.magnitude(dft_shifted[:,:,0], dft_shifted[:,:,1])))

cv2.imwrite("./imgs/dft.jpg", magnitude_spectrum)

rows, cols = img.shape
center = int(rows/2), int(cols/2)
r = 30
mask = np.ones((rows, cols, 2), np.uint8)
x, y = np.ogrid[:rows, :cols]

mask_area = (x- center[0]) ** 2 + (y-center[1]) ** 2 <= r*r
mask[mask_area] = 0

fshift = dft_shifted * mask
fshift_mask_mag = 20 * np.log(cv2.magnitude(fshift[:,:,0], fshift[:,:,1]))
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:,:,0], img_back[:,:,1])

plt.figure()
plt.imshow(fshift_mask_mag)
plt.show()
plt.close()

plt.figure()
plt.imshow(img_back)
plt.show()
plt.close()
```