Perception for Autonomous Robots
ENPM673
Project 2

Name: Balaji Selvakumar
UID : 118545745
Date: 04/04/2022
email : balase22@umd.edu

Problem 1 – Histogram Equalization:

In this project we aim to improve the quality and lighting conditions of the image sequence provided .
To achieve this goal, histogram equalization can distribute the intensities of the image and make them evenly spread.

Fig 1: Input image

The idea of histogram equalizations is cumulative distribution function (cdf), which is

$$cdf_x(i) = \sum_{j<=i} h(j)/N$$

This indicates the fraction of pixels with intensity less than or equal to i, and assumes the image has N pixels.

The image is split into three color spaces (BGR) and intensities(histogram) of the color spaces are equalized individually.

The outcome is again merged back to single frame and is shown below.

Fig 2: Image after Histogram equalization

In Fig 2, it is easy to notice the upper region of the image can show more details than original image. The darker region is also brighter than original image.

For adaptive histogram equalizations, it is based on histogram equalizations. This method divides image into several pieces and utilize histogram equalizations on each piece.

Fig 3 : Adaptive Histogram Equalization

Compared to the Normal Histogram equalization, the Adaptive Histogram Equalization works a lot better .We can view more details on the trees, people, and the car in the background.

Problem 2 – Straight Lane Detection:

In this task, we are given a short video contains lanes on the road. The goal is to classify the dashed and solid lines on the road. Green is for solid line and red is for dashed line.

To achieve this goal, the following steps should be implemented

1. Leave yellow and white pixels to remove unnecessary edges

2. Convert image into gray-scale

3. Select a smaller region which contains the lanes we want (create a mask with respect to dimensions of the lane)

4.Apply Gaussian blur to smoothen the image

5. Threshold the blurred video to get the threshold the pixels values in the video

6. Use HoughLinesP function to detect the lines in above region

7. (rho: 1, theta: np.pi/180, threshold: 15, minLineLength: 1, maxLineGap: 10)

8. Separate the lines to right and left lines using slope gradient equation

9. To be able to trace a full line and connect lane markings on the image, we must be able to distinguish left from right lanes. Fortunately, there is a trivial way to do so. If you carefully look the image (may be easier with the canny segmented images), you can derive the gradient (i.e slope) of any left or right lane line:

* left lane: as x value (i.e. width) increases, y value (i.e. height) decreases: slope must thus be negative

* right lane: as x value (i.e. width) increases, y value (i.e. height) increases: slope must thus be positive

We can therefore define a function that separates lines into a left and right one

10. Draw the lines green for solid and red for dashed

In the seventh step, HoughLinesP function is implemented. Hough Transform can detect the lines even they are broken or distorted. The theory behind it is that using (ϱ, θ) represents all the lines in the image. ϱ is the length of the line and θ is the angle of the line. By voting the cell, we can assume that there is a line in the image with maximum votes.

Fig 4 : Input video for testing

Fig 5: threshed Lanes for detection

Fig 6 : Output showing green for solid and red for dotted line

Problem 3 :Turn Prediction

This task is to detect the curved lanes and predict the turn. A short video is given. There are yellow and white colored lanes in video for predicting turn. To achieve this goal, the following steps are necessary.

1. Apply filter to show yellow and white colors only and ignore unnecessary edges
2. Convert the image to grayscale
3. Mask the video with a smaller region which contains the lanes we want (create a mask with respect to dimensions of the lane)
4. Use Gaussian Blur to smooth the image
5. Threshold the image from 150 to 255 and select the interested region
6. Apply the changes made in HLS and LAB color space
7. stitch the image with the filtered image
8. Warp the image to get the birds eye perspective
9. A series of filters are applied to improvise the lane quality(gaussian blur followed by thresholding, dialation and erosion)

10. Line fitting is done using polyfit function and the lane projection is superimposed over the original image

10. Find the radius of curvature to estimate the turn

11. Draw the lanes, the area between two lanes, and the arrows in the middle

Fig 7: Input lane frame

Fig 8 : Filtered input

Fig 9:

Masked(ROI)+ Threshed Frame

Fig 10 : Birds Eye perspective)

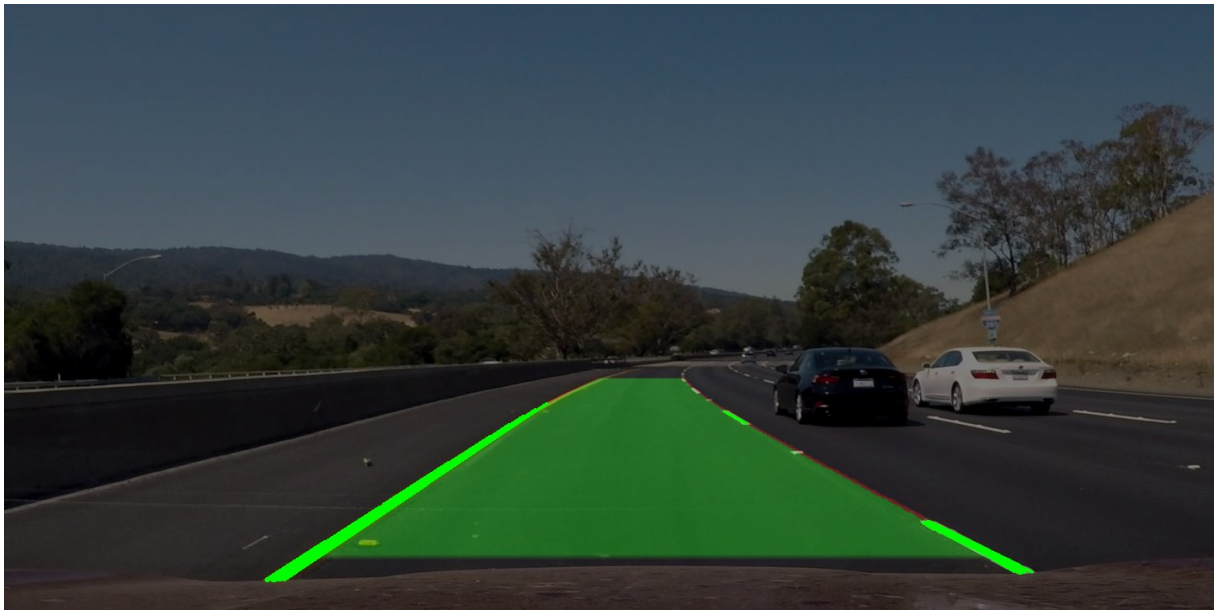

Fig 11: Turn Super Imposition