ENPM 673: Perception for Autonomous Robots
Project 2 – Report

Hemanth Joseph Raj – 117518955

Problem 1 – Part a: Histogram Equalization
Pipeline towards the solution

1. Save the image frames in a folder and rename them 0 to 24
2. Read each frame and convert to grayscale
3. Create number of pixels and histogram variables
4. Iterate through each pixel and find out its value and add it to the tally in histogram
5. Perform probability distribution and calculate cumulative frequencies
6. Equalize and round the values
7. Go through each pixel again find their values and replace them with the equalized intensity corresponding to that value

Output: Video link

(Fig. 1 Image After Histogram Equalization)

Problem 1 – Part b: Adaptive Histogram Equalization
Pipeline towards the solution

1. Save the image frames in a folder and rename them 0 to 24
2. Read each frame and convert to grayscale
3. Break down each frame into smaller blocks
4. Now for each block create number of pixels and histogram variables
5. Iterate through each pixel of the block and find out its value and add it to the tally in histogram
6. Perform probability distribution and calculate cumulative frequencies
7. Equalize and round the values
8. Go through each pixel again find their values and replace them with the equalized intensity corresponding to that value
9. Repeat steps 4 to 8 for each block
10. Now put all the blocks together

Output: Video link

(Fig. 2 Image After Adaptive Histogram Equalization)

Problem 2: Lane Detection

Pipeline towards the solution

1. Create a capture object and capture the image frame by frame
2. Convert it to grayscale, and apply Gaussian blur
3. Detect edges using cv.Canny edge detector
4. Select a region of interest (triangular in this case)
5. First we detect only solid lines
6. Using cv.HoughLinesP and setting minium length very high to eliminate small lines to detect only solid lines
7. Average out the beginning and end coordinates of the line to get single coordinates of beginning and end points
8. Find its slope and intercept
9. Using the slope and intercept to fit the entire frame along the line and finding new extreme points
10. Using these extreme lines to draw a green line denoting a solid line
11. Now to detect only dashed lines
12. Using cv.HoughLinesP to detect all lines
13. Using step 8, we have the slope of solid lines
14. If slope of solid lines is positive only save those lines whose slope is negative (denotes dashed line)
15. If slope of solid lines is negative only save those lines whose slope is positive (denotes dashed line)
16. Average out the beginning and end coordinates of the line to get single coordinates of beginning and end points
17. Find its slope and intercept
18. Using the slope and intercept to fit the entire frame along the line and finding new extreme points
19. Using these extreme lines to draw a red line denoting a dashed line

Output: Video link

(Fig. 3 Lanes Detected)

Problem 3: Lanes Detection with Turn Prediction
Pipeline towards the solution

1. Create a capture object and capture the image frame by frame
2. Convert to grayscale and threshold the image
3. Create a three channel threshold to show in final output
4. Using cv.findHomography and cv.warpPerspective perform homography of a selected region of the frame and warp its perspective
5. Convert the warped perspective image into a grayscale, threshold it and convert it to a three channel threshold to perform further operation
6. The orginal three channel threshold from step 5 will be added to the final output, we make a copy of it for futher processing
7. Creating a blank image like the warped image
8. First we detect the lane with white colour
9. Convert the warped image into grayscale and threshold it
10. Using np.where find the coordinates of pixels whose values are 255 which correspond to white pixels
11. Using np.polyfit fit a polynomial curve of degree 2 on these points
12. Find out the x and y coordinates of points aloing the curve and make it into an array
13. Using cv.polylines draw the curve on a.) blank image & b.) copy of the warped three channel threshold image
14. Using cv.circle draw circles along the detected points for the white line
15. Calculate the radius of curvature of white lane
16. Next, we detect the lane with yellow colour
17. Convert the image into the HSV domain to get only yellow lines
18. Using np.where find the coordinates of pixels whose values are 255 which correspond to yellow pixels
19. Using np.polyfit fit a polynomial curve of degree 2 on these points
20. Find out the x and y coordinates of points aloing the curve and make it into an array

21. Using cv.polylines draw the curve on a.) blank image & b.) copy of the warped three channel threshold image
22. Using cv.circle draw circles along the detected points for the yellow line
23. Calculate the radius of curvature of yellow lane
24. Take average of both the radii of curvatures to get the radius of curvature of the road
25. Using cv.fillPoly, fill the area between both the lanes i.e the road into red
26. Perform inverse homography and inverse warp perspective to put all the detected lines and plots and overlay using cv.add to the original frame
27. Based on the curvature predict the turn of the road
28. Put all the necessary outputs into smaller windows and create a frame of multiple outputs
29. Annotate each window with necessary window names and mention them and also the radius of curvature of the road

Output: Video link

(Fig. 4 Lanes Detected With Turn Prediction)

Problems Faced:

- In problem 1, I couldn't save the frames as a video using cv.VideoWriter
- In problem 2, I am facing a divide by zero error when calculating slope at a particular frame

How Generalized is my solution?

In problem 2

1. One must be solid and the other dashed
2. The Field of View must be similar to the problem statement

In problem 3

1. One lane should be yellow and the other white
2. The Field of View must be similar to the problem statement

My Understanding of Homography and its usage in the project

Homography is an important concept in the field of perception which is used to convert an image from one orientation to another whil maintaing the relation of every pixel with its

corresponding pixels when converting to a new orientation. Used to convert aspects or entire images from the camera POV to world POV. Or even vice versae (in case of inverse homography).

In problem 3, homography was used to take a certain section of the road and convert it to world POV and then perform image processing on it and then inverse homography was performed to get back the orginal frame.

My Understanding of Hough Lines and its usage in the project

Hough transform is a concept used to detect and shape when represented in a mathematical form. A line can be defined using two terms, the perpendicular distance from the origin (rho) and the angle the perpendicular distance makes with the horizontal axis (theta) in counter-clockwise direction. So the OpenCV function cv2.HoughLinesP (probablistic Hough transform) takes in informaton such as the input image, how manys lines to output, the rho and theta values, thresholding parameter, minimum and maximum line length to detect lines and return the starting and ending coordinates of all lines detected in the image.

In problem 2, probablistic hough transform was used to get the coordinates of the lines in the input image which was used to calculate a single line and draw it onto the frame to represent the lane.