Advanced Lane Detection

Ravina Lad
lad.ra@northeastern.edu

***Abstract*— In the world of autonomous systems to visualize the surroundings, computer vision plays the most important role. This paper explains the process involved in lane detection using different computer vision techniques like camera calibration, perspective transform, binary image thresholding, finding lane pixels, and recognizing the lane area. The end result of this paper is to find the radius of lane curvature. This paper does not include any neural network system. In the project, we will build a pipeline to identify lane boundaries in a video from a front facing camera mounted on a car. It will detect lanes in conditions like curved roads.**

Keywords—camera calibration, radial distortion, gradients, threshold, binary, perspective transform, polynomial fits.

I. Introduction

In the field of Advanced Driver assistive systems, lane departure detection plays an important role. It also helps to improve the vehicle's safe driving. Road accidents are a major problem worldwide. The detection of the lane works as a warning system. Therefore, in order to avoid over speeding of the vehicles during the turning on the curved roads, it is necessary to detect lanes. This paper includes python programming language and for pre-processing the frame from video openCV is utilized. The highlighted pipeline was designed to operate in below scenarios:

- If only one of two lane lines have been successfully detected, then the detection is considered invalid and will be discarded. In this case, the pipeline will instead output a lane line fit (for both left and right) based on the moving average of the previous detections.
- It can detect exactly two lane lines, i.e. the left and right lane boundaries of the lane the vehicle is currently driving in.

Implementation/ approach :

1. Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.

2. Apply a distortion correction to raw images.

3. Use color transforms, gradients, etc., to create a thresholded binary image.

4. Apply a perspective transform to rectify binary image ("birds-eye view").

5. Detect lane pixels and fit to find the lane boundary.

6. Determine the curvature of the lane.

7. Output visual display of the lane boundaries and numerical estimation of lane curvature.

Figure : 1 – Lane Detection

II. Related work

This section talks about the current-state-of-art and different proposed methods given by different scholars.

L. Mao et al [1] and C. Ma. uses a Heuristic search method to detect lane boundaries. They do this by converting RGB image into Lab color space and Heuristic Search technique narrowing the search region, this algorithm is powerful and compatible to the particular road conditions.
J. Baili et al. [2] converts the ROI into YCbCr color space. Horizontal differencing filter is used to detect edges. The modified Hough transform algorithm sorts the edge point into lines. When there is a modification in the road lane, the tracking mode has to be upgraded.
Qing Lin et al. [3] This paper detects lane positions and lane marks. It uses the technique which extracts the lane mark candidates within the input frame through using a prolonged edge link algorithm.It combines lane-mark and edge-link capabilities that are edge-link features that are orientation and width.
K. Dinakaran. Stephen SagayarajS.K. KabileshT. ManiA. AnandkumarGokul Chandrasekaran [4] This paper is quite similar to the approach that I followed but there are many prominent changes such as: they have developed two different models to detect curved and straight lanes. They use improved hough transformation to detect straight lines while curve fitting method is proposed to detect curved road lanes. They also perform sobel thresholding on HLS converted input images.

III. Methods

A. *Camera Calibration*

All cameras use lenses and radial distortion is the main problem with lenses. When lenses focus light rays on the sensor to capture wide images, these light rays tend to blend at the edge of the convex or curved lens of the camera due to the effect of refraction. Furthermore, this effect leads to distorting the images, that is they appear more or less like curved lines than they actually are. To remove radial distortion, this paper uses chessboard images to calculate distortion coefficients and camera matrix. To find the intrinsic camera parameters, first

find the inside corners within the chessboard images and use those parameters to un-distort the test image of straight lanes of road images. Figure : 2 shows the detected corners chessboard images.

We can use five distortion coefficients k1, k2, p1, p2 and k3 to undistort the images from the video frame. I have used (k1, k2, k3) to correct radial distortion. Object points that are (x,y,z) coordinates of the chessboard corners in the world. Chessboard is fixed at (x,y) keeping z zero and I assume that object points will be the same for each calibration image. Img Points will be appended with the (x, y) pixel position of each of the corners in the image plane with each successful chessboard detection.I then used the output obj points and imgpoints to compute the camera calibration and distortion coefficients using the cv2.calibrateCamera() function. I applied this distortion correction to the test image using the cv2.undistort() function. Figure : 3 demonstrates the undistorted image of the road lane and chessboard.

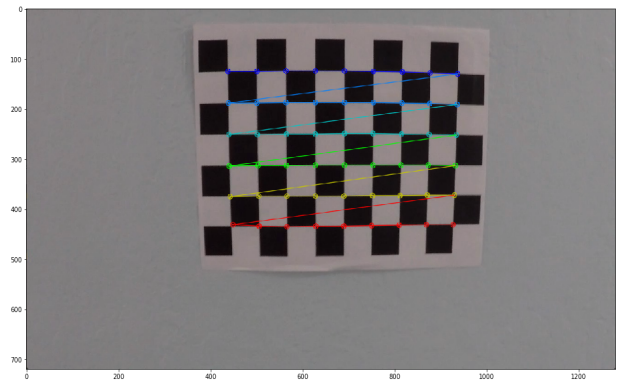

Figure : 2 – Chessboard Corners

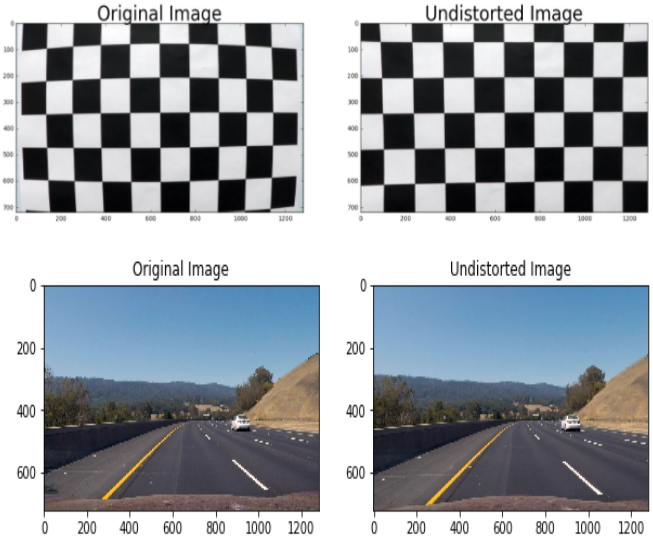

Figure : 3 – Undistorted images of road lanes and chessboard

B. *Perspective Transform*

The first step in detecting lanes is to align our visual systems to visualize the road ahead in a manner that they seem to be looking at it from a bird's eye perspective, this will help in calculating the curvature of the road, as a result, will help us predict the steering angle for the next few hundred meters. The other benefit of top-down view is it fixes the issue where lane lines seem to merge whereas lane lines are infinitely parallel lines as long as the road runs. The bird's eye view can be achieved by applying the perspective transform, essentially mapping a set of four points bounding the lane area in our input image to the desired set of points, thus generating the desired top-down view. These points are determined on a case-to-case basis, the determining factors are mainly the position of the camera in the vehicle and its field of view.

The pictures in Figure 4 represent the input and post-transformation output image respectively. The function cv2.PerspectiveTransform() will change the viewpoint. This transform does not preserve length, angle, and parallelism. This means that the straight lines will remain straight after the transformation. We make use of a function called cv2.warpPerspective() function to fit the size of the resulting image by using the cv2.getPerspectiveTransform() function to the size of the original image or video. The cv2.warpPerspective() function returns an image or video whose size is the same as the size of the original image or video.

See the Figure : 4 for better understanding:

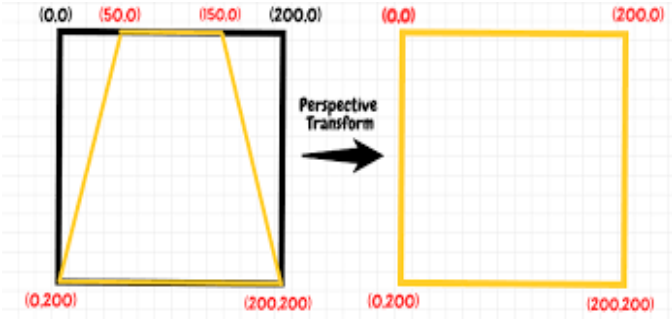

Figure : 4 – Perspective Transform

Since, the transform Matrix (M) is defined by 8 constants that is degree of freedom, thus to find the M matrix, I selected the four points (See Figure : 5) in the input image and mapped them to 4 desired location points in the unknown output image. I had to consider the image dimensions for selecting the input points. Figure : 6 will demonstrate the perspective transform applied to the road lane.

	A	B
1	Source Points	Destination Points
2	200, 460	300, 720
3	1150, 460	1000, 720
4	436, 220	400, 0
5	913, 220	1200, 0

Figure : 5 – Source and Destination Points

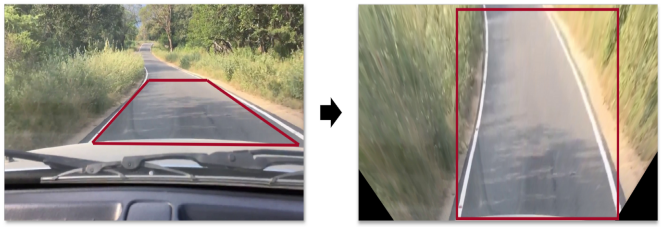

Figure : 6 – Perspective Transform on Road lane

C. *Thresholding*

This step includes the segmentation of the input image after lane lines are parallel. We know that lanes are either white or yellow based. The input image still contains R, G and B channel information. I eliminated the other channels that I do not need by converting the input transformed image into a single channel grayscale image. Another approach that can be used here was using S channel from HLS converted input image. S channel usually provide good results even if there are fluctuations in lighting conditions. This paper contains the thresholding approach used to convert perspective transform into binary image.

I have tuned the minimum and maximum threshold manually by trial and error methods. I have selected 220 values as minimum threshold while 225 as maximum threshold value.

Figure : 7 – Thresholded image of Road lane

cv2.threshold(*image*, 220, 225, cv2.THRESH_BINARY)

As I have used thresholding values manually, instead of this I can also use HED that is Holistically- Nested Edge Detection. HED employs a neural network to extract the edges without manually tuning the threshold values. This technique attempts to address the limitations of the canny edge detector. Figure : 8 will show the HED vs Canny edge detection comparison.

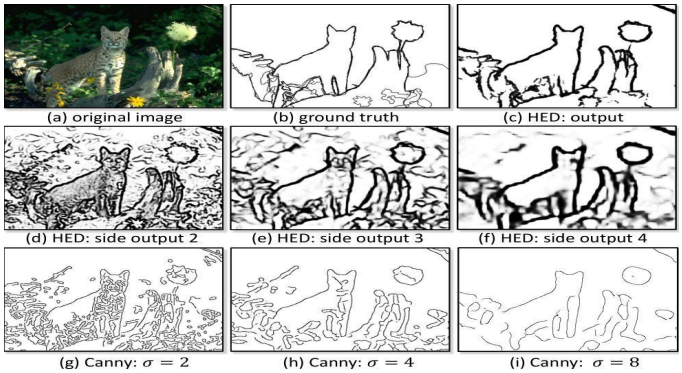

Figure : 8 – HED vs Canny Edge detector

D. *Finding Lane Pixels*

Now, we have preprocessed a transformed binary image, for finding lanes, we first locate and map the lanes in the image space. I plotted the histogram of pixels that are non zero in the lower half of thresholded binary image. The peaks in histogram of binary image indicate lane lines where non-zero pixels are present. See Figure : 9 for this. X coordinate from histogram is considered to be the starting point to search for the respective lanes. I have applied the concept of sliding window, that is a window with a margin is being placed around the line's center one for each left and right lane.

For finding lane pixels, I took histograms of regions along the columns, see attached Figure : 10. I have identified the peaks and then used these peak positions as a starting point to draw windows on the lane marks. See Figure : 11 for this.

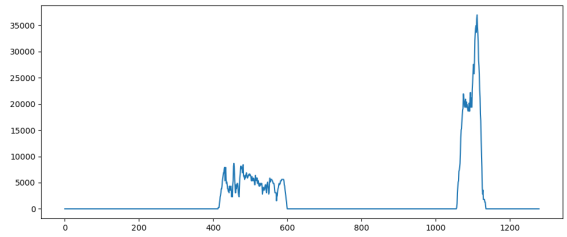

Figure : 9 – Histogram of pixel = x , y = count

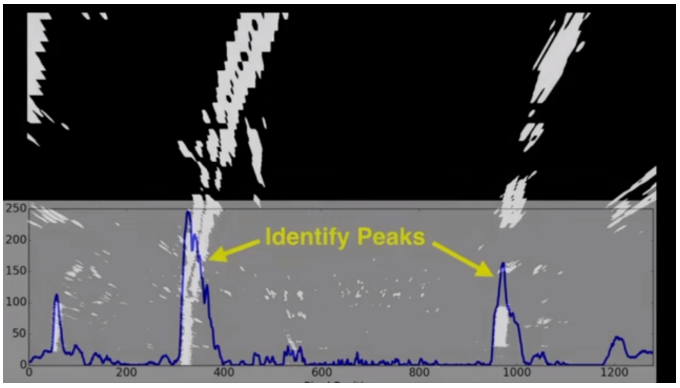

Figure : 10 – Identifying peaks from Histograms

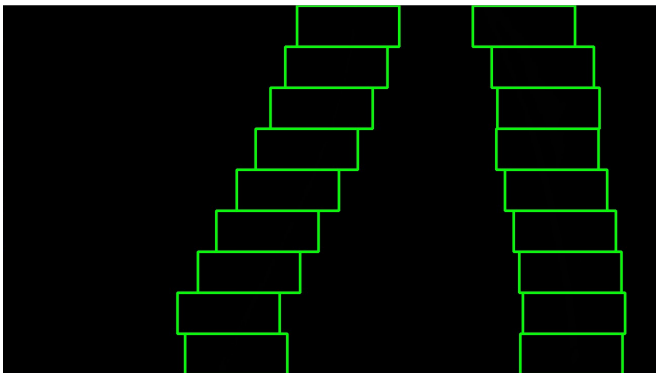

Figure : 11 – Sliding windows

E. Recognizing the lane area

The sliding window gives us the estimated center value of lane area, using these x and y pixel positions, the custom function searches around these polynomial curves and tries to fit the second order polynomial curve. See Figure : 12 for the same.

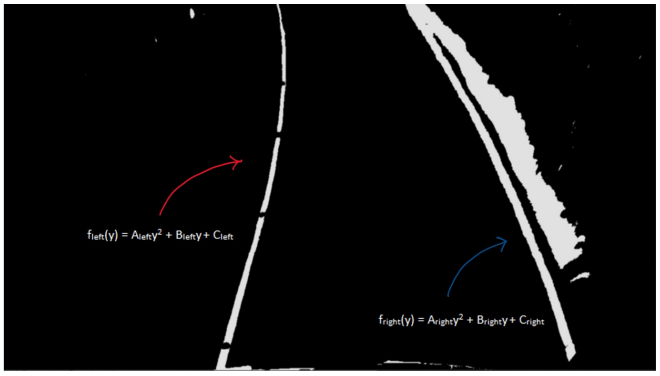

Figure : 12 – Fit the second order polynomial on detected lanes.

F. Compute the radius of curvature

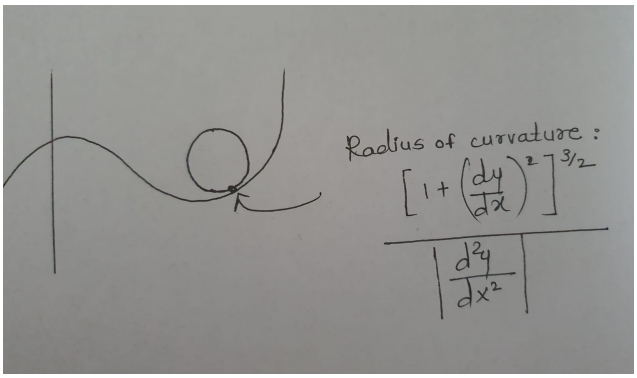

Figure : 13 – Radius of curvature

*To c*alculate radius of curvature with a circle that closely fits nearby points on a local section of a curve as seen in figure 13. The radius of curvature of the curve at a particular point can be defined as the radius of the approximating circle.

The last step would be to highlight the lane area by fitting a polygon between these points and projecting it back on to the original image .The lane area and radius of curvature have been computed in image space based on pixel values which is not the same as real world space, hence they have to be converted into real world values, this involves measuring the length and width of the section of lane on which we are projecting our warped image. For simplicity, we can assume that most lanes are going to be usually 30 meters long and 3.7 meters wide and the code uses these values to convert pixel space values to real-world meter measurements.

IV. RESULT

The project was tricky to implement. I used trial and error techniques for adjusting threshold values and changing color space. The parameters can be tuned to work well on both tracks, but it may fail under the different conditions like during night or different weather conditions. Also, it may fail if another car is close ahead.

The work-flow explained in this paper works perfectly on data such as following :

Figure : 14 – Video frame of Indian road

FIGURE : 15 – VIDEO FRAME OF US ROAD

V. FAILURES AND FUTURE SCOPE

Currently, sliding windows are stacked vertically and only move horizontally, but it is a challenge, when the road is very curved and stacking only vertically loose the line. Moving windows not only horizontally but also adjusting vertically may improve line detection. It is something worth trying in the future.

Adjusting brightness might work for some parts of the road (e.g. under the bridge) but fail when it's very light or vice versa. I believe that checking the density of the pixels would indicate that it's too bright or too dark therefore increasing/decreasing threshold runtime could give better results.

VI. CONCLUSION AND DISCUSSION

To begin with, the approach explained in this paper is a good solution to solve the lane detection problem with pure computer vision techniques. To get more consistent and comprehensive results, we can also use neural network techniques especially when input data frames contain varying lighting and surroundings conditions.

The biggest problem that I faced with this approach is that : it is a difficult process to create the composition of thresholds. This pipeline may fail if there is variation in the light, shade, and noise of the surroundings. There is also a possibility that the thresholded image may not always capture the line lanes and the information that we really need to extract. But this is not an ideal solution, and we need a more powerful algorithm that can compose a set of filters that can learn and construct the output in a more consistent way. Because of that, I think that we cannot completely solve the problem with computer vision, we need machine learning or a convolutional neural network algorithm that can learn how to annotate the road lane lines.

The pipeline fails in low light conditions where the lanes are not visible on lots of continuous frames. This was observed during the testing of the pipeline on low light condition videos.

ACKNOWLEDGMENT

I am highly grateful to Professor Bruce Maxwell for the guidance and liberty given to select any project topic. I would also like to thank kushal kusram whose tutorials and blog I followed in order to implement this project. The course material was very useful and came absolutely handy while working on the breast cancer detection project. I would also like to thank my parents and my partner for their constant support.

REFERENCES

[1] C. Ma, L. Mao, Y. Zhang, M. Xie. 2010, July. Lane detection using heuristic search methods based on color clustering. In 2010 International Conference on Communications, Circuits and Systems (ICCCAS) (pp. 368-372). IEEE.

[2] Baili, M. Marzougui, A. Sboui, S. Lahouar, M. Hergli, J.S.C. Bose, K. Besbes 2017, March. Lane departure detection using image processing techniques. In 2017 2nd International Conference on Anti-Cyber Crimes (ICACC) (pp. 238-241). IEEE.

[3] Qing Lin, Youngjoon Han, Hernsoo Hahn, —Real-time Lane Detection Based on Extended Edge-linking Algorithm, 978-0-7695-4043-6/10 $26.00 2010 IEEE DOI 10.1109/ICCRD.2010.166.

[4] K. Dinakaran. A. Stephen SagayarajS.K. KabileshT. ManiA. AnandkumarGokul Chandrasekaran Advanced lane detection technique for structural highway based on computer vision algorithm.