E-Share: Autonomous Share Public Transit using IoT-enabled AI Technologies

Qaisar Abbas[†]

[†]College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Saudi Arabia

Summary
Autonomous Vehicles in transportation sector is unstoppable. However, the transition from conventional vehicles to Autonomous Vehicles will not happen from one day to the other, instead, it will be a process of several years in which, gradually, new autonomous/automated functionalities will be equipped to the vehicles and introduced to the customers. These automated/autonomous functionalities are, nowadays, known as ADAS (Advanced Driver Assistance Systems). In this paper, through the combination of different ADAS functions, make the vehicle navigate on a highway autonomously, but at the same time, following the traffic rules and regulations requirements, and guaranteeing the safety on the road. To accomplish this objective, the proposed approach is to implement the Policy Gradient Reinforcement Learning method to select the proper function at each moment. The algorithm is tested using a five-lane highway CARLA simulator. E-share aspire to develop vehicles that rely on IoT-enabled artificial intelligence (AI) technologies to deal with self-driving vehicles and their importance to humans and provide the best solutions with easy effort and reduce the financial cost through the development of sensors. And linking them with self-driving vehicles to facilitate transportation. The simulation platform also supports all the sensors needed in smart cities and has full control in all static and dynamic directions in creating maps method for controlling a vehicle upon a roadway includes monitoring a trajectory of a host vehicle based upon a global positioning device within the host vehicle, monitoring a first wireless communication between the host vehicle and a plurality of target vehicles, the first wireless communication including a respective trajectory of each of the target vehicles based upon a respective global positioning device within reach of the target vehicles.

Keywords:
Advanced Driver Assistance Systems, Autonomous Vehicles, CARLA Simulator, Internet of things, Artificial Intelligence

1. Introduction

More than half of the world's population today lives in cities and by 2050, nearly seven out of ten people will live in cities. Cities account for more than 70 percent of the world's carbon emissions and 60 to 80 percent of energy consumption. Rapid urbanization has led to additional challenges such as traffic congestion, frequent road accidents, and related health issues. Governments can use information and communication technologies and other technologies to build smarter and more sustainable cities for their citizens and a smart sustainable city is an innovative city that uses information and communication technology. To improve the quality of life, the efficiency of civilization processes and services, and the ability to compete, while meeting the needs of current and future generations in terms of economic, social, environmental, and cultural aspects, although the cities in which all civil systems and services are connected do not exist yet, many of cities are on their way to becoming sustainable and smart cities. It relies on information and communication technology, for example, to enhance energy efficiency and waste management, improve housing and healthcare, improve traffic flow and safety, detect air quality, alert police to street crimes and improve water and sanitation networks. With the help of the Internet of Things (IoT) to communicate with each other's computer devices and exchange data at high speed, including sensors and software, it enables many devices and objects equipped with smart sensors to communicate with each other, collect information in real-time, and send this data, via wireless communication.

To central control systems. These, in turn, manage traffic, reduce energy use, and improve a wide range of urban processes and services. Artificial intelligence allows extremely large data sets to be analyzed mathematically to reveal patterns that are used to inform and enhance the decision-making process in smart cities. By using the CARLA simulator consists of a scalable client-server architecture. The server is responsible for everything related with the simulation itself: sensor rendering, computation of physics, updates on the world-state and its actors and much more. As it aims for realistic results, the best fit would be running the server with a dedicated GPU, especially when dealing with machine learning. The client side consists of a sum of client modules controlling the logic of actors on scene and setting world conditions. This is achieved by leveraging the CARLA API in (Python), a

Manuscript received June 5, 2022
Manuscript revised June 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.6.90

layer that mediates between server and client that is constantly evolving to provide new functionalities. A visual example is displayed in Figure 1.

Various methodological techniques have been developed to discover and predict the best paths for autonomous vehicles through advanced sensor processing, machine learning, and artificial intelligence algorithms. The purpose of this paper is to first investigate basic smart algorithms that are being applied to discover the best approaches for autonomous and autonomous vehicles. Then, using advanced technologies to accurately detect traffic congestion, choose the appropriate road, and have less time while driving. Although many survey articles have been written to address this issue, according to limited knowledge, none of them have built a hybrid system with cameras and sensors to study the road and provide the best solutions. There is a recent investigative article to detect and predict accidents before they happen, but that study indicated that there are still many improvements required in this area. Accordingly, in this paper, it is developed an integrated e-sharing technology for predicting best methods and incidents before they happen that supports AI and the Internet of Things.

The aim of this paper is to design, implement and test autonomous system in the CARLA simulator to share autonomous vehicles on the same road, calculate the distance between cars, increase safety and reduce congestion through artificial intelligence and augmented learning algorithms for autonomous vehicles, it has been used the Internet of Things - based on technologies, and sensors to detect conditions. Environmental independent. A CARLA simulator is used to study the performance of three autonomous driving methods: a classic modular pipeline, an end-to-end model trained through simulated learning, and a comprehensive model trained by reinforcement learning. The methods are evaluated in controlled scenarios, and their performance is examined through the metrics provided by CARLA, in addition a passenger can request a car.

A self-driving vehicle is developed in this paper that linked with sensors with help of Carla simulator to overcome some of the problems resulting from traffic congestion and avoid accidents to save the lives of others and choose the best way to reduce time and exert effort, and the Carla system provides open-source systems in Building smart cities and sensor systems and linking them to autonomous vehicles, for learning, training, planning, and design, it can be used freely. The simulation platform also supports all the sensors needed in smart cities and has full control in all static and dynamic directions in creating maps, Using the RSS library, implements a mathematical model for safety assurance. It receives sensor information and provides restrictions to the controllers of a vehicle. To sum up, the RSS module uses the sensor data to define situations. A situation describes the state of the ego vehicle with an element of the environment. For each situation, safety checks are made, and a proper response is calculated. The overall response is the result of all the combined. For specific information on the library, you can create very complex choreographies," that is going to help a lot in human-machine interaction, for example, to help improve the field of forecasting. Forecasting is one of the most important modules. You need to be able to tell what a pedestrian or a vehicle is going to be doing in the next five or 10 seconds. you need to be able to predict what that pedestrian is going to do next. Is he going to cross, or is he going to stay on the sidewalk? For that, you need to have very advanced clues.

The RSS sensor implements a mathematical model for safety assurance. It receives sensor information and provides restrictions to the controllers of a vehicle. To sum up, the RSS module uses the sensor data to define situations. For each situation, safety checks are made, and a proper response is calculated. And Pedestrian can using artificial intelligence, request an autonomous vehicle close to his location to transfer him from his current location to another location, with the help of the Internet of Things to locate the nearest available vehicle.

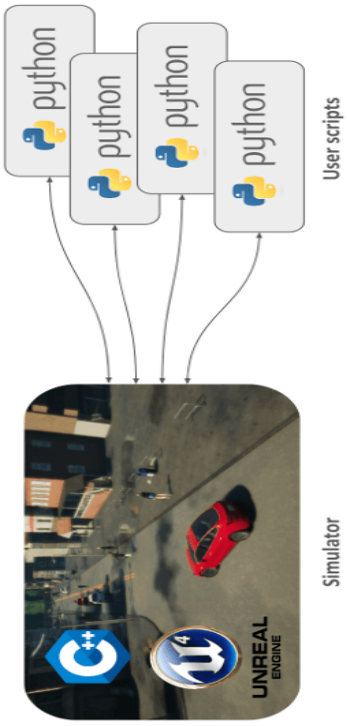

Fig.1. A visual example of CARLA Simulator along with Python tool.

The significance of the research are given below.

1. Training on Carla simulator and studying the understudying technologies
2. To develop a smart simulator, incorporate artificial intelligence (AI) technologies to handle tourism with low-cost solutions by studying autonomous cars.
3. To develop a completely autonomous solutions in a CARLA simulator by E-share autonomous vehicles on the same route It will help to build a smart city by decreasing traffic congestion and money spend to reach to the destination.
4. To build a set of algorithms that can make smart decisions based on machine algorithms for predicting the best and profitable route by the autonomous vehicle. To complete this E-share system.
5. To develop a feature that a car can requested by a passenger to go from location A to location B.
6. To different IoT-based technologies are used to detect autonomous environmental conditions.

The rest of the paper is organized as follows. Section 2 presents the background and some related work. The system design and data set collections are presented in Section 3 and system implementation in Section 4. In Section 5, the proposed system is evaluated and compared with other state-of-the-art applications. Finally, this paper concluded in section 6.

2. Background and Literature Review

The technologies have developed systems with the help of software and hardware technologies to discover the best way around by sharing information with other vehicles and predict accidents before they happen. These systems were based on information processing provided by advanced sensors and machine learning technologies. The information is reviewed using systems linked to the sensors and they study the road and provide the best information using artificial intelligence. Also, another way to spot accidents is by applying an RSS library to a mathematical model to ensure safety. It receives sensor information and provides restrictions for vehicle control units. In short, the RSS unit uses sensor data to identify situations. The position describes the state of the ego vehicle with the environment component. For each case, safety checks are performed, and an appropriate response is calculated. Several studies including recent studies have been thoroughly examined and the characteristics of the proposed methods used for classification along with special classifiers, limitations, and advantages (shortcomings), the accuracy of the methods as well as their experimental preparation.

CARLA is an open-source autonomous driving simulator [1]. It was built from scratch to serve as a modular and flexible API to address a range of tasks involved in the problem of autonomous driving. One of the main goals of CARLA is to help democratize autonomous driving R&D, serving as a tool that can be easily accessed and customized by users. To do so, the simulator must meet the requirements of different use cases within the general problem of driving (e.g., learning driving policies, training perception algorithms, etc.). CARLA is grounded on Unreal Engine to run the simulation and uses the Open DRIVE standard (1.4 as today) to define roads and urban settings. Control over the simulation is granted through an API handled in Python [2] and C++ that is constantly growing as the paper does. Several smart vehicle algorithms have been examined in previous studies to enable prediction before an accident occurs, enabling the construction of future systems. Many traditional autonomous complex systems such as fuzzy logic, statistical models, and decision trees as well as modern deep learning algorithms are widely described in automated systems in advance of predicting an accident before it happens. Therefore, these algorithms are mainly chosen to present and overcome these methodologies. In this paper report, [3] a Responsibility Sensitive Safety (RSS) model was implemented as a demonstrable safety of vehicle behaviors such as a safe minimum tracking distance. However, handling the worst-changed situations and uncertainty may drastically reduce the tolerance of the vehicle, and in some cases, safety cannot be guaranteed. It has been identified complications that result from considering vehicle condition, road engineering, and environmental standards. A particularly difficult situation occurs if these parameters change during a collision avoidance maneuver such as severe braking. As part of analysis, it expands the following distance equation in RSS to calculate edge cases involving potential collisions halfway through the braking process.

The daily routine that people encounter on the roads while on the go has become a problem with every passing day, as people get late and face accidents. [4] The self-driving car model aims to solve these problems by keeping people away from the problems of delay, congestion, and accidents so that they do not have to drive anymore, and the risks of accidents and delays can be reduced, and traffic congestion can be reduced to the lowest possible degree. Others, spotting obstacles, taking sharp bends and turns, tracking traffic.

Some research focuses on the high and tactical [5] level for making decisions related to autonomous vehicles in a complex and dynamic urban environment with PreScan,.[6] PreScan is a simulation environment for developing Advanced Driver Assistance Systems (ADAS)

and Intelligent Vehicle Systems (IV). It is a platform that can be used to build 3D traffic. And the creation of vehicles, pedestrians, traffic lights, and other control units. It also provides an interface for Simulink to allow users to easily control and operate the simulation. PreScan comes with a powerful graphics processor, an advanced 3D visual display, and standard MATLAB / Simulink connectivity. It consists of various main units. An array of sensors where the readings of multiple sensors are simulated and captured; Multiple sides can be photographed in one scene, and an integrated view can be viewed and viewed on a moving camera, with dynamics and control models simulated to gather.

By delving into previous research and studies, deep reinforcement [7] learning has led us to newer capabilities in solving complex control and mobility-related tasks. The research provides several solutions through the autonomous navigation system to enhance deep learning and avoid the obstacles of self-driving cars, which was applied with the [8] Deep Q Network to simulate a car in an advanced environment. Two types of sensor data were used: the camera sensor and the laser sensor located in front of the vehicle. It's also a cost-effective high-speed prototype car that can run the same algorithm in real-time. Also, the design uses a Hokuyo Lidar Camera and a sensor on the front of the vehicle.[9] The integrated graphics processing unit (Nvidia-TX2) is used to run deep learning algorithms based on the input of the sensors

It has been noticed through real-life that with traffic [10] congestion some unexpected accidents happen. It has been also encountered these problems with vehicles. A previous study of some solutions to this problem was presented. To avoid these accidents by providing a means to detect vehicles in the surroundings of the autonomous vehicle and issuing alerts to avoid a potential collision. With her on the road, through two stages. It is implemented in the Android Operating System (ROS). The first part is used to discover vehicles located near an autonomous vehicle. The YOLO v2 algorithm, which was developed on a set of newly created images, was used. The YOLO v2 algorithm is designed to detect three classes of objects: car, truck, and bus. The second part of the proposed method is a ROS node to evaluate the distance to avoid collision by creating two ROS nodes to evaluate distance; ROS node is used to evaluate distance in Carla simulator, other ROS node is used to evaluate the real-world distance. And display the results presented from the previous two parts.

There was another study on avoiding traffic congestion [11], which is based on detecting traffic signals and recognizing them through the camera installed in the car and to increase safe driving on the road, the camera plays an essential role in automatic detection and recognition of traffic lights (lights) in smart vehicles using assistive systems Using [12] advanced driver (ADAS). However, detecting and recognizing traffic lights is a challenging challenge due to their small size and colors (red, yellow, green) which may be like other things, lighting diversity, environmental conditions, etc. With the study and proposed work by FBL., [13] They detected traffic lights using a region-based convolutional neural network (R-CNN) and recognized traffic lights based on forked learning from Grassmann. They also used the learning transfer on the VGG16 to extract features from the detected traffic lights and use these features to create subspaces for each traffic signal (red, yellow, and green). These subspaces are in the Grassmann manifold and include variations in different states of the same traffic light expected during detection.

Traffic congestion is a serious global problem, and to avoid this in the future [14], a study in 2017 introduced Enhanced or Deep Learning Algorithms in the Traffic Simulation Study. And he relied on improved learning on a repetitive search algorithm, and the main goal of the algorithm was to find optimal vehicle lanes and avoid traffic congestion. The Deep Q Learning algorithm was able to learn appropriate strategies, considering the dynamic and sudden changes in traffic at the intersection of highways. In particular, the target network of Q learning has created a mobile system that is able to make this smart method based on reinforcement learning an effective tool for improving vehicle tracks, including the autonomous driving system. A visual example of requested by the passenger is displayed in Figure 2.

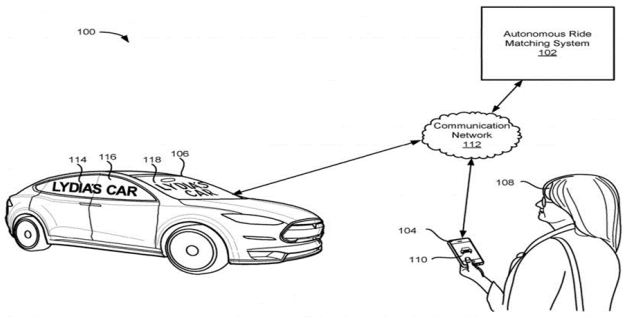

Fig. 2: An example of demonstration of an autonomous vehicle request by a passenger.

The problem of self-driving car control and its failure to deviate [15] from the lane and collision with other cars driven by humans and sharing the road with humans without any problems in the future was posted. A previous study provided a solution to this problem, which are pre-emptive collision avoidance algorithms that consider the intention of drivers of vehicles the advanced algorithm uses multi-stage Gaussian operations (GPs) to determine the transmission model for each vehicle by studying the driver's intent. It also updates its lane predictions online to

provide a collision avoidance-based lane prediction that adapts to different driving styles, whether the vehicle is driven by a human or non-human being, road, and weather conditions. The effectiveness of this concept is evidenced by a variety of simulations that use data derived from real human driving in various scenarios including highway intersection. Experiments are conducted through sophisticated driving simulations and highly realistic simulations of third-party vehicles.

As mentioned previously CARLA provides autonomous driving solutions. Basically, it was built with modular approach and scalable APIs, to tackle problems associated with autonomous driving. Along with other features CARLA aims to support democratization of simulation process, being a tool that can be easily customized by the user. To achieve this goal CARLA simulator depends upon driving learning rules along with perception algorithm depending upon semantic segmentation. The structure of CARLA simulator is based upon Unreal Engine along with Open Drive that support various functionalities like roads, urban and weather settings. Simulation control is given to the user through API that are handled through python or C++ programming. Additionally, to support smooth driving, development and testing process CARLA continues to evolve becoming a huge and more interactive paper build by the community of CARLA simulator. With the passage of time CARLA simulator is providing more options to be more efficient tool of autonomous driving and environment testing. All these extended features and modules are provided in open-source format. Simulator acts as white box with extended access to anybody letting them use these features and customize them accordingly.

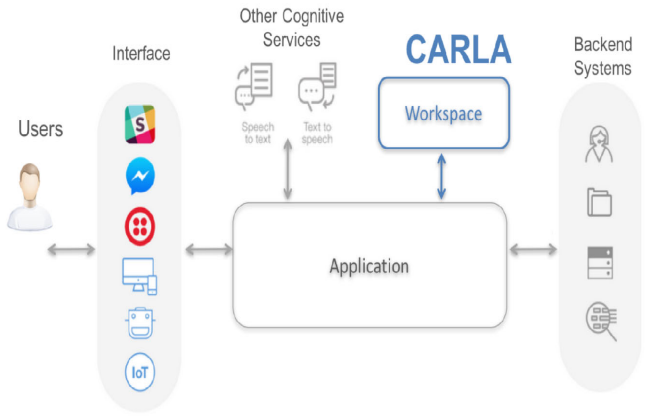

Fig.3. System architecture diagram to show in figure that represent all the components and sub-system collaborating in the system and its functionality.

It should be noted here that autonomous driving in urban traffic environment is difficult task due to various reasons specifically the involvement of high-speed intruders. CARLA being high fidelity simulator is based upon crowd detection and driving through algorithms. The simulator offers Open Street Map database that has tendency to detect the heterogeneous motion of multi agents in urban traffic in open-source environment. This summit is built in accordance with the laws of physics as well as autonomous driving realism. In this way CARLA architecture offers a variety of services including perception and planning mechanism, vehicle control system, learning approach mechanism, etc. in real time traffic behavior detection and optimum performance.

3.1 Environment Setup

To test and develop the E-share application, there are different cars created on Carla Simulator. In this paper, there are 50 cars used with different parameters setting that are explained in the subsequent paragraphs. An example is demonstrated in Figure 4.

Each database contains metadata file with used cameras, the cars number to be used in each episode, the range of cars number to be used in each episode, and the range of the number of pedestrians to be used in each episode.

- The percentage of long noise.
- The percentage of longitudinal noise.
- The set of weathers to be sampled from.

1. Each episode is stored on a different folder and for each collected episode, it has been generated a json file containing its general aspects which is:
2. Number of walkers: the total number of spawned walkers.
3. Number of Vehicles: the total number of spawned vehicles.
4. Spawned seed for walkers and vehicles: the random seed used for the CARLA object spawning process.
5. Weather: the weather applied in episode.

Every episode lasts for 1-5 minutes partitioned in simulation steps of 100ms, for each step data stored divided into two different categories, sensor data stored as PNG images, and measurement data stored as json files. All collected images are stored as PNG, all lidar sensors collected are store as PLY files. The collected sensors are the described in the configuration file. Measurements represent all floating data collected for each simulation step. Each measurement is associated with its own sensor data. The units of measure are in SI format, otherwise it specifies the format, all measurements are stored in json files and contain:

- Step Number: the number of the current simulation step, starts at zero and is increased by one for every simulation step.
- Timestamp: the time that has passed since the simulation has starts Expressed on milliseconds.

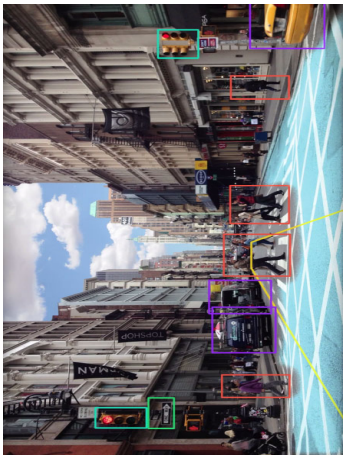

Fig.4. An example of Carla environment setup.

- Position: the world position of the ego-vehicle. It is expressed as a three-dimensional vector (x, y, z) in meters.
- Orientation: the orientation of the vehicle in the frame.
- Expressed as Euler angles (row, pitch and yaw).
- Acceleration: the acceleration vector of the ego-vehicle
- Forward Speed: expressing the linear ahead speed of the vehicle.
- Intentions: a signal that is proportional to the effect that the dynamic objects on the scene are having on the ego car actions. uses three different intention signals: stopping of car and stopping for traffic lights and coming much closer.
- High Level Commands: the high-level command tills what the ego-vehicle must do in the next intersection: like go straight or turn left or turn right or ignore (the ego vehicle could pick any option). These commands are encoded as an integer number.
- Waypoints: a set containing the 10 future positions of the vehicle.
- Steering Angle: the current angle of the vehicle's steering wheel.
- Throttle: the current pressure on the throttle pedal.
- Brake: the current pressure on the brake pedal.
- Hand Brake: is the hand brake activations.
- Steer Noise: the current steering angle in the vehicle considering the noise function.
- Throttle Noise: the current pressure on the throttle pedal considering the noise function.
- Brake Noise: the current pressure on the brake pedal considering the noise function.

- For each one of the non-player agents (Walker's, vehicles, traffic light), the following information is provided:
- Unique ID: an unique identifier.
- Type: like is it walker or a vehicle or a traffic light.
- Position: the world position of the agent. It is expressed as a three-dimensional vector
- (x, y,z) in meters.
- Orientation: the orientation of the agent with respect to the world. Expressed as Euler angles (row, pitch and yaw).
- Forward Speed: a scalar expressing the linear forward speed of the agent.
- State: only for traffic lights, contains the state of the traffic light, if it is either red, yellow or green.

3.2 Proposed Methodology

Autonomous driving is dependent upon different algorithm being used by CARLA simulator as shown in Fig.3. These algorithms include modular pipeline, imitation learning and reinforcement learning. First approach decomposes the driving mechanism into different sub-systems namely perception, planning and consecutive control. No input in terms of map is provided so map perception becomes a critical task. It should be noted that perception approach utilizes semantic segmentation technique to estimate the road boundaries, sideways, static and dynamic objects along with hazards.

A local planner utilizes rule-based state machine to help in the implementation of predefined rules and regulation of urban traffic environment. Consecutive and efficient control is performed with the help of PID control unit that helps to accurate the steering and braking mechanism.

Second approach is conditional imitation learning. This approach depends upon comparatively higher commands along with perception input. This method uses datasets in the form of tuples {oi, ci, a} (observation: oi, command: c and action: a). commands are given by the drivers, which are collected during data collection process. Datasets can suggest the future turns of the driver from perception approach. Most used commands include follow lane, go straight till next intersection, turn left and turn right. The observations include images from cameras.

Third approach is including deep enforcement learning. This methodology involves reward signal with no human involvement. For this purpose, Asynchronous Advantage Actor Algorithm is used. This algorithm has shown excellent performance in three dimensional fields against positional changes. Due to its asynchronous approach, it can allow running multiple sequences at the same time and handling the complexity of the reinforcement learning.

CARLA offers a variety of designs and features to test and verify the autonomous driving mechanism. There are other simulators that claim to provide simulation feature but not any of them have features even comparable to CARLA. These alternatives include auto ware, AirSim and TORCS which are based upon Gazebo, UE4 & Unity, and OpenGL architectures. Additionally, there are Udacity Simulator, Donkey Car Simulator, which are simpler and use unity-based architecture. In comparison to CARLA, AirSim also uses the same technology, but it has very less features and user controls. There are several setup parameters that are defined from the Table 1 to Table 3.

Table 1: Running the CARLA Simulator with expected and actual results.

Step name: Simulator running.				
Description: Testing the simulator.				
Pre-condition: should have the systems and software requirement.				
Step number	**Test steps**	**Expected result**	**Actual result**	**Statues**
1	Running the simulator Carla UE4	The system should display main frame	The map in simulator is displayed	Pass
2	Opening the python API then examples file and running the Command prompt "	The command prompt window should display	command prompt window displayed	Pass
3	Running python spawn_npc.py	The system should display default operating test	The system operations work	Pass
Post-conditions: All system requirements have been validated and simulator are working successfully.				

Table 2: Running the CARLA Simulator with Driver Control options.

Step name: Controls options				
Description: driving control, auto drive, weather dynamic				
Pre-condition: should have the systems requirement and simulator server connection.				
Step number	**Test steps**	**Expected result**	**Actual result**	**Statues**
1	Opening the examples file and running the"Command prompt" running python manual_control.py	The system should open new frame for manual driving using "w,a,s,d" press buttons	The system opened new frame for manual driving, can be control using "WASD" press buttons	Pass
2	Changing the driving mood from manual to Autopilot mood using press button "P"	The car should move by itself	The car moved by itself	Pass
3	Opening the examples file and running the"Command prompt" running python dynamic_weather.py	System should change the weather dynamically	System changed the weather dynamically and can be control using "C" press buttons	Pass
Post-conditions: All system changes have been validated and applied on the simulator successfully.				

Table 3: Proposed modification setting for the CARLA Simulator with expected and actual results.

Step name: Cars modification				
Description: Adding car, car type, gear type				
Pre-condition: should have the systems requirement and simulator server connection.				
Step number	**Test steps**	**Expected result**	**Actual result**	**Statues**
1	Opening the examples file and running the"Command prompt" running python spawn_npc.py -n 20	The system should display 20 cars on the map	The system displayed 20 cars on the map	Pass
2	Choosing a type of car using press button "backspace"	System should show you different car type with each press	System shows you different car type with each press	Pass
3	Choosing gear mood using press button "M"	System should change the gear mood from Auto to manual mood	System changing the gear mood from Auto to manual mood	Pass
Post-conditions: All system changes have been validated and applied on the simulator successfully.				

4. Experimental Results

To perform the experiments, the 64-bit Windows/Linux/MacOS, 8-GB RAM or more, Quad core Intel or AMD processor, 2.5 GHz or faster and NVIDIA GeForce 470 GTX or AMD Radeon 6870 HD series card or higher along with 10GB of hard drive space for the simulator setup needs to be setup. Whereas the following software requirements are required.

To run the interface, following software are required.

- Model Predictive Control Toolbox
- Simulink Control Design
- Carla 2.0
- Python 3.7
- ROS
- RSS.

This section includes the requirements that specify all the fundamental actions of the software system

- Feature: Admin -login

- Scenario: When the system administrator to enter the system must use the user's password and password of the administrator "key"
- because he has all the powers that help him to manage the system in the way required

  - Feature: Driver - Record Saved data
- Scenario: The driver logs on to the system and goes to the Destinations data recording screen It then He can choose one of the previously saved places like home, work

  - Feature: Driver - login to the system
- Scenario: The driver can access the system then the driver can do some operations on the system

  - Feature: driver – Estimate distance
- Scenario: when the user goes at high speed and the vehicle nearly another vehicle automatically the system decreases the speed

  - Feature: driver – Localization Stage.
- Scenario: When the user drives on the highway he turns on the Localization button that does select the best path.

  - Feature: driver – auto pilot
- Scenario: When the user wants to take a rest, he activates through the control panel the auto-driving mood.

  - Feature: passenger - request a car
- Scenario: passenger can request an autonomous car for a ride.

The CARLA simulator is based upon easy-to-use client-server architecture. It should be noted that the server is solely responsible for each and everything about simulation including the sensor controls, physical parameters as well as map and weather states. It is recommended to use dedicated GPU for this process as it helps to generate more realistic results when using machine learning. On the other hand, the client side is the sum of different modules being used at the client end. Client is allowed to communicate with the server through CARLA API based upon Python or C++ modules. This API is constantly being modified to provide more functionality. Some of the features are described hereunder in order to create better understanding about the structure of CARLA simulator.

Efficient working of CARLA simulator is dependent upon data retrieval in the form of raw data from sensors and processes data from computer. This document introduces set of entities that are collected as simulation data. By default, the simulation starts with standard settings of traffic, weather, etc. Ego vehicle roam around the city having basic sensors. On the basis of data recorded on these sensors new simulation can be generated, sensors can bead and many more customizations can be performed. CARLA simulator entities include simulation, traffic, ego vehicle, basic sensors data (including camera and RGB sensors, etc.) and advanced sensors (depth camera, semantic segmentation camera, RADAR and LIDAR sensor). All these datasets are recorded, manipulated and retrieved for accurate simulation results. Map setting includes the setting related with area and weather as show below.

Fig.5. A various visual example of the design of proposed E-Share system using machine learning and Carla Simulator

Each entity has dataset of its attributes that depends upon the following information:

- Cameras
- Image size
- Number of cars used in each view/screen
- Number of pedestrians in each view/screen
- Lateral and Longitudinal noise level
- Weather set

To store the data of each episode a separate folder is generated, that contains json file with following dataset.

- Number of spawned pedestrians and vehicles
- Speed of pedestrians as well as vehicles, normally it stores the average speed assigned by CARLA simulator
- Weather data of episode

It should be noted here that sensor data is stored in PNG format and measurements are stored in json file format.

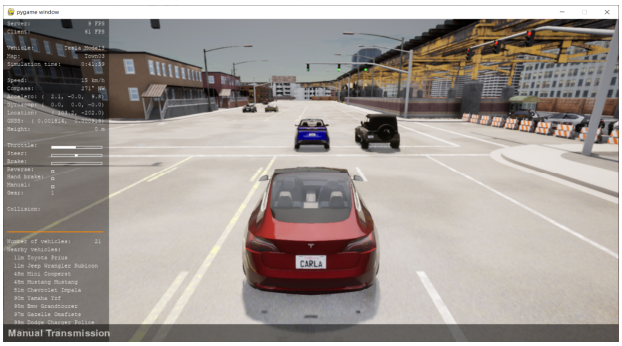

Fig.6. A visual example of the Car waiting for the signal to be green.

The transition from conventional cars to autonomous vehicles has not yet occurred, however, modern artificial intelligence technologies and the development of machine learning are making rapid leaps forward, and paper was one of the examples of modern artificial intelligence techniques to develop machine learning, and at the end of this paper, it talked about future work for Autonomous vehicles. An autonomous vehicle is a vehicle or truck that can sense its environment and control its movements without human intervention. With this type of vehicle, the human driver will be responsible for all driving tasks, and these vehicles include some driver assistance features such as lane-keeping assist or adaptive stability control Speed, the car can combine two or more automatic tasks such as steering and acceleration simultaneously, and the car can drive from point A to point B without human intervention, but only in certain circumstances. The driver still needs to be prepared to take on the task at any time because the system will require a human to intervene in critical situations. The vehicle is fully autonomous in most, but not all, driving conditions. In general, the car will be able to drive on its own and not require human intervention to complete the journey. In this paper, a model is presented for the development of autonomous vehicles through the Carla simulator, and the development of conditions similar to reality to overcome the problems faced by autonomous vehicles on the roads by developing sensors and linking them to the vehicles for experiment and drawing plans to develop them in the future more precisely. This experience with the Kala simulator had a clear effect on the study and development of autonomous vehicles. Many people strive to be the first to develop a self-driving car that can drive in all driving conditions. Big automakers and tech giants are looking for a way to get to the front of the race. They run a bunch of experiments on the road with their self-driving cars, and the data gathered from these experiments help cars navigate a world where unexpected things happen all the time. In the future of autonomous driving, cars will not only take on many of tasks, but the societies will undergo a massive redesign. The cars will drive you to the destination and continue to serve the next customer. Then there would be no need for sprawling parking and underground parking, and self-driving cars could provide people with disabilities with the basic freedom to come and go as they like.

Autonomous driving is dependent upon different algorithm being used by CARLA simulator. These algorithms include modular pipeline, imitation learning and reinforcement learning. First approach decomposes the driving mechanism into different sub-systems namely perception, planning and consecutive control. No input in terms of map is provided so map perception becomes a critical task. It should be noted that perception approach utilizes semantic segmentation technique to estimate the road boundaries, sideways, static and dynamic objects along with hazards. A local planner utilizes rule-based state machine to help in the implementation of predefined rules and regulation of urban traffic environment. Consecutive and efficient control is performed with the help of PID control unit that helps to accurate the steering and braking mechanism.

Second approach is conditional imitation learning. This approach depends upon comparatively higher commands along with perception input. This method uses datasets in the form of tuples {oi, ci, a} (observation: oi, command: c and action: a). commands are given by the drivers, which are collected during data collection process. Datasets can suggest the future turns of the driver from perception approach. Most commonly used commands include follow lane, go straight till next intersection, turn left and turn right. The observations include images from cameras.

Third approach is including deep enforcement learning. This methodology involves reward signal with no human involvement. For this purpose, Asynchronous Advantage Actor Algorithm is used. This algorithm has shown excellent performance in three dimensional fields against positional changes. Due to its asynchronous approach, it can allow running multiple sequences at the same time and handling the complexity of the reinforcement learning.

5. Conclusions

Autonomous Vehicles in transportation sector is unstoppable. However, the transition from conventional vehicles to Autonomous Vehicles will not happen from one day to the other, instead, it will be a process of several years in which, gradually, new autonomous/automated functionalities will be equipped to the vehicles and introduced to the customers. These automated/autonomous functionalities are, nowadays, known as ADAS (Advanced Driver Assistance Systems). In this paper, through the combination of different ADAS functions, make the vehicle navigate on a highway autonomously, but at the same time, following the traffic rules and regulations requirements, and guaranteeing the safety on the road. To accomplish this objective, the proposed approach is to implement the Policy Gradient Reinforcement Learning method to select the proper function at each moment. The algorithm is tested using a five-lane highway CARLA simulator. E-share aspire to develop vehicles that rely on IoT-enabled artificial intelligence (AI) technologies to deal with self-driving vehicles and their importance to humans and provide the best solutions with easy effort and reduce the financial cost through the development of sensors. And linking them with self-driving vehicles to facilitate transportation. The simulation platform also supports all the sensors needed in smart cities and has full control in all static and dynamic directions in creating maps method for controlling a vehicle upon a roadway includes monitoring a trajectory of a host vehicle based upon a global positioning device within the host vehicle, monitoring a first wireless communication between the host vehicle and a plurality of target vehicles, the first wireless communication including a respective trajectory of each of the target vehicles based upon a respective global positioning device within reach of the target vehicles.

Acknowledgments

The author would like to express their cordial thanks to the deanship of scientific research at Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Saudi Arabia.

References

[1] Thibault Buhet; Emilie Wirbel; Xavier Perrotton , Conditional Vehicle Trajectories Prediction in CARLA Urban Environment , https://ieeexplore-ieee-org.sdl.idm.oclc.org/document/9022290 , 28.Oct/2019.

[2] Tamara Naumovic; Marijana Despotovic-Zrakic; Božidar Radenkovic; Lazar Zivojinovic; Ivan Jezdovic , Development of a Continuous System Simulation Engine in Python Programing Language , https://ieeexplore-ieee-org.sdl.idm.oclc.org/document/9066334 , 20.March/2020

[3] Chen Chai; Xianming Zeng; Xiangbin Wu; Xuesong Wang , Safety Evaluation of Responsibility Sensitive Safety (RSS) on Autonomous Car-Following Maneuvers Based on Surrogate Safety Measurements , https://ieeexplore-ieee-org.sdl.idm.oclc.org/document/8917421 , 30.Oct/2019

[4] W. Farag , Z. Saleh , An Advanced Vehicle Detection and Tracking Scheme for Self-Driving Cars , https://digital-library-theiet-org.sdl.idm.oclc.org/content/conferences/10.1049/cp.2019.0222 , 26.March/2019

[5] Yan-Jyun Ou; Xiang-Li Wang; Chien-Lung Huang; Jinn-Feng Jiang; Hung-Yuan Wei; Kuei-Shu Hsu , Application and Simulation of Cooperative Driving Sense Systems Using PreScan Software , https://ieeexplore-ieee-org.sdl.idm.oclc.org/document/8479296 , 20.Nov/2017

[6] Xue-Mei Chen , Yi-Song Miao , Driving Decision-Making Analysis of Car-Following for Autonomous Vehicle Under Complex Urban Environment ,https://ieeexplore-ieee-org.sdl.idm.oclc.org/document/7830353 , 9.May/2016

[7] Abdur R. Fayjie ,Sabir Hossain , Doukhi Oualid , Driverless Car Autonomous Driving Using Deep Reinforcement Learning in Urban Environment , https://ieeexplore-ieee-org.sdl.idm.oclc.org/document/8441797 , 30.Jun/2018

[8] Tian Tan, Tianshu Chu,Jie Wang , Multi-Agent Bootstrapped Deep Q-Network for Large-Scale Traffic Signal Control ,https://ieeexplore-ieee-org.sdl.idm.oclc.org/document/9206275 . 26.Aug/2020

[9] Tanya Amert, Nathan Otterness, Ming Yang,James H. Anderson; F. Donelson Smith , GPU Scheduling on the NVIDIA TX2: Hidden Details Revealed , https://ieeexplore-ieee-org.sdl.idm.oclc.org/document/8277284 , 8.Dec/2017

[10] Mario Gluhaković : Marijan Herceg ; Miroslav Popovic ; Jelena Kovačević , vehicle Detection in the Autonomous Vehicle Environment for Potential Collision Warning , https://ieeexplore-ieee-org.sdl.idm.oclc.org/document/9161791 , 27.May/2020.

[11] Any Gupta: Ayesha Choudhary, A Framework for Traffic Light Detection and Recognition using Deep Learning and Grassmann Manifolds, https://ieeexplore-ieee-org.sdl.idm.oclc.org/document/8814062 , 12. Jun /2019.

[12] Aleksandra Simić: Ognjen Kocić; Milan Z. Bjelica; Milena Milošević , Driver monitoring algorithm for advanced driver assistance systems , https://ieeexplore-ieee-org.sdl.idm.oclc.org/document/7818908 , 23.Nov. 2016.

[13] Koji Kashihara: Deep Q learning for traffic simulation in autonomous driving at a highway junction , https://ieeexplore-ieee-org.sdl.idm.oclc.org/document/8122738 , 8.Oct / 2017 .

[14] Any Gupta: Ayesha Choudhary, A Framework for Traffic Light Detection and Recognition using Deep Learning and Grassmann Manifolds, https://ieeexplore-ieee-org.sdl.idm.oclc.org/document/8814062 , 12.Jun / 2019

[15] Shaochi Hu: Huijing Zhao ; Mathieu Moze ; Francois Aioun ; Franck Guillemard , Human-like Highway Trajectory Modeling based on Inverse Reinforcement

Learning , https://ieeexplore-ieee-org.sdl.idm.oclc.org/document/8916970 . 30.Oct / 2019
[16] Hussein, A.: García, F.; Armingol, J.M.; Olaverri-Monreal, C. P2V and V2P communication for Pedestrian warning on the basis of Autonomous Vehicles. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; pp. 2034–2039.
[17] Alvarez, W.M.: Moreno, F.M.; Sipele, O.; Smirnov, N.; Olaverri-Monreal, C. Autonomous Driving: Framework for Pedestrian Intention Estimationin a Real World Scenario. arXiv 2020, arXiv:2006.02711.
[18] Dosovitskiy, A.: Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An Open Urban Driving Simulator. arXiv 2017, arXiv:1711.03938.
[19] Krajzewicz, D.: Hertkorn, G.; Feld, C.; Wagner, P. SUMO (Simulation of Urban MObility)—An open-source traffic simulation. In Proceedings of the 4th Middle East Symposium on Simulation and Modelling (MESM20002); SCS Europe: Sharjah, UAE, 2002.
[20] De Miguel, M.Á.: Fuchshuber, D.; Hussein, A.; Olaverri-Monreal, C. Perceived Pedestrian Safety: Public Interaction with Driverless Vehicles. In Proceedings of the2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 90–95.
[21] Alvarez, W.M.: de Miguel, M.Á.; García, F.; Olaverri-Monreal, C. Response of Vulnerable Road Users to Visual Information from Autonomous Vehicles in Shared Spaces. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 3714–3719.

Qaisar Abbas is now working as an associate professor in College of Computer and Information Sciences, Al Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Saudi Arabia. His research interests include such as image processing, Deep learning and Bioinformatics.