Faculty of Engineering
Department of Mechanical & Mechatronics Engineering

ECE 493

Autonomous Vehicles

Assignment 1
Lane Detection Using Classical Computer Vision Methods

A report prepared for
Professor Krzysztof Czarnecki

Prepared by
Dhruv Sharma -

February 4, 2017

Table of Contents

List of Figures

1.0 Introduction

This assignment involves applying classical computer vision algorithms to detect lanes. Three test videos and six test images are provided to test the algorithm.

2.0 Solution

The problem of lane detection is broken down into two parts –

1. Edge detection to detect lane markings
2. Combining detected edges to determine left and right lane boundaries

2.1 Edge Detection

The edge detection pipeline is shown in Figure 1.

Figure 1: Edge detection pipeline

The pipeline is demonstrated for the following image shown in Figure 2

Figure 2: Original image

2.1.1 Color Filtering

The image is filtered to extract the white and yellow colors that correspond to the lane markings. Mostly lanes are white or yellow and for the given test videos, these colors represent the lane lines well enough.

The color filtering is applied as follows –

- RGB image is converted into a grayscale version and an HSV version.
- The HSV image is used to generate an image mask that depicts the yellow parts of the image, using the values in following ranges:
    - Hue = (20, 30)
    - Saturation = (100, 255)
    - Value = (100, 255)
- The grayscale image is used to generate an image mask that depicts the white areas in the image, with pixel in range (200, 255).
- The yellow and the white mask are combined by doing a bitwise or of the two masks
- The combined mask is applied on the grayscale image to obtain the yellow and white parts of the image

This results in the image as seen in Figure 3

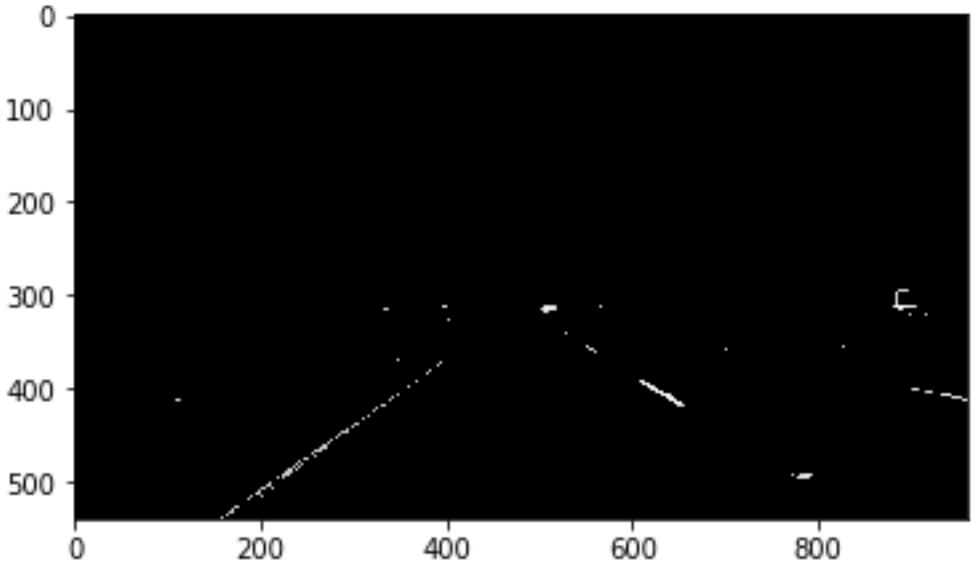

Figure 3: Image after color segmentation

2.1.2 Canny Edge Detector

Canny Edge detector is applied to the color filtered image. The canny edge detector hysteresis thresholds are decided based on experimentation. A minimum threshold of 120 and a maximum threshold of 200 is chosen. Figure 4 shows the output of the Canny edge detector.

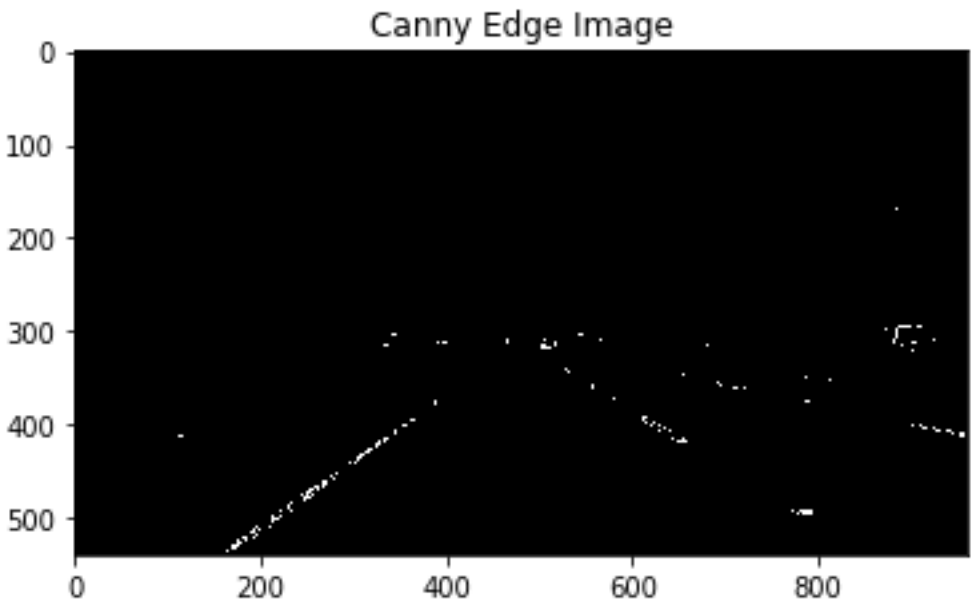

Figure 4: Canny edge detector output

The edges in the image are identified by the Canny Edge detector. Next, step we smooth the output from the Canny edge detector using a Gaussian blur. This helps to get rid of noise in the Canny outuput.

2.1.3 Gaussian Blur

Gaussian blur is applied with a 5×5 kernel is applied to the image from Figure 4. Figure 5 shows the output of this filter.

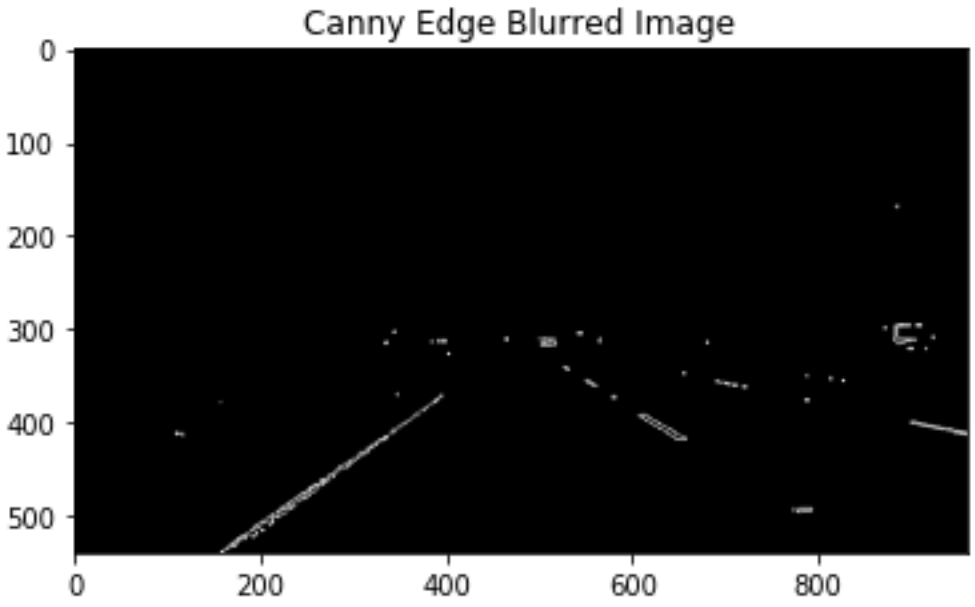

Figure 5: Canny edge detector output blurred

The filtered image can be seen to enhance the lane markings detected by the Canny edge detector.

2.1.4 ROI Cropping

A region of interest is selected that contains the ego lane lines and part of the image that has roads. In short, sky and other things not of interest to detect lane lines are not included in the region of interest.

Trapezoidal region of interest is determined with the base at the bottom of the image. The coordinates of the trapezoid are as follows –

$$A\ (bottom\ left) = (width * 0.1, height)$$

$$B\ (bottom\ right) = (width * 0.9, height)$$

$$C(top\ right) = \Big(\big(height * 0.65 - B_y\big)\tan(56.15°) + B_x, height * 0.65\Big)$$

$$D\ (top\ left) = (-\big(height * 0.65 - A_y\big)\tan(56.15°) + A_x, height * 0.65$$

Height of the ROI is reduced to 35% of the image height. The trapezoid base angle is chosen as 56.15 through visual inspection and works for all the test videos and most of camera configurations.

Figure 6 shows the ROI cropped image.

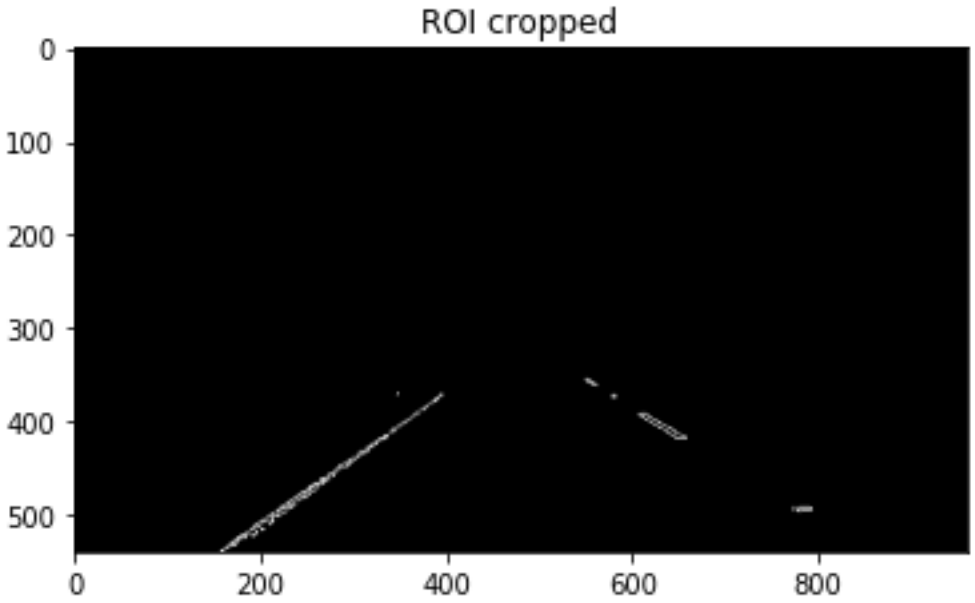

Figure 6: Cropped region of interest of the input image

2.1.5 Probabilistic Hough Transform

Probabilistic Hough transform is used to find lines in the image. The output from ROI cropped image from Canny Edge detector is fed into the HoughLinesP function in opencv. This function returns a list of end points of the lines detected in the image.

Hough transform is run with following parameters –

$$\rho\ accuracy = 2$$

$$\theta\ accuracy = 1\ rad$$

$$Vote\ threshold = 10$$

$$Minimum\ Line\ Length = 5$$

$$Max\ Line\ Length = 10$$

The ρ accuracy is chosen to be a reasonable number of 2. θ accuracy of 1 rad is unusually high, however it returned in the best result for lane detection. The vote threshold was chosen to be 10 to pick edges with high probability of lines. Minimum line length of 5 and max line length of 10 is selected through experimentation.

Figure 7 shows the Hough transform output on the given image.

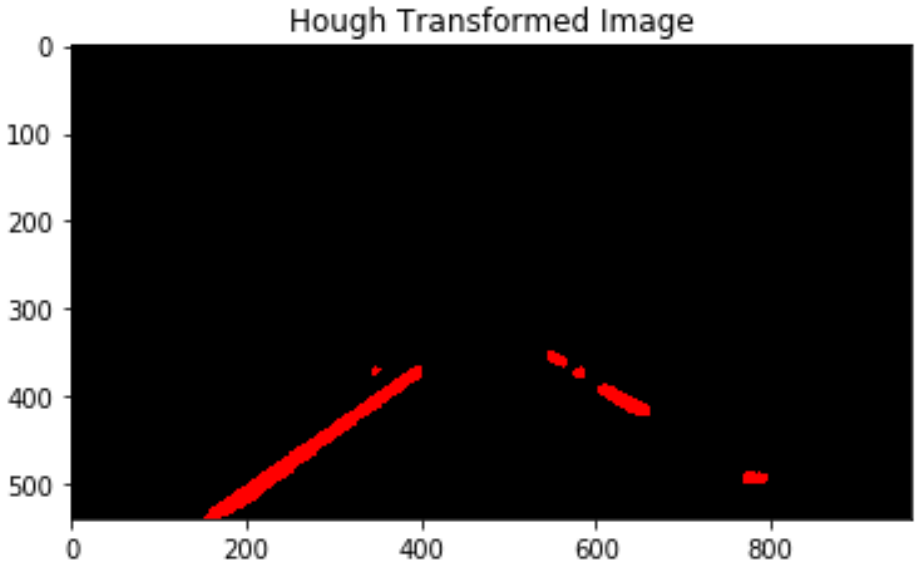

Figure 7: Output of the probabilistic Hough transform for the input image

2.2 Lane Boundary Markings

The lane boundary marking pipeline consists of a slope filter and a lane segmenter. The output from the edge detection pipeline is fed into the slope filter whose output feeds into the segmenter. Figure 8 shows the lane boundary marking pipeline.

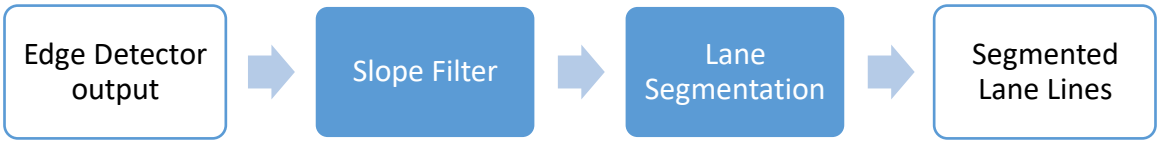

Figure 8: Lane boundary marking pipeline

In order to tag the left and right lane, the edge detector output is fed through a slope filter. The slope filter is explained in section 2.2.1.

2.2.1 Slope Filter

This filter uses the lines returned by the Hough transform as the input. The filter pipeline is as shown in Figure 9

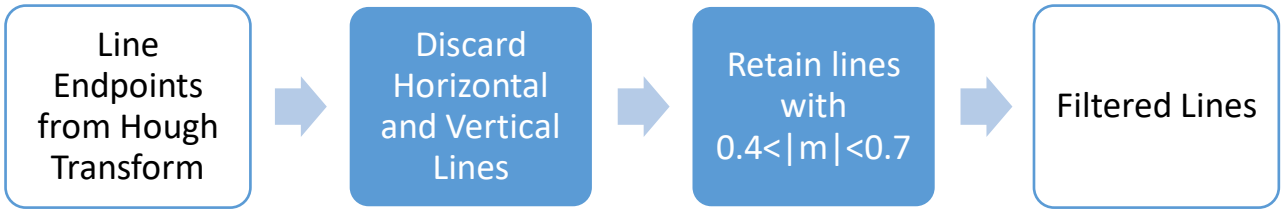

Figure 9: Slope filter pipeline

The horizontal and vertical Hough lines are discarded from consideration since from the car point of view, there is a very low probability for lane lines in the image to appear vertical. Similarly, it can be said with high certainty that lane lines cannot be horizontal. Empirically the lane lines have an absolute slope value between 0.4 and 0.7. All the other lines are discarded to filter out all the noise. This returns a list of filtered lines that are used to segment the left and right lanes.

2.2.2 Lane Segmentation

Once the lines are filtered from the slope filter, the lane segmentation is done based on the location of the points. If the both x coordinates of end points of the line lie to the left of the middle x of the image, then the line belongs to the left lane. On the other hand, if the both x coordinates of the end points of the line lie to the right of the middle x, then the line belongs to the right lane. If these conditions don't apply, the line is not classified into either. Figure 10 depicts the process.

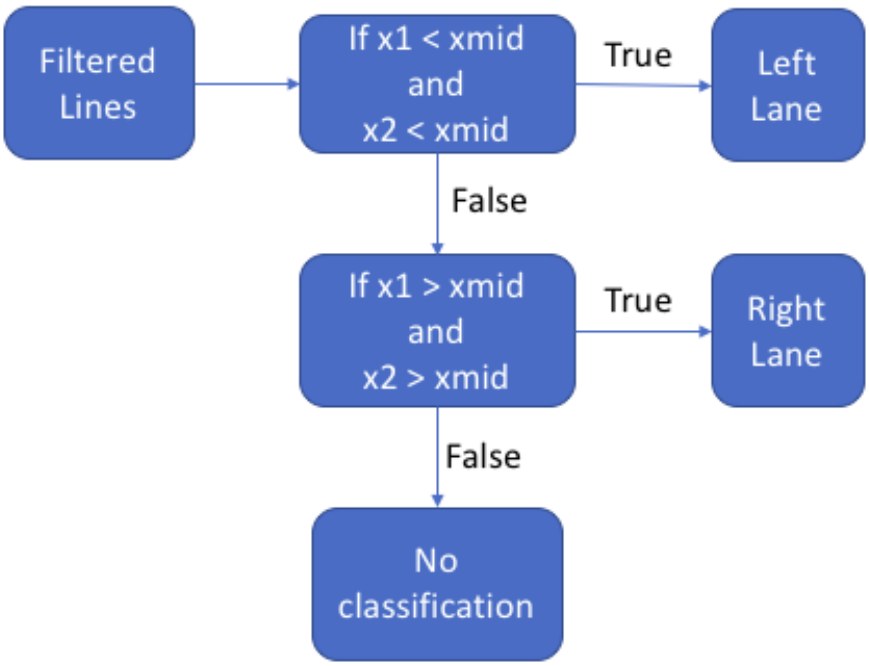

Figure 10: Lane segmentation pipeline

Once the segmentation is done, two lists of coordinates corresponding to left and right lane are obtained. A line is fit using the numpy polyfit function to each of these coordinate lists. The line parameters obtained are used to extrapolate and draw lane lines from the bottom of the image to 40% of the height of the image. Figure 11 shows the extrapolated lines drawn on the original image.

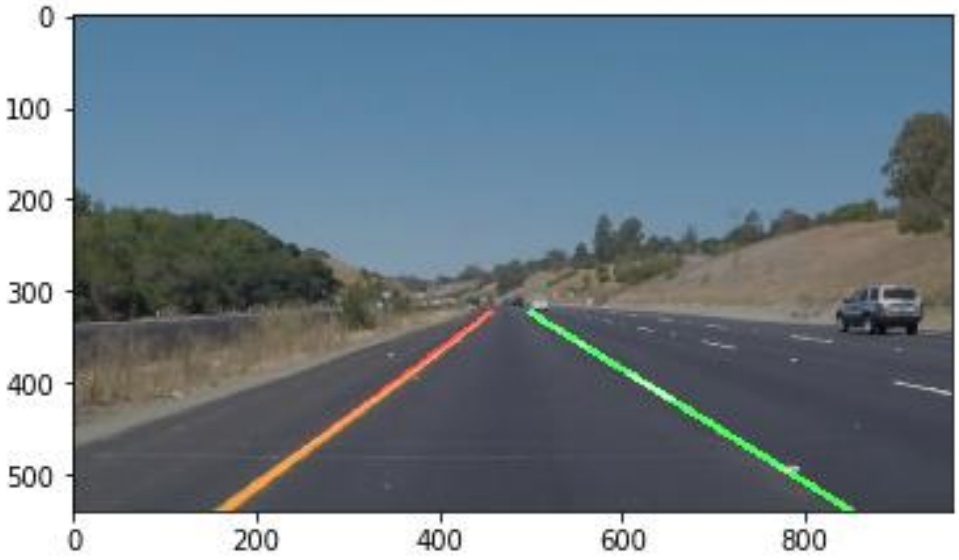

Figure 11: Extrapolated lane lines overlaid on the original image

3.0 Shortcomings

The lane detection approach used in this assignment is very basic and lacks robustness. Clearly, this approach does not work reliably in all cases and has numerous shortcomings.

Firstly, this approach depends on color segmentation. White and yellow are extracted from the image. Although, this works well for the test videos provided, it will not always work in the real world. Lane lines can be orange in construction zones. Additionally, color segmentation will not work reliably in poor lighting conditions and has a very low probability of working during the night.

ROI determination is done through a hardcoded rule that assumes that the camera is mounted in the center of the car, perfectly flat without any roll or pitch. ROI determination should be done using the calibration parameters for the respective mounting of the camera.

This approach is not suitable for curvy roads like the one shown in Figure 12. Firstly the slope filter will fail since it will discard some edge lines which are outside the acceptable slope range. In addition, the lane segmentation done based only the x coordinate value is not applicable to such images since left lane can clearly be seen to be occupy parts of the right of the image as well.

Figure 12: Curvy road

4.0 Further Improvements

Numerous improvements are possible to improve the functioning of the lane detection. Tracking can be used to predict the location of the lane lines based on prior information. This ensures any poor measurements are filtered out and gives a good estimate of the lane lines. This also helps in case of poor lane markings.

Random Sample Consensus (RANSAC) can be used to help eliminate outliers due to noise and artifacts in the road, and a Kalman filter can be used to help smooth the output of the lane tracker [1].

Other approach that can be used is a deep neural network to detect the lane lines. Using a neural network approach does not require any hand engineering of feature. All the features are learnt from the data that is used to train the network. Given vast variety of data, this approach can give very good result. The limitation however is the requirement of vast amounts of labelled data, which is hard to expensive to attain.

References

[1] Amol Borkar, Monson Hayes, and Mark T. Smith. “Robust lane detection and tracking with ransac and Kalman filter” in *Image Processing (ICIP)* 2009.