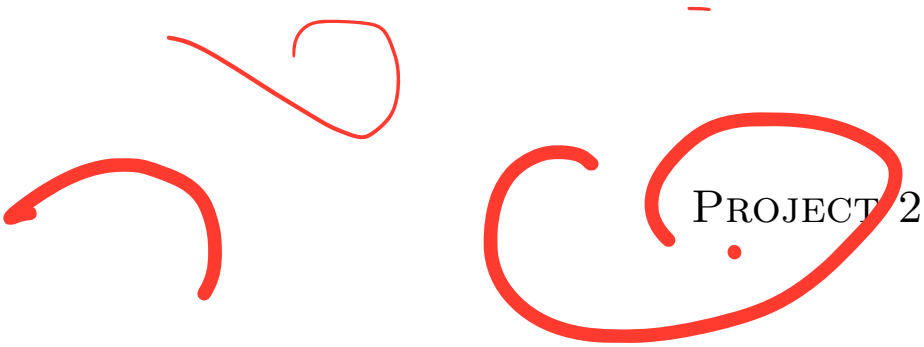
Perception For Autonomous Robots



Project 2

Lane Detection and Turn Prediction

Pranali Desai (116182935) Varsha Eranki (116204569)

Contents

List of Figures

1 Introduction

The project was centered around the application of Lane Detection on self driving cars to mimic Lane Departure warning system. This project was programmed using Python by firstly developing an algorithm that works on an different image frames of the videos provided and later applied on the video sequence. The steps of the algorithm are stated below:

1. The different image frames were taken from the video to test and understand the parameters involved for image processing.
2. The first step involves camera calibration of the frame, which was already provided in the question, followed by undistorting to remove the barrel like effect.
3. To differ the lanes (using white pixels of the lanes), we applied an edge detector followed by Warping which performed Homography giving a Perspective transform of the frame.
4. After performing Homography, binary thresholding was done to get an image in 0 and 255 intensity, which was then input to the Histogram to give a plot of the white colour inferring to the lane boundaries.
5. After obtaining the histogram, we applied the concept of sliding window to detect and extract the left and right lane boundaries.
6. The sliding window now stacks the lane parameters and predicts the next lane parameters using *ployfit* function of *numpy* library.
7. The next step involved turn prediction.
8. This code was tested on various image frames and later applied to the *challenge* video and *project* video.

2 Preparing the Input

As the project is coded in Python, the primary input will be importing the necessary libraries which mainly include *opencv*, *numpy*, *math*, *scipy* and *matplotlib*, also, defining the Camera Calibration matrix. Then the following procedure was followed in preparing the input:

1. The raw frame has to be processed before performing Homography or extracting frame arrays, so firstly using the camera calibration matrix provided, we undistorted the image and applied Gaussian Blur for noise removal followed by Homography and Warping to get the Perspective transform.
2. After this, binary thresholding was done to identify 0 and 255(high and low intensity) and we applied the Sobel operator for edge detection as Canny Edge detector was giving very fine edges which would be problematic for plotting Histogram.

3. After the Sobel operator, Histogram function was implemented and high intensity pixel values which depict the lane boundaries were plotted for the left and right lane boundaries which formed our expected output.

3 Sobel and Histogram

Sobel operator calculates the gradient of the pixel intensities of an image to output the edges being detected. Thus it is an discrete differentiation operator that combines diffrentiation and Gaussian smoothing. Canny Edge detector is another edge detector that uses a low pass filter and low level image processing is implemented. It is based on a small area around the pixel for which the edge value is computed giving out fairly weak lines. Hence, Sobel operator was implemented which produced well defined thicker lines after thresolding as well.

When the Sobel operator was applied on the thresholded frame, it clearly defined the high and low intensity pixels of the right and left lane boundaries shown in figure below.

The Histogram now gives an overall distribution of the frame intensities. It collects the data of the pixel intensity values which will help us in identifying the lane boundaries of the road.

Two peaked graph plots inferring to the high intensity white pixels of the left and right lanes are obtained as the output.

4 Application of Sliding Window

Sliding window has its wide applications in Computer vision, where a rectangular window with fixed dimensions slides over the image to recognize and detect objects at various locations and scales in the image.

1. After the Histogram plot described the distribution of white pixels in *y-direction* from the binary thresholded image, where a two-peaked histogram plot was generated. Each peak denoted the distribution of these white pixels along the *y-direction* of left and right lanes.
2. Then high intensity pixel values are extracted along *y-direction* and intensities of zero pixel arrays are eliminated. These extracted indices are then stored in the respective lists of left and right lanes.
3. The set of both lane boundaries are then plotted and fit to two lane boundaries on either side using *polyfit* that fits it to a second order polynomial in the video frames.

5 Poly fitting the Lane Lines

A curve is fitted along the *y-direction* from the extracted indices of left and right lane boundary lists. This helps us to steer the car and detect the white pixel directions along the that direction.

6 Turn Prediction

There are two ways to predict turns:

1. Computing the radius of curvature and taking a turn towards the lesser radius of curvature.
2. By computing the difference of the midvalues of the lane and car and comparing it with the predefined offset.

We implemented the later approach where we defined a certain offset value which ranges from -0.25 to +0.25.

1. If the difference of the midvalues of the lane and car is between the offsets, then there is a *no turn* prediction.
2. If the difference of the midvalues of the lane and car is more than the the positive offset, then there is a *right turn* prediction.
3. If the difference of the midvalues of the lane and car is less than the the negative offset, then there is a *left turn* prediction.

7 Display on the Original frames

1. The same concept of getting the warped image of the parallel lanes has been used.
2. The inverse homography matrix has been used in the warpperspective function to place back the detected lanes.
3. Also the turn predicted has been printed on the image by puttext function of opencv.

8 Results

8.1 Output for Project Video

Figure 1: Project video - Warped

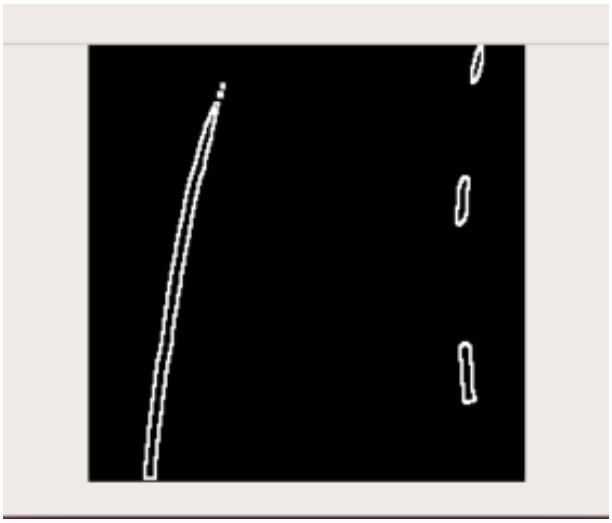


Figure 2: Project video - Sobel

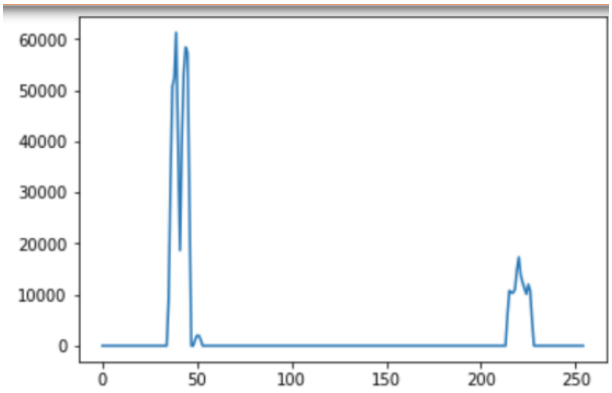


Figure 3: Project video - Histogram

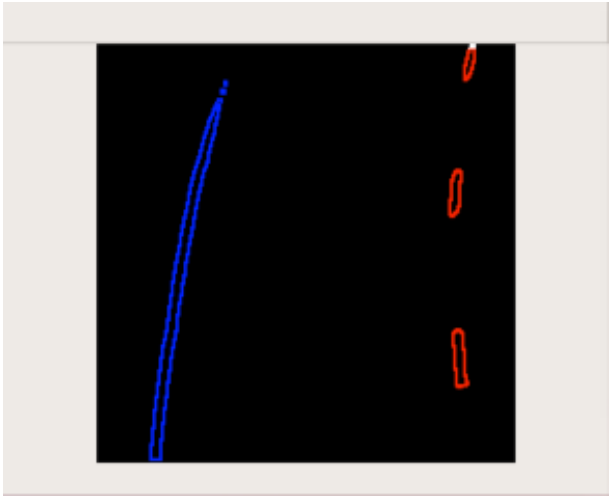


Figure 4: Project video - Sliding

Figure 5: Project video - Light pa

Figure 6: Project video - No Turn

Figure 7: Project video - Turning Right

Figure 8: Project video - Turning Left

8.2 Output for Challenge Video

Figure 9: Challenge video - Warped Image

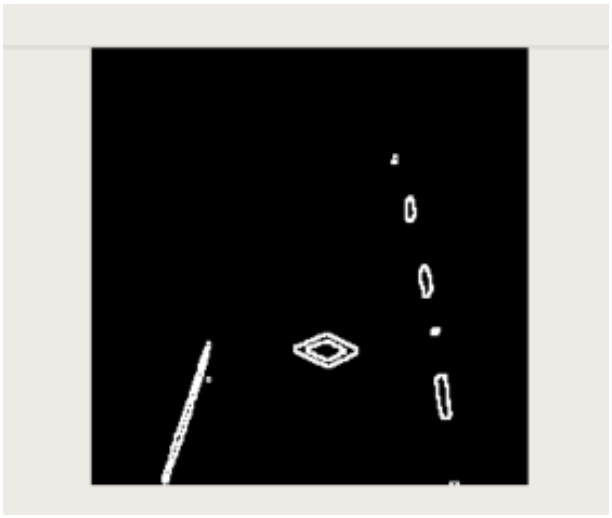


Figure 10: Challenge video - Sobel

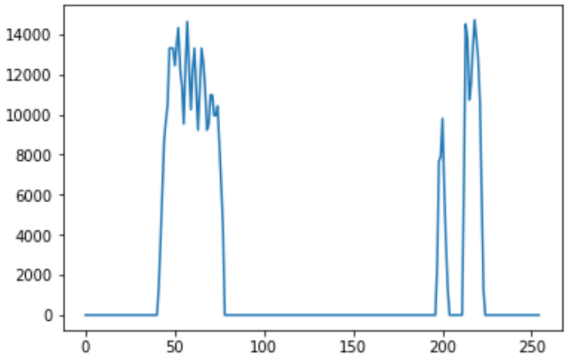


Figure 11: Challenge video - Histogram

Figure 12: Challenge video - Fitted lines

Figure 13: Challenge video - Turning Right

Figure 14: Challenge video - No Turn

Figure 15: Challenge video - Under the Bridge

9 Discussions

There were some challenges faced during the project, they are discussed below:

1. Homogrphy function from earlier project has been used to get the birds eye viewed image from the video. This helps us further to look at the image and detect lanes using the histogram.
2. Changing intensities in the video frames was slightly difficult to overcome.
   (a) Change of lightness value when the car goes under the bridge was a challenging but was tackled significantly by implementing γ correction.
   (b) Though the problem was not completed correct but significant result were achieved.
3. To get the lane lines using RANSAC was problematic as the video was running slower and hence, sliding window concept was implemented.
4. Hough lines concept did not work well for curved roads, hence we implemented Histogram lane of pixels.
5. The code can is generalized due to the fact that in both the challenge and project video red channel was used to extract the lane lines. Also the homography points were not changed and it still gave good

results for both the videos. It can be even more generalized after applying γ correction with better values.

6. When the car changes lanes the video frames will be not able to detect lanes and thus not produce good result i.e during the transition. Though once it is completely in the next lane the video wont show any problem and run in the noraml way.

10 References

1. https://www.pyimagesearch.com/2015/10/05/opencv-gamma-correction/
2. https://www.youtube.com/watch?v=VyLihutdsPkt=804s
3. https://www.youtube.com/watch?v=eLTLtUVuuy4t=4311s
4. https://towardsdatascience.com/teaching-cars-to-see-advanced-lane-detection-using-computer-vision-87a01de0424f
5. https://github.com/jeremy-shannon/CarND-Advanced-Lane-Lines