Lane Detection for Self-Driving Car Project Report

Rishabh Biyani
Saimouli Katragadda

April 3, 2017

List of Figures

1 Pipeline

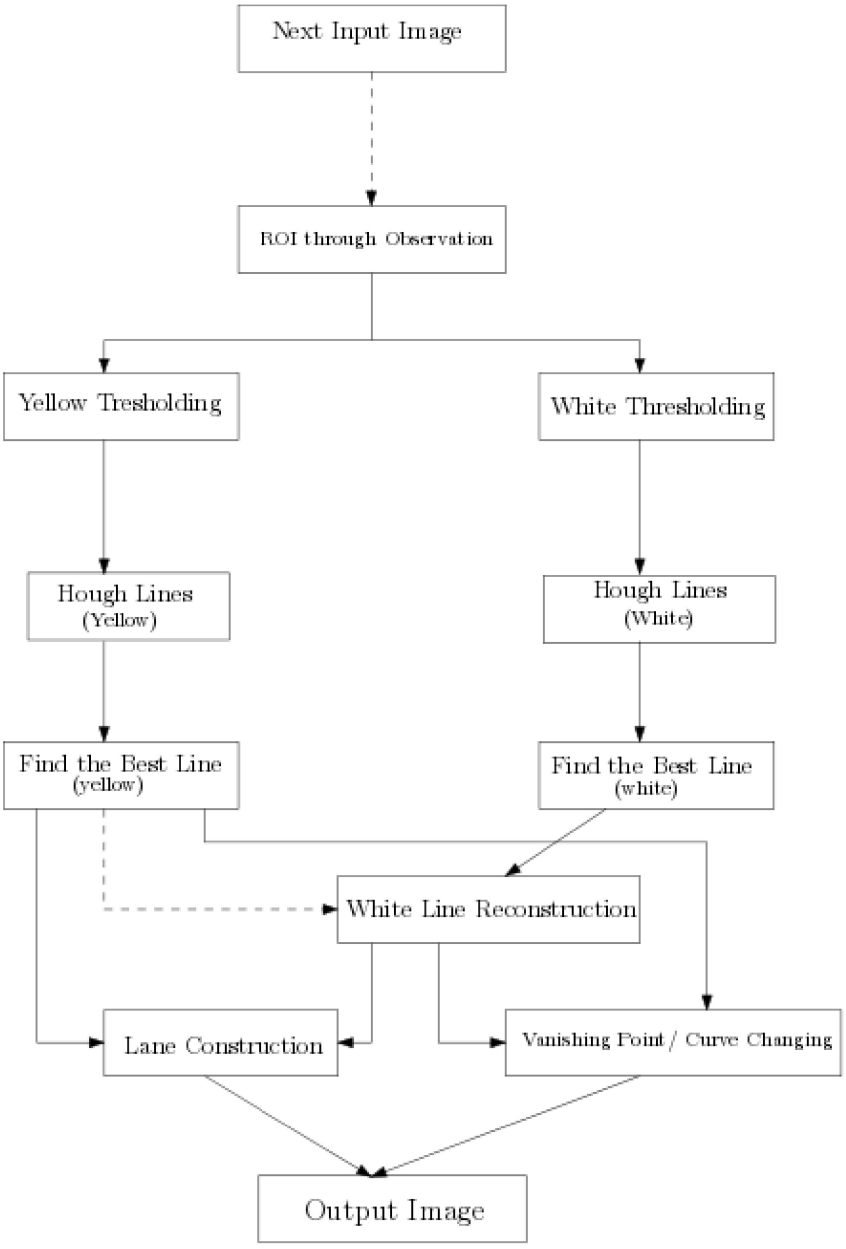
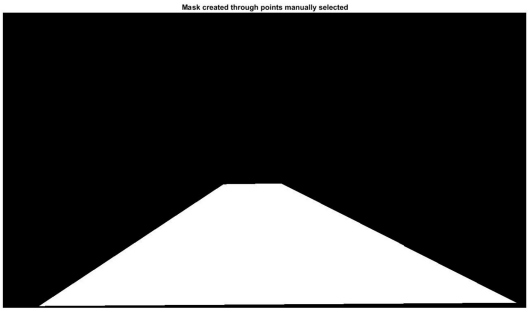


Figure 1: Pipeline for Straight Lane Detection

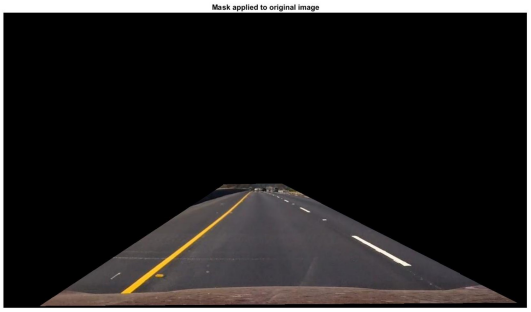
Figure 1 summarizes the pipeline in brief. The current implementation will focus on fitting the lines on the lane with just the **straight lines**. Each of these steps is further illustrated in the following sections.

1.1 Region of Interest(ROI)

The first step is to extract the region of interest from the image. For this, points are manually selected which are big enough to capture the lane spectrum of the entire video. Thus, using



(a) Lane masking through observation



(b) Mask applied on RGB Image

Figure 2: Figure illustrating region of interest extraction through **poly2mask**

(a) Yellow(left) Lane thresholding

(b) White(right) Lane tresholding

Figure 3: Figure illustrating the thresholding outputs on yellow and white lanes

poly2mask the mask for the region pertaining to lane is obtained. The main motivation to perform this operation is to reject the white noise coming from the lanes next to the current lane being tracked. Outputs for these are shown in the figure 2

1.2 Thresholding

Now output from previous step is separately processed for yellow and white line. The series of next steps look like **Denoising->Thresholding->Hough lines->Best Line**. The line corresponding to the lane will be selected by choosing the best end-point pixels in the white-lane and yellow-lane mask respectively(more on this in a later section). Because, these pixels shouldn't interfere with each other we choose to do the thresholding for yellow lane and white lane on two separate instances of the original image. In other words, Denoising->Thresholding->Hough lines->Best Line is performed separately for yellow and white lane on two separate instances of original image. Denoising and Thresholding is illustrated using figure 3. Denoising is done using median filter. The thresholding is done in HSV space. The value for yellow and white is queried across all three planes.

1.3 Hough Lines

Canny Edge Detector is applied on the mask obtained from the previous step and Hough Lines are extracted. As the output from thresholding was fairly clean, need for any morphological cleaning was not considered a requirement. Besides, by setting proper input parameter tuning to Hough Lines like minlength and fillgap tolerance, we can get rid of otherwise spurious noise. Note again that, Canny and hough Lines are being performed separately for white lane and yellow lane.

1.4 Best Fit Line

Best Fit Line is obtained by a simple algorithm. Amongst the pixels forming the lane, farthest pixel is taken as one end-point and the closest(point at the bottom of the image frame) is taken as the other end-point. Connecting these two points will give us the required line.

The line obtained out of yellow lane mask was consistent across all the frames. However, white lane mask as observed in few of the frames is shaky and not all the pixels are recovered. This is shown in the figure 4a. This is mostly because white lane is not perfectly visible in all the frames. The impact of this is that we cannot rely only on the best-fit line from the white lane mask to construct the actual lane - as it is fairly short. Moreover, different lengths on yellow and white lane doesn't look convincing. So, the white line is reconstructed by taking support of the yellow lane. This is discussed in the next section.

1.5 White Line Reconstruction

Reconstruction is based of finding the correct end-points for the white line. It uses the direction from its best-fit line to extrapolate and connect the end-points. The end points are obtained by connecting lines at slope zero from the end-points of the yellow line and finding the intersection with the direction of the best-fit white line. This is illustrated in the figure 4b and 4c. The equation of line in the image is written in polar co-ordinates as-

$$\rho = x\cos(\theta) + y\sin(\theta) \tag{1}$$

To, obtain parameters of the equation 1, we look at the equation of line of the form $ax + by + c = 0$ in the projective space. we use following equations -

$$\begin{aligned} \cos(\theta) &= \frac{a}{\sqrt{a^2+b^2}} \\ \sin(\theta) &= \frac{b}{\sqrt{a^2+b^2}} \\ \rho &= -\frac{c}{\sqrt{a^2+b^2}} \end{aligned} \tag{2}$$

To find a line passing through two points we express those co-ordinates as homogeneous co-ordinates and take a cross-product. This is achieved in MATLAB using the **cross** function. After obtaining the cross product, we normalize it according to the equation 2 to get the equation of line. Thus, obtaining the equation of line in polar co-ordinates.

(a) Best Fit Line for White lane is very short

(b) White Lane Reconstruction using support from yellow lane

(c) White Lane reconstructed

Figure 4: White Lane Reconstruction process for consistency in lengths when compared to the yellow lane

To find the intersection of two lines we perform the same cross product using $[a \quad b \quad c]$ as the co-efficients of the vector. This follows from the fact that we can express the equation of line as -

$$\begin{pmatrix} a & b & c \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0 \tag{3}$$

with a,b,c as the components of the vector. Again, after obtaining the cross product we normalize using the third component to get the co-ordinates of the point. The first two co-ordinates are the point in the image plane. Thus, because we know the end-points of the line we can find the equation of line and hence its direction. Similarly, we can find intersection of line at slope zero(or $\theta = 0$) with the white lane direction using the method just described.

1.6 Lane-Changing System

For designing the lane -changing system. We use the vanishing point concept. Again, the point is found by finding the intersection between yellow-lane direction and white lane direction using equations of line in projective space as described in the last section. We track the position of this point. If this point moves further towards right from the centre of the image that means we have to veer towards the right and vice versa for left.

Finally, to get a representation of lane we insert a green screen within the area between two lanes as shown in the figure 5

Figure 5: Final Output

1.7 Further Improvements

There are aspects in this project which can be certainly improved upon and were overlooked at this point of time due to time constraints. As mentioned, yellow lane direction is fairly stable. In few of the frames, white lane direction does not give a true representation of the lane.This can be improved my implementing a better algorithm for obtaining the best-fit line for the white lane. Also, yellow lane length can be constrained to be of fixed length or averaged out for five frames to get a more stable lane detection. Further more, a **polyfit** type functionality can be used to detect curved lanes. This approach will require the use of homographies