Rohan
Abhinav
Kamakshi

ENPM 673
July 31, 2019

Project 2

Lane Detection

The video we are provided has different lighting at various parts during the video. Therefore, we were not able to directly apply mask to the video. We first created a function adjust gamma to brightness adjustment and then converted the image from RGB to L*A*B, which is under the function preprocessing.

Homography

Using the video we have manually selected 4 points src, which is source point and dst has been chosen randomly to form a rectangle which includes the video without the sky. Now, to find Homography, we know that:

$$A * h = 0_{8x1}$$

$$\begin{bmatrix} x_1^(w) & y_1^(w) & 1 & 0 & 0 & 0 & x_1^(c)x_1^(w) & -x_1^(c)x_1^(w) & -x_1^(c) \\ 0 & 0 & 0 & x_1^(w) & y_1^(w) & 1 & -y_1^(c)x_1^(w) & -y_1^(c)y_1^(w) & y_1^(c) \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ x_4^(w) & y_4^(w) & 1 & 0 & 0 & 0 & x_4^(c)x_4^(w) & -x_4^(c)x_4^(w) & -x_4^(c) \\ 0 & 0 & 0 & x_4^(w) & y_4^(w) & 1 & -y_4^(c)x_4^(w) & -y_4^(c)y_4^(w) & y_4^(c) \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (1)$$

Here, A is not a square matrix with size 8×9. We use singular value decomposition (SVD) to find the smallest eigen vector of A. Accordingly, A=UDV^T

The elements of homography, H, is the last column of the V matrix in the above formula. Thus,

$$h = \frac{[v_{19}, ..., v_{99}]^T}{v_{99}}$$

The H found now is reshaped into a 3x3 matrix. The inverse of this square matrix is then computed and normalised.
After finding homography from source coordinates to warped coordinates, we perform warping by multiplying the source coordinates to the obtained homography matrix. The image so far is not GRAY scaled and thus before apply CV2.threshold to convert all the lane marking

to white, irrespective of colour, we will convert the image to gray scale. This converts the image to binary image.

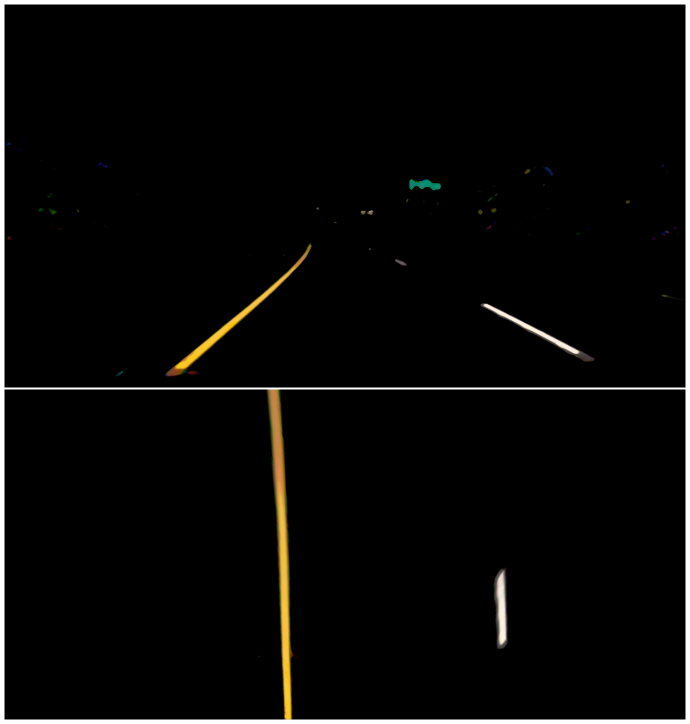
Figure 1: Original Masked Video and its Warped Image

Hough Line

In our code we created a histogram of Lane Pixels to detect lanes and not the Hough line method. The Hough Line Transformation is used to detect straight lines whereas the other method worked better in detecting lanes during turns and thus made the code more generalized.

In general for including Hough lines we need to first detect edges using canny or sobel edge detection method after the algorithm defined above. The Hough line transformation plots a family of lines that goes through a point (r,θ), such that $r > 0$ and $0 < \theta < 2\pi$. What we get is a sinusoid. On doing so for all the points in the image, if the curves of two different points intersect in the plane θ - r, that means that both points belong to a same line. Figure 2

The three plots intersect in one single point, these coordinates are the parameters (θ,r) or the line on which $(x_0, y_0), (x_1, y_1)$ and (x_2, y_2) lay.
The general equation for line in polar co-ordinate plane is :

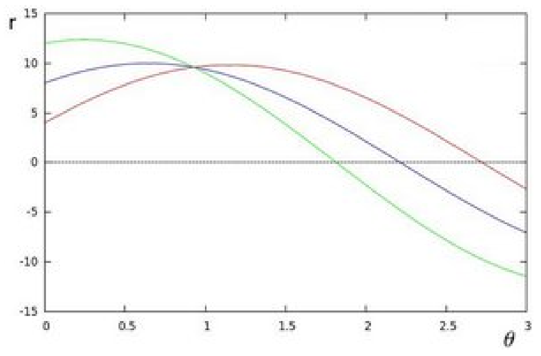

Figure 2:

$$y = \frac{cos(\theta)}{sin(\theta)}x + \frac{r}{sin(\theta)}$$

$$r = x * cos(\theta) + y * sin(\theta)$$

To detect a line we find the number of intersections between curves. More curves intersecting means that the line represented by that intersection have more points. Therefore, in Hough line we find the intersection between curves of every point in the image. Further, we define a threshold of the minimum number of intersections needed to detect a line. If the number of intersections is above the set threshold, then we get a line with the parameters (θ,r).

Finally,the number of lines detected vary as we change the threshold. For a higher threshold, fewer lines will be detected.

Histogram of Lane Pixel

We now have a thresholded warped image. Now to detect lanes we have used peaks in the histogram generated by the lane pixels.

After preprocessing and applying threshold, and perspective transform to a road image, we have a binary image where the lane lines are standing out. We now decide which pixels are part of the lines and which belong to the left line and which belong to the right line.

In my thresholded binary image, pixels are either 0 or 1, so the two most prominent peaks in the histogram were good indicators of the x-position of the lane lines. From here we have used a sliding window, to find the lines.The Sliding Window is a rectangular region of fixed width and height that slides across an image. For each of these windows, we have applied an image classifier to determine if the window has a nonzero pixel or not. Since we have a binary image if the pixel is nonzero that means its white and we append it to a queue for either of the half (left or the right halves).The stepSize for us minpixel indicates how many pixels we are going to skip in both the (x, y) direction.We then concatenate the above queues for each window to finally form the left and the right lanes.Figure 4

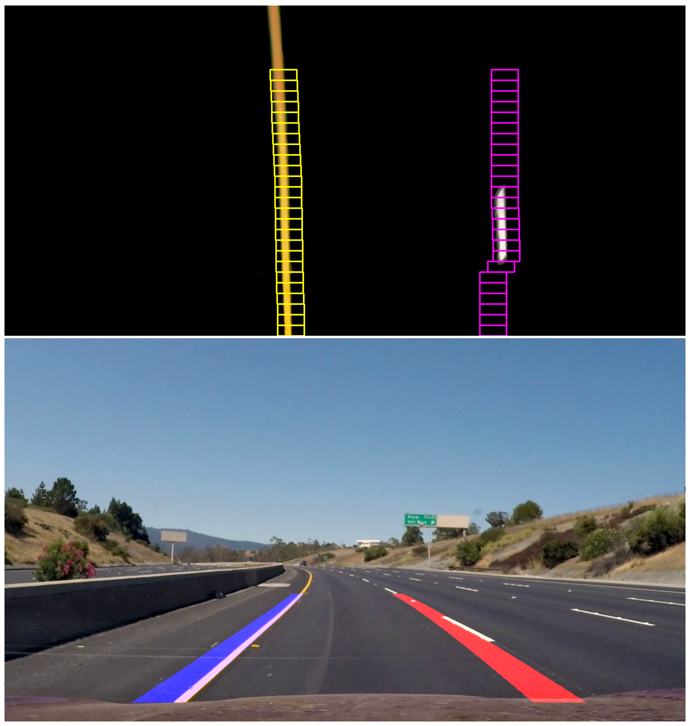

Figure 3: Lane Detection for Project Video

1 Generalization

The pipeline we have set up is generated using individual channels of the image frame. These threshold values are not specific to color and luminosity. The only variable will be the camera calibration matrix and the distortion coefficients, rest everything can be kept same for detecting lanes in a video sequence.

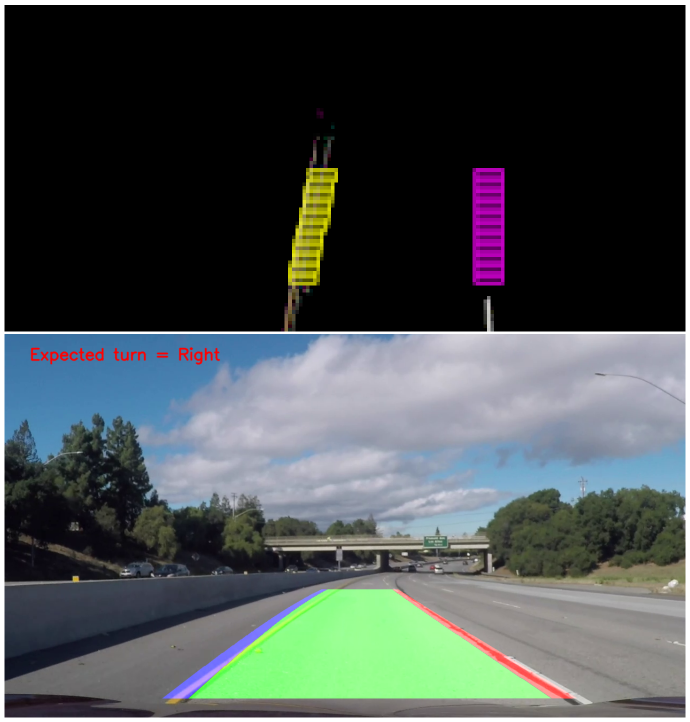

Figure 4: Lane Detection Challenge Video

Problems Encountered

1. In the Challenge video we faced problem in detecting the lane due to reduced brightness.
2. In either of the videos we faced difficulties in thresholding for desired masks in individual channels of LAB and HSL colour spaces.
3. The change in lighting of the scene was a hard problem to deal with.
4. Sliding window technique has lot of parameters to tune which was a lot of hit and trial.
5. Due to the noise in lane pixel detection and curve fitting, the radius of curvature fluctuates a little between negative and positive values.

NOTE

To run the code follow the README file.