2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Autonomous Vehicle Testing and Validation Platform: Integrated Simulation System with Hardware in the Loop*

Yu Chen[1], Shitao Chen[2], Tangyike Zhang[3], Songyi Zhang[4] and Nanning Zheng[5], Fellow, IEEE

Abstract— **With the development of autonomous driving, offline testing remains an important process allowing low-cost and efficient validation of vehicle performance and vehicle control algorithms in multiple virtual scenarios. This paper aims to propose a novel simulation platform with hardware in the loop (HiL). This platform comprises of four layers: the vehicle simulation layer, the virtual sensors layer, the virtual environment layer and the Electronic Control Unit (ECU) layer for hardware control. Our platform has attained multiple capabilities: (1) it enables the construction and simulation of kinematic car models, various sensors and virtual testing fields; (2) it performs a closed-loop evaluation of scene perception, path planning, decision-making and vehicle control algorithms, whilst also having multi-agent interaction system; (3) it further enables rapid migrations of control and decision-making algorithms from the virtual environment to real self-driving cars. In order to verify the effectiveness of our simulation platform, several experiments have been performed with self-defined car models in virtual scenarios of a public road and an open parking lot and the results are substantial.**

I. INTRODUCTION

Online and offline tests are two major approaches used in order to verify and optimize the perception and control algorithms of self-driving cars. Motivated by the necessity for new applicable technologies in autonomous driving, offline testing is transforming into a mandatory step for pre-assessment of top-level control algorithms, before the real interface application used towards a fully matured level of autonomous driving [1], [2], [3]. Usually on-site inspection of physical vehicle behavior is an expensive and time-consuming attempt that only authorizes a limited number of scenarios to be tested. By contrast, a simulation test of vehicle control using mounted sensors and comprehensive environments provides a reasonable and cost-effective option. Besides, previous research and experiments on various intelligent algorithms involved in autonomous driving, in order to achieve optimal validation results and objective performance analysis, require an efficient test and validation platform. Therefore, an efficient simulation platform with an organized architecture has been proposed, of which the detailed workflow is shown in Fig. 1. This proposed platform contains a software simulation interface and a hardware ECU interface which have been combined to form a closed test and validation loop with high efficiency. For one thing, the software simulation interface mainly consists of 3 layers: (1) the kinematic vehicle designing layer for vehicle motion control, (2) the multiple sensors' simulation layer to percept environments and assemble data for autonomous vehicles, (3)the virtual testing environment layer to simulate real scenarios and also extreme testing situations. Furthermore, the hardware ECU interface contains an essential electronic control unit layer for the overall hardware control.

The software simulation interface comprises a variety of softwares which can simulate any of the required objects to be evaluated on a virtual test bed. The software simulation interface of the proposed platform is centered around the Robot Operating System (ROS) and its embedded software Gazebo [4], [5]. ROS is a robotic software platform that provides similar operating system functions for heterogeneous computer clusters. It offers customary operating system services such as hardware abstraction, underlying device control, common feature implementation, inter-process messaging and packet management. A node is the core component of ROS graph architecture. It's usually a short piece of code scripted in programming language Python or C ++ to perform a relatively simple task or process. Multiple nodes communicate messages to each other and can be independently started or terminated. Therefore, nodes at different points of a process can accept, publish, and aggregate various categories of information for sensing, controlling, status monitoring or any other specific purpose. Further to this, Gazebo is a ROS built-in 3D simulation software that helps to accurately construct and evaluate the kinematics of robots in complex indoor and outdoor environments. It offers high-fidelity physical simulations, a large set of sensors, and numerous procedural and user-facing interfaces.

Further to the software simulation interface, a hardware ECU interface [6] based on HiL simulation system has been developed. Common offline tests merely tend to evaluate independent algorithm modules or functions, making it problematic to easily establish the link between the real car and the virtual environment easily. However, in this paper the HiL system is one kind of semi-simulation system with actual controllers but virtual objects and environments [7], [8], [9]. Real sensors and loads are replaced by virtual ones to simulate the real-time physical relationship between actuators and sensors on a simulated car model [10]. This subsequently allows the ECU to operate in an electrical environment that is considerably close to that in a real

*This research was partially supported by the National Natural Science Foundation of China (No. 61773312, 61790563), the Programme of Introducing Talents of Discipline to University (No. B13043).

Yu Chen[1], Shitao Chen[2], Tangyike Zhang[3], Songyi Zhang[4] are with Institute of Artificial Intelligence and Robotics in Xi'an Jiaotong University, Xi'an, Shannxi, P.R.China e-mail:{alan19960212, chenshitao, ericzhang, zhangsongyi}@stu.xjtu.edu.cn

Nanning Zheng[5] is the director of Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, Shannxi, P.R.China Correspondence: nnzheng@mail.xjtu.edu.cn

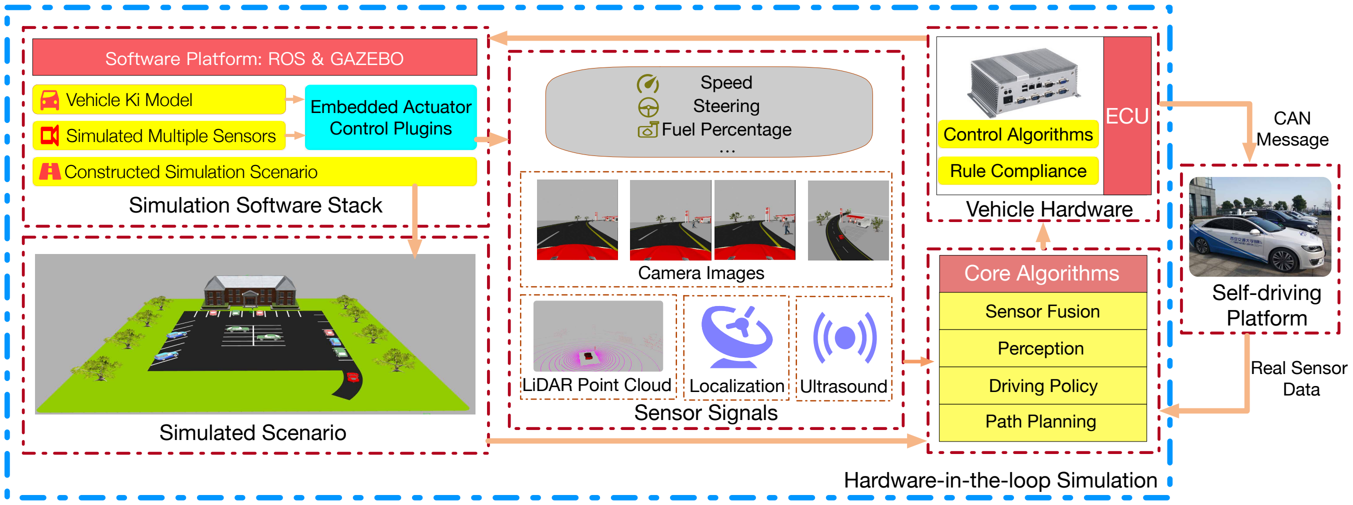

Fig. 1. Workflow of our HiL-based autonomous vehicle simulation platform. The simulation interface exports sensor signals and vehicle states to a hardware control unit. Sensor data comes from virtual sensors like LiDAR (Light Detection and Ranging), camera, millimeter radar and all other sensor devices. Vehicle states include throttle, brake, steering and fuel percentage reports for top-level control of ECU. The ECU then processes the transformed data with core algorithms and sends corresponding actuator commands to a lower simulation interface after making real-time decisions.

vehicle and performs a series of tests to verify whether its performance meets our design requirements. In terms of safety, feasibility and reasonable cost, the HiL simulation test has become an integral aspect of autonomous driving development flow, reducing the number of real road tests, shortening the development time, minimizing the cost and improving the quality of ECU development.

In this paper, the ECU being tested is connected to the simulation interface for all-round and systematic tests while realizing a closed-loop simulation system. The simulation interface exports sensor signals and vehicle states in real-time to the ECU via CAN (Controller Area Network) bus devices for data transmission. Then the ECU monitors and receives all of the data for further management with core algorithms. Finally, actuator commands for the vehicle are directed towards the simulation interface for further vehicle motion control tests. This system can not only evaluate the environment perception module of a self-driving car, but can also carry out further evaluations of path planning, decision-making and vehicle control algorithms of autonomous vehicles [11]. Thenceforth, once the algorithms have succeeded the feasibility and robustness assessments in a virtual environment, they can be swiftly migrated to a real autonomous platform for minimum error reporting in actual testing conditions. The workflow of autonomous vehicle development guaranties the quality and robustness of our working platform whilst also avoiding massive unnecessary systematic problems, and furthermore, offering a functional development platform in a closed loop. This paper aims to start with some related research works, following by an introduction to our general platform. Then we'll proceed to the realization of detailed layers. Following this, we'll verify the effectiveness of our platform with experiments of planning and decision-making algorithms. Lastly, we'll summarize and look ahead to our future work.

II. RELATED WORK

The offline simulation approach is now widely recognized and applied to automotive systems thanks to its high efficiency and low cost. Multiple modules utilize this approach in order to develop and evaluate autonomous softwares without the need for an actual car [12], [13], [14]. Meanwhile, plenty of work has been conducted around HiL systems in order to optimize offline test results and improve the control ability, as is indicated in [6]. [15] evaluated their car's engine control with simulated combustion engine in a closed loop. In [16], Oh *et al.* concluded that the HiL system is efficient for vehicle dynamic component testing. However, achieving flexible simulation interface requires various powerful offline test softwares. Technically, these softwares can be divided into two varieties: one aims to test the performance of an autonomous vehicle's different parts with real data playback. The other one is mainly used for initial development of control and planning algorithms. It is based on synthetic data simulation of vehicle model, sensors and the environment. For this paper we have chosen Gazebo for its precise kinematic modelling and advanced 3D rendering. Similar open-source softwares like Euro Truck Simulator 2 and The Open Racing Car Simulator (TORCS) [17] are frequently used for training and strengthening control algorithms. Udacity also opened its autonomous vehicle simulator to the public recently. Those interested can use this simulator to learn how to drive a car with deep learning. Apart from applications on research work, there are also simulation softwares for commercial purposes of automotive industry, like CarSim for vehicle dynamics [18], PanoSim for complex vehicle models and traffic scenarios, PreScan for Advanced Driver Assistance Systems (ADAS) [19] development as well as vehicle-to-vehicle (V2V) [20] and vehicle-to-infrastructure (V2I) [21] applications. From training to testing, all these simulation platforms are making the idea of autopilot more and more sophisticated and palpable.

III. Module Design

A. Software Simulation Interface and Hardware ECU Interface

1) Software Simulation Interface: The 3D simulation software Gazebo is used for the construction of car models using the Unified Robot Description Format (URDF). This is a standardized format in ROS that permits the description of all elements that specifies the motion and dynamics of a single robot. Rigid controlled parts like car body, steering wheel, brake, throttle and other essential components can be described here. Furthermore, physical parameters of controlled objects like mass, inertia, collision, torque, etc., are defined in order to ensure the model contains maximum necessary properties of a real car. Moreover, in order to efficiently ensure the safety of a self-driving car, parameters like velocity limit, steering angle limit, maximum tire force and torque are also specified in the construction. The kinematic car model that we created is shown in Fig. 2.

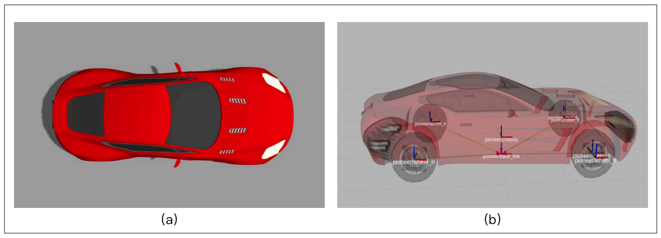

Fig. 2. Simulated kinematic car model. (a) Car model in Gazebo. (b) Transformation of vehicle coordinate systems in RVIZ

2) Hardware ECU Interface: In terms of hardware control interface, the ECU occupies all possible algorithm tests and computational labor for the vehicle dynamic control. The closed-loop connection between the electric control unit and the simulation interface has significantly enhanced the improvement and validation efficiency of core algorithms.

To achieve average movement of our simulated car and to also achieve the linear and angular velocity configuration for the car in simulation interface, position and steering controllers are necessary. Vehicle linear motion is easier to design and control in comparison to steering motion under indispensable safety requirements. In this paper we have presented and applied the Ackermann steering geometry [22] with independent control of four wheels in order to achieve safe steering performance.

Ackermann steering geometry was designed to avoid the side slipping of tires due to the different turning radii of front steering wheels when the car turns along the corners. According to this geometric steering design of the car, the steering angle of the inner front wheel should be several degrees larger than that of the outer front wheel. This is in order to acquire a common center point for all wheels' axles when turning. What's more, since two rear wheels are fixed, the center point must be on a line extending from rear axles.

As shown in Fig. 3, point O is the car's steering center, L is the car's wheelbase, T represents car tread, d signifies the distance between car mass center and rear wheel axle, σ is target steering angle and v is target velocity.

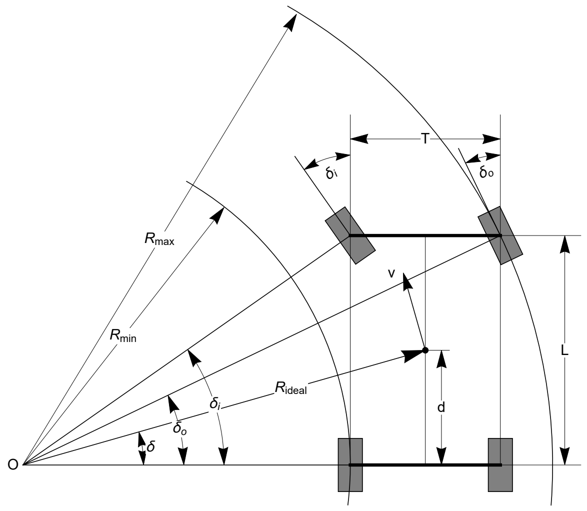

Fig. 3. Ackermann Steering Geometry. When the car is turning, the outer front wheel has the maximum turning radius and the inner rear wheel has the minimum turning radius.

The corresponding linear velocities of front left wheel, front right wheel, rear left wheel and rear right wheel are symbolized respectively as v_{fl}, v_{fr}, v_{rl} and v_{rr}. According to the Ackermann geometric movement relationship, we can obtain the wheel velocity equations as follows:

$$\begin{cases} v_{\mathrm{fl}} = \frac{R_{\mathrm{fl}} v}{R_{\mathrm{ideal}}} \\ v_{\mathrm{fr}} = \frac{R_{\mathrm{fr}} v}{R_{\mathrm{ideal}}} \\ v_{\mathrm{rl}} = \frac{R_{\mathrm{rl}} v}{R_{\mathrm{ideal}}} \\ v_{\mathrm{rr}} = \frac{R_{\mathrm{rr}} v}{R_{\mathrm{ideal}}} \end{cases} \tag{1}$$

What's more, as the inner steering angle is σ_i and the outer steering angle is σ_o. we can get their relation with different radii R_{ideal}, R_{fl}, R_{fr}, R_{rl} and R_{rr} as follows:

$$\begin{cases} \tan\delta_i &= \frac{L}{R_{\mathrm{rl}}} \\ \tan\delta_o &= \frac{L}{R_{\mathrm{rr}}} \\ R_{\mathrm{ideal}} &= \frac{d}{\sin\delta} \\ R_{\mathrm{fl}} &= \sqrt{R_{\mathrm{rl}}^2 + L^2} \\ R_{\mathrm{fr}} &= \sqrt{R_{\mathrm{rr}}^2 + L^2} \\ R_{\mathrm{rl}} &= R_{\mathrm{ideal}}\cos\sigma - \frac{T}{2} \\ R_{\mathrm{rr}} &= R_{\mathrm{ideal}}\cos\sigma + \frac{T}{2} \end{cases} \tag{2}$$

According to all of the given equations in (1) and (2), the ECU will calculate the corresponding target linear velocity values v_{fl}, v_{fr}, v_{rl}, v_{rr} as well as the steering angles σ_i, σ_o of two front steering wheels. Thenceforward, through the CAN bus communication, throttle, brake and steering commands will be sent to the lower simulation interface for corresponding vehicle actuators.

B. Sensors Simulation

It is appropriate that an efficient simulation platform stipulates the status of all sensors and relative recordings of all previous sensors' data. Hence, in our proposed simulation platform we have also simulated the real sensors mounted on our self-driving car with pre-existing or self-defined plugins, which can be combined with ROS messages and services for motor input and sensor output. It is practical to load, unload a simulated sensor or update its configuration parameters. Besides, the transformation of coordinate systems between sensors or between sensors and the car is easier to be regulated than in real environmental circumstances. The related sensors are listed as shown in Table I.

TABLE I
LIST OF SIMULATED SENSORS

Module	Sensor
Perception	Stereo camera/ Monocular camera
	LiDAR
	Millimeter-wave radar
	Ultrasonic radar
Navigation	Global Positioning System (GPS)
	Inertial Measurement Unit (IMU)

1) Stereo camera/ Monocular camera: Cameras are habitually used in autonomous driving for: lane detection, obstacle detection and traffic sign recognition (e.g., identification of traffic lights and speed limit signs). In terms of simulation, we can adjust a camera's position and rotation angle relative to the car, as well as physical elements such as horizontal field angle, sensing range and distortion. What's more, even unsystematic Gaussian noises can be added to received images or any other simulated sensor signals.

2) LiDAR: A LiDAR is primarily used for: (1) roadside detection, including lane detection, (2) the recognition of both static and dynamic objects, (3) the enhancement of the positioning of Simultaneous Localization and Mapping (SLAM), and (4) 3D environmental map creation. Through laser scanning, the 3D model of the surroundings can be acquired. Through the comparison of changes of a previous frame and its subsequent frame, we are able to detect surrounding vehicles and pedestrians more certainly. This is achieved thanks to relevant algorithms. Variable parameters for a LiDAR, such as laser number, sensing range, sampling rate, horizontal and vertical field angles can be easily defined for simulation.

3) Ultrasonic radar: Ultrasonic radar lends support to the visual system and is usually used in low-speed and close-range situations such as parking assistance, blind spot detection and automatic parking. Its scan range, data update rate and Gaussian noise can all be prearranged in Gazebo.

4) Global Positioning System: GPS is a crucial technology for driving positioning. It's a comparatively accurate sensor, and the errors will not escalate over time. Reference latitude, reference longitude, reference heading and reference altitude are four important elements in our GPS simulation.

5) Inertial Measurement Unit: The IMU is a sensor that detects acceleration and rotational movement of a vehicle. The basic inertial sensors include accelerator meters and angular velocity meters. The principle disadvantage of the inertial sensor is that its error increases over time, so we can solely rely on inertial sensors for positioning over a short period of time. In regards to simulation we can also set update rate and Gaussian noise for a simulated IMU.

Since we use accurate physical data from our real car platform for simulation modelling, the simulated car can be used to experiment with sensors in different combinations and configurations in order to help determine the best plan for sensor selection and installation on a real car. The advantages of this are the low costs and high efficiency, which are particularly beneficial since they match real world situations. A variable percentage of random noise can be added to any type of sensor data in order to improve the resemblance to real sensors, making our simulation more practical when porting perception and control algorithms to a real car platform. In addition, received sensor data is essential for multi-sensor data fusion, future path planning and vehicle control modules.

C. Environment Building

In order to verify vehicle perception and control algorithms, massive tests in various scenarios are required. Our simulation softwares enable the reproduction of every environment similar the real world. Further to this, the tests also assess certain extreme conditions that need to be taken into consideration, such as collision and high speed occasions. The simulated world includes an assortment of robots and objects, such as pedestrians, roads, and traffic lights. Global parameters like gravity, ambient light, wind speed and other physical properties can also be defined. Apart from using pre-defined object models from Gazebo model database, we principally use two methods to build simulation environments:

1) Design Self-defined Objects with 3D Modelling Softwares: We can construct desired objects with professional 3D modelling softwares like Solidworks. Subsequently we need to compile 3D object models into formats that are supported by simulation softwares. Lastly we will load the models into the virtual environment. Specific elements of an object, like position, height, scale and rotation angle are all variable within the simulation environment.

2) Create Large-scaled Scenario with OpenStreetMap + Sketchup: [1]

Different from modelling and combining different objects together to form a simulated world, OpenStreetMap is a powerful and helpful tool for transforming the map of a location to a 3D model. Acting as a platform for the collection, development and distribution of geographic data [23], it enables anyone to easily and quickly access and share the data. It further allows us to easily simulate the geographical

[1]For more information, please visit https://blog.sketchfab.com/creating-interactive-3d-maps-with-openstreetmap-and-sketchfab/

environment of different cities without being limited by the current location [24]. OpenStreetMap supports various data output formats. As shown in Fig. 4, the map of specific areas such as a campus or even a city from the real world can be exported to suitable formats. We will subsequently use the 3D modelling software Sketchup to automatically add visual textures for various objects like buildings, trees, roads and other object models online. Further procedures are similar to processing a single model as introduced above. Finally, we can import the 3D map as a complete scenario for simulation in the desired geographic space.

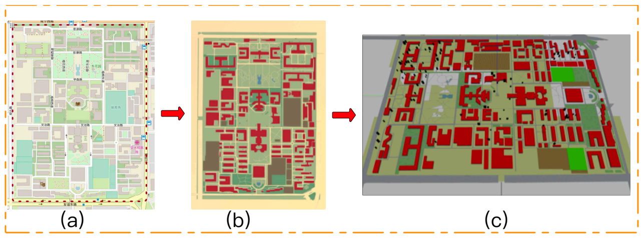

Fig. 4. An example of virtual scenario creation with OpenStreetMap and Sketchup. (a) Map of Xi'an Jiaotong University, Shannxi, China. The selected area is exported in osm format by using OpenStreetMap. (b) Further processed map in Sketchup for better visualization effects. (c) Successfully loaded 3D university map in Gazebo.

D. Simulation of Multi-agent Interaction

Variable vehicle kinematic models and simulation environments authorize us to realize a multi-agent system in order to simulate the dynamic characteristics of complex traffic scenes. The multi-agent system is an open computing system consisting of multiple intelligent agents that interact in one environment.

In this paper, with reference to Fig. 5, we consider an agent as an individual unit to represent a single vehicle, and a multi-agent system is designed to achieve possible collision avoidance, reasonable planning and simultaneous control through V2V and V2I communications. In our system, vehicles can spontaneously enter or leave the multi-agent network at any time after receiving the command. Each agent runs in accordance with critical traffic rules, specific physical parameters and motion commands given to them. In the simulated world, each vehicle runs asynchronously, therefore its own goals and behaviors are not restricted by other agent members.

The active interaction between multiple agents introduces autonomous features into the developing system. In the end the system creates different dynamic traffic scenarios over time (e.g., lane congestion). Our simulation of these scenarios can be used to assist reproduction, analysis and the judgment of certain complex traffic phenomena that people usually cannot practically observe in the real world. The development of automotive system and validation of control algorithms will also be enabled in a higher autonomic level of scene recognition, path planning and decision-making. Vehicles in the system can mutually communicate with each other and work together to coordinate their capabilities and

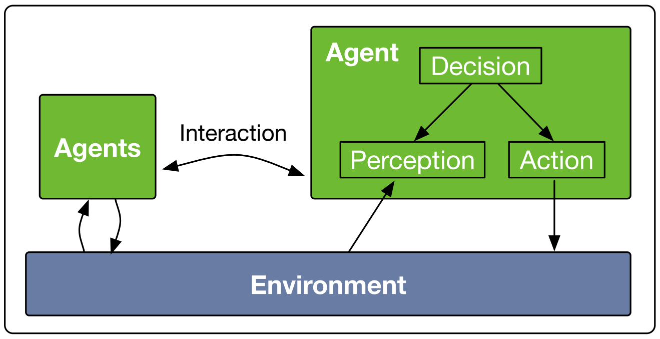

Fig. 5. Multi-agent interaction. A single vehicle is represented as an agent. It interacts with both the environment and other vehicles.

objectives in order to solve problems that cannot be solved by a single agent. This subsequently improves the capability and effectiveness of handling complex traffic scenarios. Finally a combination with road network, pedestrians, critical traffic rules and other subsystems will constitute an intelligent transportation system (ITS) with the aim of replacing traditional traffic management with higher efficiency and intelligence level.

IV. EXPERIMENTS AND APPLICATIONS

This section aims to present several experiments which will evaluate the effectiveness of our simulation platform and verify its applicability for autonomous vehicle development. First of all, under the circumstance of a virtual public road environment with traffic uncertainty, we will introduce the application of our simulation interface with the verification of vehicle dynamic path planning and intelligent decision-making algorithms. The main purpose of this is to test whether the algorithms can successfully achieve real-time path planning in the presence of multiple dynamic obstacles, and whether the chosen optimal path can meet our vehicle model's physical constraints for safety requirements. Based on that, an experiment involving a car-following model for Adaptive Cruise Control (ACC) application will then be performed to validate this model's reinforcement learning algorithm through a continuous trial and error process. This will be carried out according to the overall efficiency of strategy learning and optimal action-taking for maximum environmental rewards. Finally, we will employ our development platform to a virtual parking scenario, to provide a simulation platform for the verification of the feasibility and stability of related algorithms, achieving parking scene recognition and making optimal parking decisions according to real-time parking environment changes.

A. Path Planning and Decision-making Modules Verification on HiL Simulation Platform

Path planning is a crucial part of driver-less technology and plays an vital role between modules of environment perception and motion control. Based on the environmental data of the sensing system, the car must plan out an accessible and reliable global path in a complex road environment based on a certain performance index (i.e., the highest safety, the lowest energy cost, etc.). This path is supposed to be the shortest collision-free one from the start point to the target point. In this experiment, we load a dynamic traffic scenario in the simulation environment, including variables such as pedestrians, vehicles, traffic signs and the surrounding buildings. The workflow is as indicated in Fig. 6.

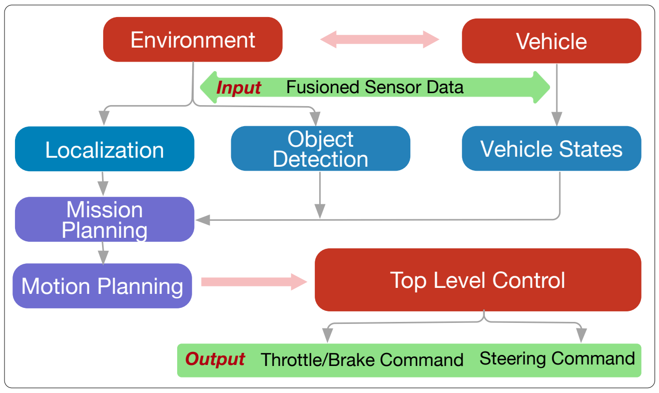

Fig. 6. Workflow of path planning and decision-making modules in the HiL system. The input is fusioned sensor data, and the output is throttle, brake and steering commands from top-level control.

Our whole process is based on an organized mechanism with specific traffic rules. After receiving a given target point, the mission planning algorithm will use multi-sensor fusioned data to map out a viable path based on given traffic rubrics and the car's current position in the present environment. Other environmental factors, such as the shape and location of obstacles, the width of the road and the curvature radii of curves are not taken into consideration during this stage. Next, when the car starts to get ahead under uncertainty of environmental factors, the motion planning needs to be triggered. In this instance, the input of the local planning algorithm includes not only the surrounding environment information acquired by sensors in real time, such as the shape and location data of the surrounding obstacles and the structure of the road, but also the traffic signals and the driving status of the car itself. Our algorithm uses quintic spline polynomials to simulate a number of viable and smooth local paths to be selected. Afterwards, the decision-making layer selects an optimal local path for the car to follow. The car is supposed to avoid obstacles and approach the initial global path as closely as possible while still moving forward. Finally, the path following module will carry out both lateral and longitudinal speed control of the vehicle and ultimately offer precise commands for the throttle, the brake and the steering wheel controllers.

During the process of algorithm verification, through our simulation platform, we set a variety of obstacles in the simulated scenario. This has confirmed the effectiveness of the corresponding algorithms. After that the algorithm migration on our real car platform has achieved satisfying results.

B. Reinforcement Learning-based Car-following Model Validation on HiL Simulation Platform

By enforcing the application of reinforcement learning algorithm, which in-cooperates heuristic methods into the control strategy of ACC, optimal behaviors for speed adjustment can be learned and to handle vehicle control problems under complex dynamic traffic scenarios. In this experiment, the same simulated public road scenario is utilized, in which a following car should keep maintaining a minimum distance from the preceding car for a smooth and stable on-road driving performance.

In order to conduct calculations, the vehicle control is divided into two parts: lateral control (e.g., steering) and longitudinal control (e.g., throttle and brake). According to the following car's present lane line deviation, lateral control parameters are obtained. Simultaneously, the car's current state is represented by a quaternion, including its speed, the preceding car's speed, the real-time distance between two cars and the desired safe distance. Following this, our reinforcement learning algorithm generates appropriate acceleration control commands for subsequent modules by using vehicle state information. This information then updates along with the quaternion components for the next state with a predefined time interval of one second for a smooth control process. Therefore, through continuous learning and training, while interacting with the environment, the car under test is capable of going after the car that proceeds it while ensuring a proper car-to-car distance and a reasonable relative speed. The training scenario is shown in Fig. 7.

Fig. 7. Reinforcement learning-based car-following model on a simulated public road. Data collected from this process are sent to ECU interface for further computation and decision making, thus enhancing our simulation with hardware in the loop.

The experiment result shows that our car-following model is gradually optimized through constant modification of the mapping strategy from state to action in a learning manner and along with a confirmed low probability of vehicle collision or irregular motion. Moreover, the ECU module in our HiL simulation platform remains the same whether in the simulation environment or on the real car, which helps to acquire training results very similar to that of reality.

C. Autonomous Parking Algorithm Verification on HiL Simulation Platform

The autonomous parking system is one specific and important application of autonomous driving. This experiment is carried out with the aim of validating the feasibility of our simulation platform in the verification of a vehicle-centered autonomous parking algorithm. The target car is intended to find the most appropriate parking space rapidly and complete the parking behavior with the correct strategy according to parking space type. The stability and accuracy of whole process is also taken into consideration when validating algorithms. In order to offer a reasonable testing field, we have constructed a comprehensive simulated parking lot. As is visible in Fig. 8, within this parking lot three different types of parking spaces are provided. Other ordinary cars are randomly distributed in the parking lot. The exit and entrance is the same and the target car is initially positioned at the entrance.

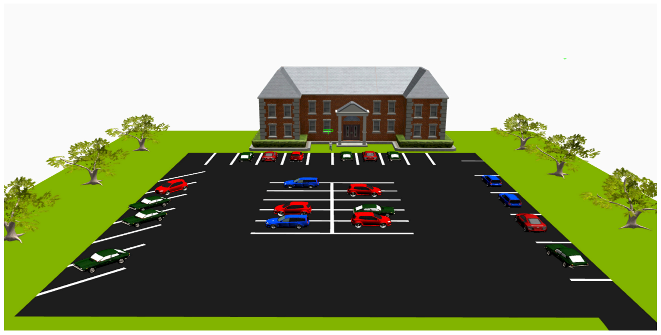

Fig. 8. A simulated parking lot. Multiple ordinary cars will appear or disappear randomly in any parking space.

Firstly, in our simulation we assume that the car has already got all of the physical parameters regarding of the parking lot's model, and will start moving while following a global route in the parking lot. There will be a random generation or disappearance of ordinary cars in different parking spaces over time. These dynamic changes are not automatically transmitted to our target car. Therefore, in the process of the car's movement along the known route, multi-sensor detection will allow it to react to the surroundings in time. Following this, our RRT-based (Rapid-exploring Random Tree) algorithm is used to determine whether there already exists suitable parking spaces in the sensing area, and calculate each one's corresponding accessibility rate according to pre-defined selective rules. The parking space with highest accessibility rate is then selected for further path planning and decision-making. In this simulated scenario three different types of parking spaces are available to verify the intelligence of the parking strategy implementation of our algorithm.

By measuring whether the car is able to park in the very center of the parking space and monitoring the distance from adjacent obstacles during parking, we confirmed that our algorithm for the autonomous parking system was effective on the simulation platform. In addition, this algorithm also helps to verify that our simulated car model is a stable one which is close to the real car platform, because rare irregular behaviors caused due to unreasonable initial design were observed.

V. CONCLUSION

This paper proposed a novel simulation platform based on HiL system. A flexible construction of kinematic car model, virtual sensors and various simulated scenarios was accomplished in the software simulation interface. The hardware control interface was developed with the aim of combining the software interface for efficient closed-loop simulation. What's more, the simulation of multi-agent interaction and V2I communication was also introduced in detail. In the end several self-driving experiments were executed to verify the effectiveness of our HiL simulation platform. Moreover, in our future work we are planning to further improve the capability of our platform for ECU development and optimization, with offline tests in more complex virtual situations.

REFERENCES

[1] G. Ros, S. Ramos, M. Granados, A. Bakhtiary, D. Vazquez, and A. M. Lopez, "Vision-based offline-online perception paradigm for autonomous driving," in *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*. IEEE, 2015, pp. 231–238.

[2] J. Florbäck, L. Tornberg, and N. Mohammadiha, "Offline object matching and evaluation process for verification of autonomous driving," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, 2016, pp. 107–112.

[3] S. Chen, J. Shang, S. Zhang, and N. Zheng, "Cognitive map-based model: Toward a developmental framework for self-driving cars," in *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*. IEEE, 2017, pp. 1–8.

[4] W. Qian, Z. Xia, J. Xiong, Y. Gan, Y. Guo, S. Weng, H. Deng, Y. Hu, and J. Zhang, "Manipulation task simulation using ros and gazebo," in *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2594–2598.

[5] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache, "Simulation environment for mobile robots testing using ros and gazebo," in *System Theory, Control and Computing (ICSTCC), 2016 20th International Conference on*. IEEE, 2016, pp. 96–101.

[6] O. Gietelink, J. Ploeg, B. De Schutter, and M. Verhaegen, "Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations," *Vehicle System Dynamics*, vol. 44, no. 7, pp. 569–590, 2006.

[7] P. Wältermann, "Hardware-in-the-loop: The technology for testing electronic controls in automotive engineering," in *6th Paderborn Workshop" Designing Mechatronic Systems" Paderborn*, 2009.

[8] Ş. Y. Gelbal, S. Tamilarasan, M. R. Cantaş, L. Güvenç, and B. Aksun-Güvenç, "A connected and autonomous vehicle hardware-in-the-loop simulator for developing automated driving algorithms," in *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 3397–3402.

[9] R. K. Bhadani, J. Sprinkle, and M. Bunting, "The cat vehicle testbed: A simulator with hardware in the loop for autonomous vehicle applications," *arXiv preprint arXiv:1804.04347*, 2018.

[10] M. Lee, H. Lee, K. Lee, S. Ha, J. Bae, J. Park, H. Park, H. Choi, and H. Chun, "Development of a hardware in the loop simulation system for electric power steering in vehicles," *International journal of Automotive technology*, vol. 12, no. 5, pp. 733–744, 2011.

[11] H. Zhang, Y. Zhang, and C. Yin, "Hardware-in-the-loop simulation of robust mode transition control for a series–parallel hybrid electric vehicle," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 3, pp. 1059–1069, 2016.

[12] S. Chen, S. Zhang, J. Shang, B. Chen, and N. Zheng, "Brain-inspired cognitive model with attention for self-driving cars," *IEEE Transactions on Cognitive & Developmental Systems*, vol. PP, no. 99, pp. 1–1, 2017.

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, and V. Koltun, "Carla: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.

[14] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu, "Real-time trajectory planning for autonomous urban driving: framework, algorithms, and verifications," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 2, pp. 740–753, 2016.

[15] R. Isermann, J. Schaffnit, and S. Sinsel, "Hardware-in-the-loop simulation for the design and testing of engine-control systems," *Control Engineering Practice*, vol. 7, no. 5, pp. 643–653, 1999.

[16] S. C. Oh, "Evaluation of motor characteristics for hybrid electric vehicles using the hardware-in-the-loop concept," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 3, pp. 817–824, 2005.

[17] E. Armagan and T. Kumbasar, "A fuzzy logic based autonomous vehicle control system design in the torcs environment," in *Electrical and Electronics Engineering (ELECO), 2017 10th International Conference on*. IEEE, 2017, pp. 737–741.

[18] E. Khalili, J. Ghaisari, and M. Danesh, "Control and analysis of the vehicle motion using sliding mode controller and carsim software," in *Control, Instrumentation, and Automation (ICCIA), 2017 5th International Conference on*. IEEE, 2017, pp. 1–5.

[19] S. Alvarez, Y. Page, U. Sander, F. Fahrenkrog, T. Helmer, O. Jung, T. Hermitte, M. Düering, S. Döering, and O. Op den Camp, "Prospective effectiveness assessment of adas and active safety systems via virtual simulation: A review of the current practices," in *25th International Technical Conference on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration*, 2017.

[20] S. Biswas, R. Tatchikou, and F. Dion, "Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety," *IEEE communications magazine*, vol. 44, no. 1, pp. 74–82, 2006.

[21] J. Gozálvez, M. Sepulcre, and R. Bauza, "Ieee 802.11 p vehicle to infrastructure communications in urban environments," *IEEE Communications Magazine*, vol. 50, no. 5, 2012.

[22] W. C. Mitchell, A. Staniforth, and I. Scott, "Analysis of ackermann steering geometry," SAE Technical Paper, Tech. Rep., 2006.

[23] F. Ramm, J. Topf, and S. Chilton, *OpenStreetMap: using and enhancing the free map of the world*. UIT Cambridge Cambridge, 2011.

[24] M. A. Brovelli, M. Minghini, M. Molinari, and P. Mooney, "Towards an automated comparison of openstreetmap with authoritative road datasets," *Transactions in GIS*, vol. 21, no. 2, pp. 191–206, 2017.