System configuration of Human-in-the-loop Simulation for Level 3 Autonomous Vehicle using IPG CarMaker

Cheok Jun Hong
Department of Electrical and Electronic Engineering
Faculty of Science and Engineering
University of Nottingham Malaysia
Jalan Broga, 43500, Semenyih, Selangor, Malaysia
kecy6cjh@nottingham.edu.my

Vimal Rau Aparow
Department of Electrical and Electronic Engineering
Faculty of Science and Engineering
University of Nottingham Malaysia
Jalan Broga, 43500, Semenyih, Selangor, Malaysia
Vimal.Rau@nottingham.edu.my

2021 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS) | 978-1-6654-2035-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/IoTaIS53735.2021.9628587

***Abstract*— The increasingly automated vehicles (AV) have increased the complexity of the testing methods and number of driven miles required to demonstrate the vehicle system's reliability. Most modern autonomous driving systems also used deep neural networks which requires a large amount of data to develop. Physical driving alone to collect driving data and test system's safety is no longer suitable for development of AV as this is costly, time consuming and could harm the road users if the safety system failed. This paper proposed human-in-the-loop simulation testing for evaluation of an autonomous vehicle using a 3D virtual vehicle driving platform that can be used for safety assessment of autonomous vehicle. The aim of this study is to establish human-computer interaction platform that can be used as safety testing for Level 3 autonomous vehicle whereby an emergency takeover is required during critical driving conditions. The proposed platform make use of IPG CarMaker to provide 3D virtual environment with accurate vehicle dynamics model, sensor model and environment model. We are able to interface the IPG CarMaker with Simulink and successfully developed a Simulink model that can interface a steering and pedal driving hardware with the virtual vehicle in the simulation. We can also collect driving data and simulation data from the IPG CarMaker as well as accessing the variable in the IPG CarMaker in real-time using Python. The recorded data can be used to train and fine-tune autonomous system based on machine learning.**

Keywords—Human-in-the-loop, IPG CarMaker, Python, Autonomous Vehicle, YOLOv5

I. INTRODUCTION

As autonomous vehicle (AV) reaches higher Society of Automotive Engineers (SAE) levels with increasingly complex self-driving system and Advanced Driver Assistance System (ADAS), testing the vehicle's safety also become a challenging task due to complex testing procedures [1] and would need as much as hundreds of millions of testing kilometres to demonstrate the system's reliability in critical scenarios [2]. The virtual and real approaches testing methods are the most used test methods for AVs. Conventionally, before testing a vehicle in a real environment, the vehicle's system will be tested in a virtual simulation platform with built-in simulated sensor models, vehicle dynamics model and virtual driver model. The advantages of using such platforms are low cost, easy to reproduce and it is possible to test the vehicle's system in critical scenarios that are hard to generate in real world and might cause injuries to road users if the safety system failed. However, as the testing result reliability of virtual simulation depends on the accuracy of the simulated sensor model, vehicle dynamic model, and environment model, it is important to reproduce the testing scenarios as close to the real-world environment in the virtual simulation platform to ensure a good representative of the vehicle's safety in these scenarios.

Most modern self-driving systems are driven by deep neural network (DNN) that requires a large amount of labeled driving dataset to train. Even though there have been many AV datasets published such as nuScenes [3], Waymo [4] and Ford [5], these datasets do not have enough critical scenarios to train the DNN. Besides, labelling dataset collected from driving in real world is often time consuming and expensive. Since humans are prone to make errors, the low quality and accuracy of data labelling will also cause a DNN to fail. In addition, most real-world driving datasets are designed for road and traffic environment in developed countries. Therefore, these datasets might not be suitable to train autonomous vehicle system in developing countries such as Malaysia. On the other hand, large amount of synthetic training data can be generated from virtual simulation platforms easily and at low cost. This is due to the ground truth information can be obtained directly from the virtual sensors [6]. This also ensure that the quality of the labelled data is under control to maximize the accuracy of the DNN.

In general, simulation platform for automated vehicle can be categorized into three types: Low-level simulator that used fixed-base (FB) simulator with fixed user display; Mid-level simulator consists of advanced imaging technologies such as large projection screen, real vehicle, and a motion base; High-level simulator is accelerated using motion platform with more than 6 degree of freedoms [7]. Car Manufacturers such as General Motors has been using simulator for testing of design concept, advanced driver assistance and safety system beginning in the 1960s [8]. There have been lots of development of virtual testing platforms in the past few years [9-14]. Most of them were built on top of physics/dynamic engines such as Unity game engines and Unreal Engine to create realistic graphics, sensors, vehicle dynamics and environments. However, these platforms lack the capabilities of configurable vehicle dynamics model, personalized driver model, and vulnerable road user's behaviors that are essential for automotive testing. A widely used software in the automotive industry that satisfy these requirements is IPG CarMaker [15]. Although it is a closed-source commercial software, it is open for integration of C code, MATLAB/Simulink models [16] as well as interfacing using Robot Operating System (ROS) [17] and Python through the CarMaker's Applications Online (APO)[18] or TCP/IP socket. This enable the software to be used not only as a virtual simulator for AV testing but also as a driving simulator for

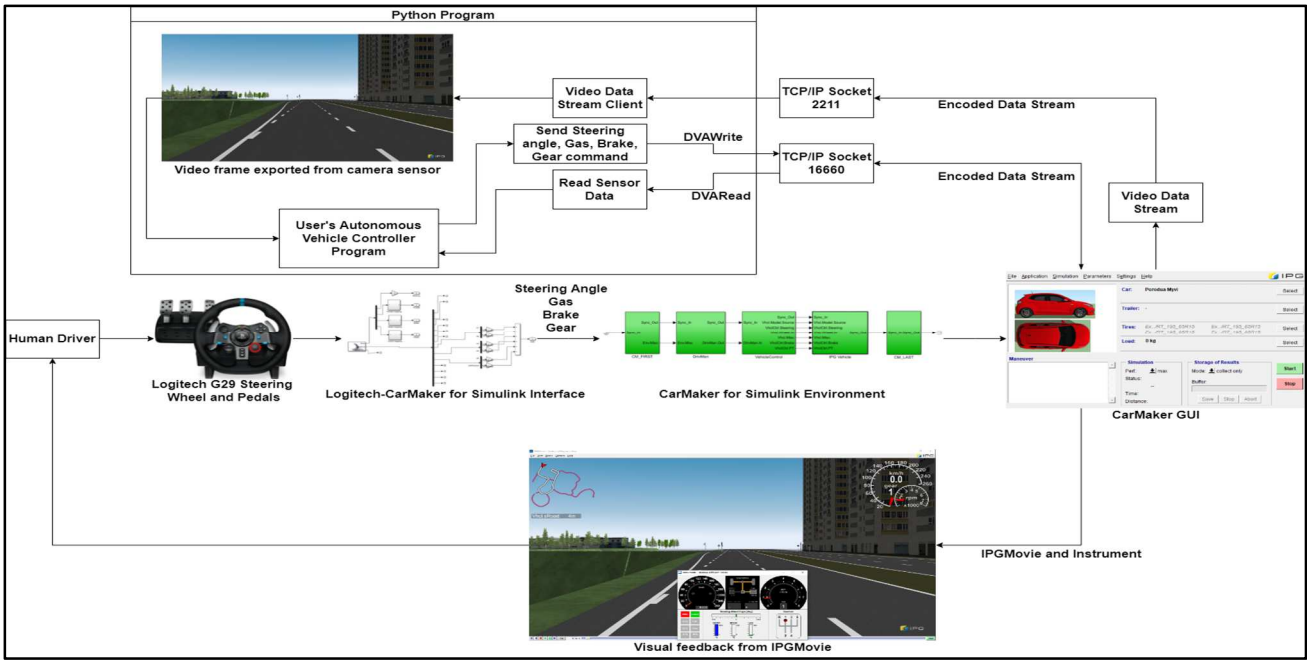

Fig. 1. Virtual platform architecture using Simulink and Python

human's driving data collection for development of driver model and controller for the AV's system. Besides, most of the driving platform are focusing on the road environment in Western countries and the environment does not consider the traffic conditions in developing countries such as Malaysia.

In this paper, the main goal of this study is focused on the system configuration of human-in-the-loop 3D virtual simulation platform for performance evaluation of autonomous vehicles. The system is designed as an end-to-end autonomous vehicle development platform that can provide support from generation and collection of human's driving data for development of controllers to safety testing of the controllers developed in critical scenarios in the virtual platform. The remaining of the paper is organized as follow: In Section 2, the architecture of the virtual simulation platform is described. In Section 3, the configuration of the IPG CarMaker is discussed in detail. In Section 4, the application of the virtual platform using human-in-the-loop simulation is presented. In Section 5, the conclusion of this study is presented.

II. Human-In-The-Loop Simulation Architecture

The high-level overview of the virtual platform is illustrated in Fig. 1. IPG CarMaker is the main component in the architecture as it provides the 3D virtual environment, vehicle dynamic modelling and sensors simulation. Simulink is used to design sub-models and algorithm such as ADAS controller and driver models, which can then integrate them in the CarMaker. In this paper, Simulink model is used to create an interface of Logitech G29 Driving Force Steering Wheel and Pedals kit to CarMaker. Meanwhile, Python programming is used in this study to read data and send control command to the ego vehicle. Furthermore, video stream of the virtual camera on an ego vehicle from CarMaker also been captured using Python and this is important for vision-based autonomous driving vehicle. In the next section, the detail configuration of IPG CarMaker such as the vehicle model and the sensors configuration alongside with the environment model and scenarios configuration are explained. Then, the details configurations of interfacing the Logitech G29 Steering Wheel and Pedal with the IPG CarMaker as well as interfacing IPG CarMaker with Python is explained in this study.

III. IPG CarMaker Configuration

In this section, the detailed configuration of the IPG CarMaker is described. The configuration of the IPG CarMaker can be divided into three sub-sections: vehicle model, environment model and scenario modeling. After that, details on interfacing IPG CarMaker with Simulink and Python will be discussed.

A. Vehicle Model

IPG CarMaker has a range of demo vehicle models available for users, such as Honda CVT, BMW, Audi TT and etc. However, the vehicle model that required in this study is not listed in the available vehicle model since the simulation required Malaysian national vehicle. As each vehicle has its own dynamics properties, it is important to have a customize vehicle dynamics model that can be modified in the software, so that the virtual vehicle model can represent the actual vehicle in the virtual environment. IPG CarMaker provides built-in vehicle dynamics model which the vehicle's parameters can be configured as shown in Fig. 2.

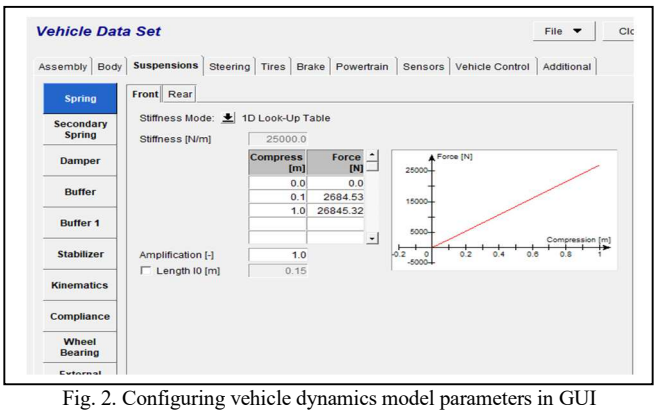

Fig. 2. Configuring vehicle dynamics model parameters in GUI

In this study, Malaysian National vehicle, Perodua Myvi car was used as the vehicle model as shown in Fig. 3. The dimension of the vehicle, suspension, powertrain, gearbox, engine, steering type are configured according to the actual Myvi car so that the vehicle model can represent the dynamics of actual vehicle configuration.

Fig. 3. Malaysian 2nd National Vehicle - Perodoa Myvi 3D model.

B. Environment model

In order to test the vehicle in the driving simulator, a virtual test track is required. IPG CarMaker provides built-in Scenario/Road editor, which can be used to develop 3D virtual environment model. The test tracks, static objects such as trees, traffic lights, sign boards, buildings, and dynamic traffic objects such as vehicles and pedestrians are defined in the Scenario/Road editor based on the selected actual road environments in Malaysia. An example of virtual road environment developed based on road environment of University of Nottingham Malaysia is shown in Fig. 4. Additionally, the starting point and end point of the test track is also defined in the Scenario/Road editor.

C. Sensor module configuration

IPG CarMaker provides a range of sensor models that can be mounted on the vehicle model, such as RADAR, LiDAR, camera, inertial sensor, road preview sensor and etc. These sensor models can be customized based on real sensor's parameters and sensor's noise can be generated as well to simulate the behavior of a real sensor. Sensor model can be mounted on the ego vehicle by configuring the model's attributes, which include relative distance, azimuth angle and orientation. In this paper, four camera sensors are mounted

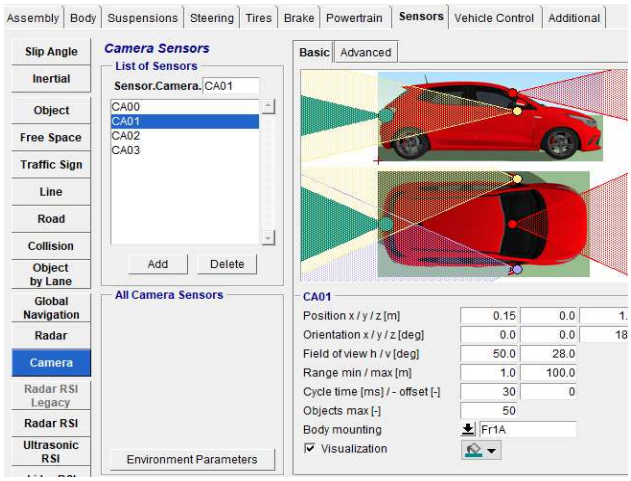

Fig. 5. Camera sensors module configuration on ego vehicle

on the ego vehicle to capture the front, rear, left and right-side mirror images, as shown in Fig. 5.

D. CarMaker for Simulink

Simulink is one of the most popular tools used for Model Based Software Development in the automotive industry. Therefore, IPG CarMaker also provides advanced software package for easier integration of custom Simulink models such as controller and driver models into the software as shown in Fig. 6. In this study, instead of developing the autonomous vehicle controller, an integration of external driver model into the IPG CarMaker software only is mainly focused. This is to ensure that the autonomous vehicle able to receive driving inputs from the human driver during emergency takeover conditions.

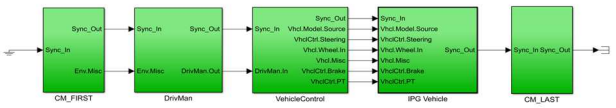

Fig. 6. CarMaker built-in Simulink model blocks.

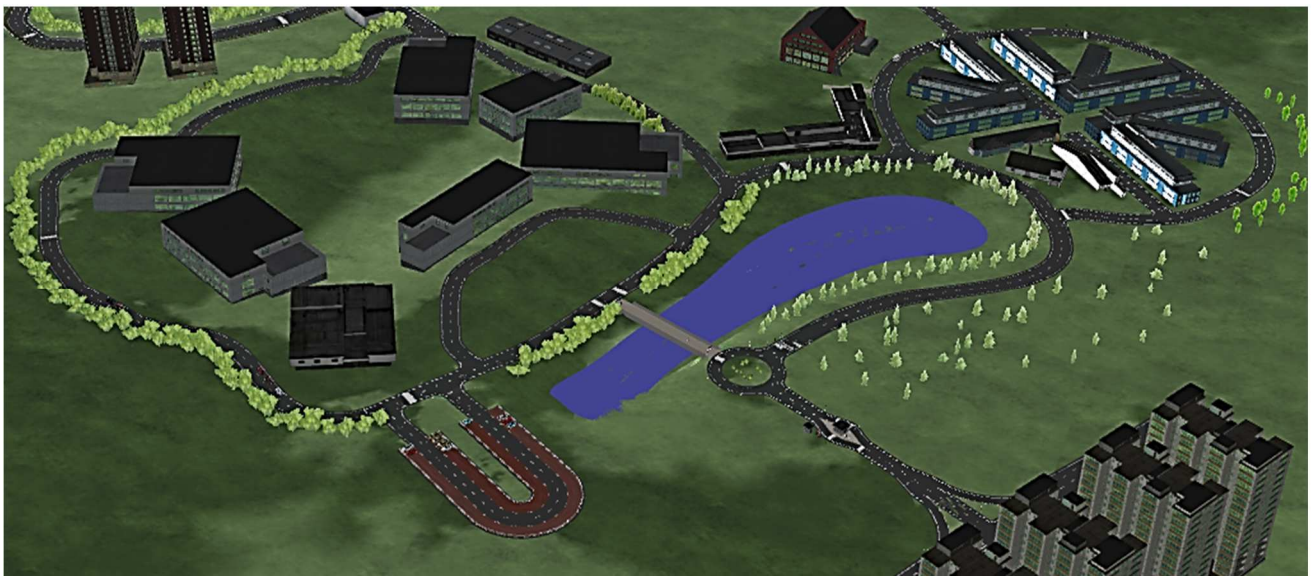

Fig. 4. University of Nottingham Malaysia 3D model

In order to achieve this goal, a human interface hardware device is required to allow human's control over the software. The hardware device used is the Logitech G29 Driving Force Steering Wheels and Pedals to enable human control of vehicle steering angle, gear, gas pedal position and brake pedal position as shown in Fig. 7. The Steering wheel allows 900 degrees lock-to-lock rotation and force feedback to provide realistic driving experience. This is important to ensure that the human's action data collected when driving in the simulator are close to actual actions taken by same driver while driving in real life environment.

Fig. 7: Logitech G29 Driving Force Steering Wheel, Pedals and Gear Shifter

To interface the Logitech G29 kit to IPG CarMaker, a Simulink model that maps the control signals from the Logitech G29 kit need to be developed. The control signals are channeled into steering angle, gear, gas pedal and brake pedal signals in the IPG CarMaker as shown in Fig. 8. By default, IPG CarMaker used built-in driver model, known as IPG Driver to drive the virtual vehicle. In order to bypass the IPG Driver and use external driver's control input, the output of the default IPG's Simulink driver model is replaced with the corresponding outputs of the external driver model, as shown in Fig. 9.

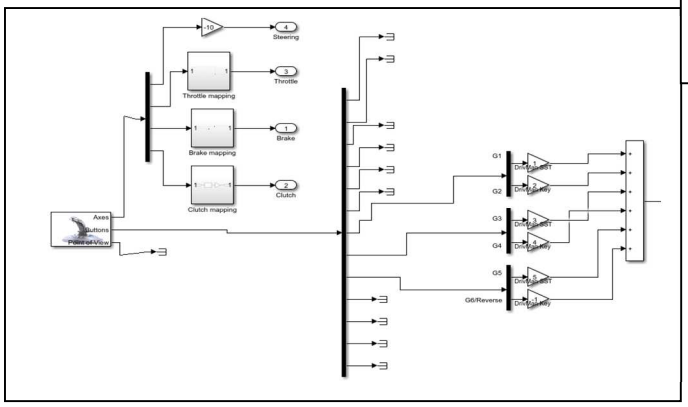

Fig. 8. Logitech G29 Driving Force Simulink model

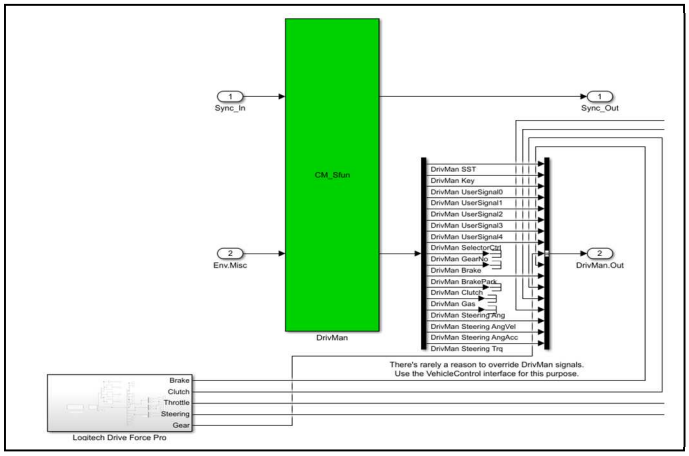

Fig. 9. Logitech G29 interface with CarMaker for Simulink.

E. *Interface Python with CarMaker*

Simulink works great with IPG CarMaker for model-based development and simulation. However, when comes to developing machine learning based self-driving model prototypes, Simulink is not as convenient as Python due to more powerful and easy to use libraries such as TensorFlow and PyTorch, which have vast community resources and supports. However, there is no direct method to integrate Python with IPG CarMaker such as shown in IPG CarMaker for Simulink. On the other hand, IPG CarMaker allows using TCP/IP sockets for communication. By establishing a TCP/IP bridge between the IPG CarMaker and the Python program, the program can access and manipulate the quantities in IPG CarMaker via Direct Variable Access (DVA) during simulation. Once the TCP/IP bridge is established, the program can write the value of a quantity by simply transmitting a DVAWrite message encoded with the name of the quantity, duration and the desired parameters' values to IPG CarMaker. Whereas to read a quantity can be done easily by simply sending a DVARead message encoded with the name of the quantities. Then, IPG CarMaker will return the instantaneous value of quantity specified as shown in Fig. 10.

Fig. 10. Reading vehicle speed, brake pedal, gas pedal and steering angle quantities from IPG CarMaker using Python

Besides accessing and manipulating the quantities of IPG CarMaker, Video Data Stream (VDS) generated by virtual camera mounted on the ego vehicle can be exported over TCP/IP and read by Python program, which is very important for vision-based autonomous vehicle controllers that take in images as data input. Unlike accessing the quantities of the IPG CarMaker, a configuration file for IPGMovie is required

CarMaker - VDS.cfg

File Edit Section

```
nViews 2
SyncWindow   0.1
Socket  2211
ExclusiveMode 1
# Front camera
0 {
Width 640
Height 480
Export {rgb}
VPtOffset_x 2.5
VPtOffset_y 0
VPtOffset_z 1.3
FrameRate 25
}
# Rear camera
1 {
Width 640
Height 480
Export {rgb}
VPtOffset_x 0.15
VPtOffset_y 0
VPtOffset_z 1.0
zrot 180
FrameRate 25
}
```

Fig. 11. Example of VDS configuration with two cameras (front and rear) with TCP/IP socket of 2211

Fig. 12. Video stream data exported from four cameras mounted on ego vehicle, where frame 1 is from front camera, frame 2 is from rear camera, frame 3 is from right side mirror camera and frame 4 is from left side mirror camera.

before using the VDS to define the TCP/IP socket for streaming and camera's configuration such as resolution, framerate, relative distance and orientation of the camera.

Additionally, more than one camera can be defined in the same configuration file as shown in Fig. 11. After the configuration is completed, once the simulation starts, in each frame cycle, each VDS camera frame will be rendered one by one before start of the next frame cycle. Therefore, increasing VDS exported will increase the latency of each frame cycle, which in result will slow down the simulation speed. Each transmitted image is encoded with a header information string that have a format as follow:

"*VDS <label of camera> <image format> <time> <width>x<height> <length of the image in bytes>\n".

Python is used to decode and extract the image data from the VDS received. Once the image data is extracted, the image data is reshaped into RGB three color channels image based on the width and height information decoded from the header information string. Finally, the image can be used according to user's requirement such as feeding into a vision-based controller's input and displaying on the screen using OpenCV library as shown in Fig. 12.

IV. Application of the Virtual Platform

In previous section, the development and design of the virtual simulation platform are shown. In this section, the application of the virtual simulation platform for collecting driving data and testing of a vision-based machine learning model will be discussed in detail.

A. Data Collection of Human's Driving Behavior

Human's driving data is very important for human driver modeling, self-driving model based on imitation learning. Before simulation start, it is important to know what data are required. In this study, steering wheel angle, gas pedal, brake pedal and vehicle speed are selected as simulation output because these quantities are the most used data types in the automotive industry. Besides, the sampling rate used for data logging is set to 100 Hz so that high resolution data can be collected as shown in Fig. 13.

After initial setup is completed with road model and vehicle model loaded and ready for simulation, the simulation is started, then the human driver can start driving the virtual vehicle from the start point to the end point of the test track defined. Upon starting the simulation, IPG CarMaker will create a folder with the name of the current date, e.g. 20210412. All the *.erg files and a configuration file created on the same day will be stored inside the folder. The *.erg file is created by IPG CarMaker upon each simulation run and

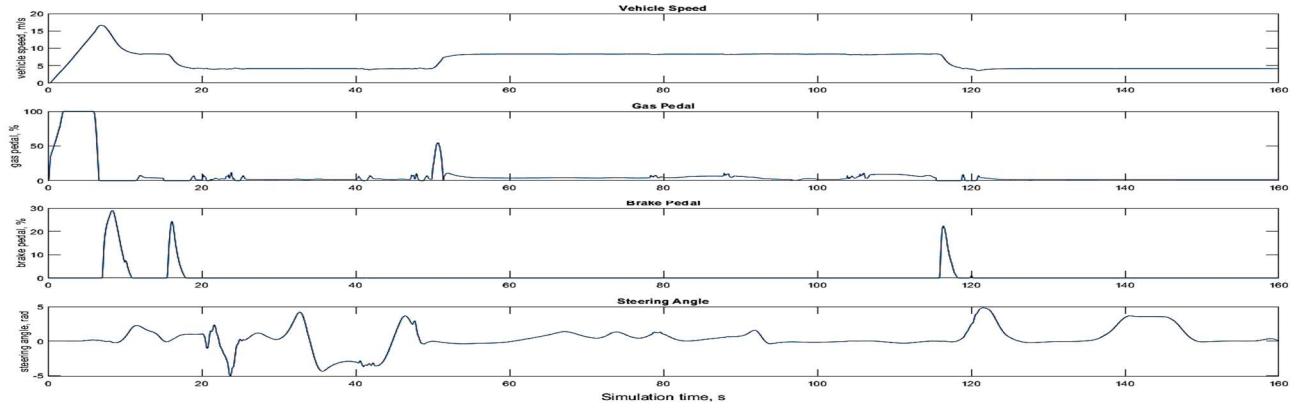

Fig. 14. Output quantities plotted using MATLAB.

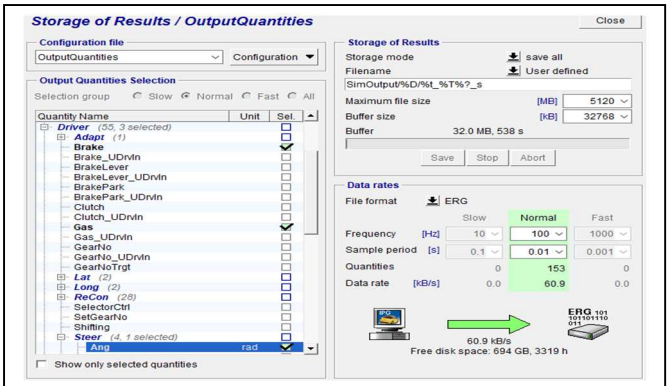

Fig. 13. Configuring output quantities in GUI

contains the data that were selected to store earlier, whereas the configuration file contain the parameter used for simulation. The data collected can then be plotted using MATLAB as shown in Fig. 14.

Besides human input data collection, the video data from the virtual camera mounted on the ego vehicle can also be recorded using IPG Movie's built-in tool. The resolution, framerate, quality and file format of the recording can be customized according to user's needs. An example video

Fig. 15. Example screenshot of video recorded from front camera

Fig. 16. Night driving environment in the virtual simulator

frame of the recording is shown in Fig. 15. Furthermore, the virtual simulator can also simulate driving at the night by configuring the environment parameters in the IPG CarMaker. Fig 16 shows the driving simulation at night using IPG CarMaker.

B. Object Detection using Deep Neural Network

From previous section, it can be observed that the video data captured by virtual cameras can be exported using VDS in real-time while simulation is running. In this section, the video data exported is fed into a state-of-the-art object detection deep neural network to detect objects present in the video. The state-of-the-art object detection model used is YOLOv5 [19] which is implemented using Python PyTorch library. YOLOv5 is selected due to advantage in terms of inference speed and model size, at the same time still able to achieve high level of accuracy in detection. This enable the YOLOv5 to run with real-time speed even with a mid-range consumer Graphics Processing Unit (GPU). The YOLOv5l model is used and some object detection results are shown in Fig. 17.

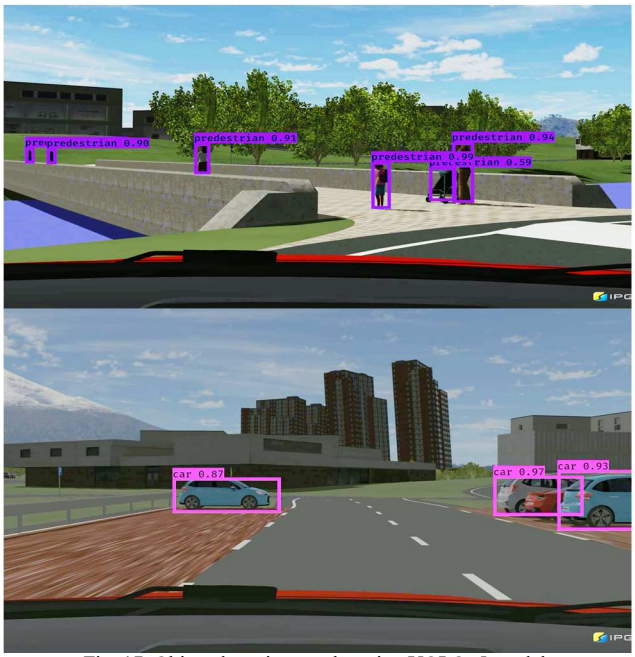

Fig. 17. Object detection results using YOLOv5 model.

By applying object detection to the video stream, the static and dynamic on-road obstacles can be categorized into their respective classes such as car, truck, motorbike, pedestrian, traffic sign etc., and locate them by using bounding boxes. This will provide information of the surrounding and road condition to the level 3 ADAS such as collision avoidance, lane centering, lane change assistance, adaptive cruise control etc. to manoeuvre a vehicle properly [20]. The vision information obtained can also be combined with range sensor such as RADAR and LiDAR for autonomous emergency braking [21] and simultaneous localization and mapping (SLAM) used in self-driving vehicle [22].

V. CONCLUSION

In this study, it can be concluded that an end-to-end virtual simulation platform that can be used for human's driving data collection and testing Level 3 autonomous vehicle's safety system. The system is developed by integrating the IPG CarMaker with Simulink and Logitech G29 Driving Force Steering Wheel and Pedals kit. Using this driving interface, the virtual vehicle can be driven by human on a virtual road network created based on actual road environment. The data collected also can be successfully stored in file for further usage. In addition, we have also shown that Python can be used to deploy machine learning algorithm with IPG CarMaker. Therefore, this virtual simulation platform developed will ease the development of autonomous vehicle and improve safety testing reliability. As for future work, more rare critical scenarios will be developed based on the road environment models to generate driving data using the

driving hardware. Then, the collected data will be used to develop a driver model that can mimic driving behaviour of human drivers using Python which will then integrate with the simulator for testing of level 3 autonomous vehicle.

ACKNOWLEDGMENT

This work is part of the research project entitled "Vehicle-in-the-loop Safety Testing Platform for Autonomous Vehicle using Malaysian Road and Traffic Environment". This research is fully supported by Yayasan Tun Ismail Mohamed Ali Berdaftar (YTI) under Permodalan Nasional Berhad and the grant is led by Assistant Professor Dr. Vimal Rau Aparow. I also would like to thank University of Nottingham Malaysia for continuous support to conduct research in this area.

REFERENCES

[1] W. Huang, K. Wang, Y. Lv and F. Zhu, "Autonomous vehicles testing methods review," *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, Brazil, pp. 163-168, 2016, doi: 10.1109/ITSC.2016.7795548.

[2] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?," *Transportation Research Part A: Policy and Practice*, vol. 94, 2016, pp. 182–193.

[3] H. Caesar *et al*., "nuScenes: A Multimodal Dataset for Autonomous Driving," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, pp. 11618-11628, 2020, doi: 10.1109/CVPR42600.2020.01164.

[4] P. Sun *et al*., "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, pp. 2443-2451, 2020, doi: 10.1109/CVPR42600.2020.00252.

[5] S. Agarwal, A. Vora, G. Pandey, W. Williams, H. Kourous, and J. McBride, ''Ford multi-AV seasonal dataset,'' *Int. J. Robot. Res*., vol. 39, no. 12, pp. 1367–1376, Oct. 2020

[6] N. Ruiz, S. Schulter and M. Chandraker, "Learning to Simulate," in *International Conference on Learning Representations (ICLR)*, 2019. [Online]. [Accessed: 12-Apr-2021].

[7] S. Yuan, B. Monica and C. Giandomenico, "User Studies by Driving Simulators in the Era of Automated Vehicle," *Computer-Aided Design and Applications*, 2020, doi: 10.14733/cadaps.2021.211-226.

[8] G. Bertollini, C. Johnston, J. Kuiper, J. Kukula, K. Malgorzata and T. William, "The General Motors Driving Simulator," *SAE transactions*, pp. 54-67, 1994.

[9] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity Visual and Physical Simulation for Autonomous Vehicles," in *Field and Service Robotics*, 2017. [Online]. [Accessed: 12-Apr-2021].

[10] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez and V. Koltun, "CARLA: An Open Urban Driving Simulator," In *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1 – 16, 2017.

[11] Waymo, "Waymo Safety Report," 2020. [Online]. [Accessed: 12-Apr-2021].

[12] G. Rong *et al*., "LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving," *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, Rhodes, Greece, pp. 1-6, 2020, doi: 10.1109/ITSC45102.2020.9294422.

[13] NVIDIA, "Self-Driving Safety Report," 2021. [Online]. [Accessed: 12-Apr-2021].

[14] Uber, "A Principled Approach to Safety," 2020. [Online]. [Accessed: 12-Apr-2021].

[15] IPG Automotive GmbH, "CarMaker," *CarMaker | IPG Automotive*, 31-Mar-2021. [Online]. [Accessed: 12-Apr-2021].

[16] IPG Automotive GmbH, "CarMaker APO Manual Version 1.36," pp. 1 – 69, 2020.

[17] M. Quigley *et al*., "ROS: an open-source Robot Operating System," *ICRA Workshop on Open-Source Software*, vol. 3, pp. 5, 2009.

[18] V. R. Aparow, A. Choudary, G. Kulandaivelu, T. Webster, J. Dauwels and N. d. Boer, "A Comprehensive Simulation Platform for Testing Autonomous Vehicles in 3D Virtual Environment," *2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*, Singapore, pp. 115-119, 2019, doi: 10.1109/ICMSR.2019.8835477.

[19] G. Jocher *et al.*, "ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations," *Zenodo*, 11-Apr-2021. [Online]. [Accessed: 12-Apr-2021].

[20] A. Gupta, A. Anpalagan, L. Guan and A. S. Khwaja, "Deep Learning for Object Detection and Scene Perception in Self-driving Cars: Survey, Challenges, and Open Issues," *Array*, vol. 10, 2021. [Online]. [Accessed: 15-Oct-2021].

[21] W. Hulshof, I. Knight, A. Edwards, M. Avery and C. Grover, "Autonomous emergency braking test results", *Proc. International Technical Conference Enhanced Safety of Vehicles*, pp. 1-13, 2013.

[22] G. A. Kumar, J. H. Lee, J. Hwang, J. Park, S. H. Youn and S. Kwon, "LiDAR and Camera Fusion Approach for Object Distance estimation in Self-driving Vehicles," *Symmetry*, vol. 12, no. 2, pp. 324, 2020.