Nat. Volatiles & Essent. Oils, 2021; 8(5): 3269-3281

Tracking and Estimation of Ego - Vehicle

[1]M. Saranya, [2]N. Archana, [3]B. Arundhathi,

[1]Assistant Professor, Department of Instrumentation and Control Systems Engineering, PSG College of Technology, Coimbatore, TamilNadu - 641004, India.
E-mail: msa.ice@psgtech.ac.in
[2]Assistant Professor (Sr. Gr.), Department of Electrical & Electronics Engineering, PSG College of Technology, Coimbatore, TamilNadu - 641004, India.
E-mail: naa.eee@psgtech.ac.in
[3]PG Scholar, Department of Instrumentation and Control Systems Engineering, PSG College of Technology, Coimbatore, TamilNadu - 641004, India.
E-mail: arubalakrish@gmail.com

Abstract

In future, Autonomous vehicles will play a significant part in the urban transportation networks, since they increase safety and productivity with expanded accessibility to improve road efficiency, and provide a favorable environmental impact. Major goal of this research is to use edge computing techniques to locate and track automobiles in real-time circumstances. The Extended Kalman Filtering algorithm is used to perform the localisation. The latency has been minimised because to the use of edge computing. The ultimate goal of designing a self-driving cars using edge computing system is to provide more computing power, with reduction in redundancy, and more security. Motion planning methods include searching for a path to follow, avoiding obstacles, and determining the optimal trajectory path that ensures safety and efficiency when conveying persons or commodities from an origin to a destination.

Keywords*- Autonomous vehicles, Self-driving, Ego Vehicle, Edge computing.*

I. INTRODUCTION

Edge computing is a kind of distributed computing concept which moves data storage closer to the point of use with high computation capability. It reduces reaction times while while conserving bandwidth. It revolutionizes the data handling, processing, and transmit powerof billions of devices all over the world. Edge computing systems are driven by sudden expansion of internet-enabled devices, along with novel application which requires real-time processing power.

Application such as self-driving cars and robots require image processing, video processing, data analytics, and artificial intelligence. These application benefits a lot from edge computing platforms.

Edge computing enables real-time data, does not suffer from latency concerns. Also it helps us to reduce the amount of data that must be stored and processed in a cloud-based location. Thereby local processing helps in saving power and money.

A. Smart Vehicle

B. The term "smart car" refers to a vehicle that is capable of calculating, storing data, and making decisions based on environmental factors. The EGO vehicle is classified as a smart car. Instead of the egovehicle driver, the automated driving system (ADS) controls these EGO cars. The ego-vehicle driver, on the other hand, is responsible for the autonomous driving of the ego car at all times. Smart vehicles are utilizes several types of internal (onboard) and external sensors along with multiple interface cards.

C. Statistics of Accidents

Based on the Global Status Report framed on Road Safety during 2018 states that road traffic continues to be a major developmental issue which basically affects public health and leads to death and injury, and surveyed about 1.35 million people worldwide, with 90 percent of these fatalities occurring in developing countries and India accounting for 11 percent of all fatalities. According to the 2018 statistics on road accidents in India, 1, 51,417 people died as a result of them. Despite the government's ongoing efforts, roughly 85 percent of death caused due to accident occurs in the age group of 18 to 60 years old. Road traffic deaths not only cause great emotional distress to the victims' families, but they also cost the country a lot of money.

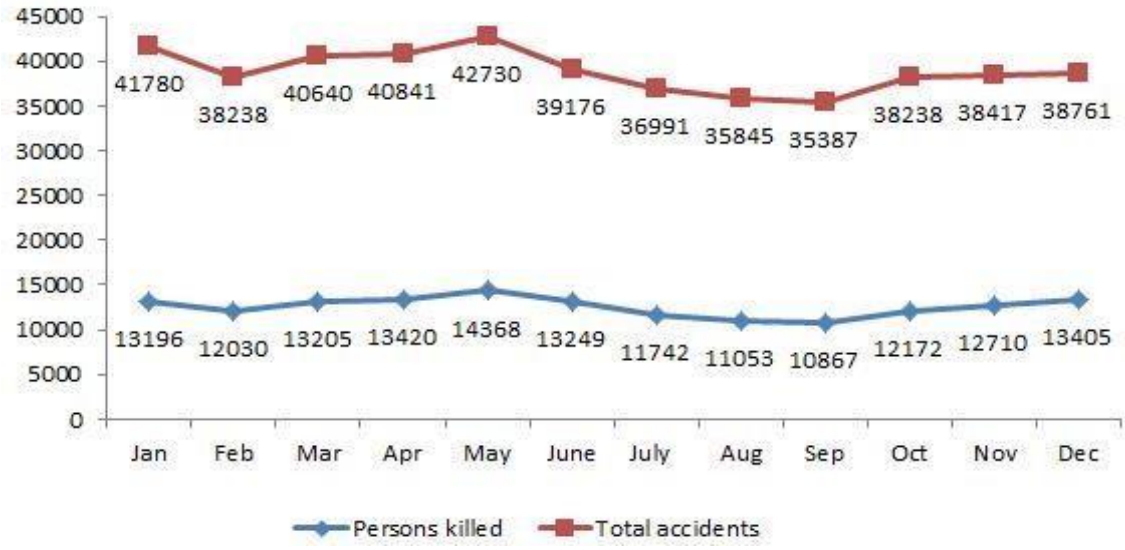

Figure 1.1 Cumulative Accident report in 2018(India)

D. Developing Stages of Smart Vehicles

Vehicle automation is divided into five stages by the NHTSA (National Highway Traffic Safety Administration):

1. *Primary Automation:*

The driver has complete control (in terms of steering, braking, throttle, and motive power) of the vehicle.

2. *Assisted Driving:*

Certain control functionalities are automated, such as ESC (Electronic stability control) & pre-charged brakes.

3. *Partially Self Driving:*
   Lane centering and Adaptive Cruise Control are two main control functions that have been automated.
4. *High Level Self Driving:*
   Under certain environmental conditions, the driver takes over the complete control of every safety-critical activities by relying on the sensors mounted on the vehicle to monitor. The driver will take back the control of the vehicle, whenever they need to do so.
5. *Fully Self Driving:*
   Car will act as intelligent system that is intended to monitor road conditions and drive alone throughout the trip, following safety–critical driving functions. This type is completely driverless level.

E. Network of Smart Vehicles Intra Vehicle Communication:

Automobiles have On Board Units (OBU) embedded with numerous sensors to identify the environmental conditions and interface cards to facilitate the communication within the car.

1. Inter Vehicle Communication: This refers to communication between vehicles or between vehicles and sensors. The variety of applications for this inter-vehicle connection is limited.

2. V2I (vehicle-to-internet) Communication: V2I (vehicle-to-internet) communication takes place between the car and the rest of the world (vehicle to infrastructure).

E. Autonomous Driving Technologies

This technology utilizes many algorithms such as sensing, perception, planning, and control, as well as a vehicular edge subsystem. This subsystem comprises of the hardware platform, operating system and a cloud platform. The cloud platform provides facility of data storage, computational power for simulating HD mapping, and for training using deep learning model. The subsystem pulls valuable data from sensor to gain a better awareness of its surroundings and make dynamic decisions about its actions.

1. *Perception:*
   The feature of an autonomous driving system to perceive the information and extract the required knowledge from it, is referred to as perception. Environmental perception is the process of gaining a contextual awareness of the environment, such as locating barriers, detecting road signs and markings, and categorising data according to its semantic significance. The ability of a robot to detect its position in relation to its surroundings is referred to as localization.

2. *Planning:*
   The steps involved to make a deliberate decisions to achieve the autonomous system's higher-order goals, such as pathway of the vehicle to move from one area to another by avoiding obstacles, is referred to as planning.

3. *Control:*

The ability of the robot to carry out the planned actions generated by higher level processes is referred to as control.

F. Extended Kalman Filter

EKF is a non-linear filter that linearizes nonlinear function around the current mean and covariance, allowing us to estimate the state even in the presence of nonlinear interactions. The transmission of Gaussian random noise across the system dynamics is an important function performed by the Kalman filter. In nonlinear estimating and machine learning applications, the Extended Kalman filter (EKF) is a nonlinear filter. The state of a nonlinear dynamic system is estimated using parameters for nonlinear system identification and dual estimation, are all instances of nonlinear system estimation. The extended Kalman filter is divided into two phases: prediction and updating. To estimate the state, the predict phase applies the state estimate from the previous time step to the current time step. During the update phase, the forecast is modified using measurement data retrieved from the current time step, resulting in a far more accurate state estimate for the same time step.

II. LITERATURE SURVEY

Some of the problem's remedies have already been recognised. A careful evaluation of three existing solutions was undertaken with the support of those existing solutions, and a proposed solution was established. This chapter gives a quick rundown of the various options.

Raza et al. [1] designed a specialized networking architecture called Vehicular Edge Computing to increase the computing capability of the vehicle network (VEC). The ultimate difficulty lies in satisfying the requirements of processing and communication has grown rapidly with introduction of ever-increasing modern vehicular application. Service providers can now host services near smart cars, reducing latency with improving service quality. This paper projects the VEC architecture. In addition to that all technical concerns in the VEC architecture are analyzed in regard to all current and relevant solutions. They also identified current research difficulties as well as uncovering some new ones. This work provides other researchers with a research topic in the field of VEC along with naive readers a good detailed understanding of the current research field.

Liu et al. [2] examine the challenges related to the security in autonomous driving and how edge computing can be utilized in relation to that. The most important aspect related to autonomous vehicles is the security that is achieved along with the provision of enough processing power, redundancy, and security is the ultimate difficulty of developing an edge computing system for them. Autonomous driving cars found to be extremely complex as it combines various range of technologies such as sensing, perception, localization and decision-making, along with the smooth connectivity over the cloud platforms for map building along with high definition data storage.

Alexey Dosovitskiy et al. [3] introduced autonomous driving simulator tool named CARLA, an open-source simulator. CARLA was designed for developing, training and certifying self-driving urban vehicles easier. CARLA also offers open digital assets such as buildings, city layouts and vehicles that were designed particularly for this purpose and are available for free use. The performance of three

autonomous driving systems is tested using this simulator: a typical modular pipeline, an end-to-end model learned using supervised learning, and an end-to-end model defined using reinforcement learning.

Autonomous vehicles play a critical role in the urban transportation systems, according to Ghafoorianfar et al.,[4], since they provide better safety, productivity, accessibility, improved road efficiency, and a positive impact on the environment. As a result of increased available processing power and lower prices in sensing and computing technologies, research in autonomous systems has advanced dramatically in recent years, resulting in a mature technological readiness level of totally driverless cars. The purpose of this study is to provide a comprehensive assessment of current advances in autonomous vehicle software systems.

The new assisted framework has three main features: a drone based camera for acquisition of the image for the overview of the autonomous vehicle's area of subject; (ii) GPU embedded on-board is trained with a model for situational awareness about geo-tagged images and landmarks to define the map path; and (iii) integration of perception with prediction for driving of the autonomous vehicle.

III. SYSTEM DESCRIPTION

A. Autonomous Vehicles

An autonomous vehicle is a ground vehicle that can sense its surroundings, such as other vehicles, pedestrians, and traffic safety guidelines, and navigate itself without human intervention. The autonomous vehicles utilize a large number of sensors to detect their environment using LIDAR, RADAR, GPS, ODOMETRY and ACCELEROMETER. The autonomous vehicle's control system uses sensory data to navigate the roadways, avoid obstacles, and adhere to traffic rules. The concept of automating driving dates back to the 1920s, when autonomous vehicles were dubbed "phantom motor cars."In the 1980s, Carnegie Mellon University's Navlab project developed a self-driving vehicle that used neural networks to analyse raw visuals from the road and control the steering wheel in real time. Some automakers, such as Toyota, BMW, and Ford, began offering automatic parking assistance for their vehicles in 2009. Tesla Model S was released in 2012, and it is widely regarded as the world's first autonomous vehicle. Autonomous vehicles are now being developed by practically all vehicle manufacturers and Internet companies such as Google, Uber, and Baidu.

B. Sensing In Autonomous Vehicles

There are four major possible advantages for self-driving cars. The first and most critical consideration is safety. Autonomous vehicles are safer because human error, delayed reaction time, and distraction will almost entirely eliminate traffic accidents. Furthermore, compared to a human driver, autonomous vehicles have a better vision of the boundary and a faster reaction time, which can make passengers safer. Second, there is the issue of welfare. Traveling in autonomous vehicles is significantly easier than in conventional automobiles. In autonomous vehicles, the driver can perform exactly the same things as the passengers. The third factor is traffic.If all of the vehicles on the road are autonomous, the traffic control department can raise speed limits and close safety gaps, resulting in increased roadway capacity and reduced traffic congestion. Finally, transportation costs such as vehicle insurance and fuel consumption will decrease as a result of safer driving and decreased traffic congestion.

In order to achieve the objectives, the autonomous vehicle must be able to avoid obstacles and collect information about traffic rules. It is also necessary for the autonomous vehicle to detect its

surroundings. For autonomous vehicles to perceive their surroundings, perceptible sensors such as cameras, radar, and laser light have been frequently used.

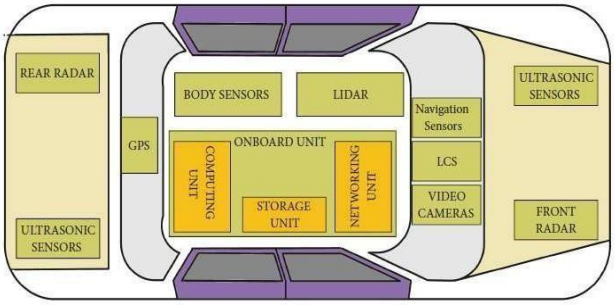

Figure 3.1: Components of Autonomous Vehicles[1].

C. Requirements

Hardware Environment:

This proposal work is carried out on a personal computer that has the following specifications.

- Processor: 3.40 GHz Intel(R) Core (TM) i7-4770
- GPU: Nvidia GTX GeForce TITAN X (4GB), Nvidia GTX GeForce TITAN X (4GB), Nvidia GTX GeForce TITAN
- RAM (Random Access Memory): 8 GB

This project environment is powered by the following software:

- Windows 10 Pro 64-bit operating system

Python is the programming language of choice.

- Tensorflow-gpu, Keras, and OpenCV-Python are included as master libraries.

D. EKF Algorithmbased Localization:

Our ego-initial vehicle's ability to communicate with CARLA is localization, which allows it to figure out where it is in the world. The current Extended Kalman Filter (EKF) localization strategy in autonomous driving software, which takes input from three potential sensors - GPS, IMU, and wheel encoders - to anticipate the vehicle's posture. This pose estimation is provided by the EKF as a coordinate transformation between two coordinate frames: the map frame, which is attached to the vehicle's centre rear axle, and the base link frame, which is set at the vehicle's initial location.Implementation of a GPS sensor based on the Oxford GPS sensor used in Toyota's self-driving Prius. To determine the vehicle's true position in reference to a fixed global frame provided by CARLA. A simulated IMU sensor also provides the vehicle's orientation, angular velocity, and translational acceleration. The ground truth vehicle location determines the node's orientation, and CARLA ego-vehicle measurement data determines its acceleration.

IV. SOFTWARE OVERVIEW

This project depends purely on software simulation and hence the CARLA SIMULATOR version 0.9.9.11 has been used. *Necessary Software:*

- Python 3.7.9
- CMake
- Git
- Make
- Unreal-Engine 4.0
- Visual Studio 2019

A. CARLA *Simulator:*

CARLA is a photorealistic simulator that was created as an open source project to train, validate, and test autonomous driving algorithms. It's written in C++ and uses Unreal Engine for its driving scenarios. It includes digital assets as well as comprehensive control over map actors, environmental conditions, a sensor suite, map production, a customizable API, and server-client communication.

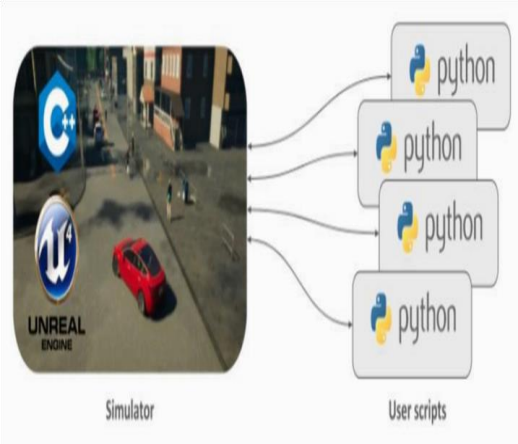

Figure 4.1 CARLA client-server communication.

CARLA includes a Python API module in addition to the CARLA Simulator, which contains all of the control logic, graphics, physics, and actor characteristics. So the Simulator is the server, and the Python API controls the client-server communication.

The Python API provides access to the majority of the simulation's features. Python scripts can be used to receive raw data from CARLA sensors installed to the ego vehicle, parse it, compute all the parameters required by the controller, and provide throttle, braking, and steering inputs to the CARLA Simulator.

B. *Features:*

• Scalability via a server multi-client architecture: distinct actors can be controlled by several clients in the same or separate nodes.

• CARLA's API is extremely flexible, allowing users to manage every aspect of the simulation, including traffic creation, pedestrian behaviour, weather, sensors, and more.

• Sensor suite for autonomous driving: clients can create sensor suites with LIDARs, numerous cameras, depth sensors, and GPS, among other things.

• Fast simulation for planning and control: This option turns off rendering so you can quickly model traffic and road behaviour without having to look at anything. Users can easily create their own maps with Open Drive-compliant tools like Road Runner.

• Traffic simulation: Using our Scenario Runner engine, users may develop and run a wide range of traffic scenarios based on modular behaviour.
• ROS integration: Through our ROS-bridge, CARLA may be integrated with ROS.
• Autonomous Driving baselines: In CARLA, we provide Autonomous Driving baselines in the form of runnable agents, such as an Autoware agent and a Conditional Imitation Learning agent.

C. Carla Architecture:

The central server communicates with one or more clients via a TCP (Transmission Control Protocol) connection in CARLA's client-server architecture. The CARLA server generates vehicle measurement and sensor data by modelling the dynamics of the world, including the ego-vehicle and other actors in the environment. The measurement and sensor data are received by a CARLA client, which can then send control commands to the server to change the state of the ego-vehicle.The CARLA client is in charge of configuring each sensor and initializing the ego-vehicle in the CARLA environment. The CARLA codebase contains the CARLA-ROS bridge, a one-of-a-kind CARLA client that acts as a link between the CARLA server and any ROS-compatible software framework. The CARLA-ROS Bridge receives measurement and sensor data from the CARLA server and publishes it in a ROS-compatible format using the CARLA client libraries. The bridge can also subscribe to ROS control messages and send them to the CARLA server using the CARLA client libraries.

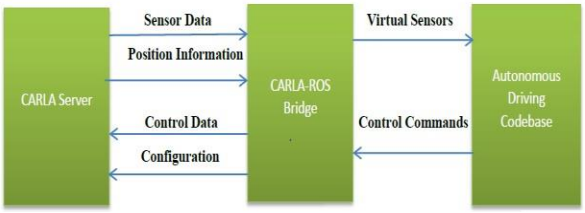

Figure 4.2: Bridge between Autonomous vehicle code base, CARLA server and CARLA-ROS bridge.

D. Carla Sensor Data:

CARLA works with a variety of sensor types, including cameras and multi-channel LIDAR. To produce the whole LIDAR point cloud, the present LIDAR implementation in CARLA generates distinct ray traces for each LIDAR channel at uniform angular increments. This cloud is represented in R3 as an unorganized collection of points in the local coordinate frame of the vehicle.

E. Carla Control:

CARLA enables external control of the ego-vehicle by providing unitless, standardised throttle and steer data to the CARLA server. The CARLA-ROS bridge comprises a module that transforms ROS steer and speed values into CARLA throttle and steer data using proportional gains, because the autonomous driving software outputs ROS commands in the form of steer (rad) and speed (m/s) values.The CARLA vehicle is unable to sustain the necessary velocity over time when a proportional gain is applied to the ROS steer value (a velocity to produce the CARLA throttle value acceleration), and a PID controller gives a more reliable technique to calculate the desired acceleration (as exists in the development version of the CARLA-ROS bridge).

V. RESULTS AND OBSERVATION

The output simulations were created with the use of the Python API and python coding. The Carla server is connected to the Python. Python is used as a client, and the CARLA simulator is used as a server.

A. Run Carla

Figure 5.1 provides information about how the location of the Carla file is given as the input to the command window.

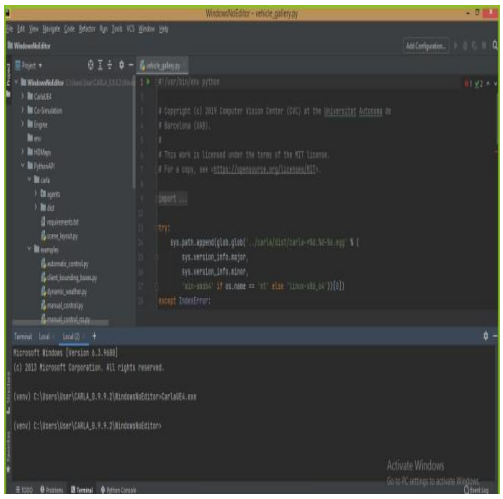

Figure 5.1: How to Run Carla

B. Setup Environment

Figure 5.2 depicts Carla setup environment that has totally 7 towns.

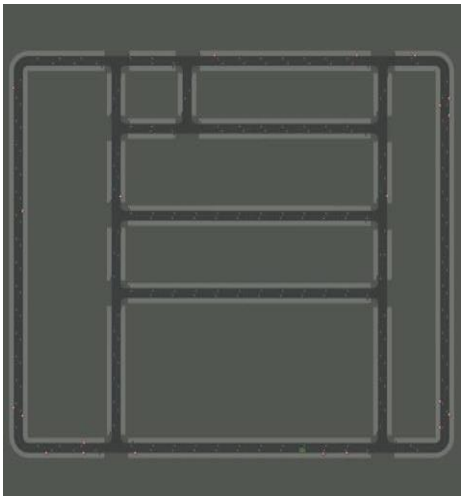

Figure 5.2: Setup Environment

C. Carla Environment

From the figure 5.3(a) and figure 5.3(b) the overall view of the CARLA town 01 can be seen by adjusting the WASD keys and mouse.

D. Spawn Control

In figure 5.4, the code spawn_npc python code is run. The code will be initially present in the python examples folder.

By using this code the 80 vehicles will be spawned in the Carla server.

Figure 5.3(a): View of CARLA Town 01 environment.

Figure 5.3(b): Top view of CARLA town 01

Figure 5.4: Spawning of 80 cars.

5.5 Manual Control

From the figure 5.5(a) and 5.5(b) the manual control using python code is run in the command window. Figure

5.5(c) shows the vehicle will be now controlled by the client. It contains the information of the vehicle like its location, speed, town, etc.

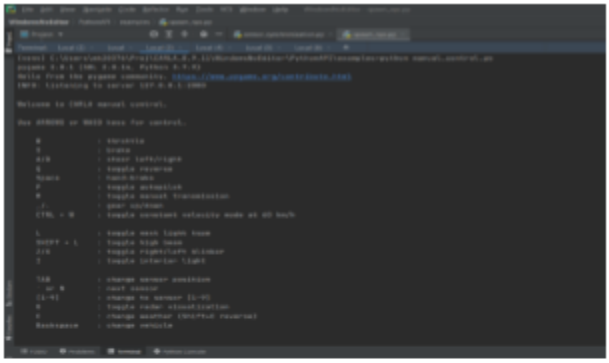

Figure 5.5(a): Manual Control In Command Prompt.

Figure 5.5(b): Manual Control In Command Prompt.

Figure 5.5(c): Manual Control Of Vehicle In CARLA.

5.6 Getting Co-ordinates

Figure 5.7(a) and 5.7(b) shows the change present in the X, Y and Z coordinates. That is occurreddue to the changes in the location of the EGO vehicle.

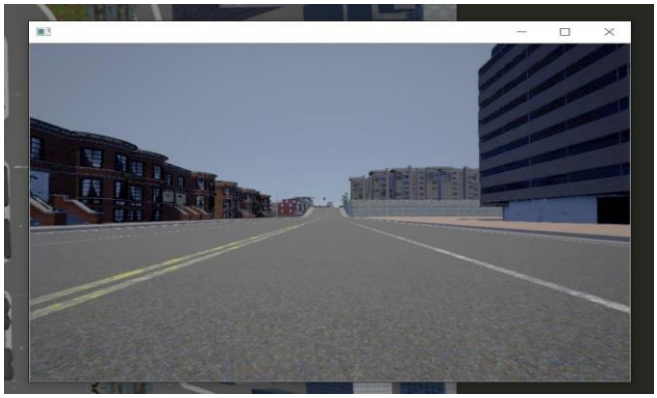

Figure 5.6: Camera setup is done in CARLA Server.

Figure 5.7(a): Getting Coordinates of the vehicles.

Figure 5.7(b): The Coordinates of the vehicles is changed.

VI. CONCLUSION AND FUTURE SCOPE

This paper proposes to identify the position of an EGO vehicle is determined and tracked using an edge computing algorithm and a camera sensor. Many sensors, such as LIDAR and RADAR, can be utilised to obtain a precise position of the vehicles. Various research activities are carried out with respect to Autonomous vehicles (AV) to reduce the transportation costs significantly. The route is still difficult and fraught with obstacles.

There is a security risk in regard to the Autonomous vehicles as an attacker may attackeither one of the sensors or processing systems or control systems or communication network. These attacks on each autonomous vehicle may even take control of the vehicle. As a result, the problem is to ensure security. Hence, the research is carried out on identifying the type of attack surfaces against edge computing ecosystems based autonomous driving to safeguard autonomous systems from these attacks.

REFERENCES

1. S. Raza, S. Wang, M. Ahmed, and M. R. Anwar, “A Survey on Vehicular Edge Computing: Architecture, Applications, Technical Issues, and Future Directions,” Wireless Communications and Mobile Computing, Feb. 24, 2019.
2. S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, “Edge Computing for Autonomous Driving: Opportunities and Challenges,” Proceedings of the IEEE, vol. 107, no. 8, pp. 1697–1716, Aug. 2019.
3. Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez and VladlenKoltun, “CARLA: An Open Urban Driving Simulator,” 1st Conference on Robot Learning (CoRL 2017), Nov. 2017.
4. Ghafoorianfar, Nima, and Mehdi Roopaei. “Environmental Perception in Autonomous Vehicles Using Edge Level Situational Awareness,” 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, 2020, pp. 0444–0448.
5. S. D. Pendleton et al., “Perception, Planning, Control, and Coordination for Autonomous Vehicles,” Machines, vol. 5, no. 1, p. 6, Mar. 2017.
6. Banerjee, Rohan Bandopadhay, “Development of a Simulation-Based Platform for Autonomous Vehicle Algorithm Validation,” Massachusetts Institute of Technology, 2019.
7. Keyan Cao, Yefan Liu, GongjieMeng, and Qimeng Sun, “An Overview on Edge Computing Research,” Special Section on Edge Computing and Networking For Ubiquitous AI, May 6, 2020.
8. Rajasekhar, M. V., &Jaswal, A. K, “Autonomous vehicles: The future of automobiles,” 2015 IEEE International Transportation Electrification Conference (ITEC), 2015.