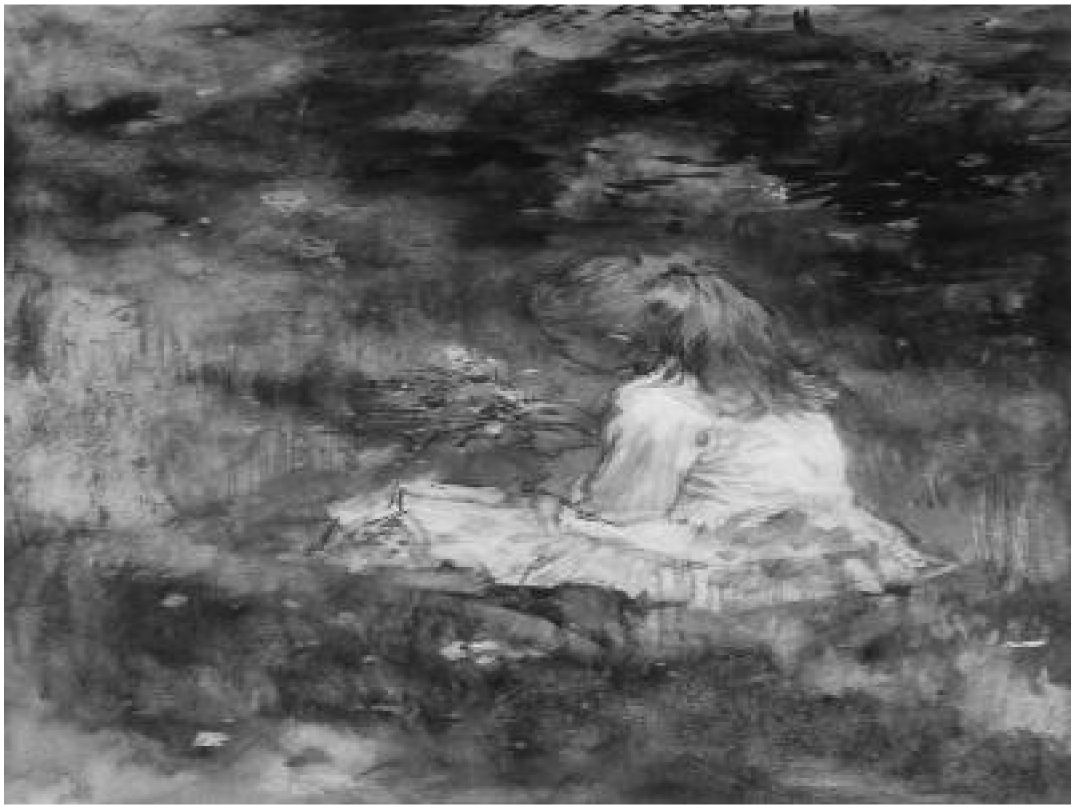
Lab Exercise 4: Edge Detection and Line Detection

• **Objective:** Detect edges and lines in an image using various algorithms.

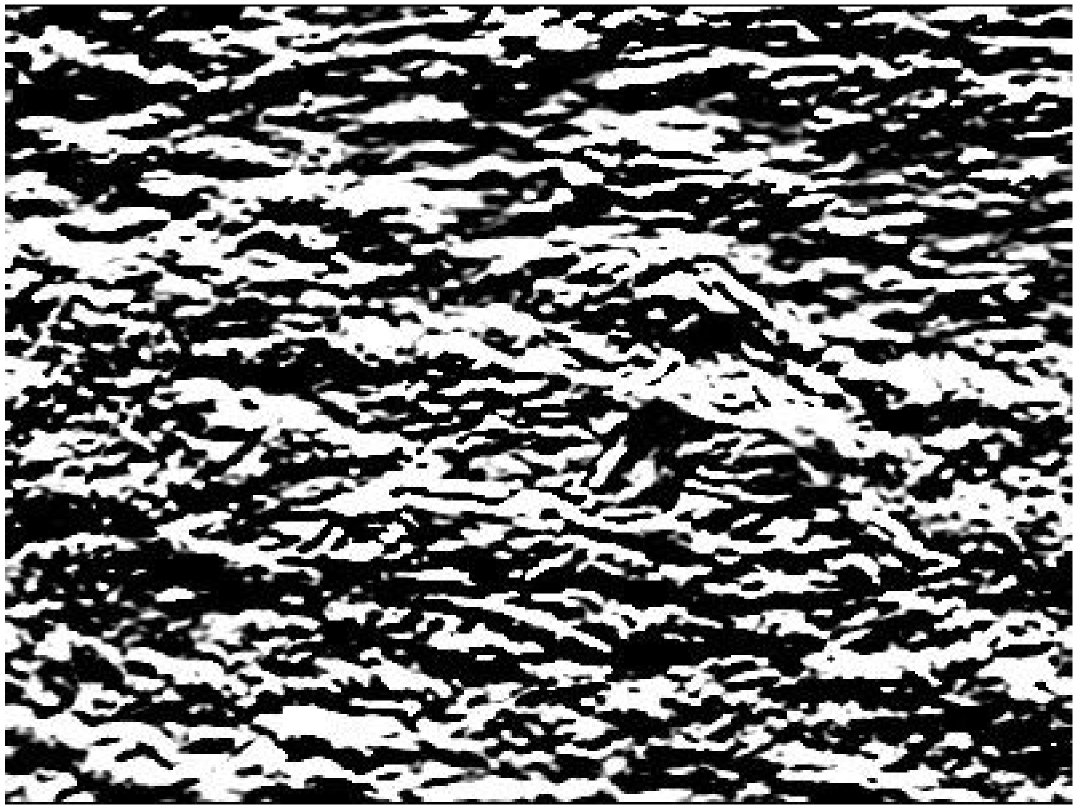
• **Task:** Implement Canny edge detection, Sobel, and Hough transform to detect edges and lines in a sample image.

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow # Import cv2_imshow from
google.colab.patches
import matplotlib.pyplot as plt

# Read the original image
img = cv2.imread('portrait.jpg')
# Resizing the image
Tiger_resized =cv2.resize(img, (400, 300))

# Save the resized image
cv2.imwrite("portrait Resized.jpg", Tiger_resized)
img_resized = cv2.imread('portrait Resized.jpg')
# Display original image using cv2_imshow instead of cv2.imshow
cv2_imshow(img_resized) # Use cv2_imshow to display the image in Colab
```

```
# Convert to graycsale
img_gray = cv2.cvtColor(img_resized, cv2.COLOR_BGR2GRAY)
# Blur the image for better edge detection
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)

cv2_imshow(img_gray) # Use cv2_imshow to display the image in Colab
```



```
# Sobel Edge Detection
sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0,
ksize=5) # Sobel Edge Detection on the X axis
sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1,
ksize=5) # Sobel Edge Detection on the Y axis
sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1,
ksize=5) # Combined X and Y Sobel Edge Detection

# Display Sobel Edge Detection Images

cv2_imshow(sobelx)
cv2_imshow( sobely)
cv2_imshow(sobelxy)
```



```
# Canny Edge Detection
edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200) #
Canny Edge Detection

# Display Canny Edge Detection Image with Title
plt.imshow(edges, cmap='gray') # Display the image using Matplotlib,
"cmap='gray'" display a grayscale image.
plt.title("portrait - Canny Edge Detection") # Set the title of the
image.
plt.axis('off') # Turn off the axis.
plt.show() # Show the image with the title in Colab.
```

portrait - Canny Edge Detection

Line Detection

```
# Read image
img = cv2.imread('Lanes.png', cv2.IMREAD_COLOR) # road.png is the
filename

# display the image
cv2_imshow(img)
```

```python
# Convert the image to gray-scale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2_imshow(gray)
```

```python
# Find the edges in the image using canny detector
edges = cv2.Canny(gray, 50, 200)

# Detect points that form a line
# Assuming max_slider should represent a threshold, set it to a
```

```
suitable value
threshold = 50  # You can adjust this value as needed
lines = cv2.HoughLinesP(edges, 1, np.pi/180, threshold,
minLineLength=10, maxLineGap=250)

# Draw lines on the image
for line in lines:
    x1, y1, x2, y2 = line[0]
    cv2.line(img, (x1, y1), (x2, y2), (255, 0, 0), 3)

# Show result
cv2_imshow(img)
```