Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS 2019)
IEEE Xplore Part Number: CFP19K34-ART; ISBN: 978-1-5386-8113-8

A Self-Driving Car Implementation using Computer Vision for Detection and Navigation

Bhaskar Barua
Information Technology
Sardar Patel Inst. of Technology
Mumbai, India
bhaskar11.bb@gmail.com

Clarence Gomes
Information Technology
Sardar Patel Inst. of Technology
Mumbai, India
gomes.clarence97@gmail.com

Shubham Baghe
Information Technology
Sardar Patel Inst. of Technology
Mumbai, India
shubhambaghe8@gmail.com

Jignesh Sisodia
Information Technology
Sardar Patel Inst. of Technology
Mumbai, India
jsisodia@spit.ac.in

***Abstract*—In recent years, lidar has been used as primary sensors for self-driving cars, however, due to their high expense, it becomes infeasible for mass production. Hence, we present the working of a self-driving car prototype that relies upon a cheaper alternative, viz. cameras. The primary objective of our prototype is to navigate safely, quickly, efficiently and comfortably through our virtual environment using computer vision. We have performed detection of lanes, traffic cars, obstacles, signals, etc. and have used the concept of stereo vision for depth calculation. Trajectory planning and steering control have also been implemented. Experimental results show that camera-based self-driving cars are viable and thus our paper can provide a foundation for all future real-world implementations.**

Keywords—self-driving cars; autonomous vehicles; computer vision; stereo vision.

I. INTRODUCTION

Top companies in the world are investing billions of dollars in building autonomous vehicles. This technology has vast disruptive potential. Furthermore, researchers predict that using self-driving cars on a large scale level will reduce CO2 emissions by 60% and enhance fuel efficiency. It will also help to reduce around 90% of vehicular accidents which happen due to human error. Hence, self-driving cars will greatly improve road safety, enable increased mobility and help avoid traffic congestion.

In recent years, lidar [1] (light detection and ranging) has been used as primary sensors for self-driving cars. Implementations such as [2] and [3] rely heavily on lidar for detection and navigation. However, currently Lidar is an expensive technology and thus mass production of autonomous vehicles using it is not feasible. The simplest alternative is to use camera sensors. The main idea behind this is that, if it is possible for us humans to drive using only our eyes, then it should also be possible to develop an autonomous vehicle to do the same. Hence, we aim to build a self-driving car prototype (in a simulation) that will completely rely on cameras for navigation. For the prototype, we have designed a 3D virtual city depicting a realistic environment with traffic cars, obstacles and traffic signals. The primary objective of our self-driving car is to navigate safely (avoid traffic violations such as collisions, disobeying signals and crossing speed limit), quickly (reach the destination as fast as possible within speed limit), efficiently (avoid frequent accelerations as it leads to more fuel consumption) and comfortably (experience minimum jerk) around the city.

II. PROPOSED PIPELINE

Several camera based self-driving car implementations such as [4] operates as a black box. This may not be the best approach as we don't have much control of our system. Hence, we have created a detailed pipeline with the freedom to apply safety protocols at any level.

The pipeline of our project is shown in Fig. 1. We have broadly classified modules as perception, coordinate transformation, trajectory planning, and control. The pipeline will be executed for each frame captured by our camera sensors.

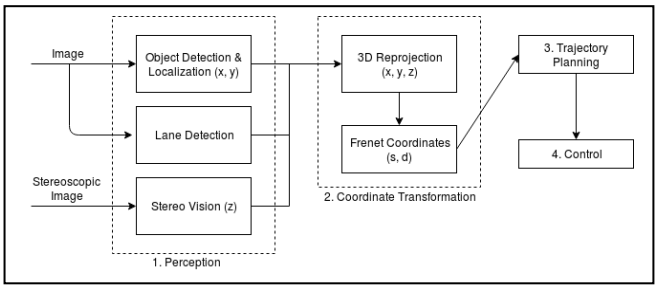

Fig. 1. Overview of the project pipeline

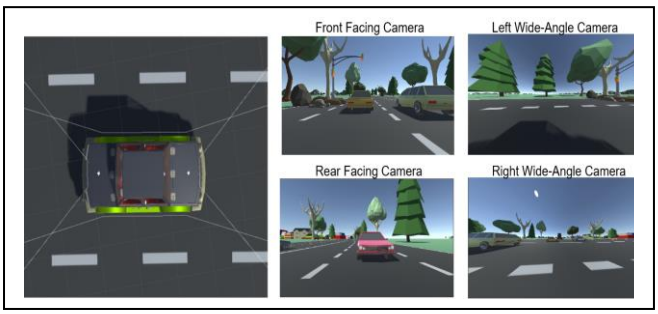

Fig. 2. 360-degree coverage using 4 camera

A. Perception

We intend to build a self-driving car that relies only on cameras for navigation. We have used four cameras - a front-facing camera, a rear-facing camera and one wide-angle camera on either side of our car (as shown in Fig. 2). This paper mostly relies on the front-facing camera for all the operations.

All the cameras are also coupled with stereo cameras. This allows the camera to simulate human binocular vision and therefore gives it the ability to capture three-dimensional images. This brings in the concept of stereo vision which will be used to calculate the depth of each object in the environment.

1) Object Detection and Localization: We detect the objects and find their respective positions in our environment with the help of YOLOv3 [5] object detection algorithm. We have used YOLOv3 since it is faster and yields higher accuracy than R-CNN as verified in this paper [6]. We have trained our model to detect the following classes - traffic cars, obstacles and traffic signals. In this algorithm, we consider the center of each bounding box as the x, y cartesian coordinates of the corresponding objects.

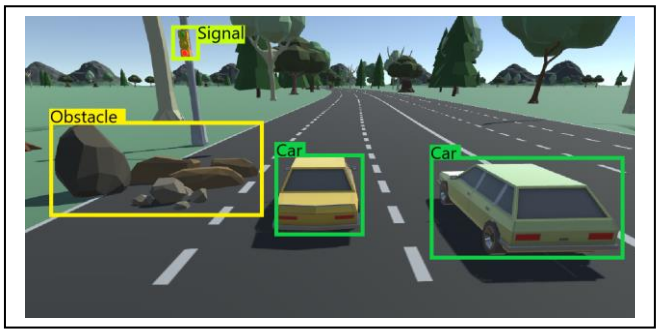

Fig. 3. YOLOv3 detecting cars, obstacles and traffic signals.

Fig. 3 illustrates the detection of various classes in our environment. We have a limited number of classes as well as limited variation in each class (since we are using low-poly 3D simulation). Hence, we have manually generated our training dataset by taking screenshot images of each class and then used data augmentation techniques as proposed in [7-8] to increase our dataset size.

2) Lane Detection: We have to detect both straight and curved lanes present in our environment as proposed in [9-10]. The first step includes inverse perspective mapping which gives us the bird's eye view of the road ahead of us. In order to reduce noise, we apply gaussian processing and thresholding.

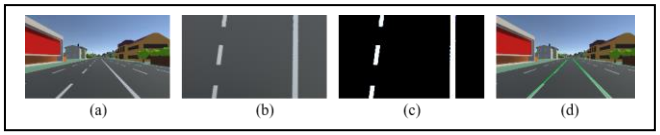

Fig. 4. (a) original image; (b) bird's eye view; (c) threshold image; (d) lanes detected

We use the RANSAC [9] (random sample consensus) algorithm to fit appropriate curve representing our current road segment.

Fig. 4 shows the operations performed to detect lanes. Finally, after all these operations, we get the equation of cubic spline r representing our current road segment.

3) Stereo-Vision: 3D information can be extracted by studying relative positions of objects in the two frames by comparing information about a scene from two different points as proposed in [11]. For example, when we close one of our eyes to view an object at an arbitrary distance, we see the object at some position and when we open the closed eye by simultaneously closing open eye, we will find that it has moved by a slight distance. This distance is called disparity.

The two images captured by the stereo camera can be used to calculate the disparity map (2D matrix containing the disparity of each pixel). Finally, depth *D* for each pixel can be calculated as,

$$D = Bfd^{-1} (1)$$

where B is the distance between two cameras which is known and f is the focal length of the camera and d is the disparity of each pixel.

Fig. 5 shows the disparity map obtained from the stereo camera. Lighter the pixel value, closer is the object from our vehicle. In the previous module, we detected the objects in our environment and found their bounding boxes. The median depth of each bounding box gives us the depth of the object belonging to that box.

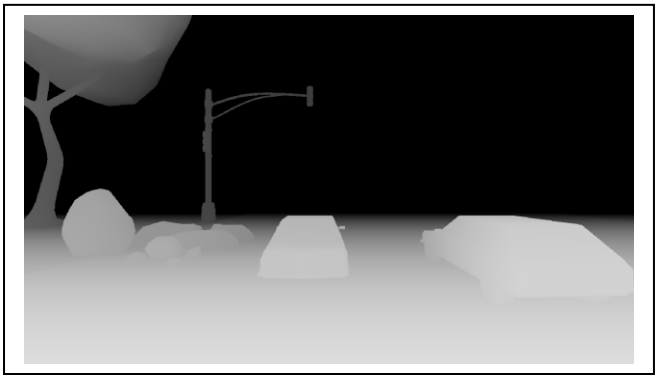

Fig. 5. Disparity Map obtained from the front-facing stereo camera

B. Coordinate Transformation

1) 3D Reprojection: In this module, we calculate 3D cartesian coordinates of all the other objects relative to us. Using stereo-vision, we have calculated the depth of each object in our view. The depth of an object is its z coordinate if the reference point (our car) is located at z=0. Similarly, we have already calculated x, y coordinates of each object. Combining them, we get x, y, z cartesian coordinates of all the objects visible to us.

2) Frenet Coordinate System: Frenet coordinates include two parameters, s and d. The first parameter, s is the distance of objects from our car along the road. In case the road is curved, s helps to give us a better idea of how far an object is located from us. The second parameter, d is the distance of each object from the divider of the road. d helps determine the horizontal position of objects on the road and also the lane in which objects are present.

In the previous module, we found the equation of cubic spline r representing our current road. To find s, we iteratively calculate point at r(s) such that its Euclidean distance from the object is minimum. This Euclidean distance is d and the line connecting the point at r(s) and the object gives us the normal of the curve.

C. Trajectory Planning

Trajectory planning deals with finding a safe, comfortable and efficient path to maneuver around the traffic [12-13]. So far, we have calculated the Frenet coordinates of all the objects (cars, obstacles and traffic signals) and also the generated a cubic spline representing our current road segment. Using Frenet coordinates we determine the lane in which other objects are present. We would need to change our lane if a slower vehicle is present in front of us or if we need to make a turn in the near future. For each frame, we calculate several candidate trajectories (based on whether we need to change lanes or not). We select the trajectory that offers the minimum jerk as our optimal trajectory [14] (as shown in Fig. 6). This selected trajectory is in the form of cubic spline equation.

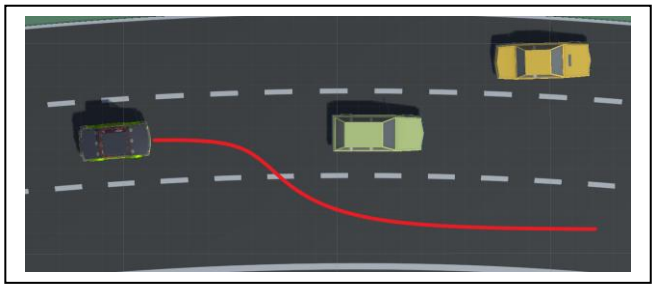

Fig. 6. Optimal trajectory while changing lanes

D. Control

In this module, we design a control system that will help to move our car along the planned trajectory. Driving along a planned path is a very intuitive task for humans, however, while designing the system, we must also consider the consequences of our current actions in the near future. For example, if we don't slow down before a turn, we may crash while making the turn.

Our self-driving car is front steering, gearless, and with acceleration (throttle) and steering angle as the actuators. Paper [15] verifies the usage of model predictive controller (MPC) as a robust approach for active steering control. For implementation of MPC, we use the kinematical bicycle model as presented in [16]. The model is optimized by minimizing costs such as CTE (cross track error), variation from reference velocity, and sudden changes in acceleration and steering angle.

III. RESULTS

Fig. 7 illustrates detection of traffic cars, their coordinates, and also the lanes in our current road segment are detected.

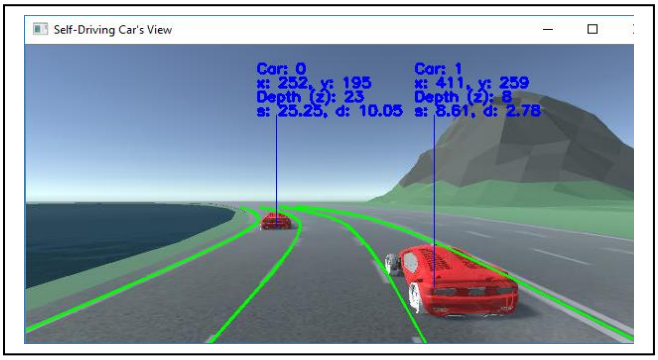

Fig. 7. Final integrated result

We analyzed performance with respect to various parameters between a manually driven car and our self-driving car. We ran the simulation 100 times with our implementation and also drove the car manually 100 times (equally divided amongst 4 people).

The parameters for testing include average completion time, cross-track error (CTE), deviation from reference velocity which we have set as 80km/hr, traffic violations which include collision with other vehicles in the environment and not following traffic protocols and also having jerk as minimum as possible for improving our car's performance. The results of the analysis are tabulated below:

TABLE I. PERFORMANCE ANALYSIS (SELF-DRIVING CAR VS HUMAN)

Parameters tested	Self-driving car	Manually controlled car
Average time of completion	193.37 s	221.64 s
Average CTE	0.09 m	0.38 m
Standard deviation from reference velocity (80km/hr)	1.5342 m/s	4.1727 m/s
Traffic violations	2	17
Maximum jerk	1.56 m/s^3	9.18 m/s^3

From the table, it seems clear that our implementation of a self-driving car using Computer Vision outperforms us humans in all the crucial parameters tested.

IV. CONCLUSION

We aimed to provide the proof of concept that self-driving cars can solely rely on vision-based sensors for navigation. The results obtained using our implementation made it clear that our self-driving car has either outclassed or is on-par with the humans in the parameters tested. This can be used as a prototype for future citywide self-driving cars project. Thus, automobile transport will become cheaper, faster and safer.

However, while implementing in the real world, many more parameters will be introduced which will increase the complexity of the system and also affect the performance of the car. Moreover, as we are dealing with images taken from cameras, it is difficult for our car to function optimally in adverse weather conditions.

REFERENCES

[1] B. Li, T. Zhang and T. Xia, "Vehicle detection from 3D lidar using fully Convolutional Network", 2016.

[2] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, et al., "Towards fully autonomous driving: Systems and algorithms", in Intelligent Vehicles Symposium (IV), 2011 IEEE, pages 163–168. IEEE, 2011.

[3] R.W. Wolcott and R.M. Eustice, "Visual localization within lidar maps for automated urban driving", in Intelligent Robots and Systems 2014 IEEE/RSJ International Conference on, pp. 176-183, 2014.

[4] M. Bojarski, D.D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, et al., "End to End learning for self-driving cars", 2016.

[5] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement", 2018.

[6] K.-J. Kim, P.-K. Kim, Y.-S. Chung, D.-H. Choi, "Performance enhancement of YOLOv3 by adding prediction layers with spatial pyramid pooling for vehicle detection", in 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2018.

[7] L. Taylor and G. Nitschke, "Improving deep learning using generic data augmentation", 2017.

[8] Z. Zhong, L. Zheng, G. Kang, S. Li and Y. Yang, "Random erasing data augmentation", 2017.

[9] Y. Ding, Z. Xu, Y. Zhang and K. Sun, "Fast lane detection based on bird's eye view and improved random sample consensus algorithm", Multimedia Tools and Applications, 76(21), 22979–22998, 2016.

[10] Z. Ying and G. Li, "Robust lane marking detection using boundary-based inverse perspective mapping", in 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016.

[11] Y. M. Mustafah, R. Noor, H. Hasbi and A. W. Azma, "Stereo vision images processing for real-time object distance and size measurements", in 2012 International Conference on Computer and Communication Engineering (ICCCE), 2012.

[12] M. Werling, J. Ziegler, S. Kammel and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét frame", in 2010 IEEE International Conference on Robotics and Automation, 2010.

[13] J. Fickenscher, S. Schmidt, F. Hannig, M. Bouzouraa and J. Teich, "Path planning for highly automated driving on embedded GPUs", Journal of Low Power Electronics and Applications, 8(4), 35, 2018.

[14] A. Gasparetto, P. Boscariol, A. Lanzutti and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview mechanisms and machine science", 3–27, 2015.

[15] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng and D. Hrovat, "Predictive active steering control for autonomous vehicle systems", in IEEE Transactions on Control Systems Technology, 15(3), 566–580, 2007.

[16] J. Kong, M. Pfeiffer, G. Schildbach and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design", in 2015 IEEE Intelligent Vehicles Symposium (IV), 2015.