Mujtaba Shahid Faizi

BSCS-5A

#131818

Lab 6 of Computer Vision

**CODE:**

**import** cv2  
**import** numpy **as** np  
  
video = cv2.VideoCapture(**"road\_car\_view.mp4"**)  
  
**while True**: *#start the video again after ending* ret, orig\_frame = video.read()  
 **if not** ret:  
 video = cv2.VideoCapture(**"road\_car\_view.mp4"**)  
 **continue** frame = cv2.GaussianBlur(orig\_frame, (5, 5), 0) *#to remove noise* hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)  
 low\_yellow = np.array([18, 94, 140]) *#range of yellow (i.e. from low to high yellow)* up\_yellow = np.array([48, 255, 255])  
 mask = cv2.inRange(hsv, low\_yellow, up\_yellow)  
 edges = cv2.Canny(mask, 75, 150)  
  
 lines = cv2.HoughLinesP(edges, 1, np.pi / 180 , 100, 50, maxLineGap=50)  
 **if** lines **is not None**:  
 **for** line **in** lines:  
 x1, y1, x2, y2 = line[0]  
 cv2.line(frame, (x1, y1), (x2, y2), (0, 255, 0), 5) *#green lines with 5 thickness* cv2.imshow(**"frame"**, frame)  
 cv2.imshow(**"edges"**, edges)  
  
 key = cv2.waitKey(25)  
 **if** key == 27: *#press esc key to EXIT* **break**video.release()  
cv2.destroyAllWindows()

**SCREENSHOT:**

* cv2.HoughLinesP() 🡪 First parameter is the Input image after canny edge. Second and third parameters are \rho and \theta accuracies respectively. Fourth argument is the threshold, which means minimum vote it should get for it to be considered as a line, where number of votes depend upon number of points on the line, and it is set as 100. So it represents the minimum length of line that should be detected. Fifth is the minimum length of line which is set 50. Line segments shorter than this are rejected. Sixth is the Maximum allowed gap between line segments to treat them as single line which is also set as 50.

