ENPM 673: Perception for Autonomous Robots
PROJECT 2 - REPORT

Rishabh Mukund (UID: 117556124)

1) Comparison between histogram equalization and adaptive histogram equalization for a given set of images.

To perform histogram equalization, we can use any linear function that is strictly increasing or decreasing. Based on the histogram data, I used the cumulative summation function which is a strictly increasing function.

For histogram equalization, I calculated the histogram of the frame, then performed cumsum on the generated histogram. For equalization, I multiplied the cumsum function corresponding to the intensity value of the given pixel with 255.

For adaptive histogram, we divide the frame into tiles and perform histogram individually for each tile. This helps us to retain the parts of the image that are already with good contrast. But it is slow as compared to normal histogram equalization.

The main difficulty was to perform adaptive histogram equalization as we have to pad the input frame accordingly. Also adaptive histogram equalization will give an output in which the corner of tiles are very evident and not smooth. To overcome this problem, we should perform contrast limiting and interpolation.

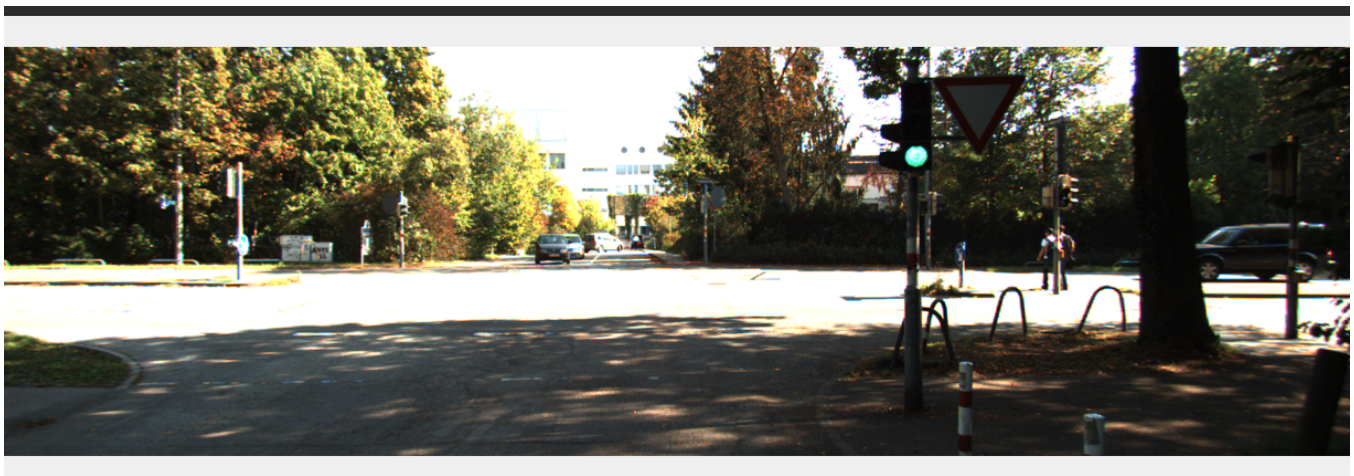


Fig 1: Input Frame

Fig 2: After Histogram Equalization

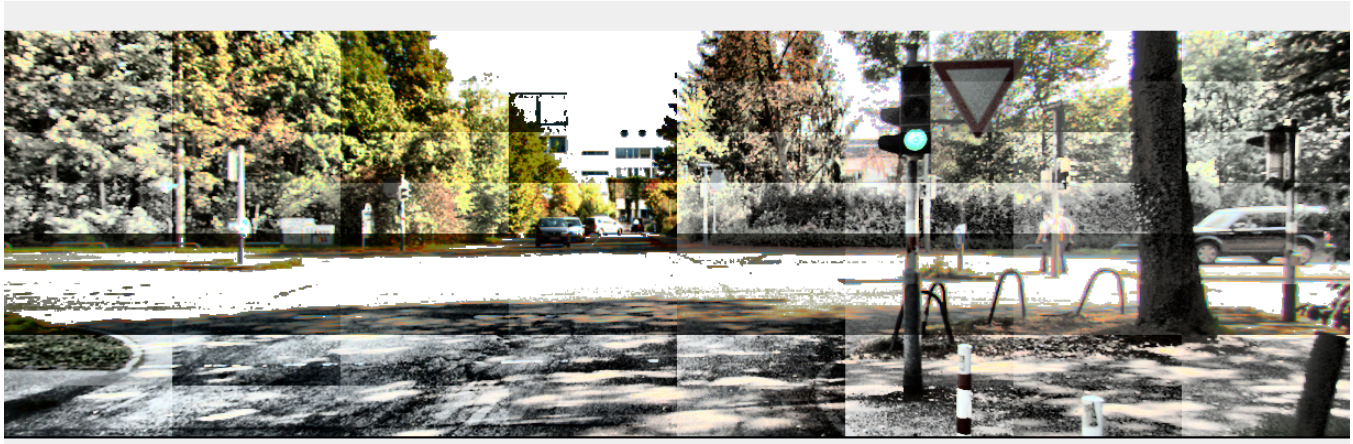


Fig 3: Adaptive Histogram Equalization (Tiles = 8)

2) Straight Lane Detection

Pipeline:

BRG Frame -> grayscale -> Thresholding -> Region of interest (triangle) -> Hough Transform -> Sort lines based on slope -> find the longest line on each side -> compare the lengths to find longest lines -> Color the longest line in green and other in red.

The difficulties encountered are to fine tune the hough transform parameters, and then detecting the dashed line from solid line.

Fig 4: (Top left to bottom right)
Input Image, Grayscale with region of interest, Thresholding, Output

3) Turn Prediction

Pipeline:

BRG Frame -> HSV frame -> White and Yellow Detection -> Region of interest (trapezoid) -> Warp Image using homography -> Lane points detection using sliding window -> Polyfit the lane points -> Get Curvature of turn and shade the region between the curves -> inverse warping to BRG frame

The difficulties encountered are implementing sliding windows, and getting radius of curvature based on the poly fir lines. I was also difficult to detect yellow lines when the white space in between the video.

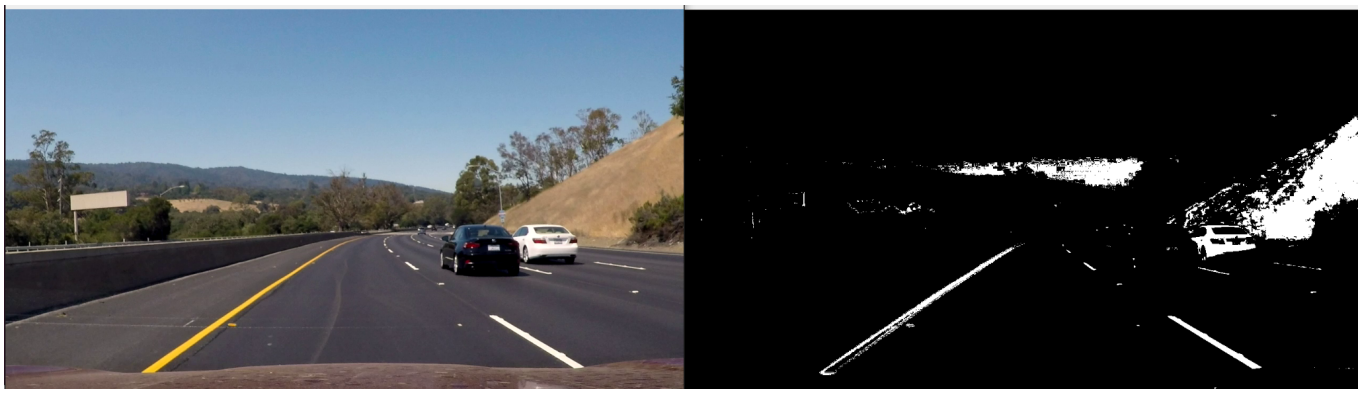
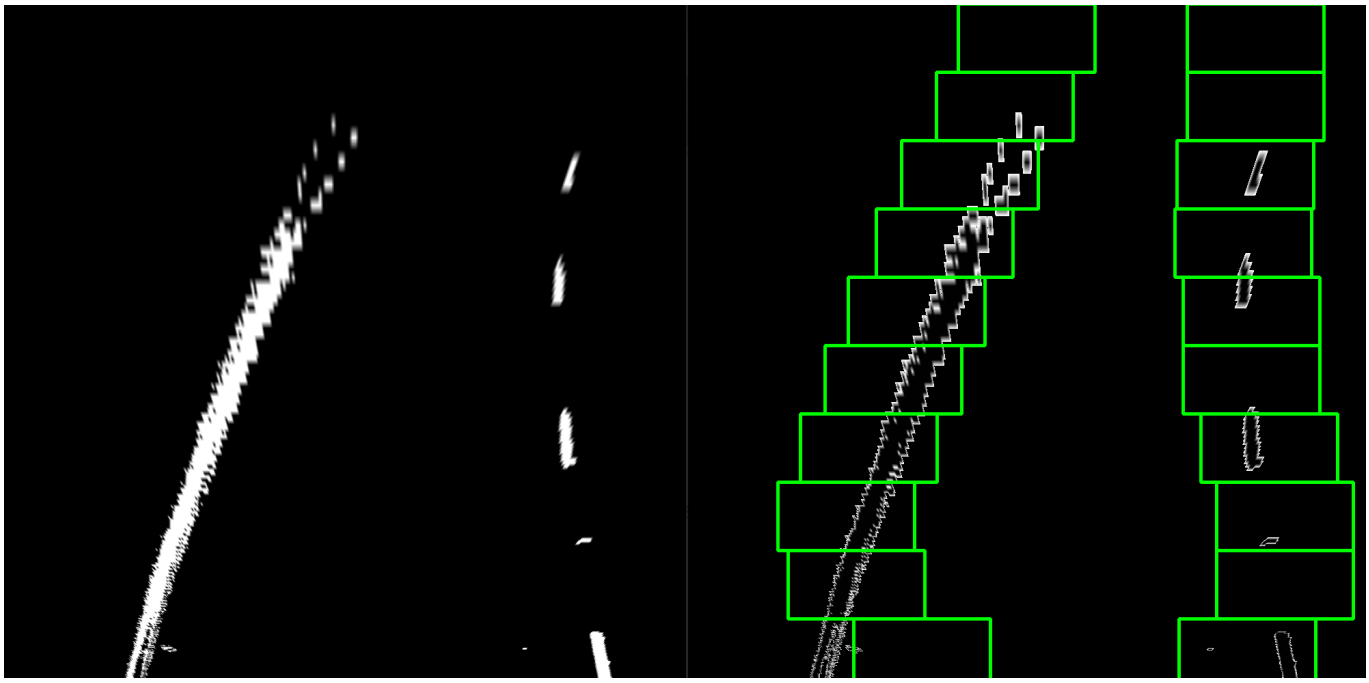
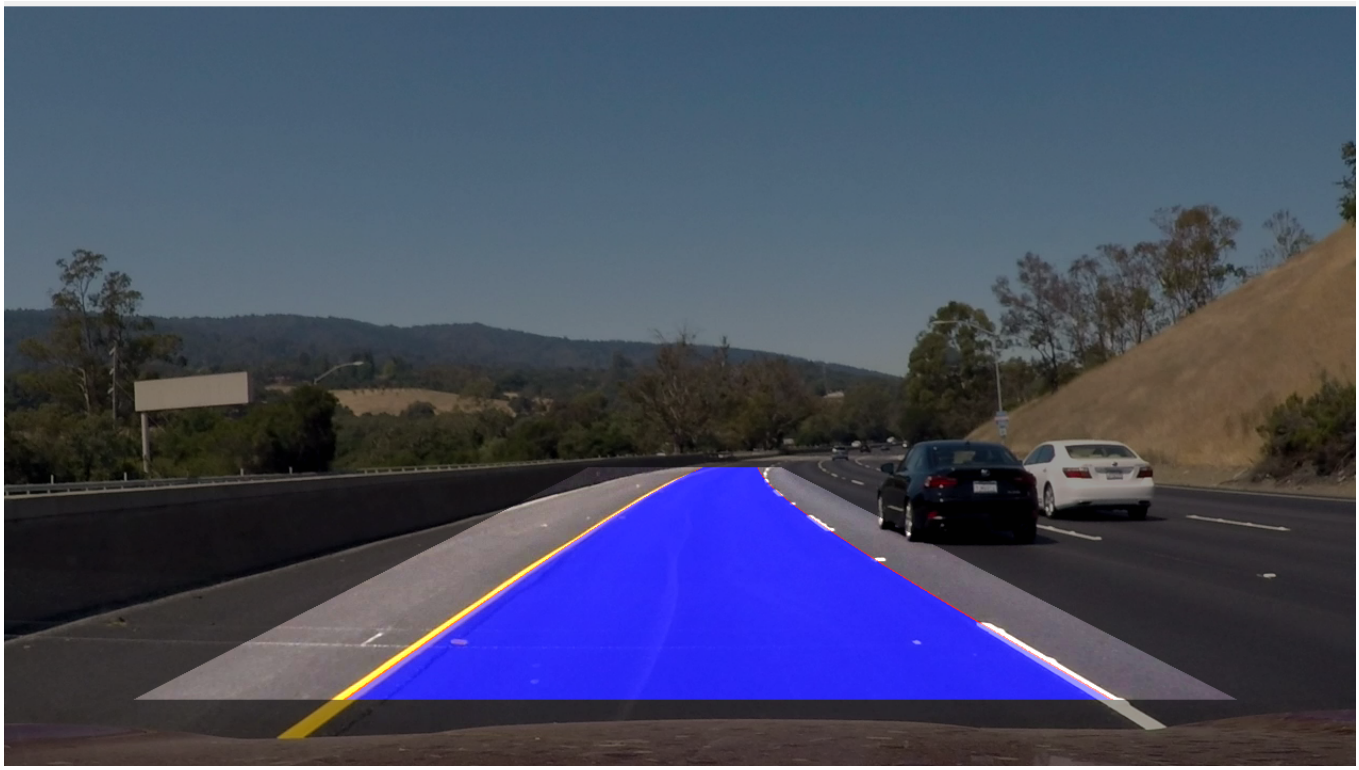


Fig: Input Image, Filtering out white and yellow





A. What is homography and how it works

Homography (a.k.a Perspective Transform) enables us to go from one view of a scene to another by multiplying the Homography matrix with the points in one view to discover their corresponding locations in another view.

The desired top-down view may be obtained by applying the perspective transform, which effectively maps a set of four points bounding the lane region in our input image to the desired collection of points. These points are decided on a case-by-case basis, with the primary influencing variables being the camera's position in the vehicle and its field of view.

To recognize lanes, we should arrange our visual systems to view the road ahead in such a way that they appear to be looking at it from a bird's eye perspective. This will assist us in estimating the curvature of the road and, as a result, anticipate the steering angle over the following few hundred meters. Another advantage of using a top-down view is that it eliminates the problem where lane lines appear to overlap, despite the fact that lane lines remain infinitely parallel lines as long as the road continues.

B. How Hough Line works

Hough Lines Transform is the key method used to detect lines. It is very helpful in many Computer Vision applications. The processed picture must be binary in order to utilize the HoughLines Transform. However, we'd want to look for straight lines in a color image. As a result, the most typical technique is to grayscale the image first and then identify edges. The Hough Lines technique may then use this mask of edges to generate a set of straight lines discovered on an image.

The straight line can be represented by two parameters. The simplest and most widely used pair of parameters is (a, b) which correspond to slope and intercept. The line can be described as:

$$y = ax + b$$

We can also describe the line using the pair (ρ, θ) in the polar system. The first parameter, ρ, is the shortest distance from the origin to the line (approaching the line perpendicularly). The second, θ, is the angle between x-axis and the distance line. The line can be described as:

$$\rho = x \cos(\theta) + y \sin(\theta)$$

One of the benefits of such representation is that we can describe vertical lines by a point (ρ, θ), we call it Hough Space

In Hough Space
Straight line -> Point
Point -> Sine wave
To identify lines, we draw points in the image space that form a line, and in the Hough space, we get a set of sinusoids. A straight line's points will meet at one point. It suggests that we should look for intersections in Hough space to find candidates for being a straight line.

The Hough space of an image is divided into uniform clusters in real-world applications, such as those that use the OpenCV library. The grid is defined by two factors known as rho resolution and theta resolution. They frequently correspond to 1 pixel and 1 degree, respectively. We count how many votes there are for each line by sweeping through all cells in the grid. We assert that the straight line has been found if there are more votes than a set threshold, and it is described by and parameters from the considered cluster.

C. How Likely will my pipeline work for other similar videos?

I am confident that my pipeline will work for most of the similar videos, we have to only tweek the region of interest (points for homography) based on the position of camera in the car and resolution of the camera. And tweek the values for white and yellow detection.

My pipeline might not work when the lane folks into two during any turns or other such scenarios.

Output Videos are at:
https://drive.google.com/drive/folders/1eHFouHVX53qSM4wy3kKEu7WZ3T-3UJaP?usp=sharing