IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Advance Lane Detection for Autonomous Driving

[1]Sahil Gupta, [2]Niteesh Pal, [3]Vetrivendan L
[1]Student, [2]Student, [3]Assistant Professor
[1]Computer Science & Engineering,
[1]Galgotias University, Greater Noida, India

Abstract : —The objective of this paper is to use computer vision techniques to detect the lanes for an autonomous driving car. Perspective transform together with poly fit functions plays a great role in giving an accurate result. Vision based approach is utilized as it performs well in a wide variety of situations by extracting rich set of information compared to other sensors. The proposed method takes frames from the input video , processes it and detects the lanes present.

IndexTerms **- Lane; Lane detection; Poly curve Fit ;Perspective Transformation; Sliding Window**

.

I. INTRODUCTION

It is well known that lane recognition on freeways is an essential part of any successful autonomous driving system. An autonomous car consists of an extensive sensor system and several control modules. The most critical step to robust autonomous driving is to recognize and understand your surroundings. However, it is not enough to identify obstacles and understand the geometry around a vehicle. Camera-based Lane detection can be an essential step towards environmental awareness. It enables the car to be correctly positioned within the lane, which is crucial for every exit and the back lane - planning decision. Therefore, camera-based accurate lane detection with real-time edge detection is vital for autonomous driving and avoiding traffic accidents.

Lane detection and tracking has been an active research area in the past twenty years mainly for the driver assistance application. Due to the large variations of traffic scenes and illumination conditions, this problem causes the usage of diverse approaches and sensing modalities. Approaches that adopt conventional computer vision techniques are reviewed and compared according to the separate functional modules in a generic framework.
We capture the road image at the beginning, then pre-processing the images as the original images are not suitable for further operation. Assumed that in the initial state, road is approximation straight within a certain length and parallel to the car's central axis. After obtain the first frame image we will do the image pre-processing, and then use perspective transform to detect the edge of the road, and define the detected edges as the initial path of the border. After that the initial boundary conditions were considered as a priori of the follow-up frames for processing. In this step of this algorithm, there are two tasks to be done. Firstly calculated out the ROI base on the priori, then carry through the transform on the ROI, to get the follow-up frames' road edge. The follow-up frames' ROI can be worked out based on the deceived road lane of the last frame. Using this method our algorithm can reduce the search area of the region, at the same time obtain higher detection accuracy, and the algorithm is not so complicated.
The Advanced Lane Detection project is an implementation of computer vision techniques to detect lane lines in images and video footage taken from a camera mounted on a self-driving car. The project uses OpenCV to detect lane lines and to warp the images to a bird's-eye view perspective. The goal of this project is to create an accurate and robust system for detecting lane lines that can be used in a self-driving car.

II. NEED OF THE STUDY.

The main goal of this project is to use traditional Computer Vision (i.e. non-machine learning) techniques to develop an advanced and robust algorithm that can detect and track lane boundaries in a video which would aid in the autonomous car driving system to assist the car to drive in the respective lanes there by preventing accidents and mishaps.

III. RESEARCH METHODOLOGY

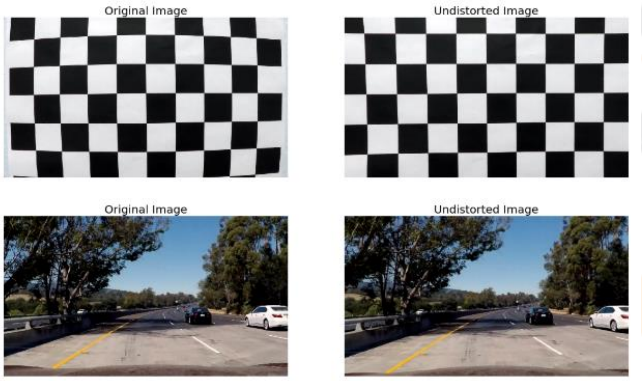
The various steps involved in the pipeline are as follows, each of these has also been discussed in more detail in the sub sections below:

1. Compute the camera calibration matrix and distortion coefficients given a set of chessboard images. Apply a distortion correction to raw images.
2. Capture the video from the camera. (For the testing purpose pre recorded videos are used.
3. Color and Gradient Thresholding
4. Apply a perspective transform to rectify image ("birds-eye view").
5. Detect lane pixels and fit to find the lane boundary.
6. Determine the curvature of the lane and vehicle position with respect to center.
7. Warp the detected lane boundaries back onto the original image.
8. Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position

III.I Camera calibration and distortion correction

The first step in the process is to calibrate the camera to remove distortion from the images. Distortion occurs due to the shape of the camera lens, which can cause straight lines to appear curved in the images. To correct this distortion, the project uses a calibration process in which a set of chessboard images is used to determine the distortion parameters of the camera. These parameters are then used to remove the distortion from the images in subsequent steps.
The next step is to apply the distortion parameters obtained from the calibration step to the images captured by the camera. This step ensures that the straight lines in the images appear straight and the distance between objects is preserved.
OpenCV provides three functions, namely, cv2.findChessboardCorners, cv2.calibrateCamera and cv2.undistortto do just that. Let's unpack this to see how it works:

- Firstly, we define a set of object points that represent inside corners in a set of chessboard images. We then map the object points to images points by using cv2.findChessboardCorners.
- Secondly, we call cv2.calibrateCamera with this newly created list of object points and image points to compute the camera calibration matrix and distortion coefficients. These are henceforth stored in a pickle file in the root directory for future use.
- Thirdly, we undistort raw images by passing them into cv2.undistort along with the two params calculated above.
- This process has been visualised below



III.II Image Capturing

The input data is a color image sequence taken from a moving vehicle. The lane detection system reads the image sequence or a video from the camera and starts processing. The format of the video taken for the project was mp4. For the testing purpose a pre captured video was taken but for final model a camera would be mounted and the video would be taken from that. In order to obtain good estimates of lanes and improve the speed of the algorithm, the original image size was reduced to 1280 x 720 pixels.

III.III Color and Gradient Thresholding

The project then applies color and gradient thresholding techniques to the images to obtain a binary image that highlights the lane lines. The color thresholding technique is used to isolate the yellow and white colors of the lane lines. The gradient thresholding technique is used to detect the edges of the lane lines.

III.IV Perspective Transformation & ROI selection

The perspective transform is a process in which an image is transformed from one perspective to another. In the context of the lane detection project, the perspective transform is used to obtain a bird's-eye view of the road, which helps in detecting the lane lines more accurately.
The perspective transform used in the project is performed using the OpenCV getPerspectiveTransform() and warpPerspective() functions. The process involves selecting four points on the original image that define a trapezoidal region that encompasses the lane lines. These points are chosen based on their location on the road and the camera's position. Once these points are selected, the transform matrix is calculated using getPerspectiveTransform(). The transform matrix maps the four points in the original image to four points in the transformed image.

The next step is to apply the transform matrix to the original image using warpPerspective(). This results in a transformed image that shows a bird's-eye view of the road. The transformed image has the advantage of presenting the lane lines as straight lines, making it easier to detect and track them.
Overall, the perspective transform is an important step in the lane detection process as it provides a bird's-eye view of the road that makes it easier to detect and track the lane lines. The accuracy of the transform depends on the careful selection of the four points that define the trapezoidal region.

The ROI is the portion of the image where the lane lines are expected to be present. The ROI selection is a crucial step in the project as it helps to focus on the important areas of the image and avoid any noise or irrelevant information.
In the project, the ROI selection is done after the perspective transform step. Once the perspective transform is applied to the original image, the ROI is selected using a trapezoidal region that encompasses the lane lines. This region is defined by four points that are carefully selected based on the location of the lane lines in the image. The four points are chosen to create a trapezoidal shape that is wider at the bottom of the image and narrower at the top.
The selected ROI is then used to mask the transformed image, so that only the portion of the image within the ROI is used for further processing. This helps to remove any noise or irrelevant information outside the ROI and focus only on the area of the image where the lane lines are expected to be present.

Overall, the ROI selection is a critical step in the lane detection process, as it helps to focus on the important areas of the image and avoid any noise or irrelevant information. The selection of the ROI is carefully done based on the location of the lane lines in the image, and it ensures that only the relevant portion of the image is used for further processing.

III.V Lane Line Detection

The lane line detection step uses the binary image obtained from the color and gradient thresholding and the perspective transform. The project uses a sliding window technique to search for the lane lines in the image. The sliding windows are positioned at the bottom of the image and are moved up in a search for the lane lines. Once the lane lines are found, a second-order polynomial is fit to the points that define the lane lines.

This was by far the most involved and challenging step of the pipeline. An overview of the test videos revealed the following optical properties of lane lines (on US roads):

- Lane lines have one of two colours, white or yellow
- The surface on both sides of the lane lines has different brightness and/or saturation and a different hue than the line itself,
- Lane lines are not necessarily contiguous, so the algorithm needs to be able to identify individual line segments as belonging to the same lane line.

Many techniques such as gradient thresholding, thresholding over individual colour channels of different color spaces and a combination of them were experimented with over a training set of images with the aim of best filtering the lane line pixels from other pixels. The experimentation yielded the following key insights:

1. The performance of indvidual color channels varied in detecting the two colors (white and yellow) with some transforms significantly outperforming the others in detecting one color but showcasing poor performance when employed for detecting the other. Out of all the channels of RGB, HLS, HSV and LAB color spaces that were experiemented with the below mentioned provided the greatest signal-to-noise ratio and robustness against varying lighting conditions:
   - White pixel detection: R-channel (RGB) and L-channel (HLS)
   - Yellow pixel detection: B-channel (LAB) and S-channel (HLS)
2. Owing to the uneven road surfaces and non-uniform lighting conditions a strong need for Adaptive Thresholding was realised
3. Gradient thresholding didn't provide any performance improvements over the color thresholding methods employed above, and hence, it was not used in the pipeline

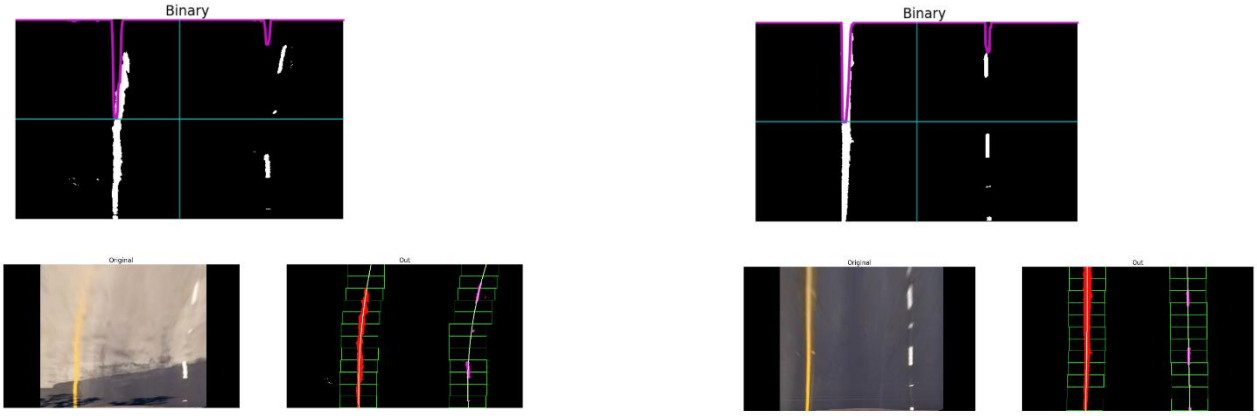
III.VI Sliding Window Technique:

We now have a warped, thresholded binary image where the pixels are either 0 or 1; 0 (black color) constitutes the unfiltered pixels and 1 (white color) represents the filtered pixels. The next step involves mapping out the lane lines and determining explicitly which pixels are part of the lines and which belong to the left line and which belong to the right line.
The first technique employed to do so is: Peaks in Histogram & Sliding Windows

1. We first take a histogram along all the columns in the lower half of the image. This involves adding up the pixel values along each column in the image. The two most prominent peaks in this histogram will be good indicators of the x-position of the base of the lane lines. These are used as starting points for our search.
2. From these starting points, we use a sliding window, placed around the line centers, to find and follow the lines up to the top of the frame.

The parameters used for the sliding window search are:

- nb_windows = 12 # number of sliding windows
- margin = 100 # width of the windows +/- margin
- minpix = 50 # min number of pixels needed to recenter the window
- min_lane_pts = 10 # min number of 'hot' pixels needed to fit a 2nd order polynomial as a lane line



Here, the red and pink color represents the 'hot' pixels for the left and right lane lines respectively. Furthermore, the line in green is the polyfit for the corresponding 'hot' pixels.

III.VII Adaptive Search

Once we have successfully detected the two lane lines, for subsequent frames in a video, we search in a margin around the previous line position instead of performing a blind search.
Although the Peaks in Histogram and Sliding Windows technique does a reasonable job in detecting the lane line, it often fails when subject to non-uniform lighting conditions and discolouration. To combat this, a method that could perform adaptive thresholding over a smaller receptive field/window of the image was needed. The reasoning behind this approach was that performing adaptive thresholding over a smaller kernel would more effectively filter out our 'hot' pixels in varied conditions as opposed to trying to optimise a threshold value for the entire image.Hence, a custom Adaptive Search technique was implemented to operate once a frame was successfully analysed and a pair of lane lines were polyfit through the Sliding Windows technique. This method Follows along the trajectory of the previous polyfit lines and splits the image into a number of smaller windows.These windows are then iteratively passed into the get_binary_image function (defined in Step 4 of the pipeline) and their threshold values are computed as the mean of the pixel intensity values across the window
Following this iterative thresholding process, the returned binary windows are stacked together to get a single large binary image with dimensions same as that of the input image.
At the end to get the real dimensions the pixel is converted to metres according to the us standards . After the conversion from the wrapped images it was found that a minimum lane width of 12 feet or 3.7 meters, and the dashed lane lines length of 3.048 meters was required.

III.VIII Curvature and Offset

Following this conversion, we can now compute the radius of curvature at any point x on the lane line represented by the function x = f(y) as follows:

$$R_{curve} = \frac{[1+(\frac{dx}{dy})^2]^{3/2}}{|\frac{d^2x}{dy^2}|}$$

In the case of the second order polynomial above, the first and second derivatives are:

$$f'(y) = \frac{dx}{dy} = 2Ay + B$$

$$f''(y) = \frac{d^2x}{dy^2} = 2A$$

So, our equation for radius of curvature becomes

$$R_{curve} = \frac{[1+(\frac{dx}{dy})^2]^{3/2}}{|\frac{d^2x}{dy^2}|}$$

Code for the same is represented below:

```
left_curveR = ((1 + (2*left_fit[0] *y_eval + left_fi
t[1])**2)**1.5)  / np.absolute(2*left_fit[0])

right_curveR = ((1 + (2*right_fit[0]*y_eval + righ
t  fit[1])**2)**1.5) / np.absolute(2*right  fit[0])
```

III.IX Visulaization

Finally, the detected lane lines and the radius of curvature are visualized on the original image. This step provides a visual representation of the lane lines and their position on the road.
To make it look appealing and clear the lanes are coloured in separate colour and the direction and offset from centre are represented in numbers. For a upcoming turn a sign board board is used of the respective direction..

IV. RESULTS AND DISCUSSION

Images taken from input video:

Images taken from output video:

As we can see the curved lanes are being perfectly detected as well as the distance from the centre is also corrected mentioned in the output.
In this paper, a real time lane detection algorithm based on video sequences taken from a vehicle driving on highway was proposed. As mentioned above the system uses a series of images taken frame by frame from a input video. The lanes were detected using edge detection, perspective transformation with restricted search area. The proposed lane detection algorithm can be applied in both painted and unpainted road, as well as curved and straight.

The accuracy for curved lines is also good as we can see from the results that the curved lanes are being perfectly detected and the upcoming turns are being predicted correctly.

Majority of the projects available uses Hough Transform for the detection of lines but the main issue with this method is that it fails to detect the curved lines which can be very dangerous for a autonomous driving car system in real life as there we do often encounter turnings and curved lanes. The method uswed in this projects differentiates it dfrom that of hough transform as it can perfectly detect the curved lanes as well as straight lanes.

The major challenge for the project was the lighting conditions in the video which kept differencing from video to video. So for different videos parameters like color thresholding and the region of interests were tweaked to make them fit the model. This algorithm was tested on three different videos and the accuracy was good for every video.

REFERENCES

[1] J. C. McCall and M. M. Trivedi, "An Integrated Robust Approach to Lane Marking Detection and Lane Tracking", Proc. of IEEE Intelligent Vehicles Symposium

[2] Wenjie Song, Yi Yang, Mengyin Fu, Yujun Li, Meiling Wang, "Lane Detection and Classification for Forward Collision Warning System Based on Stereo Vision", IEEE Sensors Journal, vol.18, 2018.

[3] Huang Y, Li Y, Hu X, Ci W. Lane detection based on inverse perspective transformation and Kalman filter. KSII Trans Internet Inf Syst 2018

[4] Wang Z, Li X, Jiang Y, et al. swDMR: a sliding window approach to identify differentially methylated regions based on whole genome bisulfite

[5] Dahyot Rozenn, "Statistical hough transform", Pattern Analysis and Machine Intelligence IEEE Transactions

[6] Bradski Gary and Adrian Kaehler, "Learning OpenCV: Computer vision with the OpenCV library" in , O'Reilly Media, Inc., 2008.

[7] W. T. Freeman and E. H. Adelson. "The design and use of steerable filters", PAMI, 13(9):891-906, 1991.

[8] J. McDonald. "Detecting and tracking road markings using the Hough transform." Proc. of the Irish Machine Vision and Image Processing Conference, 2001.

[9] D. Pomerleau. Ralph: "Rapidly adapting lateral position handler." Proc. IEEE Symposium on Intelligent Vehicles, September 25-26, 1995.Stefan Gehrig, Axel Gern, Stefan Heinrich and Bernd Woltermann, Lane recognition on poorly structured roads-The Bot Dot problem in California", Proc. of the 5 Conference on Intelligent Transportation Systems, 2002.

[10] J. B. Southall and CJ. Taylor "Stochastic road shape estimation" International Conference on Computer Vision . pp. 205-212, June 2001

[11] M. Bertozzi and A. Broggi. "GOLD: a Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection." IEEE Transactions on Image Processing. 1997.

[12] J. Kosecka, R. Blasi, C. Taylor and J. Malik. "A Comparative Study of Vision-Based Lateral Control Strategies for Autonomous Highway Driving". In IEEE Int. Conf. on Robotics and Automation, pp 1903-1908, May 1998.

[13] CJ. Taylor and Jitendra Malik and Joseph Weber. "A Real-Time Approach to Stereopsis and Lane-Finding." Intelligent Vehicles 1996, pp 207-212

[14] K. Huang, M. M. Trivedi, T. Gandhi, "Driver's View and Vehicle Surround Estimation using Omnidirectional Video Stream," Proc. IEEE Intelligent Vehicles Symposium, Columbus, OH, pp. 444-449, June 9-11, 2003

[15] H. Furusho, R Shirato, M. Shimakage. "A Lane Recognition Method Using the Tangent Vectors of White Lane Markers," 6th International Symposium on Advanced Vehicle Control, Sept. 9-13, 2002

[16] J. McCall, O. Achler, M. M. Trivedi. "The LISA-Q Human-Centered Intelligent Vehicle Test Bed" To Appear in Proc. IEEE Intelligent Vehicles Symposium, Parma, Italy, June 14-17, 2004

[17] Apostoloff, N. and A. Zelinsky, 2003. Robust vision based lane tracking using multiple cues and particle filtering. Proceeding of the Intelligent Vehicles Symposium, IEEE, July 28, 2003, Dept. of Eng. Sci., Oxford Univ., UK, pp: 558-563.
CrossRef

[18] Arlicot, A., B. Soheilian and N. Paparoditis, 2009. Circular road sign extraction from street level images using colour, shape and texture database maps. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., 38: 205-210.
Direct Link