# LANE DETECTION FOR SELF DRIVING CARS

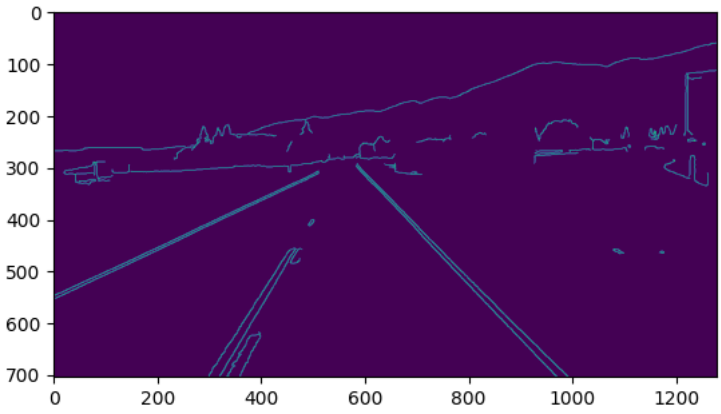
Importing the dependencies

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
```

IMAGE LANE DETECTION

```
1 image = cv2.imread('/home/mush/Computer_vision/project1/Lane detection/test_image.jpg')
2
3 gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
4 blur = cv2.GaussianBlur(gray, (5,5), 0)
5 canny_img = cv2.Canny(blur, 50, 150)
6 cv2.imshow('gray',gray)
7 cv2.waitKey(5000)
8 cv2.destroyAllWindows()
```

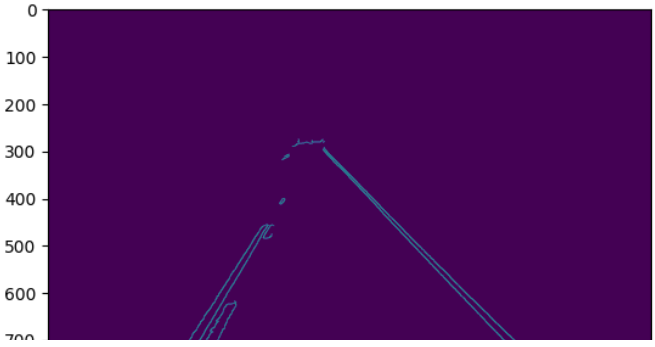
```
1 plt.imshow(canny_img)
2 plt.show()
```

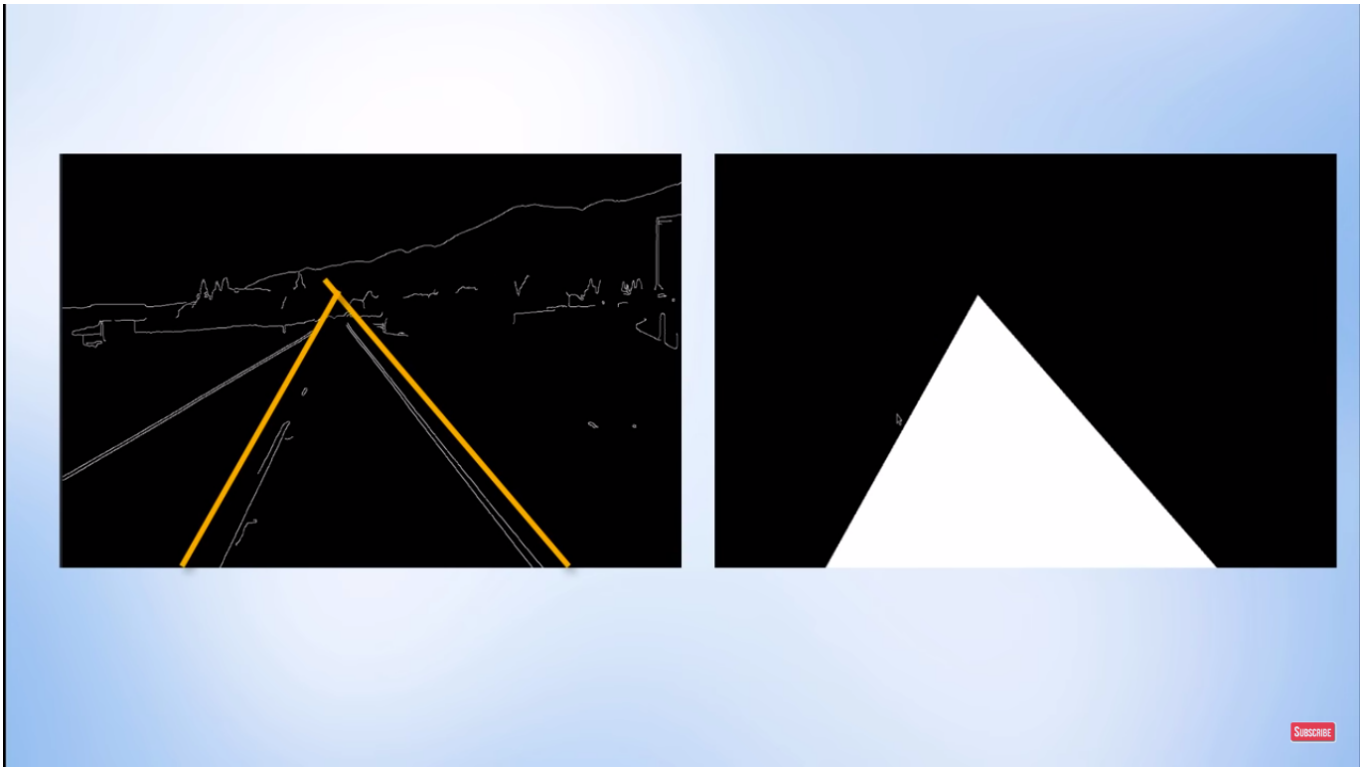


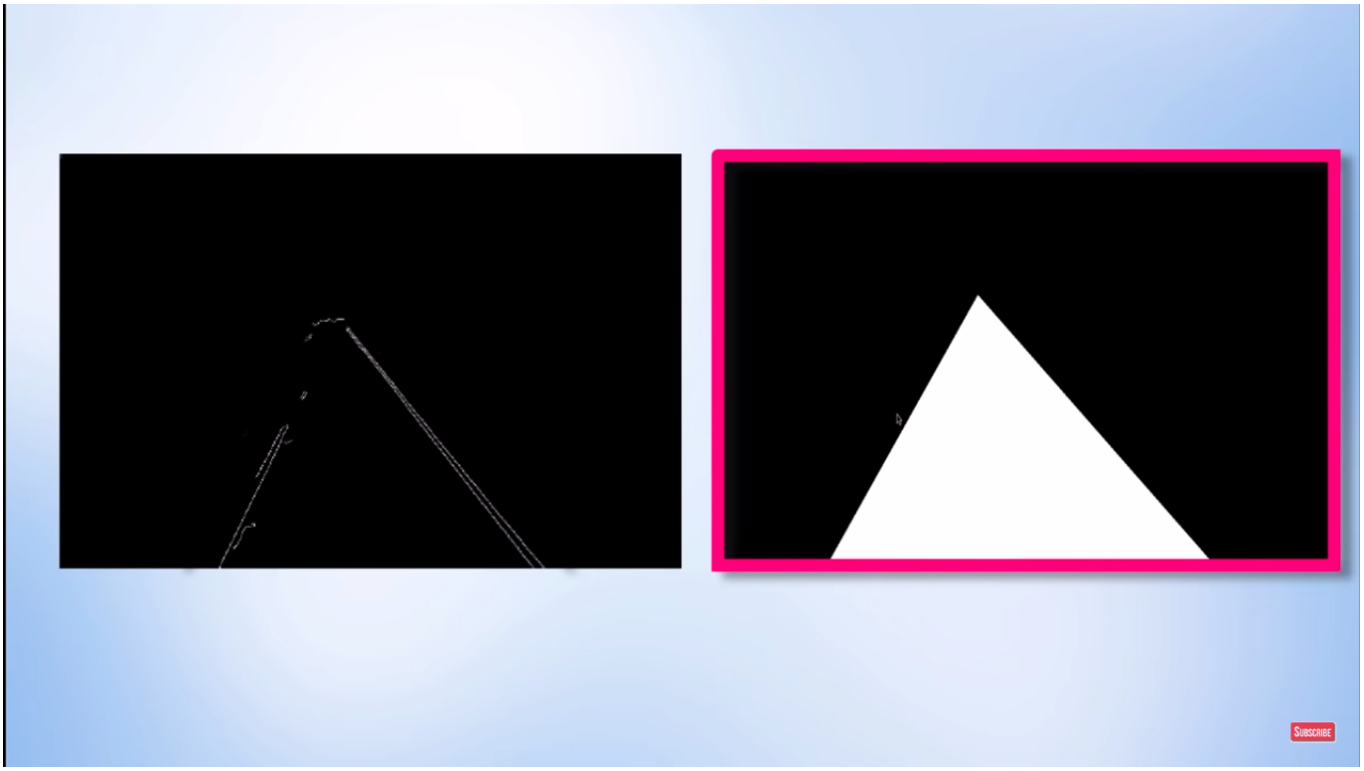

Double-click (or enter) to edit

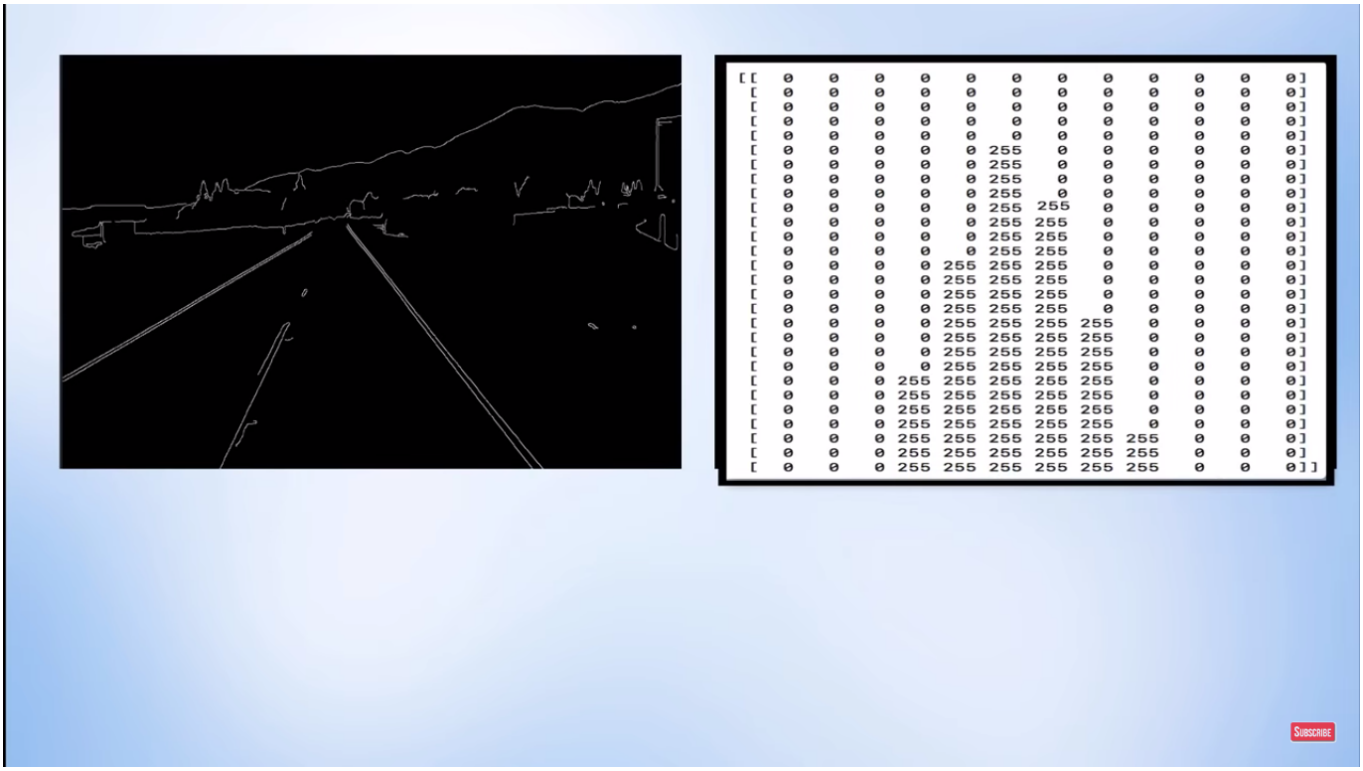
```
1 height = canny_img.shape[0]
2 polygons = np.array([[(200, height), (1100, height), (550, 250)]])
3 mask = np.zeros_like(canny_img)
4 cv2.fillPoly(mask, polygons, 255)
5 cropped_img = cv2.bitwise_and(canny_img, mask)
6 lines = cv2.HoughLinesP(cropped_img, 2, np.pi/180, 100, np.array([]), minLineLength=40, maxLineGap=5
7 plt.imshow(cropped_img)
```

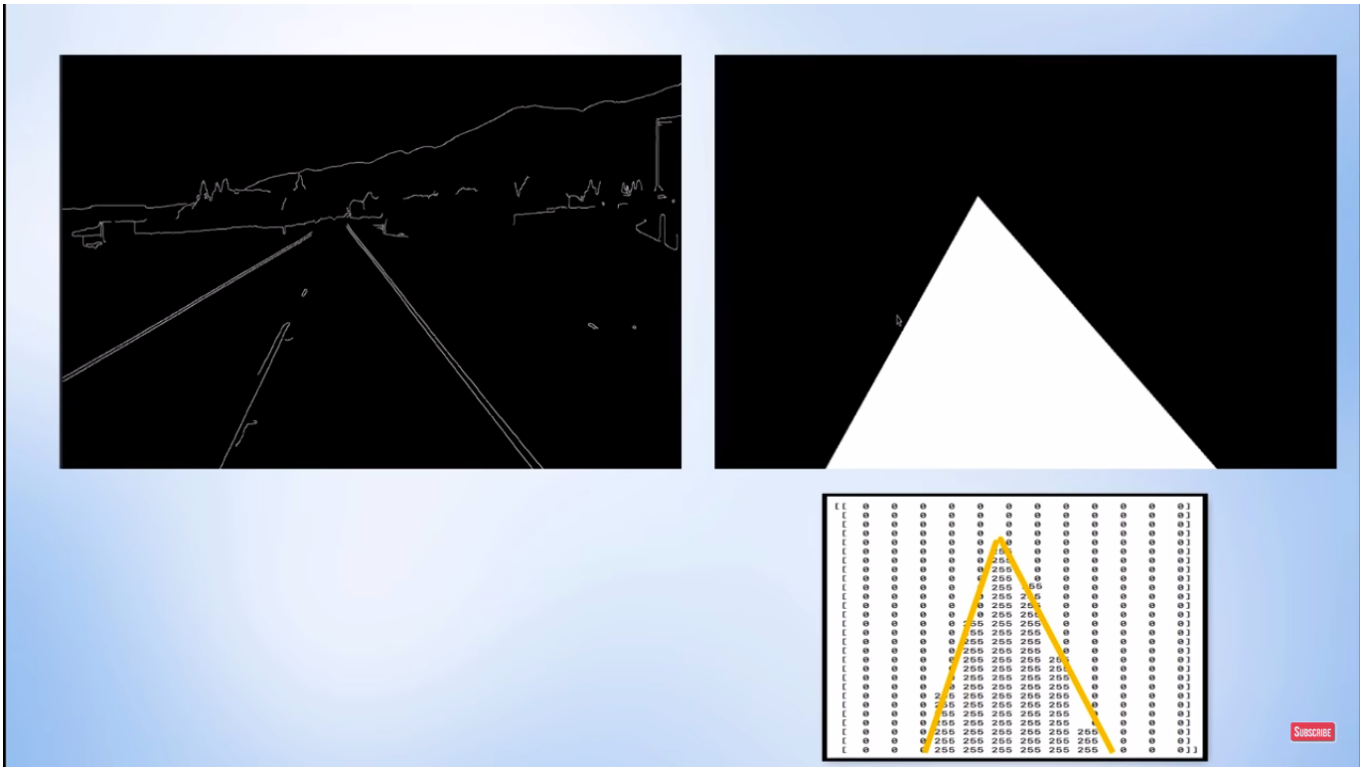
```
<matplotlib.image.AxesImage at 0x7fb8481dd4c0>
```

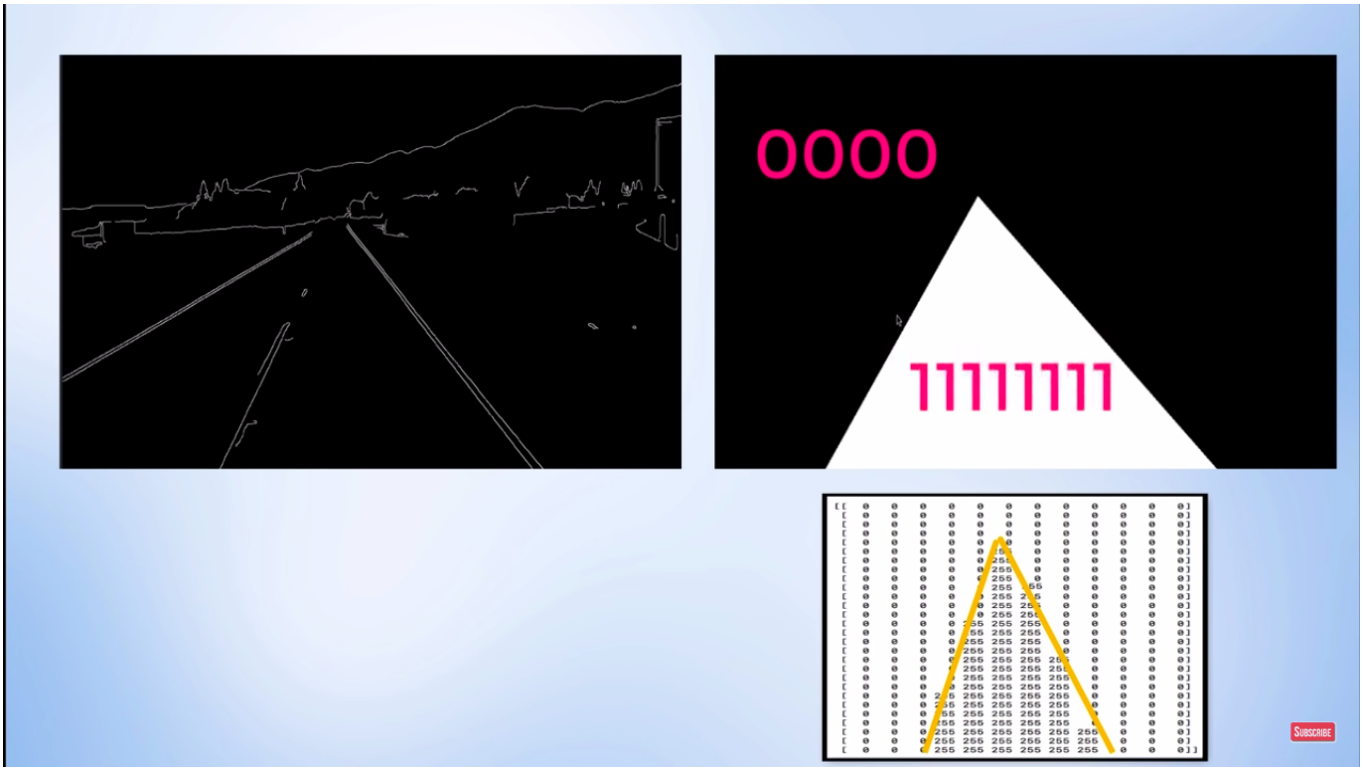



SUBSCRIBE


SUBSCRIBE






0000
11111111
Subscribe


0000
0000
11111111
Subscribe

VIDEO LANE DETECTION

```
cap=cv2.VideoCapture('/home/mush/Computer_vision/project1/Lane detection/Lane.mp4')
while True:
    success,frame=cap.read()
    cv2.imshow('orginal',frame)
    if cv2.waitKey(1) & 0xFF==ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

```
cap=cv2.VideoCapture('/home/mush/Computer_vision/project1/Lane detection/Lane.mp4')
```

```
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret:
        # slicing ht,wd.
        height,width= frame.shape[:2]
        # (ROI) vertices
        roi_vertices = np.array([[(200, height), (width/2, height/1.37), (width-300, height)]], np.i
        # Converting image to grayscale
        gray_img = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)

        # edges of objects in image is detected
        edges = cv2.Canny(gray_img, 50, 100)

        # Mask image to keep only ROI
        mask = np.zeros_like(edges)
        cv2.fillPoly(mask, roi_vertices, 255)
        masked_image = cv2.bitwise_and(edges, mask)

        # Using of HoughLinesP to detect lines in image
        lines = cv2.HoughLinesP(masked_image, rho=2, theta=np.pi/180, threshold=50, minLineLength=10
        # Drawing lines on blank image
        blank_img = np.zeros((height, width, 3), np.uint8)
        for line in lines:
            for x1, y1, x2, y2 in line:
                cv2.line(blank_img, (x1, y1), (x2, y2), (0, 255, 0), 2)

        # Overlay blank image with lines on original image
        output_img = cv2.addWeighted(frame, 0.8, blank_img, 1, 0.0)
```