edge-detection-and-line-detection

November 28, 2024

0.1 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.

[10]:
```
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('Tiger.jpg')
# Resizing the image
Tiger_resized =cv2.resize(img, (400, 300))

# Save the resized image
cv2.imwrite("Tiger Resized.jpg", Tiger_resized)
img_resized = cv2.imread('Tiger 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
cv2.waitKey(0)
```

[10]:
```
-1
```

[11]:
```
# 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)
```

[12]:
```
cv2_imshow(img_gray) # Use cv2_imshow to display the image in Colab
cv2.waitKey(0)
```

[12]:
```
-1
```

[13]:
```
# 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
```

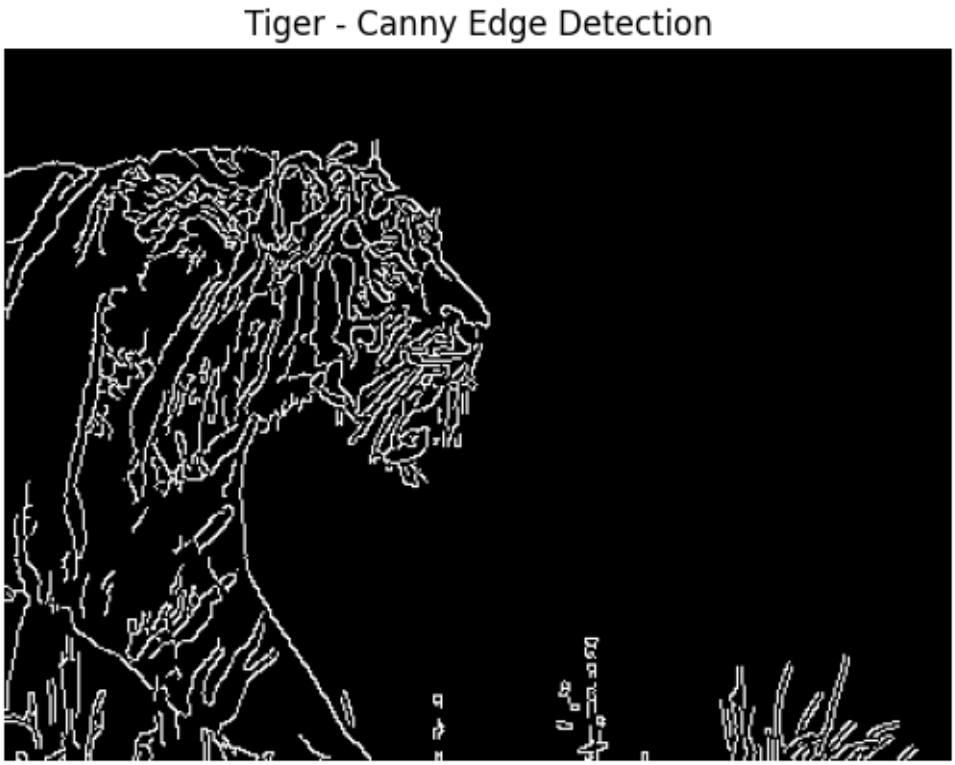
[14]:
```
# Display Sobel Edge Detection Images

cv2_imshow(sobelx)
cv2.waitKey(0)
cv2_imshow( sobely)
cv2.waitKey(0)
cv2_imshow(sobelxy)
cv2.waitKey(0)
```

[14]:
```
-1
```

[15]:
```
# Canny Edge Detection
edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200) # Canny Edge␣
↪Detection
```

[16]:
```
# 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("Tiger - 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.
```



0.2 Line Detection

[17]:
```
# Read image
img = cv2.imread('Lanes.png', cv2.IMREAD_COLOR) # road.png is the filename
```

[18]:
```
# display the image
cv2_imshow(img)
cv2.waitKey(0)
```

[18]: -1

[19]:
```
# Convert the image to gray-scale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2_imshow(gray)
cv2.waitKey(0)
```

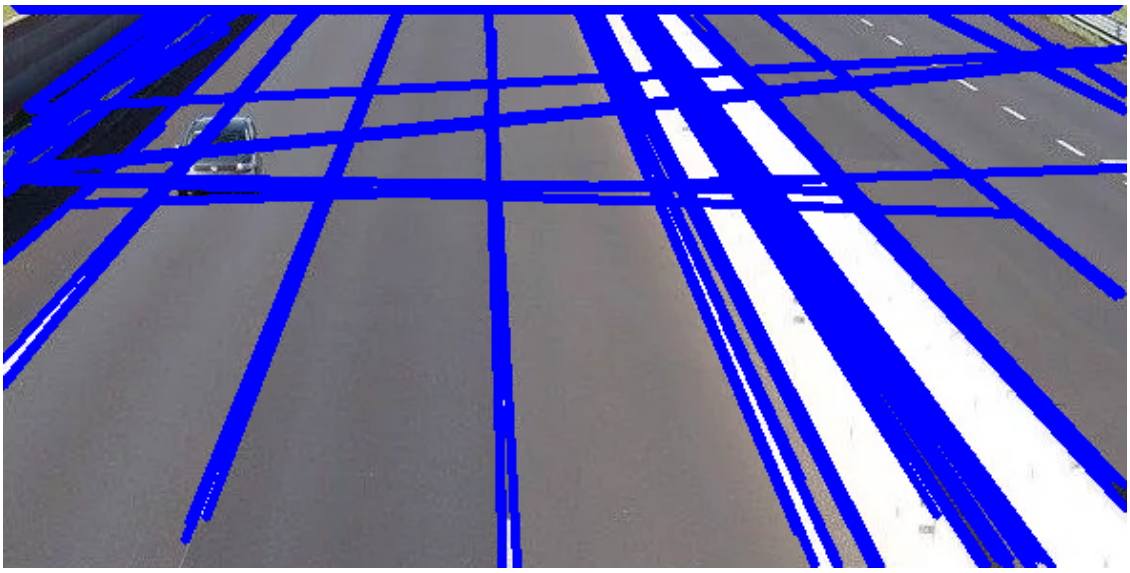
[19]: -1

[20]:
```
# Find the edges in the image using canny detector
edges = cv2.Canny(gray, 50, 200)
```

[21]:
```
# 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)
```

[22]:
```
# 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)
```

[23]:
```
# Show result
cv2_imshow(img)
```



[23]:
```

```