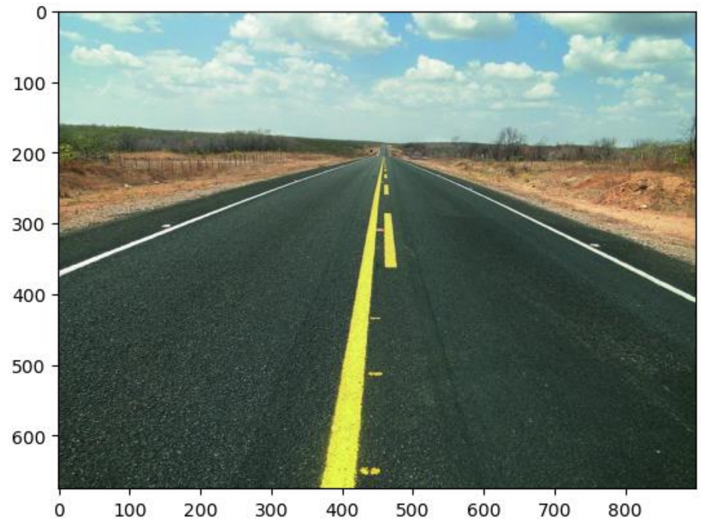
Visual Recognition

Assignment1

Q1a. Lane Detection:

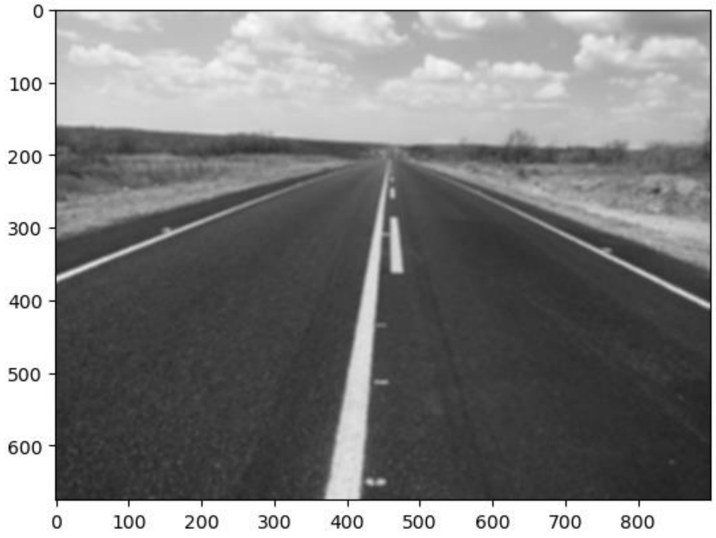
Used Image:



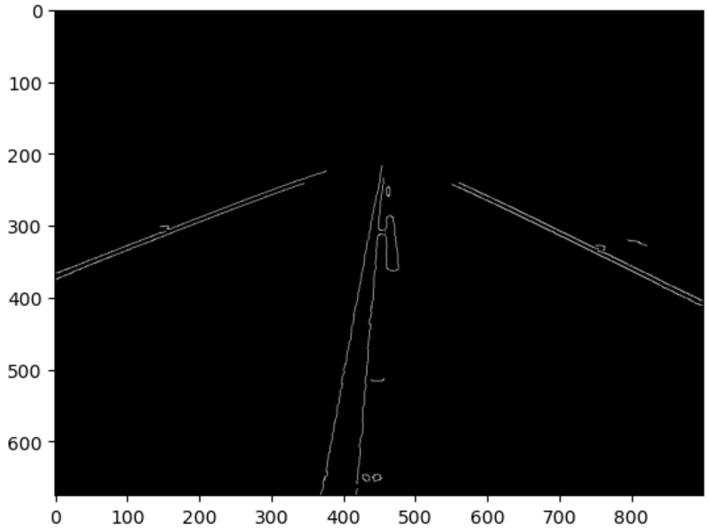
Pre Processing

- First, I converted this image into RGB color scheme as openCV by default uses BGR color scheme.
- Then I converted this image into gray scale image

- Then for removing noise I used GaussianBlur with a kernel size of(7,7) .I tried with different sized kernels like (3,3) ,(5,5),(9,9) . But (7,7) gave me best result. While using (3,3) or (5,5) noises like lines on road or bushes outside the road was coming (when I appiled Canny Edge Detector upon it) & for (9,9) the middle lane was getting removed while applying canny edge detector.
- After blurring using (7,7) kernel:



- Then I applied Canny Edge Detector upon it using the thresold values 100 & 200. I found this values after manually doing lot of trial and errors. 100 & 200 gave me the best results.

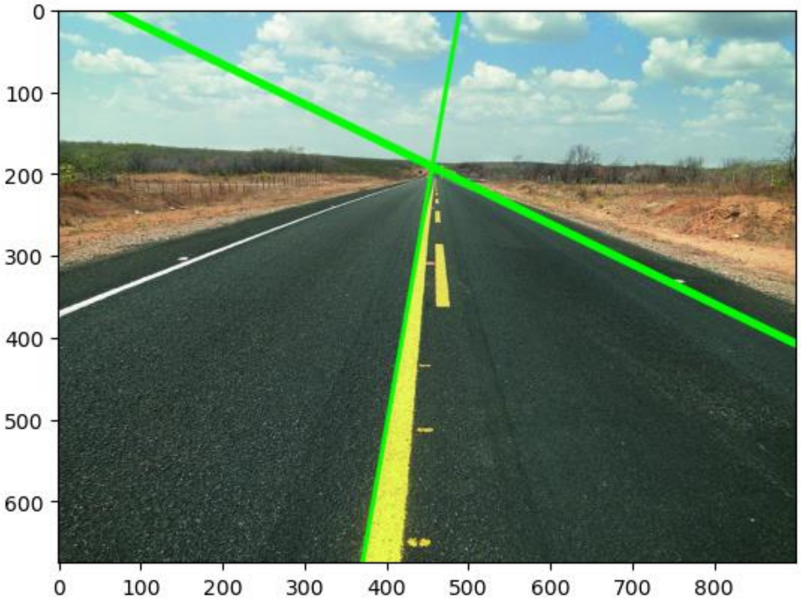


Line Fitting:

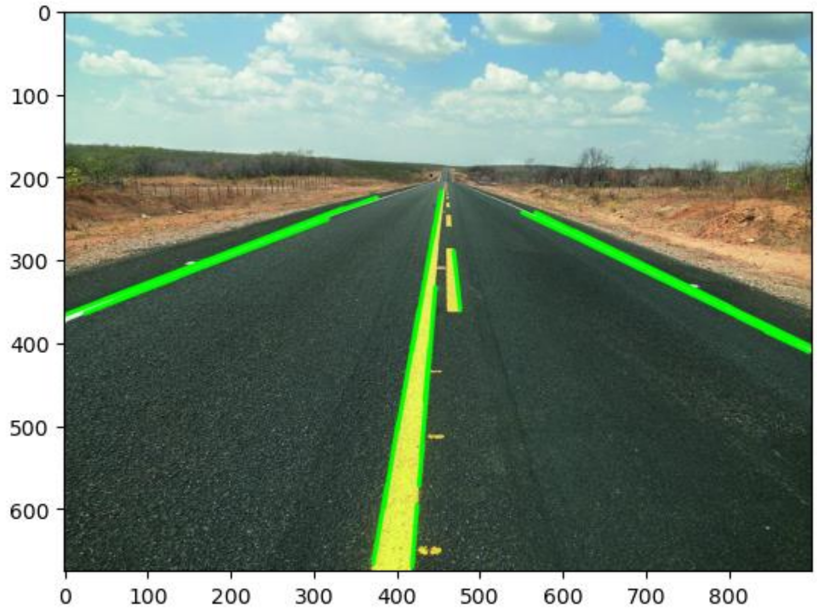
Then I basically tried 3 methods to fit lines on the detected edge pixels of lanes (basically I was trying to detect the right lane of the road).

- **Method 1(Standard Hough Line Transform):** I used cv2.HoughLines() on the above pre-processed image with rh0=1, theta= 1 degree and thresold = 150 (minimum number of votes to count that point of hough sapce as a line in image space. Then I got 3 lines with their respective rho and theta. Then I drew them on the source image Inside a for loop which goes through all the edges, for each edge storied the value of cos(theta) in a and the value of sin(theta) in b. x0 stores the value rcos(theta). y0 stores the value rsin(theta). x1 stores the rounded off value of (rcos(theta)-1000sin(theta)). y1 stores the rounded off value of (rsin(theta)+1000cos(theta)). x2 stores the rounded off value of (rcos(theta)+1000sin(theta)). y2 stores the rounded off value of (rsin(theta)-1000cos(theta)). Then used cv.line function by taking (x1, y1), (x2, y2) as two points.

But in this method, I was getting infinite lines which were not serving the purpose of lane detection.



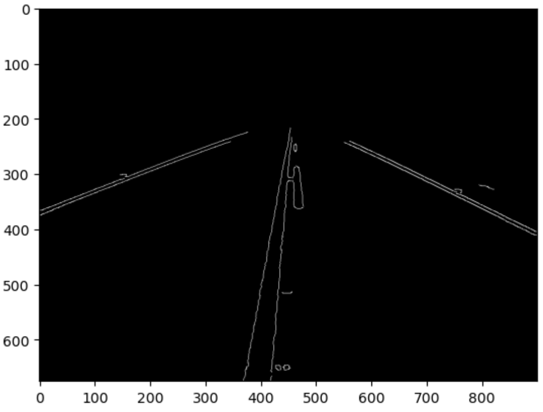
- **Method 2(Probabilistic Hough Line Transform):** In this method I used cv2.HoughLinesP rh0=1, theta= 1 degree and minLineLength = 50 and maxLineGap = 10 on canny edge detected image . I got 22 lines with their respective two end points. So I drew them on the source image using cv2.line().



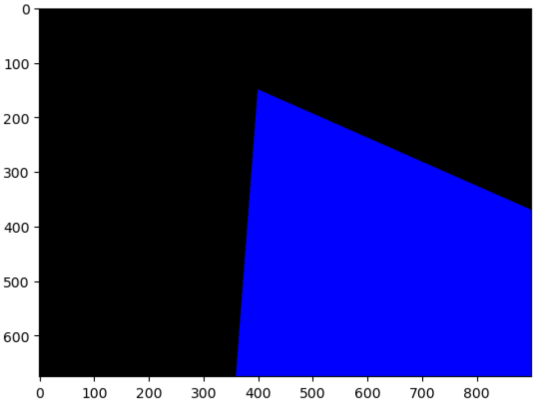
Here the problem of infinite lines got solved but also the left lane was also getting detected. Our goal was only to detect the right lane.

- **Method 3(Masking & Probabilistic Hough Line Transform):**

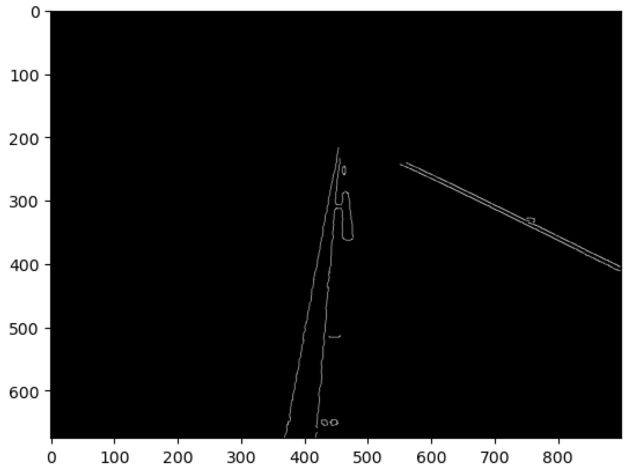
Here I first create an array of zeros which has the shape of the source image. Then I drew a polygon whose 4 corners' co-ordinates were (360, 675),(400, 150),(width of source image,370), (width of source image,height of source image). This was the mask. Then I did bitwise_and between canny edge detected image and mask and basically got the below image.



After canny edge detection

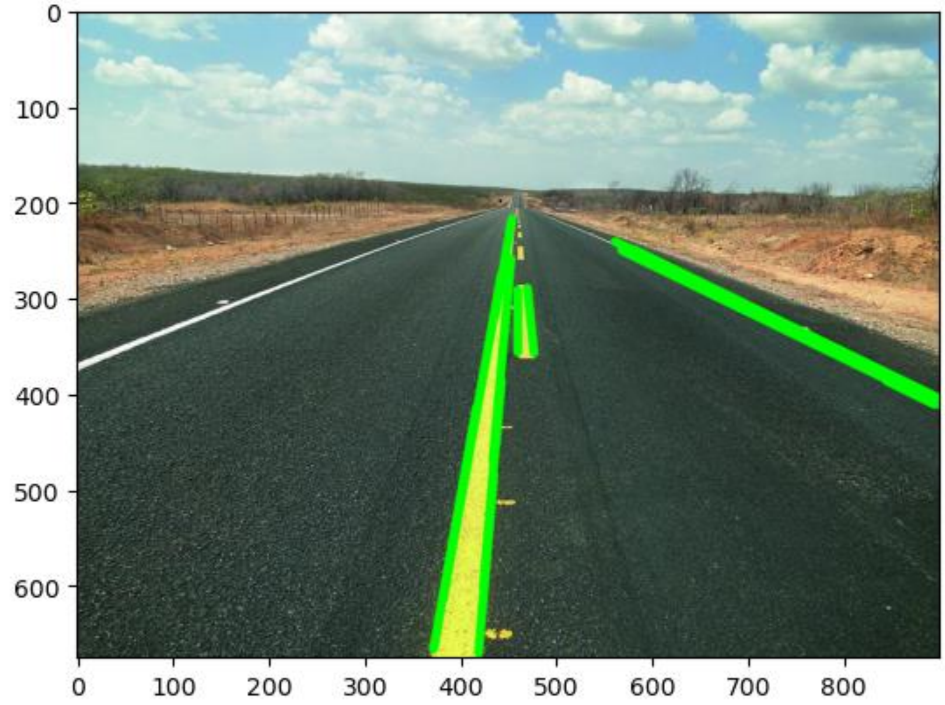


Mask



Resultant Image

Then I again drew the lines on the source image using cv2.HoughLinesP() like I did in Method 2.



As we can see here right lane of the road is properly detected.