Testing Diverse Geographical Features of Autonomous Driving Systems

Seongdeok Seo∗
Hyundai Motors
South Korea
Emille@hyundai.com

Judy Lee∗
ADP
United States
judy.lee@adp.com

Mijung Kim†
UNIST
South Korea
mijungk@unist.ac.kr

***Abstract*—Testing in various driving scenarios is one of the essential methods to enhance the reliability of autonomous driving systems (ADS). Existing ADS testing research has shown effectiveness in detecting safety violations by generating diverse driving scenarios. However, they do not consider the various geographical features and thus have limited ability to find safety violations caused by complex geographical features. Our paper addresses this limitation by analyzing a given high-definition map and collecting its geographical features. We leverage this information and develop a technique for generating corner case scenarios that exercise diverse geographical features such as curves and slopes. Our approach first generates the ego-vehicle's driving routes so that they achieve full lane coverage on the entire map, then clusters those routes by geographical features, and constructs driving scenarios by adding other objects and environments. In our experiments on Autoware-Universe, we evaluate our technique with six high-definition maps from the Carla simulator. Our results show that driving scenarios generated by our tool effectively exercise more diverse geographical features than existing work. As a result, our tool uncovers new safety violations that are caused by complex geographical features and would not be detected by existing work.**

***Index Terms*—Testing Autonomous Driving System, Autonomous Driving Scenario Generation, Model-based Scenario Generation**

I. Introduction

Recent tests of self-driving cars have faced significant challenges, including traffic congestion from stuck vehicles [1]–[5] and collisions [6], [7] leading to operational restrictions. Autonomous vehicles experience twice as many accidents per mile as conventional cars, causing 93% of Americans to express safety concerns [8]. Consequently, testing autonomous systems in diverse settings is essential to identify and address safety issues, enhancing overall road safety.

When testing autonomous driving systems, exploring various road features is essential. Roads with vertical variables, such as slopes, and roads with curves require precise steering control and thus strongly correlate with accident risks [10]–[12]. Different road types, such as highways and general roads, also affect accident risks [10]–[12]. Therefore, including these geographical features in testing scenarios is crucial. Real-world maps are usually complex, featuring various intersections and highly unpredictable elements [13]. Recent

Fig. 1: An example of complex junctions that autonomous driving systems fail to navigate [9]

investigations reveal that autonomous vehicles often fail to navigate complex junctions [9], as illustrated in Fig. 1. Thus, considering geographical features is essential for ensuring the high reliability of autonomous driving systems.

Virtual simulation testing has emerged as a crucial method for verifying autonomous driving systems, offering controlled, modifiable, and reusable scenarios without incurring financial or physical risks. This approach allows for handling a large number of scenarios without the need for physical hardware [14]. Various techniques have been developed for testing autonomous driving systems in virtual simulation. These techniques can be categorized into two groups based on the outputs they generate: (i) scenario-generation approach and (ii) map-generation approach.

The scenario-generation approach [15]–[36] automatically generates autonomous driving scenarios on given maps. While scenarios generated by these scenario-generation techniques are effective in uncovering safety violations, caused by other vehicles, obstacles, or environmental variables, they often overlook violations arising from geographical features. Such features can introduce significant complexity and confusion on roads, leading to potential safety hazards. Prior scenario-generation research has not adequately addressed these factors, largely due to a lack of criteria for identifying geographical features along driving routes and the difficulty in achieving lane coverage when generating routes for the ego vehicle. Additionally, geographical features that contribute to violations are rare, further complicating their identification and analysis.

∗ This work was done when Seongdeok Seo and Judy Lee were at UNIST.
† Corresponding author

The map-generation approach [37]–[44] focuses on creating maps with newly generated road structures, such as merging lanes and curved roads. While these techniques are useful for testing various geographical features, they are limited in their ability to accurately represent complex real-world geographical characteristics, such as elevation changes and distinctions between highways and regular roads. Moreover, because these techniques produce artificially generated maps as outputs, they may not be as effective for testing geographical features within simulated environments typically converted from real-world maps. The ability to handle real-world maps is crucial for testing autonomous driving systems, as these maps often contain complex and unpredictable elements [13] that autonomous systems must be capable of navigating. Unfortunately, the map-generation approach may fall short of fully representing and testing the complex geographical features found in real-world environments.

In this paper, to address these challenges from the scenario-generation and map-generation approaches, we aim to test diverse geographical features by *generating scenarios that thoroughly exercise geographical features on a given map*. Our work is based on two key insights. First, it is crucial to generate driving scenarios that encompass all roads on a given map, ensuring that no geographical features are overlooked. Second, to effectively test driving routes with diverse geographical features, it is essential to analyze and classify the geographical characteristics of the roads within each route.

Based on these insights, we propose MAPSCE-GEN, a model-based scenario-generation technique for detecting safety violations specifically caused by geographical features within the given map. Unlike previous scenario-generation approaches [15]–[36], which primarily address safety violations from other sources, our focus is specifically on those arising from geographical features. MAPSCE-GEN constructs a data structure called the route dictionary, which classifies extracted driving routes from the map into their geographical characteristics. This classification enables the generation of driving scenarios tailored to specific geographical features, facilitating the detection of challenging corner-case scenarios. Unlike map-generation approaches [37]–[44], which focus on directly creating road structures, MAPSCE-GEN identifies and utilizes existing geographical features within a given map to generate scenarios that thoroughly exercise those features. Additionally, MAPSCE-GEN considers diverse geographical features such as speed limits and elevation, which have not been addressed by previous techniques [37]–[44].

We demonstrate the effectiveness of MAPSCE-GEN using Autoware-Universe [45], an industrial-grade autonomous driving system platform, in conjunction with the CARLA simulator [46]. The experimental results show that MAPSCE-GEN can utilize all roads in given maps to generate driving routes and identify safety violations caused by previously overlooked geographical features. Additionally, our findings indicate that MAPSCE-GEN outperforms existing methods [34]–[36] in generating corner cases related to geographical features and significantly improves lane coverage compared to existing methods. The new routes generated by MAPSCE-GEN have a violation rate approximately 2.63 times higher than those identified by existing methods [34]–[36]. Furthermore, our approach effectively covers a wide range of road characteristics.

In summary, the contributions of this paper are as follows:

- MAPSCE-GEN ensures comprehensive coverage of all lanes within the provided high-definition map. By generating driving routes for every road present on the map, the tool ensures that no geographical feature or lane is overlooked during scenario generation.
- MAPSCE-GEN constructs a data structure for route classification which provides a method to organize driving routes based on their geographical features. This facilitates the analysis of driving scenarios with diverse geographical characteristics, aiding in the discovery of challenging corner cases.
- We implement MAPSCE-GEN and demonstrate its effectiveness on the Autoware-Universe platform using the CARLA simulator. This implementation provides a practical solution for detecting safety violations caused by geographical features. The implementation details, as well as our source code, are available on our website: **https://sites.google.com/view/mapsce-gen**

II. BACKGROUND

A. Autonomous Driving Scenarios

Given that driving scenarios—serving as test inputs in simulation testing—involve numerous parameters, it is crucial to determine how to construct them and which parameters to prioritize. The scenario layer model [47] has been proposed to represent the structure of driving scenarios. According to this model, an autonomous driving scenario comprises three layers: geographical layer, object layer, and environment layer.

Geographical Layer contains parameters related to the geographical scope of the high-definition map where the driving scenarios are executed. It represents both the features and the state of roads, including road-level details, traffic infrastructure, and temporary manipulations of road-level and traffic infrastructure. During scenario generation, the parameters in the geographical layer are determined by the driving route of the ego-vehicle. **Object Layer** represents other vehicles, obstacles, or pedestrians in driving scenarios, while **Environment Layer** encompasses external environmental factors such as road friction and weather conditions.

B. Scenario-generation Approach

The scenario-generation approach can be categorized into two groups based on how seed scenarios are prepared: (i) manually crafted seeds [15]–[31] and (ii) automatically generated seeds [32]–[36].

The scenario-generation approach with manually crafted seeds [15]–[31] involves incrementally modifying existing driving scenarios to detect safety violations. Most utilize mutation-based fuzzing, a feedback-driven approach that mutates manually crafted scenarios to generate those violating predefined safety requirements. While effective in identifying

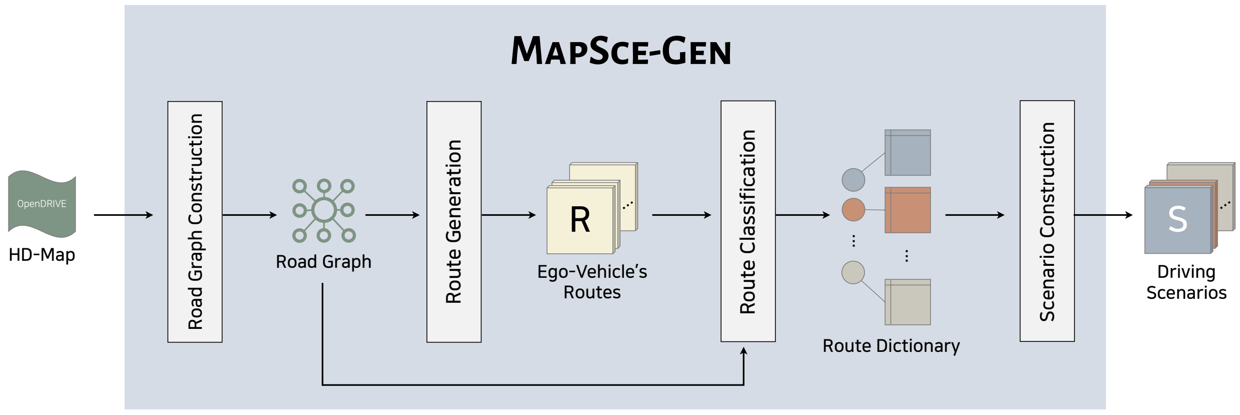


Fig. 2: Overview of our approach, MAPSCE-GEN

potential vulnerabilities, these generated scenarios often lack diversity in the geographical layer due to the similarity in geographical features between seed scenarios and mutated scenarios. This limitation affects the robustness and adaptability evaluation of autonomous driving systems, potentially impeding the detection of safety violations associated with various geographical features.

To address the limitations of manually crafted seeds and increase the variety of driving routes, techniques for automatic seed generation [32]–[36] have been proposed. These methods utilize automated input generation to construct driving routes without manual crafting, aiming to enhance the diversity of generated scenarios. However, these existing studies primarily focus on identifying safety violations caused by the object layer and do not address safety violations resulting from the geographical layer. Furthermore, they often fail to cover all lanes on the entire map and struggle to incorporate diverse geographical features, which are essential for identifying corner case scenarios causing safety violations.

C. Map-generation Approach

To test the geographical layer, some studies directly construct an artificial one. The map-generation approach is specifically designed to generate maps or road structures [37]–[44]. These tools construct challenging road networks and focus on predicting whether a particular road structure may cause safety violations. These studies have made significant efforts to generate complex road networks, which can be utilized to identify various safety violations, including lane invasions.

However, most existing approaches focus only on 2D geographical features, such as road curvature [37]–[42] or lane changes (e.g, merge, split, etc.) [43], while neglecting other types of geographical features commonly found in real-world scenarios, like elevation, the number of connected lanes, and speed limits, which are specified in high-definition maps.

Due to significant demands for testing autonomous driving systems on real-world roads in industry, real-world maps are often converted into simulation-compatible formats, such as OpenStreetMap [48]. Consequently, it becomes necessary to test autonomous driving systems on given maps rather than artificially generated ones. Given this context, it is also essential to adopt an approach that explores possible driving routes within a given map to generate various driving scenarios.

III. APPROACH OVERVIEW

Our approach called MAPSCE-GEN is a map-aware scenario generation technique designed for testing diverse geographical features of autonomous driving systems. Fig. 2 presents the overview of the workflow employed by MAPSCE-GEN. It consists of 4 phases: road graph construction, ego vehicle's route generation, route classification, and scenario construction.

First, our approach constructs a road graph (Section IV-A), which is a data structure designed to represent connections between roads and junctions in the given high-definition map. The road graph contains essential map information such as the connectivity information between lanes and the geographical features of roads. This component serves as the input for MAPSCE-GEN's route generation (Section IV-B) and route classification (Section IV-C) components. Next, our approach generates driving routes based on their geographical features. Unlike previous work that solely considers only roads connected to a junction [34]–[36], our approach extends driving routes by leveraging the road-tracking methods to track the full route. Our approach then takes the ego vehicle's routes that are generated in the previous step, categorizes them based on their geographical characteristics, and creates a route dictionary (Section IV-C2). The dictionary has unique keys representing geographical features and stores as values the driving routes exercising the corresponding geographical features. Here, route keys are systematically computed using comprehensive route information such as roads and their geographical features (e.g., curvature, elevation, etc.). Finally, in the scenario construction step (Section IV-D), MAPSCE-GEN utilizes the geographically diverse routes selected from the route dictionary and constructs complete driving scenarios by adding the object layer and environment layer. This process involves generating NPC vehicles that perform legal driving behaviors, akin to existing work [34], [35]. It also integrates environmental elements such as weather conditions and road friction, thereby generating realistic driving scenarios.

IV. MapSce-Gen

In this section, we discuss how MapSce-Gen generates driving scenarios that exercise diverse geographical features.

A. Road Graph Construction

Our approach first constructs Road Graph, that represents connections of roads and junctions of a given high-definition map. The primary aim of the road graph is to develop a data structure facilitating the easy storage and management of essential information extracted from high-definition maps, thereby organizing and managing map data effectively to enable accurate and efficient utilization of map information. In this paper, precise map data in the ASAM OpenDRIVE format [49] serves as the input, transformed into the road graph object. Processing OpenDRIVE is cost-effective because the file size is much smaller than the lanelet format [50]. OpenDRIVE, a standard format for representing high-definition map data, enhances the organization and management of data when converted into the road graph data structure. The developed road graph data structure takes the form of a directed graph.

The road graph consists of nodes representing lanes and directed edges indicating connections between lanes. This structure is ideal for representing the layout and connectivity of roads, enabling the depiction of the road network's structure and navigation routes between lanes. Edges within the road graph are arranged in a successor-predecessor format, with relationships defined based on lane directionality. This arrangement facilitates tracking lane traversal within the road graph. The detailed contents of the road graph are implemented using internal classes, with concepts such as lanes, roads, and junctions each represented as individual classes. Their relationships are established through a parent-child structure, defining that lanes are part of roads, and roads are components of junctions, thereby maintaining a hierarchical representation of data. These dependencies adhere to OpenDRIVE format, the standard of high-definition maps.

Algorithm 1 presents the process of generating the road graph. The algorithm uses a string, hd_map_str, representing the high-definition map as input. The output comprises two dictionaries: the Road Dictionary $\mathbb{R}$ and the Junction Dictionary $\mathbb{J}$. $\mathbb{R}$ maps road identifiers to their corresponding road information, while $\mathbb{J}$ maps junction identifiers to the associated junction data. Initially, both dictionaries are empty (Line 1–2). The algorithm then enters a junction analysis phase, iterating over all junctions in the high-definition map (Lines 3–9). For each junction j, the algorithm creates a set c_road to store the identifiers of roads incoming to the junction (Lines 5–7). This set is used to create a Junction object, which is added to $\mathbb{J}$ with j's identifier as the key. Following this, the algorithm begins a road analysis phase, iterating over all roads in the high-definition map. For each road r, a dictionary $lane\_dict$ is constructed to store the lanes belonging to the road (Lines 10–21). The algorithm also retrieves the $link$ and $road\_length$ information about the road (Lines 12–14). For each lane section in r, a lane classification process updates $lane\_dict$. Using this data, a new_road object is created. If r's junction value is "-1," it means that this road is not part of any junction. In such cases, the new_road is added to $\mathbb{R}$ (Lines 16–17). Conversely, if the junction value is not "-1," then the new_road is added to $\mathbb{J}$ with the junction value as the key (Lines 18–19). Finally, the dictionaries $\mathbb{R}$ and $\mathbb{J}$, which comprehensively map the road and junction data of the high-definition map, are returned as output of Algorithm 1.

Algorithm 1: RoadGraph Construction

```
Input: $hd\_map\_str$ String of high-definition map
Output: RoadGraph
1:  $\mathbb{R} \leftarrow dict()$
2:  $\mathbb{J} \leftarrow dict()$
    // Junction Analysis
3:  for all $j$ in $hd\_map\_str$.findall("junction") do
4:      $c\_road \leftarrow set()$
5:      for all $c$ in $j$.findall("connection") do
6:          $c\_road.add(c.get(\text{"incomingRoad"}))$
7:      end for
8:      $\mathbb{J}[j.get(\text{"id"})] \leftarrow Junc(j.get(\text{"id"}), c\_road)$
9:  end for
    // Road Analysis
10: for all $r$ in $hd\_map\_str$.findall("road") do
11:     $lane\_dict \leftarrow extract\_lanes(r)$
12:     $link \leftarrow r.find(\text{"link"})$
13:     $road\_length \leftarrow r.get(\text{"length"})$
14:     $new\_road \leftarrow Road(lane\_dict, link, road\_length)$
15:     CalcualteCharacteristics($new\_road$)
16:     if $r$.get("junction") == "-1" then
17:         $\mathbb{R}[r.get(\text{"id"})] \leftarrow new\_road$
18:     else
19:         $\mathbb{J}[r.get(\text{"junction"})] \leftarrow new\_road$
20:     end if
21: end for
22: return RoadGraph($\mathbb{R}, \mathbb{J}$)
```

B. Route Generation

The route generation aims to create extended driving routes that were missed by existing work [34]–[36]. Specifically, extended routes generated by MapSce-Gen can handle routes like those illustrated in Fig. 3, where existing approaches fail to cover roads not directly linked to junctions. In the syntax supported by OpenDRIVE [49], it is possible to connect single roads, which commonly exist in high-definition maps. However, existing approaches [34]–[36] do not consider such high-definition map grammar. This leads to gaps in achieving full lane coverage within the map. With the road graph, representing all connection information in the map, MapSce-Gen extends connected single roads to generate driving routes that achieve 100% lane coverage using road-tracking.

In contrast to prior work [34]–[36] that can only handle roads directly connected to junctions, our approach leverages the method of road-tracking to track full routes. It creates driving routes that traverse a single junction and yet utilize all

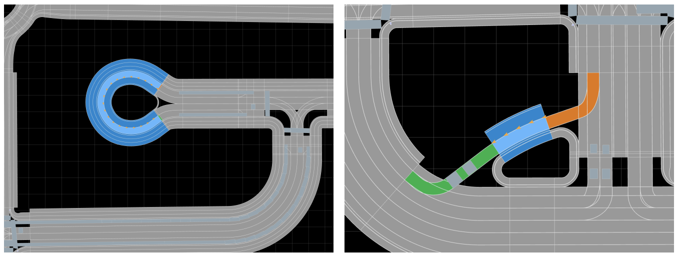
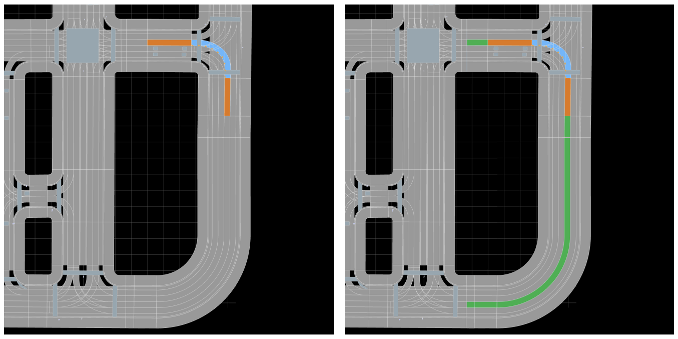


Fig. 3: Examples of complex lanes that are not handled in existing work

individually connected roads. This approach improves the lane coverage of the entire map and provides an advantage in diversifying ego-vehicle's driving routes. As a result, MapSce-Gen helps test a broader area in the map.

Our approach generates the driving route for all junction lanes by covering every lane in the map. Fig. 4 visually represents the full-route generation process, illustrating how routes are expanded. Previous work [34]–[36], which limits route generation to roads directly connected to junctions, presents a significant drawback by omitting lanes not covered. To address this limitation, our methodology aims to overcome the challenge by extending routes beyond those exclusively associated with junctions. This approach ensures a more comprehensive coverage of lanes in the map, thereby enhancing the overall effectiveness of the route generation process.



(a) Driving route generated in existing work (b) Extended driving route generated by MapSce-Gen

Fig. 4: Route extension by our road-tracking method

In Algorithm 2, we explain the process of generating a full route for the ego-vehicle. To create driving routes for the ego-vehicle, MapSce-Gen gathers information about the lanes from the road graph. By leveraging Junction Dictionary $\mathbb{J}$ of the road graph, MapSce-Gen locates all lanes within each junction and their connection information. For each junction lane l, we define l_{pred} and l_{succ} as the predecessor and successor lanes of l, respectively. Starting with l_{pred} and l_{succ}, the algorithm creates two lists, L_{pred} and L_{succ}, respectively (Lines 4–5). The algorithm continues adding predecessors to L_{pred} until the type of the first lane is not a junction (Lines 8–11). Similarly, the algorithm continues adding successors to L_{succ} until the type of the last lane is not a junction (Lines 12–15). Finally, the algorithm concatenate L_{pred}, l, and L_{succ} to form a route R.

Algorithm 2: Route Generation

```
Input: RoadGraph
Output: RouteDictionary
1:  for each junction $j$ in $\mathbb{J}$ of RoadGraph do
2:    for each road $r$ in $j$ do
3:      for each lane $l$ in $r$ do
4:        $L_{pred} \leftarrow list()$
5:        $L_{succ} \leftarrow list()$
6:        $l_{pred} \leftarrow$ Predecessor of $l$
7:        $l_{succ} \leftarrow$ Successor of $l$
          // Tracking roads to find full-route
8:        while type of $l_{pred}$ is not "junction" do
9:          $L_{pred}.add(l_{pred})$
10:         $l_{pred} \leftarrow$ Predecessor of $l_{pred}$
11:       end while
12:       while type of $l_{succ}$ is not "junction" do
13:         $L_{succ}.add(l_{succ})$
14:         $l_{succ} \leftarrow$ Predecessor of $l_{succ}$
15:       end while
16:       $R \leftarrow L_{pred} + [l] + L_{succ}$
17:       $Key_R \leftarrow$ CalculateRouteKey$(R)$
18:       RouteDictionary$(Key_R) \leftarrow R$
19:     end for
20:   end for
21: end for
22: return RouteDictionary
```

C. Route Classification

To uncover safety violations caused by the geographical layer, it is essential to create geographically complex routes for the ego-vehicle's driving paths. To identify such routes, they must be distinguished based on their geographical features.

Unlike previous methodologies that only consider the geographical features of junctions [34]–[36], our approach distinguishes driving routes by considering the geographical features of not only the junction but also the predecessor and successor roads associated with the driving route. Furthermore, our approach does not rely on categorizing junctions into manually predefined classes such as T-shaped junction and Y-shaped junction. Instead, MapSce-Gen performs comprehensive classification by representing each distinguishing feature.

In contrast to existing work [32]–[36], our route classification approach selects ego-vehicle driving routes based on the geographical features of all lanes in the route. The existing work, as depicted in Fig. 5, tends to classify the two routes shown in the figure as identical. However, roads connected to actual junctions can exhibit significantly different features. For instance, the successor road of Fig. 5a is the straight lane, while the successor road of Fig. 5b is the left-curved lane. This limitation in the existing work highlights challenges in finding geographically complex routes. Specifically, the oversimplified

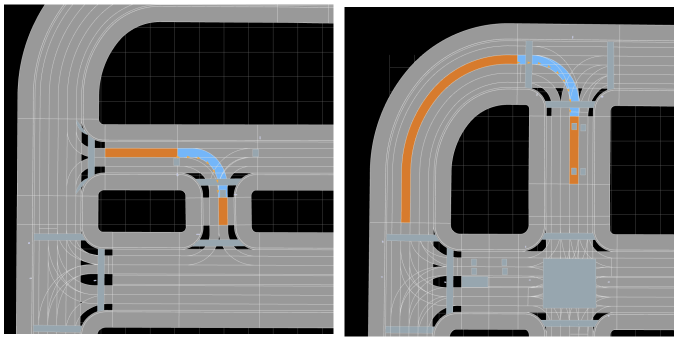

(a) Successor road is straight. (b) Successor road is left-curved.

Fig. 5: Two routes classified identically in existing work, but differently in our approach

classification of routes as identical overlooks the geographical features associated with roads connected to junctions.

1) Geographical Features: The geographical features, such as differences in curvature, elevation, speed limit, and the number of connected lanes or roads, are crucial factors contributing to the uniqueness of each ego-vehicle driving route. In line 15 of Algorithm 1, three analyses are conducted to classify routes based on lane characteristics ($curvature$, $elevation$, $speed$, and $interactability$). MAPSCE-GEN represents these geographical features in binary codes as shown in Table I. Each geographical feature is used to classify driving routes.

TABLE I: Binary code of each geographical feature. Numbers in parenthesis indicate binary code of CH_{speed}.

	00 (0)	01 (1)	10	11
CH_{curv}	STRAIGHT	LEFT	RIGHT	COMPLEX
CH_{elev}	FLAT	DOWNHILL	UPHILL	COMPLEX
CH_{speed}	NORMAL	HIGH	-	-

- *Curvature*

To analyze the curvature, CH_{curv}, MAPSCE-GEN collects the curvature ($curv$) of an "arc" in the "geometry" element found in the "planView" of a road from a high-definition map. In Eq. (1), we formulate the criteria for classifying CH_{curv} of lanes. Initially, the CH_{curv} of the road is set as STRAIGHT. This state is updated based on the value of $curv$: if $curv > 0.02$, indicating a left curvature, CH_{curv} is set to LEFT_CURV. However, if $curv < -0.02$, indicating a right curvature, CH_{curv} is set to RIGHT_CURV. If both left and right curvatures are detected, CH_{curv} is set to COMPLEX_CURV.

$$CH_{curv} = \begin{cases} 11 & \text{if } \exists curv > 0.02 \text{ and } \exists curv < -0.02 \\ 01 & \text{elif } \forall curv > 0.02 \\ 10 & \text{elif } \forall curv < -0.02 \\ 00 & \text{otherwise} \end{cases} \tag{1}$$

The curvature state CH_{curv} provides a simple categorization of the road's curvature into either straight (00), left curve (01), right curve (10), or complex curve (11).

- *Elevation*

To identify unevenness and vertical undulations, MAPSCE-GEN analyzes the road's elevation, denoted as CH_{elev}. This classification helps comprehend the necessary topographical features for testing. Following OpenDrive syntax, a, b, c, and d represent the elevation parameters of a road. A list of elevations, referred to as e_{list}, is formed by summing these parameters for each elevation element e_i in the elevation profile, calculated as $e_i = a + b + c + d$. Initially, the road's elevation state, represented by CH_{elev}, is designated as "FLAT". This designation evolves based on the values in e_{list}. If the maximum or minimum value in the list is non-zero and the difference z between the maximum and minimum values exceeds 3, CH_{elev} is updated as follows:

$$CH_{elev} = \begin{cases} 01 & \text{if } z > 3 \text{ and } e_{list} = \text{descending order} \\ 10 & \text{if } z > 3 \text{ and } e_{list} = \text{ascending order} \\ 11 & \text{if } z > 3 \text{ and } e_{list} = \text{unsorted} \\ 00 & \text{otherwise} \end{cases} \tag{2}$$

This elevation characteristic, CH_{elev}, provides a simple categorization of the road's elevation into either flat (00), downhill (01), uphill (10), or complex hill (11).

- *Speed Limit*

The speed of lane characteristics is denoted by s, representing the maximum speed limit for the road obtained from the road's "type" attribute. Initially, the speed state of the road, denoted by CH_{speed}, is set as NORMAL_SPEED. However, if s exceeds the threshold of 60, the speed state S is updated to HIGH_SPEED as follows:

$$CH_{speed} = \begin{cases} 0 & \text{if } s \geq 60 \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

This speed state S provides a categorization of the road's speed limit into either normal (0) or high (1). Using this feature, we can distinguish whether a lane is a highway or a regular road. Therefore, CH_{speed} characteristic does not require execution of the ego-vehicle.

- *Interactability*

Finally, MAPSCE-GEN calculates the $interactability$, $CH_{interact}$, to assess the complexity of the lane. In Eq. (4), we formulate the criteria for classifying $CH_{interact}$ of lanes. For a single road, MAPSCE-GEN determines the number of lanes N_l contained within it, while for a junction road, MAPSCE-GEN ascertains the number of connected roads N_c. This feature is crucial as it indicates the potential degree of lane-changing actions that a vehicle can undertake. Additionally, the number of lanes is significant as it also reflects the potential degree of lane-changing actions. Moreover, the number of connected roads aids in understanding the characteristics of the junction.

$$CH_{interact} = \begin{cases} N_c \leftarrow \text{\# of connected roads} & \text{if } lane \in \mathbb{J} \\ N_l \leftarrow \text{\# of lanes in road} & \text{otherwise} \end{cases} \tag{4}$$

For the interactability characteristic, 3 bits are allocated to represent the number of interactable roads or lanes, with a maximum value limited to 7.

2) Route Dictionary: The route dictionary categorizes all routes based on their geographical features. Each route generated in the high-definition map is assigned a corresponding Key_R. In Algorithm 2, the function CALCULATEROUTEKEY is used at line 17 to compute the route key, denoted as Key_R. This process involves converting various characteristics into binary codes, which are essential components in calculating the 24-bit binary route key. Key_R is structured so that each bit corresponds to specific information, as illustrated in Fig. 6.

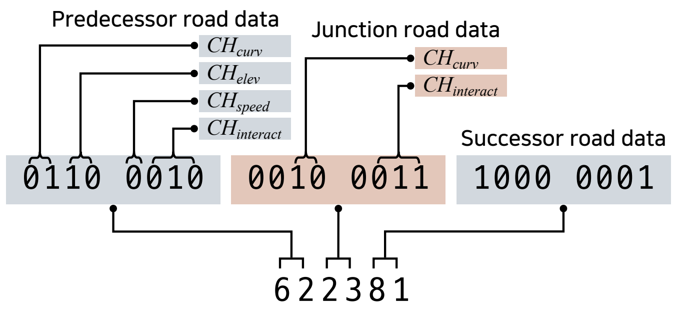


Fig. 6: Structure of a route key

As shown in Fig. 6, the first 8 bits and the last 8 bits collectively represent the geographical features of predecessor lanes and successor lanes, which are not considered in existing scenario generation approaches [32]–[36]. Meanwhile, the middle 8 bits capture the characteristics of the junction lane traversed by the ego-vehicle. In driving routes where multiple lanes are connected to one, individual features are combined using the logical OR operation. For example, the operation LEFT_CURV + RIGHT_CURV results in COMPLEX_CURV through the logical OR of "01" OR "10" = "11". Through this logical combination of various characteristics and geographical features when calculating the route key, every route aligns with its unique Key_R, which enables systematic and comprehensive classification of routes.

Once route keys are computed, MAPSCE-GEN classifies those routes generated in the route generation step into the corresponding route key. After this classification, the route dictionary contains a set of route keys and for each key, it contains a list of routes associated with the key as values.

D. Scenario Construction

In this step, MAPSCE-GEN first selects a route key from the route dictionary using a roulette wheel selection algorithm [51] to enhance the diversity of selected routes. The roulette wheel selection algorithm is implemented to prioritize routes with fewer occurrences in the route dictionary by assigning them higher priority. Therefore, this contributes to testing various geographical features. MAPSCE-GEN then retrieves the list of routes associated with the selected key.

After selecting an ego-vehicle's route, MAPSCE-GEN generates the object layer and environment layer to create complete and realistic scenarios. When MAPSCE-GEN generates NPC vehicles, their starting points are selected based on the coordinates around the selected route, and NPCs' driving is executed using CARLA's autopilot. The number of NPCs is determined by calculating the total road length of the selected route divided by 20. Then, the environment layer, encompassing weather conditions, road friction, and other environmental elements, is randomly generated.

V. EVALUATION

In this section, we utilize MAPSCE-GEN to generate geographically complex scenarios that cause safety violations due to geographical features. To evaluate the efficiency and effectiveness of MAPSCE-GEN, we investigate the following research questions.

- *RQ1: Can* MAPSCE-GEN *achieve full lane coverage on high-definition maps?*
- *RQ2: How effective is* MAPSCE-GEN *in exercising diverse geographical features?*
- *RQ3: How effective are the scenarios generated by* MAPSCE-GEN *in detecting safety violations caused by geographical features?*

A. Experimental Setup

We conduct the experiments on Ubuntu 20.04 with 32 GB memory, an Intel Core i9 CPU, and an NVIDIA RTX 3080 TI. We use CARLA simulator [46] version 0.9.13 to simulate autonomous driving scenarios and Autoware-Universe [45], representing a full-fledged Level 4 automation as the autonomous driving system. The system is integrated with the CARLA simulator, providing a robust platform for conducting comprehensive autonomous vehicle testing.

In the evaluation, we utilize 7 high-definition maps: 6 are defaults in the CARLA simulator [46], and 1 is provided as default in the LGSVL simulator [52]. For *RQ3*, we refrain from comparing MAPSCE-GEN against certain approaches. This decision stems from the fact that the research problem addressed by MAPSCE-GEN differs from that of existing approaches, rendering direct comparison impractical. For instance, studies such as [37]–[44] primarily focus on generating maps or road structures themselves. As such, there is no existing generation-based testing approach specifically targeting safety violations caused by geographical features, including [15]–[36].

Safety violations arising from geographical features can be broadly classified into two types: lane invasion and stuck. The lane invasion detector which is provided by CARLA [53] is set on the ego-vehicle to monitor safety violations of the autonomous driving system. We develop a stuck detector that detects situations where there is no change in the control values of the ego-vehicle for 5 minutes and no objects within a radius of 10 meters. Additionally, the driving scenario in

TABLE II: Results of RQ1: Lane coverage compared with a baseline called DLink that implements the route generation method used in existing work [34]–[36]

	DLink			MapSce-Gen		
HD-Map	Lane Cov.	# Miss. / # Total.	len(Miss.)	Lane Cov.	# Miss. / # Total.	len(Miss.)
San Francisco	100%	0 / 768	0m	100%	0 / 768	0m
Town01	90.32%	12 / 124	1364.91m	100%	0 / 124	0m
Town02	88.63%	10 / 88	483.78m	100%	0 / 88	0m
Town03	96.7%	13 / 396	108.02m	100%	0 / 396	0m
Town04	97.8%	10 / 454	573.67m	100%	0 / 454	0m
Town05	97.5%	12 / 486	1806.80m	100%	0 /486	0m
Town10HD	88.1%	20 / 168	713.54m	100%	0 / 168	0m
Total	96.9%	77 / 2484	5050.72m	100%	0 / 2484	0m

which a stuck situation occurred is executed once more to verify if it is consistently reproducible.

B. RQ1: Lane Coverage

In this section, we address *RQ1*, which investigates the effectiveness of our approach in covering lanes on the high-definition map. Higher lane coverage does not always lead to greater road diversity. However, achieving high lane coverage is essential for achieving diversity because if some lanes are not covered at all, we may lose a chance to exercise potential new geographical features that exist only in those uncovered lanes. This may result in reducing diversity.

We conduct experiments comparing MapSce-Gen with a baseline named DLink, in which we implement the route generation method used in existing work [34]–[36]. This existing route generation method selects a junction and connects only the roads directly linked to it.

The evaluation metric used for *RQ1* is Lane Coverage, calculated as the percentage of lanes covered by the generated routes out of the total number of lanes in the high-definition map. This metric offers a quantitative measure of how thoroughly each approach explores the lane network. The term # Miss. indicates the number of lanes not covered by the generated routes, highlighting instances where the method falls short in exploring specific lanes. The metric len(Miss.) represents the cumulative length of all the uncovered lanes, providing insights into the spatial extent of the missed coverage. This comparative analysis enables us to quantitatively measure and compare the effectiveness of MapSce-Gen against DLink.

The experimental results confirm MapSce-Gen's efficacy in achieving comprehensive lane coverage across high-definition maps (HD-Maps). Our method consistently covers all lanes in every evaluated scenario, demonstrating its capacity to generate routes traversing the entire lane network. To quantify this improvement, we compare our lane coverage with the baseline, DLink, as summarized in Table II. In Table II, the lengths of the missing lanes in scenarios where DLink falls short are highlighted. For instance, in Town05, DLink misses 12 lanes covering a total length of 1806.80m. In contrast, MapSce-Gen achieves complete coverage, eliminating the occurrence of missing lanes. Therefore, we experimentally demonstrate the ability to cover all lanes by applying route generation, ensuring coverage of every lane in the map.

Overall, the experimental results substantiate MapSce-Gen's effectiveness in covering the map, surpassing the capabilities of existing methods. The ability to cover approximately 5050.72m of previously inaccessible lanes demonstrates the significant improvement brought by MapSce-Gen. This enhancement is particularly evident in generating geographically complex scenarios, as can be seen in the following *RQ*s.

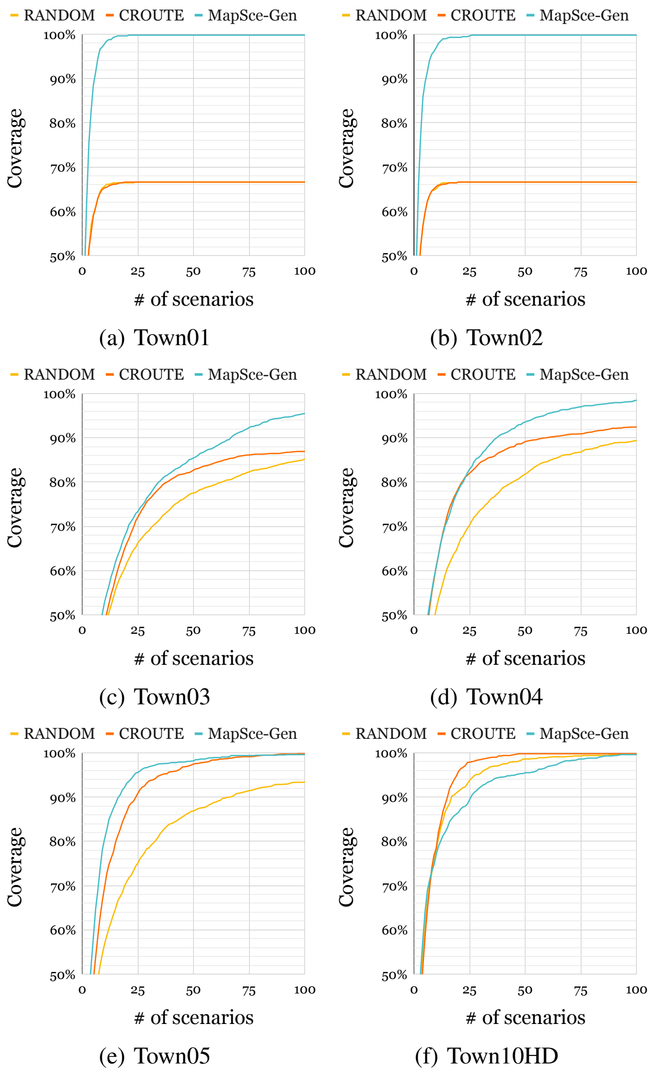
Answer to *RQ1*: MapSce-Gen achieves comprehensive coverage of all lanes, including approximately 5050.72m of previously inaccessible lanes.

C. RQ2: Road Characteristics Coverage

For *RQ2*, we undertake a systematic approach. We consecutively select 100 driving scenarios and compute the coverage increase rate. To ensure statistical robustness, we repeat this process 100 times, calculating the average coverage increase rate. This iterative methodology ensures a more precise evaluation of how effectively each scenario generation and selection method covers diverse road characteristics across the map.

In this experiment, we compare MapSce-Gen's effectiveness in generating geographically complex scenarios with two baselines: RANDOM and CROUTE [36]. RANDOM and CROUTE utilize a DLink method for route generation. Regarding scenario selection, RANDOM randomly selects one scenario from all possibilities, while CROUTE chooses from classes classified solely based on junction lane characteristics. The evaluation utilizes the Road Characteristics Coverage metric, which measures the coverage of diverse road characteristics across the map. This metric indicates the percentage of covered characteristics among all lanes in the map.

The Road Characteristics Coverage graphs for Town01 and Town02, depicted in Fig. 7a and Fig. 7b respectively, illustrate the effectiveness of scenario generation methods in covering diverse road characteristics. These maps, with relatively fewer diverse road characteristics, require a lower number of driving scenarios to achieve 100% coverage. However, existing methods such as RANDOM and CROUTE fail to



(a) Town01 (b) Town02

(c) Town03 (d) Town04

(e) Town05 (f) Town10HD

Fig. 7: Results of RQ2: road characteristics coverage

reach full coverage due to their inability to cover certain road characteristics. In contrast, MapSce-Gen quickly achieves complete coverage in both Town01 and Town02.

Town03 and Town04, characterized by a wide variety of road characteristics, serve as ideal test cases for evaluating the effectiveness of selecting geographically diverse lanes. As shown in Fig. 7c, Fig. 7d, and Fig. 7e, MapSce-Gen demonstrates superior exploration of diverse road characteristics compared to RANDOM and CROUTE. The experiments highlight MapSce-Gen's ability to outperform in rapidly exploring road characteristics on these maps, showcasing its effectiveness in selecting geographically diverse lanes.

In Town10HD, depicted in Fig. 7f, all three methods achieve coverage close to 100%. However, MapSce-Gen records over 90% coverage later than the other methods. This delay is attributed to the map's lack of diverse geographical features, which makes it challenging to find driving routes similar to those already identified.

TABLE III: Results of RQ3: safety violations detected by MapSce-Gen. Newly-covered lanes refer to those that cannot be covered by existing work [34]–[36]. Previously-covered lanes refer to those covered by existing work as well as MapSce-Gen.

	Safety Violations by Geographical Features				
	In Newly-Covered Lanes		In Previously-Covered Lanes		Safe Scenarios
	Lane-Invasion	Stuck	Lane-Invasion	Stuck	
Number	17	22	6	10	89
Rate	11.81%	15.28%	4.17%	6.94%	61.81%

In conclusion, the investigation into *RQ2* provides valuable insights into the effectiveness of selecting geographically diverse lanes. Through a comprehensive comparison of three driving scenario generation and selection methods—RANDOM, CROUTE, and MapSce-Gen —using the Road Characteristics Coverage metric, we assess their ability to cover diverse road characteristics across various town scenarios. The visual representation in Fig. 7a and Fig. 7b highlights the challenges faced by RANDOM and CROUTE in achieving 100% coverage in Town01 and Town02, which have relatively fewer diverse characteristics. The results for Town03, Town04, and Town05 underscore the effectiveness of MapSce-Gen in rapidly exploring diverse road characteristics compared to the other methods, providing valuable insights into the selection of geographically diverse lanes for driving scenario generation.

Answer to *RQ2*: MapSce-Gen efficiently explores diverse geographical features across various high-definition maps and rapidly covers road characteristics.

D. RQ3: Detection of Safety Violations

We address *RQ3* by evaluating MapSce-Gen's effectiveness in detecting safety violations caused by geographical features. Specifically, we categorize the detected violations into two groups: 1) violations caused by geographical features in the lanes uniquely covered by MapSce-Gen, and 2) violations caused by geographical features in the lanes covered by existing work as well. We denote the lanes uniquely covered in MapSce-Gen as Newly-Covered Lanes and those already covered in existing work [34]–[36] as Previously-Covered Lanes. For *RQ3*, we ran MapSce-Gen for 12 hours. All violations detected in *RQ3* occur due to geographical features such as highway exits, uphill and downhill slopes, and curved roads. We excluded from the *RQ3* results any safety violations caused by NPCs or other factors.

Table III presents an evaluation of safety violations caused by geographical features in both Newly-Covered Lanes (previously uncovered) and Previously-Covered Lanes (already covered in existing work). The majority of scenarios fall under safe categories, with no safety violations occurring (61.64%). Lane-invasion and stuck violations are significantly higher in newly covered lanes compared to previously covered lanes. This indicates that safety violations are more prevalent in the lanes uniquely covered by our approach, which considers diverse geographical features.

The result of the violation occurrences in Newly-Covered Lanes compared to Previously-Covered Lanes reveals substantial differences. Lane-Invasion Violations are almost three times more frequent in Newly-Covered Lanes, with an increase factor of approximately 2.99 ($\textit{increase factor} = \frac{12.33\%}{4.11\%} \approx 2.99$). Similarly, Stuck Scenarios occur about 2.20 times more often in Newly-Covered Lanes compared to Previously-Covered Lanes ($\textit{increase factor} = \frac{15.07\%}{6.85\%} \approx 2.20$). In both cases, the increase factor indicates how many times more frequently violations occur in Newly-Covered Lanes compared to Previously-Covered Lanes. This suggests that violations are significantly more common in the Newly-Covered Lanes.

Combining the results of both types of violations (Lane-Invasion and Stuck), Newly-Covered Lanes exhibit an overall violation rate approximately 2.63 times higher than in Previously-Covered Lanes. This comprehensive analysis highlights the effectiveness of MAPSCE-GEN in identifying and addressing safety violations caused by geographical features.

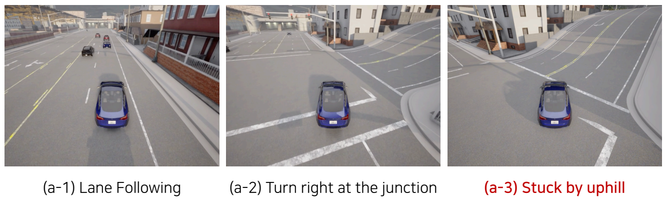
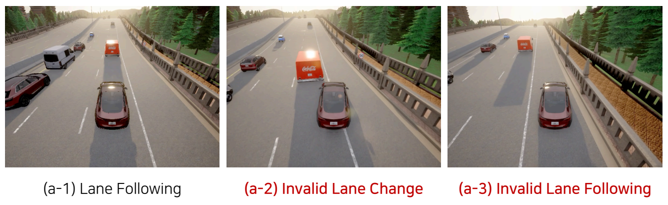


Fig. 8: Stuck caused by CH_{curv} and CH_{elev}

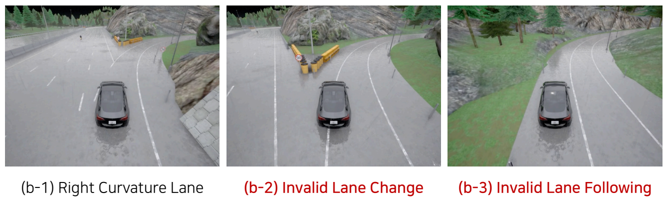
Fig. 8 and Fig. 9 illustrate examples of safety violations detected by MAPSCE-GEN. Fig. 8 presents a scenario where the vehicle attempts a right turn at a junction and then gets stuck upon encountering an uphill slope. This scenario is challenging to discover using traditional generation techniques that do not consider vertical geographical features of junctions and connected roads because safety violations do not occur when encountering the uphill slope on a straight drive.

Fig. 9a illustrates a safety violation due to the speed limit geographical feature (i.e., CH_{speed} defined in Section IV-C). Lane-invasion is critical not only because it violates traffic regulations but also because it can lead to collision situations or pose risks by interfering with the driving of other vehicles. Furthermore, it is particularly significant as it represents a safety violation occurring in a lane not previously covered in existing work, highlighting the importance of addressing such scenarios. The scenario depicted in Fig. 9b, where lane invasion occurs while transitioning to another road, represents a very challenging driving scenario to detect. The reason is that there are very few classified driving routes falling into this category. Among all the driving routes analyzed from the 7 high-definition maps, there are only 2 routes.

These examples demonstrate the capability of our approach to uncover scenarios that are both geographically complex and



(a) Lane-invasion caused by CH_{speed}


(b-1) Right Curvature Lane
(b-2) Invalid Lane Change
(b-3) Invalid Lane Following

(b) Lane-invasion caused by CH_{speed} and CH_{curv}

Fig. 9: Invasions of solid lanes caused by geographical features

present safety-violating situations. Each example highlights specific characteristics or combinations of road features that lead to unconventional and challenging driving scenarios. Importantly, all these driving scenarios depict instances where safety violations occur due to the geographical features that can lead to hazardous situations.

Answer to *RQ3*: MAPSCE-GEN effectively detects safety violations caused by geographical features. Newly covered lanes exhibit a violation rate 2.63 times higher than that of previously covered lanes.

VI. RELATED WORK

Random-based Scenario Generation. ScenoRITA [32] and DoppelTest [33] use random-based generation to construct driving scenarios. When generating ego-vehicle routes, two valid points on the entire map are randomly selected to set as the starting and ending points for the ego-vehicle. However, it is necessary to perturb these scenarios into meaningful driving scenarios through mutation because the generated scenarios did not consider the geographical features. Furthermore, random-based generation tests autonomous driving systems without precise knowledge of the driving routes.

Model-based Scenario Generation. ComOpT [34], ATLAS [35], and CROUTE [36] utilize predefined models for generating driving routes through model-based generation. In these approaches, the basic unit for generating an ego-vehicle's route is a path that passes through only one junction. Consequently, scenarios generated using these methods often overlook roads that possess unique geographical features. These methodologies [34]–[36] also classify scenarios based on specific criteria, prioritizing the execution of a diverse set of routes during each generation and evaluation, which is more effective for testing than running similar routes. For example, ComOpT [34] checks scenarios against 9 predefined junction classes based on factors like the number of connected roads and their angles, categorizing them accordingly. On the other hand, ATLAS [35] categorizes junction lanes, identifying 13 junction lanes based on lane count and junction vector. CROUTE [36] further refines this by classifying features into 29 detailed junction types, contributing to generating various routes for ego-vehicles. However, these methods mainly focus on the diversity of traversed junctions, posing a challenge in categorizing based on the starting and destination roads' characteristics. In contrast, MapSce-Gen enables detailed classification by considering not only junctions but also connected lanes' geographical features. Unlike existing work that analyzes only a few predefined types of junction characteristics, MapSce-Gen extracts geographical features from the entire map and represents them in binary codes, allowing for classification into a wider range of characteristics.

Road Structure Generation. AsFault [37] first proposed a method for generating road structures. Additionally, in the SBFT (Search-Based and Fuzz Testing) tool competition (formerly SBST) [40]–[42], several studies have focused on autonomous driving tests that generate road structures. Most prior work [37]–[39], [44] constructs only curved road structures without considering vertical features such as elevation and number of connected lanes. EvoScenario [43] defines and generates complex lane structures such as contract, expand, merge, and split. Our work achieves a different goal from existing works that generate road structures [37]–[44]. These works generate maps with newly created road structures in small units. In contrast, our work uses an existing map to thoroughly explore geographical features across the entire map. This offers a practical way to test geographical features, especially when using real-world maps converted into simulation-compatible. Additionally, our work complements the road structure generation works by applying our approach to the maps they produce.

VII. Threats to Validity

The threats to internal validity may arise from numerous parameters inherent in simulation testing. While the safety violations identified by our approach appear to be caused by geographical features, there remains a possibility of safety violations resulting from other factors. To address this concern, we configure the object layer to ensure that other objects perform only legal actions. Additionally, conservative conditions are integrated into the lane detector and stuck detector to exclude cases potentially caused by other elements.

The threats to external validity is that we apply MapSce-Gen to single autonomous driving system, Autoware-Universe [45]. While Autoware-Universe is widely utilized as an open-source autonomous driving software, it is important to note that it is research-grade. software. In the future, we plan to extend our experiments to include production-grade autonomous driving software, such as Baidu Apollo [54].

VIII. Conclusion and Future Work

In this paper, we propose MapSce-Gen, a model-based scenario generation approach for autonomous driving systems, that targets safety violations caused by geographical features. MapSce-Gen introduces route generation and route classification to generate geographically complex driving scenarios. Our evaluation results show that MapSce-Gen effectively identifies safety violations arising from diverse geographical features, demonstrating its capability to enhance the safety and reliability of autonomous driving systems.

The future work involves integrating our generated driving scenarios into mutation-based testing as seed scenarios. This integration can enhance the diversity and scope of test inputs by introducing realistic and geographically complex scenarios, and allow for a more comprehensive evaluation across challenging situations. Furthermore, applying our scenarios in mutation-based testing may uncover novel corner cases and safety-critical scenarios. This will contribute to a more robust safety assessment by revealing unforeseen interactions in the autonomous driving system's behaviors.

IX. Acknowledgements

We thank the anonymous reviewers for their constructive feedback. This work was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (2022R1A2C10911913 and 2021R1G1A109355411).

References

[1] J. Bote, "Cruise vehicle gets stuck in wet concrete while driving in san francisco," *SFGATE*, 2023, [Online].
[2] J. Valinsky, "Complete meltdown: Driverless cars in san francisco stall causing a traffic jam," *CNN Business*, 2023, [Online].
[3] J. Mulach, "Autonomous cars cause gridlock in yet another us city," *DRIVE*, 2023, [Online].
[4] B. Yu, "Robotaxis halt traffic in san francisco's north beach day after expansion approval," *CBS News*, 2023, [Online].
[5] K. Truong, "Sf officials describe chaos from cruise, waymo cars as they try to slow their rollout," *The San Francisco Standard Business*, 2023.
[6] C. News, "Driverless robotaxi crashes with fire truck in san francisco; passenger injured," *CBS News*, 2023, [Online].
[7] A. Rose, "A woman was found trapped under a driverless car. it wasn't the first car to hit her," *CNN Business*, 2023, [Online].
[8] A. R. C. Bieber, "93% have concerns about self-driving cars according to new forbes legal survey," *Forbes Advisor Legal*, 2024, [Online].
[9] C. Spondent, "Report: self-driving car 'unable to navigate birmingham's spaghetti junction'," *BBC Top Gear*, 2023, [Online].
[10] H. Wen, Z. Ma, Z. Chen, and C. Luo, "Analyzing the impact of curve and slope on multi-vehicle truck crash severity on mountainous freeways," *Accident Analysis & Prevention*, vol. 181, p. 106951, 2023.
[11] R. Fu, Y. Guo, W. Yuan, H. Feng, and Y. Ma, "The correlation between gradients of descending roads and accident rates," *Safety science*, vol. 49, no. 3, pp. 416–423, 2011.
[12] G. Cheng, R. Cheng, Y. Pei, and L. Xu, "Probability of roadside accidents for curved sections on highways," *Mathematical Problems in Engineering*, vol. 2020, pp. 1–18, 2020.
[13] K. Czarnecki, "Operational world model ontology for automated driving systems–part 1: Road structure," *Waterloo Intelligent Systems Engineering Lab (WISE) Report, University of Waterloo*, 2018.
[14] 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.

[15] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing advanced driver assistance systems using multi-objective search and neural networks," in *Proceedings of the 31st IEEE/ACM international conference on automated software engineering*, 2016, pp. 63–74.

[16] R. B. Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, "Testing autonomous cars for feature interaction failures using many-objective search," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 143–154.

[17] J. Zhou and L. del Re, "Safety verification of adas by collision-free boundary searching of a parameterized catalog," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 4790–4795.

[18] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek, "Generating adversarial driving scenarios in high-fidelity simulators," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8271–8277.

[19] J. Norden, M. O'Kelly, and A. Sinha, "Efficient black-box assessment of autonomous vehicle safety," in *NeurIPS 2019 Workshop on Machine Learning for Autonomous Driving*, 2019.

[20] Z. Ghodsi, S. K. S. Hari, I. Frosio, T. Tsai, A. Troccoli, S. W. Keckler, S. Garg, and A. Anandkumar, "Generating and characterizing scenarios for safety testing of autonomous vehicles," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 157–164.

[21] Y. Luo, X.-Y. Zhang, P. Arcaini, Z. Jin, H. Zhao, F. Ishikawa, R. Wu, and T. Xie, "Targeting requirements violations of autonomous driving systems by dynamic evolutionary search," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 279–291.

[22] G. Li, Y. Li, S. Jha, T. Tsai, M. Sullivan, S. K. S. Hari, Z. Kalbarczyk, and R. Iyer, "Av-fuzzer: Finding safety violations in autonomous driving systems," in *2020 IEEE 31st international symposium on software reliability engineering (ISSRE)*. IEEE, 2020, pp. 25–36.

[23] A. Calò, P. Arcaini, S. Ali, F. Hauer, and F. Ishikawa, "Generating avoidable collision scenarios for testing autonomous driving systems," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*. IEEE, 2020, pp. 375–386.

[24] S. Kim, M. Liu, J. J. Rhee, Y. Jeon, Y. Kwon, and C. H. Kim, "Drivefuzz: Discovering autonomous driving bugs through driving quality-guided fuzzing," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1753–1767.

[25] Z. Zhong, G. Kaiser, and B. Ray, "Neural network guided evolutionary fuzzing for finding traffic violations of autonomous vehicles," *IEEE Transactions on Software Engineering*, 2022.

[26] Z. Wan, J. Shen, J. Chuang, X. Xia, J. Garcia, J. Ma, and Q. A. Chen, "Too afraid to drive: Systematic discovery of semantic dos vulnerability in autonomous driving planning under physical-world attacks," *ISOC Network and Distributed Systems Security (NDSS) Symposium*, 2022.

[27] 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.

[28] Y. Sun, C. M. Poskitt, J. Sun, Y. Chen, and Z. Yang, "Lawbreaker: An approach for specifying traffic laws and fuzzing autonomous vehicles," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–12.

[29] M. Cheng, Y. Zhou, and X. Xie, "Behavexplor: Behavior diversity guided testing for autonomous driving systems," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2023, pp. 488–500.

[30] Z. Hu, S. Guo, Z. Zhong, and K. Li, "Coverage-based scene fuzzing for virtual autonomous driving testing," *CoRR*, vol. abs/2106.00873, 2021. [Online]. Available: https://arxiv.org/abs/2106.00873

[31] J. Zhou, S. Tang, Y. Guo, Y.-F. Li, and Y. Xue, "From collision to verdict: Responsibility attribution for autonomous driving systems testing," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023, pp. 321–332.

[32] Y. Huai, S. Almanee, Y. Chen, X. Wu, Q. A. Chen, and J. Garcia, "sceno rita: Generating diverse, fully-mutable, test scenarios for autonomous vehicle planning," *IEEE Transactions on Software Engineering*, 2023.

[33] Y. Huai, Y. Chen, S. Almanee, T. Ngo, X. Liao, Z. Wan, Q. A. Chen, and J. Garcia, "Doppelgänger test generation for revealing bugs in autonomous driving software," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 2591–2603.

[34] C. Li, C.-H. Cheng, T. Sun, Y. Chen, and R. Yan, "Comopt: Combination and optimization for testing autonomous driving systems," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7738–7744.

[35] Y. Tang, Y. Zhou, T. Zhang, F. Wu, Y. Liu, and G. Wang, "Systematic testing of autonomous driving systems using map topology-based scenario classification," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021, pp. 1342–1346.

[36] Y. Tang, Y. Zhou, F. Wu, Y. Liu, J. Sun, W. Huang, and G. Wang, "Route coverage testing for autonomous vehicles via map modeling," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 450–11 456.

[37] A. Gambi, M. Mueller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019, pp. 318–328.

[38] E. Castellano, A. Cetinkaya, and P. Arcaini, "Analysis of road representations in search-based testing of autonomous driving systems," in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2021, pp. 167–178.

[39] 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.

[40] S. Panichella, A. Gambi, F. Zampetti, and V. Riccio, "Sbst tool competition 2021," in *2021 IEEE/ACM 14th International Workshop on Search-Based Software Testing (SBST)*. IEEE, 2021, pp. 20–27.

[41] A. Gambi, G. Jahangirova, V. Riccio, and F. Zampetti, "Sbst tool competition 2022," in *Proceedings of the 15th Workshop on Search-Based Software Testing*, 2022, pp. 25–32.

[42] M. Biagiola, S. Klikovits, J. Peltomäki, and V. Riccio, "Sbft tool competition 2023-cyber-physical systems track," in *2023 IEEE/ACM International Workshop on Search-Based and Fuzz Testing (SBFT)*. IEEE, 2023, pp. 45–48.

[43] S. Tang, Z. Zhang, J. Zhou, Y. Zhou, Y.-F. Li, and Y. Xue, "Evoscenario: Integrating road structures into critical scenario generation for autonomous driving system testing," in *IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, 2023, pp. 309–320.

[44] Z. Bao, S. Hossain, H. Lang, and X. Lin, "A review of high-definition map creation methods for autonomous driving," *Engineering Applications of Artificial Intelligence*, vol. 122, p. 106125, 2023.

[45] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 287–296.

[46] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

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

[48] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive computing*, vol. 7, no. 4, pp. 12–18, 2008.

[49] ASAM, "Asam opendrive," *Association for Standardization of Automation and Measuring Systems*, 2023, [Online].

[50] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr, "Lanelet2: A high-definition map framework for the future of automated driving," in *2018 21st international conference on intelligent transportation systems (ITSC)*. IEEE, 2018, pp. 1672–1679.

[51] D. E. Goldberg, *Genetic algorithms*. pearson education India, 2013.

[52] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta *et al.*, "Lgsvl simulator: A high fidelity simulator for autonomous driving," in *2020 IEEE 23rd International conference on intelligent transportation systems (ITSC)*. IEEE, 2020, pp. 1–6.

[53] Carla, "Lane detector," *Carla Simulator*, 2021, [Online].

[54] Baidu, *Apollo: Open source autonomous driving*, [Online].