ROADGEN: Generating Road Scenarios for Autonomous Vehicle Testing

Fan Yang*, You Lu*, Bihuan Chen*†, Peng Qin*, and Xin Peng*

Abstract— With the rapid development of autonomous vehicles, there is an increasing demand for scenario-based testing to simulate diverse driving scenarios. However, as the base of any driving scenarios, road scenarios (e.g., road topology and geometry) have received little attention by the literature. Despite several advances, they either generate basic road components without a complete road network, or generate a complete road network but with simple road components. The resulting road scenarios lack diversity in both topology and geometry. To address this problem, we propose ROADGEN to systematically generate diverse road scenarios. The key idea is to connect eight types of parameterized road components to form road scenarios with high diversity in topology and geometry. Our evaluation has demonstrated the effectiveness and usefulness of ROADGEN in generating diverse road scenarios for simulation.

I. INTRODUCTION

Autonomous vehicles have been widely developed in the last decades due to their impact on automotive transportation and their benefit to society (e.g., reducing vehicle collisions, and providing personal mobility to disabled people) [15]. It is important to ensure the safety and reliability of autonomous vehicles for world-wide adoption. Therefore, on-road testing is widely used by leading companies. However, autonomous vehicles would have to be driven more than 11 billion miles to demonstrate with 95% confidence that they are 20% safer than human drivers [10]. It is expensive for on-road testing to achieve this goal, and it is also impossible for on-road testing to test corner cases or dangerous situations.

To this end, scenario-based testing [17], [30] is also widely used by leading companies to simulate diverse driving scenarios. As of February 2021, Waymo's autonomous vehicles have been tested with over 15 billion miles of simulated driving [27]. Various approaches have been developed to generate driving scenarios [28], [6], [21]. However, they mainly focus on the diversity in vehicle/pedestrian behaviors and weather conditions, but overlook the diversity in roads [30], [21]. Several recent advances have been proposed on generating road scenarios [31], [18], [23], [16]. However, they either generate basic road components (e.g., highway interchanges) without a complete road network [31], [18], or build a complete road network but with simple road components (i.e., straight roads and junctions) [23], [16]. Therefore, their generated road scenarios lack diversity in both topology and geometry.

To address the problem, we propose ROADGEN to systematically generate diverse road scenarios. First, we define and

implement eight types of basic road components. Each road component is parameterized to reflect its geometry diversity. Second, we connect road components to form road scenarios. This process is guided by favoring the selection of least used road components so as to ensure the geometry diversity of road scenarios. Third, we remove duplicated road scenarios that have the same topology to ensure the topology diversity of road scenarios. Finally, we convert the generated road scenarios into high-precision (HD) map files and 3D scene files, which can be used for joint simulation by simulators.

To evaluate the effectiveness of ROADGEN, we use ROADGEN to generate road scenarios that have 4, 5, 6, 7 and 8 road components. Our results have demonstrated that ROADGEN can generate more diverse road scenarios than a baseline approach that randomly selects and connects road components. Furthermore, to evaluate the usefulness of ROADGEN, we sample road scenarios and convert them into HD map files and 3D scene files for joint simulation on SORA-SVL [9] with Apollo 8.0 [2]. Our results have indicated that over 92% of the road scenarios can be useful for joint simulation.

In summary, this work makes the following contributions.

- We define and implement eight types of parameterized basic road components.
- We propose a guided approach to connect road components to generate diverse road scenarios.
- We conduct experiments to demonstrate the effectiveness and usefulness of ROADGEN, and build a dataset of road scenarios for simulation testing.

II. RELATED WORK AND PROBLEM STATEMENT

A. Related Work

Scenarios [25], [13] are important for developing and testing autonomous vehicles. To represent scenarios, Bagschik et al. [1] develop a 5-layer model, including road-level (layer 1), traffic infrastructure (layer 2), manipulation of layer 1 and 2 (layer 3), objects (layer 4), and environment (layer 5), while Scholtes et al. [20] extend it by adding digital information as layer 6. To generate scenarios, various approaches have been developed [22], [5], [28], [6], [21], e.g., extracting scenarios from driving data [11], [3] or crash data [8], [4], [7], [29], and searching scenarios by evolutionary algorithms [26], [12], [24] or combinatorial interaction testing [19].

However, recent surveys [30], [21] show that most scenario generation approaches have focused on scenarios at layer 4 and 5 by manipulating vehicles, pedestrians and weather conditions, while little attention has been paid on road topology and geometry, and traffic signs (i.e., layer 1, 2 and 3), which serve as the base of any scenario. Zhou et al. [31] propose a

Fig. 1.   Approach Overview of ROADGEN

(a) Straight

(b) Curve

(c) Lane Switch

(d) Fork

(e) T-Intersection

(f) Intersection

(g) U-Shaped Road

(h) Roundabout

Fig. 2.   Eight Types of Typical Road Components
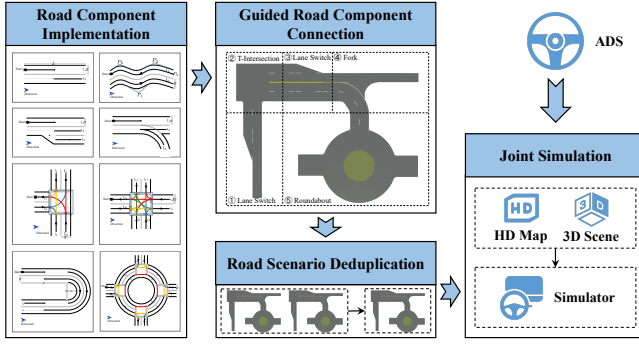
model-driven method to generate highway interchanges that are one basic component in road networks. Rietsch et al. [18] also attempt to generate basic components, i.e., roundabout, intersections, highway entry, drive and exit. Differently, our work aims to compose road networks based on basic components. Tang et al. [23] first extract junction features from HD maps, and then build road networks by connecting the junctions in a grid layout. Paranjape et al. [16] generate road networks with different road sizes and intersections. However, these approaches only consider straight roads and junctions, and thus the generated road networks lack diversity.

B. Problem Statement

This work is focused on the road-level (layer 1) scenario of the 5-layer model [1], which describes the topology and geometry of road scenarios. Specifically, the topology of a road scenario can be characterized by how different road components (e.g., straight road, curve road, fork road, and intersection) are connected together. The geometry of a road scenario can be characterized by factors like the number of lanes and the type of lane markings. The diversity of road scenarios in both topology and geometry is important for developing and testing autonomous vehicles. Therefore, our problem can be stated as *how to systematically generate road scenarios such that they have diverse topology and geometry.*

III. METHODOLOGY

We propose ROADGEN to systematically generate diverse road scenarios. An overview of ROADGEN is presented in Fig. 1. Each step of ROADGEN is explained below.

A. Road Component Implementation

Based on our understanding of real-life roads, we decompose roads into distinct segments, and define eight types of typical road components, as illustrated in Fig. 2. To reflect geometry diversity, each road component can be parameterized by road length (L), lane width (W), the number of lanes ($LaneNum$), the type of lane markings ($LaneMarks$), the coordinate of the starting point ($Start$), and the direction to position the component from the starting point ($Direction$).

- **Straight.** As shown in Fig. 2(a), a straight road follows a linear trajectory without any curves, bends, or turns.
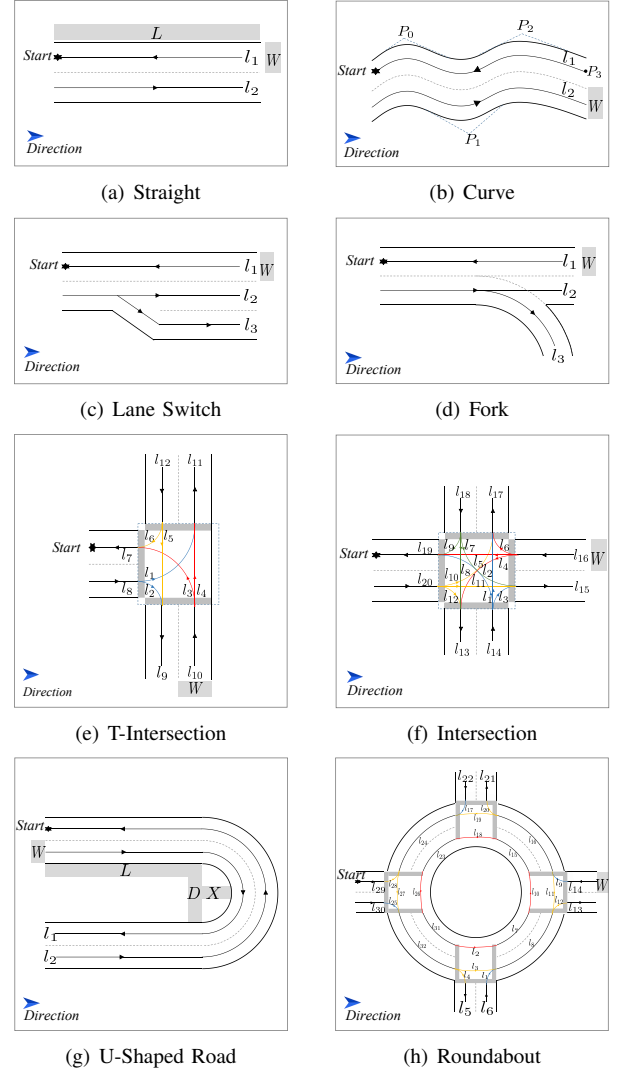
- **Curve.** As shown in Fig. 2(b), a curve road changes its direction as it progresses. We apply Bézier curves [14] to construct the lane curve. A Bézier curve $B(t)$ can be constructed by four control points $P_0 - P_3$, i.e., $B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3$, $t \in [0, 1]$.

- **Lane Switch.** As shown in Fig. 2(c), a lane switch road undergoes a transition about the number of lanes (e.g., changing from 2 lanes to 3 lanes).

- **Fork.** As shown in Fig. 2(d), a fork road is a road segment where a single road splits into two separate roads, or two separate roads merge into a single road.

- **T-Intersection.** As shown in Fig. 2(e), a T-intersection is a junction where one road meets another road at a right angle. It creates a three-way intersection where vehicles on the main road can continue straight, while vehicles on the intersecting road can turn left or right.

- **Intersection.** As shown in Fig. 2(f), intersection is a junction where two roads meet or cross each other.

- **U-Shaped Road.** As shown in Fig. 2(g), a U-shaped road is a road segment where a 180-degree turn is re-

```
1  % Initialize lanes in a component using LaneNum.
2  Lanes(LaneNums) = roadrunner.hdmap.Lane();
3  % Draw coordinates of lanes and boundaries.
4  Lanes.Geometry = deal([coordinates of lanes]);
5  LaneBoundaries.Geometry = deal([coordinates of boundaries]);
6  % Set lane markings according to LaneMarks.
7  LaneBoundaries.ParametricAttributes = deal(LaneMarks)
8  % Combine lanes with predecessors and successors
9  % where i,j,k represent the ID of lanes.
10 Lane(i).Predecessors = roadrunner.hdmap.Reference(Lane(j));
11 Lane(i).Successors = roadrunner.hdmap.Reference(Lane(k));
```

Listing 1.   Part of a Sample Template for Road Components

quired to go in the opposite direction. It can be considered as being composed of two straight road segments and a circular arc segment.

- **Roundabout.** As shown in Fig. 2(h), a roundabout is a junction where traffic from four roads flows around a central island in a counterclockwise direction (in countries with right-hand traffic).

After defining the eight types of road components, we use Python to implement programming templates for these road components. These templates can be instantiated to generate MATLAB scripts which describe the road components. As illustrated in Listing 1, all templates are primarily composed of the following steps, i.e., initializing the lanes contained in a component according to $LaneNum$, drawing coordinates of lanes and boundaries based on L, W, $Start$ and $Direction$, setting lane markings according to $LaneMarks$, and combining lanes to form the component by assigning references to lane predecessor and successor.

We provide different templates for each road component based on the number of lanes ($LaneNum$) and the type of lane markings ($LaneMarks$). These templates are distinct from each other, encompassing 1 to 6 lanes and any of the seven types of lane markings based on real-world road rules (i.e., white dashed lane lines, white solid lane lines, white double solid lane lines, yellow dashed lane lines, yellow solid lane lines, yellow double solid lane lines, and yellow dashed-solid lane lines). Specifically, we provide a total of 242 unique templates for these eight types of road components.

When instantiating these templates, we only need to specify the road length (L), lane width (W), coordinate of starting point ($Start$), and direction to position the road component ($Direction$). For curve road, we also need to specify $P_0 - P_3$. For U-shaped road, we also need to specify D to denote the distance between the two straight road segments, and X to control the arc segment. We can calculate the coordinates of the road and boundaries to fill in the template based on these input parameters, thereby instantiating the component.

B. Guided Road Component Connection

We design a guided algorithm to connect instantiated road components to generate diverse road scenarios, as presented in Algorithm 1. It has four inputs: (1) $CompList$, the 242 programming templates for the road components as imple-

---

Algorithm 1: Guided Road Scenario Generation

Input: $CompList$, $TotalCount$, $Constraints$, $Candidates$
Output: $RoadScenSet$

1 $CompCount = \emptyset$;
2 **while** $budget\ is\ not\ reached$ **do**
3 $\quad Count = 0;\quad CoveredArea = \emptyset$;
4 $\quad EndpointQueue = \emptyset;\quad RoadScen \Longleftarrow \emptyset$;
5 $\quad C = selectFirstComp(CompList, CompCount)$;
6 $\quad c = instFirstComp(Constraints, C)$;
7 $\quad RoadScen \Longleftarrow c.generateRoad()$;
8 $\quad Count ++$;
9 $\quad CoveredArea += c.getCoveredArea()$;
10 $\quad CompCount[C] += 1$;
11 $\quad EndpointQueue.add(c.getEndpoints())$;
12 $\quad$**while** $Count < TotalCount\ \&\&\ EndpointQueue \neq \emptyset$ **do**
13 $\quad\quad p = EndpointQueue.pop()$;
14 $\quad\quad$**if** $random()\ ||\ isLast(EndpointQueue, p)$ **then**
15 $\quad\quad\quad Cands = Candidates[p.Type]$;
16 $\quad\quad\quad$**while** $length(Cands) > 0$ **do**
17 $\quad\quad\quad\quad D = selectLeastUse(Cands, CompCount)$;
18 $\quad\quad\quad\quad d = inst(p, Constraints, D, CoveredArea)$;
19 $\quad\quad\quad\quad$**if** $d \neq NULL$ **then**
20 $\quad\quad\quad\quad\quad RoadScen \Longleftarrow d.generateRoad()$;
21 $\quad\quad\quad\quad\quad RoadScen \Longleftarrow connect(p, d)$;
22 $\quad\quad\quad\quad\quad Count ++$;
23 $\quad\quad\quad\quad\quad CoveredArea += d.getCoveredArea()$;
24 $\quad\quad\quad\quad\quad CompCount[D] += 1$;
25 $\quad\quad\quad\quad\quad EndpointQueue.add(d.getEndpoints())$;
26 $\quad\quad\quad\quad\quad$**break**;
27 $\quad\quad\quad\quad$**end**
28 $\quad\quad\quad\quad$**else**
29 $\quad\quad\quad\quad\quad Cands.remove(D)$;
30 $\quad\quad\quad\quad$**end**
31 $\quad\quad\quad$**end**
32 $\quad\quad$**end**
33 $\quad$**end**
34 $\quad RoadScenSet += RoadScen$;
35 **end**

---

mented in Sec. III-A; (2) $TotolCount$, the number of instantiated road components included in one road scenario; (3) $Constraints$, the constraints about parameters of each road component (e.g., the valid range of road length and lane width), which need to be satisfied, when road components are instantiated, to make the generated road scenarios as realistic as possible; and (4) $Candidates$, the candidate road components from the 242 programming templates that can be connected to each type of endpoints. Each road component has one or multiple endpoints that can be connected to the starting point of other road components. For example, a roundabout has three endpoints. Hence, we summarize the types of different endpoints, and compute their candidate road components. For example, a 2-lane bidirectional solid-line endpoint can be connected to a straight road with 2-lane bidirectional solid-line. The output of Algorithm 1 is a set of road scenarios $RoadScenSet$ in the form of MATLAB scripts.

This algorithm starts by initializing $CompCount$, which records the number of times each of the road components in $CompList$ is used (Line 1). $CompCount$ is used to guide our algorithm to favor the selection of least used road components during generation. Then, it generates one road scenario $RoadScen$ in each loop iteration until certain type of budget is reached, e.g., a time budget of 24 hours is reached (Line 2-35). Specifically, in each iteration, it selects a least used road

component C from $CompList$ as the first road component to ensure geometry diversity (Line 5), and instantiates it to get an instance c of C which satisfies $Constraints$ (Line 6). Then, it updates $RoadScen$, updates $Count$ to record the number of used road components, updates $CoveredArea$ to record the occupied area of the road scenario, updates $CompCount$ to record the usage of C, and adds the endpoints of c to a queue $EndpointQueue$ (Line 7-11).

As long as the current road scenario can be potentially further connected to other road components, i.e., $Count$ is less than $TotalCount$ and $EndpointQueue$ is not empty (Line 12), it pops from $EndpointQueue$ an endpoint p from which $RoadScen$ is further expanded (Line 13). If $random()$ (returning either $true$ or $false$) returns $true$ or p is the last one in $EndpointQueue$ (Line 14), it starts to expand $RoadScen$ at p (Line 15-31). Here, the randomness caused by $random()$ is leveraged to ensure topology diversity.

Then, it obtains from $Candidates$ the candidate road components $Cands$ that can be connected to p according to the type of p (Line 15). Next, it selects the least used road component D from $Cands$ to ensure geometry diversity (Line 17), and instantiates it to get an instance d of D which satisfies $Constraints$, matches with p, and does not overlap with the current road scenario (Line 18). Specifically, it uses the covered area of $RoadScen$ and the covered area of d, i.e., $CoveredArea$ and $d.getCoveredArea()$, to determine whether overlap occurs. If such a d is found (Line 19), it updates $RoadScen$ by adding d and connecting to d through p (Line 20-21), updates $Count$, $CoveredArea$, $CompCount$, and $EndpointQueue$ (Line 22-25), and breaks to continue expanding $RoadScen$ at other endpoints (Line 26). If such a d is not found, it removes D from $Cands$ (Line 29), and tries to select the next least used road component from $Cands$.

C. Road Scenario Deduplication

The generated road scenarios can be similar in their topology, i.e., the types of road components and the connections between road components in two road scenarios can be similar, thus hurting the topology diversity. Hence, we propose a similarity metric to measure the topology similarity, and use it to remove duplicated road scenarios to ensure diversity.

We first define a road scenario as an undirected graph in Definition 1 to model the topology of a road scenario.

Definition 1: A generated road scenario can be modeled as an undirected graph $\mathbb{G} = \langle V, E \rangle$, where V is a set of vertices denoting the road components within the road scenario, and $E \subseteq V \times V$ is a set of undirected edges denoting the connections between road components.

Definition 2: Given two road scenarios $\mathbb{G} = \langle V, E \rangle$ and $\mathbb{G}' = \langle V', E' \rangle$, and $e = (u, v) \in E$, $u, v \in V$, if $\exists\ u', v' \in V'$, $(u', v') \in E'$ such that $Type(u, v) = Type(u', v')$, e is regarded as duplicated in $\mathbb{G}'$, denoted as $DE(e, \mathbb{G}') = 1$; otherwise, $DE(e, \mathbb{G}') = 0$. Here, $Type(u, v)$ returns the types of road component u and v.

Definition 3: Given two road scenarios $\mathbb{G} = \langle V, E \rangle$ and $\mathbb{G}' = \langle V', E' \rangle$, and $u \in V$, if $\forall\ (u, v_i) \in E$, $v_i \in V$ such that $DE((u, v_i), \mathbb{G}') = 1$, u and its connections are regarded

as duplicated in $\mathbb{G}'$, denoted as $DV(u, \mathbb{G}') = 1$; otherwise, $DV(u, \mathbb{G}') = 0$.

Based on Definition 2 and 3, we define a similarity metric $Sim_{(\mathbb{G}_1, \mathbb{G}_2)}$ to measure the topology similarity between two road scenarios $\mathbb{G}_1$ and $\mathbb{G}_2$, as formulated in Equation 1,

$$Sim_{(\mathbb{G}_1, \mathbb{G}_2)} = \frac{\sum_{i=1}^{\|V_1\|} DV(u_i, \mathbb{G}_2) + \sum_{i=1}^{\|V_2\|} DV(v_i, \mathbb{G}_1)}{\| V_1 \| + \| V_2 \|} \quad (1)$$

where $\mathbb{G}_1 = \langle V_1, E_1 \rangle$, $\mathbb{G}_2 = \langle V_2, E_2 \rangle$, $u_i \in V_1$ and $v_i \in V_2$.

Definition 4: Given two road scenarios $\mathbb{G} = \langle V, E \rangle$ and $\mathbb{G}' = \langle V', E' \rangle$, $\mathbb{G}$ and $\mathbb{G}'$ are regarded as duplicated in the topology if $Sim_{(\mathbb{G}, \mathbb{G}')} = 1$.

Based on Definition 4, we remove duplicated road scenarios to ensure the topology diversity. Notice that we can also use this similarity metric to keep the road scenarios whose similarity to existing road scenarios is below a threshold.

D. Joint Simulation

Given the deduplicated road scenarios in the form of MATLAB scripts, we first adopt MATLAB to compile the scripts into RoadRunner HD map files (i.e., rrhd files). Here we use the rrhd format because RoadRunner provides programmatic interfaces to import rrhd files and export HD map file types needed by various autonomous driving systems (e.g., Apollo and Autoware) as well as 3D scene file types required by various simulators (e.g., SORA-SVL and CARLA). Then, we use RoadRunner to convert rrhd files into target HD map files and 3D scene files for joint simulation on a target simulator with a target autonomous driving system.

IV. EVALUATION

We have implemented a prototype of ROADGEN in 354K lines of Python and MATLAB code, and released the source code of our prototype as well as all the experimental data at our website `https://roadgen.github.io/`. To evaluate the effectiveness and usefulness of ROADGEN, we design the following two research questions (RQs).

- **RQ1 Effectiveness Evaluation**: How is the effectiveness of ROADGEN in generating diverse road scenarios?
- **RQ2 Usefulness Evaluation**: Can the road scenarios generated by ROADGEN be used for simulation?

A. Evaluation Setup

RQ Setup. To answer **RQ1**, we compare the effectiveness of ROADGEN with a baseline approach, referred to as RAND, which randomly selects and connects road components. We set the number of road components included in each road scenario to 4, 5, 6, 7 and 8, respectively, and continuously generate road scenarios for 24 hours using ROADGEN and RAND. We record the number of generated road scenarios after deduplication, the uniqueness rate (i.e., the rate between the number of generated road scenarios after and before deduplication), and the time for covering different road components. We run the experiment 5 times, and report the average results.

To answer **RQ2**, we leverage Apollo 8.0 [2] and SOAR-SVL [9] to demonstrate the usefulness of ROADGEN from three perspectives, the usability of 3D scenes, the usability of HD maps, and the usability for joint simulation.

(a) 5 Road Components
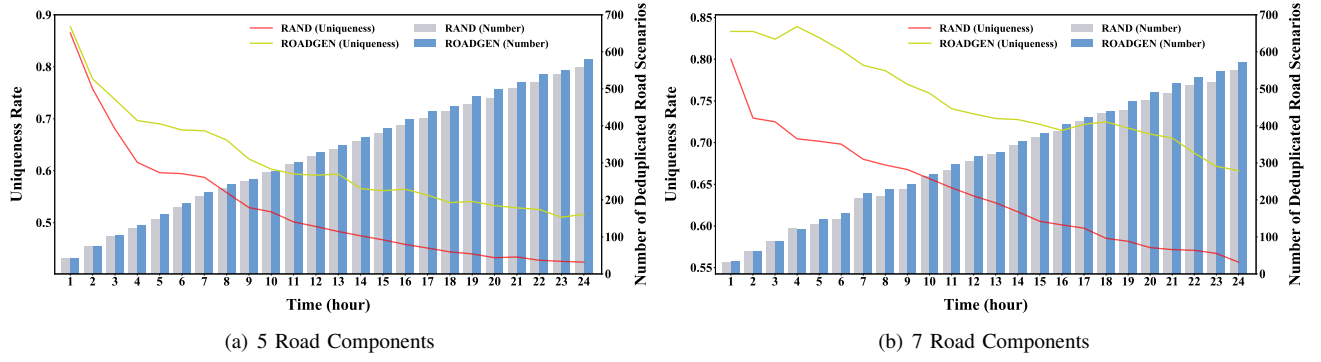

(b) 7 Road Components

Fig. 3.   The Number of Deduplicated Road Scenarios and Uniqueness Rate of ROADGEN and RAND over Time

TABLE I

STATISTICS ABOUT ROAD SCENARIOS GENERATED WITHIN 24 HOURS

Road Scenario Size	Number		Uniqueness	
	ROADGEN	RAND	ROADGEN	RAND
4 Road Components	400	390	0.185	0.150
5 Road Components	580	558	0.516	0.424
6 Road Components	549	530	0.629	0.534
7 Road Components	571	551	0.666	0.557
8 Road Components	458	432	0.602	0.527

Fig. 4.   Comparison of Time for Covering Different Road Components

Experimental Environment Setting. We conduct all the experiments on a Ubuntu 20.04.4 LTS server with 4 NVIDIA GeForce RTX 3090 GPUs, Inter Core i9-10980XE CPU with 3.00GHz processor, and 128GB memory.

B. Effectiveness Evaluation (RQ1)

Overall Results. Table I presents the statistics about the road scenarios generated by ROADGEN and RAND within 24 hours. The first column gives the road scenario size in terms of number of road components, the second and third columns show the number of generated road scenarios after deduplication, and the fourth and fifth columns list the uniqueness rate.

Specifically, across the five groups of experiments with respect to different road scenario sizes, ROADGEN generate a minimum number of 400 deduplicated road scenarios when road scenario size is set to 4, and a maximum number of 580 deduplicated road scenarios when road scenario size is set to 5. In addition, on average, ROADGEN generates 3.9% more deduplicated road scenarios than RAND across all the five groups of experiments, which is statistically significant.

Besides, in terms of uniqueness rate, ROADGEN significantly outperforms RAND by 19.3% on average. In the four groups of experiments with road scenario size ranging from 5 to 8, the uniqueness rate of ROADGEN fluctuates between 0.5 and 0.7. However, in the group of experiments with road scenario size setting to 4, ROADGEN exhibits a relatively low uniqueness rate. This also holds for RAND. It is because the fewer the number of used road components, the more likely the generated road scenarios will have a higher similarity.

Detailed Results over Time. Fig. 3 illustrates a detailed comparison over time between ROADGEN and RAND when road scenario size i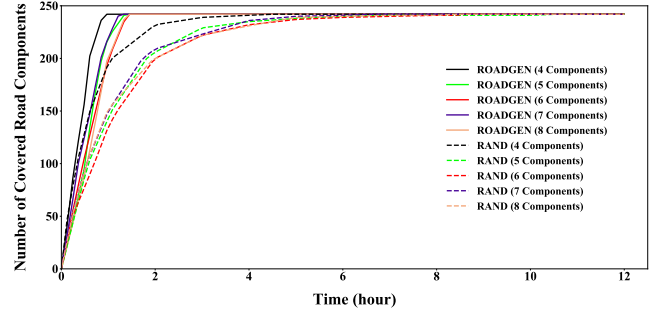s set to 5 and 7 in terms of the number of deduplicated road scenarios and the uniqueness rate. Due to space limitation, we provide the results when road scenario size is set to 4, 6 and 8 at our website.

Specifically, in the early stages of generation, the number of deduplicated road scenarios generated by ROADGEN is smaller than that of RAND. This is because ROADGEN consumes more time than RAND due to our guidance computation, and thus generates a smaller number of road scenarios before deduplication. However, as time goes by, the number of covered road components increases, and RAND gradually generates more duplicated road scenarios, and thus its deduplicated road scenarios are gradually less than those generated by ROADGEN. This results in a gradual decline in the uniqueness rate, while the difference between ROADGEN and RAND becomes more significant.

Time for Covering Road Components. Fig. 4 presents the number of different road components covered by generated road scenarios over time. The solid lines represent the five groups of experiments of ROADGEN, and the dashed lines represent the five groups of experiments of RAND.

Specifically, ROADGEN respectively spends 0.89, 1.34, 1.40, 1.23 and 1.38 hours to cover all the 242 road components when road scenario size is respectively set to 4, 5, 6, 7 and 8, while RAND respectively costs 4.53, 10.57, 8.61, 6.64 and 8.42 hours. On average, ROADGEN spends 1.24 hours to cover all the 242 road components, while RAND costs 7.75 hours; i.e., ROADGEN is 83.9% faster than RAND in covering all the 242 road components.

Summary. These results indicate that our guided approach

(a) Demonstration of 3D Scenes   (b) Demonstration of HD Maps   (c) Demonstration of Joint Simulation
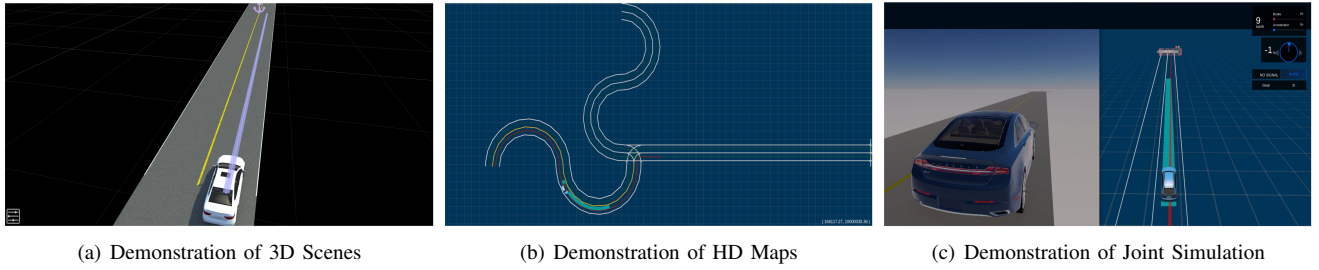
Fig. 5. A Case Study that Demonstrates the Usability of 3D Scene Files and HD Maps and the Usability for Joint Simulation

TABLE II
SUCCESS RATE OF COMPILING SCRIPTS INTO 3D SCENE FILES

Road Scenario Size	Total Number	Sample Size	Success Rate
4 Road Components	400	196	96%
5 Road Components	580	231	96%
6 Road Components	549	226	95%
7 Road Components	571	230	93%
8 Road Components	458	228	92%

ROADGEN is effective in generating more diverse road scenarios in the same time than the baseline random approach, and covers all the different road components more quickly.

C. Usefulness Evaluation (RQ2)

Due to the large number of road scenarios in the format of MATLAB scripts generated in **RQ1**, we evaluate their usefulness by a sampling approach. Specifically, we set the confidence level to 95% and the margin error to 5% to determine the sample size, and randomly sample road scenarios generated by ROADGEN in **RQ1**. The third column of Table II reports the sample size under each road scenario size.

Usability of 3D Scenes. We leverage RoadRunner to compile the sampled road scenario scripts into 3D scene files. The last column of Table II lists the success rate of compilation. Specifically, more than 92% of the road scenarios scripts can be successfully compiled into 3D scene files. We investigate the cases where compilation fails, and find that it is caused by the excessive curvature of certain U-shaped road components. We will address this issue in future updates. Moreover, we validate all 3D scene files using the built-in scenario simulation tool in RoadRunner, and the vehicles included therein are able to recognize lane markings, and perform path planning, lane changes and other behaviors with all 3D scene files. One sample demonstration is illustrated in Fig. 5(a).

Usability of HD Maps. We utilize RoadRunner to convert those sampled road scenario scripts, which are successfully compiled into 3D scene files, into HD map files for the latest Apollo 8.0. We import these HD map files into Apollo, and restart Dreamview (i.e., Apollo's fullstack HMI service) to load the HD maps. We determine the usability of HD maps through manual verification, i.e., running the routing testing in Apollo's built-in sim-control mode, which does not require any third-party simulators. We set points of interest in sim-control, and let Apollo conduct path planning. All HD map files are successfully used in routing testing in sim-control mode. One sample demonstration is shown in Fig. 5(b).

Usability for Joint Simulation. We first import each 3D scene file into Unity, and manually set the SpawnInfo which contains the starting position and end position of a target vehicle. Then, we import the resulting scene binary file called AssetBundle into SORA-SVL, and bridge it with Apollo 8.0. We use the SORA-SVL's Python API to start joint simulation in Dreamview and SORA-SVL. All the joint simulations are successful. One sample demonstration is shown in Fig. 5(c).

Summary. These results indicate that more than 92% of the generated road scenarios are useful for joint simulation.

D. Limitations

While the proposed approach demonstrates promising results, it still suffers several limitations. First, we define eight types of typical road components, which is not meant to be exhaustive but is to illustrate the feasibility of our approach. We plan to further extend the types of road components according to regulations in different countries.

Second, we currently only focus on the road-level scenario (i.e., layer 1 of the 5-layer model [1]) without taking into account traffic signs, static and dynamic objects, etc. in the upper layers. We are integrating these elements into ROADGEN.

Third, we evaluate the usefulness of ROADGEN only on SORA-SVL with Apollo. However, RoadRunner supports the export of HD map files and 3D scene files in various formats required by various simulators and autonomous driving systems. We believe that ROADGEN is still applicable in other simulators and autonomous driving systems.

V. CONCLUSIONS

This paper propose ROADGEN to systematically generate diverse road scenarios in both topology and geometry. First, eight types of typical road components are defined and implemented. Then, ROADGEN uses a guided algorithm to connect road components to generate road scenarios, and uses a similarity metric to remove duplicated road scenarios. Our experimental results have demonstrated the promising effectiveness and usefulness of ROADGEN in generating diverse road scenarios for joint simulation. Our dataset of road scenarios is also released for fostering simulation testing. In the future, we plan to extend ROADGEN to support more types of road components and integrate upper-layer scenario elements, and investigate the applicability of ROADGEN in other simulators and autonomous driving systems.

REFERENCES

[1] G. Bagschik, T. Menzel, and M. Maurer, "Ontology based scene creation for the development of automated vehicles," in *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium*, 2018, pp. 1813–1820.

[2] Baidu. (2022) Apollo: An open autonomous driving platform. [Online]. Available: https://github.com/ApolloAuto/apollo

[3] L. Balasubramanian, J. Wurst, M. Botsch, and K. Deng, "Traffic scenario clustering by iterative optimisation of self-supervised networks using a random forest activation pattern similarity," in *Proceedings of the 2021 IEEE Intelligent Vehicles Symposium*, 2021, pp. 682–689.

[4] S. K. Bashetty, H. B. Amor, and G. Fainekos, "Deepcrashtest: Turning dashcam videos into virtual crash tests for automated driving systems," in *Proceedings of the 2020 IEEE International Conference on Robotics and Automation*, 2020, pp. 11 353–11 360.

[5] J. Cai, W. Deng, H. Guang, Y. Wang, J. Li, and J. Ding, "A survey on data-driven scenario generation for automated vehicle testing," *Machines*, vol. 10, no. 11, p. 1101, 2022.

[6] W. Ding, C. Xu, M. Arief, H. Lin, B. Li, and D. Zhao, "A survey on safety-critical driving scenario generation—a methodological perspective," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 7, pp. 6971–6988, 2023.

[7] E. Esenturk, S. Khastgir, A. Wallace, and P. Jennings, "Analyzing real-world accidents for test scenario generation for automated vehicles," in *Proceedings of the 2021 IEEE Intelligent Vehicles Symposium*, 2021, pp. 288–295.

[8] A. Gambi, T. Huynh, and G. Fraser, "Generating effective test cases for self-driving cars from police reports," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 257–267.

[9] Y. Huai. (2023) Sora-svl. [Online]. Available: https://www.ics.uci.edu/~yhuai/SORA-SVL/

[10] 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, pp. 182–193, 2016.

[11] J. Kerber, S. Wagner, K. Groh, D. Notz, T. Kühbeck, D. Watzenig, and A. Knoll, "Clustering of the scenario space for the assessment of automated driving," in *Proceedings of the 2020 IEEE Intelligent Vehicles Symposium*, 2020, pp. 578–583.

[12] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *Proceedings of the 2019 IEEE Intelligent Vehicles Symposium*, 2019, pp. 2352–2358.

[13] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium*, 2018, pp. 1821–1827.

[14] M. E. Mortenson, *Mathematics for computer graphics applications*. Industrial Press Inc., 1999.

[15] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[16] I. Paranjape, A. Jawad, Y. Xu, A. Song, and J. Whitehead, "A modular architecture for procedural generation of towns, intersections and scenarios for testing autonomous vehicles," in *Proceedings of the 2020 IEEE Intelligent Vehicles Symposium*, 2020, pp. 162–168.

[17] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE access*, vol. 8, pp. 87 456–87 477, 2020.

[18] S. Rietsch, S.-Y. Huang, G. Kontes, A. Plinge, and C. Mutschler, "Driver dojo: A benchmark for generalizable reinforcement learning for autonomous driving," *arXiv preprint arXiv:2207.11432*, 2022.

[19] E. Rocklage, H. Kraft, A. Karatas, and J. Seewig, "Automated scenario generation for regression testing of autonomous vehicles," in *Proceedings of the 2017 IEEE 20th international conference on intelligent transportation systems*, 2017, pp. 476–483.

[20] M. Scholtes, L. Westhofen, L. R. Turner, K. Lotto, M. Schuldes, H. Weber, N. Wagener, C. Neurohr, M. H. Bollmann, F. Körtke, J. Hiller, M. Hoss, J. Bock, and L. Eckstein, "6-layer model for a structured description and categorization of urban traffic and environment," *IEEE Access*, vol. 9, pp. 59 131–59 147, 2021.

[21] B. Schütt, J. Ransiek, T. Braun, and E. Sax, "1001 ways of scenario generation for testing of self-driving cars: A survey," in *Proceedings of the 2023 IEEE Intelligent Vehicles Symposium*, 2023, pp. 1–8.

[22] J. Sun, H. Zhang, H. Zhou, R. Yu, and Y. Tian, "Scenario-based test automation for highly automated vehicles: A review and paving the way for systematic safety assurance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 14 088–14 103, 2022.

[23] Y. Tang, Y. Zhou, K. Yang, Z. Zhong, B. Ray, Y. Liu, P. Zhang, and J. Chen, "Automatic map generation for autonomous driving system testing," *arXiv preprint arXiv:2206.09357*, 2022.

[24] H. Tian, Y. Jiang, G. Wu, J. Yan, J. Wei, W. Chen, S. Li, and D. Ye, "Mosat: finding safety violations of autonomous driving systems using multi-objective genetic algorithm," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 94–106.

[25] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving," in *Proceedings of the IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 982–988.

[26] J. Wang, A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, and R. Urtasun, "Advsim: Generating safety-critical scenarios for self-driving vehicles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9909–9918.

[27] Waymo. (2021) Waymo safety report - 2021. [Online]. Available: https://ltad.com/resources/waymo-safety-report-2021.html

[28] X. Zhang, J. Tao, K. Tan, M. Törngren, J. M. G. Sánchez, M. R. Ramli, X. Tao, M. Gyllenhammar, F. Wotawa, N. Mohan, M. Nica, and H. Felbinger, "Finding critical scenarios for automated driving systems: A systematic mapping study," *IEEE Transactions on Software Engineering*, vol. 49, no. 3, pp. 991–1026, 2023.

[29] X. Zhang and Y. Cai, "Building critical testing scenarios for autonomous driving from real accidents," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2023, pp. 462–474.

[30] Z. Zhong, Y. Tang, Y. Zhou, V. d. O. Neves, Y. Liu, and B. Ray, "A survey on scenario-based testing for automated driving systems in high-fidelity simulation," *arXiv preprint arXiv:2112.00964*, 2021.

[31] Y. Zhou, G. Lin, Y. Tang, K. Yang, W. Jing, P. Zhang, J. Chen, L. Gong, and Y. Liu, "Flyover: A model-driven method to generate diverse highway interchanges for autonomous vehicle testing," in *Proceedings of the 2023 IEEE International Conference on Robotics and Automation*, 2023, pp. 11 389–11 395.