ENPM 673: Project 2

Param Dave
Master of Robotics
University of Maryland , College Park
Email: pdave1@umd.edu

I. Problem 1

One of the problems cameras face is the inability to see in the dark or in environment with lack of visibility. We try to solve this problem by some rudimentary methods namely Histogram Equalization and Adaptive Histogram Equalization. In Histogram Equalization we try to enhance the contrast by taking the histogram of the image and distribute the frequencies by calculating Cumulative Distribution function,normalizing it for each bin and multiplying it with 255.

In Adaptive Histogram Equalization, we apply Histogram Equalization to a grid of 8x8 block. We divide the image into 8x8 blocks and apply Histogram Equalization on each block.

The Outputs of Histogram Equalization is shown in figure 1 and figure 2 and outputs of Adaptive Histogram Equalization is shown in figure 3 and figure 4.

II. Problem 2

We are presented a simply lane detection problem. To solve the problem, we will first select a region of interest(ROI) as the camera does not move much compare to the road. We select a triangle which contains the road. The shape is triangle as this is a projective transformation and parallel lines meet at vanishing point. After selecting the ROI, we separate the yellow and white lanes. This can be done in RGB space with bit complexity but for more robust and general case separating colors based on HSV values seems appropriate. To separate yellow color we take lower HSV values as and upper HSV values as .For white color we take upper and lower HSV values as and respectively.These values were selected by manually creating a track-bar to calculate the appropriate combination of upper and lower HSV values for individual colors. We separate the yellow color and threshold it and create a yellow mask which contains only the yellow line in a binary image. We do the same for white color.Finally using binary_and to add the two mass. The output of masks is shown in figure 5. the final output of problem 2 is shown in figure 6.

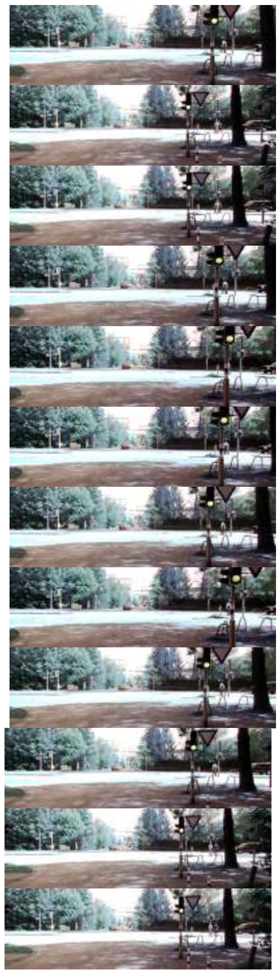
Fig. 1: Histogram Equalization of images 1-12

After this step, we detect the edges using canny edge detection and give its output to cv2.HoughLinesP(). Line detection using hough space is a voting scheme which we will discuss in the appendix. The output of the cv2.HoughLinesP() gives us coordinates of detected lines.We separate the lines based on their slope and intercept.For lines having slope and intercept withing a permissible limit, we add them to a list and take average of them. Finally we have two lines corresponding to each line on the road. We save this lines in another list in case the next frame does not contain any detected lines. We will use the previous lanes for that case. To classify the dashed and dot lanes, we go to yellow mask image and separate it into left and right segment.For each segment we find the number of white pixels and which side has the more pixels we classify that lines as solid lane and the other lane as dot. This is done by previous knowledge of slope and intercept of the lines. Final output of the lane detection is shown in figure 4. The code also works well if the video is flipped and generalizes well as the yellow and white lanes are detected using HSV values. The only problem would be occlusion due to other vehicles.

Fig. 2: Histogram Equalization of images 13-25

III. Problem 3

The problem deals with finding the curvature of the road. We first select the region of interest and in that region of interest, warp the road to get a birds eye view. We detect yellow and white lanes. One way is to detect the yellow and white curve and perform curve fitting using np.polyfit. In order to do that we make use of sliding window technique to select the point in the window. We initialize a window by using histogram of the image. The top twp peaks of the image will give us the two lane coordinates.We initialize the windows and calculate the mean of all the points in the window and save all such point in a list.We slide the window throughout the line. We feed these lines to cv2.polyfit and it gives us three values corresponding to a,b,c in the equation 1

$$y = ax^2 + bx + c \tag{1}$$

We calculate the curvature of the line by following equation 2. We warp back the lane to original frame along with the

Fig. 3: Adaptive Histogram Equalization of images 1-12

curve and apply cv2.polyfit.

$$R = \frac{[1 + (\frac{dy}{dx})^2]^{3/2}}{|\frac{d^2y}{dx^2}|}$$

(2)

We find the left and right curvatures and find the relative direction the car should go depending on the difference between curvatures of left and right lanes based on a threshold limit. The masked image is shown in figure 7.The warped image of this masked image is shown in figure 8. The output of sliding window method after curve fitting is shown in figure 9 .The final video output is shown in figure 10 . The combined video output is shown in figure 11.

Appendix

A. Homography

Homography relates the transformation between two planes.Homography lets us relate two cameras viewing the same planar surface; Both the cameras and the surface that

Fig. 4: Adaptive Histogram Equalization of images 13-25

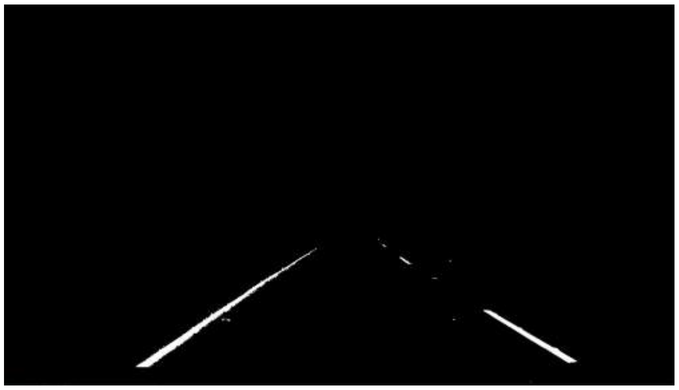

Fig. 5: Mask Output

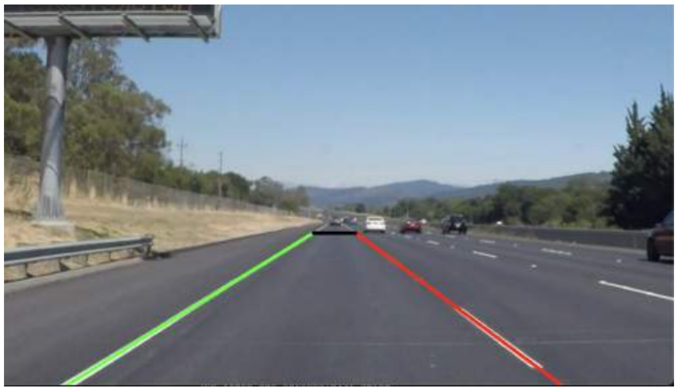

Fig. 6: Problem 2 final output

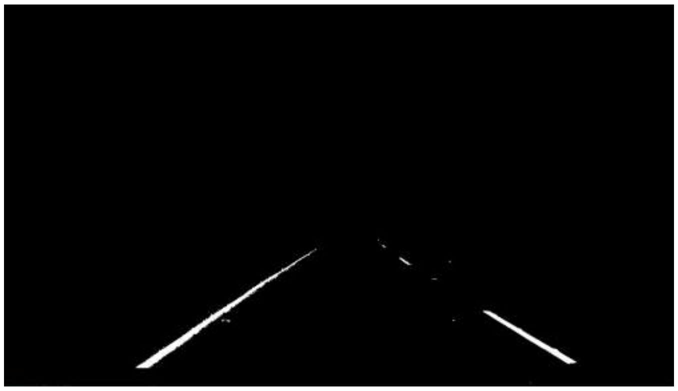

Fig. 7: Mask Output

they view (generate images of) are located in the world-view coordinates. In other words, two 2D images are related by a homography H, if both view the same plane from a different angle. The homography relationship does not depend on the scene being viewed. The homography matrix is a 3x3 matrix but with 8 DoF (degrees of freedom) as it is estimated up to a scale. It allows us to shift from one view to another view of the same scene by multiplying the Homography matrix with the points in one view to find their corresponding locations in another view . It is generally normalized as shown in equation

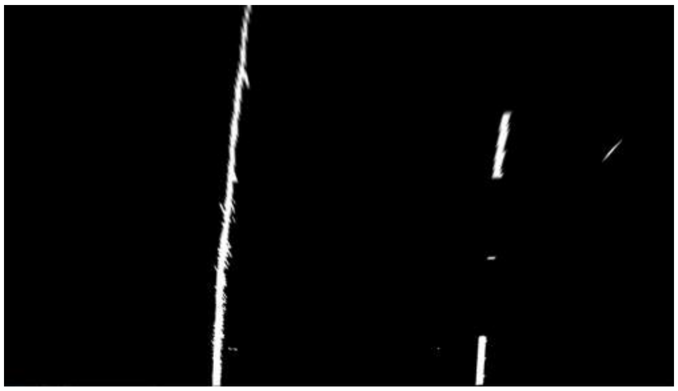

Fig. 8: Warped Output

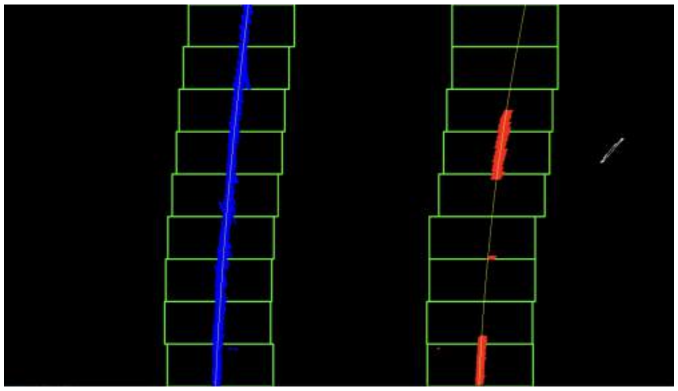

Fig. 9: Polyfit Output

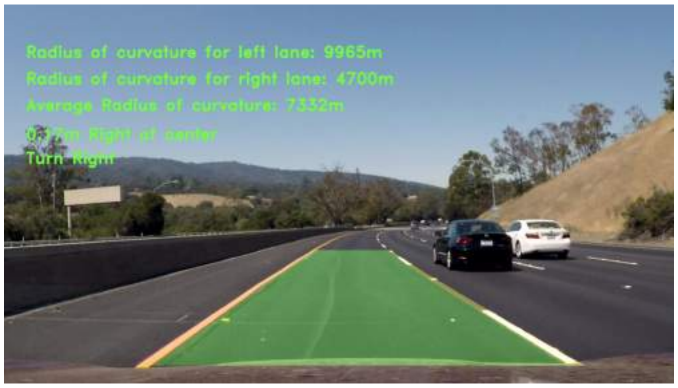
Radius of curvature for left lane: 9965m
Radius of curvature for right lane: 4700m
Average Radius of curvature: 7332m
Right of center
Turn Right

Fig. 10: Final Output

.

$$h_{11}^2 + h_{12}^2 + h_{13}^2 + h_{21}^2 + h_{22}^2 + h_{23}^2 + h_{31}^2 + h_{32}^2 + h_{33}^2 = 1. \quad (3)$$

Figure 12 shows the relation between two images and importance of homography.

B. Hough Transform

Hough Line Transformation is basically a line detection algorithm. Hough transformation is a voting scheme where all the points vote to finally get a line. We define a Cartesian co-ordinate system of parameters (b,m) , where b is the intercept and m is the slope of line. y = mx +b . We make use of Polar coordinate system: Parameters: (r,). So a point in Cartesian coordinate is a line in polar coordinates and vise verse. Thus we pass multiple edges of the images in Cartesian coordinates and these are converted into polar coordinates which becomes a set of lines, we find the point where maximum lines intersect and that point corresponds to our line in Cartesian space. We can find multiple lines by calculating multiple intersections.

Figure 13 shows two points on a line in Cartesian space and same points as curves in polar coordinates.

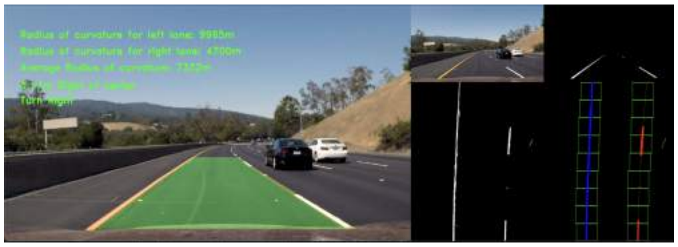

Fig. 11: Combined Output

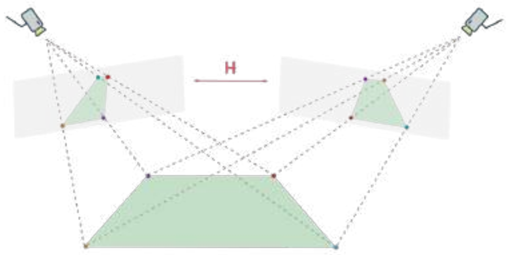

Fig. 12: Visualization of Homography

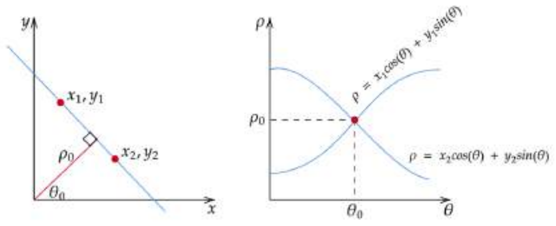
y
x
x_1, y_1
x_2, y_2
ρ_0
θ_0
ρ
$\rho = x_1\cos(\theta) + y_1\sin(\theta)$
$\rho = x_2\cos(\theta) + y_2\sin(\theta)$
ρ_0
θ_0
θ

Fig. 13: Line in Polar Coordinates