Perception for Autonomous Robots

Project 2

# Lane Detection and Turn Prediction

Raghav Nandwani (116321549) - Sanket Goyal (116155144)

# Contents

# List of Figures

# 1 Introduction

Self Driving cars require Lane Detection Algorithms to stay in their respective lanes to avoid collisions or abrupt lane changes when not required. Lane Detection is one of the most important requirements to be accomplished before a driver-less car can hit the road.

This project focuses on Lane Detection followed by calculating the curvature of the road and whether the car is taking a left turn, a right turn or is going straight.

# 2 Lane detection algorithm

## 2.1 Preparing the input

The video is taken and fragmented into frames, so that all the process happens in a frame and then the frames are played together to be converted into a video file again. The image from the video is distorted at the very beginning to remove either barrel or the fish eye effect. The inputs to perform this function are the image, the camera calibration matrix and the distortion coefficients. After this Gausssian blur is applied to remove noise and smooth the frame.

## 2.2 Sobel Edge Detection

After smoothing the image the homography of road is carried out to obtain birds eye view of the road and the lanes which were converging in the orignal image becomes parallel. The four points for homography are selected in such a way that two points are taken on each side of the primary lane. Once Homography is completed, Warp Perspective is done of the image which takes the homography matrix as the input and displays the top view of the image as the output.

To detect yellow and white colors of the lanes, it is found that the best results are witnessed in the red channel, thus the red channel of the lane is extracted for further processing. Binary thresholding is done to seperate pixels of high intensity i.e. yellow and white pixels from the less intensity pixels of the road. next applying Sobel edge detector.

## 2.3 Histogram application of sliding window to polyfit lines between the line pixels

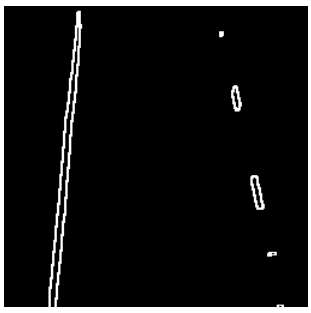

Figure 1: Histogram for Detected Lanes

The above shown output is generated after performing Sobel edge detector. The histogram is generated for the half of the image in such a way that the number of white pixels are represented in y direction for the particular value of x. On the basis of this the two peaks as the result of histogram gives the approximate x location for the starting point of lane in the image. At these two x position a box window is created and all the position of white points in these boxes is stored in two separate lists for two boxes that are generated. After this a rectangle window is moved in y direction based on the height of the window selected and in x direction based on the mean of all the x values in the last window moving vertically upwards.

After we generate two lists of location of points that corresponds to all the pixels of two lines detected. Line fitting is done for the list of points of each list, which provide the left and right line of the lane in which car is moving.

### 2.4 Turn prediction

Turn prediction is done based on the location of pixel for the lines in the world frame in the horizontal direction which are then compared to the center of the car i.e. if the center of the line is aligned with the center of car then no turn is required and if it is offset in right direction then vehicle should turn left and vice versa for the right.

### 2.5 Display on the original frames

Achieving the required lanes, after lane detection, inverse homography is used to add in the generated colored lane space onto the original video, with the turn prediction. This is achieved using fillpoly command, which fills in the space between the two polyfit lines. Thus, the final frames, is an addition of the original frame with the newly generated filled color space describing the lane lines.

## 3 Discussion

The problems and general takeaway from this project were:

1. Homography plays a vital role in lane detection to convert the lanes into a parallel lines rather than converging to a single point. The parallel lines can then be processed easily to yield the final Lane Detection.

2. This pipeline if applied to simillar videos might work perfectly given the constraint that the lane doesnt go into a darker shade. If this is witnessed, this lane detection pipeline tends to not give proper results.

3. If the car changes lanes, it will be observed that the lanes being displayed do get slightly haywire, but then comes back to the correct display of the lanes. More testing on the datasets will give better results.

4. The car going under the bridge in the challenge video changes the brightness of the image frames, which results in change in intensity of the threshold giving varied results.

5. The color texture of the road is not same in both the video given, thus, additional processing is required according to the intensity to get accurate results.

6. Hough lines does yield into accurate lane detection but the curvature for the road creates issues and it is difficult to find it.

7. Gamma Function is one solution to brighten the image, for dark spots like under the bridge or for shadows from nearby trees.

# 4 Result - Screenshot

## 4.1 Project Video Results

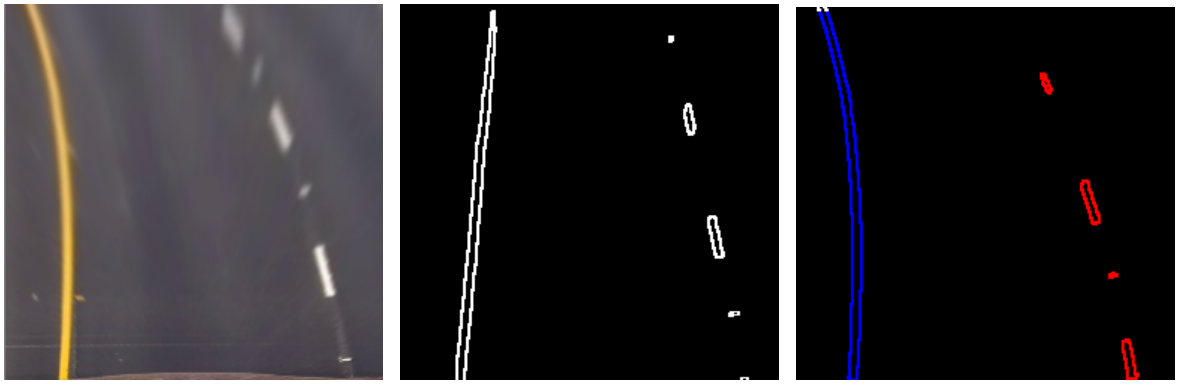

Figure 2: (From Left to Right) Warp-perspective, Sobel operation, Sliding Window Operation

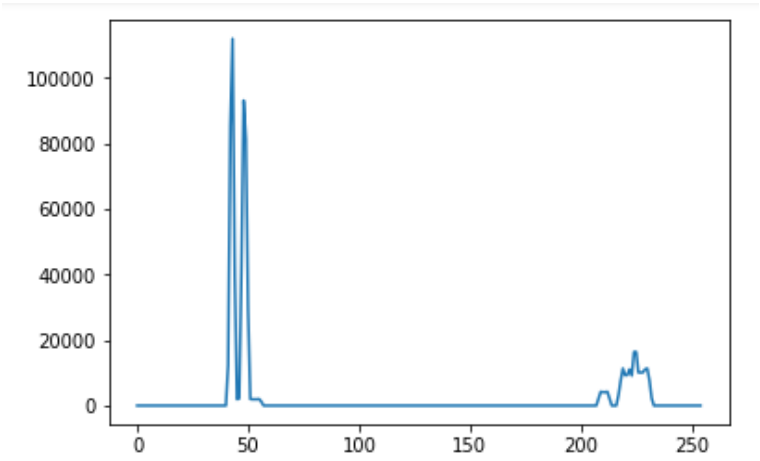


Figure 3: Histogram for Detected Lanes

Figure 4: Lane Detection: Going Straight


Figure 5: Lane Detection: Left Turn


Figure 6: Lane Detection: Right Turn


Figure 7: Lane Detection: On Bridge

## 4.2 Challenge Video Results

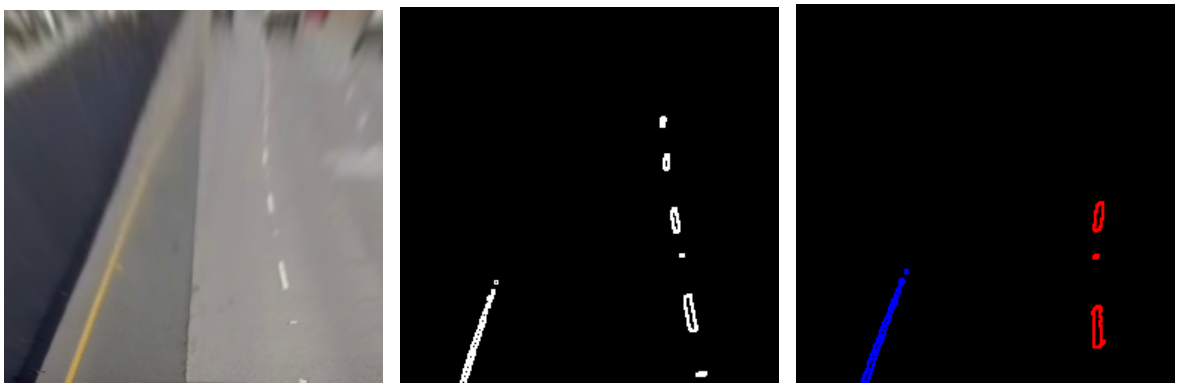

Figure 8: (From Left to Right) Warp-perspective, Sobel operation, Sliding Window Operation

Figure 9: Challenge Video: Going Straight

Figure 10: Challenge Video: Under the bridge

# 5 References


1. https://github.com/thomasantony/CarND-P04-Advanced-Lane-Lines

2. https://www.youtube.com/watch?v=eLTLtUVuuy4t=4311s

3. https://www.youtube.com/watch?v=VyLihutdsPkt=804s

4. https://www.pyimagesearch.com/2015/10/05/opencv-gamma-correction/

5. https://github.com/jeremy-shannon/CarND-Advanced-Lane-Lines