Real-time Lane Detection, Fitting and Navigation for Unstructured Environments

Apoorve Singhal*, Arvind Jha*, Deepank Agrawal*, Vibhakar Mohta*, Yash Khandelwal*, Shreyas Kowshik*, Siddhant Agarwal*, Shrey Shrivastava*, Vaibhav Lodhi, Debashish Chakravarty

ABSTRACT

In the outdoor environment, robot perception is a challenging task encompassing several layers of abstractions like lane detection, object detection and avoidance, and way-point navigation. Intelligent Ground Vehicles are becoming popular and having an efficient perception stack is quintessential to its scaling for different tasks. Several issues like illumination variance, shadows, occlusions, etc. cause researchers to adopt computationally heavy approaches for improving generalization. We present a novel, real-time approach for combined lane detection, obstacle detection, and way-point navigation using features from a 2D-LiDAR and camera. A robust curve fitting algorithm has been implemented, adhering to the minimization of computation. The overall processing pipeline has been tested and validated to work well in outdoor conditions.

Keywords: Image Processing, Lane Detection, Computer Vision, RANSAC, Low computation, Lane Classification, Robotics, Autonomous Driving

1. INTRODUCTION

Autonomous vehicles are becoming increasingly popular in a multitude of applications and are the rightful subject of enormous research by academia as well as the industry. As the environments grow in complexity, so do the on-board perception systems required to make intelligent decisions with respect to the environment. The significant challenges faced by any autonomous ground vehicle include drivable region identification, obstacle detection, and way-point generation for path planning. The complexity of such algorithms, however, makes them quite resource intensive. With the advent of enormous computing power, there has been a boost in several image processing based perception algorithms in the last decade, which are crucial to advanced driver assistance systems (ADAS).

A future with autonomous vehicles is inevitable; however, it will be difficult to realize without a significant reformation of our existing infrastructure, namely proper lane markings, stop signs, traffic lights, and traffic signs. From the current state of autonomy till the time full self-driving capabilities are achieved, a transition period will have to be endured while the infrastructure catches up. Even today, only a few major cities possess the required groundwork. Despite these reasons, we feel that research on off-road autonomy has been underwhelming in comparison to structured environments.

Much work has been done on lane detection in structured environments, like searching for bright vertical lines in a top view image,[1] using a variety of features such as Gaussian kernels for template-based matching and edge-based features to extract lane edges.[2] Previous work on off-road driving scenarios includes using the method of mixed channels to segment lane pixels[2] and vanishing point estimation.[3] We, however, argue that unstructured environments, by definition, lack the same set of constraints. Finally, we hope that our work inspires further developments in the domain.

Lane detection poses the problem of identifying the pixels which correspond to the lane markers. However, several challenges interfere with the proper detection of lanes such as the presence of shadows, glares, variations in lighting conditions, and occlusions.[4] Moreover, the absence of constant width lanes and the presence of very sharp turns further complicates the problem. Before identifying the lane pixels, obstacles present in the image have to be removed. The proposed algorithm detects and removes obstacles by thresholding on linear combinations of the color spaces. Using inverse homographic projection[5,1] of the laser scanner points, the vision approach validates the obstacles detected. This two-step process prunes the result by eliminating false positives.

* Denotes Equal Contribution

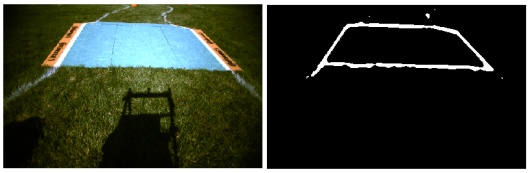

Figure 1. Ramp detection (a) Original image (b) Pre-processed image

Owing to nearby objects and varied lighting conditions, shadows are cast on regions of interest. We refer to[6,7,8] for a comprehensive literature survey on shadow removal. Here, we briefly discuss the work relevant to ours.[8] uses a two-step process to identify and remove shadows. Our application does not require very finely removed shadows, so we trade off finer texture for faster processing times and obtain a coarser image by removing the second fine-tuning step.

For lane model estimation, even though multiple approaches have been discussed at length in the literature, spline fitting[9] and polynomial fitting[10] is the most widely used. Although splines can achieve better generality than polynomials but constraining the degrees of freedom works better for our purpose. The final task performed by the perception system is generating optimal way-points for the planning subsystem.

The paper is divided into the following sections. Section 2 describes the proposed methodology. Improvements over other approaches have been described in Section 3, and experimental results are discussed in Section 4. The conclusions have been presented in Section 5.

2. PROPOSED METHODOLOGY

Inverse homography matrix is calculated for the monocular RGB camera setup on the mobile vehicle. Obstacles and ramp (if any) are removed from the front view image to facilitate lane detection and curve estimation. The resulting top-view image is pre-processed to remove noise and identify lane points. The filtered points then undergo a homogenization step to provide a uniform input to the curve fitting algorithm. The curve parameters obtained are then transferred to the way-point generation pipeline for navigation.

The entire processing pipeline can be divided into different sub-sections, as given in Fig 4. 2.1 presents the obstacle detection module. 2.2 describes the ramp detection methodology. Lane feature extraction, lane model determination, and parameter estimation have been elaborated in sub-sections 2.3, 2.4 and 2.5, respectively. 2.6 explains lane classification (left lane or right), which is further used for way-point navigation (2.7).

2.1 Obstacle Detection

There are two primary uses of obstacle detection, informed path planning, and removal of obstacle features for better lane detection. Pre-processing of images is done primarily using the color channels. Obstacles containing white stripes pose a problem in our lane detection algorithm, and such obstacles (mainly orange) are our primary point of concern. Thresholding on the $R - G$ channel followed by morphological dilation operation removes the orange obstacles. We empirically found that the pixels with $B/2 > G$ and $B/2 > R$ belong to the blue obstacles while those with $B < R/2$ and $G < R/1.5$ belong to the orange obstacles. However, this approach is not reliable independently. For validation, the laser scan points, coming from a LiDAR sensor mounted on the vehicle, are projected onto the front view image. This gives a set of points which lie on the base of the obstacles. These points are then stretched to cover the obstacles with the aid of a hyper-parameter. Both these approaches are fused to increase robustness.

2.2 Ramp Detection

Our obstacle detection algorithm flags the ramp that appears in the IGVC arena. Hence it is not possible to detect the lanes on the ramp without a dedicated module for ramp detection. A separate ramp detection step can also serve future purposes of changing the vehicles velocity on similar inclines. Fig 1(a) shows the ramp image input and 1(b) shows how the ramp appears after the initial pre-processing. Although, at first impression,

it seems that ramp detection can be seen as a problem of trapezoid detection, but a slight reformulation gives better results. Redefining the problem as detection of three geometrically constrained lines, one horizontal and two vertical, instead of a trapezoid saves computation and leads to more robust detection since the top edge of the trapezoid is not always appropriately detected.

2.3 Lane Feature Extraction

Once the obstacles and ramp have been segmented out of the image, lane detection is carried out on the remnant. We empirically formulate three filtered channels: $2B-G$, B and $2B-R$ and take the intersection of these images to reduce falsely detected lane markings. A median blur is applied to the pre-processed image to remove any salt and pepper noise. The resulting image undergoes an adaptive thresholding step to find distinctive lane regions from local neighborhoods, leaving out all white noise like glares since they don't stand out from the lanes.

To aid curve fitting, the image is divided into grids of size 3x3, and the central element of each is replaced with a white pixel only if there are at least three white pixels in that grid. This decreases the number of inliers and outliers by the same factor. This reduction in the density of inliers, combined with the reduced density of outliers available for curve fitting, significantly decreases the number of wrongly fit curves. A consistent density of inliers also ensures uniformity in the number of points for fitting across different frames, making it threshold independent and invariant to lighting conditions.

In our case, images of dimensions 300x480 were used, for which the number of inliers required by the lane parameter estimation step was about 65. Although this number can vary for different camera setups, it should be reasonably constant for any particular camera setup, independent of lighting conditions.

2.4 Lane Model Determination

A continuous curve has to be fit on the detected lane pixels for the following reasons :

- The lanes are published on the cost map, and must not have any gaps to ensure that a path is not planned through them.
- The curve must be continuous and differentiable to facilitate the generation of way-points.

The lanes are initially modeled with straight lines in the Hough space. The resulting lines are judged on three parameters - length of the line, the angle subtended with the horizontal and the number of inliers used. However, if a satisfactory score is not obtained, then the algorithm uses second-degree parabolic equations.

2.5 Lane Parameter Estimation

In case of a linear lane model, the lane parameters are estimated using the Hough Lines Algorithm.[11] If the lane model is determined to be a second-degree polynomial, we use RANSAC,[12] an iterative algorithm for parameter estimation robust to outliers. Since outliers do not influence the parameter estimates, RANSAC ensures an unbiased fit over the observed data.

In the front view, the axis of the parabolas is constrained to be at the bottom-most row of the image for the estimation to be more robust to noise. This reduces one degree of freedom from the lane model, and the lane equation becomes : $y^2 = \lambda(x - c)$. As the way-points are to be generated in real-world distances, we transform the lane equations to the top view. To do so, we first transform the points of the estimated lane from front view to the top view. However, the lane detection in top view requires free movement of the parabola along the y-axis, and thus, the parabolic equation used is: $x = ay^2 + by + c$. For both the parabolas, the origin is defined at the left bottom of the image, with the positive x-axis extending to the right and the positive y-axis pointing upwards.

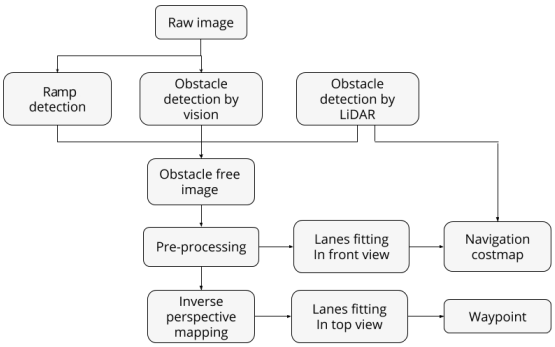

Figure 2. Overall Processing Pipeline

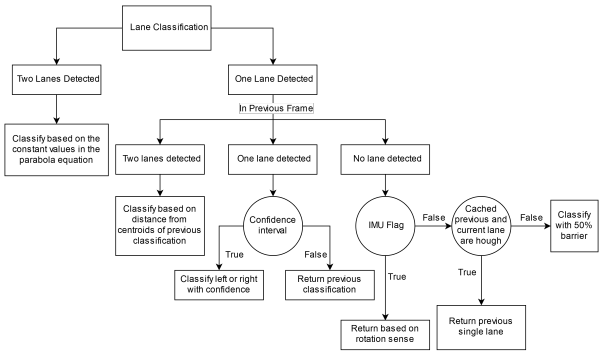

Figure 3. Control flow of Lane Classification

2.6 Lane Classification

Lane Classification is the task of classifying detected lanes as left or right. When two lanes are detected, it is intuitive that the lane with the smaller x-intercept is the left lane and vice versa. However, the problem is non-trivial in the case of one-lane detection. We use the information from the previous frame as a prior to classifying in the current frame. If two lanes were detected in the last frame, we would classify the single-lane based on the distance of its centroid from the centroids of previous lanes. In case of only one lane detected in the previous frame, we define the region outside the central two-thirds part of the image as the confidence region. If the lane detected intercepts the x-axis inside this region, we would classify the lanes based on its x-intercept. For detections outside the confidence region, we return the previous classification.

If no lane was detected in the previous frame, it is possible that the lane lines might not be in the field of view of the camera or the lanes might be occluded by obstacles. In that case, it might be possible that suddenly one of the lanes gets detected, then it would be wrong to classify the lane just on the previous frame data or confidence region. We propose an approach, which uses vehicle motion information for classification in such a case. We keep track of the classification of the last detected lane (called previously cached lane) and relative orientation of the mobile base concerning the lane markings in a global frame. The absolute yaw of the base and the lane markings can be measured using an Inertial Measurement Unit (IMU) mounted on the base. If the change in the relative orientation of the mobile base concerning the lane markings were around 90° between the two instances of lane detection, we would classify the lanes based on the sense of rotation in the global frame. For anti-clockwise motion, the lane is classified as left, and for clockwise motion, a right lane is classified.

2.7 Way-point Navigation

Once we get the equations of the lane parabola, we need to generate an appropriate way-point for the mobile bot. The navigation goal has to be nearly halfway across the two lanes and also far enough from the bot to enable the planning algorithm to plan a path. This is achieved by simultaneously minimizing the cost of distance from obstacles and the lane centerline. Many times we do not get both the lanes in our field of view. In that case, we have to predict the other lane on our own to get the way-point. To do so, we keep an exponential running average of the width between the two lanes. Using this and knowing whether it is a left or right lane, we predict the other lane and then generate a mid lane. The way-point P is chosen such that it is at a distance of 3.5 meters from the bot and minimizes a cost function γ_P defined as $\gamma_P = \alpha C_P + \beta \frac{1}{O_P}$ where C_P is the distance from the center lane and O_P is the distance from the nearest inflated obstacle. The orientation of the way-point is decided based on the average of the slope of both lanes in the current frame.

3. IMPROVEMENTS OVER OTHER APPROACHES

In order to increase robustness and generalization, several pipelines use computationally heavy approaches like Deep Neural Networks[13] and Simple Linear Iterative Clustering.[5] Taking into account the power and computational constraints of mobile robots, our design relies mostly on traditional image processing techniques. This allows real-time running of the entire software stack on a 7th generation i5 Intel processor without any dedicated

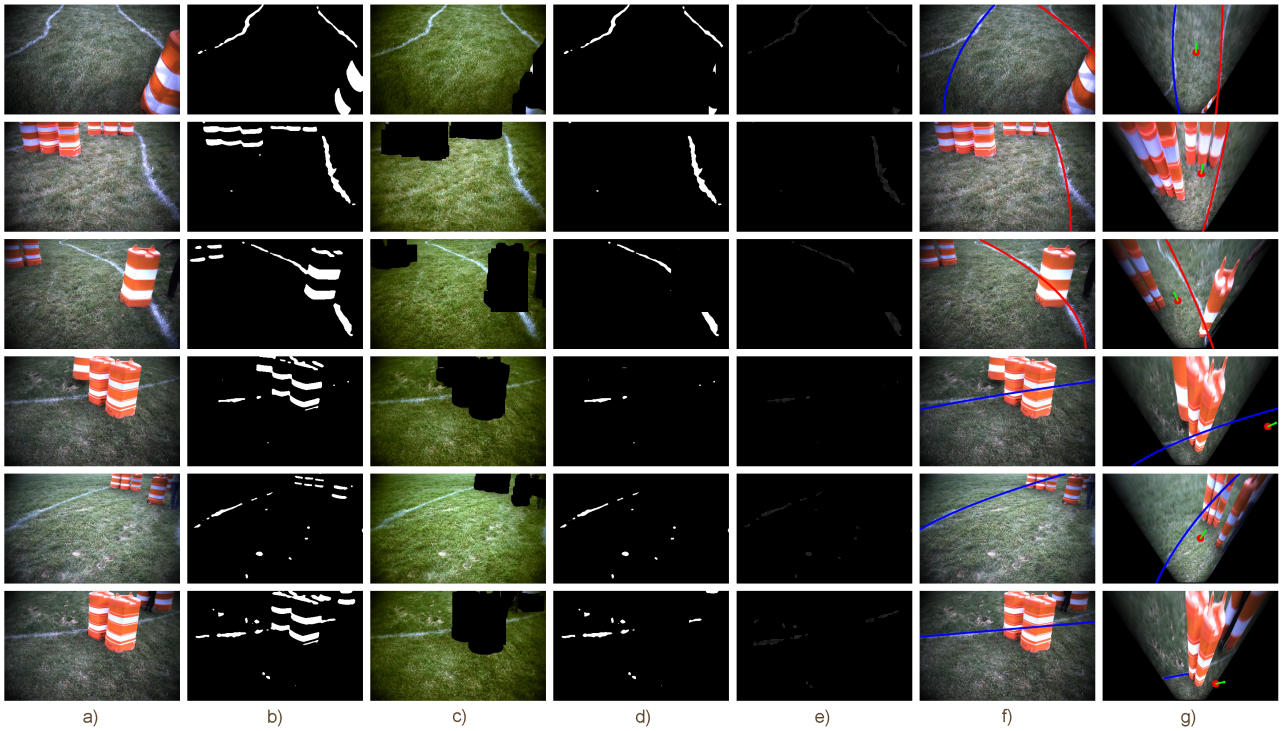

Figure 4. Overall Processing Pipeline From left to right (a)Acquired image from the camera (b) Pre-processed image (c) Obstacle detection (d) Pre-processed image after obstacle removal (e) Grid formation (f) Lane Model Estimation (g) Way point generation

GPU support. However, this comes with a trade-off between robustness and run-time of the algorithm. Traditional image processing approaches, in general, involve setting hard thresholds for feature extraction. This makes the pipeline highly sensitive to changes in environmental conditions and generalizes poorly. In order to reduce reliance on hard thresholds and ensure generalization, multiple processing strategies were employed parallelly and fused later to obtain the processed lane pixels. This increases the robustness across a range of conditions involving light patches, shadows, and patchy lane clusters.

Curve fitting using RANSAC generally employs a hard threshold on the number of inliers. This requires manual fine-tuning and is quite sensitive to changes in features. So in order to reduce reliance on the inlier threshold, a grid-based approach is used. This decreases the number of inliers and outliers by the same factor and brings them to an equivalent count. This curbs the need for fine-tuning and increases the robustness in cases where features are sparse.

4. EXPERIMENTS

Previous works have used the inverse perspective transformed image for lane fitting, in which a generalised parabola was fit. Such works calculate the lane equation in front view and transform it in top view. This is quite intuitive, as we can expect the parabolas to accommodate to any arbitrarily shaped lanes. During our experiments, however, we found a common situation where the above approach fails. For instance, when the mobile robot changes its orientation from being parallel to the lanes to an orientation at a steep angle relative to the lane, the earlier algorithm increasingly started to fit lanes on the noise, in an attempt to gather more number of inliers. We have two important findings, that we should calculate the lane equation in both front and top view separately to prevent error propagation; and that constraining the degrees of freedom (in top view) of the lane curve reduces lane misfitting on noise. Constraining the axis of the parabola (in front view) to the bottom-most row of the image turns out to be a valid assumption. Combining these two, we chose to fit lanes in the front view (2 DOF) and then transform it to top view (3 DOF).

5. CONCLUSIONS

This paper presents an optimized RANSAC-based curve fitting and way-point navigation algorithm, robust to several external environmental conditions. A Hokuyo UTM-30 LX Lidar and a Blackfly 2.3 MP Mono GigE PoE camera were mounted on a three-wheeled differential drive robot for the on-ground testing and validation of the algorithm. The lane detection algorithm was tested on Intel i5 7th generation processor without GPU support, running at a frame rate of 20fps.

We managed to secure the 2nd position in the AutoNav challenge of 2019 using the designed algorithm, which requires completing a course by respecting the lane boundaries, avoiding obstacles and following a set of GPS way-points with intermediary switching between the computer-vision and GPS-navigation stack. The environment consists of lane lines drawn on the grass and cylindrical barrels serving as obstacles. Hence, the algorithm has been validated satisfactorily on a variety of externally controlled environments, and the result was way more than satisfactory. However, there is still scope for future work on improving robustness.

6. ACKNOWLEDGEMENT

We are highly indebted to Autonomous Ground Vehicle(AGV), IIT Kharagpur, and Sponsored Research Industrial Consultancy(SRIC), IIT Kharagpur for providing us with the resources and financial assistance for the project. We would like to acknowledge the dedicated efforts put in by Naman Jain, Gaurang Mohta, Rutav Shah, Nikhil Popli, Arnesh Kumar Issar, Shubhesh Anand, Indranil Ray, Raj Gupta, Shreyansh Darshan, Kirtan Mali, Siddharth Gupta, and Dvij Kalaria during this task.

REFERENCES

[1] M.-S. Wang C.-C. Lin. A vision based top-view transformation model for a vehicle parking assistant. *Sensors*, 12(4):4431–4446, 2012.

[2] Ajaykumar R, Arpit Gupta, and Prof S N Merchant. Automated lane detection by k-means clustering: A machine learning approach. *Electronic Imaging*, 2016(14):1–6, 2016.

[3] Manh Cuong Le, Son Lam Phung, and Abdesselam Bouzerdoum. Lane detection in unstructured environments for autonomous navigation systems. In Daniel Cremers, Ian Reid, Hideo Saito, and Ming-Hsuan Yang, editors, *Computer Vision – ACCV 2014*, pages 414–429, Cham, 2015. Springer International Publishing.

[4] Yuan-Hsin Chen Shin-Ping Lin and Bing-Fei Wu. A real-time multiple-vehicle detection and tracking system with prior occlusion detection and resolution, and prior queue detection and resolution. *18th International Conference on Pattern Recognition (ICPR)*, 2006.

[5] S. Agrawal, I. K. Deo, S. Haldar, G. R. Kranti Kiran, V. Lodhi, and D. Chakravarty. Off-road lane detection using superpixel clustering and ransac curve fitting. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1942–1946, Nov 2018.

[6] Yael Shor and Dani Lischinski. The shadow meets the mask: Pyramid-based shadow removal. *Computer Graphics Forum*, 27(2):577–586, April 2008.

[7] Li Xu, Feihu Qi, Renjie Jiang, Yunfeng Hao, Guorong Wu, L Xu, F Qi, R Jiang, Y Hao, and G Wu. Shadow detection and removal in real images: A survey. *Shanghai JiaoTong University, PR China*, 2006.

[8] Kaushik Deb and Ashraful Huq Suny. Shadow detection and removal based on ycbcr color space. *Smart CR*, 4:23–33, 2014.

[9] Eam Khwang Teoh Yue Wang, Dinggang Shen. Lane detection using spline model. *Elsevier Science B.V.*, 2012.

[10] Eva Ostertagova. Modelling using polynomial regression. *Elsevier Science B.V.*, 2000.

[11] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972.

[12] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[13] D. V. Prokhorov J. Li, X. Mei and D. Tao. Deep neural network for structural prediction and lane detection in traffic scene. *IEEE Transactions on Neural Networks and Learning Systems*, 28:690–703, 2017.