Introduction

The goal of this task was to detect lanes by traditional image processing. My approach for the task had been to convert the image into greyscale, blurring of the image, detection of edges, masking of our region of interest to get edges of lanes, detection of lines using Hough transform and finally making masks and highlighting the lanes over the original image. The detailed approach had been mentioned further explain each damn line of code.

Importing Libraries and Loading Image

First, importing the necessary libraries:

I used:

- `cv2` for image processing, as it provides powerful tools like edge detection and transformations.
- `numpy` to handle matrix operations since images are essentially multi-dimensional arrays.
- `matplotlib.pyplot` to visualize the intermediate processing steps.

I loaded the image stored locally as image.png using `cv2.imread()`. Then, I displayed it using matplotlib to verify that the image was correctly loaded.

Image Processing

Grey Scale and Blurring

Since color information is not necessary for lane detection, the image was converted to grayscale using `cv2.cvtColor()`. Following this I applied Gaussian Blurring. This step was necessary to reduce noise, i.e., as one can see there are bushes around which are adding more unnecessary edges in the image, which is slightly problematic for the approach I had been using.

Although on increasing the kernel size to about 25 I was able to reduce the other noise to almost 0, but too much blurring can cause the edge detection harder, so a good result was obtained from kernel size 9.

Following this I tried to detect edges from the image.

Edge Detection

To detect edges, I applied the Canny edge detection algorithm using `cv2.Canny()` which resulted in a black and white image representing the edges from the image.

Since lane markings are generally in a fixed region of the image, which can vary based on the position and image of camera, and the width of the road, but assuming these to be constants I defined a trapezium shaped region of interest. Then masked the edges detected image with this. Which resulted me exactly the edges of the lanes and now I was ready to detect the lines from the image.

Hough Transform

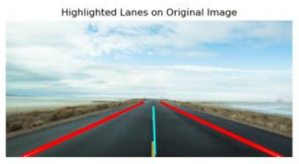
To detect lane lines, I used the Hough Transform function `cv2.HoughLinesP()`. Then I chose the middle line, that is with maximum absolute to be representative of yellow and prepared a mask accordingly. Now I was able to see that the absolute slopes of the white lanes were all less than 1 for being a safer side I chose it to be 2 and chose the lines for the white lanes and accordingly prepare black and white mask.

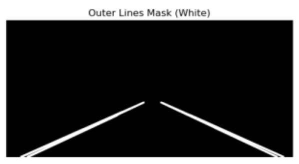
Drawing Lines

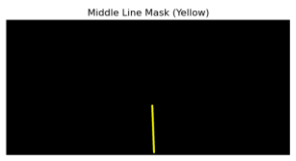
I just drew cyan colored line for highlighting the yellow lane and red colored lines for highlighting the white on the original image.

Summary

The task was executed successfully primarily using OpenCV Library using traditional image processing techniques and the workflow followed had been inspired from the shared resources for preparation. The following were the results-




Outer Lines Mask (White)


Middle Line Mask (Yellow)