Different Edge detection Algorithms

Armita Ashabyamin

How did we start?

- We first start by importing numpy and opencv ,and also matplot in order to show our results:

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount('/content/drive')
```

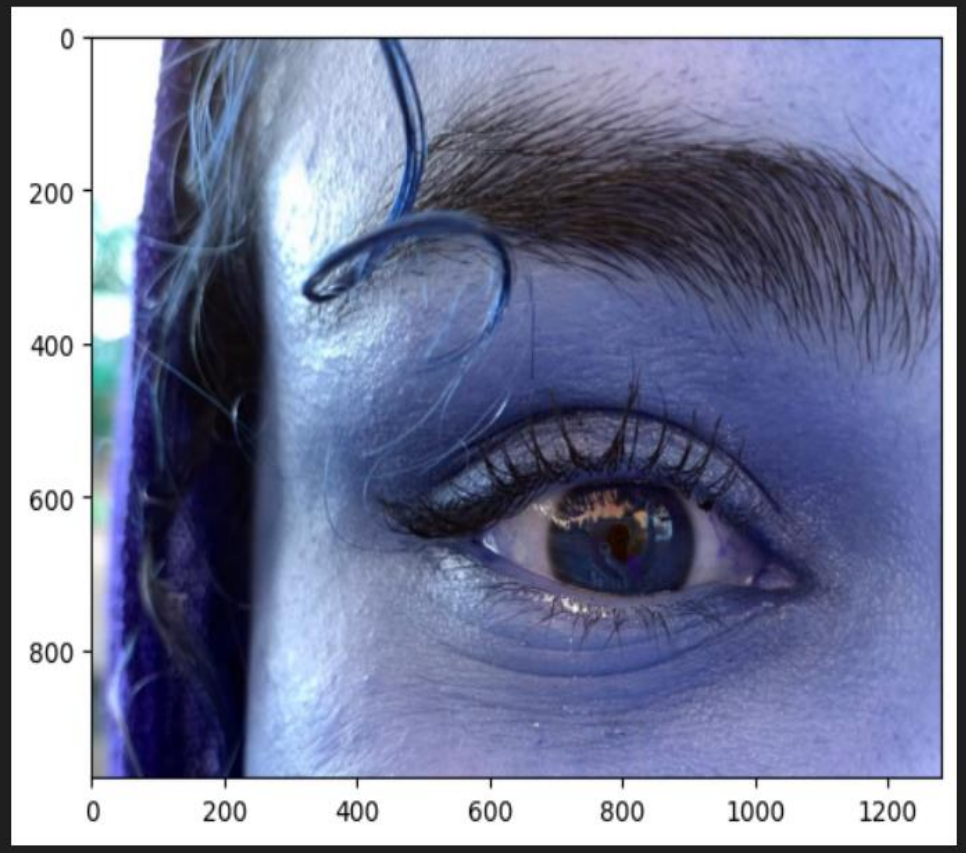
And then we read our input images:

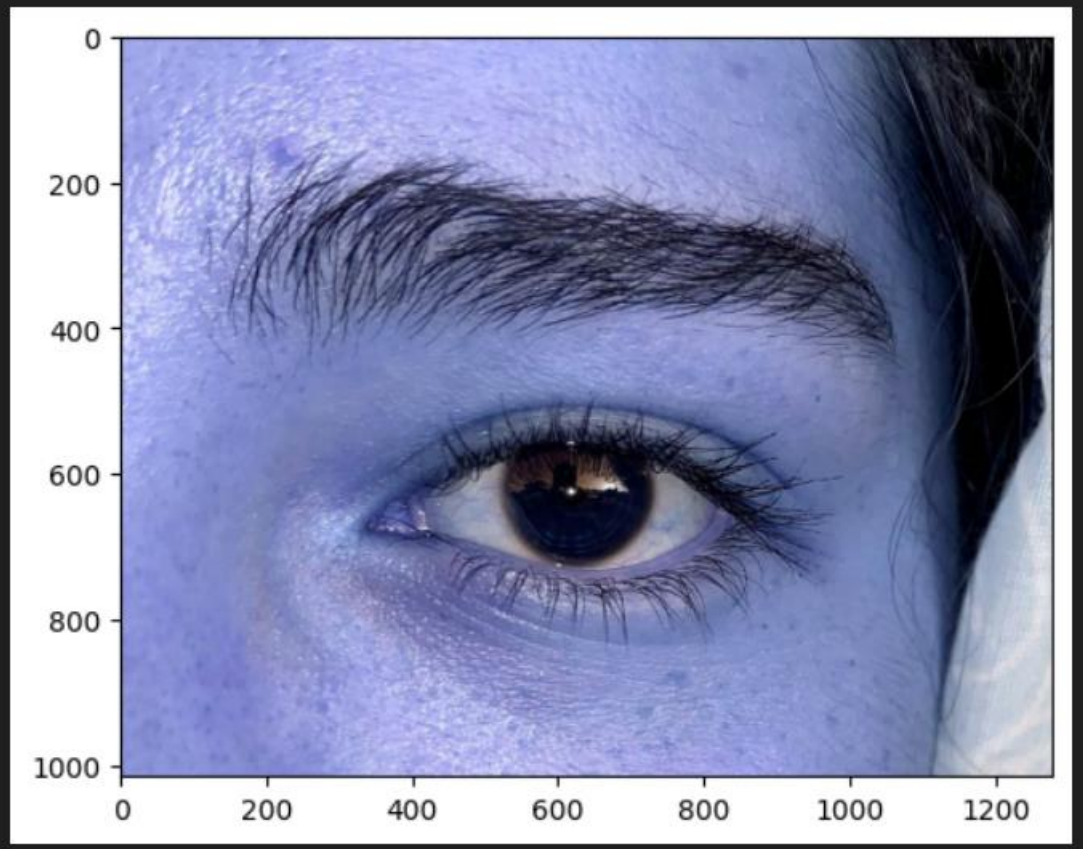
```
# Read the original image
img1 = cv.imread('/content/drive/My Drive/IMG1.jpg',cv.IMREAD_COLOR)
img2 = cv.imread('/content/drive/My Drive/IMG2.jpg',cv.IMREAD_COLOR)
img3 = cv.imread('/content/drive/My Drive/IMG3.jpg',cv.IMREAD_COLOR)
img4 = cv.imread('/content/drive/My Drive/IMG4.jpg',cv.IMREAD_COLOR)
```

We use matplot to show our images:

```
# Display original image
plt.imshow(img1)
plt.show()
plt.imshow(img2)
plt.show()
```

These are our results:





And then we convert our picture into greyscale using cv2 built-in function :

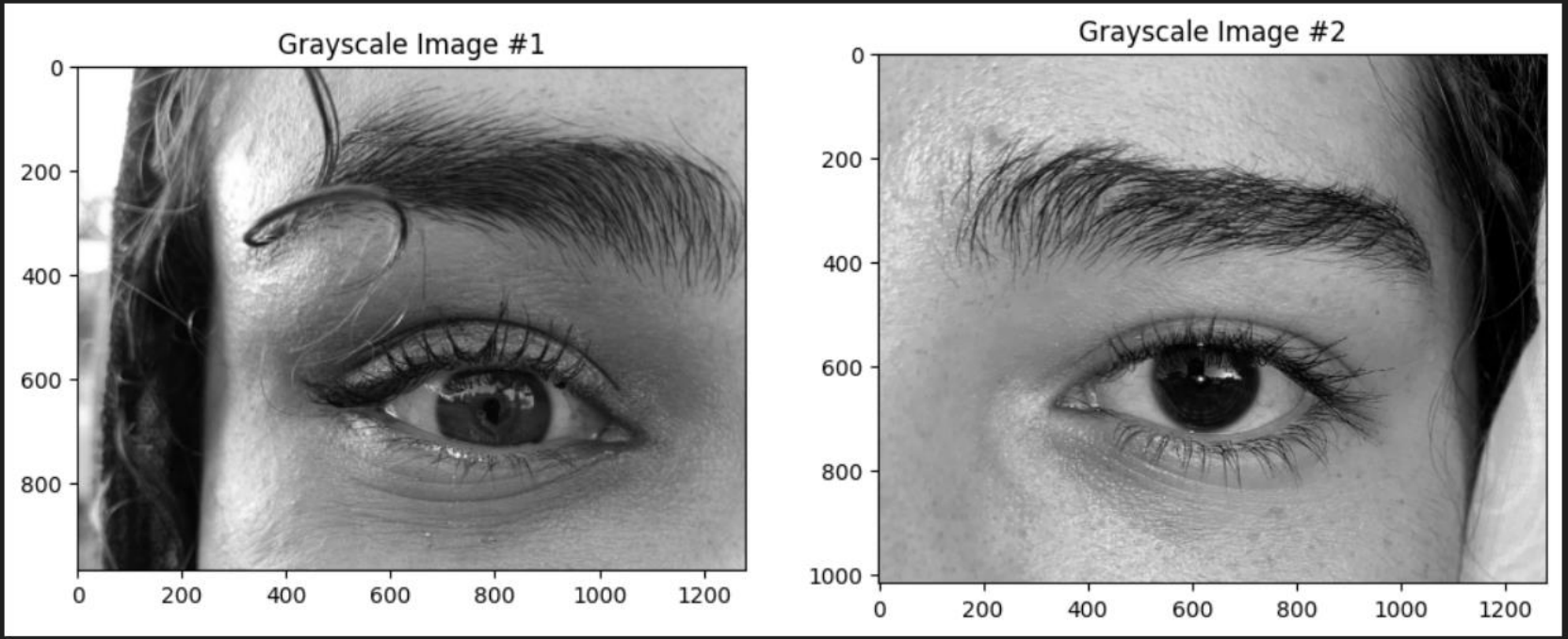
```
# Convert to Grayscale
img1_gray = cv.cvtColor(img1, cv.COLOR_BGR2GRAY)
img2_gray = cv.cvtColor(img2, cv.COLOR_BGR2GRAY)
```

We use matplot to show the greyscale version of our input images:

```
# Display Grayscale image
plt.figure(figsize=(12,7))
plt.subplot(121)
plt.imshow(img1_gray,cmap="gray")
plt.title('Grayscale Image #1')
plt.subplot(122)
plt.imshow(img2_gray,cmap="gray")
plt.title('Grayscale Image #2')
plt.show()
```

This is the result:

...



Since edge detection algorithms are sensitive to noises, we first run a blurring 3 by 3 kernel algorithm on our images.

```
# Blur the image for better edge detection
img1_blur = cv.GaussianBlur(img1_gray, (3,3), 0)
img2_blur = cv.GaussianBlur(img2_gray, (3,3), 0)
```

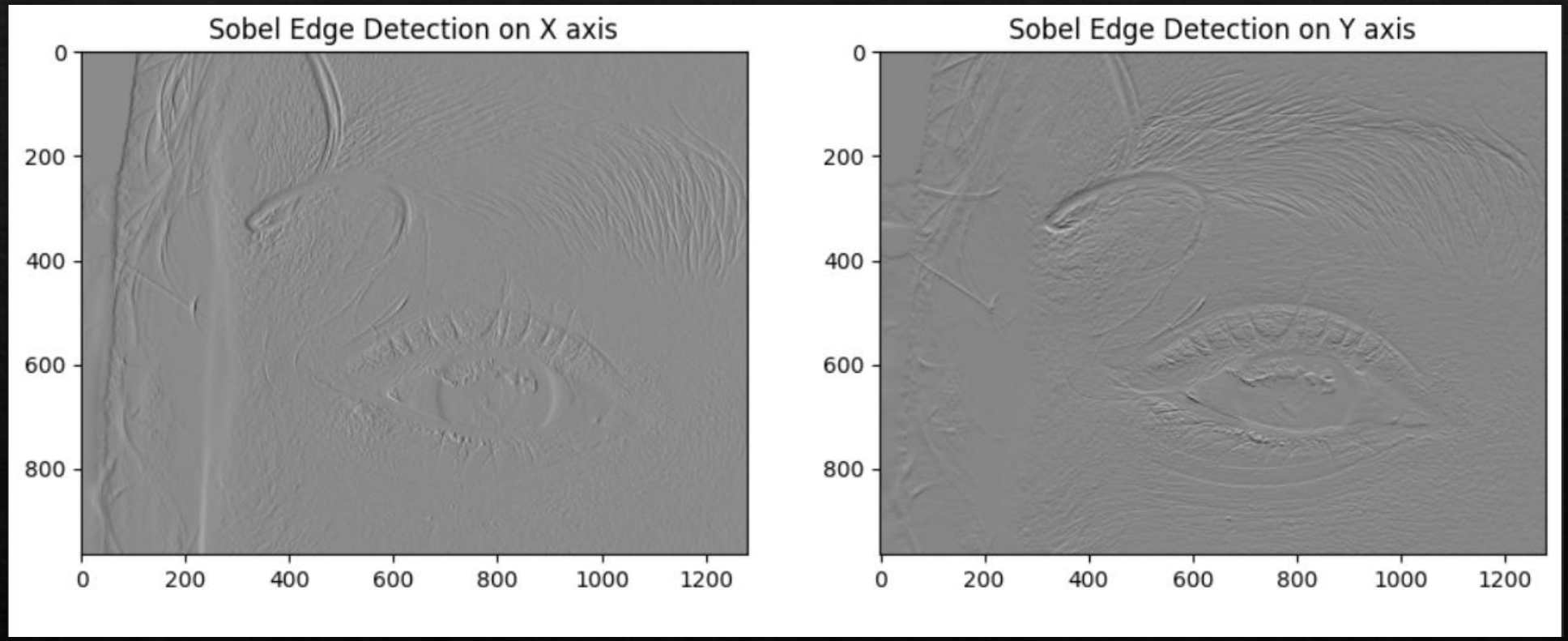
Sobel

After that, we use the sobel edge detection built-in function in cv.

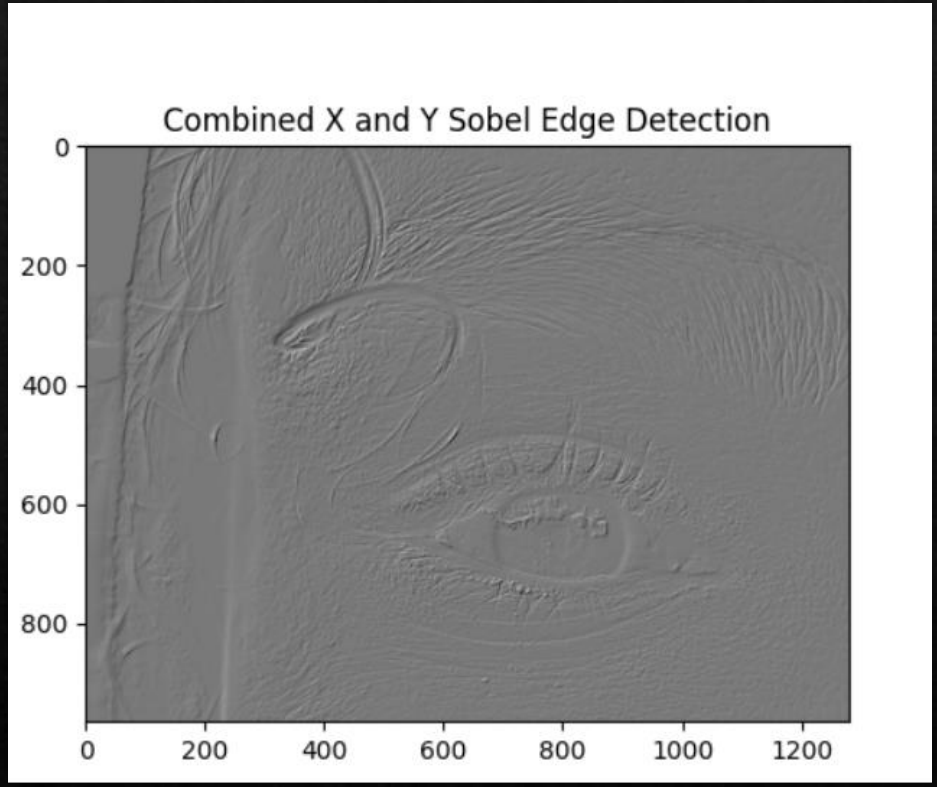
```
# Sobel Edge Detection
sobelx = cv.Sobel(src=img1_blur, ddepth=cv.CV_64F, dx=1, dy=0, ksize=3) # Sobel Edge Detection on the X axis
sobely = cv.Sobel(src=img1_blur, ddepth=cv.CV_64F, dx=0, dy=1, ksize=3) # Sobel Edge Detection on the Y axis
sobelxy = cv.add(sobelx,sobely) # Combined X and Y Sobel Edge Detection
```

```
# Display Sobel Edge Detection Images
plt.figure(figsize=(12,12))
plt.subplot(221)
plt.imshow(sobelx, cmap='gray')
plt.title('Sobel Edge Detection on X axis')
plt.subplot(222)
plt.imshow(sobely, cmap='gray')
plt.title('Sobel Edge Detection on Y axis')
plt.subplot(223)
plt.imshow(sobelxy, cmap='gray')
plt.title('Combined X and Y Sobel Edge Detection')
plt.show()


# Sobel Edge Detection
sobelx = cv.Sobel(src=img2_blur, ddepth=cv.CV_64F, dx=1, dy=0, ksize=3) # Sobel Edge Detection on the X axis
sobely = cv.Sobel(src=img2_blur, ddepth=cv.CV_64F, dx=0, dy=1, ksize=3) # Sobel Edge Detection on the Y axis
sobelxy = cv.add(sobelx,sobely) # Combined X and Y Sobel Edge Detection
# Display Sobel Edge Detection Images
plt.figure(figsize=(12,12))
plt.subplot(221)
plt.imshow(sobelx, cmap='gray')
plt.title('Sobel Edge Detection on X axis')
plt.subplot(222)
plt.imshow(sobely, cmap='gray')
plt.title('Sobel Edge Detection on Y axis')
plt.subplot(223)
plt.imshow(sobelxy, cmap='gray')
plt.title('Combined X and Y Sobel Edge Detection')
plt.show()
```

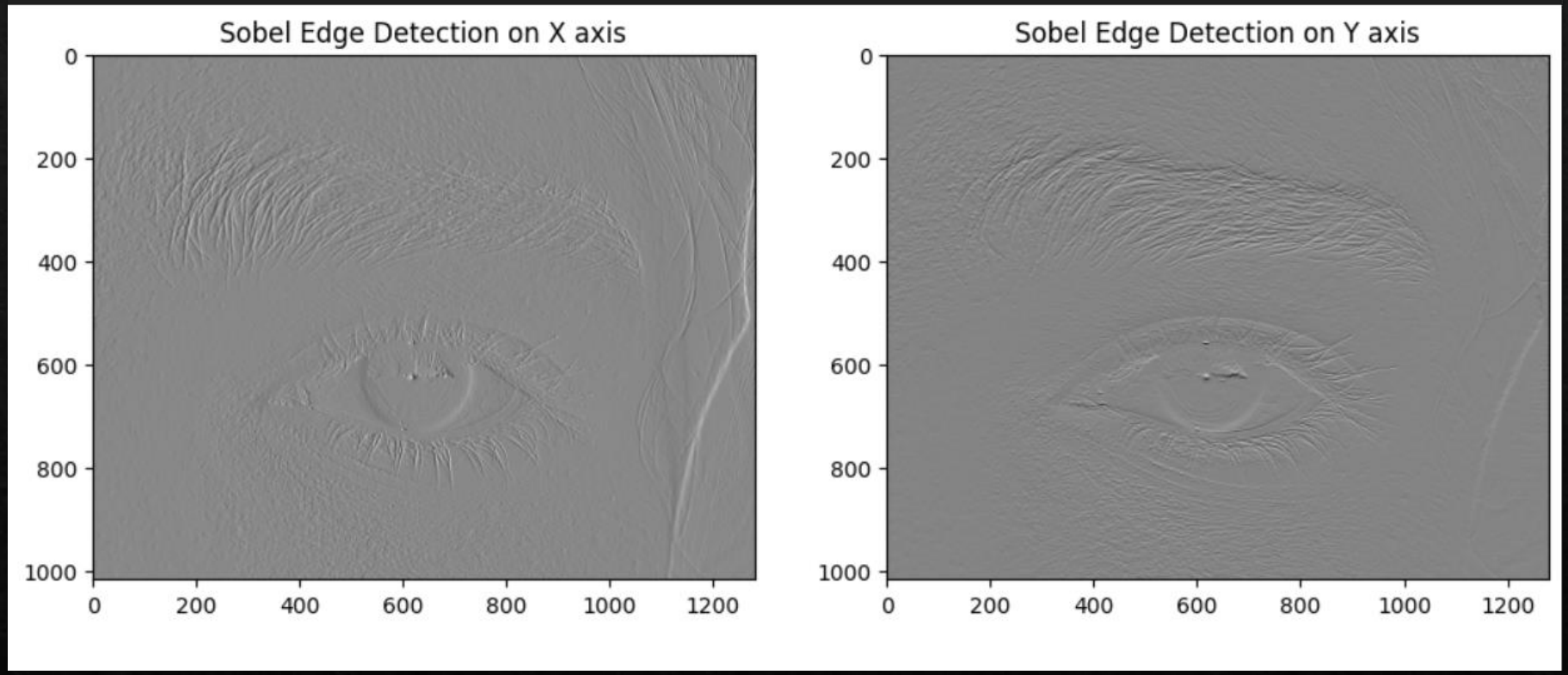
The result of image 1:



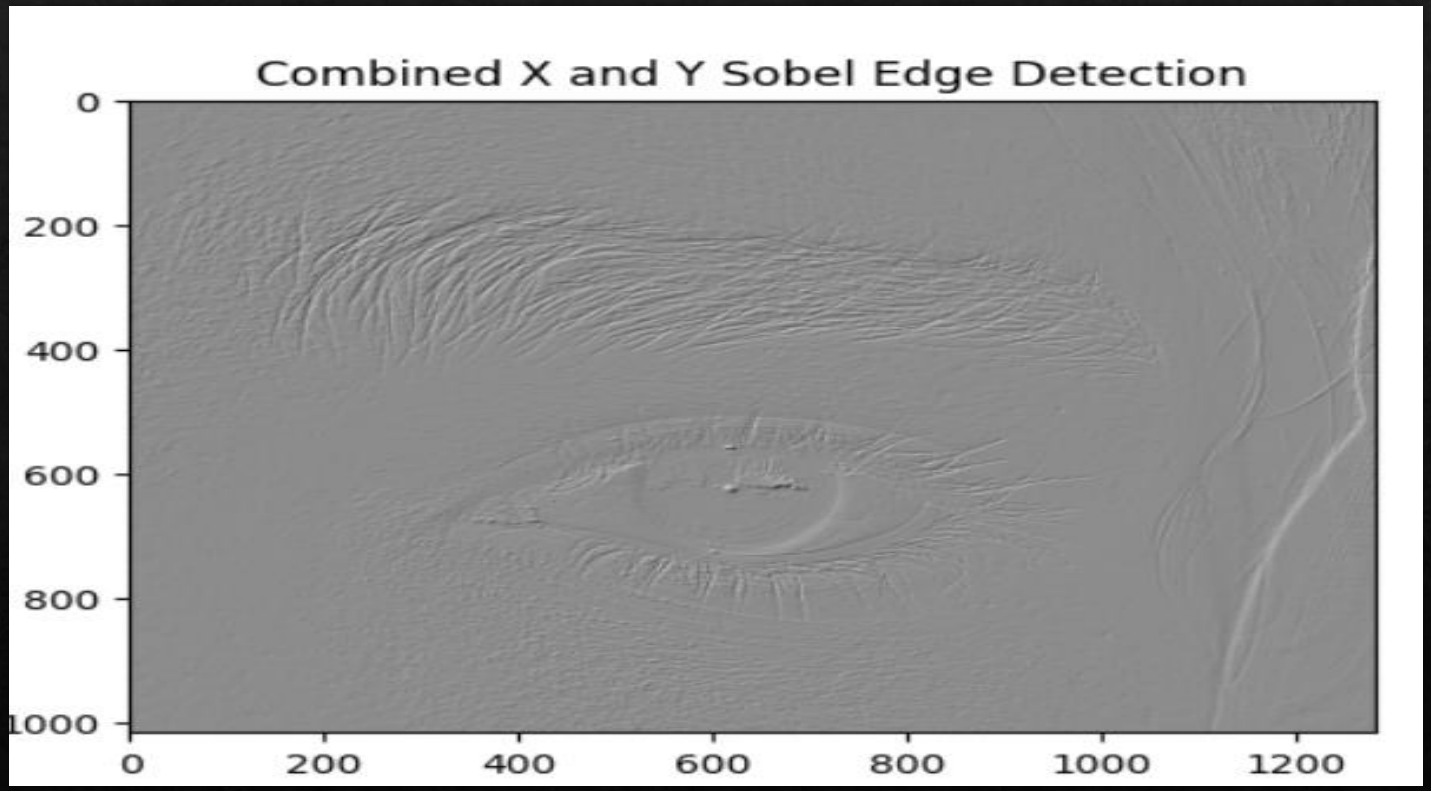
The result of image 1:



The result of image 2



The result of image 2



Enhanced Sobel

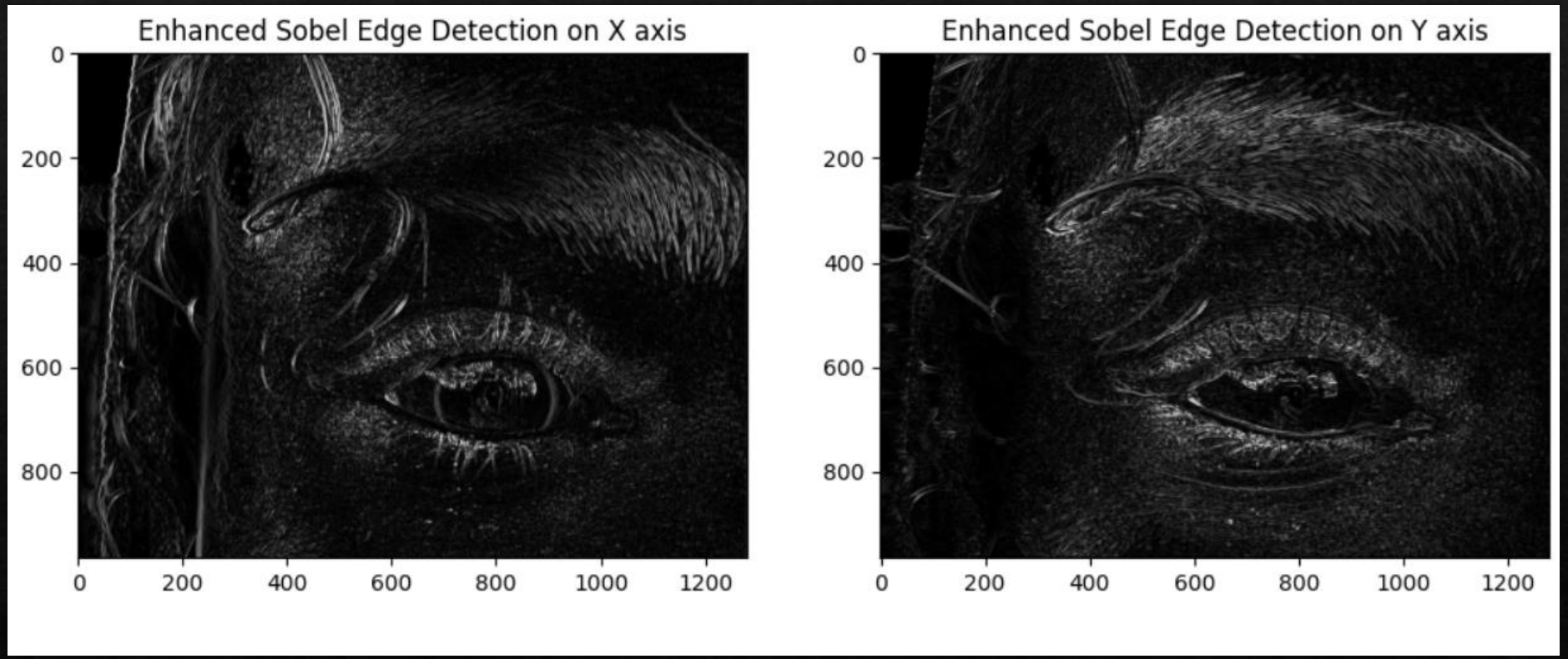
We also used the enhanced sobel edge detection on our images:

```
# Enhanced Sobel Edge Detection
sobelx = cv.convertScaleAbs(cv.Sobel(src=img1_blur, ddepth=cv.CV_16S, dx=1, dy=0, ksize=3)) # Sobel Edge Detection on the X axis
sobely = cv.convertScaleAbs(cv.Sobel(src=img1_blur, ddepth=cv.CV_16S, dx=0, dy=1, ksize=3)) # Sobel Edge Detection on the Y axis
sobelxy = cv.addWeighted(sobelx , 0.5 , sobely , 0.5 , 0) # Combined X and Y Sobel Edge Detection
```

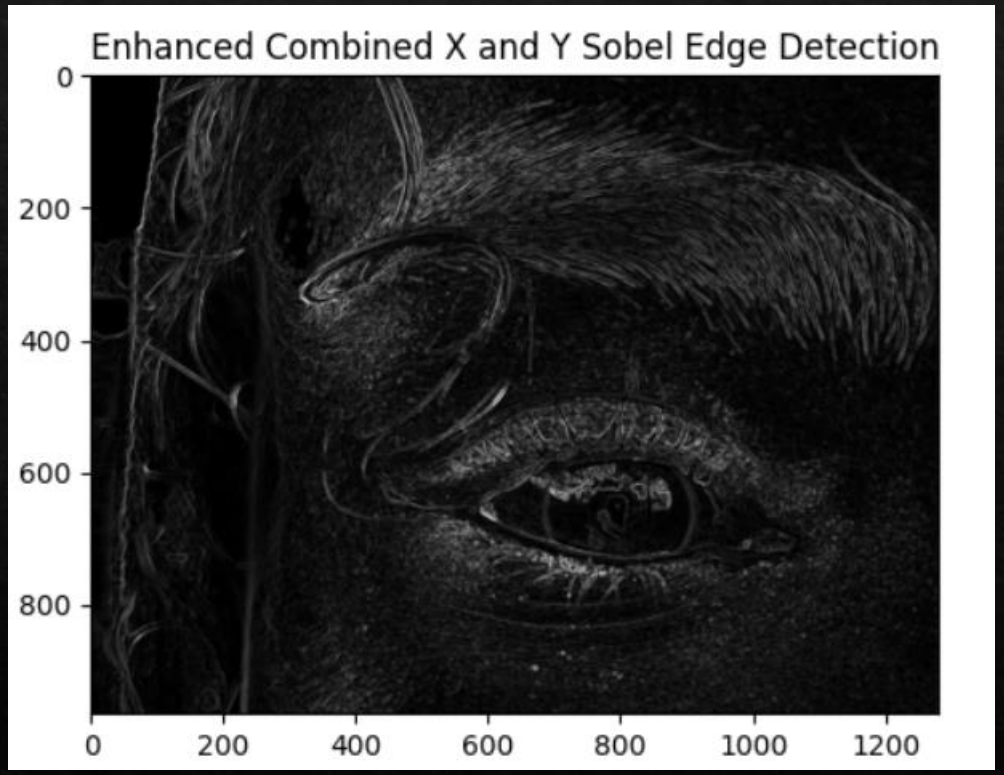
We used the following code to display our results:

```
# Display Sobel Edge Detection Images
plt.figure(figsize=(12,12))
plt.subplot(221)
plt.imshow(sobelx, cmap='gray')
plt.title('Enhanced Sobel Edge Detection on X axis')
plt.subplot(222)
plt.imshow(sobely, cmap='gray')
plt.title('Enhanced Sobel Edge Detection on Y axis')
plt.subplot(223)
plt.imshow(sobelxy, cmap='gray')
plt.title('Enhanced Combined X and Y Sobel Edge Detection')
plt.show()
```

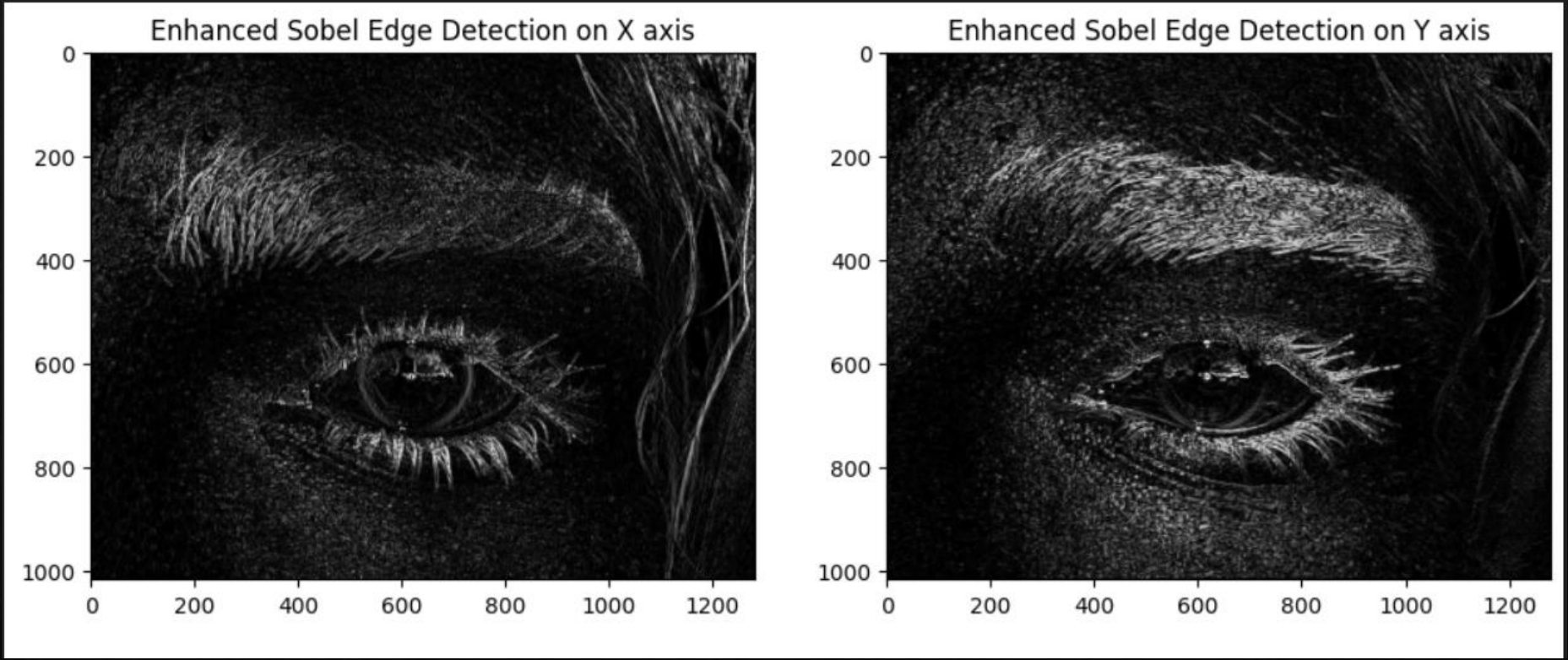
Our results on image 1 :



Our results on image 1 :



Our results on image 2:



Our results on image 2:

Canny

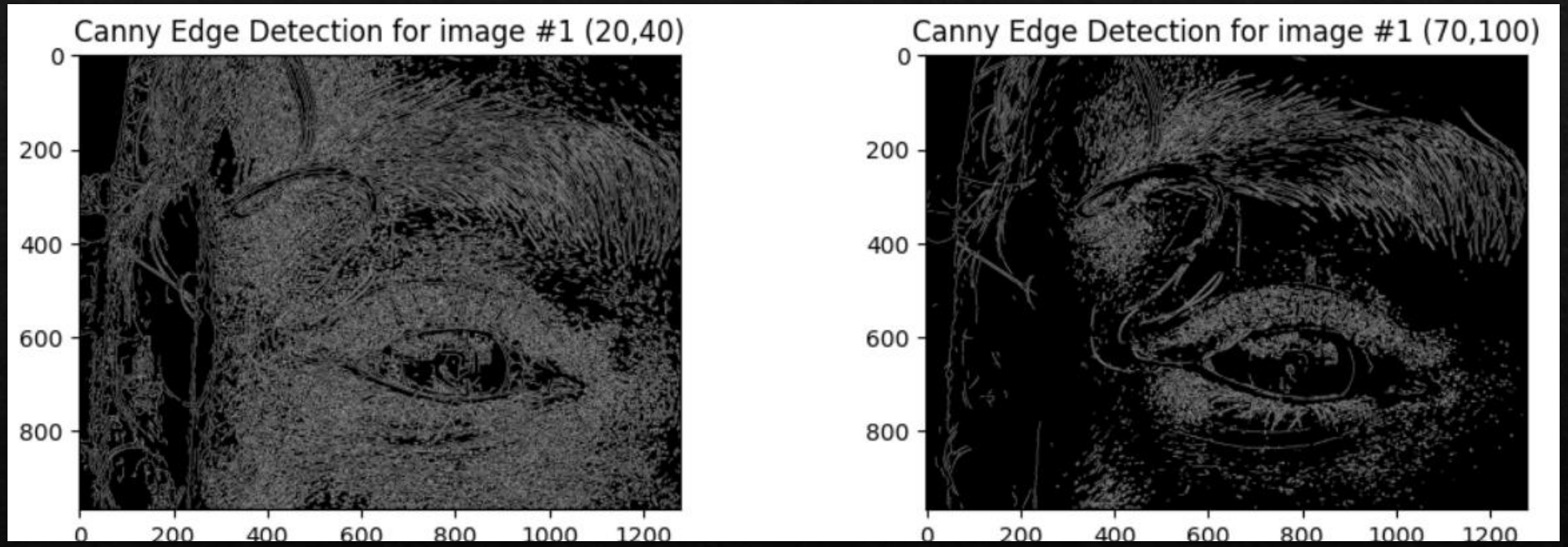
Our code for using the canny edge detection:

```
# Canny Edge Detection
img1_canny = cv.Canny(image=img1_blur, threshold1=20, threshold2=40)
img2_canny = cv.Canny(image=img2_blur, threshold1=20, threshold2=40)

img1_canny2 = cv.Canny(image=img1_blur, threshold1=70, threshold2=100)
img2_canny2 = cv.Canny(image=img2_blur, threshold1=70, threshold2=110)

img1_canny3 = cv.Canny(image=img1_blur, threshold1=50, threshold2=90)
img2_canny3 = cv.Canny(image=img2_blur, threshold1=60, threshold2=100)

img1_canny4 = cv.Canny(image=img1_blur, threshold1=150, threshold2=180)
img2_canny4 = cv.Canny(image=img2_blur, threshold1=150, threshold2=180)

img1_canny5 = cv.Canny(image=img1_blur, threshold1=90, threshold2=150)
img2_canny5 = cv.Canny(image=img2_blur, threshold1=90, threshold2=150)

img1_canny6 = cv.Canny(image=img1_blur, threshold1=80, threshold2=120)
img2_canny6 = cv.Canny(image=img2_blur, threshold1=80, threshold2=120)
```

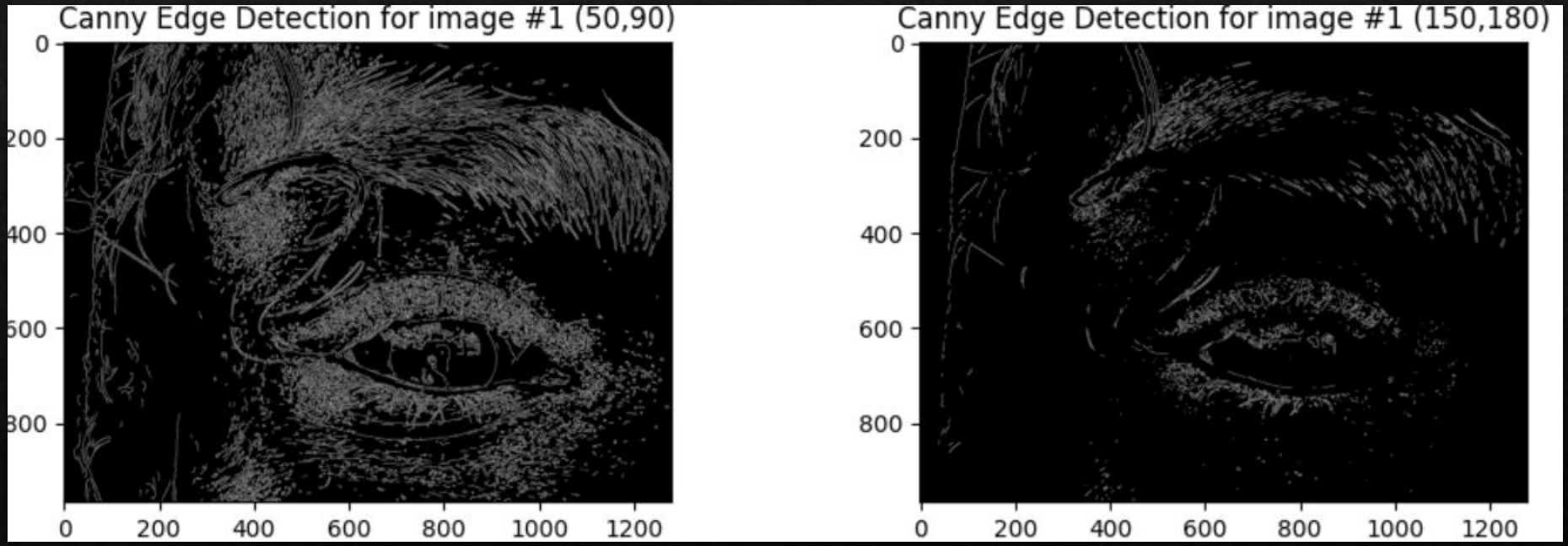
Matlplot code to show our result:

```
# Display Canny Edge Detection Image1
plt.figure(figsize=(12,12))
plt.subplot(321)
plt.imshow(img1_canny, cmap='gray')
plt.title('Canny Edge Detection for image #1 (20,40)')
plt.subplot(322)
plt.imshow(img1_canny2, cmap='gray')
plt.title('Canny Edge Detection for image #1 (70,100)')
plt.subplot(323)
plt.imshow(img1_canny3, cmap='gray')
plt.title('Canny Edge Detection for image #1 (50,90)')
plt.subplot(324)
plt.imshow(img1_canny4, cmap='gray')
plt.title('Canny Edge Detection for image #1 (150,180)')
plt.subplot(325)
plt.imshow(img1_canny5, cmap='gray')
plt.title('Canny Edge Detection for image #1 (90,150)')
plt.subplot(326)
plt.imshow(img1_canny6, cmap='gray')
plt.title('Canny Edge Detection for image #1 (80,120)')
plt.show()
```

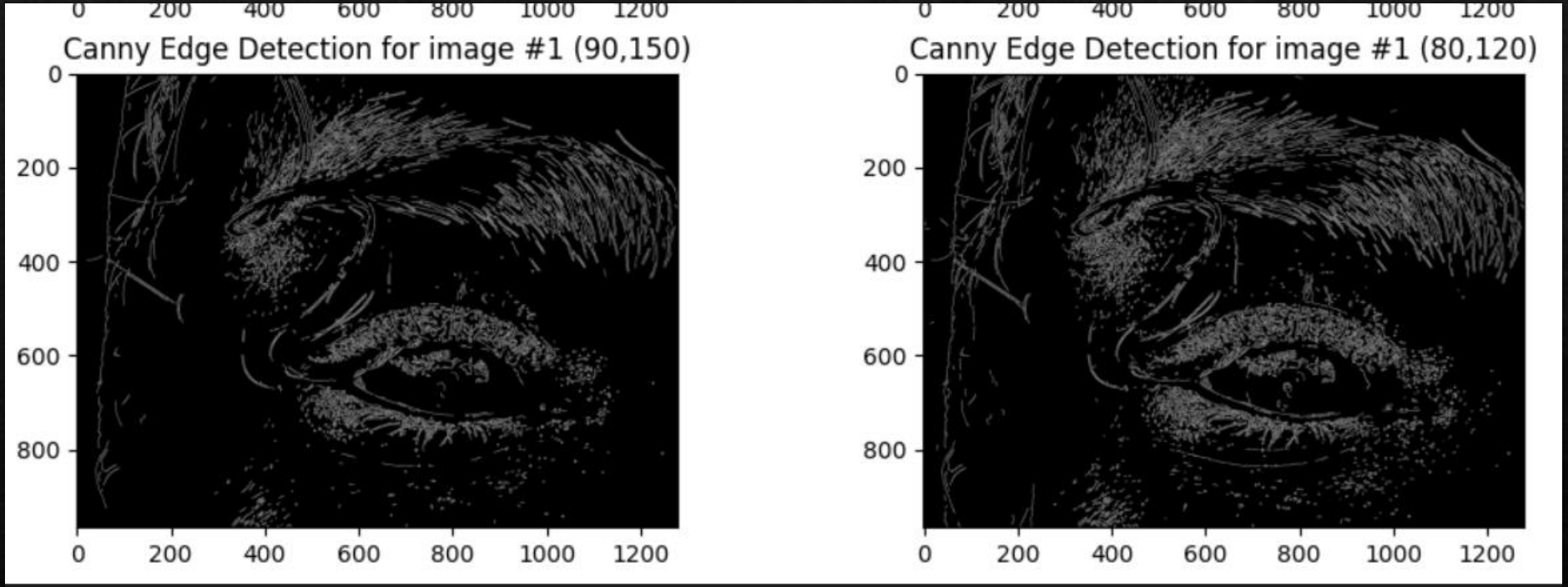
Our result:


Canny Edge Detection for image #1 (20,40)
0
200
400
600
800
0
200
400
600
800
1000
1200
Canny Edge Detection for image #1 (70,100)
0
200
400
600
800
0
200
400
600
800
1000
1200

Our result:

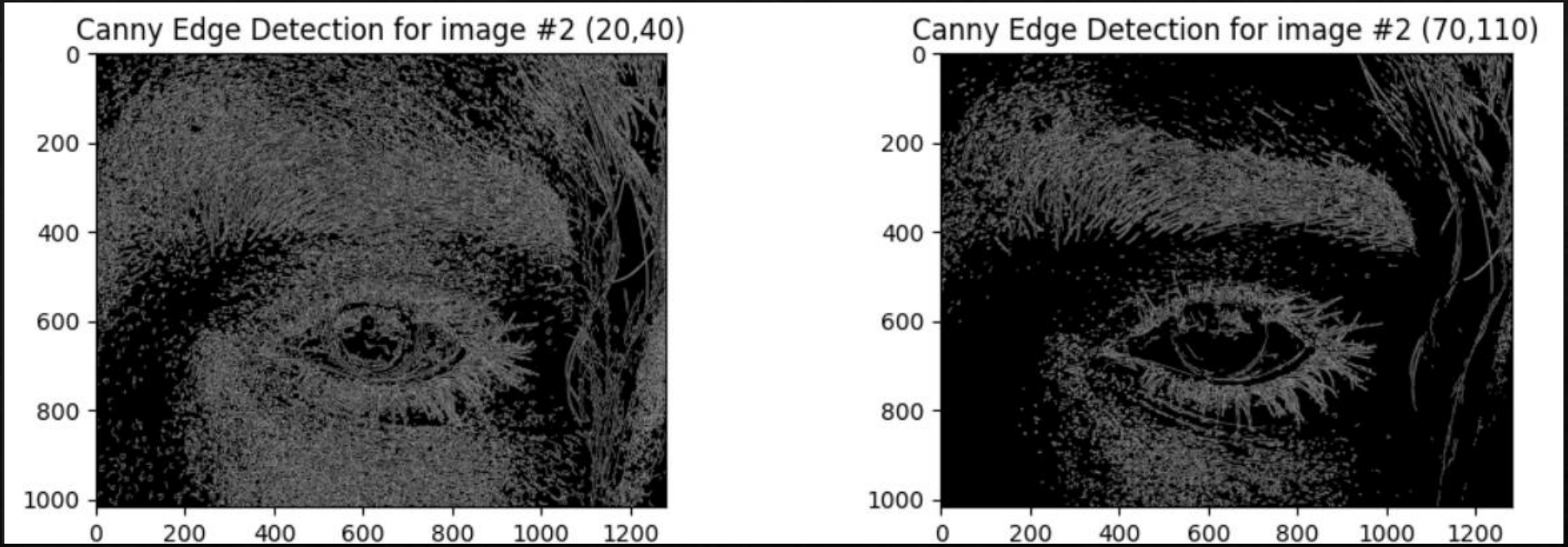

Canny Edge Detection for image #1 (50,90)
0
200
400
600
800
0
200
400
600
800
1000
1200
Canny Edge Detection for image #1 (150,180)
0
200
400
600
800
0
200
400
600
800
1000
1200

Our result:

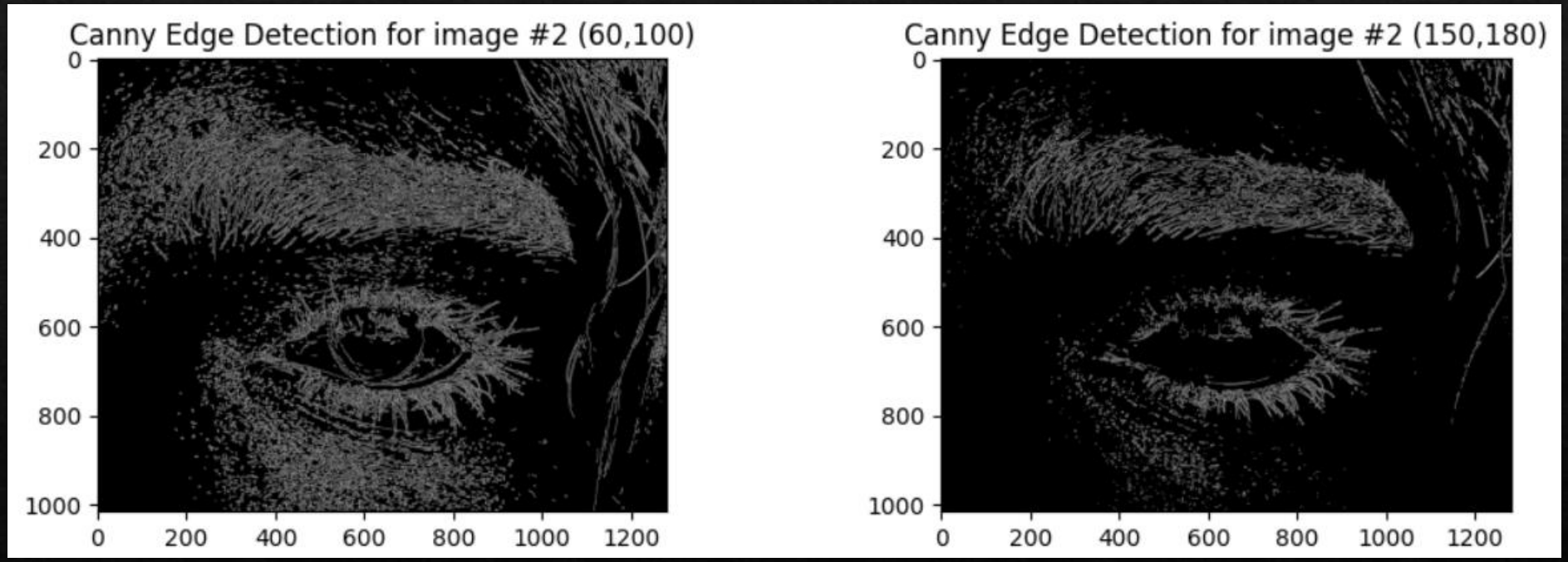

Canny Edge Detection for image #1 (90,150)
0
200
400
600
800
1000
1200
Canny Edge Detection for image #1 (80,120)

```
# Display Canny Edge Detection Image2
plt.figure(figsize=(12,12))
plt.subplot(321)
plt.imshow(img2_canny, cmap='gray')
plt.title('Canny Edge Detection for image #2 (20,40)')
plt.subplot(322)
plt.imshow(img2_canny2, cmap='gray')
plt.title('Canny Edge Detection for image #2 (70,110)')
plt.subplot(323)
plt.imshow(img2_canny3, cmap='gray')
plt.title('Canny Edge Detection for image #2 (60,100)')
plt.subplot(324)
plt.imshow(img2_canny4, cmap='gray')
plt.title('Canny Edge Detection for image #2 (150,180)')
plt.subplot(325)
plt.imshow(img2_canny5, cmap='gray')
plt.title('Canny Edge Detection for image #2 (90,150)')
plt.subplot(326)
plt.imshow(img2_canny6, cmap='gray')
plt.title('Canny Edge Detection for image #2 (80,120)')
plt.show()
```

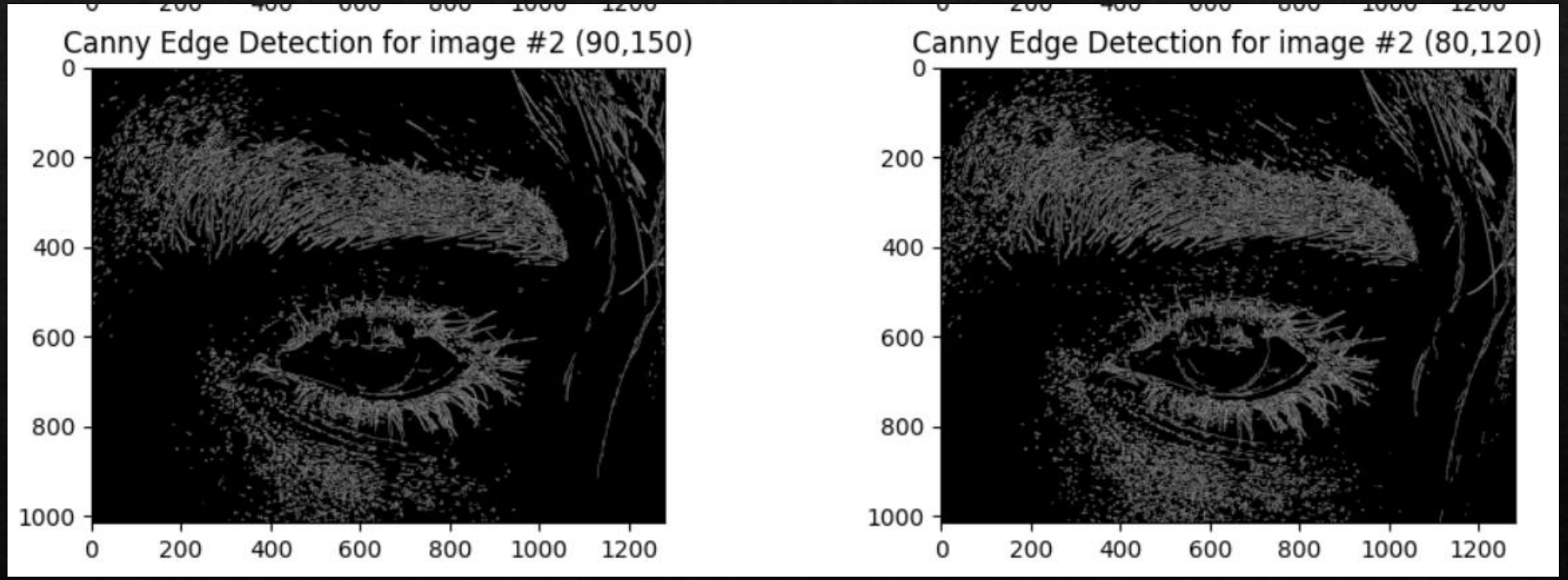
The result for image 2:



The result for image 2:



The result for image 2:



Marr-Hildreth algorithm:

```
# Marr-Hildreth Edge Detection
img1_laplacian = cv.convertScaleAbs(cv.Laplacian(img1_blur , cv.CV_16S , ksize=3))
img2_laplacian = cv.convertScaleAbs(cv.Laplacian(img2_blur , cv.CV_16S , ksize=3))



# Display Canny Edge Detection Image
plt.imshow(img1_laplacian, cmap='gray')
plt.title('Marr-Hildreth Edge Detection for image #1')
plt.show()
plt.imshow(img2_laplacian, cmap='gray')
plt.title('Marr-Hildreth Edge Detection for image #2')
plt.show()
```

Image 1:


Marr-Hildreth Edge Detection for image #1
0
200
400
600
800
0
200
400
600
800
1000
1200

Image 2:


Marr-Hildreth Edge Detection for image #2
0
200
400
600
800
1000
0
200
400
600
800
1000
1200

Hough

This is our hough algorithm:

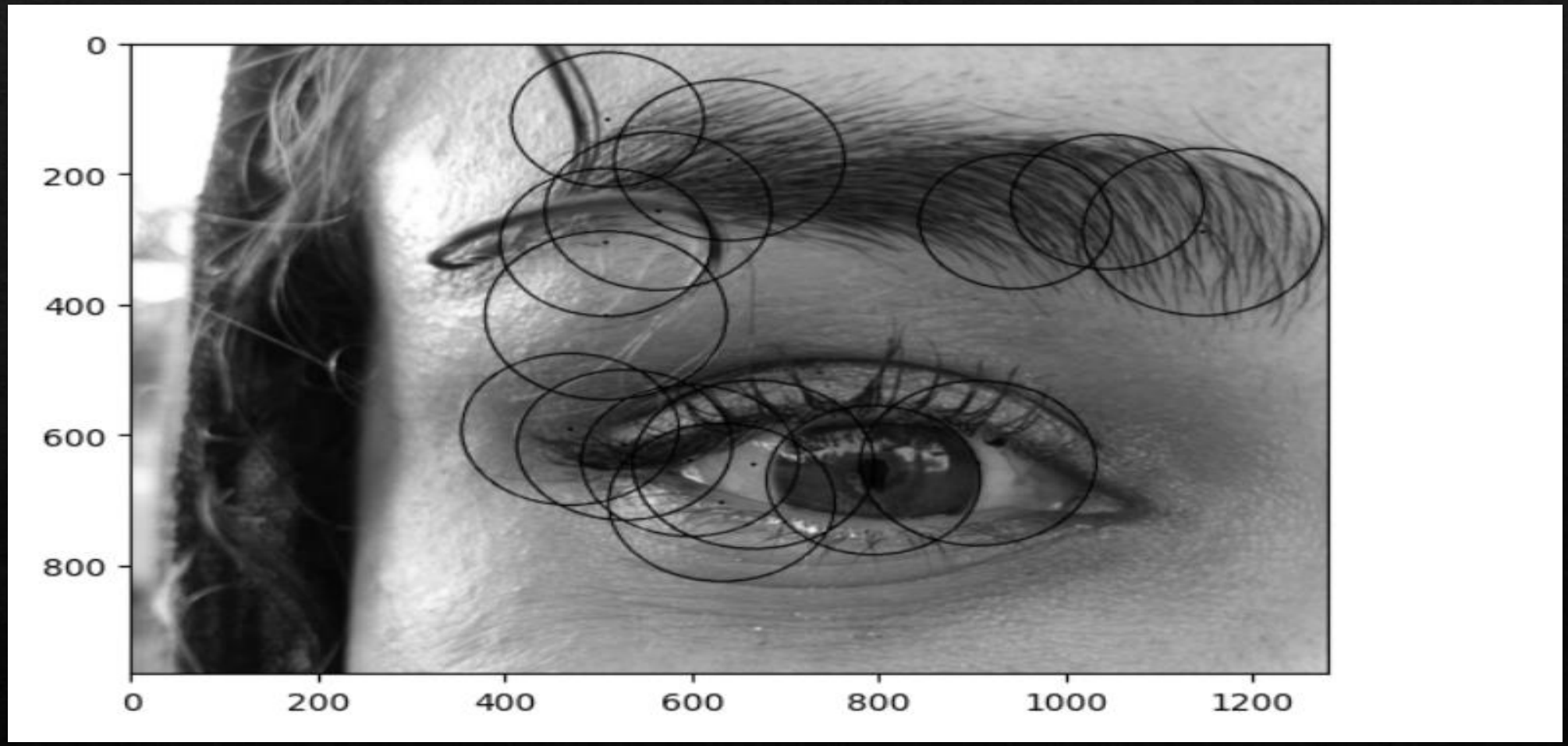
```
# Hough transform for first image

detected_circles = cv.HoughCircles(img1_blur,
                   cv.HOUGH_GRADIENT, 1, 60 ,
                  param1 = 80, param2 = 55,
                  minRadius = 30, maxRadius = 130)

# Draw circles that are detected.
if detected_circles is not None:

    # Convert the circle parameters a, b and r to integers.
    detected_circles = np.uint16(np.around(detected_circles))

    for pt in detected_circles[0, :]:
        a, b, r = pt[0], pt[1], pt[2]

        img1_gray, axes = plt.subplots()
        # Draw the circumference of the circle.
        cv.circle(img1_blur, (a, b), r, (0, 255, 0), 2)

        # Draw a small circle (of radius 1) to show the center.
        cv.circle(img1_blur, (a, b), 1, (0, 0, 255), 3)

        # Display the image.
        plt.imshow(img1_blur,cmap="gray")
        plt.show()
```

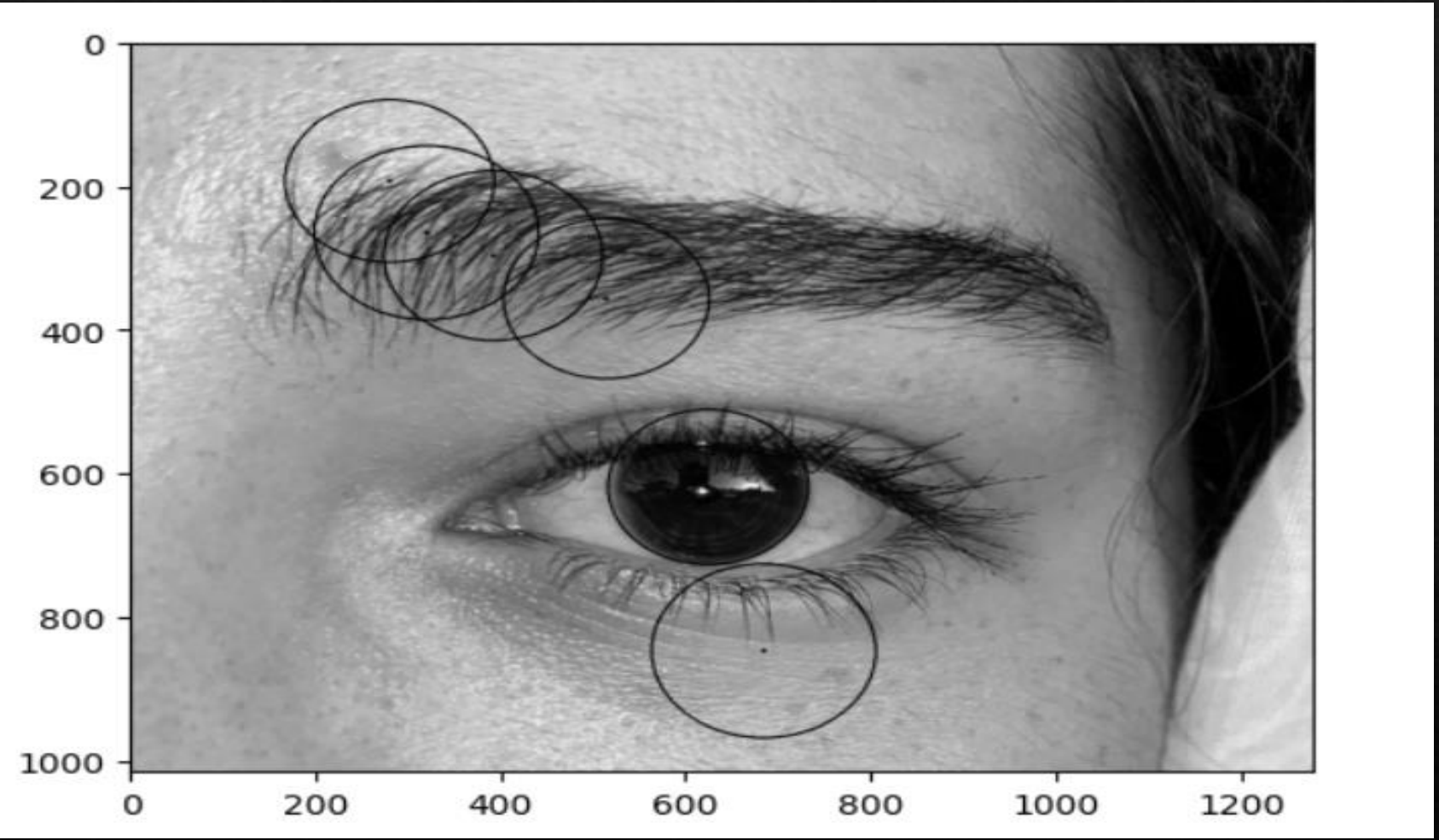
These are our arguments:

- *gray*: Input image (grayscale).
- *circles*: A vector that stores sets of 3 values: xc,yc,r for each detected circle.
- *HOUGH_GRADIENT*: Define the detection method. Currently this is the only one available in OpenCV.
- *dp = 1*: The inverse ratio of resolution.
- *min_dist = gray.rows/16*: Minimum distance between detected centers.
- *param_1 = 200*: Upper threshold for the internal Canny edge detector.
- *param_2* = 100*: Threshold for center detection.
- *min_radius = 0*: Minimum radius to be detected. If unknown, put zero as default.
- *max_radius = 0*: Maximum radius to be detected. If unknown, put zero as default

```python
# Hough transform for second image


detected_circles = cv.HoughCircles(img2_blur,
                   cv.HOUGH_GRADIENT, 1, 60 ,
                  param1 = 80, param2 = 70,
                  minRadius = 30, maxRadius = 130)

# Draw circles that are detected.
if detected_circles is not None:

    # Convert the circle parameters a, b and r to integers.
    detected_circles = np.uint16(np.around(detected_circles))

    for pt in detected_circles[0, :]:
        a, b, r = pt[0], pt[1], pt[2]

        img1_gray, axes = plt.subplots()
        # Draw the circumference of the circle.
        cv.circle(img2_blur, (a, b), r, (0, 255, 0), 2)

        # Draw a small circle (of radius 1) to show the center.
        cv.circle(img2_blur, (a, b), 1, (0, 0, 255), 3)

        # Display the image.
        plt.imshow(img2_blur,cmap="gray")
        plt.show()
```

This is the result:

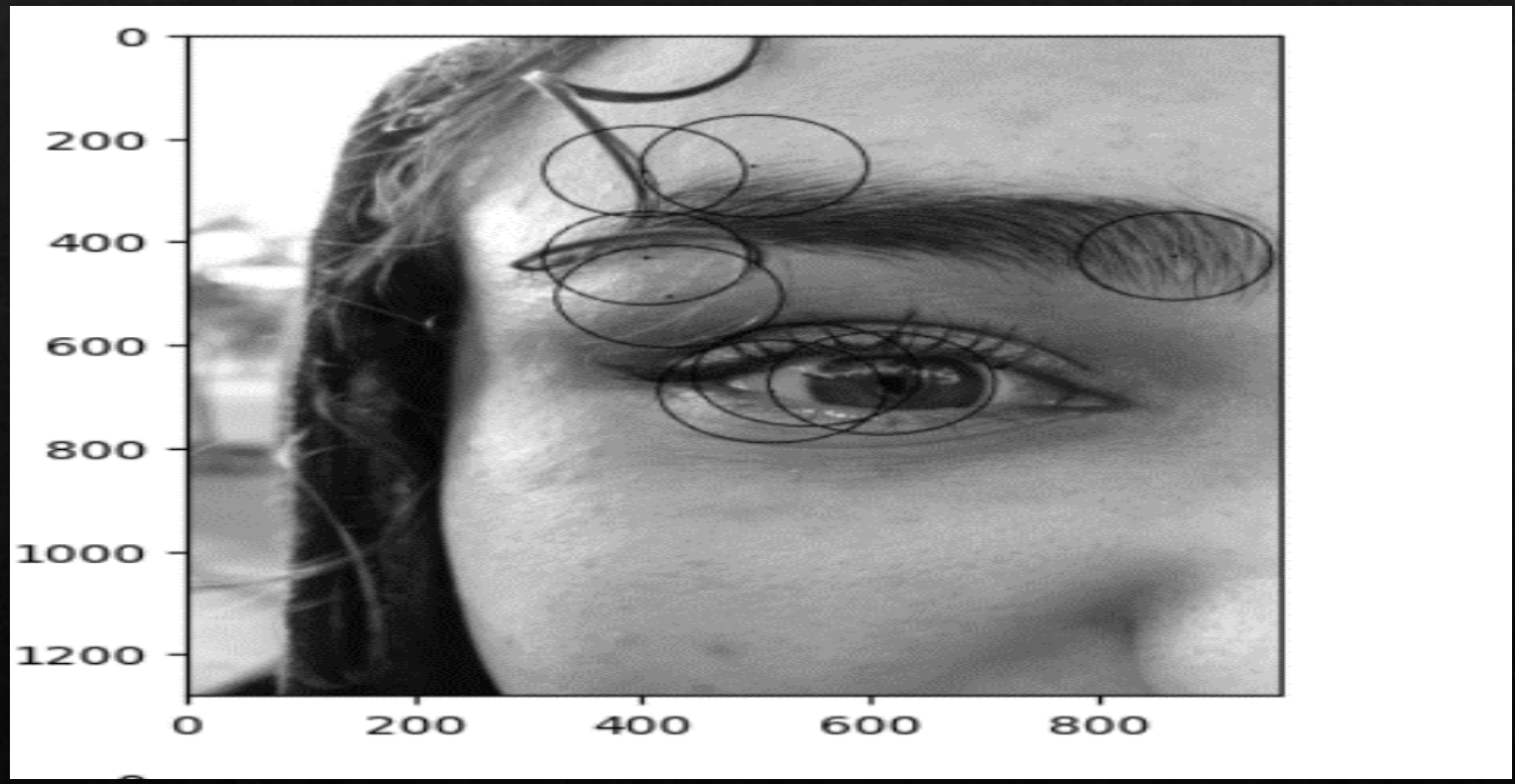


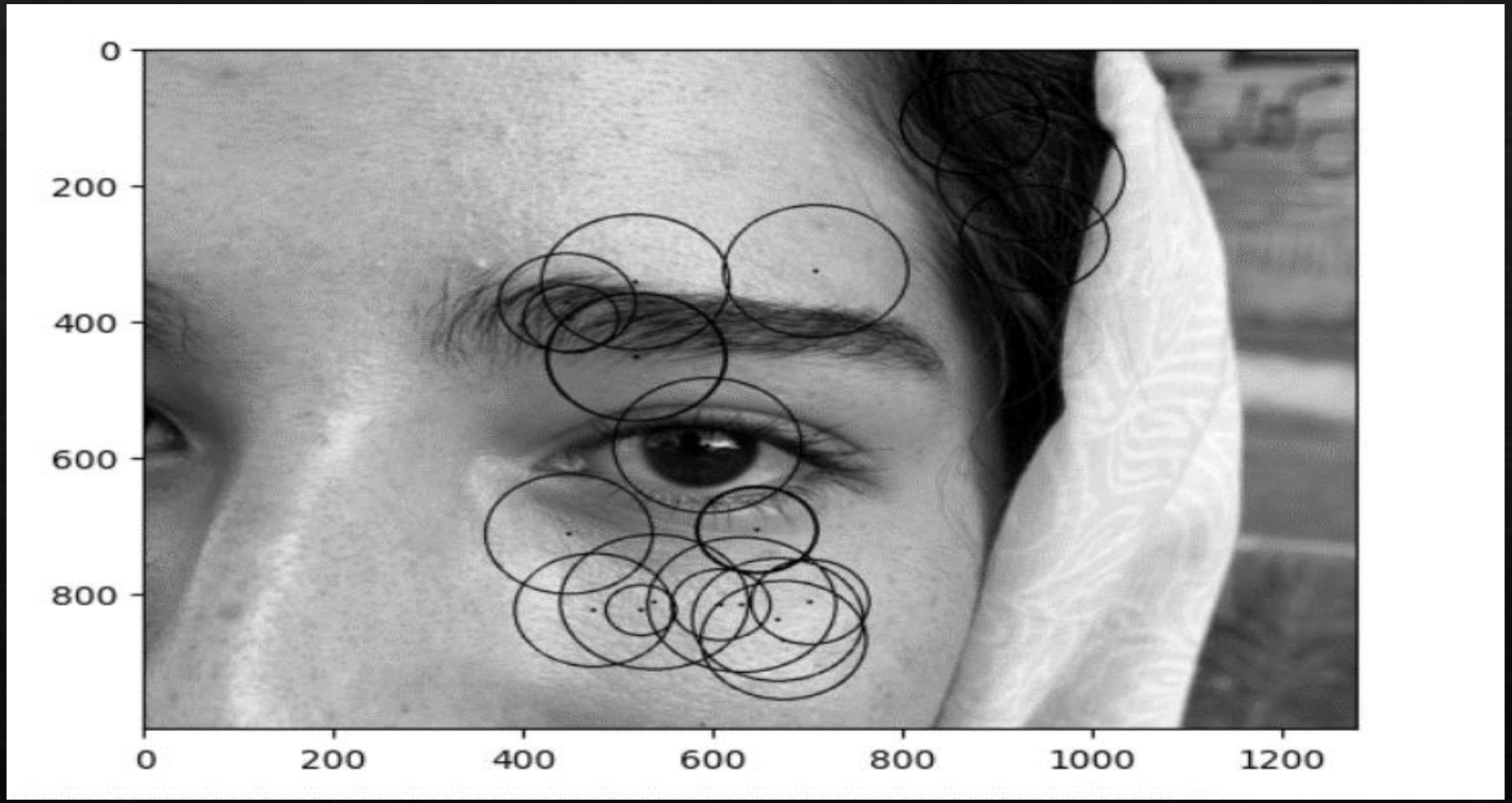

0
200
400
600
800
1000
0
200
400
600
800
1000
1200

```python
# Hough transform for first image (zoomed out)


detected_circles = cv.HoughCircles(img3_blur,
                   cv.HOUGH_GRADIENT, 1, 60 ,
                  param1 = 80, param2 = 55,
                  minRadius = 30, maxRadius = 130)

# Draw circles that are detected.
if detected_circles is not None:

    # Convert the circle parameters a, b and r to integers.
    detected_circles = np.uint16(np.around(detected_circles))

    for pt in detected_circles[0, :]:
        a, b, r = pt[0], pt[1], pt[2]

        img1_gray, axes = plt.subplots()
        # Draw the circumference of the circle.
        cv.circle(img3_blur, (a, b), r, (0, 255, 0), 2)

        # Draw a small circle (of radius 1) to show the center.
        cv.circle(img3_blur, (a, b), 1, (0, 0, 255), 3)

        # Display the image.
        plt.imshow(img3_blur,cmap="gray")
        plt.show()
```

```python
# Hough transform for second image (zoomed out)


detected_circles = cv.HoughCircles(img4_blur,
                   cv.HOUGH_GRADIENT, 1, 60 ,
                  param1 = 80, param2 = 70,
                  minRadius = 30, maxRadius = 130)

# Draw circles that are detected.
if detected_circles is not None:

    # Convert the circle parameters a, b and r to integers.
    detected_circles = np.uint16(np.around(detected_circles))

    for pt in detected_circles[0, :]:
        a, b, r = pt[0], pt[1], pt[2]

        img1_gray, axes = plt.subplots()
        # Draw the circumference of the circle.
        cv.circle(img4_blur, (a, b), r, (0, 255, 0), 2)

        # Draw a small circle (of radius 1) to show the center.
        cv.circle(img4_blur, (a, b), 1, (0, 0, 255), 3)

        # Display the image.
        plt.imshow(img4_blur,cmap="gray")
        plt.show()
```

This is our result:




0
200
400
600
800
0
200
400
600
800
1000
1200

The End

authors:
Armita Ashabyamin
under the surveillance of :
Professor Khosravi