Assignment 1

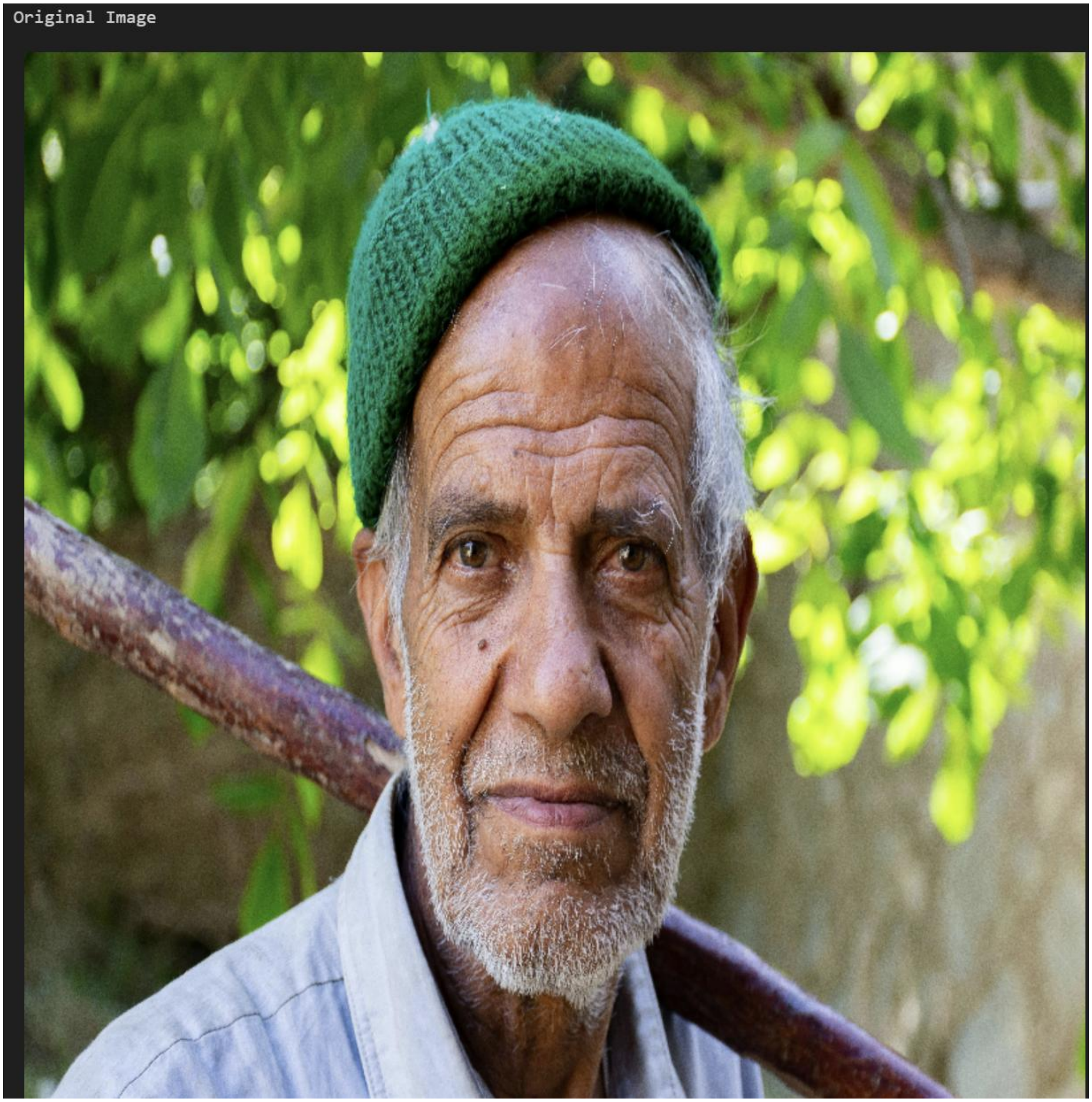
Presented by:

Abdelrahman Moataz ElBorgy 8065

Bassem Hesham 8000

Esraa Ahmed 8162

PART 1



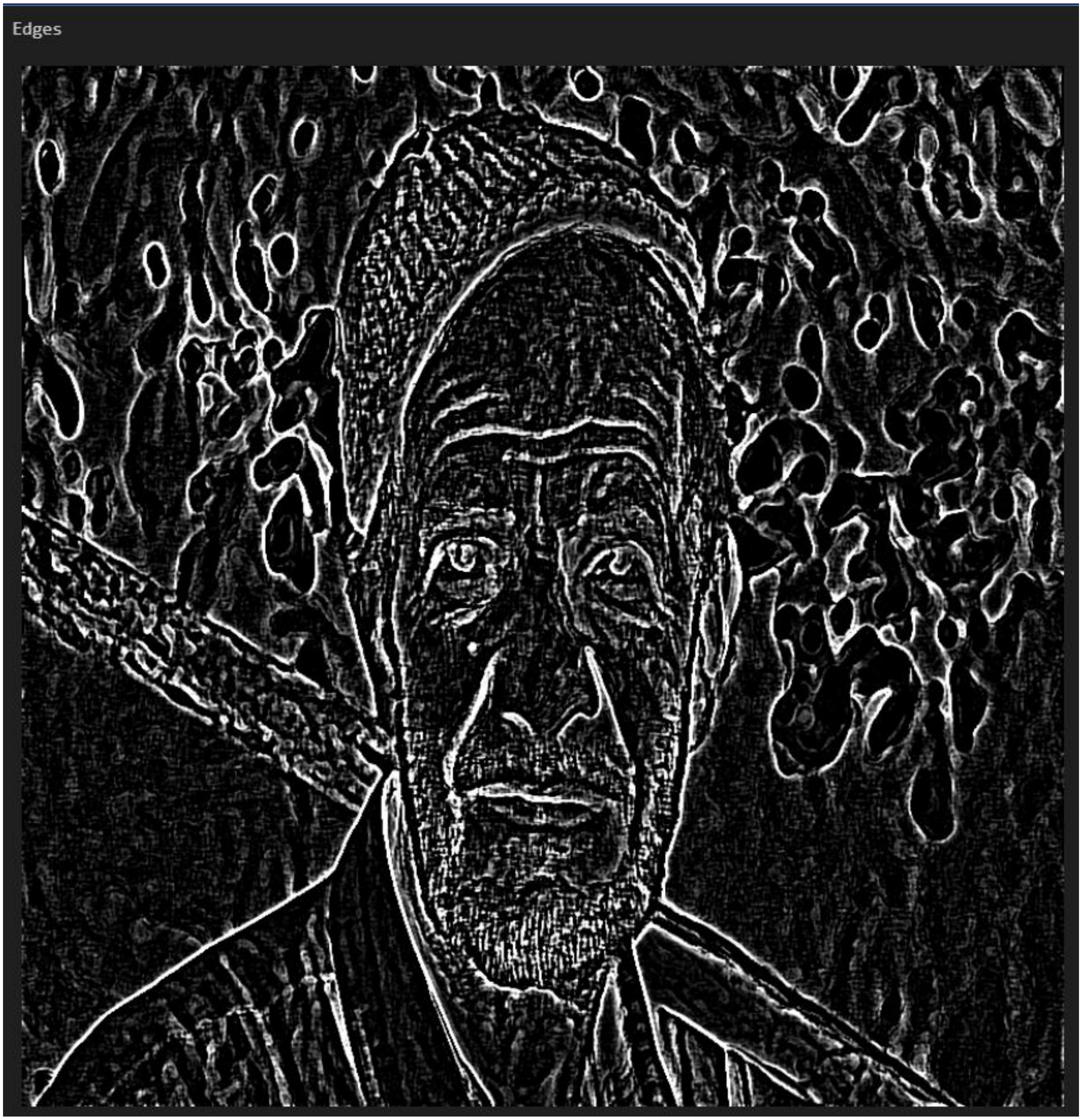
Original Image

second derivative filter which measures how fast the intensity changes in all directions

strong positive or negative means that there is a strong edge

cv2.CV_8U tells that the output is 8bit unsigned integer to represent values from 0-255

ksize=5 kernel size is 5 by 5

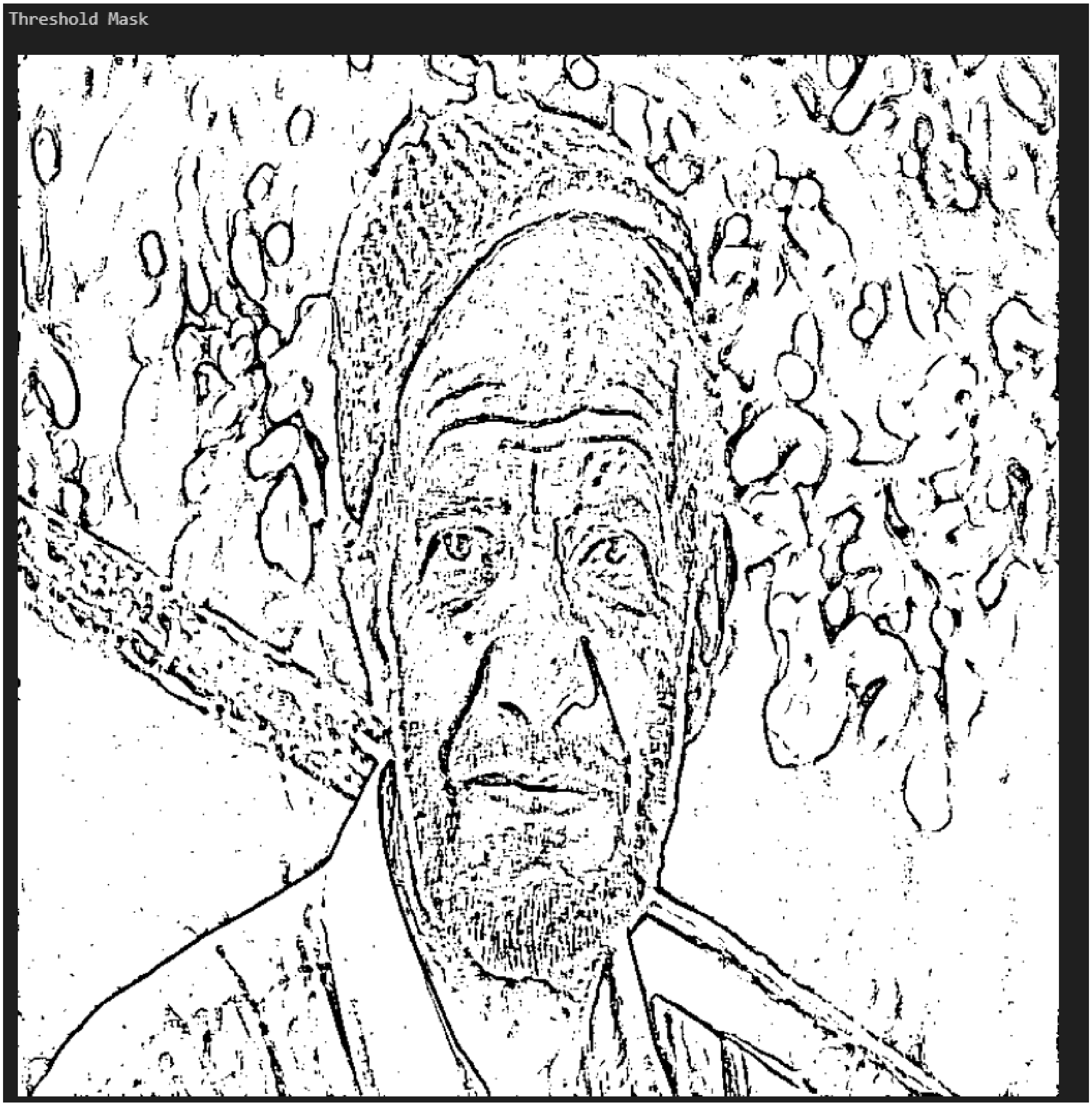


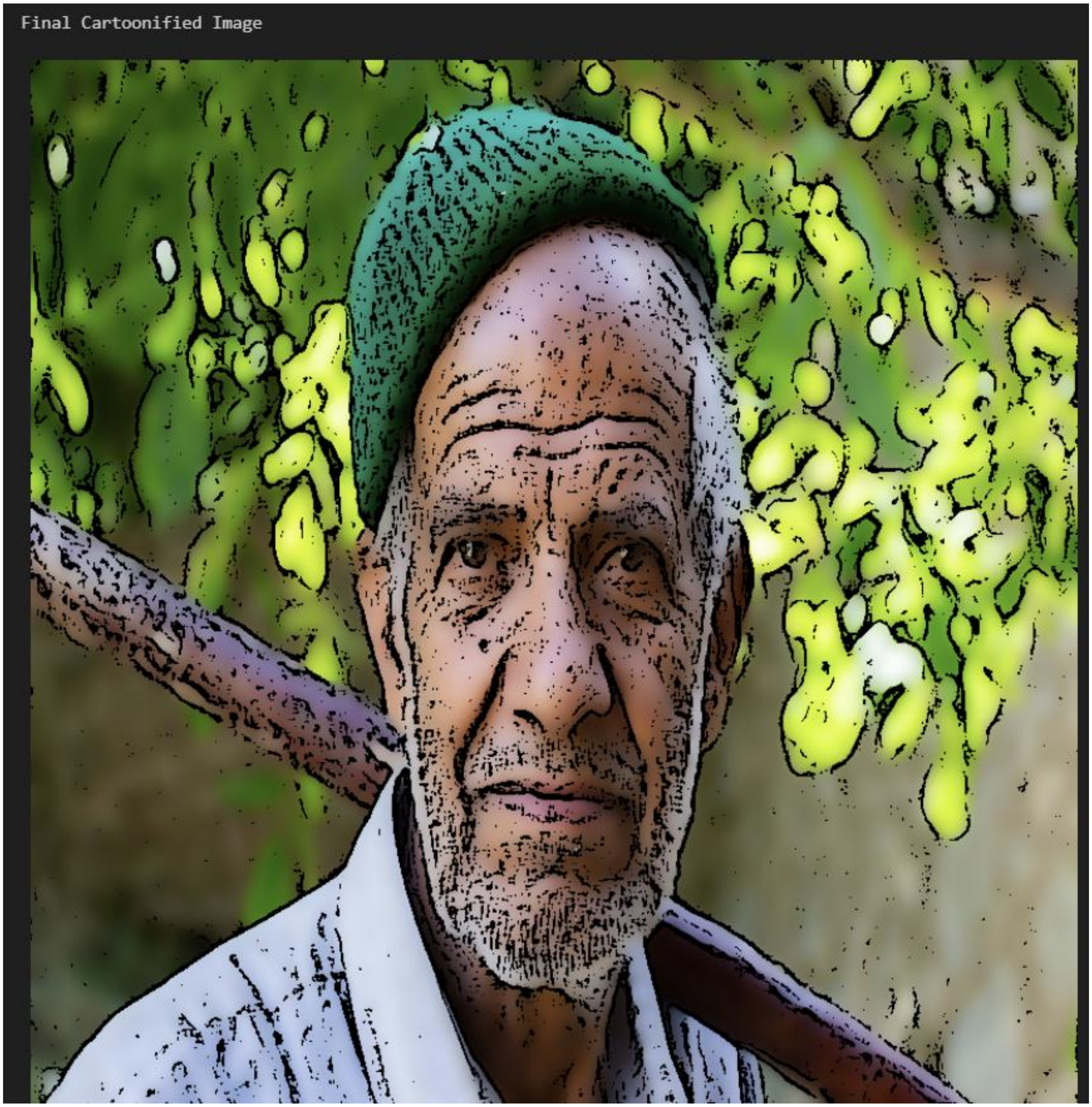
ret contain the value of the threshold which is equal to 100 here

mask contain the imaage after thresholding

if the value is greater than 100 it convert it to black and if smaller it convert it to white

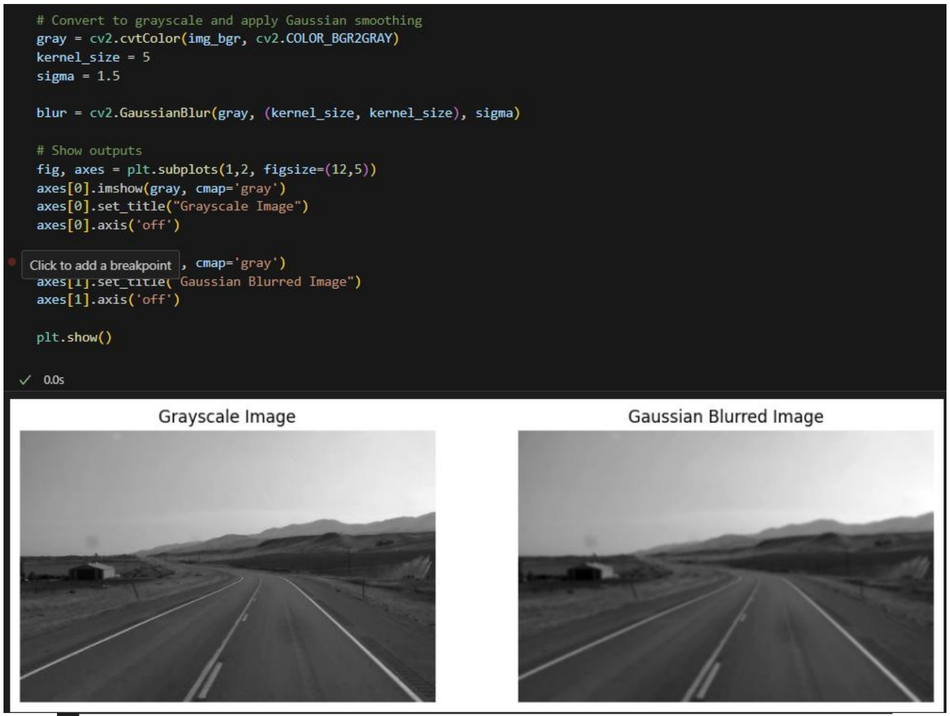
cv2.THRESH_BINARY_INV cause the laplacian return high values for places having strong edge





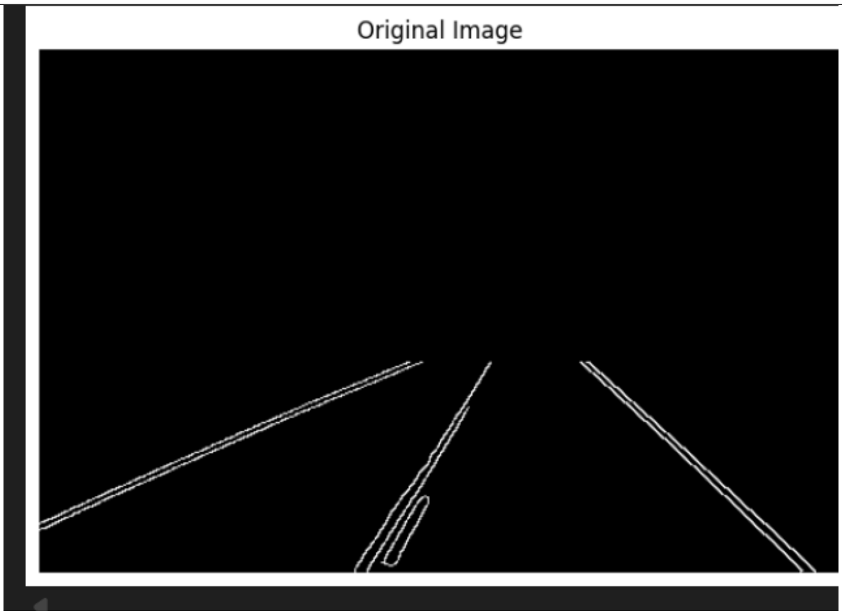
Final Cartoonified Image

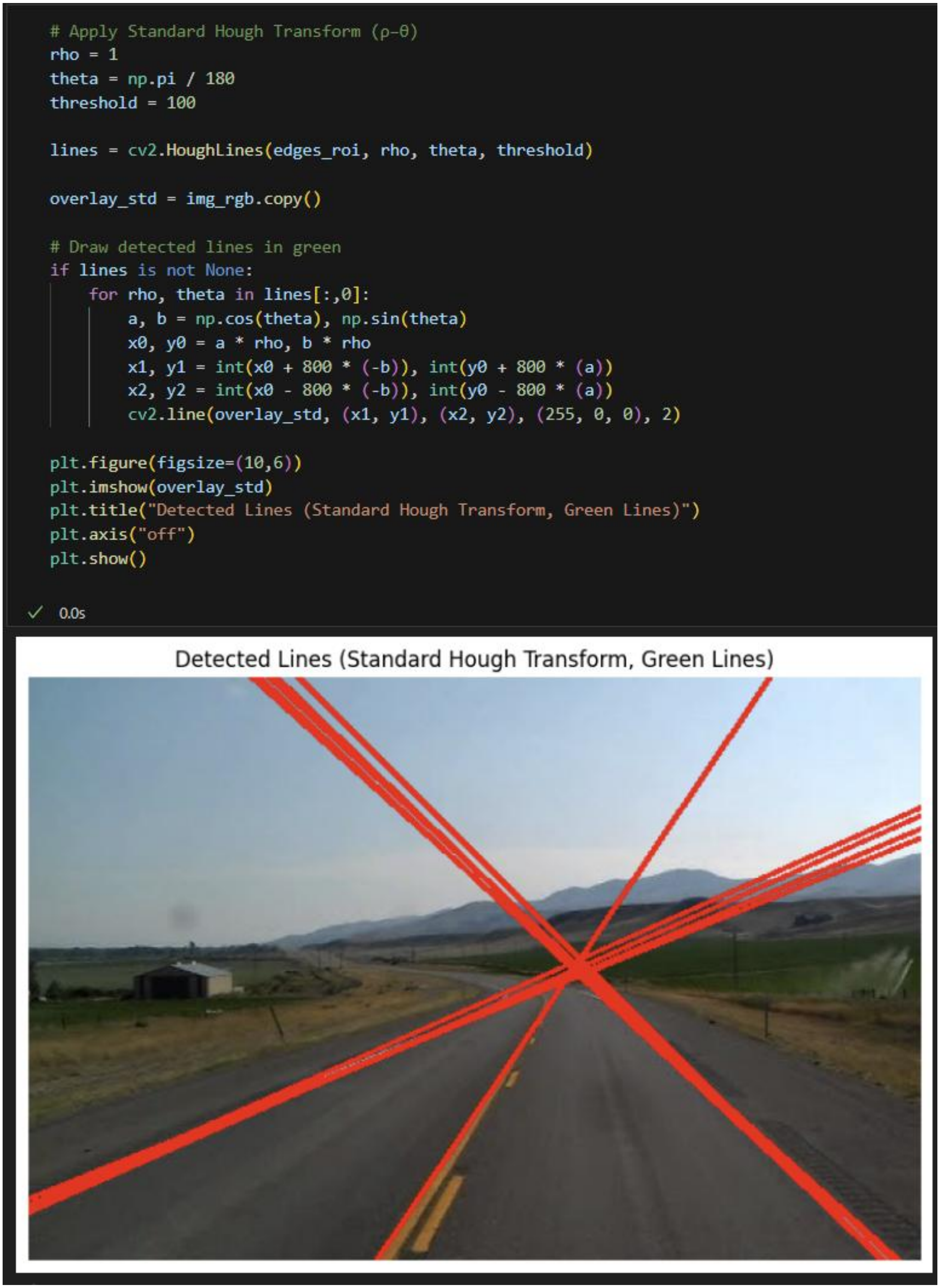
PART 2



```
# Convert to grayscale and apply Gaussian smoothing
gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
kernel_size = 5
sigma = 1.5

blur = cv2.GaussianBlur(gray, (kernel_size, kernel_size), sigma)

# Show outputs
fig, axes = plt.subplots(1,2, figsize=(12,5))
axes[0].imshow(gray, cmap='gray')
axes[0].set_title("Grayscale Image")
axes[0].axis('off')

Click to add a breakpoint , cmap='gray')
axes[1].set_title("Gaussian Blurred Image")
axes[1].axis('off')

plt.show()
```

✓ 0.0s

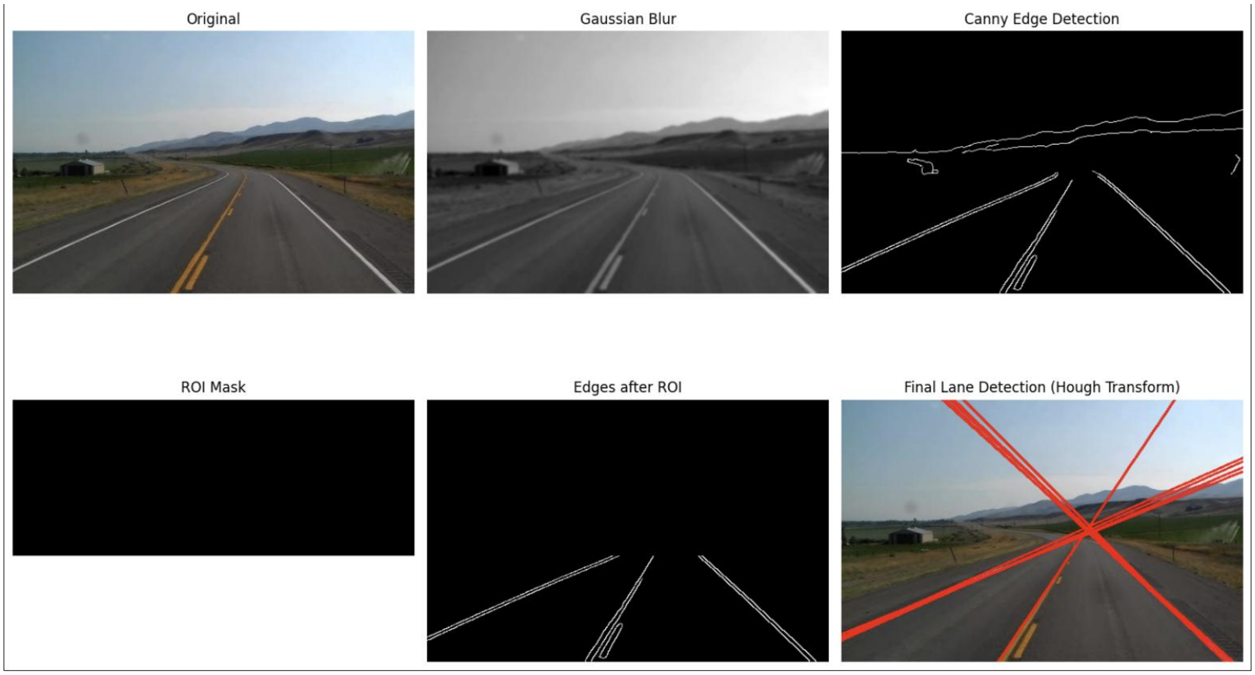


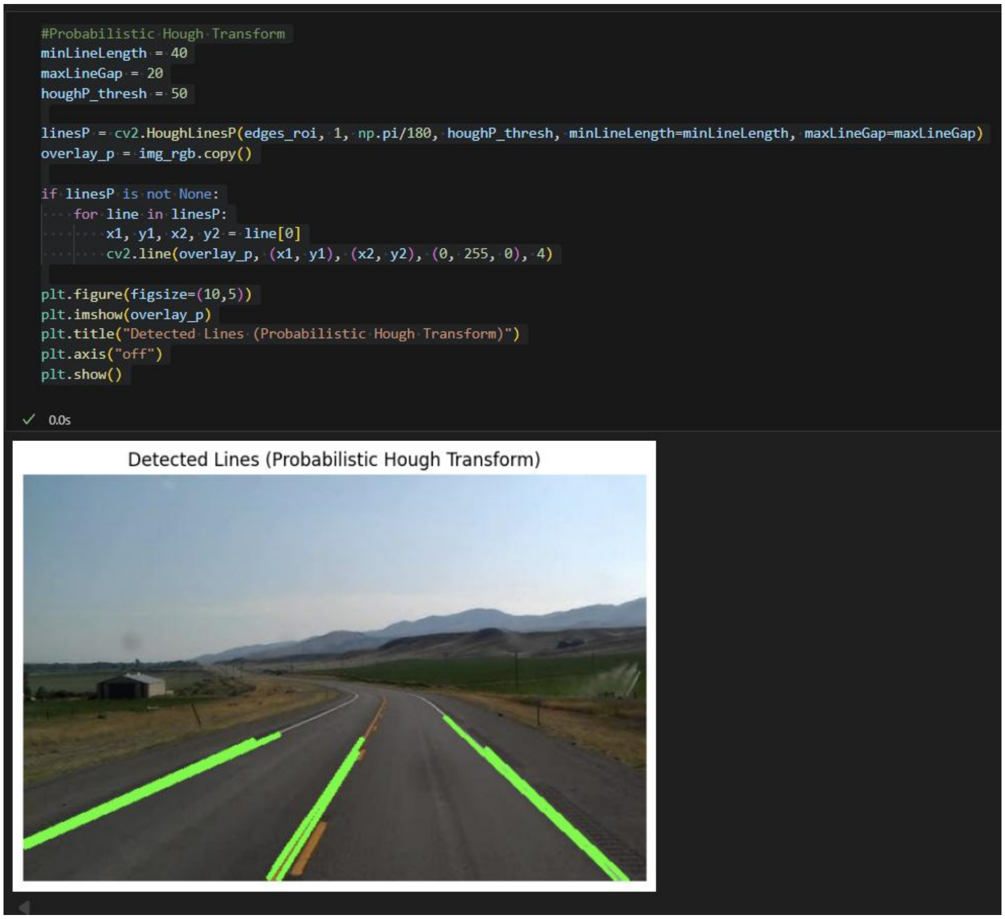


```
# Apply Standard Hough Transform (ρ-θ)
rho = 1
theta = np.pi / 180
threshold = 100

lines = cv2.HoughLines(edges_roi, rho, theta, threshold)

overlay_std = img_rgb.copy()

# Draw detected lines in green
if lines is not None:
    for rho, theta in lines[:,0]:
        a, b = np.cos(theta), np.sin(theta)
        x0, y0 = a * rho, b * rho
        x1, y1 = int(x0 + 800 * (-b)), int(y0 + 800 * (a))
        x2, y2 = int(x0 - 800 * (-b)), int(y0 - 800 * (a))
        cv2.line(overlay_std, (x1, y1), (x2, y2), (255, 0, 0), 2)

plt.figure(figsize=(10,6))
plt.imshow(overlay_std)
plt.title("Detected Lines (Standard Hough Transform, Green Lines)")
plt.axis("off")
plt.show()
```

✓ 0.0s

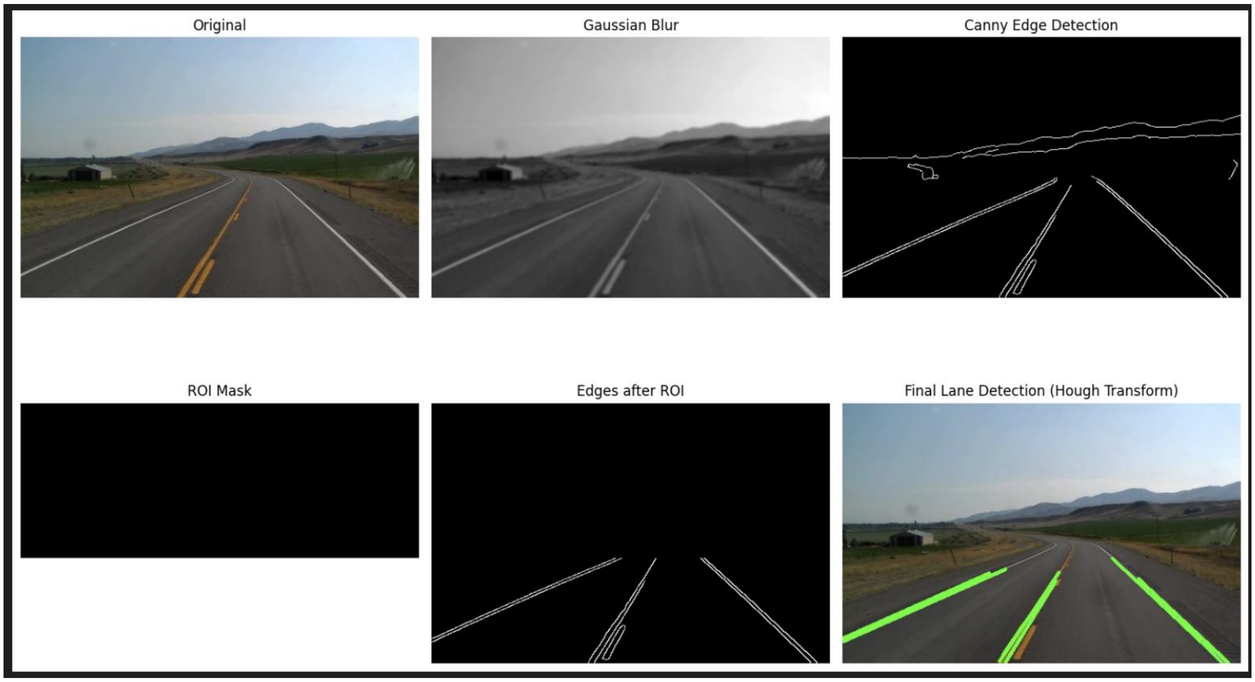
Detected Lines (Standard Hough Transform, Green Lines)





```python
#Probabilistic Hough Transform
minLineLength = 40
maxLineGap = 20
houghP_thresh = 50

linesP = cv2.HoughLinesP(edges_roi, 1, np.pi/180, houghP_thresh, minLineLength=minLineLength, maxLineGap=maxLineGap)
overlay_p = img_rgb.copy()

if linesP is not None:
    for line in linesP:
        x1, y1, x2, y2 = line[0]
        cv2.line(overlay_p, (x1, y1), (x2, y2), (0, 255, 0), 4)

plt.figure(figsize=(10,5))
plt.imshow(overlay_p)
plt.title("Detected Lines (Probabilistic Hough Transform)")
plt.axis("off")
plt.show()
```

✓ 0.0s


Original
Gaussian Blur
Canny Edge Detection
ROI Mask
Edges after ROI
Final Lane Detection (Hough Transform)