IJCRT.ORG ISSN : 2320-2882
INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

SEMI AUTONOMOUS GROUND VEHICLE NAVIGATION USING DEEP LEARNING

[1]Akshaykumar M. Pardhi, [2]Prakash J. Kulkarni,
[1]M.Tech student, [2]Professor,
Department of Computer Science and Engineering, Walchand College of Engineering
(An Autonomous Institute), Sangli, Maharashtra.

Abstract: The automobile industry has put a lot of effort and innovative work over recent years which has made the automobile industry one of the most exciting engineering disciplines. Self-driving Vehicles are constantly a piece of Sci-fi films and are considered as the eventual fate of the automobile industry. Path-keeping, adaptive cruise control, and automatic braking are the semi-autonomous functions introduced in the market for modern vehicles. The aim of this project is to design and simulate autonomous ground vehicle which is able to move autonomously while detecting and avoiding obstacles with the help of deep learning. Autonomous vehicle development and machine learning both are using deep learning as the most important frontier. This idea can be extended to a wide array of applications such as transportation in malls, hotels or schools. The training is conducted using the CARLA(Car Learning to Act) simulator annually and it achieves good results on recognizing lane lines on road and traffic signs. The goal of the work is to prepare a convolutional neural network for control directions via delineating pixels from a single front-facing camera which is looking straightforwardly. This approach is demonstrated in many autonomous vehicles. The framework explains here how the system learns and what are the different decisions it takes while driving on highways, in heavy traffic on nearby roads, and also on roads with or without lane markings with less training data collected from peoples. It also gives good results in areas with an unclear visual condition, such as in parking areas, and on unpaved streets
***Index Terms* - Autonomous Ground Vehicle (AGV), CARLA simulator, Convolutional Neural Network (CNN), Artificial Intelligence (AI), Deep Learning (DL), Advanced Driving Assistance System (ADAS), Frames Per Second (FPS).**

I. INTRODUCTION

A vehicle that can accelerate, decelerate, stop, steer and switch to another lane with or without human interaction is called semi-autonomous vehicle. Semi-autonomous ground vehicles can keep in the path and even switch to another path, and they may likewise have the option to park without the intervention of driver but, they are semi-autonomous. While driving any vehicle drivers consistently keep their hands on the steering wheel. The main aim of the system is a cooperation between the assistance system and the driver. Therefore, both can apply a torque on the steering wheel [1]. Human errors are basic clarification behind destructive driving disasters. 75% of car crashes are identified with disappointments [1]. A steering control system gives an incredible perspective to improve security in an individual vehicle, such as a collision-control system and lane-keeping system [1]. Self-sufficient and semi-autonomous vehicles have a tremendous perspective for increasing the productivity of street transportation frameworks by means of safe increase in traffic driving, minimizing CO2 emissions, and energy waste [4].
An essential research domain for these vehicles is the plan of powerful and computationally possible control systems that assure collision-free direction for self-driving vehicles under the limitation of knowing street limits, different items, and guidelines which are given for traffic [4]. Human factor remains the most widely recognized reason for street mortality. To be sure, the driver takes possibly risky decisions that may be purposeful (such as fast driving) that can be the consequence of physically exhausted, sleepiness, low recognition and translation of visual scenes. Semi-autonomous ground vehicle functions will reduce these types of mistakes or even cause them to vanish [2]. Motivated by the accomplishment of detection and classification techniques, in various domains, in light of deep learning, machine learning also utilizes these new signs of progress for traffic signs recognition. For this, the inquires about are moving towards the improvement of recognition frameworks that are increasingly adjusted to genuine pictures of street signs that don't, for the most part, resemble their models [3].
Autonomous vehicles can get into a wide range of circumstances on the road. On the off chance that drivers will rely on their lives to autonomous vehicles, they should be certain that these vehicles will be prepared for any type of circumstances. Likewise, an autonomous vehicle ought to respond to these circumstances superior to anything a human driver would. The vehicle cannot be constrained to dealing with a couple of essential situations. A vehicle needs to train and learn how to adjust to the regularly changing conduct of different vehicles which are moving around it. Any autonomous vehicle is capable of making decisions in real-time due to machine learning algorithms. Therefore, safety and trust in autonomous cars are increasing [6].
The system automatically learns inside a representation of the crucial steps which are required for handling a vehicle, such as, identifying valuable lane marking with the help of drivers steering angle as the input signal. The framework not ever constructs it to differentiate the layout of lanes. As compared to the in detail decomposition of the issue, for example, path-keeping recognition, lane managing, and control, our end-to-end system increases all handling advances at a given time. The system contends that this will, in the end, lead to better execution and small systems [8].

II. LITERATURE SURVEY

The beginning of driverless vehicles was introduced as right on time as the mid-1920s. In the city of New York in 1926, "Iinrrican Wonder" a radio-controlled Chandler was illustrated. It was the first real open showcase of a driverless vehicle in the US. Fatin Zaklouta and Bogdan Stanciulescu have proposed to work in real-time traffic sign recognition using different tree classifiers [1]. The result of random forests yields the highest classification accuracy of up to 85%. The limitation of the work is that the traffic signs with similar symbols look-alike sometimes [1]. An Overview of traffic signs recognition methods is proposed by P. Dewan et.al which gives a result of autonomous cars following a forward vehicle and making a safe and precise detour, if necessary. Advanced Driving Assistance System (ADAS) system is a smart system that is designed to improve the comfort and safety of drivers [2]. M. Flad, L. Frhlich, and S. Hohmann have proposed a framework which is mostly focused on semi-autonomous driving [3]. It is the system that supports the human driver in only steering activity. Therefore both systems and drivers both apply the torque on steering [3]. Semi-autonomous vehicle predictive guidance and vehicle control in public traffic by Thomas Weiskircher, Qian Wang, and Beshah Ayalew shows the framework, that enables the safe operation of autonomous and semiautonomous vehicles [4]. In this paper, they focused on two modes; first one is tracking mode and the second one is planning mode [4]. Another paper prepared a study that helps the system to recognize the brake lights and recognize likely collision of vehicles [5]. In this paper they are just recognizing brake lights and detect the vehicles [5]. In densely populated urban environments navigation is the most difficult task [9]. Research in autonomous urban driving is most difficult by infrastructure costs and the logistical difficulties of training and testing systems in the physical world [9]. The last decade shows a growth evolution in the development of intelligent transportation systems (ITS). And it is especially in the Advanced Driver Assistance System (ADAS) and Autonomous vehicles play a major role in growth evolution in intelligent transportation systems [10].
From the literature review, it is found that in the earlier study there is no focus on sensorimotor control in 3D- environment. Sensorimotor control in 3D-environment is a most difficult task in AI, ML, and robotics. Another most difficult task is navigation in a highly crowdie urban area. Training and validation of system in physical world is the toughest task in autonomous driving research. Solution for given problem is an alternative to train, test, and validate all the driving strategies in simulation.

III. METHODOLOGY

In this project, the simulation is carried out by the CARLA simulator. For autonomous driving research all the activities are perform in CARLA simulator which is an open-source simulator. CARLA simulator have many features such as scalability via a server multi-client architecture, flexible APIs, autonomous driving sensor suit, fast simulation for planning and control, and maps generation. Rendering of all the actors such as cars, pedestrians, building, and sensors in the scene are all handle by simulator. Most of the heavywork such as controlling of the logic and physics of actors is also done by the simulator. CARLA simulator provides a python API module which include python scripts that provides an intermediate communication for retrieving data and controlling the simulator. The framework is basically working like a client server application. In which simulator is working as server that is waiting for a client to connect and communicate with the world which is provided by that simulator. There are a few core concepts that are needed to introduce first.

Fig.1 Spectator view of CARLA simulator

a. **Actor:** This is the entity that plays a major role in the simulation and can be moved around, examples of actors are sensors, vehicles, and pedestrians.
b. **Blueprint:** Blueprints are the attributes of actors which we need to specify before spawning an actor. In CARLA simulator they provide a blueprint library with the definitions of all the actors available.
c. **World:** When the server shows a map which is represented in the simulator and contains different functions for converting a blueprint into a living actor, among others.

CARLA allows for a scalable configuration of the actor's sensor suite. At the time of writing, sensors are limited to RGB cameras and to pseudo-sensors that provide ground-truth depth and semantic segmentation. The sensing modalities provided by the simulator are shown in Fig.2.

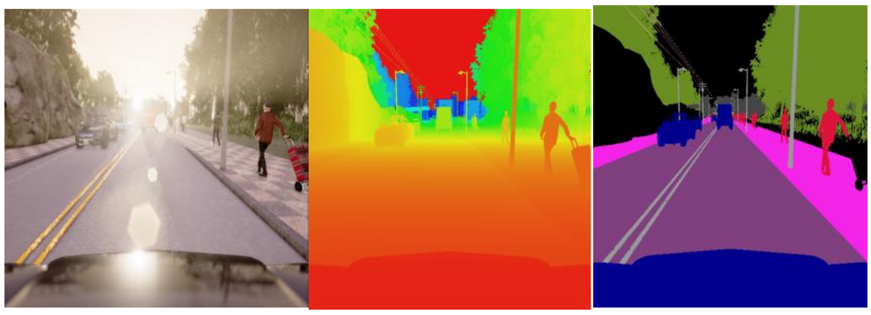


Fig.2 Three of the sensing qualities provided by CARLA

A block diagram of our training framework is shown in Figure 3. Images are given to CNN which at that point figures out different actions of agent.

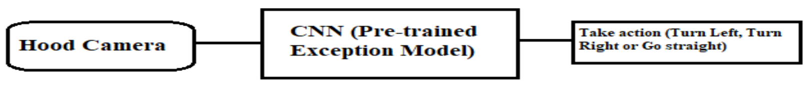

Hood Camera
CNN (Pre-trained Exception Model)
Take action (Turn Left, Turn Right or Go straight)

Fig.3 The trained CNN network is used to take action from a single-center camera

In an information-gathering vehicle a forward-looking camera combined with the time-synchronized steering angle recorded from a human driver.

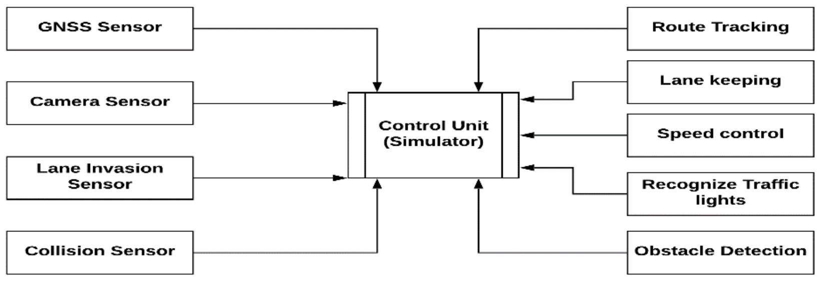


Fig.4 Block diagram of system

Five core components of an autonomous vehicle are as follows:

a. **Computer vision:** The technology of machines, computer systems, robots, and AI for analyzing images, recognizing objects, and act accordingly is known as computer vision [12]. In any case, the manner in which a vehicle sees unmistakable information is very unique in relation to the manner in which a person sees it. AI frameworks do not perceive questions as individuals do. [12].
b. **Sensor fusion**: It is technique of merging data gathered from many sensors in which it reduces the amount of uncertainty that may be included in a vehicle navigation or different work performing [14].
c. **Localization:** Localization is the usage of mathematical calculations for assessment of any robot or vehicle with an error of not more than 10 cm. Localization is a stage executed in most of robots and vehicles to situate with a really small margin of error [16].
d. **Path Planning:** From point A to point B locate the shortest path which is most reliable, most secure, and economically beneficial are find out using path planning and decision making in autonomous vehicles [14].
e. **Control:** Controllers map the association in reality in terms of forces, and vitality, while the cognitive navigation and planning algorithms in an autonomous framework are typically worried about the speed and position of the vehicle concerning its condition[14, 16].

Above are the core components used in autonomous vehicle navigation. The working of all components in simulation is as follows:

Carla simulator acts as a server. Carla runs the simulation and renders the scenes. For connecting this server, create a client and connect via socket. The simulator simulates a dynamic map of urban environment and provides a simple intermediate communication between an actor and the world in which actor connects with the world via sockets. The python API is used in a client application which is implemented in python language those are responsible for interaction between any actor created in CARLA and the simulator which is act as a server that are connects via sockets. Sensor readings are return from the simulator when client sends commands and meta-commands to the server. Steering control, throttle, braking, and vehicle is control via commands. Behavior control of the simulator is handle by the meta-commands and this are also used for resetting the server, modifying the sensor suit, and changing the properties of the environment. Weather conditions, density of cars and pedestrians, and illumination are the different environmental properties consider during simulation of the scene. At the time of resetting the server, the objects are re-initialized at a random position specified by the client.

3.1 **Implementation phase in CARLA simulator**

CARLA consists mainly of two modules, the CARLA Simulator and the CARLA Python API module. The simulator does most of the heavy work, controls the logic, physics, and rendering of all the actors and sensors in the scene; it requires a machine with a dedicated GPU to run. The CARLA Python API is a module that you can import into your Python scripts, it provides an interface for controlling the simulator and retrieving data. With this Python API you can, for instance, control any vehicle in the simulation, attach sensors to it, and read back the data these sensors generate. There are some essential step in implementation phase which are shown as follows:

- **Find CARLA egg file**
- **Establish a connection between client and server**
- **Cleaning of actors**

In this framework, four sensors are used for sensor fusion. This sensor are the crucial part of any autonomous vehicle. Sensors help vehicle to take good decision at risky situation. There are some sensors which gives information of latitude and longitude of vehicle in a given map. Many autonomous vehicle are used different type of sensors for fast and accurate decision. Some of them are given below, which are the crucial part of this framework.

1. **Collision sensor:** This sensor, when attached to an actor, it registers an event each time the actor collisions against something in the world. This sensor does not have any configurable attribute.
2. **Lane invasion sensor**: This sensor, when attached to an actor, it registers an event each time the actor crosses a lane marking. This sensor is somehow special as it works fully on the client-side. The lane invasion uses the road data of the active map to determine whether a vehicle is invading another lane. This information is based on the OpenDrive file provided by the map, therefore it is subject to the fidelity of the OpenDrive description. In some places there might be discrepancies between the lanes visible by the cameras and the lanes registered by this sensor.
3. **Global Navigation Satellite System (GNSS) sensor:** This sensor, when attached to an actor, reports its current gnss position. The gnss position is internally calculated by adding the metric position to an initial geo reference location defined within the OpenDRIVE map definition.
4. **Camera sensor**: The ”RGB” camera acts as a regular camera capturing images from the scene.

This above sensors are attached to our agent (CAR). And then spawn this agent in a simulator environment. By using reinforcement learning the agent take its decision to navigate safe and as much as fast in the simulation environment. Above sensors are help a lot to this agent to navigate fast and accurate in simulation.

3.2 Reinforcement learning

Reinforcement Learning is a type of machine learning where an agent learns to behave in a environment by performing actions and seeing the results. Reinforcement learning plays a major role in autonomous driving research. In a CARLA environment, they provide many sensors such as collision sensors, lane invasion sensors, and Cameras. These types of sensors are attached to our agent i.e. vehicle and then spawn in a CARLA environment [8]. After so many episodes agent is a good train from its penalties in a 3D environment. Here the Association Reward Penalty (ARP) algorithm is used to train our agent. This framework is designed for semi-autonomous ground vehicle navigation, hence the manual control is also an important part of system. The main goal of the project is sensorimotor control in 3D-environment and train our agent with the sensory data generated from these sensors [9].

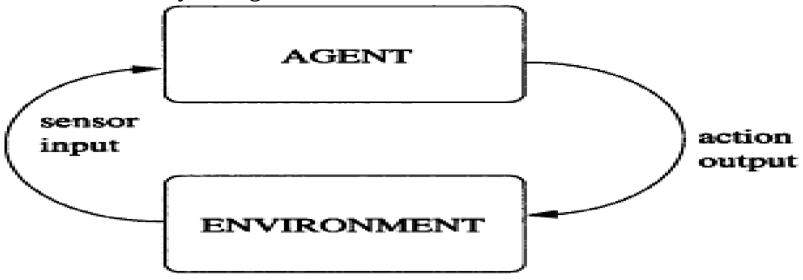


Fig.5 Reinforcement learning model

3.3 Working of project

After implementation phase, to begin the working of project, let us just make a vehicle (car) that spawns in environment and simply drives forward and then see the information from a regular RGB camera that has been placed on the hood of the car. Next step is to, recall the 3 major ”things” in Carla: World, blueprint, and actors. After completing the connection with our server, get the world, and then access the blueprints. Now, after getting the blueprint, spawn this vehicle at any random point. Carla comes with something like 200 spawn points, so just pick one of those randomly. Next step is to put a camera on our vehicle, and to figure out how to access that data. This hood camera would ideally be our main sensor. Other sensors are also important, but a hood camera seems to be a good start. Before proceed further, add this camera sensor to our vehicle. Then, adjust the sensor from a relative position, after that attach this to our vehicle. Finally, the camera sensor gave imagery data from it, so that data is very essential for vehicle to take further decision in difficult situation. To do something with the data which is taken from the RGB camera sensor, lambda function is used. In this case, after getting the data from the RGB camera sensor, and pass it through some function i.e., processed image. This should pop up a new window to display the camera sensor.

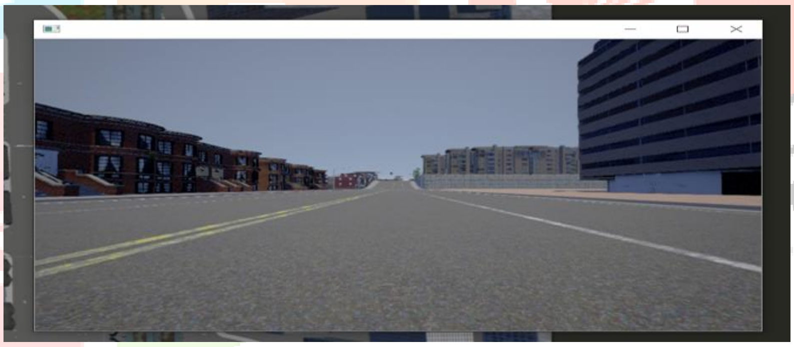
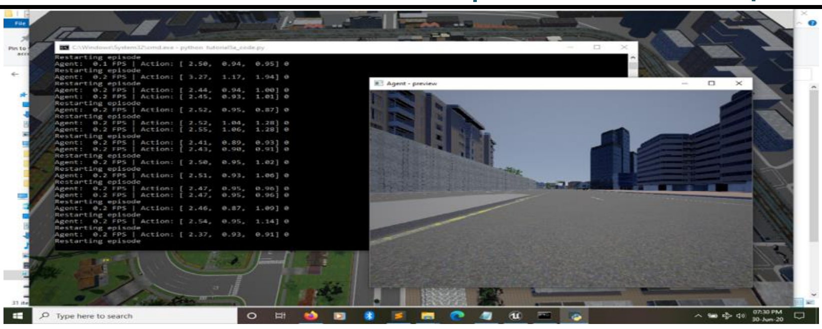


Fig.6 Window to display the working of camera sensor

Now, next step is to take the Carla API, and try to convert this problem to a reinforcement learning problem. The idea here is that our environment will have a step method, which returns: observation, reward, done, extra info, as well as a reset method that will restart the environment based on some sort of done flag. Next, step is to create environment’s class, which is called as CarEnv. Further need to do our step method. This method takes an action, and then returns the observation, reward, done, any extra info as per the usual reinforcement learning paradigm. Everyone know that every step an agent takes comes not only with a plausible prediction (depending on exploration/epsilon), but also with a fitment! This means that training and predicting at the same time, but it’s pretty essential that our agent gets the most FPS (frames per second) possible. In this paradigm, our trainer to go as quick as possible. Hence, to achieve this, both techniques can be use either multiprocessing, or threading. Threading allows us to keep things still fairly simple. Now, next step is to create a new class, i.e. the DQNAgent class. This is the same concept as like the reinforcement learning, where the main network, which is constantly evolving, and then the target network, which will be update every n things, where n is whatever you want and things is something like steps or episodes. Now let’s create our model. Here, I am going to make use of the premade Xception model. Now, only 2 more methods are remaining and here complete with our agent class. First method is to get q values (basically to make a prediction) and the next is to actually do training. Finally got our environment and agent, after that add a bit more logic to tie these together. Next step is to create the modified tensorboard class. Normally, there is a log file per fitment, and a data point per step, which becomes very absurd, very quickly, with reinforcement learning (where you fit every step!). First, set some FPS value (frames per second). When the training is start, the client selects randomly, rather than predicting it with our neural network. A random choice is going to be much faster than a predict operation, so here can arbitrarily delay this by setting some sort of general FPS. You should set this to be whatever your actual FPS is when epsilon is 0. Here set random seeds for repeatable results, then specify GPU memory fraction. Next making the models directory if it doesn’t exist yet, this is where our models will go. Then create our agent and environment class. Finally, if agent actually iterated through all of our target episodes, then it can exit. For example, here’s my agent in action is shown in below figure 7.



```
C:\Windows\System32\cmd.exe - python tutorial5a_code.py
Restarting episode
Agent:  0.1 FPS | Action: [ 2.50,  0.94,  0.95] 0
Restarting episode
Agent:  0.2 FPS | Action: [ 3.27,  1.17,  1.94] 0
Restarting episode
Agent:  0.2 FPS | Action: [ 2.44,  0.94,  1.00] 0
Agent:  0.2 FPS | Action: [ 2.45,  0.93,  1.01] 0
Restarting episode
Agent:  0.2 FPS | Action: [ 2.52,  0.95,  0.87] 0
Restarting episode
Agent:  0.2 FPS | Action: [ 2.52,  1.04,  1.28] 0
Agent:  0.2 FPS | Action: [ 2.55,  1.06,  1.28] 0
Restarting episode
Agent:  0.2 FPS | Action: [ 2.41,  0.89,  0.93] 0
Agent:  0.2 FPS | Action: [ 2.43,  0.90,  0.91] 0
Restarting episode
Agent:  0.2 FPS | Action: [ 2.50,  0.95,  1.02] 0
Restarting episode
Agent:  0.2 FPS | Action: [ 2.51,  0.93,  1.06] 0
Restarting episode
Agent:  0.2 FPS | Action: [ 2.47,  0.95,  0.96] 0
Agent:  0.2 FPS | Action: [ 2.47,  0.95,  0.96] 0
Restarting episode
Agent:  0.2 FPS | Action: [ 2.46,  0.87,  1.09] 0
Restarting episode
Agent:  0.2 FPS | Action: [ 2.54,  0.95,  1.14] 0
Restarting episode
Agent:  0.2 FPS | Action: [ 2.37,  0.93,  0.91] 0
Restarting episode
```

Fig.7 Agent in action

IV. RESULTS AND ANALYSIS

The agent could take one of three total actions: Turn Left, Turn Right, Go straight. In this framework FPS (Frames Per Second) is most important. When it comes to something like driving a car, the higher FPS is better to take fast decision in difficult situation. The project aimed to average around 15FPS, controlling it by how many agents we decided to spawn. But in this project only one agent is spawn at different locations. The max reward over time does trend up until 100,000 episodes, and may have continued back up given more time.

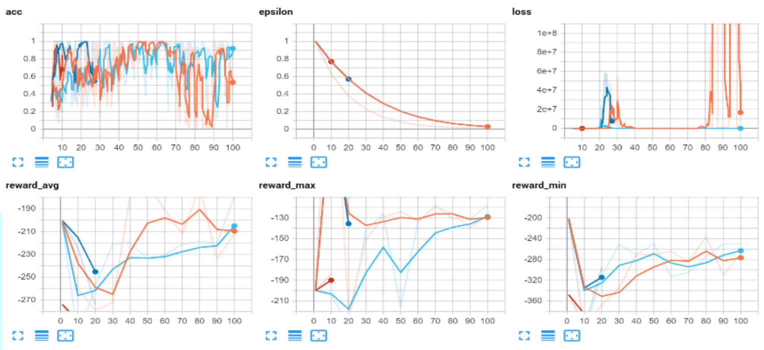


Fig.8 Results and analysis

The minimum reward (ie: the worst an agent did), did not seem to change trend-wise, which is not too shocking. Sometimes the car is just dropped into an unfortunate setting. Finally, the reward average improved pretty consistently. In the end, it really may just be the case that more time is required. A DQN can be extremely good at learning things, just not quickly it takes some time to get its best. I would like to keep training the agent for a while to see if it continues to improve or not, but I had to eventually release this update, because this agent that spawn in environment take much more time for training.In the below figure there are some results and analysis related to our agent are shown on tensorboard.

V. CONCLUSION

Design features of low-cost semi-autonomous ground vehicle using deep learning have been discussed in this paper. Semi-Autonomous Ground Vehicle could be an alternative to various current transportation facilities, however, making it more reliable and safe to all people is a matter of great concern. Different design methods for recognizing the patterns in input video stream by which DL makes its steering decisions, i.e., salient objects have also been discussed.

References

[1] F. Zaklouta and B. Stanciulescu, Real-Time Traffic-Sign Recognition Using Tree Classifiers, IEEE Transactions On Intelligent Transportation Systems, Vol. 13, N. 4, December 2012, p. 1507-1514.

[2] P. Dewan, R. Vig, N. Shukla, and B. K. Das, An Overview of Traffic Signs Recognition Methods, International Journal of Computer Applications, Vol. 168 N..11, June 2017.

[3] M. Flad, L. Frhlich, and S. Hohmann, Cooperative shared control driver assistance systems based on motion primitives and differential games, IEEE Trans. Human-Mach. Syst., vol. 47, no. 5, pp. 711722, Oct. 2017.

[4] Thomas Weiskircher, Qian Wang, and Beshah Ayalew, Member, IEEE, Predictive Guidance and Control Framework for (Semi-) Autonomous Vehicles in Public Traffic, IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, VOL. 25, NO. 6, NOVEMBER 2017.

[5] Jian-Gang Wang, senior member, IEEE, Lubing Zhou, Yu Pan, Serin Lee, senior member, IEEE Zhiwei Song, senior member, IEEE, Boon Siew Han and Vincensius Billy Saputra, Appearance-Based Brake-Lights Recognition Using Deep Learning and Vehicle Detection*, 2016 IEEE Intelligent Vehicles Symposium (IV) Gothenburg, Sweden, June 19-22, 2016.

[6] Djebbara Yasmina, Rebai Karima, and Azouaoui Ouahiba, "Traffic signs recognition with deep learning",2018 International Conference on Applied Smart Systems (ICASS'2018)24-25 November 2018, Mda, ALGERIA.

[7] Mariusz Bojarski, Philip Yeres, Anna Choromanaska, Krzysztof Choromanski and Bernhard Firner, Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car, Computer Vision and Pattern Recognition, Submitted on 25 Apr 2017.

[8] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Pra-soon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. "End to end learning for self-driving cars", April 25 2016. URL: http://arxiv.org/abs/1604.07316, arXiv:arXiv:1604.07316.

[9] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez and Vladlen Koltun, "CARLA: An Open Urban Driving Simulator",1st Conference on Robot Learning (CoRL 2017), Mountain View, United States.

[10] Gautam R Gare, "Autonomous Vehicle Technology A brief overview of the technology and current trends in Autonomous Systems",International Journal of Modern Trends in Engineering and Research (IJMTER)Volume 03, Issue 05, [May 2016]

[11] Yourdictionary.com

https://www.yourdictionary.com/semiautonomous-vehicle

[12] Ittelias.com

https://www.intellias.com/how-machine-learning-algorithms-make-self-driving-cars-a-reality/

[13] popularmechanics.com
https://www.popularmechanics.com/cars/cartechnology/a27269684/self-driving-cars/

[14] www.researchgate.net
https://www.researchgate.net/publication/316270100Computer Vision for Autonomous Vehicles Prob

[15] www.wikipedia.org
https://en.wikipedia.org/wiki/Sensor fusion

[16] www.towardsdatascience.com
https://towardsdatascience.com/self-driving-car-localization-f800d4d8da49