Generating Autonomous Driving Test Scenarios based on OpenSCENARIO

He Chen[1], Hongping Ren[1], Rui Li[1], Guang Yang[1], and Shanshan Ma[2,*]
[1]Institute of Software Chinese Academy of Sciences, Beijing, China
[2]Electronics Standardization Institute, Beijing, China
chenhe@iscas.ac.cn, hongping@iscas.ac.cn, lirui@iscas.ac.cn,
yangguang@iscas.ac.cn,mass1@cesi.cn
*corresponding author

2022 9th International Conference on Dependable Systems and Their Applications (DSA) | 978-1-6654-8877-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/DSA56465.2022.00093

***Abstract*—At present, with the rapid development of autonomous driving technology, simulation test has become an important means of autonomous driving technology verification and evaluation due to its high-test efficiency, strong repeatability, low cost, and process safety. However, the parameters in the test scenarios in the current simulation test are all fixed values, the test is only a discrete test, and a comprehensive test of the entire parameter space is lacking. Therefore, this paper proposes a generating method of autonomous driving test scenarios based on OpenSCENARIO and constructs a test scenario including parameter space. This method automatically builds the test scenario by constructing the configuration file of the test scenario and writing Python scripts. When executing the test scenario, the road elements, weather elements and traffic participant elements are used to screen the test scenarios that conform to the operation design domain of the autonomous driving, and the autonomous driving function is tested and verified through the Carla simulation platform. Finally, a test scenario is built for the lane keeping function. Experiments show that the test scenarios constructed automatically in this paper can not only meet the needs of autonomous driving simulation test, but also simplify the writing of test scenarios and greatly improve the efficiency of scenario construction.**

Keywords-autonomous driving; simulation test; OpenSCENARIO; test scenario; batch SCENARIO generation

I. INTRODUCTION

With the rapid development of current artificial intelligence technology, autonomous driving systems based on artificial intelligence technology have attracted extensive attention from the government, academia, and industry. Compared with traditional cars, self-driving cars can reduce the harm caused by human mis operation, improve road safety, efficiently plan driving paths, relieve road congestion, free up human operations, and improve the convenience of travel.

However, the autonomous driving system is a very complex system, including a series of intelligent algorithms such as environmental perception, perception fusion, path planning, and vehicle control. How to ensure the safety, reliability and stability of the autonomous driving system has become an urgent problem that needs to be solved. Autonomous driving simulation test is one of the powerful tools to solve the above problems.

Autonomous driving simulation test is the application of computer simulation technology in the automotive field. Its principle is to digitally restore the application scenarios of autonomous driving by means of mathematical modeling, establish a system model as close to the real world as possible, replace the real controller with algorithms, and combine sensor simulation and other technologies to complete the autonomous driving simulation test[1]. The purpose of testing and verifying autonomous driving systems and algorithms can be achieved by analyzing and researching through simulation tests.

In terms of testing methods, the industry currently divides autonomous driving simulation tests into five types: model-in-the-loop testing, software-in-the-loop test, hardware-in-the-loop test, driver-in-the-loop test, and vehicle-in-the-loop test. The specific classification of autonomous driving simulation test is shown in the following table.

The autonomous driving simulation test has the characteristics of high-test efficiency, strong repeatability, low cost, and process safety, which can help R&D personnel and automobile certification agencies to optimize and verify the autonomous driving algorithm. For autonomous driving simulation test, it is first necessary to build a high-coverage and high-complexity test scenario. A scenario is a dynamic description of how the world changes over time. In the field of autonomous driving simulation, a complete scenario includes driving situations (such as road layout, road facilities, and road materials) and driving scenarios (such as weather, lighting, vehicles, objects, pedestrians, and the status of traffic lights). Driving scenario is a description of static environment including logical road network and optional road material, while driving scenario is an overall description of dynamic entity behavior and optional dynamic entity model[2].

Scenario-based simulation test can not only test the scope and complexity of operating conditions, but also realize repeated testing of key dangerous scenarios to ensure the adequacy of test verification. In addition, in simulation test, automatic testing and accelerated testing are used to improve testing efficiency.

This paper mainly focuses on the simulation test of the autonomous driving model in the loop. Unless otherwise specified below, the simulation test refers to the model-in-the-loop simulation test. The autonomous driving simulation test process is shown in Figure 1.

2767-6684/22/$31.00 ©2022 IEEE
DOI 10.1109/DSA56465.2022.00093

Table 1 Classification of autonomous driving simulation tests

num	name	content	purpose
1	Model-in-the-loop	Connect the control algorithm model and the controlled algorithm model for simulation test.	Integration tests for model-level tasks.
2	software-in-the-loop	Derived from model-in-the-loop testing, the difference is that the controller's model is replaced with C code generated by the controller's model .	Verify that the generated code and the model are functionally consistent .
3	hardware-in-the-loop	Provide a dynamic system model that can simulate the real system environment as "controlled equipment simulation", and connect it with the simulation system platform through the input and output of the embedded system to form a closed loop.	The purpose of hardware-in-the-loop testing is to validate the controller.
4	driver-in-the-loop	On the basis of simulating the vehicle, engine and other systems, on the premise of ensuring the test accuracy, the simulation of traffic and environment is added, and the real driver is introduced into the closed-loop simulation test. system to verify.	The purpose of driver in-loop test is to verify the system.
5	Vehicle-in-the-loop	Integrate the test system into a real vehicle and simulate roads, traffic scenarios, and sensor signals through a simulation platform to form a complete test loop.	The purpose of the vehicle-in-the-loop test is to verify the function of the test system, the simulation test of each scenario, the matching and integration test with the relevant electronic control system of the vehicle.

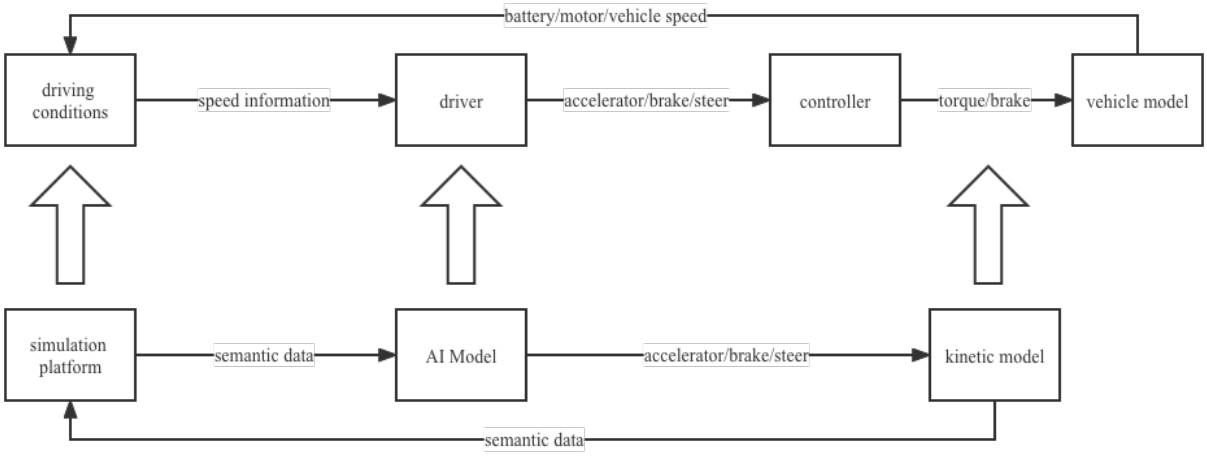

Figure 1 The process of autonomous driving simulation test

For non-autonomous vehicles in the real world, the driver makes a driving operation (e.g., accelerator, brake, and steering) based on the driving purpose (passing through a straight intersection) according to the driving conditions (e.g., own speed, and surrounding environment), and the controller outputs torque conversion The corresponding vehicle speed is fed back to the driver to form a closed loop for the driver.

For autonomous vehicles in the simulation world, the autonomous driving system is based on the function under test (e.g., passing through the intersection), and the system under test decides control plan (e.g., local path) based on the semantic data in the simulation platform (e.g., self-vehicle speed), the dynamic model is converted into the corresponding response speed, etc., and fed back to the simulation platform, the user can evaluate the performance of the autonomous driving system through the recorded data in the simulation platform, so as to evaluate the autonomous driving system.

II. Related Work

At present, in the field of autonomous driving simulation, A SAM has proposed the Open SCENARIO standard, which developers can easily use to describe test scenarios and clearly describe the dynamic behavior of traffic participants[3]. However, due to the relatively short time for this standard to be proposed, the generation technology of how to generate scenario files in OpenSCENARIO format is still insufficient. To this end, this paper proposes a method for generating autonomous driving test scenarios based on OpenSCENARIO .

2.1 Introduction to OpenSCENARIO

OpenSCENARIO is a standard developed by the Association for Standardization of Automation and Measurement Systems (ASAM), which is dedicated to the description of dynamic scenarios in the field of scenario simulation for the virtual development, testing and validation of advanced autonomous driving assistance and autonomous driving functions.

Only the dynamic content of the scenario is defined in Open SCENARIO, and the static content is not included in Open SCENARIO, but refers to other standards, such as OpenDRIVER and OpenCRG.

Up to now, the ASAM organization has released the OpenSCENARIO1.2 version.The version of Open SCENARIO1.2 mainly consists of six parts, as shown in the figure. Among them, the file header describes the author, date, version number, relevant description, and other content of creating the file. The parameter declaration describes the value, type, etc. of the relevant parameters, and the parameters can be referenced in the scenario. Directory definitions describe directory addresses for vehicles, controllers, pedestrians, obstacles, environments, actions, trajectories, and routes, where specific information can be referenced in the scenario. Road network references to specific road networks and references to road material 3D models. Entities define the physical information of traffic participants (including vehicles, pedestrians, and static objects). The storyboard depicts the entire complete dynamic scenario, describing "who" and "when" doing "what action".

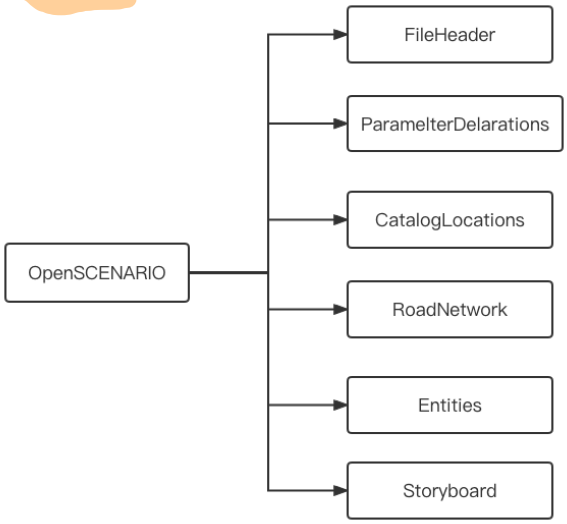

Figure 2 Basic framework of Open SCENARIO

2.2 OpenSCENARIO Standard of ASAM Application

The OpenSCENARIO standard is a constantly developing standard, and more and more scientific research projects, engineering projects and tool software are applying the OpenSCENARIO standard to complete the scenario construction in the test and verification process.

2.2.1 Application of ASAM OpenSCENARIO in scientific research and engineering projects

At present, the OpenSCENARIO standard is being widely used and developed in scientific research and engineering projects.

(1) CARLA

CARLA is an open-source autonomous driving simulator. It was built from the ground up to serve as a modular and flexible API to solve a range of tasks involving autonomous driving problems. One of CARLA's primary goals is to help democratize autonomous driving research and development as a tool that users can easily access and customize. To do this, the simulator must meet the requirements of different use cases in general driving problems (e.g. learning driving policies, training perception algorithms, etc.). CARLA runs simulations based on Unreal Engine and uses the OpenDRIVE standard (currently 1.4) to define roads and urban environments. By having an API handled by Python and C++ that grants control over the simulation, the API will grow as the project continues to evolve[4].

To make the process of developing, training, and validating driving systems a smooth process, CARLA has grown into an ecosystem of projects built by the community around the main platform. Among them, the ScenarioRunner tool in the CARLA ecosystem uses the OpenSCENARIO standard to define scenarios and uses the CARLA simulator to define and execute dynamic scenarios.

(2) Simulation Scenarios

The project was initiated to make virtual test system creation scenarios more efficient and versatile by using a standard test scenario format. Three key scenarios were selected for the project: cut-in, left-turn crossing path-opposite direction (LTAP-OD), and highway merging. The criteria for selecting these were their relevance to project stakeholders and their extensive and challenging requirements for the format used. Key scenarios are broken down and described in the XML-based open data format OpenSCENARIO.

Finally, the project demonstrates key scenarios in all specified test environments, validating the use of OpenSCENARIO to coordinate test environments in terms of environment simulation and traffic simulation. At least for the various scenarios and test environments the project covers[5].

2.2.2 Software that currently supports the ASAM OpenSCENARIO standard

At present, many simulation platforms have begun to support ASAM The Open SCENARIO standard, through the scenario files defined by the ASAM OpenSCENARIO standard, can run in different simulation platforms, which greatly improves the efficiency of simulation test.

(1) 51Sim-One

51Sim-One is an autonomous driving system independently developed by 51WORL D that integrates multi-sensor simulation, vehicle dynamics, road and scenario simulation, traffic flow and intelligent body simulation, perception and decision simulation, evaluation indicators, and autonomous driving behavior training. The simulation and test platform can build a complex traffic environment with various traffic participants in the loop and realize data-driven cloud simulation[6].

(2) AD Chauffeur

AD Chauffeur is an intelligent vehicle virtual simulation cloud platform independently developed by China Automobile Data Co., Ltd. The platform adds AD Scenario

generator to realize automatic scenario generation and splicing; in-depth optimization of dynamic simulation and sensor simulation functions; the built-in scenario library is expanded to 4,000 One, fully meet the needs of users out of the box; continue to add hardware supported by XIL-in-the-loop testing, including bus tools, real-time systems, sensor simulation equipment, driving simulators, etc. Among them, the AD Scenario generator solves the problems of scenario construction, multi-source data format conversion OpenSCENARIO, logical scenario splicing and reorganization, and provides important support for scenario generation[7].

(3) VTD

VTD (Virtual Test Drive) is a complete modular simulation tool developed by German VIRES company for driver assistance systems, active safety and autonomous driving. It runs on the Linux platform, and its functions cover road environment modeling, traffic scenario modeling, weather and environment simulation, simple and physically realistic sensor simulation, and high-precision real-time image rendering.

The scenario construction of VTD mainly includes two steps: road environment construction and dynamic scenario configuration. The construction of static scenarios such as road environment is completed by Road Network Editor (ROD); the configuration of dynamic scenarios is realized by Scenario Editor[8].

III. GENERATION OF AUTONOMOUS DRIVING TEST SCENARIOS

Although, scenario files in Open SCENARIO format describe the actions and behaviors of vehicles, pedestrians, and other traffic participants, as well as changes in weather conditions. However, in the infinitely rich real world, in the test scenario with fixed parameters, the parameter values are fixed, and the simulation test based on this test scenario can only be a discrete test, lacking a comprehensive test of the entire parameter space.

In addition, test scenarios require additional information to describe the complete simulation setup and test cases. For example, the measured object, advanced dynamic model settings and other information.

Therefore, in view of the problems caused by the direct use of OpenSCENARIO scenario files for testing, this paper proposes a method for generating autonomous driving test scenarios based on OpenSCENARIO when constructing the autonomous driving test scenario, automatically builds a test scenario including parameter space files, tested object files, and specific scenarios, and the generated specific scenario storage format uses the OpenSCENARIO1.2 standard format.

3.1 Scenario Construction Route tot Autonomous driving

This article starts from the autonomous driving function, builds the test scenario configuration file, and automatically constructs the test scenario by writing Python footsteps. The test scenario consists of the measured object file, the parameter space file, and the specific scenario file of OpenSCENARIO. When executing the test scenario, the road elements, weather elements and traffic participant elements are used to screen the test scenarios that conform to the autonomous driving design and operation domain, and then the autonomous driving function is tested and verified through the Carla simulation platform. The autonomous driving test scenario generates a route, as shown in Figure 4.

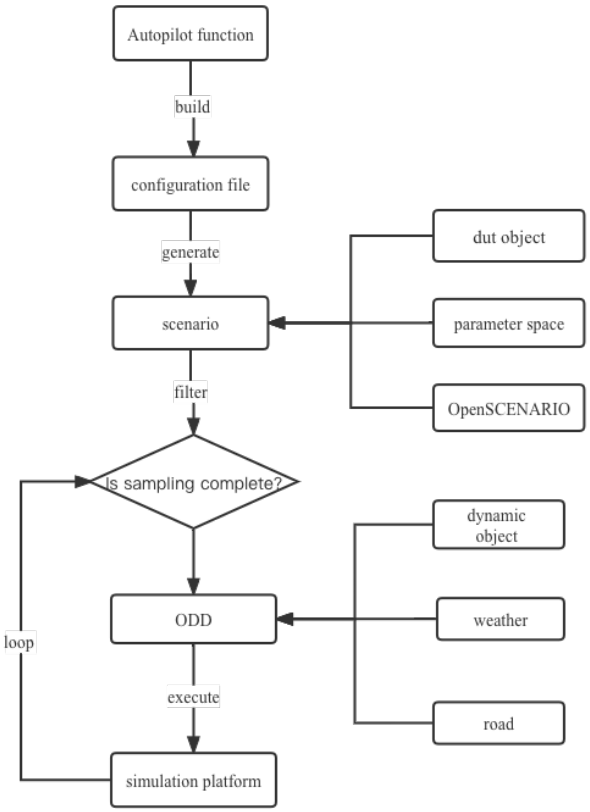

Figure 3 Autonomous driving test scenario generation

3.2 Analysis of scenario configuration files

The scenario configuration file consists of three parts, namely Dut, Scenarios and oracles. Among them, Dut describes the mission information and optional parameters of the system under test, such as destination location, target speed, etc. Scenarios describes the entire dynamic scenario, which is composed of scenario name, scenario parameters, static map, entity object, scenario execution and other modules. Oracles are the assertion part of the test scenario, if the assertion in that part is not true, the scenario run will be aborted.

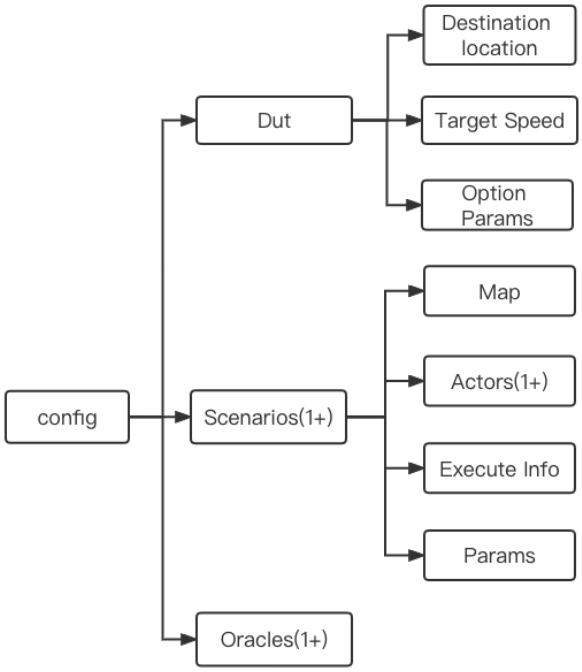

Figure 4 Scenario configuration file

Execute is a tree-like structure consisting of leaf nodes and compound nodes. Leaf nodes are used to describe the dynamic behavior of traffic participants; compound nodes are used to manage child nodes and apply execution logic to child nodes. The composite node type has two modes: Sequence and Parallel. Sequence mode requires that child nodes execute serially; Parallel mode requires child nodes to execute in parallel.

OpenSCENARIO according to the configuration file is shown in the following table.

The scene behavior tree of the test scene can be constructed through the behavior execution information. Here, the depth of the scene behavior tree structure can be positively correlated with the complexity of the behavior execution information according to the atomic actions and their sequential logical relationship. The more complex the logical relationship, the deeper the depth of the scene behavior tree.

Extract all the atomic actions of the scene elements in the test scene from the behavior execution information, as well as the temporal logic relationship between the atomic actions; determine the action execution conditions of each atomic action according to the temporal logic relationship between the atomic actions; in each atomic action Before, add the action execution conditions of the corresponding atomic action to obtain the corresponding updated atomic action; combine all the updated atomic actions in parallel to obtain the target behavior execution information.

The transformation of the behavior execution information into target behavior execution information mainly includes the following steps.

Step1: Breadth-first traverse the behavior tree of behavior execution information to obtain the deepest node.

Step2: Traverse the behavior tree of behavior execution information in reverse order from the deepest layer, and obtain the end condition of each node until the layer has no nodes.

The end condition rules for each node are as follows:

1) For an unconditional action node, its end condition is the action node itself;

2) For a conditional action node, its end condition is the judgment condition of the action node;

3) For the combined node in serial mode, its end condition is the end condition of the last child node;

4) For a combined node in parallel mode, its end condition is the union of the end conditions of all child nodes.

Step3: Traverse each layer of the behavior tree in order from bottom to top, repeat step 2 in the traversal of each layer, and obtain the end conditions of all nodes in each layer until the behavior tree of the behavior execution information has no parent node.

Step4: traverse the behavior tree of behavior execution information in positive order from the top level, and obtain the start condition of each node until the layer has no nodes.

The start condition rules for each node are as follows:

1) For the top-level parent node, its start condition is NULL;

2) For a node whose parent node is a combined node in parallel mode, its start condition is the start condition of its parent node;

3) For a node whose parent node is a combined node in serial mode, and the position of the node is not the first child node of its parent node, its start condition is the start condition of its parent node;

4) For a node whose parent node is a combined node in serial mode, and the position of the node is not the first child node of its parent node, its start condition is the end condition of the previous node of the same level node;

Step 5: Traverse each layer of the behavior tree in order from top to bottom, repeat step 4 in the traversal of each layer, and obtain the start conditions of all nodes in each layer until the behavior tree of the behavior execution information has no child nodes;

Step 6: Traverse all the action nodes of the behavior tree, write the start conditions of the action nodes into each action node, and rearrange all the action nodes to the deepest level of the behavior tree, and the parent nodes of all the action nodes are the combination of the parallel mode node.

3.3 Parameter sampling in compliance based on the autonomous driving ODD

The test scenario with parameter space avoids that only a set of fixed parameter values can be tested in the simulation test and ensures the comprehensiveness of the simulation test. However, in the test scenario with parameter space, the parameters can be arbitrarily combined, and the test workload is relatively large. Therefore, we can pre-screen some test scenarios according to the autonomous driving design running domain before large-scale simulation test.

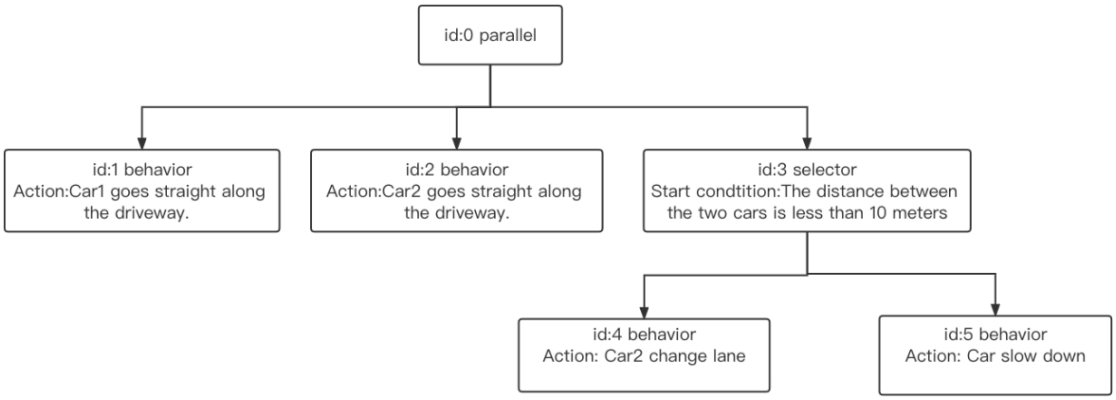

Figure 5 Behavior information in configuration file

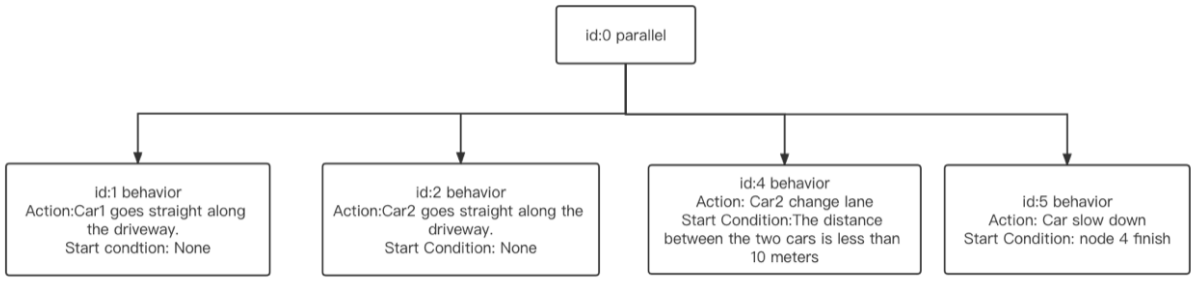

Figure 6 Behavior information in OpenSCENARIO file

Usually, the autonomous driving operational design domain is pre-defined by developers when developing autonomous driving functions, and includes road types, weather conditions, traffic conditions, running time, and other scenario parameters related to safe driving for the autonomous driving system to operate safely. These scenario parameters are an important basis for assisting and determining the safe operating boundaries of autonomous vehicles. The autonomous driving system can operate safely in the autonomous driving operational design domain, and beyond the operational design domain, the safety of autonomous driving cannot be guaranteed. For example, there are self-driving systems on urban expressways that can only be up and running on a clear day. When the system starts and runs, it is necessary to judge whether the current environment is a sunny day, so as to judge whether the autonomous driving system starts and runs. Autopilot can start and run only if it conforms to the autopilot Operational Design Domain[9].

The autonomous driving operational design domain is the first step in developing an autonomous driving system. By defining the autonomous driving operational design domain, the safety of the autonomous driving system is guaranteed. Therefore, we can first filter the test scenarios that conform to the autonomous driving operational design domain through road elements, weather elements, and traffic participant elements.

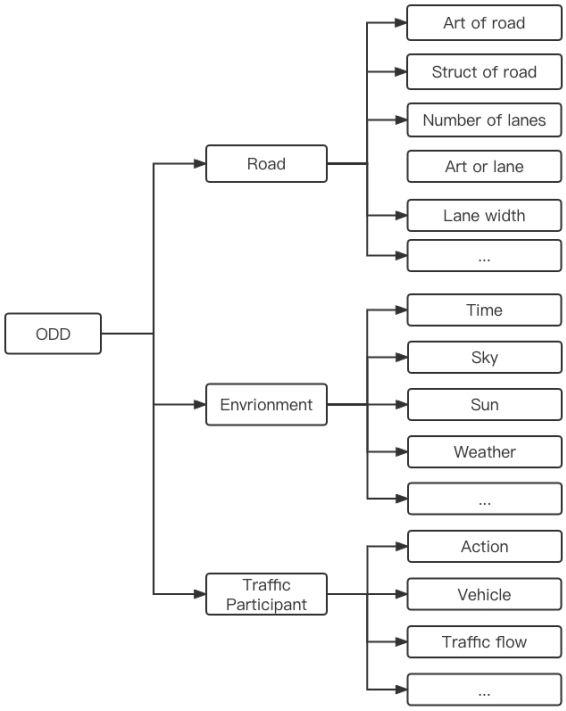

Figure 7 Autonomous Driving Operational Design Domain Structure

◆ **Road element**

When designing a scenario, road elements include road type, road structure, number of lanes, lane width, lane surface, lane curvature, special areas, traffic signs, etc.

◆ **Weather element**

When designing a scenario, weather elements include time, weather, rain and snow intensity, sunlight, street lights, etc.

◆ **Traffic Participant Elements**

When designing a scenario, the factors of traffic participants include vehicle type, vehicle behavior, pedestrian behavior, animal behavior, traffic flow, obstacle type, etc.

3.4 Execute test scenarios

Executing the test scenario is to realize the scheduling and control of the behavior of the scenario participants, so that the scenario participants can run according to the sequential logic described by the test scenario. To execute the test scenario, first verify whether the test scenario file conforms to the schema definition of OpenSCENARIO; then, parse the behavior actions and judgment conditions of the scenario file, and use the Py-trees toolkit to build the moving atom action behavior tree and atomic condition behavior tree[10]; then the scenario behavior tree is generated based on the life cycle of nodes in OpenSCENARIO.

The execution of the test scenario is the process in which the behavior tree controls the participants in the scenario. The behavior tree runs through tick. When a behavior tree is tick, the child nodes are traversed from the root node of the tree, and those nodes are ticked according to the structure of the behavior tree, the next tick is determined according to the result of the node state change, those child nodes are traversed until the state of the root node changes. When the behavior tree is tick, the node in the tree whose state is running transmits parameters such as target speed, target direction, running distance and running time to the controller of the character executing the behavior. The speed controller calculates the values of accelerator, brake and reverse gear, and the direction controller plans the driving route of the vehicle according to the type of behavior performed by the vehicle, combined with the high-precision map, and calculates the control amount of the direction of the vehicle. The behavior tree applies the control to the vehicle in the simulator every tick, the attributes of the vehicle change when the vehicle moves, the behavior and event nodes obtain the participant's data from the simulation environment and determine whether their own state needs to be updated, and if the conditions for ending are met, the behavior and event end, and the new behavior begins to execute. After the test criterion has the result, the behavior tree tick ends, the object in the simulator is destroyed, and the test ends. During the operation of the behavior tree, the scenario in the simulation environment is obtained by setting sensors such as cameras on the vehicle.

OpenSCENARIO file generated in the previous section through the Carla simulation platform is shown in the figure below.

```
<OpenSCENARIO xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="OpenScenario.xsd">
    <FileHeader description="cut-int" author="hee0624" revMajor="1" revMinor="2" date="2022-05-10T18:06:46.389234"/>
    <ParameterDeclarations>
        <ParameterDeclaration name="target_speed" parameterType="double" value="5"/>
        <ParameterDeclaration name="target_length" parameterType="double" value="4"/>
        <ParameterDeclaration name="target_time" parameterType="double" value="4"/>
        <ParameterDeclaration name="change_length" parameterType="double" value="2"/>
    </ParameterDeclarations>
    <CatalogLocations/>
    <RoadNetwork>
        <LogicFile filepath="Town05"/>
    </RoadNetwork>
    <Entities>
        <ScenarioObject name="hero">
            <Vehicle name="vehicle.tesla.model3" vehicleCategory="car">
                <ParameterDeclarations/>
                <BoundingBox>
                    <Center x="1.5" y="0.0" z="0.9"/>
                    <Dimensions width="2.1" length="4.5" height="1.8"/>
                </BoundingBox>
                <Performance maxSpeed="69.444" maxDeceleration="20.0" maxAcceleration="200.0"/>
                <Axles>
                    <FrontAxle maxSteering="0.5" wheelDiameter="0.6" trackWidth="1.8" positionX="3.1" positionZ="0.3"/>
                    <RearAxle maxSteering="0.0" wheelDiameter="0.6" trackWidth="1.8" positionX="0.0" positionZ="0.3"/>
                </Axles>
                <Properties>
                    <Property name="type" value="ego_vehicle"/>
                </Properties>
```

Figure 8 Vehicle cut into scenario file

Figure 9 The vehicle cuts into the scenario

IV. CONSTRUCTION OF TEST SCENARIOS FOR AUTONOMOUS DRIVING FUNCTIONS THAT RESPOND TO CUT-IN TARGET OBJECT

Self-driving vehicles need to face complex road conditions in the real world, and self-driving vehicles need to probe the surrounding environment all the time and respond to emergencies in a timely manner. Especially when the autonomous vehicle is driving on the road, it can detect and respond reasonably in time to the cutting target, avoid collision, and ensure the safety of the vehicle. This section will focus on the response function of the cut-in target, construct the test scenario, and conduct the simulation test of the autonomous driving system in the Carla simulation platform.

Step1. Build the scenario configuration file. When the ego vehicle is driving in the innermost lane of the two-way 2-lane, and the rear target vehicle is traveling from the outermost lane to a certain distance from the front of the ego vehicle, it changes lanes to the lane to which the ego vehicle belongs, and then decelerates until it stops. The specific parameter space is as follows:

Step2. Parse the scenario configuration file to generate the measured object file, parameter space file and scenario file.

Step3. Sample the parameter space and select suitable test scenarios based on the design running domain. For example, regarding this scenario, select the weather as sunny and light rain. If the test scenario is daytime, the lane change distance d needs to satisfy formula $d > (v2 - v1) * 3$;if the test scenario is night, the lane change distance d needs to satisfy formula $d > (v2 - v1) * 5$

Step4. Execute the test scenario, first verify whether the test scenario file conforms to the schema definition of OpenSCENARIO. Then, parse the behavior action and judgment condition of the scenario file, and use the py -trees toolkit to build the moving atom action behavior tree and atomic condition behavior tree; then which generates a scenario behavior tree based on the life cycle of nodes in OpenSCENARIO. Then run the entire scenario behavior tree.

Experiments show that the test scenarios constructed automatically in this paper can not only meet the needs of autonomous driving simulation test, but also simplify the writing of test scenarios and greatly improve the efficiency of scenario construction.

Table 2 The target vehicle cuts into the parameter space

type	Cut-in /Slow target vehicle cut-in		
description	slow vehicle cut in		
Hierarchy	element	parameter	range
road type			
road structure		road (road Type) road width (road Width) …	
environmental conditions	weather	Cloud Density _ _ Rain Density (rain Density) Snow density (snow density) fog density _ _ height angle (heightAngle) Directional Light (directionlLight)	W1: [0,1] W2: [0,1] W3: [0,1] W4: [0,1] W5: [0,90] W6: [0,1]
initialization	ego car1	initial position p 0 initial speed v0 initial position p 1 initial speed v1 Ego speed (v0)	V0: [30, 80] V1: [30, 80]
behavior control		When the target vehicle is d from the front of the ego vehicle, the lane change is completed in time t1, and the mountain brake is completed in time t2.	d: [30,80] t1: [3,5] t2: [10,20]
Assert	ego scenario	Ego is Collided () Scenario is Timeout ()	

V. SUMMARY AND OUTLOOK

In order to build autonomous driving test scenarios, this paper surveys ASAM The OpenSCENARIO standard, introduces the components of the scenario file in the standard and proposes a method for generating autonomous driving test scenarios based on OpenSCENARIO. Finally, to verify the validity and correctness of the test scenario generation method, the decision planning algorithm in autonomous driving was tested on the Carla simulation platform. Experiments show that, based on the autonomous driving test scenario generation method, the test scenario constructed by Python script can be executed normally in the Carla simulation platform, and the decision-making algorithm in the evaluation of autonomous driving can be reasonably verified.

With the rapid development of autonomous driving technology, the verification and evaluation of autonomous driving has also received widespread attention. At the same time, how to verify and evaluate autonomous driving is becoming more and more difficult. Therefore, building simulation test scenarios is a very valuable direction. In the future, there is still a huge research space for the construction of autonomous driving simulation test scenarios. We can continue to study in the direction of rapid scenario generation and test scenario language compilation to jointly promote the development of autonomous driving technology.

REFERENCES

[1] Mo Chunmei,Zhan Ziqi,Liu Bing,Zhou Yongfeng. Automated Simulation Test Research of Autonomous Driving Based on Parameterized Scenarios[C]//.2019 Proceedings of the Annual Meeting of China Society of Automotive Engineers（1）.,2019:9-17.

[2] Liu Jingyu,Ma Hui,Zhang Xuewen ,Yang wei. Research and development of autonomous driving virtual simulation test technology [J]. Scinecepaper Online,2021,16(06):571-577+584.

[3] ASAM OpenSCENARIO: User Guide [EB/OL].[2022-05-13]. https://www.asam.net/index.php?eID=dumpFile&t=f&f=4908&token=ae9d9b44ab9257e817072a653b5d5e98ee0babf8

[4] CARLA Documentation: Introduction [EB/OL].[2021-11-17].https://carla.readthedocs.io/en/latest/

[5] Simulation Scenarios [EB/OL].[2019-03] https://sites.google.com/view/simulationscenarios

[6] Automotive Autonomous Driving Simulation Test Blue Book[M]

[7] AD Chauffeur Autonomous Simulation Cloud Platform Helps Scenario-Based Development and Verification [EB/OL].[2020-11]. http://www.c-asam.net/ueditor/php/upload/file/20201202/1606892474721212.pdf

[8] VTD[EB/OL][2022-04-20] https://www.mscsoftware.com/product/virtual-test-drive

[9] Sun Hang,Zhang Hang, Zhang Miao, Gao Yongqiang, Xia Yuan, Chen Yao. Research on the Construction Method of Operational Design Condition of Intelligent and Connected Vehicle [J].China Auto,2020(12):41-49.

[10] LIU Jingyu1, MA Hu,ZHANG Xuewen,YANG Wei1. Research and development of autonomous driving virtual simulation technology [J]. Modern Electric Power,2020,37(04):399-407.DOI:10.19725/j.cnki.1007-2322.2020.009