This is a preprint version of K. Gulati, K.J. Kubiak, Create a virtual learning environment to test and validate the behaviour of autonomous vehicles, published in Proceedings of the FISITA 2021 World Congress, Prague, 13–17 September 2021.

CREATE A VIRTUAL LEARNING ENVIRONMENT TO TEST AND VALIDATE THE BEHAVIOUR OF AUTONOMOUS VEHICLES

Komal Gulati[1] Krzysztof Kubiak[1]

[1]University of Leeds Woodhouse, Leeds LS2 9JT, United Kingdom (E-mail: gulatikml1@gmail.com)

https://doi.org/10.46720/F2021-ACM-114

Abstract

Climate change and the need for renewable energy are driving the development of electric and hybrid vehicles, however, concerns about road safety still remain. To address this issue and provide better safety and increased mobility there is a need for the development of autonomous vehicle technology and now the automotive industry is heading towards bringing fully autonomous vehicles on the public roads in the next few decades. The major concern with these technology-driven vehicles is testing of autonomous vehicles on public roads as no human intervention would be allowed while driving and this may involve some risk for the driver and the surrounding environment as any error or fault in the system may lead to damage of that environment, loss of manufacturing cost, time, energy and even severe accidents could lead to loss of life. In addition, these vehicles consist of more complex designs than traditional vehicles and thus comparatively would require billions of miles of testing. Considering the above factors, the industry has come up with the solutions to test these vehicles in a virtual environment first using the software in the loop approach.

This concept is still in development and therefore this paper aims to develop a virtual learning environment where the performance of the control algorithms for an autonomous vehicle can be tested and validated under different driving scenarios.

Rigorous research was first carried out to find out the available testing methods and software for performing simulations using different algorithms imposed on the software model for object and path detection. Based on this review a modelling design approach was chosen to perform simulations in MATLAB software. Different driving test scenarios such as a roundabout and a parking lot were created in the Automated Driving System Toolbox and simulation was run in Simulink to test the behaviour of vehicle model in terms of Automated Emergency Braking, Lateral Control, Cruise Control, and results were observed and analysed in Bird's Eye Scope view and in 3-Dimensional Environment using Unreal Engine. The Sensor Fusion technique was used to obtain more precise and accurate results. Vehicle dynamics of the model were also tested in order to compare the stability of the vehicle on the basis of the Kinematic and Dynamic Model respectively. The functionality provided by the software was fully explored and relevant results were presented. This paper is focusing on building a flexible virtual testing environment that can be easily deployed by SME's and start-up companies to develop and test autonomous driving algorithms using the software in the loop approach.

Key Words: Autonomous Vehicles, MATLAB, Simulink, Automated Driving System Toolbox, Virtual Learning Environment, Model Based Design, Deep Learning.

1 INTRODUCTION

The massive transformation of the Automotive Industry from combustion vehicles to fully autonomous vehicles is soon becoming a reality. The Industry is reforming its ideas, beliefs, and strategies and is working really hard to reshape the basic way of living, thus increasing the mobility and connectivity of vehicles at a global level. A large amount of work has already been carried out by a few of the automakers in the industry whether it's Tesla, Uber, or Waymo who have been carrying out intensive testing of these vehicles on the public roads but there is still a lot that needs to be done.

The biggest challenge to bring fully autonomous vehicles on road is that the driver would not be having any controls while the vehicle is in driving mode. This means drivers will lack controllability; thus, safety standards would have to be set very high. In Advanced driver-assistance systems (ADAS), drivers still have controls in their hands, but in the case of level-5 autonomy, the computer embedded in the vehicle would have to take care of all the types of circumstances that could arise on the road while driving. (PHILIP, 2016)

Also, it has been estimated that an autonomous vehicle demands testing of about 1 billion miles(approx.) which seems impractical to achieve if relied only on physical road testing of vehicles taking safety issues into consideration. Waymo has also not been able to achieve these many miles of testing so far. In addition, the number of scenarios under which Autonomous Vehicles need to be tested turns out to be in millions

which are not possible to achieve just by vehicle on-road testing. This indicates that road testing would not be sufficient enough for autonomous vehicles, instead, software testing seems mandatory for future vehicles. (PHILIP, 2016)

This dissertation carries out rigorous research on understanding the developments that have been done so far by various automakers, different testing methods that are being used for testing and validating self-driving vehicles, the current progress of the software companies in providing a platform to carry out virtual simulations, recognizing and drawing the attention of the automakers and the concerned software companies towards the areas that are still concerning and needs to be worked upon. Above all, emphasizing the need for virtual testing of autonomous vehicles and understanding their behaviour in the virtual environment by creating different driving scenarios for the test model and testing different algorithms under various testing methods taking into account the academic limitations of the software.

1.1 Comparison between different parameters required for testing Conventional and Autonomous Vehicles

Autonomous Vehicles testing differs from that of conventional vehicle testing. For the past several years, combustion vehicles have been tested by automotive companies using CAE software such as ADAMS, SolidWorks, ANSYS, and many more to perform simulations to test parameters such as vehicle dynamics, steering controls, NVH, aerodynamics, material strength analysis in order to improve the quality and performance of the ride. (LUCA, 2018)

Whereas in the case of Autonomous Vehicles, not just the test vehicle but also the environment under which the test vehicle would be tested needs to be captured including barriers, walking pedestrians, different types of weather conditions, visibility types such as day or night-time, moving vehicles, road type, animals' sudden intervention while driving and the list goes on. Here, the vehicle is unaware of its surroundings and behaves just based on the perception made by the sensors attached to it such as the camera, LiDAR, RADAR, etc. Therefore, not every CAE software which is used to test conventional vehicles would be suitable to test Autonomous Vehicles. (LUCA, 2018)

1.2 Significance of various types of available sensors

Sensors can be broadly classified into two categories:

1.2.1Exteroceptive sensors

They are responsible for perceiving the information from the outside world which could be either stationary or moving objects and are further classified as active and passive sensors due to their "signal" and "no signal" emitting nature respectively for any action they perform.

Active sensors such as Radar and Ultrasonic sensor when used in fusion with monocular cameras improves the vision by providing large visibility range. They provide 3-dimensional view along with accurate measurement of distance of objects being covered within the visibility range from the ego vehicle. They are appreciated due to their very less weight and cost but are comparatively less accurate than LiDAR. Usually found fitted in the side mirrors of the vehicle. LiDAR is used to generate infrared light waves unlike radar, so they provide better accuracy than radar but fail to function during foggy or snowy season. (EKIM, 2020)

Passive sensors such as Monocular Cameras are majorly used to detect colours and are best suitable for 2-dimensional vision in a computer simulation. Another one is Omnidirectional Cameras which are currently widely used in many applications involving path modelling and for navigation purposes due to their facility of recording and providing images showing 360-degree pan view. (EKIM, 2020)

1.2.2Proprioceptive sensors

Based on the perception of the external environment by the Exteroceptive sensors, a closed-loop is run within the system providing feedback for the vehicle to act accordingly such as in terms of controlling speed and direction of the vehicle.(EKIM, 2020). Currently, almost all modern cars are equipped with such types of sensors which include "Wheel encoders" that are used to measure the travel distance covered by the vehicle. Changes in the speed and location could be monitored using the sensor known as "Inertial Measurement Units (IMU)" . Another type of sensor that could be used for velocity measurement is "Tachometers" . Also, "Altimeters" is used for measuring the height of the vehicle inclination while driving on any ramp road or hilly areas.

2 Literature Review

2.1 Types of Virtual Testing methods for Autonomous Vehicles

One of the methods to test vehicles virtually is by writing control algorithms manually by capturing requirements based on the available documents, also called as trial and error method. Another way could be to test the vehicles directly on physical roads. Both of the methods involve a high level of risk in terms of safety and security, as well as results, may not be accurate enough. In addition, these methods would prove to be more time-consuming as one amendment

in the requirement may lead to rebuilding and recoding of the entire control system. (MathWorks, 2020)

On the other hand, MathWorks has introduced the Model-Based Design method in which a software model is created and can be simulated under several different scenarios using MATLAB and Simulink. Thus, reducing the cost of production. The system also generates automatic code while running the simulation in Simulink. The generated code can be further used and deployed in the hardware to perform physical testing and verification. Hence reduces the chance of any error which leads to greater accuracy and efficiency of the product. This also helps in the reduction of the development time. (MathWorks, 2020)

2.2 Different Levels of Virtual Testing of Autonomous Vehicles

Argo AI is a company in USA which develops and tests their own self driving technology on the vehicles of Ford and Volkswagen who are their business partners. According to them before the vehicle is brought down on road for testing, it needs to undergo many other stages of testing. Firstly, they do "Development Testing" in which the individual components like radar, lidar, camera sensors and various other parts of the self-driving system are tested separately and later they are tested as a whole system in the lab.

The second stage testing involves "Simulation/Regression Testing" in which a virtual learning environment is created in a software and scenarios from a simpler to most complex like that of a complete city are tested. If there is any amendment in the hardware component or the software in which simulation was being conducted, then in that case scenario would have to be re-simulated, also known as Regression Testing. This would be an effective way of testing. (Argo AI website)

In the third stage, "Closed Course Testing" is conducted in which the software model is implemented on a prototype designed for testing on a privately owned closed circuit to test and compare its performance with that of virtual testing. In case any unusual behaviour is suspected, it goes again back to stage 1 for improvisation. After the self-driving technology passes all the 3 stages successfully, it reaches the last stage of "Public Road Testing". This is essential because complications are far more in the real world as compared to the simulation environment. (Argo AI website)

2.3 Feedbacks from Startups and Companies about Model Based Design Testing method

Shivaram N.V from the Ather Energy, India claims to have built an intelligent electric scooter using a Model-Based Design approach. They have been able to test their vehicle in complex scenarios such as in difficult weather conditions, drained-out batteries, and inclined roads. They analysed the results and found out that though by increasing the capacity of the battery would provide better mileage but would impact the size and cost of the vehicle and would disturb the centre of gravity of the vehicle. They were able to optimize the design based on their requirements. Control algorithms were developed in order to charge the battery, manage the temperature of the vehicle, and control the power. Software-in-loop type of testing was performed to test and validate the control algorithms and was later deployed in the scooter's processor. (MathWorks, 2020)

Claes Lindskog from Bombardier Transportation, Germany claims to have built local trains using the Model-Based Design method. Earlier, they used to follow the traditional method in which there was a separate team who used to write the specifications and design the model and another team who would perform simulations using handwritten coding. Later, hardware-level of testing was also performed but if any errors used to come at the final stage, the production used to get delayed for months affecting the cost of the vehicle. Since they have started using the Model-Based Design method, they are able to fulfil all the requirements of the customers as well as there is a reduction in the development time of the vehicle. They also found out an electrical issue at an early stage which according to them otherwise would have not come into consideration. (MathWorks, 2020)

Voyage Auto is a small start-up company located in USA. They are working on autonomous technology for developing self-driving taxis. They aim to reduce the iteration time before the vehicle is actually made available for usage. They use "Model Based Design" approach which creates a model of the actual vehicle in a virtual environment and perform testing the model in critical real-time scenarios which is not feasible to test practically in real environment. This way of modelling and testing iterations is very beneficial especially during the early phase of development as it improves the quality of the software model thereby reducing the number of errors at later stage of testing. The software they used for testing is MATLAB which comprises a toolbox named "Robot Operating System (ROS)" that helps in generating C++ code automatically after running the software model in Simulink. This code can be later directly deployed in the hardware component for physical testing. This saves time as it reduces the chance of error when compared to testing manually coded algorithms. (Alan, 2018)

From the Literature Survey, it was examined that there is a high market demand of Autonomous Vehicles for future mobility but there is still a gap in proper methodologies for testing Autonomous Vehicles and comparatively there is a very less progress on

developing Virtual Learning Environment which could lead to huge damage to environment, loss of manufacturing cost and even human life if these vehicles are not tested first virtually before bringing down on roads for testing. This dissertation focuses on to develop different driving scenarios virtually to test and analyse the behaviour of Autonomous Vehicles in a simulation software in order to emphasize on the importance of Virtual Test Environment for future fully self-driving vehicles as well as going deeper into the available testing methods and software for developing Virtual Learning Environment.

3 Methodology

3.1 Research on Testing Methods available for Virtual Learning Environment

There are 2 main ways currently available for testing autonomous vehicles. The first one is Model-based testing which requires expertise in certain software that provides this type of testing and involves collaboration of different techniques such as path planning, computer vision, mapping and sensor fusion required for simulation purposes. Another one is end-to-end testing which involves deep learning technique that uses deep neural networks. These networks are trained to imitate the behaviour of expert driver in different driving scenarios. These methods are explained in more detail as below:

3.1.1Model-based Testing Methods

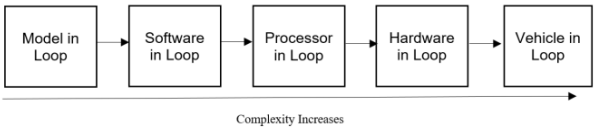


Figure 1. Representation of Increase in Level of Complexity from left to right of different available Model-based Testing Methods.

From the rigorous research it was found that there are 5 types of Testing Methods available for Virtual Learning Environment that uses Model-based design approach. Now, this approach requires to create a model of the available hardware component in the simulation environment that needs to be tested virtually. The first type of Testing Method available is the Model In Loop (MIL) in which a software model is created and is attached to a controller block in the simulation environment which after running the simulation verifies that whether the Controller Block will be able to control the model or not. (Fadaie, 2018)

Once this verification is done, move on to the next level of Testing Method that is Software in Loop (SIL) in which the Controller Block automatically generates a C-Code and this control algorithm that has been generated needs to be again tested in the simulation environment in order to identify that whether this control algorithm can be implemented in the Hardware component or not. (Mangesh, 2020)

Once this verification is done, next is Processor in Loop (PIL) type of testing method in which the simulated control algorithm can be implemented on the embedded processor which acts like a CPU of the vehicle and is responsible for controlling the mechanical and electrical functions of the vehicle. Now, again a closed-loop simulation is run in order to verify that whether this embedded processor is successfully being able to run the control algorithm or not. (Mangesh, 2020)

Once this verification is done, move on to the next Testing Method that is Hardware in Loop (HIL). This is a very important type of testing method in which a real time simulation is run on the PC. Now, for that we need to connect the software model with the real time target machine and run the simulation. A CAN Bus connection needs to be attached that stands for “Controller Area Network” which helps in intercommunication between the controller that has been placed onto the embedded processor and the software model. (Fadaie, 2018)

Once this simulation is run, it needs to be verified that whether there is any error in the communication or not. If it is not then one can proceed to another stage or else first need to rectify the errors and then could move on to the next stage that is the final stage known as Vehicle in Loop (VIL) type of Testing Method in which after all the testing and verification, one can deploy the software model directly into the Hardware component and perform the physical testing which is known as Vehicle in Loop type of testing. (Fadaie, 2018)

The benefits of Model Based Design over traditional method of testing are:

1. In the traditional method, Control System Engineers are responsible for writing the requirements and then Software Engineers are supposed to write algorithms according to the given specifications, further these codes are debugged in the system. This method is very time taking and more prone to errors. This creates a gap in the simulation world and thus Model-Based Design approach is used to bridge this gap. Unlike the traditional approach, in Model-Based Design, the requirements written by Control System Engineer can be customized according to the level of complexity of the testing method and directly linked to the specified system model where unlimited simulations can be performed. The code is automatically generated from the system which helps in performing Rapid Prototyping. Thus, making it a speedy and cost-effective way to perform simulations in real-time on hardware components. To further reduce the development time, requirements from the external sources can be further directly imported and

synchronized within the system using the specified requirement tool. (MathWorks, 2020)

2. Amendments received in the requirements are automatically notified. This helps in the quick detection of the affected part in the system due to the change executed, thereby increasing the response time of resolving any issue. Thus, reducing the development time. In terms of designing, the number of scenarios that could be designed to test the prototype is restricted in the case of the traditional method as every algorithm written manually has to be tested physically whereas the Model-Based Design approach removes all sorts of restrictions in testing and there is no limit to test the model virtually. Questions or errors that could arise at the time of building the prototype could be investigated and identified long before during the time of software simulation in the case of the Model-Based Design approach as testing and verification is carried at each step of development from preparing requirements for the design of software model till the integration of generated code in the hardware model. (MathWorks, 2020)

3.1.2End-to-End Learning Method

Deep Learning provides an end-to-end solution to train and test autonomous vehicle in virtual environment. Hege et al. in their research proposed two architecture models (1) “Convolutional Neural Networks (CNN)” (2) “Recurrent Neural Networks (RNN)”.

CNN is one of the class of Deep Learning. It is majorly used for visualizing images and predicting control signals such as brake, acceleration, and steering angle by using a “feature extractor” with the help of which essential features from the perceived image could be extracted and a “prediction module” with the help of which essential features could be combined with the additional inputs such as speed limit, traffic light. These two modules combine together to help in predicting the control signals. RNN also known as Long Short-Term Memory (LSTM) when combined with CNN helps in enhancing the performance of the vehicle as unlike CNN, it consists of “temporal prediction module” with the help of which temporal dependencies could be learned by the model in between the time steps, thus improvising vehicle’s response in every iteration. (Hege,2019)

Reinforcement Learning is another type of end-to-end learning technique which requires a prototype that needs to be trained to drive in real world. It works on trial-and-error methodology which could cause lots of damage if tried directly on roads thus it is better to use this technology in simulations. Not much human involvement is required as well as consumes less time when compared to traditional method known as supervised learning in which a large amount of training data is used which requires large amount of human involvement for the driving scenarios to be properly captured and labelled, thus this process is highly time consuming. In Reinforcement Learning, the virtual images are translated first to segmented model and then to their counterparts in real world using “scene parsing” method because of which scene structure remains maintained. (Xinlei Pan et al., 2017)

Discussion: After a rigorous research, it was found that there are two types of testing methods that are currently in practise out of which Model-based approach is widely in use while end-to-end learning is still new in the market. Waymo has been the only company who has been successful in end-to-end testing till date. NVIDIA also promises to provide such platform but nothing in news yet about this software in terms of providing successful Level 5 type of testing to any automotive company. It was analysed that there are a lot of benefits of using deep learning technique, but the disadvantage is that it requires large amount of training data to test the model. Also, it seems infeasible for the projects at University level as the testing needs to be carried out either in Lab for a prototype or for Student Formula Cars and unlike passenger cars it does not require as such complex scenarios for testing purposes. Thus, considering all of the above factors as well as the benefits of the latter, this project focuses on using Model-based design approach.

3.2 Research on Software available for Virtual Learning Environment

There are various software companies such as MATLAB, DSpace and ANSYS , start-ups like Cognata and RightHook, Open Sources including CARLA and Microsoft AirSim who are all working on a common goal to provide a virtual platform that could test and validate autonomous vehicle’s functionality up to Level 5 autonomy efficiently. Currently, majorly all software can test Level2/Level3 type of autonomy effectively whereas the Google’s Waymo company uses a commercial software that is Waymo’s “Carcraft” which has been successful in testing Level 5 type of autonomy virtually and has successfully simulated the model for around 5 billion miles on software and approximately 5 million miles on public roads of USA covering more than 25 cities as per the latest statistics.(Fadaie,2018) Few of the available software in the market that provide platform to develop Virtual Learning Environment and simulate Artificial Intelligence algorithms designed for self-driving vehicles are discussed as below.

3.2.1CARLA

Carla Simulator is an open source platform which is used to test different Artificial Intelligence (AI) algorithms used for the autonomous vehicle model in different driving scenarios that is based on an Unreal

Engine which is a gaming engine used to provide 3D animated visual graphics of the simulated scenario. LiDAR and camera are usually used as perception sensors that are attached to the model for testing purposes in this software. Limitation is that this software allows the model to drive only in its prebuilt scenarios and does not provide the platform to create our own virtual learning environment. The software is limited to 7 different prebuilt scenarios which can be accessed by any user. It uses a client-server form of communication medium in which Carla Simulator is the server and the interaction with the client happens using Python API which stands for "Application Programming Interface" that is responsible for inter communication between the server and the client. Raw data from the sensors placed on the model are retrieved and processed by the client using Python scripts after which it is send to Carla Simulator for testing vehicle's brake, acceleration and steering functionality. (Marco et al., 2019). Jussi et al. in their paper emphasizes on the safety of the autonomous vehicle and thus believe in creating machine learning environment for testing these vehicles which is based on Convolutional Neural Networks (CNN). For this they worked on building a hybrid simulation environment which involves Carla Simulator that provides sensor data generated from the simulation in the virtual world to the AI Client software which further transfers to Tensor Flow Machine Learning environment where the CNN models analyse the data and provides autonomous controlling of the vehicle. This type of testing is essential to train the vehicle for the most unexpected and dangerous situation.

3.2.2MATLAB

A new application has been released by MathWorks in MATLAB known as " Automated Driving System" Toolbox. This toolbox provides the platform to create our own scenario and even customize or use as it is the prebuilt driving scenarios in the application for simulating and testing autonomous vehicle's functionality. Also, it contains driving scenarios designed by EuroNCAP such as Automated Lane Change, Automatic Emergency Braking and many other situations. Different types of sensors such as radar, camera can be attached to the ego vehicle, which has to be tested for object and path detection, navigation purposes and path planning. Also, their positioning and placement could be adjusted based on the previous simulation's results for improvisation. Each simulation can be viewed in three different types of views 2-Dimensional, 2.5- Dimensional in the application and Bird's eye view which is the top view of the scenario with respect to the ground as well as ego vehicle coordinates respectively in the Simulink environment. The sensor data of the ego vehicle from the simulated scenario could be exported to the Simulink where the vehicle could undergo all different types of testing methods as discussed in the above section for testing vehicle control and stability. C-Code generation is also supported by the toolbox to conduct software-in-the-loop type of testing which could be further taken to higher levels of testing. Simulink also provides platform to many other software companies such as MSc Software and Carla to test their driving scenario by allowing their coded script to be made shareable to MATLAB Workspace and thus run the simulation. (Marco, 2020). MATLAB also supports deep learning level of testing which uses a Deep Learning HDL toolbox.(F. Wotawa et al., 2018) MATLAB also provides another application known as "Ground Truth Labeler" which contains prebuilt videos used for testing purposes. Various objects in the video like vehicles, pedestrians can be labelled and classified and later provides inbuilt algorithms such as for object detection that can be used for simulation as well as allows to import own created algorithms to test and validate to label the ground truth data automatically. (MATLAB Documentation, R2019b)

Discussion: Among the software, MSc Software company's idea to combine Virtual Test Drive (VTD) with Adams software for more accurate results looks good but it fails in providing currently that much intensive testing which other software like Carla and MATLAB are providing. NVIDIA is a promising software but as Model-based testing has limitation of having intense software knowledge so no as such training was found to be provided for this software at academic level. Carla and MATLAB seems to be widely used at both academic and industry level but they both stand different in certain aspects from each other. Carla is an open source software which is an advantage, but the drawback is that it is restricted with its available scenarios for testing and does not even provide a platform to create own driving scenarios unlike MATLAB which is far more advance in these aspects. In addition, if the user wants to extend the testing further in Carla to Deep Learning testing, it would require some external software like Tensor Flow, MATLAB to execute its algorithm on their platform while MATLAB provides all the facilities within the software itself and it is also student friendly. Therefore, as per the research this project focuses on using Automated Driving System Toolbox along with Simulink environment provided by the MATLAB for developing the scenarios and testing various algorithms implemented on the test vehicle

3.3 Different driving scenarios tested under different Testing Methods using Model-based design approach are explained in detail as below:

3.3.1Model-In-Loop (Open-Loop) Simulation

To perform "Open-Loop" type of testing, a driving scenario considering a Roundabout road was created in

the Driving Scenario Designer application of Automated Driving Toolbox provided by MATLAB. Four types of actors were placed on the road as shown in Figure2.1: (1) Ego Vehicle which is the test vehicle on which ADAS (Advance Driver Assistance System) technology was tested, (2) Truck, (3) Pedestrian, (4) Barrier.

Cameras were placed at the front and rear end of the vehicle to detect the objects and lanes ahead and at the rear of the vehicle and Radars were attached on all the four corners covering all the possible range around the vehicle to detect any upcoming or passing by object.as shown in Figure 2.2. Waypoints also known as trajectory points were marked for all the moving objects and their speed and timings were adjusted and set as per the designed scenario.

The Ego Vehicle was pre-trained to apply Automatic Emergency Brakes few seconds before the truck coming from the West direction road suddenly approaches and comes ahead of the Ego Vehicle. Once, truck went out of the test vehicle's trajectory path, the vehicle regained back it's speed. Then moving further, it found that unexpectedly a pedestrian started crossing the road so again it applied automatic emergency brakes. After the pedestrian crossed the road, the vehicle started moving again and prepared to take a left turn to enter the approaching road and there after a certain distance, it saw a barrier, therefore it reduced the speed and changed its lane.

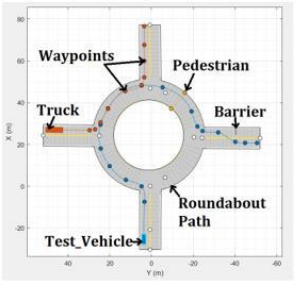


Figure 2.1 Roundabout driving scenario

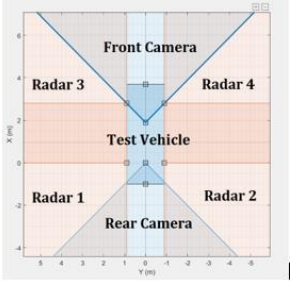


Figure 2.2. Sensors positioning and placement on the Ego Vehicle (R2020a, MATLAB)

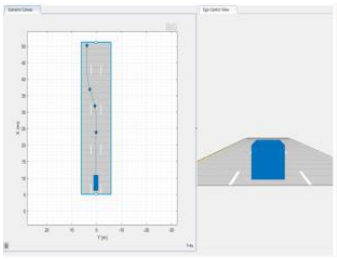
This scenario was then exported as a Simulink model (comprising of Scenario Reader and Detection Generator Blocks) and as a MATLAB function algorithm as shown in Appendix A to perform Open-Loop testing in which the source of the Ego vehicle was set as "scenario" that means no feedback would be provided from the output to the vehicle and would perform based upon its predefined behaviour only. The results were visualized, and sensors performance was analysed using Bird's Eye Scope provided by Simulink. This setup is suitable to test the open loop algorithms and performance of such test can be compared with other algorithms for example closed loop as presented in next.

3.3.2Model-In-Loop (Closed-Loop) Simulation

In this case, "Automatic Steering Control" functionality of a vehicle has been tested as the vehicle was commanded to perform lane change on a single three-lane road by following the desired reference path.

A simple 'Lane Changing' scenario was created in the application along with the ego vehicle and waypoints markings to obtain the desired reference path as shown in Figure 3.

There are two ways by which steering can be controlled – Lateral and Longitudinal. In Lateral steering control, distance between the current position of the test vehicle and the trajectory path is reduced by adjusting the steering angle. Vehicle is set to run at a constant low speed of 12m/s in order to avoid any additional parameters such as tire slip and wind effect, thus maintaining the control and grip of the vehicle. Unlike Longitudinal steering control in which the speed may vary depending on the applied acceleration or brake. For this project, simulation has been restricted to Lateral Steering Control only.



Waypoints

	x (m)	y (m)	z (m)
1	7.2000	0.3000	0
2	23.9000	0.2000	0
3	31.9000	0.7000	0
4	36.9000	2.7000	0
5	50.2000	3.9000	0

Figure 3. Driving scenario created for a Lane Change Steering Control Testing (R2019b, MATLAB)

Then this scenario was configured on the Simulink Model for further closed-loop testing of Vehicle Dynamics. Closed-loop testing indicates that the output would provide feedback to the input for every iteration. In order to test vehicle control, a Bicycle model was chosen comprising of 3 Degree of Freedom that is Lateral, Yaw and Longitudinal motion. Further, the vehicle was examined based on its Kinematic and Dynamic behaviour in order to compare the lateral error at the end of the simulation run time.

In case of Kinematic model, inertia of the vehicle is considered negligible, therefore low speeds are preferable to avoid any inertial effect. In order to execute the steering command, only three factors are responsible – velocity of the test vehicle, current and reference position which is based on the desired trajectory path. Whereas in Dynamic model , inertial effects such as slipping of tires and actuation of steering servo motors are considered. It requires additional parameters such as Yaw Rate and current steering angle which plays a major role in deciding the amount of lateral error experienced by the vehicle. Hence, the simulation was run considering one model at a time by

activating which ever needed to be tested and the results were recorded.

3.3.3A Closed-Loop Testing based on path planning algorithm in a prebuilt 3-Dimensional Automatic Car Parking Environment

This simulation was carried out with reference to an example provided in MATLAB Guide on a prebuilt 3-Dimensional Automatic Car Parking Environment. The objective was to define a path for the automated vehicle to follow and visualize the results in an Unreal Engine which is integrated in Simulink in order to get the feel of Real Environment.

The pre-stored path planning and vehicle control algorithm was customized based on the chosen parking lot scenario and a Costmap was generated as shown in Figure 4 in order to check and verify which path is free from obstacles and collisions such as cones, pedestrians, barriers, parked vehicles and accordingly it was decided which path would be suitable for the test vehicle to follow.

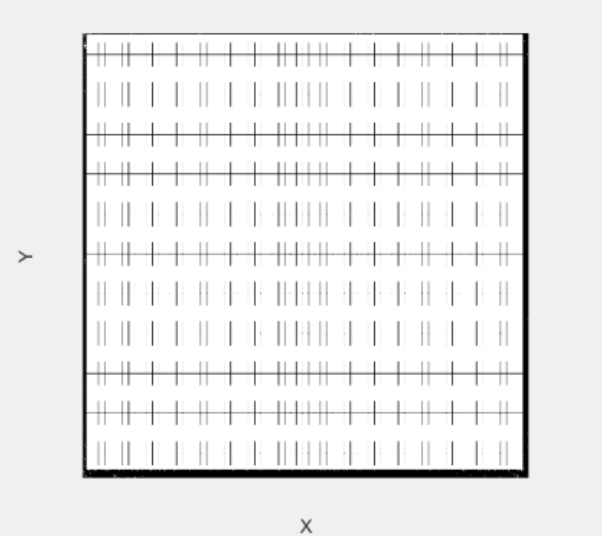


Figure 4. Costmap Representation of the chosen Parking Lot (R2019b, MATLAB)

Based on the above analysis, route plan was marked by setting waypoints for the test vehicle to reach its final destination that is the available vacant parking spot as shown in Figure 5.

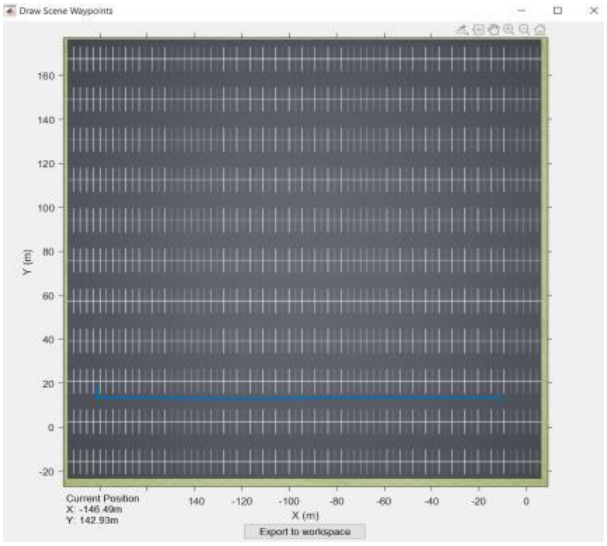


Figure 5. Waypoints marked for the vehicle to follow the defined trajectory path (R2019b, MATLAB)

The chosen 3-Dimensional Scenario "Parking Lot" and Ego Vehicle were configured in the Simulink Model and the results were observed and analysed in 2-Dimensional view on Costmap and 3-Dimensional view in Unreal Engine Environment.

3.3.4Testing and verifying Automatic Emergency Braking based on Euro NCAP scenario using Software in Loop Testing Method

A driving scenario was created in the application with reference to EuroNCAP protocols but was customized as instead of straight road, this time a flyover was taken, and the scenario represented an ego vehicle and a bicycle running at a constant speed on the same path as shown in Figure 6. Once the flyover bridge was crossed, the vehicle encounters the bicycle on its path, but the driver fails to apply emergency brakes due to distraction or unawareness in terms of how to react in emergency situations. Thus, colliding with the bicycle's rear end as shown in Figure 7.

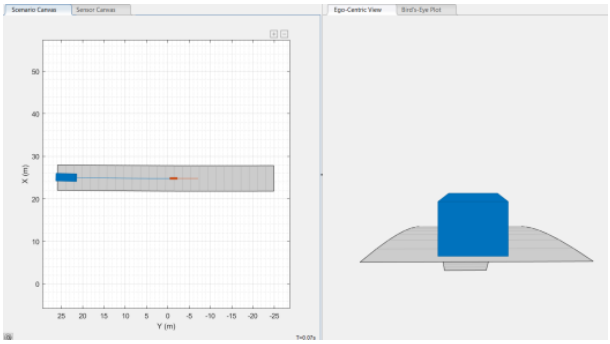


Figure 6. Flyover driving scenario when the vehicle is at the start position (R2019b, MATLAB)

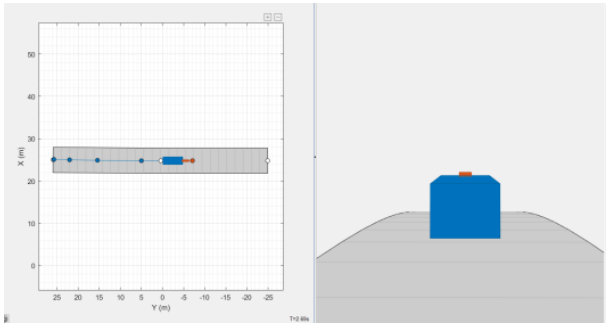


Figure 7. Flyover driving scenario when the vehicle collides with the bicycle (R2019b, MATLAB)

Therefore, using sensor fusion of camera and radar as shown in Figure 8, the scenario was tested in the application and the Simulink model was configured. This testing was done with reference to an example provided by MATLAB Guide. Simulink Model was run, and the results were noted.

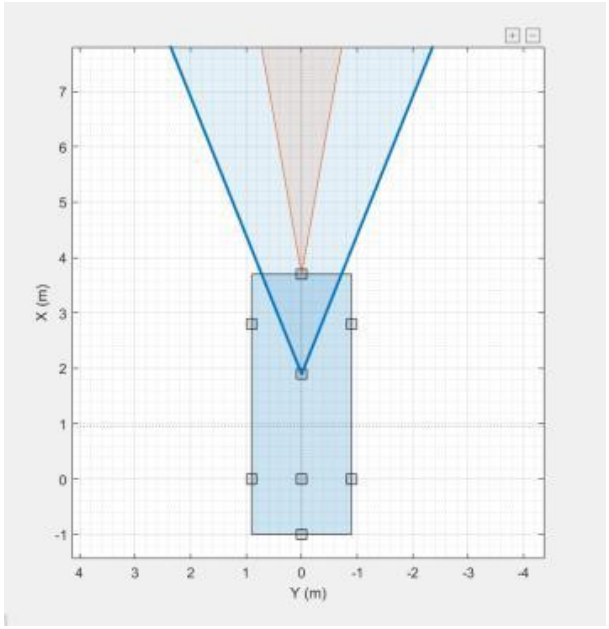


Figure 8. Sensor fusion of radar (red wave) and camera (blue wave) sensors (R2019b,2020)

4 Results

This chapter contain main results of the simulations and testing undertaken in this project in 4 different scenario setups presented in previous chapter.

4.1 Model-in-Loop (Open-Loop) Simulation

The simulation was run, and the results were visualized in the Bird's Eye Scope of the Simulink which is the top view of the scenario indicating visualization with respect to ground and ego vehicle respectively as shown in Figure 9.

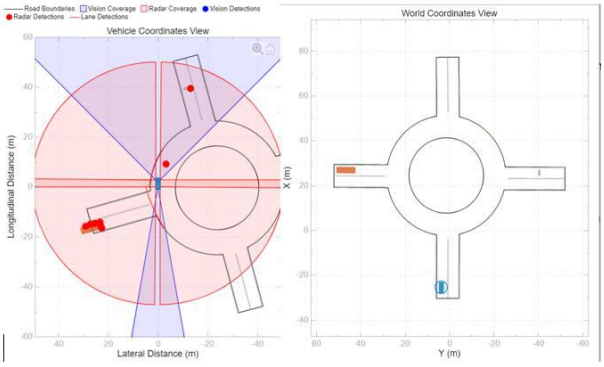


Figure 9. Bird's Eye view captured in case of emergency brakes applied for pedestrian crossing (left) view with respect to ground (right) (R2019b, MATLAB)

It could be observed that the sensors attached to the ego vehicle were very precisely able to detect objects and lanes in the developed scenario as seen in Figure 9, red and blue coloured signals are indicating radar and camera sensors respectively while the black lines are representing the path to be followed. The scenario in Figure 9 was captured while the vehicle applied automatic emergency brakes when the pedestrian unexpectedly started crossing the road. Thus, an expected outcome of sensors behaviour from an Open-Loop model was achieved successfully. This scenario shows potential for testing vehicle control algorithms for unexpected inputs of the sensors in open loop control.

4.2 Model-in-Loop (Closed-Loop) Simulation

In a Closed-Loop Simulation, feedback is provided from the output back to the input in order to improve its performance in every next simulation. Hence, in this case a Closed-Loop Simulink Model was customized as per the designed scenario in order to test the Kinematic Bicycle Model. Simulation was run and results were obtained in the Bird's Eye View as shown in Figure 10.

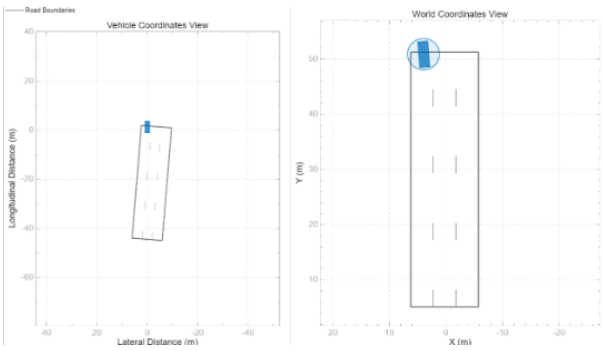


Figure 10. Bird's Eye view indicating vehicle position at SimStopTime for Kinematic Bicycle Model (R2019b, MATLAB)

Lateral error also known as "cross-track error" which is the angle formed due to deviation between the current steering position and the actual reference position. This error was analysed using the scope block as shown in Figure 11.

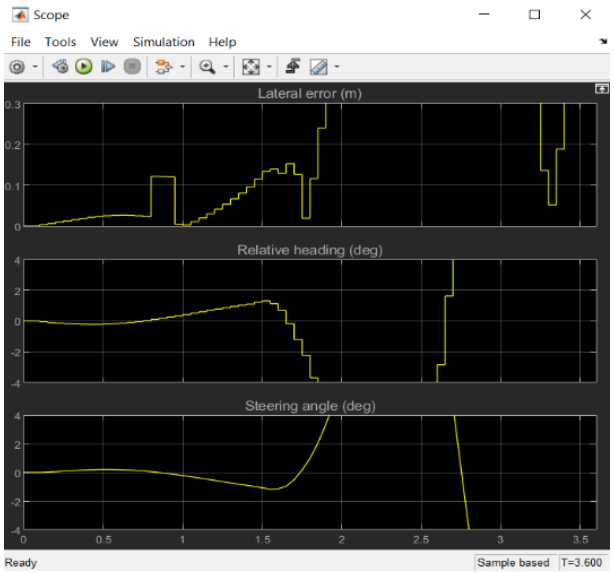


Figure 11. Graph representing the lateral steering error for Kinematic Bicycle Model (R2019b, MATLAB)

Similarly, simulation was carried for Dynamic Bicycle Model and the results were noted as shown in Figure 12 and 13.

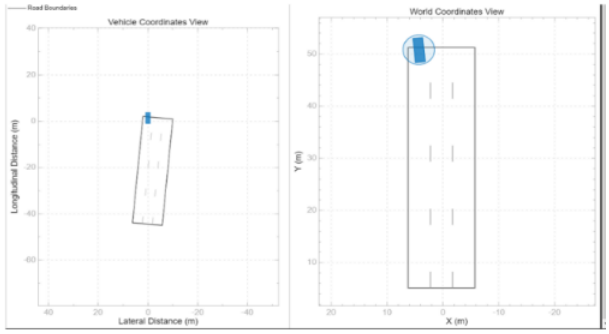


Figure 12. Bird's Eye view indicating vehicle position at SimStopTime for Dynamic Bicycle Model (R2019b, MATLAB)

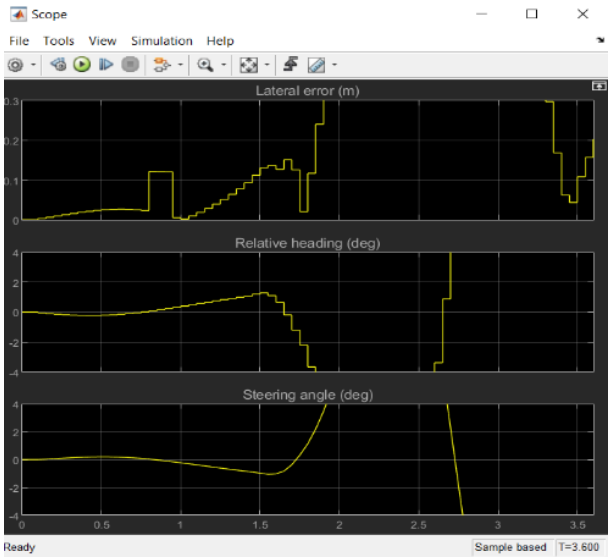


Figure 13. Graph representing the lateral steering error for Dynamic Bicycle Model (R2019b, MATLAB)

The below data of Kinematic and Dynamic Bicycle Model was obtained from Figure 10 -13 and compared as shown below:

Model Type	Lateral Error	Simulation Stop Time
Kinematic Model	0.3	3.6
Dynamic Model	0.2	3.6

Table 1. Comparison of Lateral Steering Error between Kinematic and Dynamic Bicycle Model (R2019b, MATLAB)

From the above Table 1, it was analysed that at the end of the simulation run time, for the same heading and steering angle, Dynamic Bicycle Model shows less deviation from the reference path in comparison to Kinematic Bicycle Model which means latter shows more vehicle stability and control. Also, it was observed while simulating that with decrease in Yaw rate, there is reduction in lateral error. This parameter is not available for adjusting as Kinematic Model is assumed to have no inertia thus no Yaw rate factor comes into picture. Therefore, emphasizes the need for Dynamic Model Testing for more accurate and better results.

4.3 A Closed-Loop Testing based on path planning algorithm in a prebuilt 3-Dimensional Automatic Car Parking Environment

The pre-stored algorithm for testing path planning was customized as per the chosen prebuilt driving scenario of "Parking Lot". This algorithm was then later configured on the Simulink Model. The scene description parameter was set to "Parking Lot". The simulation was run, and the results were obtained based on the waypoints located on the trajectory path during coding.

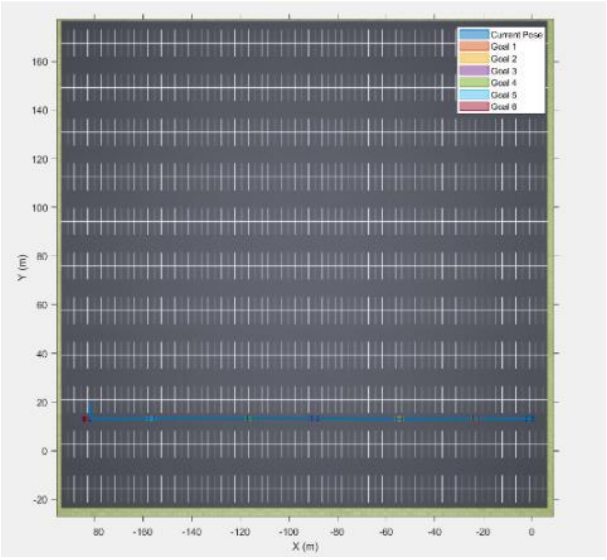


Figure 14. Global Costmap representing the vehicle following the trajectory path (R2019b, MATLAB)

While the simulations runs, vehicle's position and orientation got updated in the 3-D Environment which appeared in the new window as shown in Figure 15.

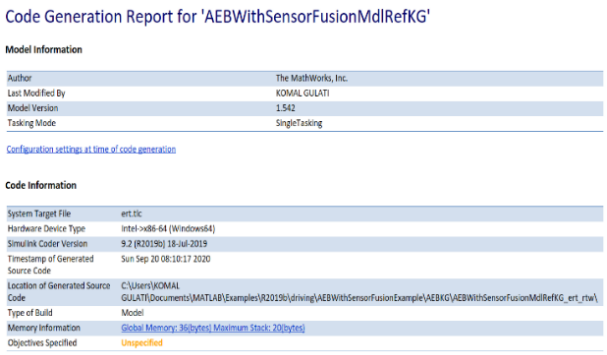
Figure 15. 3-Dimensional view of the final parked position of the vehicle in Unreal Engine (R2019b, MATLAB)

As shown in Figure 14, a 2-Dimensional view of the test vehicle travelling towards the parking spot following the desired waypoints was obtained on a Global Costmap after running the Simulink model. The different colour vehicles on the map represent the different waypoints located in order to reach the final goal spot. Simultaneously, Unreal Engine integrated in the Simulink runs the simulation in 3-Dimensional Environment and a screenshot was captured of the scenario when the vehicle finally reached its final destination as shown in Figure 15. Thus, obtaining the desired results.

This scenario provides additional versatility to the virtual testing environment created in MATLAB and demonstrate potential for path optimisation algorithm testing. Path length and driving time can be compared between different tested algorithms.

4.4 Testing and verifying Automatic Emergency Braking based on EuroNCAP scenario using Software in Loop

A prebuilt scenario designed by EuroNCAP was customized and the pre-stored algorithm was further customized based on the designed scenario. After running the customized set up file for Automatic Emergency Braking, a C-Code was generated as shown in Appendix C and the published report is attached here for reference as shown in Figure16.



Code Generation Report for 'AEBWithSensorFusionMdlRefKG'

Model Information

Author	The MathWorks, Inc.
Last Modified By	KOMAL GULATI
Model Version	1.542
Tasking Mode	SingleTasking

Configuration settings at time of code generation

Code Information

System Target File	ert.tlc
Hardware Device Type	Intel->x86-64 (Windows64)
Simulink Coder Version	9.2 (R2019b) 18-Jul-2019
Timestamp of Generated Source Code	Sun Sep 20 08:10:17 2020
Location of Generated Source Code	C:\Users\KOMAL GULATI\Documents\MATLAB\Examples\R2019b\driving\AEBWithSensorFusionExample\AEBKG\AEBWithSensorFusionMdlRefKG_ert_rtw\
Type of Build	Model
Memory Information	Global Memory: 36[bytes] Maximum Stack: 20[bytes]
Objectives Specified	Unspecified

Figure 16. Code Generation Report of Automatic Emergency Braking using Software-in-Loop Testing Method (R2019b, MATLAB)

The generated algorithm was implemented on the Simulink Model for further testing the Automatic Emergency Braking functionality and the result obtained satisfied the expectation as the vehicle applied emergency brakes after the vehicle entered the Forward Collision Warning zone. The brakes were applied gradually by first partially braking and later followed by full braking.

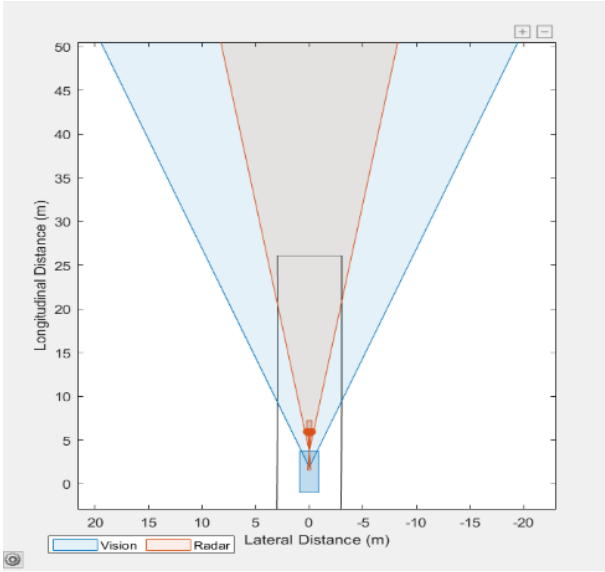


Figure 17. Sensor Fusion representing sensor detection of bicycle ahead (R2019b, MATLAB)

After customizing and running the pre-stored algorithm based on the designed scenario, a C-Code was generated as shown in Figure 16 and Appendix C. This code was implemented in the Simulink model and was run in order to test the Automatic Emergency Braking functionality. Sensor fusion technique helped in providing a more robust and accurate result as shown in Figure 17, where vehicle was able to detect the bicycle ahead and applied brakes before reaching the Time to Collision (TTC). This is a very important functionality of assisted mode of driving as it helps in critical situation identification and thus preventing or reducing the severity of road accidents.

Results of this scenario confirms versatility of the created virtual environment and its ability to combine different sensors input for algorithm testing.

It can be seen that different scenario presented in this section can be considered and vehicle control algorithm can be tested at early stage of development. Further testing should include hardware in loop and real road conditions where vehicle dynamics can have significant input on the vehicle path and control.

5 Conclusion

The project aimed to create virtual learning environment for autonomous vehicles in order to train and test the vehicles before they are brought down on roads for physical testing. In order to fulfil that rigorous research on various available Testing Methods and Software was done. The chosen software was MATLAB and two different testing methods under Model-based design approach were chosen – Model-in-Loop (Open-Loop and Closed-Loop) and Software-in-Loop. The vehicle model was tested in four different types of test scenarios and satisfactory results were obtained for object and path detection. As seen in Case Scenario 1, test vehicle was able to detect the actors such as truck, pedestrian crossing the road and a barrier as it performed the desired action to prevent any kind of damage. Similarly, in case of other scenarios as well sensors were able to effectively perform their defined function. With the increase in level of complexity of the testing model, more robust and accurate results were achieved. Therefore, the overall objective of the dissertation was accomplished successfully to test different algorithms on the vehicle model for object and path modelling in the own created virtual driving scenarios.

6 References

[1] Yurtsever, EKIM et al. 2020. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. IEEE Access

[2] FISITA.2020. Mobility Engineer 2030.[Online].FISITA White Paper [20th September 2020]. Available from: file:///C:/Users/KOMAL%20GULATI/Downloads/FISITA_White_Paper_Mobility_Engineer_2030.pdf

[3] Nasseri Ali et al. 2020. Autonomous Vehicle Technology Report.[Online].WEVOLVER. [20th September 2020]. Available from: https://www.wevolver.com/article/2020.autonomous.vehicle.technology.report#thinking

[4] Bergen, MARK. 2018. How Self-Driving Cars Can Get Past the Learning Permit Stage, Without Any Risk. Bloomberg, p.1

[5] Cattaruzza, MARCO. 2019. Design and Simulation of Autonomous Driving Algorithms. Master of Science, POLITECNICO DI TORINO

[6] HEXAGON| MSC Software. 1963. Virtual Test Drive Toolkit. [Online]. [Accessed 20th June 2020]. Available from: https://www.mscsoftware.com/product/virtual-test-drive

[7] Huang WULING et al. 2016. Autonomous Vehicles Testing Methods Review. [Online]. IEEE 19th International Conference on Intelligent Transportation Systems (ITSC). [Accessed 28 June 2020]. Available from: file:///C:/Users/KOMAL%20GULATI/Downloads/2016Autonomous%20vehicles%20testing%20methods%20review.pdf

[8] Little and Bangert. 1984. MATLAB (R2020a). [Software]. [Accessed 29th June 2020]

[9] Thorn ERIC et al. 2018. A Framework for Automated Driving System Testable Cases and Scenarios. [Online]. U.S Department of Transportation: National Highway Traffic Safety Administration. [Accessed 26 June 2020]. Available from :https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/13882automateddrivingsystems_092618_v1a_tag.pdf

[10] Nascimento, ALEXANDRE M. 2019. The Role of Virtual Reality in Autonomous Vehicles' Safety. IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)

[11] Sportillo, DANIELE et al.2018. Get Ready for Automated Driving using Virtual Reality. [Thesis]. MINES ParisTech, PSL Research University. Centre for robotics, 60 Bd St Michel 75006 Paris, France PSA Group, Technical Center of Velizy

[12] Leudet JEROME et al. 2018. Virtual Environment for Training Autonomous Vehicles. Department of Computer Science, University of Helsinki, Gustaf Hallstrominkatu 2b, Helsinki, Finland

[13] NHTSA. 2020. Driver Assistance Technologies. [Online]. [20th September 2020]. Available from: https://www.nhtsa.gov/equipment/driver-assistance-technologies

[14] Pan XINLEI et al. 2017. Virtual to Real Reinforcement Learning for Autonomous Driving. [Online].Research Gate. [20th September 2020]. Available from: https://www.researchgate.net/publication/316098277

[15] Haavaldsen, HEGE et al. 2019. Autonomous Vehicle Control: End-to-end Learning in Simulated Urban

Environments. [Thesis]. Norwegian University of Science and Technology, Trondheim, Norway

[16] MATLAB.2020. Lateral Control Tutorial. . [Online]. [20th September 2020]. Available from: https://uk.mathworks.com/help/driving/ug/lateral-control-tutorial.html

[17] MATLAB.2020. Visualize Automated Parking Valet Using Unreal Engine Simulation . [Online]. [20th September 2020]. Available from: https://uk.mathworks.com/help/driving/ug/lateral-control-tutorial.html

[18] MATLAB.2020.Autonomous Emergency Braking with Sensor Fusion . [Online]. [20th September 2020]. Available from: https://uk.mathworks.com/help/driving/ug/lateral-control-tutorial.html

[19] Kale, MANGESH et al. 2020. Processor-In-Loop Simulation: Embedded Software Verification & Validation In Model Based Development. [Online]. Design and Reuse. [20th September 2020]. Available from: https://www.design-reuse.com/articles/42548/embedded-software-verification-validation-in-model-based-development.html

[20] Fadaie, Joshua G et al. 2020. The State of Modeling, Simulation, and Data Utilization within Industry. [Online]. The Boeing Company. Available from: https://arxiv.org/ftp/arxiv/papers/1910/1910.06075.pdf

[21] EuroNCAP. 2020. Euro NCAP Improves Tertiary Safety by Introducing a Mobile App for First Responders in Europe. [Online]. Available from: https://www.euroncap.com/en/press-media/press-releases/euro-ncap-improves-tertiary-safety-by-introducing-a-mobile-app-for-first-responders-in-europe/

[22] MathWorks.2020.Model-Based Design for Embedded Control System.[Online]. Available from: https://www.mathworks.com/content/dam/mathworks/white-paper/gated/model-based-design-with-simulation-white-paper.pdf

[23] Koopman PHILIP et al.2016.Challenges in Autonomous Vehicle testing and Validation.[Online]. Available from: http://users.ece.cmu.edu/~koopman/pubs/koopman16_sae_autonomous_validation.pdf

[24] Castignani LUCA.2018.Achieving Autonomous Driving with Simulation and Testing.[Online]. Available from: https://www.mscsoftware.com/sites/default/files/Achieving-Autonomous-Driving-with-Simulation-and-Testing.pdf

[25] Castro ALEX. 2018. INSIDE WAYMO'S STRATEGY TO GROW THE BEST BRAINS FOR SELF-DRIVING CARS. [Online]. THE VERGE. Available from: https://www.theverge.com/2018/5/9/17307156/google-waymo-driverless-cars-deep-learning-neural-net-interview

[26] Ellis, GEORGE. 2012.Hardware-in-the-Loop. [Online]. ScienceDirect. Available from: https://www.sciencedirect.com/topics/computer-science/hardware-in-the-loop/pdf

7 Acknowledgement

Special thanks to my mentor Dr Krzysztof Kubiak whose constant supervision and guidance helped me move in the right direction throughout the project. I would also like to express my gratitude towards my parents, the members of University of Leeds and everyone who supported and encouraged me towards the successful completion of the project.