TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 07/38 pp1440–1459
DOI: 10.26599/TST.2025.9010178
Volume 31, Number 3, June 2026

Research Article

Realistic Corner Case Generation for Autonomous Vehicles with Multimodal Large Language Model

Qiujing Lu, Meng Ma, Zheng Wang, Wei Lu, Mingyue Ma, Guangming Zhao*, and Shuo Feng*

Abstract: To guarantee the safety and reliability of Autonomous Vehicle (AV) systems, corner cases play a crucial role in exploring the system's behavior under rare and challenging conditions within simulation environments. However, current approaches often fall short in meeting diverse testing needs and struggle to generalize to novel, high-risk scenarios that closely mirror real-world conditions. To tackle this challenge, we present AutoScenario, a multimodal Large Language Model (LLM)-based framework for realistic corner case generation. It converts safety-critical real-world data from multiple sources into textual representations, enabling the generalization of key risk factors while leveraging the extensive world knowledge and advanced reasoning capabilities of LLMs. Furthermore, it integrates tools from the Simulation of Urban Mobility (SUMO) and Car Learning to Act (CARLA) simulators to automatically interpret and execute the scenario code by LLMs. Our experiments demonstrate that AutoScenario can generate realistic and challenging test scenarios, precisely tailored to specific testing requirements or textual descriptions. Additionally, we validated its ability to produce diverse and novel scenarios derived from multimodal real-world data involving risky situations, harnessing the powerful generalization capabilities of LLMs to effectively simulate a wide range of corner cases. The implementation is available at https://github.com/THU-AI-Testing/AutoScenario.

Key words: autonomous driving safety testing; corner case generation; vision-language models; automated scenario generation

1 Introduction

Currently, the safety of Autonomous Vehicles (AVs) remains a critical barrier to their widespread deployment on public roads. Discovering and testing corner cases in advance helps secure AV safety and accelerate development cycles. However, as AV performance improves, further advancements become increasingly difficult. This is due to corner cases emerging less frequently and exhibiting greater diversity[1]. As a result, defining and identifying the most relevant corner cases has become increasingly critical to achieving further performance gains.

Significant efforts have been devoted to this area. For

• Qiujing Lu and Shuo Feng are with Department of Automation, Tsinghua University, Beijing 100084, China. E-mail: qiujinglu@mail.tsinghua.edu.cn; fshuo@tsinghua.edu.cn.
• Meng Ma is with Department of Civil and Environmental Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180, USA. E-mail: mam7@rpi.edu.
• Zheng Wang and Wei Lu are with Beijing DiDi Infinity Technology and Development Co., Ltd., Beijing 100095, China. E-mail: irwintiger@hotmail.com; luweift@gmail.com.
• Mingyue Ma and Guangming Zhao are with Research Institute for Road Safety of the Ministry of Public Security, Beijing 100062, China. E-mail: mmy1986@foxmail.com; zgmbright@163.com.
* To whom correspondence should be addressed.
Manuscript received: 2025-09-26; revised: 2025-11-17; accepted: 2025-11-21

© The author(s) 2026. The articles published in this open access journal are distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/).

instance, Ref. [2] carefully mined corner cases from large-scale autonomous driving datasets[3–5]. However, this approach is limited by its reliance on real-world driving data collected from AVs, which is costly and restricted in scope. Additionally, replaying pre-collected data lacks dynamic interaction with the AV under test, reducing its effectiveness. On the other hand, various methods have been explored to synthesize safety-critical scenarios. These include rule-based[6, 7] and data-driven techniques[8–10]. However, these methods often suffer from limited diversity due to their dependence on initial conditions from given scenes. Furthermore, scenarios generated through predefined rules or adversarial learning may lack realism, as the applied perturbations can deviate from plausible real-world behaviors, thus diminishing the effectiveness of the testing process.

Meanwhile, there has been limited progress in developing effective control mechanisms for flexible scenario generation driven by abstract requirements. This is crucial as developers often conceptualize scenarios broadly, whereas simulations require detailed configurations, such as road geometry and precise vehicle placements. A mechanism that enables developers to control scenario generation through language descriptions provides a natural solution to bridge this gap, making scenario-based testing more practical and accelerating performance evaluation of AVs.

However, building a text-conditioned generation mechanism is inherently challenging. It requires modeling everything from static environment elements to agent behaviors, while accurately translating narrative language to detailed, structured configurations. The rise of Large Language Models (LLMs) and Vision-Language Models (VLMs), trained on vast internet data, presents a promising direction. These models have demonstrated exceptional capabilities in learning, reasoning, and solving complex problems. Their applications span fields like medicine, education, finance, and engineering[11–14], demonstrating substantial advancements and broad generalization capabilities.

Driven by these advancements and the need for realistic and diverse safety- critical scenarios, we developed AutoScenario, a fully automated pipeline with high controllability as shown in Fig. 1. It generates realistic and diverse scenarios containing main components that closely mimic real-world environments through prompt engineering and the integration of tools from Simulation of Urban Mobility (SUMO), an open-source traffic simulation package[15], Car Learning to Act (CARLA)[16], an open-source simulator powered by Unreal Engine[17] with high-fidelity digital assets, and data-driven deep learning models.

Our contribution can be concluded as follows:

- We propose AutoScenario, a framework that automates the safety-critical scenario generation pipeline while providing a high degree of controllability.
- We extract key risk factors from multimodal real-world corner cases to improve the realism and diversity of generated scenarios.
- We utilize large language model to generalize scenarios through reasoning and open-world knowledge.
- We integrate simulation tools and data-driven models to inject domain knowledge, ensuring physical consistency, behavioral plausibility, and stable performance during scenario generation.

2 Related Work

2.1 Safety-critical driving scenario generation

The widespread deployment of AVs continues to be impeded by safety concerns. Significant efforts have been devoted to identifying and mitigating unsafe components through testing[18–20]. Among these, scenario-based testing has demonstrated its potential for evaluating AV performance under rare but critical corner-case conditions[21]. Nevertheless, the generation of realistic and plausible corner cases remains a substantial challenge, primarily due to the inherent complexity of physical environments and traffic conditions encountered in the real world. One approach to improving realism is replaying driving data collected from real-world scenarios. However, such method falls short in creating realistic interaction with AV being tested, limiting their effectiveness in evaluating how the AV responds to unpredictable or adversarial agents during testing.

Significant efforts have been made in creating challenging corner cases, broadly classified into two main approaches: data-driven generation and knowledge-based generation. Data-driven models leverage rich information from real-world driving datasets to enhance the realism and diversity of

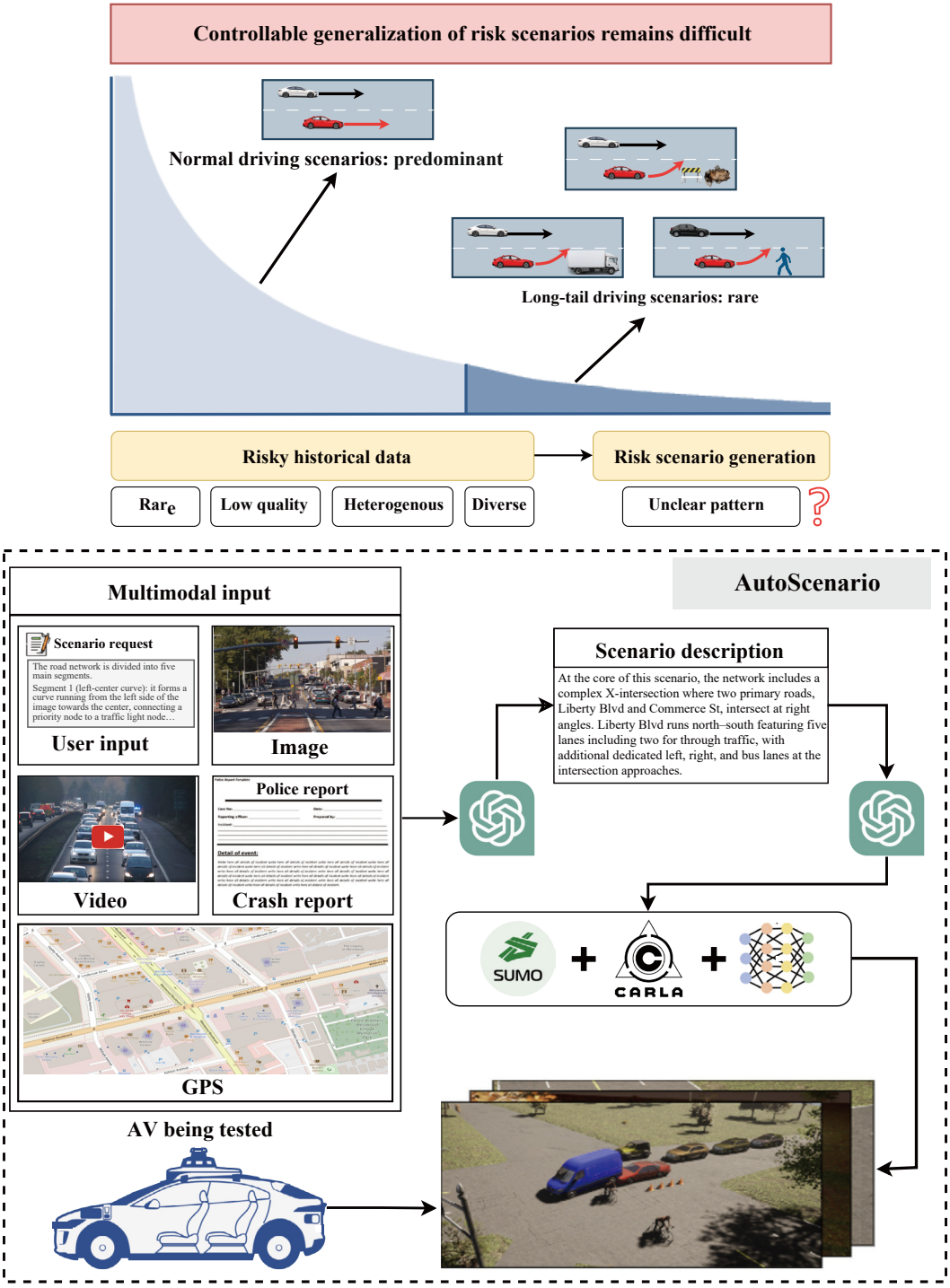


Fig. 1 AutoScenario: an LLM based framework for automated generation of realistic corner cases. GPS: Global Positioning System.

generated scenarios[22–24]. For instance, Ref. [25] employed a Transformer-based network with safety mapping to generate realistic agent behaviors, achieving distribution-level alignment with real-world data. Reference [10] learnt a graph-based conditional Variational AutoEncoder (VAE) as traffic prior, optimizing each agent's behavior to provoke collisions with a rule-based AV planner. Reference [26] used an encoder-decoder architecture and retrieval-based in-context learning to synthesize realistic traffic scenarios. However, these methods are limited to generating scenarios derived from existing datasets, lacking the

capability to produce controllable, specialized scenarios tailored to specific testing objectives.

Knowledge-based generation represents another promising direction, where external knowledge is explicitly incorporated into the scenario generation process. Reference [27] utilized an evolutionary method to minimize the drivable area by strategically manipulating surrounding vehicles. Reference [28] emphasized the importance of diversity and high driving skills within scenarios, proposing a distinct policy set selector within their reinforcement learning method to balance these two aspects. Reference [29] proposed Semantically Adversarial Generation (SAG), which leverages domain knowledge encoded as first-order logic in a hierarchical tree structure to guide scenario creation. However, these knowledge-based methods generally suffer from limited realism, limiting their applicability to high-fidelity simulation or real-world testing.

2.2 Scenario generation with LLMs

Multimodal LLMs have seen extensive application in autonomous driving systems since their inception[30–33]. Specifically, LLMs are increasingly employed for generating diverse and realistic scenarios[34–36], which are crucial for testing and evaluating AVs. For instance, ChatScene uses LLMs to generate scenarios from a pre-existing scenic library[37–38], while ChatSim produces photo-realistic Three-Dimentional (3D) driving simulations using external digital assets[39]. However, both methods face limitations in scene diversity: ChatScene is constrained by its fixed library, and ChatSim can only modify existing scenes rather than generate entirely new ones from scratch.

LLMs are also being utilized for the generation of corner cases. CRITICAL[40] uses LLMs to refine critical cases by updating scenario configurations for AV training. However, its evaluation is confined to freeway scenarios[41], restricting its generalizability across different traffic conditions and road layouts. Reference [42] incorporated an LLM-enhanced adaptive genetic algorithm to search for safety-critical scenarios. However, the assumption that Background Vehicles (BVs) and pedestrians strictly adhere to traffic laws may overlook safety-critical situations resulting from anomalous participant behaviors. Reference [43] leveraged an LLM to transform user queries into a loss function, guiding a diffusion model to generate query-compliant trajectories. However, it is limited to manipulating agent behaviors within a predefined road map with specific initial conditions. Reference [34] employed LLMs to generate brief agent trajectories based on minimal scenario descriptions, thereby facilitating scenario engineering. Nevertheless, it remains restricted to highway scenarios and requires further exploration in more complex environments.

Ensuring safety through scenario-based testing remains one of the most pressing challenges in autonomous driving. Effective validation demands a large collection of diverse and highly controllable safety-critical scenarios. However, existing methods still struggle to generate such scenarios efficiently and at scale. Our work aims to leverage LLMs to facilitate the generation of diverse safety-critical scenarios in a controllable and efficient manner through multiple input modalities, including text, images, and videos.

3 Method

In this section, we present the core methodology of AutoScenario, a multimodal, LLM-driven framework designed to convert real-world multimodal inputs into safety-critical corner-case scenarios. We first introduce the problem formulation of the automated scenario generation system. Next, we describe the overall pipeline of the framework, followed by the prompting techniques that support its key components. We then detail the design of the generalized interpreter, which serves as the central module for the encoding process. Finally, we discuss the framework's integration with domain-specific simulation tools, which form the foundation of the decoding process.

3.1 Problem statement

We model the entire process as an encoding-decoding framework, where multimodal real-world scenarios are first encoded into a universal, interpretable language space. It is then decoded into executable scenarios for AV testing through a set of LLM-guided generators coupled with simulation tools.

Let S_E represent a real-world scenario encompassing road topology, diverse road users, static objects, and weather conditions. We construct a consistent linguistic description, $E_l = \{E_{road}, E_{objects}, E_{agents}, E_{weather}\}$, where each component describes a semantic aspect of the scene. The encoding process is implemented via large language models, can be expressed as $E_l = f(S_E)$, where $f(\cdot)$ maps multimodal environment states to structured textual descriptions.

In the decoding process, we generate the final scenario through a sequence of specialized generators powered by LLM and simulation tools. Given the encoded description E_l and auxiliary domain knowledge k, the framework first employs a network generator, $n \sim \text{NetGenerator}(E_l, k)$, to produce feasible lane and topology configurations. Subsequently, an object generator, $o \sim \text{ObjectGenerator}(E_l, n, k)$, positions static objects within the environment, while the agent generator, $v \sim \text{AgentGenerator}(E_l, n, k)$, initializes dynamic agents and synthesizes their behaviors consistent with the described context. Finally, a scenario generator, $s \sim \text{ScenarioGenerator}(E_l, n, v, o, k)$, integrates all generated components into a coherent, simulation-ready scene.

To ensure semantic alignment between the generated scenario and the original description, we formulate the decoding objective as Formula (1),

$$\begin{aligned} \min_{k} \quad & \text{dist}(\text{Emb}(E_l), \text{Emb}(f(s))), \\ \text{s.t.,} \quad & E_l = f(S_E), \\ & n \sim \text{NetGenerator}(E_l, k), \\ & v \sim \text{AgentGenerator}(E_l, n, k), \\ & o \sim \text{ObjectGenerator}(E_l, n, k), \\ & s \sim \text{ScenarioGenerator}(E_l, n, v, o, k) \end{aligned} \tag{1}$$

where $\sim$ denotes sampling from the distribution, $\text{Emb}(\cdot)$ denotes the embedding function that maps linguistic descriptions to a high-dimensional semantic space, and $\text{dist}(\cdot,\cdot)$ measures semantic discrepancy, implemented as the cosine distance between embeddings obtained from the text-embedding-ada-002 model[44]. Domain knowledge k is dynamically injected through prompting strategies to minimize the semantic gap between the source and reconstructed descriptions. In practice, a balance between fine-grained descriptive detail and generalizability of risk factors is maintained to ensure high fidelity and robust scenario generation.

3.2 Pipeline

The generation framework is illustrated in Fig. 2. The pipeline accepts multimodal inputs, which includes but not limited to the following: user request expressed in the text, image taken from random viewpoints, and videos from a driving vehicle's perspective. They are sent through specially designed interpreter powered by LLM to generate standard scenario descriptions E_l, which extracts key components from the input while adding diverse details. See more details in Section 3.4. This is followed by scenario generation conditioned on E_l, including constructing road structure, placing static objects, initializing and dynamically generating agents. See Section 3.5 for more details.

Multimodal inputs are preprocessed with tailored attention mechanisms to generate consistent descriptions from the provided real-world data. For short user requests, we expand them into richer, more detailed scenario descriptions. For longer texts with a fixed narrative style, such as crash reports, we restructure the information into four perspectives: E_{road}, E_{static}, E_{agents}, and $E_{weather}$. For image inputs, we use Chain-of-Thought (CoT) prompting strategy[45] to guide the model toward layered, risk-aware reasoning that is otherwise difficult to elicit from a single-pass description. For video data, we apply temporal downsampling and memory-based processing to to balance efficiency with temporal fidelity, allowing the model to focus on key scene changes while reconstructing the underlying road network and temporal continuity of agent motion. Section 3.3 elaborates on its key prompt designs, and Section 3.4 outlines the multimodal input pipeline.

Constructing the road structure based on E_l serves as an important intermediate step. It directly impacts the subsequent generation of other elements and the risk factors within the final corner cases. To achieve this, two approaches are employed. In the first approach, E_l generated by interpreters are used to invoke the net generator, which produces the network in XML format. Alternatively, real-world road geometry is retrieved from OpenStreetMap[46] using GPS input and converted into the net.XML format. Following this, the object generator creates static elements such as lane markings, traffic signs, fences, and traffic cones, which remain unchanged unless acted upon during the simulation. Meanwhile, the agent generator, creates a set of agents based on E_{agents}, which includes a diverse range of road users, such as pedestrians, cyclists, and various types of vehicles including trucks and passenger cars. LLMs are used to place agents within the scenario and assign appropriate speeds. Finally, closed-loop simulations are performed using a data-driven agent model to replicate human-like behaviors and interactions under the specified traffic conditions.

3.3 Prompting techniques

AutoScenario is designed to construct safety-critical

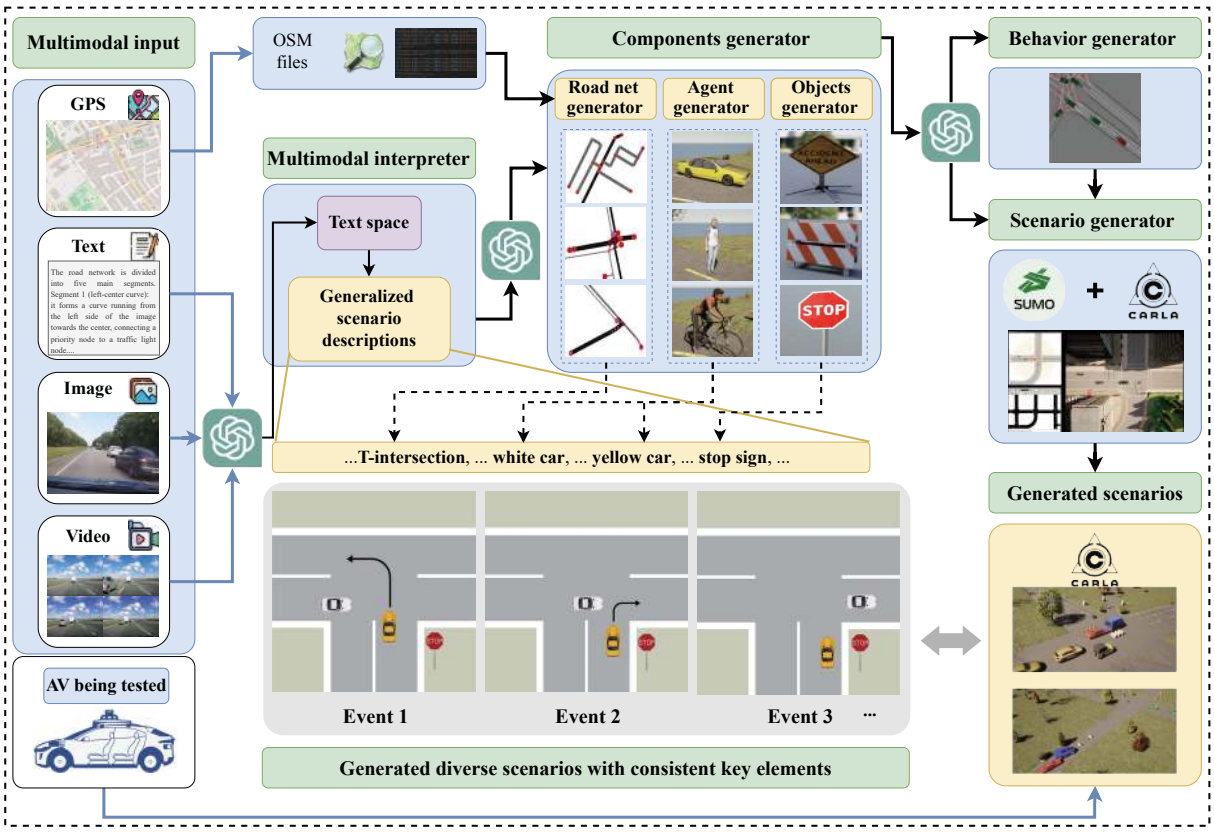


Fig. 2 AutoScenario system overview: it accepts multimodal inputs, which are processed by the multimodal interpreter. Based on the generalized scenario description, the components generator constructs the key scenario elements. These components are then passed to the scenario generator to enable scenario-based testing. OSM: OpenStreetMap.

scenarios from diverse input sources, including text, images, and videos. Risky factors embedded in these multimodal inputs are extracted using interpreters powered by VLMs. To enhance diversity, the core conflict elements are preserved in scenario descriptions, while other components are systematically diversified.

Serving as the foundational step in translating real-world inputs into textual descriptions, the VLM's capability in understanding complex scenarios plays a pivotal role. To improve its performance, we incorporate a range of prompting strategies within the Multimodal Interpreter, enabling more accurate scenario interpretation, key risk factor extraction, and critical element generation. Taking the user request input as an example, as illustrated in Fig. 3, we apply role prompting to define the model's persona and task objective, thereby focusing its attention on relevant aspects of the scenario. Instruction prompting and CoT prompting are used to guide the reasoning process through structured steps, improving the model's understanding of both the task and its constraints. The ## Description section prompts the model to restate and internalize the task, the ## Reasoning section encourages step-by-step thinking, and the ## Decision section ensures that the final output adheres to the required format. Additionally, we provide relevant examples to serve as references for the VLM, enabling it to generate more accurate and detailed road descriptions. Finally, a structured output format is established, incorporating key elements such as road geometry, traffic conditions, and testing objectives, to support reliable automated parsing and downstream task execution.

To improve the accuracy of information extraction from images and videos, additional prompting techniques are integrated into the full pipeline. As illustrated in Fig. 4, the multimodal interpreter for image input is equipped with instructions to extract fine-grained road structure details and infer layout based on visual cues such as lane markings, surrounding buildings, and trees. For video inputs, the interpreter guides the model to reason from a first-person perspective, enabling it to track motion across

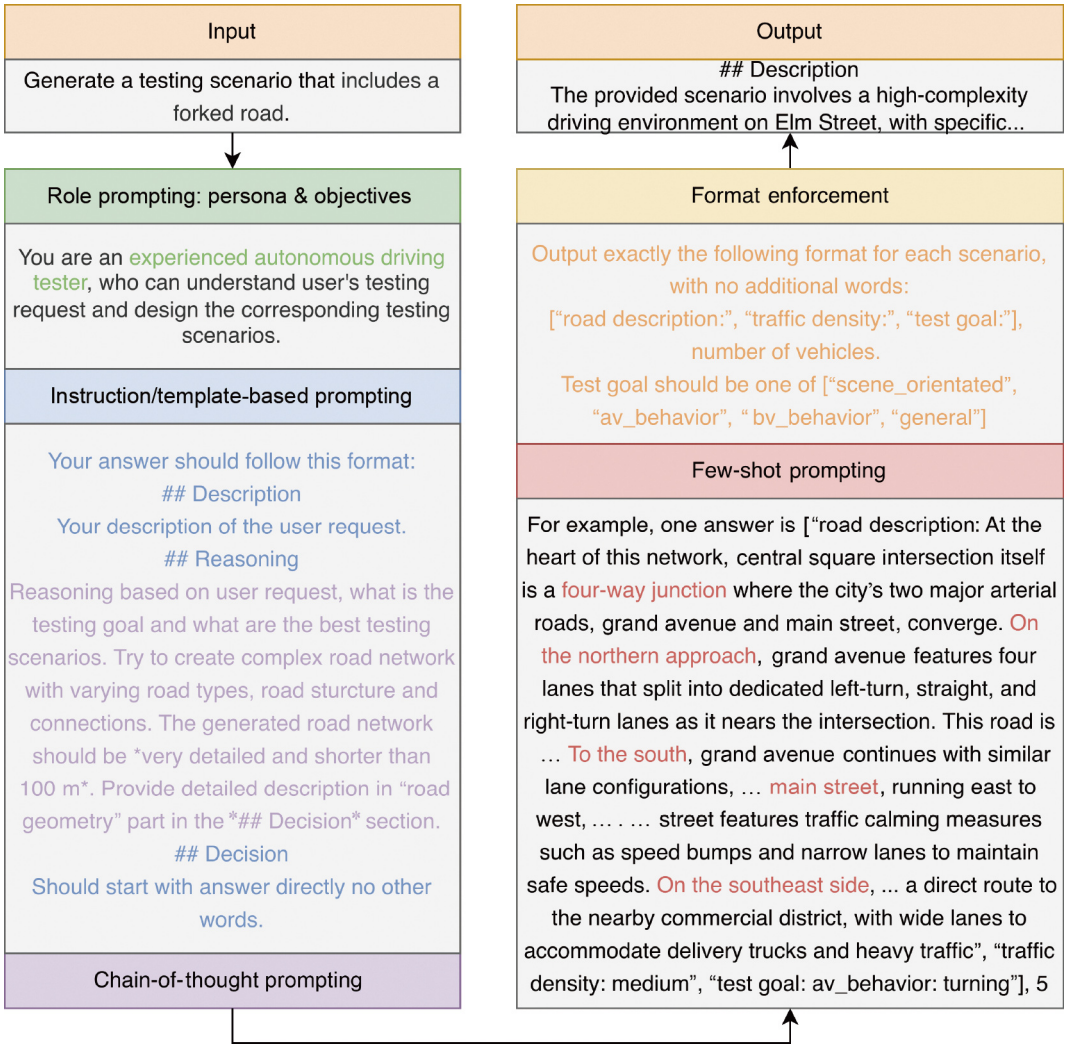


Fig. 3 Overview of core prompting strategies employed by the multimodal interpreter to interpret and structure user requests. The symbol "*" is intentionally included in the prompts to emphasize the key terms.

consecutive frames. Furthermore, a code example is incorporated to support the estimation of forward travel distance between the ego vehicle and leading vehicles using depth maps.

3.4 Generation with diverse input sources

Text inputs are processed by text interpreter to generate the final scenario description. AutoScenario supports user testing requests that describe scenarios at an abstract level, as demonstrated in Fig. 5. To expand these requests into detailed scene descriptions, an LLM is activated, using few-shot examples embedded in the prompts to guide generation. In addition to abstract descriptions, more detailed scenario specifications are also supported. One such application involves reconstructing critical moments leading up to a crash based on crash reports. An LLM is activated to extract the most relevant information from these reports and generate detailed scene descriptions. This process is guided by carefully designed prompts that supports the extraction and formulation of key elements.

A properly prompted VLM is used as the image interpreter to convert the input image into structured scenario descriptions. It is guided to carefully consider the four key perspectives of the scenario for comprehensive information extraction. During experimentation, we observed that the interpreter's performance degraded in complex scenes, particularly those with numerous buildings and dense traffic, leading to inaccuracies in extracting road network features. To address this, we introduce an enhanced prompt that guides the model to infer the geometric structure of the road network by leveraging contextual cues from surrounding buildings and parked vehicles.

Image interpreter

Role prompting: persona & objectives

You are an assistant designed to generate detailed autonomous vehicle testing scenarios based on given image data. Your primary tasks are as follows:
(1) Understand the user's request.
(2) Analyze the testing goal.
(3) Generate a rich description of the road geometry, vehicle behaviors, and spatial configuration of the scenario.

Instruction prompting

When describing the road network, aim for realism. Common road structures include: Crossroads, T-intersections, Y-intersections, ramps and merge zones, deformed/angled intersections...
Geometric accuracy is important:
- Lane widths: standard ~3.6 m; turn lanes may be wider.
- Intersection angles: report as ~30°, 45°, 90°, etc.
- Road length: keep total network within 100 m
You can use surrounding elements like: building facades, roadside trees, moving or parked vehicles. These references help infer road directions, lane divisions, and agent positioning.
Provide relative or absolute positions for all surrounding objects in the scene.

Video interpreter

Instruction prompting

Based on the input video, reconstruct the road geometry and layout, and narrate the scene from the agent vehicle's point of view. Describe in detail what the vehicle sees at each moment—including road types, lane directions, nearby vehicles or pedestrians, and static objects—and explain how it makes navigation decisions such as turning, yielding, or lane changes.

Few-shot prompting

The following code realize the function of calculate the distance of the vehicle's forward movement.
The input is the first-person perspective image of the vehicle at the previous and next time points, and the depth map generated by these two images.
The output is the estimated distance of the vehicle's forward movement.
The code is as follows:

```
#####################################
def calculate_distance_from_optical_flow(depth1_path,
    depth2_path, frame1_path, frame2_path):
    # Read and resize two frames
    scale_factor = 0.5
    frame1 = cv2.imread(frame1_path)
    frame2 = cv2.imread(frame2_path)
    ...
```

You can use the provided code as a reference to compute the estimated forward distance....

CARLA API

Few-shot prompting

```
# Function to spawn a dynamic vehicle
def spawn_vehicle(blueprint_name, location, rotation,
color=None):
    blueprint = blueprint_library.find(f'vehicle.omni.
{blueprint_name}')
    If color:
        blueprint.set_attribute('color', color)
    transform = carla.Transform(location, rotation)
    vehicle = world.try_spawn_actor(blueprint, transform)
    If vehicle is not None:
        vehicle.set_autopilot(False)  # Control manually
    vehicle.apply_control(carla.VehicleControl(brake=1.0))
    return vehicle
...
# Change the weather
weather = carla.WeatherParameters.ClearNoon  # Change if
the weather is in description
```

Instruction prompting

(1) Determine the number of vehicle and static objects based on the given scene information and context.
(2) The vehicles can only be chosen from bike, car, jeep, motorcycle, suv, truck, and van. The static objects can only be chosen from warningconstruction, streetbarrier, constructioncone, and warningaccident.
(3) Generate vehicle colors.

Fig. 4 **Main prompting techniques employed across other components. API: Application Programming Interface.**

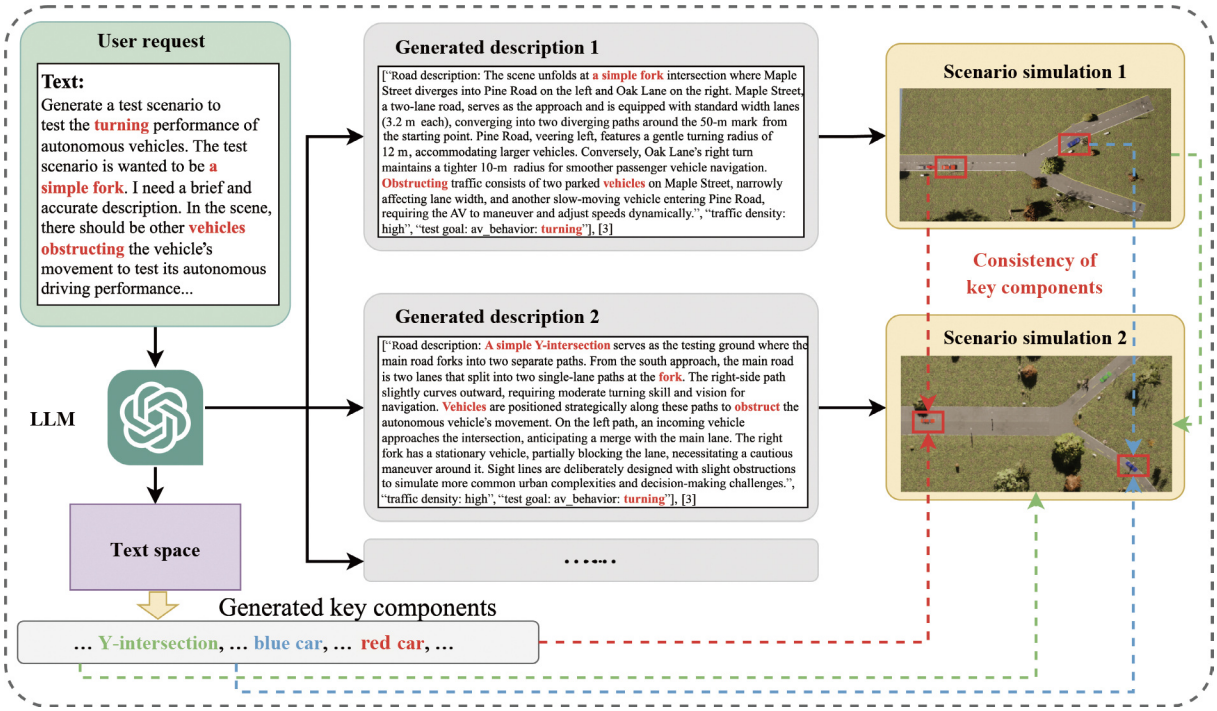


Fig. 5 **AutoScenario generation pipeline with user request. The user specifies the testing requirements (left), and multiple scenario descriptions are generated by the Text Interpreter (middle). Irrelevant parts of the description are omitted, while key elements are highlighted in bold. On the right, a sampled scenario generated by CARLA and LLM for each description is displayed.**

This enhancement notably improves the model's performance in complex scenes. The scenario generation process is depicted in Fig. 6.

Additionally, when GPS data are provided, we can generate testing scenarios based on the text description while incorporating real-world road structures. This approach enables effective testing for deployment in this region, reflecting realistic traffic conditions, as demonstrated in Fig. 7.

For video input from dash camera, a VLM-based video interpreter is developed to extract road geometry and environmental features. In our experiments, we observe that the model's estimation of distances between surrounding vehicles and the ego vehicle, especially during driving trajectory analysis, was often inaccurate. To address this limitation, we introduce a code prompt that calculates the forward distance traveled by the vehicle between two timestamps, using the corresponding images and their depth maps. Unlike the original approach, which relied solely on video frames, our method incorporates depth map frames aligned with each image, providing richer spatial information. With these designs, the model's accuracy in estimating the distance traveled in the video, as well as the length and proportions of the road network, has significantly improved. The process of scenario generation is depicted in Fig. 8.

We demonstrate scenario generation using various types of inputs, including text, image, and video, along with the key design of their corresponding interpreters. In each case, road information and environmental features are extracted from the respective input. This

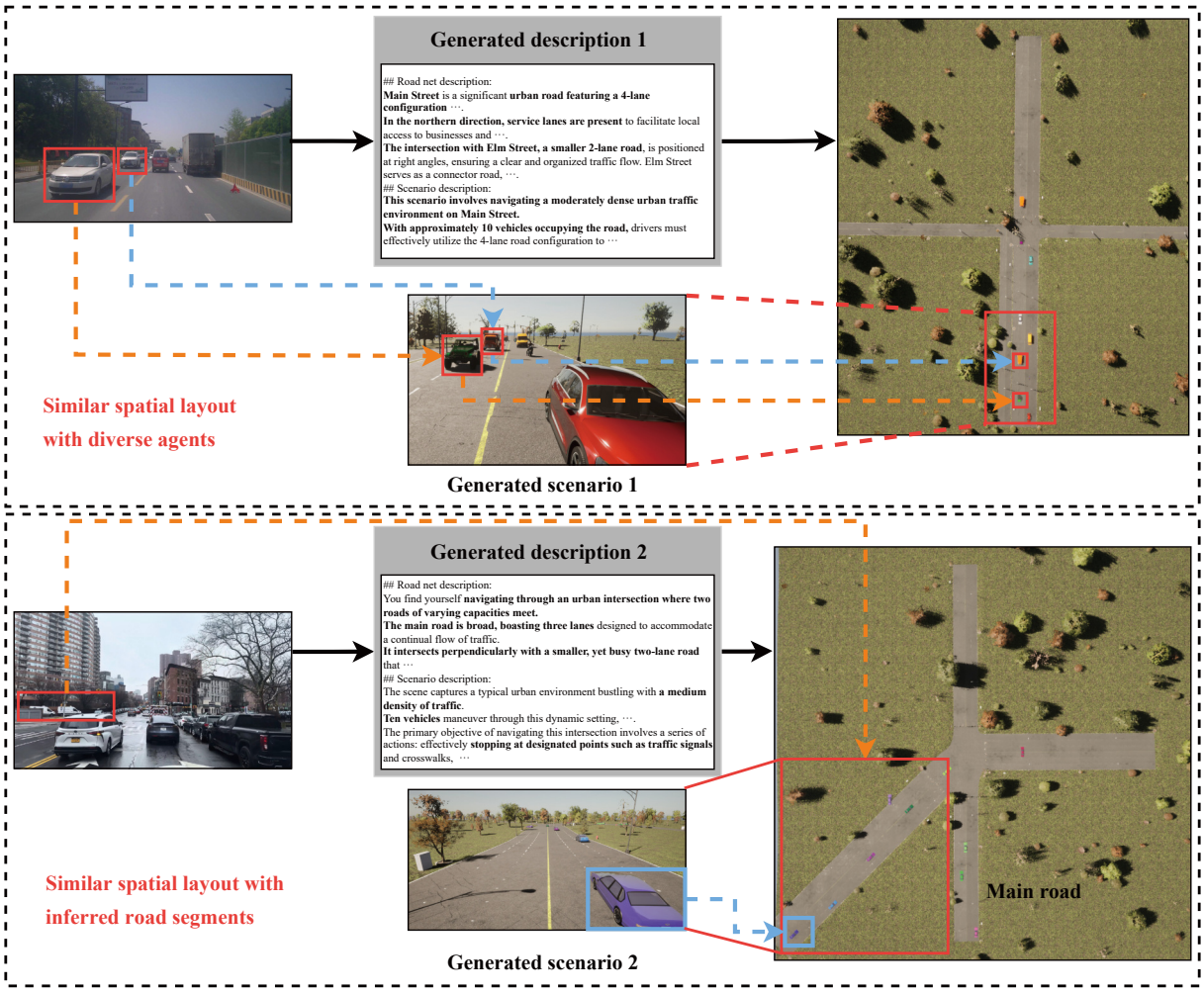


Fig. 6 AutoScenario generation pipeline with image input. Left column: real-world scenarios (top-left image from the CODA dataset), representing the ego perspective of the AV. Middle column (text): scenario description generated by the image interpreter; middle column (image): simulation scenario generated by CARLA and LLM, shown from a perspective different from the overhead view. Right column: overhead-view simulation scenarios generated by CARLA and LLM.

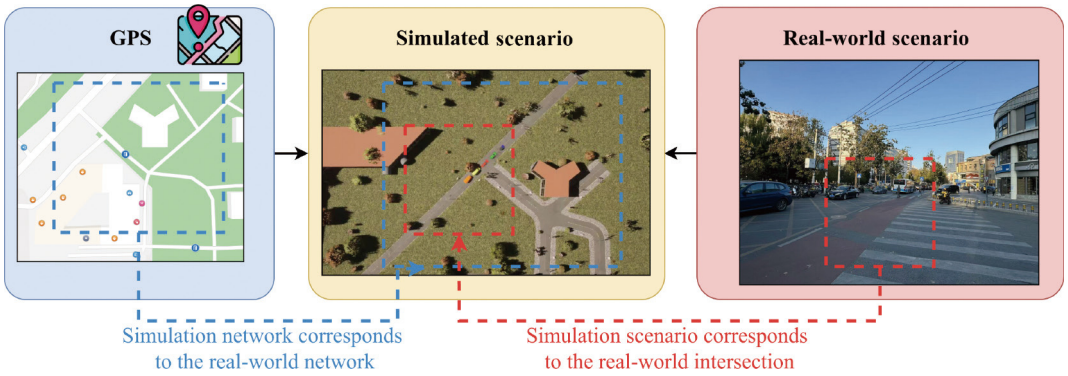


Fig. 7 AutoScenario generation with additional GPS input.

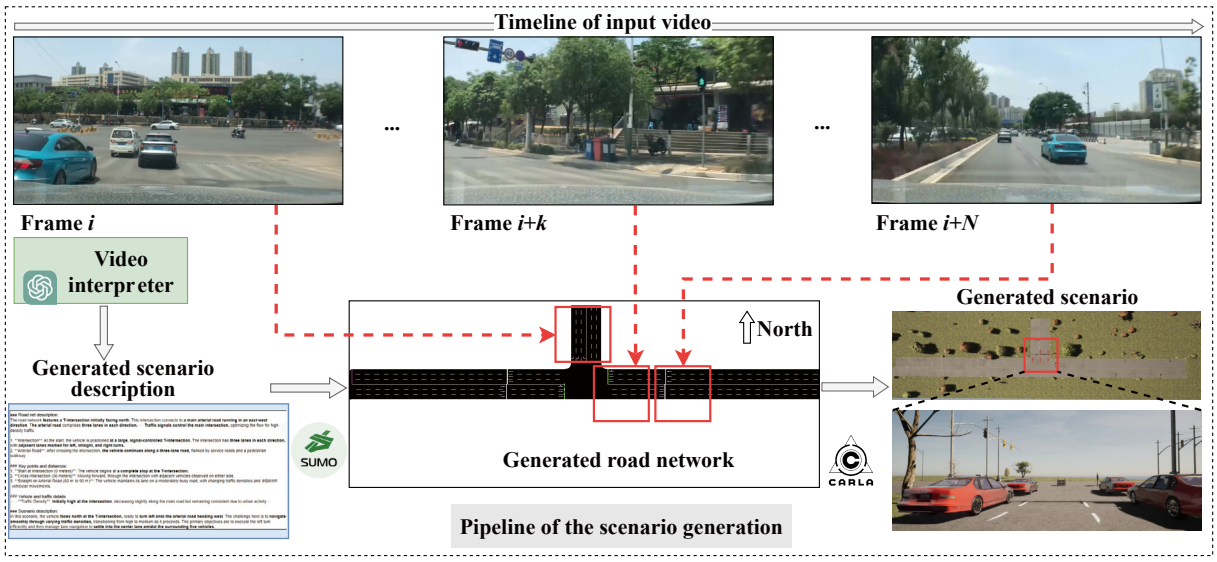


Fig. 8 Conversion from dashcam video to network file and simulation scenario. Top: frames extracted from the video. Bottom: scenario generation process—left: scenario description generated by the video interpreter; middle: corresponding network file generated by road net generator; right: generated scenario.

capability remains a critical component of the AutoScenario system, as it enables the translation of real-world information into a unified text-based format.

3.5 Collaborating with domain tools

Guided by the scenario description, SUMO, CARLA, and a data-driven behavior model are seamlessly integrated with the help of LLM to generate the final scenario, as illustrated in Fig. 9.

Net generator. Road geometry is a well structured object that can be represented by graph with nodes and edges. To reduce the error rate for pure generation, we prompt LLM to generate SUMO compatible node and edge file defined in XML format, then convert to full graph with domain knowledge, i.e., rules with SUMO tools. See Fig. 9 for an illustration and Section 4 for more detailed examples. This design is not confined to SUMO or its XML formats. Since road network naturally represents a graph structure, it can be represented by other structured languages[47] and processed by graph tools[48], compatible to other simulators such as MATSim[49].

Scenario generator. We use LLM to directly generate Python code that controls 3D scenario agents via the CARLA Python API. Blueprints from CARLA's library are automatically utilized to construct diverse road users and simulate varying weather conditions. The digital twin tool is further integrated to enhance visual realism, while additional critical risk factors are automatically generated through the API.

Agent behavior modeling. Once the initial state at a critical moment is generated based on the universal scene description E_l, a trained data-driven behavior

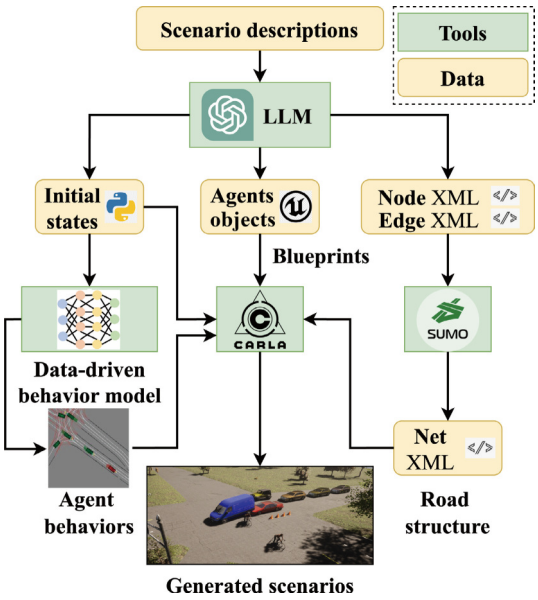


Fig. 9 Tools utilized in AutoScenario: SUMO, CARLA, and a data-driven behavior model.

model is employed to simulate agent behaviors that closely replicate human driving actions within the scenario.

4 Experiment

In this section, we evaluate the performance of our AutoScenario framework both quantitatively and qualitatively. First, we examine its ability to generate realistic, diverse, and controllable scenarios. Next, we demonstrate its application in creating safety-critical scenarios from multimodal inputs for AV testing, and additionally, we quantitatively assess the challenges posed by these scenarios through the performance of LLM-based AVs. Finally, we highlight the key components of the framework through an ablation study.

4.1 Experiment setup

We utilize the Generative Pre-trained Trainsformer (GPT-4) API (gpt-4o-05-23) with carefully designed prompts as the multimodal LLM, and leverage a pretrained data-driven model to simulate agent behavior. Our framework is model-agnostic and can be adapted to alternative GPT versions or API-based LLMs with only minor interface adjustments. Unlike locally deployed models, the API-based approach requires no environment-specific setup or Graphics Processing Unit (GPU) resource management, enabling faster responses and more stable deployment. Compared with encoder-only models such as Bidirectional Encoder Representations from Transformers (BERT) or sequence-to-sequence models like T5, GPT-4 is trained on web-scale data, offering significantly stronger semantic reasoning, broader world knowledge, and more reliable instruction following—capabilities that are essential for interpreting complex traffic scenes and generating coherent scenario descriptions. The pretrained data-driven model takes in vectorized map features and historical agent trajectories to predict future motions. Implementation details of this model follow Ref. [50]. SUMO 1.21.0 and CARLA 0.9.15 are integrated into the pipeline and are invoked as needed during scenario generation and simulation. All experiments are conducted on a single NVIDIA RTX 3090 GPU.

To ensure computational efficiency, the AutoScenario pipeline is structured so that its core components scale favorably with scenario complexity. The scenario-description module relies on the GPT-based API, thus for a fixed input modality the number of processed tokens remains within a narrow and nearly constant range. As a result, its runtime does not grow with the number of agents or interaction complexity, contributing a stable and predictable latency per generated scenario. For agent-level behavior simulation, AutoScenario employs a data-driven behavior model with GPU-accelerated parallelization where all agents within a scenario are evaluated in batched operation. This design prevents computational cost from scaling proportionally with scenario complexity and enables efficient generation of rich, high-interaction multi-agent scenarios.

4.2 Qualitative evaluation

We present several samples generated by AutoScenario using multimodal inputs.

Figure 10 shows two scenarios reconstructed from crash reports involving vehicle-to-vehicle conflicts. In both cases, the generated scenarios successfully capture key elements and critical moments described in the original reports. The relative positions of agents are well aligned with the text description. Additionally, Fig. 11 shows two more examples focusing on interactions between vulnerable road users and vehicles involved in the accidents. These scenarios further demonstrate AutoScenario's ability to reconstruct complex, safety-critical interactions based on crash reports.

Furthermore, beyond the high controllability of text

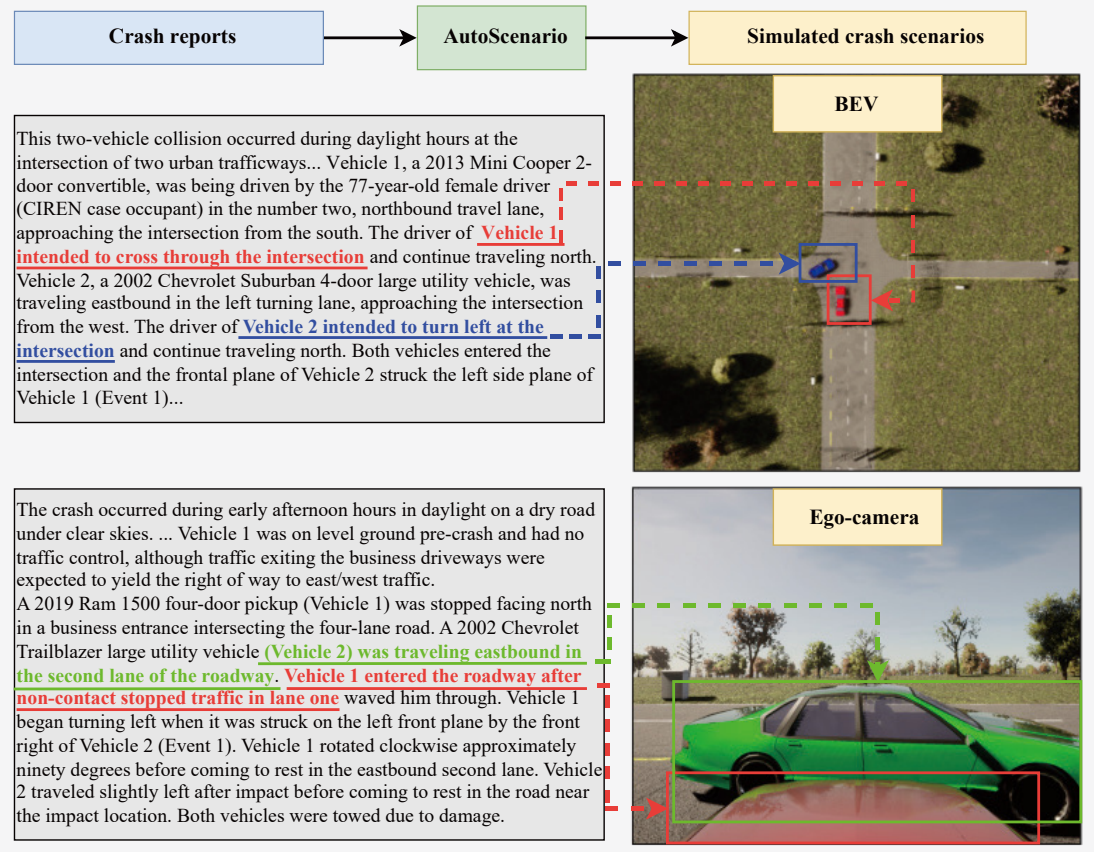


Fig. 10 AutoScenario using crash report from Ref. [51] as input.

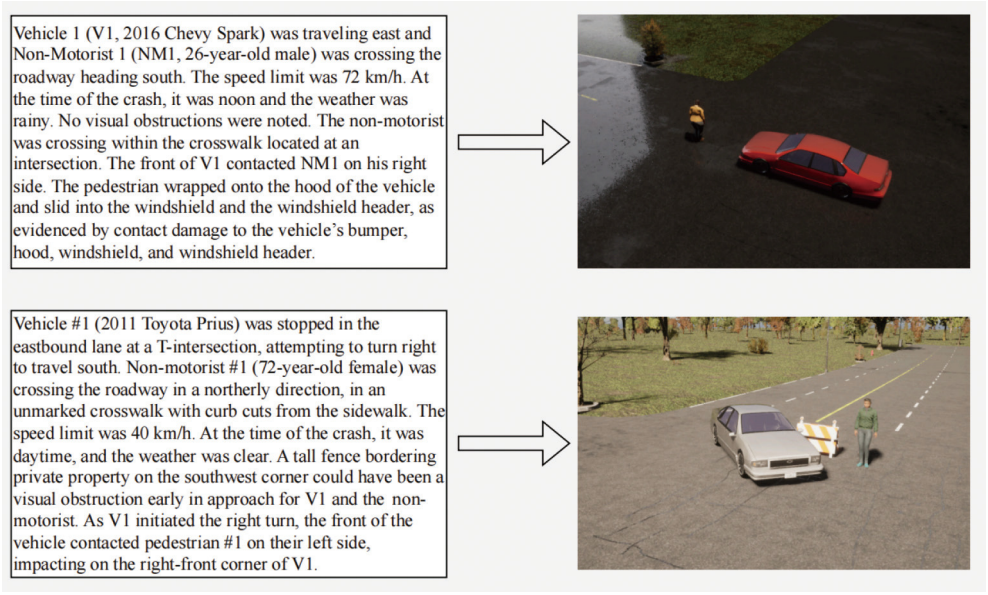


Fig. 11 Corner cases involving vulnerable road users generated by AutoScenario.

description, the system also demonstrates strong diversity in the generated scenarios when guided by well-designed prompts. Figure 12 listes four samples generated with image interpreter. The interpreter effectively identified critical scene elements, including road network configurations, vehicle count and color, and obstacles such as construction cones. This information is then seamlessly translated into the

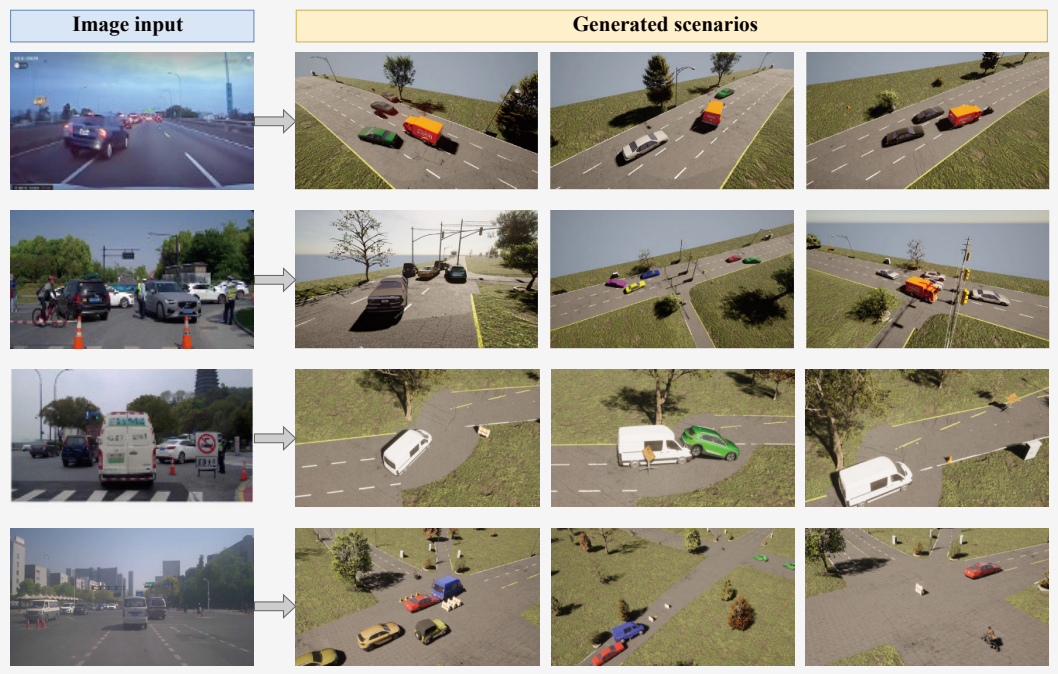


Fig. 12 Diverse scenarios generation based on the same input from image interpreter.

generated 3D scene by the components generator and the scenario generator.

Figure 13 further illustrates the challenging scenarios generated by AutoScenario. These include challenging scenarios near curved roads (top) and intersections (bottom), driven by a combination of factors: carefully designed initial states and the interactive behaviors of agents. On one hand, it incorporates elements such as BV lane changes and AV turns at intersections, reflecting risky situations observed in real-world driving. On the other hand, by integrating with data-driven simulated agents, AutoScenario creates safety-critical scenarios, enabling interactive and robust testing of AVs.

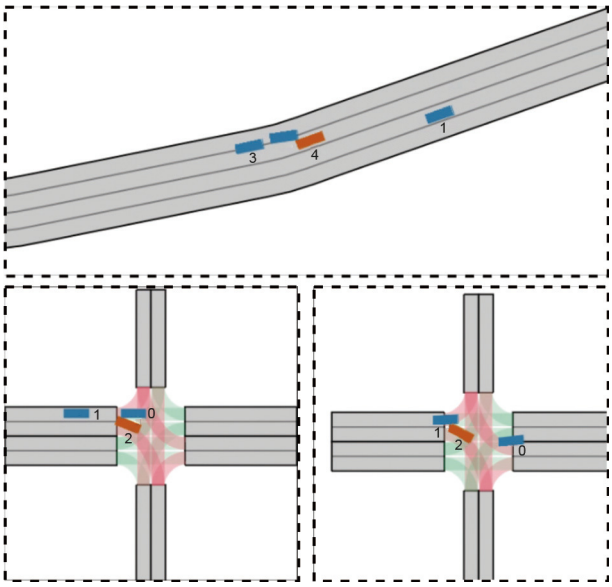


Fig. 13 AV collides with other vehicles following multiple simulation steps from initial states. The orange vehicle represents the LLM-based autonomous vehicle, while the blue vehicles are simulated agents controlled by data-driven models. All vehicles are initialized using the components generators.

4.3 Quantitative evaluation

4.3.1 Metric based evaluation

To further systematically evaluate the diversity and fidelity of generation, we introduce a set of metrics to capture the quality of the two main components inside the generation process, the interpreter and the generator. The metrics are grouped into two categories as follows:

(1) Conformity metrics. These metrics assess how well the generated scenarios align with the text descriptions. They evaluate multiple elements, including the road network structure, static objects, agents, and intermediate python codes. In particular, we measure the generation success rate, which quantifies the percentage of intermediate codes that are successfully parsed and executed by the simulation engines, i.e., SUMO and CARLA. A high success rate indicates that the generated outputs are not only

syntactically valid but also semantically consistent with the target platform.

(2) Diversity metrics. These metrics quantify the variation in key scenario attributes across generated samples. We focus on attributes such as road network configurations, object placements, and agent distributions. To ensure meaningful diversity analysis, we apply these metrics to scenarios derived from identical input prompts but diversified through the generation process.

We evaluate the framework across three representative scenario categories that are prone to corner cases: general, intersections, and Construction Zones (CZs) scenarios. For each category, five input images are processed through the AutoScenario pipeline. Each input image is then diversified into ten distinct testing scenarios.

As shown in Table 1, AutoScenario exhibits high success rates in producing meaningful scenarios. Additionally, the system achieves high accuracy in generating the specified number and color of vehicles, as well as the type of road obstacles, according to descriptions generated by the interpreter in most cases. The main causes of failures include incorrect formatting of keywords (e.g., extra "#" characters) in network generation and blueprint name reusing errors (e.g., "vehicle.omni.vehicle.omni.bus"). Given the complexity of the prompt, these success rates and accuracy levels represent quite favorable results.

To evaluate road network complexity, we calculate the mean and standard deviation of the total number of lanes, edges, and route lengths in the generated road networks. As shown in Table 2, these metrics span a wide range across each scenario set, demonstrating the consistent diversity of the generated road layouts.

Beyond the road network structure, other critical scenario elements also exhibit substantial diversity. As presented in Table 3, the standard deviation values indicate a wide distribution range for each metric, reflecting substantial diversity. Compared to the general and CZ scenarios, the intersection scenarios feature the highest number of agents. This is expected as intersections typically involve traffic approaching from multiple directions. The CZ scenarios have the most static objects, aligning with their characteristic environment. To simulate potential corner cases, the shortest distances between agents across all three types of scenarios tend to be about 4 to 5 m, reflecting dense and potentially risky interactions. In general scenarios, straight roads without intersections are prevalent, the mean vehicle yaw angle remains close to 0°. Meanwhile, in intersection and CZ scenarios, vehicles are more likely to perform turning, lane-changing, or merging maneuvers, leading to a higher standard deviation in yaw angles.

Table 1 Alignment of generation with the text description.

Metric	General	Intersection	CZ
Accuracy	1.0	1.0	1.0
Vehicle attribute	0.9	0.7	0.7
Static objects attribute	0.9	0.9	1.0
Generation success rate	0.9	0.7	0.6

Table 2 Diversity of generated road network.

Metric	General	Intersection	CZ
Number of lanes	6.0 ± 3.3	11.0 ± 3.0	11.0 ± 2.9
Number of edges	5.7 ± 3.7	7.7 ± 2.4	6.0 ± 1.6
Route length (m)	260.7 ± 114.6	540.0 ± 200.2	420.6 ± 64.6

4.3.2 Text based similarity validation

To validate the similarity between generated scenario and original input image, we design an automated pipeline that performs comparison within the text space produced by the VLM interpreter. A detailed example of this process is shown in Fig. 14. In addition to comparing overall scene descriptions, we further decompose them into their main components and conduct fine-grained comparisons for each element. Moreover, leveraging the ability to manipulate scene perspectives in CARLA, we select two distinct views of each generated scene: the Bird's-Eye View (BEV) and the ego vehicle view. These complementary perspectives enable cross-verification of the semantic and structural similarity between the generated scenario and the original image.

As shown in Table 4, the generated scenarios exhibit a high degree of similarity compared with original inputs in the universal text space while maintaining diversity in specific scenario details.

In the analysis of separate description sections from the ego car's perspective, the vehicle behavior section shows the highest similarity. This could be

Table 3 Diversity of generated scenarios.

Metric	General	Intersection	CZ
Number of agents	4.2 ± 2.2	5.6± 2.3	3.3 ± 0.9
Number of cbjects	2.6 ± 2.3	2.4± 1.1	3.6 ± 3.0
Shortest distance (m)	4.4 ± 2.3	4.3± 2.8	5.1 ± 2.8
Vehicle yaw (°)	0.3 ± 69.8	14.8± 93.4	16.5 ± 90.4

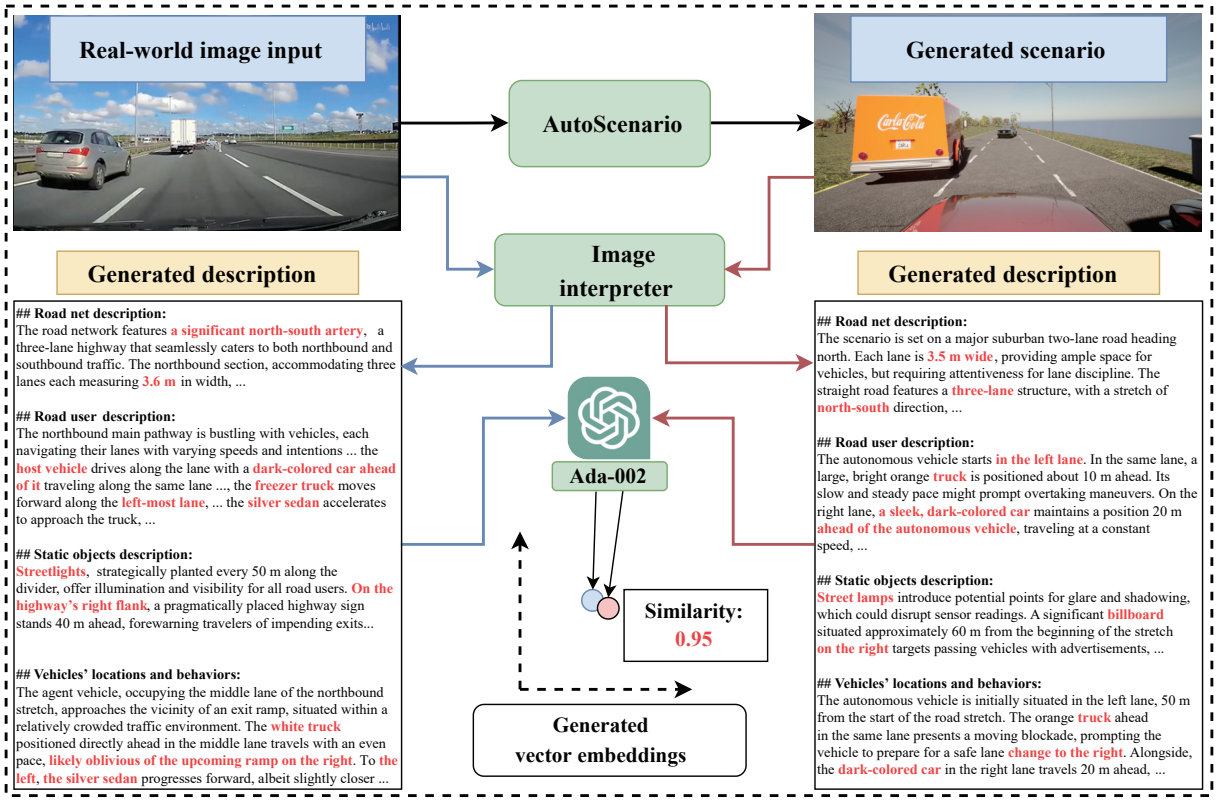


Fig. 14 Description generated from a real-world image (left) and description generated from a scenario created by AutoScenario (right).

Table 4 Similarity between original input and generated scenarios with ego vehicle view and BEV.

Similarity	Ego car view	BEV
Overall scene	0.93 ± 0.02	0.94 ± 0.01
Net	0.91 ± 0.02	0.92 ± 0.02
Road user	0.91 ± 0.02	0.90 ± 0.02
Static object	0.91 ± 0.02	0.90 ± 0.01
Vehicle behavior	0.93 ± 0.04	0.86 ± 0.10

attributed to the prominence of vehicles in the image from the car's viewpoint, as they occupy the majority of the visual space. Conversely, in the analysis of separate description sections from the BEV, the vehicle behavior section exhibits the lowest similarity. This may be due to the overhead perspective, where the road network and broader scene occupy most of the visual space, and vehicles appear very small, making their behavior more challenging to describe accurately.

4.3.3 Testing based complexity evaluation

Existing LLM-based methods such as LLMScenario[34] and ChatScene[37] rely on retrieving and modifying examples from a pre-existing scenario database, limiting their ability to generate fully novel traffic configurations. Our method, by contrast, synthesizes complete scenarios directly from multimodal inputs without template dependence. Therefore, instead of comparing to retrieval-based LLM frameworks with different assumptions, we use SUMO's RandomTrip as a non-LLM baseline to focus on generative quality. We compare AutoScenario-generated scenarios with those produced by randomly placing vehicles using RandomTrip from SUMO toolkit, with both evaluated on road networks reconstructed from real-world images.

As shown in Fig. 15, AutoScenario strategically positions AV and BVs based on spatial cues extracted from input images, while their behaviors are governed by simulation-based agent models. In contrast, scenarios generated by SUMO initialize the AV and BV at random positions, with BV behaviors controlled by Intelligent Driver Model (IDM) -based models. The results demonstrate that AutoScenario provides higher controllability and realism, particularly in generating complex and safety-critical scenarios.

The challengingness of a generated scenario is relative to the capabilities of the AV being tested. Although traditional metrics such as Time-To-Collision

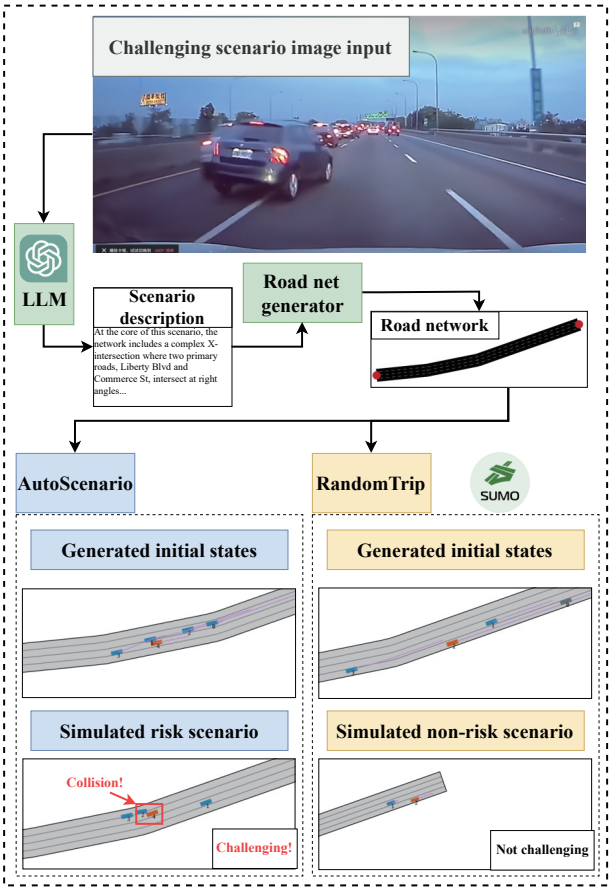


Fig. 15 Comparison of scenario generation pipelines.

(TTC) capture local, momentary risk, they are difficult to compare across different interaction types (e.g., cut-ins versus rear-end conflicts) and thus cannot fully represent overall scenario difficulty. Therefore, we assess challengingness using the AV's integrated performance in safety, comfort, and task completion, as a more holistic and comparable indicator of scenario difficulty. The driving score is calculated as a weighted combination of ride comfort, driving efficiency, and driving safety. The route completion value is defined as the ratio of the distance traveled by the driver agent to the total length of the predefined route. The total score is calculated by multiplying the driving score by the route completion. For more details, see LimSim++[52]. In our experiments, we utilize five different road networks, with each network generating five distinct initial vehicle positions using both our proposed method and RandomTrip. Beyond the evaluation metrics mentioned earlier, we also record the number of collisions across these 25 experiments to compute the overall collision rate. As shown in Table 5,

Table 5 Comparison of scenario difficulty for AutoScenario and RandomTrip, measured through LLM-based AV performance metrics. Arrows indicate metric preference: ↓ denotes metrics where lower values are better, and ↑ denotes those where higher values are better.

Metric	Ours	RandomTrip
Route completion ↓	0.86 ± 0.26	0.92 ± 0.24
Driving score ↓	65.24 ± 16.43	72.87 ± 19.24
Total score ↓	59.47 ± 25.25	69.66 ± 26.15
Use time (s)	84.09 ± 44.62	107.96 ± 49.86
Success rate ↓	0.76 ± 0.44	0.88 ± 0.33
Collision rate ↑	0.20	0.08

AutoScenario generates more challenging scenarios compared to RandomTrip.

4.4 Ablation study

We conduct an ablation study to evaluate key design choices across the three main stages of the generation pipeline: the interpreter, the components generator, and the scenario generator. For each stage, we remove its critical element—specifically, the full interpreter, the CoT reasoning in the components generator prompt, and the simulator-related prior knowledge in the prompts used by the scenario generator for text-based inputs. We then compare the success rate, defined in Section 4.3.1, which measures the proportion of scenarios whose intermediate codes are successfully parsed and executed by the simulation engines. The result is presented in Table 6.

After removing the interpreter, which generates detailed narrative descriptions for each scenario, AutoScenario fails to produce diverse network structures in a single pass, let alone complete the subsequent steps. This highlights the necessity of the multi-stage generation process and underscores the importance of the comprehensive global scenario description.

We evaluate the impact of removing the CoT mechanism, a crucial reasoning technique, from its prompt. This removal leads to a noticeable decline in generation accuracy, as shown in Table 6, primarily

Table 6 Ablation study: generation success rate under different ablations.

Method	Generation success rate
Ours	0.8
Without interpreter	0
Without reasoning section	0.4
Without prior knowledge	0.2

due to three recurring issues. First, critical attributes are missing in element definitions, such as the absence of the "shape" attribute in the "lane" element, preventing SUMO from correctly interpreting lane configurations. Second, attribute values fall outside the allowed enumeration, as with the "spreadType" attribute in the "edge" element, where "left" was used instead of the valid values ("right", "center", or "roadCenter"). Third, undeclared attributes are used, like the "function" attribute in the "edge" element, which references an incorrect XML schema. These issues collectively highlight the importance of CoT prompting in generating well-formed, schema-compliant network configurations.

In the scenario generator, we evaluate the impact of removing the code examples and prior knowledge constraints from the prompt. As shown in Table 6, the generation success rate drops significantly, from 0.8 to 0.2, for the generation process. This decline is expected, as CARLA follows a specific protocol, and without the code examples, the world knowledge embedded in the LLM is insufficient to fully generate the required functionality.

4.5 Online learning for AV with corner cases

AVs often underperform when encountering rare or safety-critical situations that are underrepresented in training data. To evaluate how effectively our generated corner cases can mitigate this limitation, we introduce an online learning mechanism that enables iterative adaptation of the LLM-based decision module using corner-case scenarios identified through experiments in Section 4.3.3.

As illustrated in Fig. 16, once crash scenarios are detected from previous simulation rounds, we extract two types of information to support the next iteration of online learning. First, we store the AV's input prompts that describe the collision or near-miss context, along with the corresponding decision outputs generated by the LLM-based AV. Second, we incorporate human feedback—such as suggestions to decelerate earlier, increase headway, or choose a safer lane—to provide corrective guidance. All of these elements are accumulated in a memory buffer and serve as targeted examples to refine the model's reasoning and enhance its safety-oriented decision-making in subsequent rounds.

To evaluate whether the AV has improved its ability to handle similar situations, we compare its previous performance with its updated performance on the 25 challenging scenarios introduced in Section 4.3.3. Note that the testing is conducted in a closed-loop manner, meaning the adapted AV encounters scenario variations that differ from those seen during prior runs. As summarized in Table 7, emphasizing corner cases during online adaptation consistently enhances overall driving performance and strengthens the AV's safety awareness.

5 Conclusion and Future Work

We present a scenario generation framework that integrates LLMs, VLMs, and data-driven models. This is the first system to seamlessly translate multimodal real-world data into simulated scenarios, offering a highly controllable and flexible simulation tool. Furthermore, the framework is readily extensible to diverse environmental conditions, including snow, rain, nighttime lighting, and other adverse settings—through the underlying simulator API. In CARLA, for instance, these variations can be introduced by adjusting weather parameters and asset configurations. Transferring and generalizing risky scenarios from the real world to simulators like CARLA, in terms of both objects and behaviors, is a foundational step toward robust AV testing. In the future, we aim to enhance photorealism using 3D Gaussian Splatting (3DGS) or diffusion models.

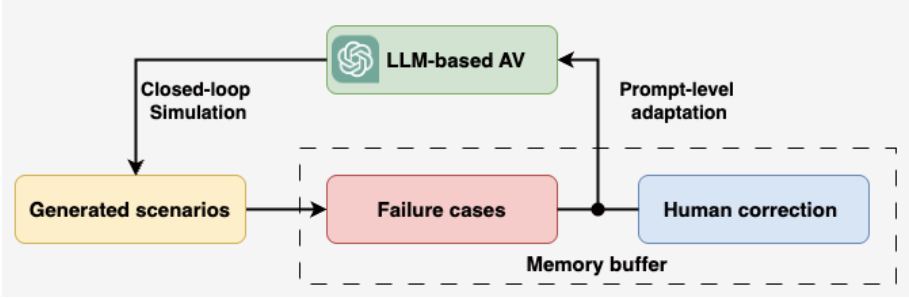


Fig. 16 Online learning pipeline for AV.

Table 7 Improving AV performance with corner cases. ↓ indicates that lower values correspond to better AV performance, while ↑ indicates that higher values are better.

Metric	AV without hints	AV with hints
Route completion ↑	0.86 ± 0.26	0.96 ± 0.19
Driving score ↑	65.24 ± 16.43	70.71 ± 19.08
Total score ↑	59.47 ± 25.25	70.06 ± 21.01
Use time (s)	84.09 ± 44.62	141.80 ± 90.21
Success rate ↑	0.76 ± 0.44	0.88 ± 0.33
Collision rate ↓	0.2	0.12

Acknowledgment

The authors gratefully acknowledge the support of our collaborators. Runze Yuan contributed to refining the manuscript figures, Ximiao Dai refined the video interpreter component, and Xuanhan Wang conducted the experiments for the online learning of AV with corner cases. This work was supported by the National Natural Science Foundation of China (No. 62473224) and DiDi GAIA Collaborative Research Funds.

References

[1] H. X. Liu and S. Feng, Curse of rarity for autonomous vehicles, *Nat. Commun.*, vol. 15, no. 1, p. 4808, 2024.

[2] K. Li, K. Chen, H. Wang, L. Hong, C. Ye, J. Han, Y. Chen, W. Zhang, C. Xu, D. Y. Yeung, et al., CODA: A real-world road corner case dataset for object detection in autonomous driving, in *Proc. 17th European Conf. Computer Vision*. Tel Aviv, Israel, 2022, pp. 406–423.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, nuScenes: A multimodal dataset for autonomous driving, in *Proc. 2020 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Seattle, WA, USA, 2020, pp. 11618–11628.

[4] A. Geiger, P. Lenz, and R. Urtasun, Are we ready for autonomous driving? The KITTI vision benchmark suite, in *Proc. 2012 IEEE Conf. Computer Vision and Pattern Recognition*, Providence, RI, USA, 2012, pp. 3354–3361.

[5] J. Mao, M. Niu, C. Jiang, H. Liang, J. Chen, X. Liang, Y. Li, C. Ye, W. Zhang, Z. Li, et al., One million scenes for autonomous driving: ONCE dataset, arXiv preprint arXiv: 2106.11037, 2021.

[6] A. Rana and A. Malhi, Building safer autonomous agents by leveraging risky driving behavior knowledge, in *Proc. 2021 Int. Conf. Communications, Computing, Cybersecurity, and Informatics*, Beijing, China, 2021, pp. 1–6.

[7] M. Althoff and S. Lutz, Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles, in *Proc. 2018 IEEE Intelligent Vehicles Symp.*, Changshu, China, 2018, pp. 1326–1333.

[8] 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 *Proc. 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Nashville, TN, USA, 2021, pp. 9904–9913.

[9] W. Ding, M. Xu, and D. Zhao, CMTS: A conditional multiple trajectory synthesizer for generating safety-critical driving scenarios, in *Proc. 2020 IEEE Int. Conf. Robotics and Automation*, Paris, France, 2020, pp. 4314–4321.

[10] D. Rempe, J. Philion, L. J. Guibas, S. Fidler, and O. Litany, Generating useful accident-prone driving scenarios via a learned traffic prior, in *Proc. 2022 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, New Orleans, LA, USA, 2022, pp. 17284–17294.

[11] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al., Evaluating large language models trained on code, arXiv preprint arXiv: 2107.03374, 2021.

[12] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. Dal Lago, et al., Competition-level code generation with AlphaCode, *Science*, vol. 378, no. 6624, pp. 1092–1097, 2022.

[13] S. Imani, L. Du, and H. Shrivastava, MathPrompter: Mathematical reasoning using large language models, in *Proc. 61st Annu. Meeting of the Association for Computational Linguistics*, Toronto, Canada, 2023, pp. 37–42.

[14] V. Kumaran, J. Rowe, B. Mott, and J. Lester, SCENECRAFT: Automating interactive narrative scene generation in digital games with large language models, in *Proc. 19th AAAI Conf. Artificial Intelligence and Interactive Digital Entertainment*, Salt Lake City, UT, USA, 2023, pp. 86–96.

[15] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner, Microscopic traffic simulation using SUMO, in *Proc. 2018 21st Int. Conf. Intelligent Transportation Systems*, Maui, HI, USA, 2018, pp. 2575–2582.

[16] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, CARLA: An open urban driving simulator, in *Proc. 1st Annu. Conf. Robot Learning*, Mountain View, CA, USA, 2017, pp. 1–16.

[17] Unreal Engine, Unreal engine, https://www.unrealengine.com, 2024.

[18] P. Zhang, B. Zhu, J. Zhao, T. Fan, and Y. Sun, Performance evaluation method for automated driving system in logical scenario, *Automot. Innov.*, vol. 5, no. 3, pp. 299–310, 2022.

[19] S. Zhao, J. Duan, S. Wu, X. Gu, C. Li, K. Yin, and H. Wang, Genetic algorithm-based SOTIF scenario construction for complex traffic flow, *Automot. Innov.*, vol. 6, no. 4, pp. 531–546, 2023.

[20] F. Fahrenkrog, S. Reithinger, B. Gülsen, and F. Raisch, European research project's contributions to a safer automated road traffic, *Automot. Innov.*, vol. 6, no. 4, pp. 521–530, 2023.

[21] D. Nalic, T. Mihalj, M. Baumler, M. Lehmann, A. Eichberger, and S. Bernsteiner, Scenario based testing of

automated driving systems: A literature survey, in *Proc. FISITA Web Congress 2000*, 2020, p. 1. https://www.fisita.org/library/f2020-acm-096.

[22] Z. Yang, Y. Chen, J. Wang, S. Manivasagam, W. C. Ma, A. J. Yang, and R. Urtasun, UniSim: A neural closed-loop sensor simulator, in *Proc. 2023 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Vancouver, Canada, 2023, pp. 1389–1399.

[23] W. Li, C. W. Pan, R. Zhang, J. P. Ren, Y. X. Ma, J. Fang, F. L. Yan, Q. C. Geng, X. Y. Huang, H. J. Gong, et al., AADS: Augmented autonomous driving simulation using data-driven algorithms, *Sci. Robot*., vol. 4, no. 28, p. eaaw0863, 2019.

[24] S. Tan, K. Wong, S. Wang, S. Manivasagam, M. Ren, and R. Urtasun, Scenegen: Learning to generate realistic traffic scenes, in *Proc. 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Nashville, TN, USA, 2021, pp. 892–901.

[25] X. Yan, Z. Zou, S. Feng, H. Zhu, H. Sun, and H. X. Liu, Learning naturalistic driving environment with statistical realism, *Nat. Commun.*, vol. 14, no. 1, p. 2037, 2023.

[26] W. Ding, Y. Cao, D. Zhao, C. Xiao, and M. Pavone, RealGen: Retrieval augmented generation for controllable traffic scenarios, in *Proc. 18th European Conf. Computer Vision*, Milan, Italy, 2024, pp. 93–110.

[27] M. Klischat and M. Althoff, Generating critical test scenarios for automated vehicles with evolutionary algorithms, in *Proc. 2019 IEEE Intelligent Vehicles Symp*., Paris, France, 2019, pp. 2352–2358.

[28] S. Shiroshita, S. Maruyama, D. Nishiyama, M. Y. Castro, K. Hamzaoui, G. Rosman, J. DeCastro, K. H. Lee, and A. Gaidon, Behaviorally diverse traffic simulation via reinforcement learning, in *Proc. 2020 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Las Vegas, NV, USA, 2020, pp. 2103–2110.

[29] W. Ding, H. Lin, B. Li, and D. Zhao, Semantically adversarial scene generation with explicit knowledge guidance, *IEEE Trans. Intell. Transp. Syst.*, vol. 26, no. 2, pp. 1510–1521, 2025.

[30] C. Cui, Y. Ma, X. Cao, W. Ye, Y. Zhou, K. Liang, J. Chen, J. Lu, Z. Yang, K. D. Liao, et al., A survey on multimodal large language models for autonomous driving, in *Proc. 2024 IEEE/CVF Winter Conf. Applications of Computer Vision*, Waikoloa, HI, USA, 2024, pp. 958–979.

[31] X. Tian, J. Gu, B. Li, Y. Liu, Y. Wang, Z. Zhao, K. Zhan, P. Jia, X. Lang, and H. Zhao, DriveVLM: The convergence of autonomous driving and large vision-language models, in *Proc. 8th Conf. Robot Learning*, Munich, Germany, 2024, pp. 4698–4726.

[32] Z. Xu, Y. Zhang, E. Xie, Z. Zhao, Y. Guo, K. Y. K. Wong, Z. Li, and H. Zhao, DriveGPT4: Interpretable end-to-end autonomous driving via large language model, *IEEE Robot. Autom. Lett.*, vol. 9, no. 10, pp. 8186–8193, 2024.

[33] C. Cui, Y. Ma, X. Cao, W. Ye, and Z. Wang, Drive as you speak: Enabling human-like interaction with large language models in autonomous vehicles, in *Proc. 2024 IEEE/CVF Winter Conf. Applications of Computer Vision*, Waikoloa, HI, USA, 2024, pp. 902–909.

[34] C. Chang, S. Wang, J. Zhang, J. Ge, and L. Li, LLMScenario: Large language model driven scenario generation, *IEEE Trans. Syst. Man Cybern. Syst*., vol. 54, no. 11, pp. 6581–6594, 2024.

[35] S. Li, T. Azfar, and R. Ke, ChatSUMO: Large language model for automating traffic scenario generation in simulation of urban mobility, arXiv preprint arXiv: 2409.09040, 2024.

[36] X. Li, E. Liu, T. Shen, J. Huang, and F. Y. Wang, ChatGPT-based scenario engineer: A new framework on scenario generation for trajectory prediction, *IEEE Trans. Intell. Veh.*, vol. 9, no. 3, pp. 4422–4431, 2024.

[37] J. Zhang, C. Xu, and B. Li, ChatScene: Knowledge-enabled safety-critical scenario generation for autonomous vehicles, in *Proc. 2024 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Seattle, WA, USA, 2024, pp. 15459–15469.

[38] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, Scenic: A language for scenario specification and scene generation, in *Proc. 40th ACM SIGPLAN Conf. Programming Language Design and Implementation*, Phoenix, AZ, USA, 2019, pp. 63–78.

[39] Y. Wei, Z. Wang, Y. Lu, C. Xu, C. Liu, H. Zhao, S. Chen, and Y. Wang, Editable scene simulation for autonomous driving via collaborative LLM-agents, in *Proc. 2024 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Seattle, WA, USA, 2024, pp. 15077–15087.

[40] H. Tian, K. Reddy, Y. Feng, M. Quddus, Y. Demiris, and P. Angeloudis, Enhancing autonomous vehicle training with language model integration and critical scenario generation, arXiv preprint arXiv: 2404.08570, 2024.

[41] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems, in *Proc. 21st Int. Conf. Intelligent Transportation Systems*, Maui, HI, USA, 2018, pp. 2118–2125.

[42] H. Tian, X. Han, G. Wu, Y. Zhou, S. Li, J. Wei, D. Ye, W. Wang, and T. Zhang, An LLM-enhanced multi-objective evolutionary search for autonomous driving test scenario generation, arXiv preprint arXiv: 2406.10857v1, 2025.

[43] Z. Zhong, D. Rempe, Y. Chen, B. Ivanovic, Y. Cao, D. Xu, M. Pavone, and B. Ray, Language-guided traffic simulation via scene-level diffusion, in *Proc. 7th Conf. Robot Learning*, Atlanta, GA, USA, 2023, pp. 144–177.

[44] A. Neelakantan, T. Xu, R. Puri, A. Radford, J. M. Han, J. Tworek, Q. Yuan, N. Tezak, J. W. Kim, C. Hallacy, et al., Text and code embeddings by contrastive pre-training, arXiv preprint arXiv: 2201.10005, 2022.

[45] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in *Proc. 36th Int. Conf. Neural Information Processing Systems*, New Orleans, LA, USA, p. 1800, 2022.

[46] OpenStreetMap Foundation, OpenStreetMap Foundation, https://osmfoundation.org/, 2024.

[47] GraphML File Format, The GraphML file format, http://graphml.graphdrawing.org/, 2024.
[48] A. Hagberg, P. J. Swart, and D. A. Schult, Exploring network structure, dynamics, and function using NetworKX, https://www.osti.gov/biblio/960616,no.LA-UR-08-05495, 2024.
[49] MATSim, The multi-agent transport simulation MATSim, https://library.oapen.org/handle/20.500.12657/32162, 2024.
[50] H. He, S. Li, J. Yang, L. He, Y. Zhang, Q. Lu, and S. Feng, KnowMoformer: Knowledge-conditioned motion transformer for controllable traffic scenario simulation, in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2024. https://agents4ad.github.io/assets/cvpr2024/posters/11.pdf.
[51] NHTSA, NHTSA crash viewer, https://crashviewer.nhtsa.dot.gov/, 2023.
[52] D. Fu, W. Lei, L. Wen, P. Cai, S. Mao, M. Dou, B. Shi, and Y. Qiao, LimSim++: A closed-loop platform for deploying multimodal LLMs in autonomous driving, in *Proc. 2024 IEEE Intelligent Vehicles Symp.*, Jeju Island, Republic of Korea, 2024, pp. 1084–1090.



Qiujing Lu received the bachelor degree in automation from Tsinghua University, Beijing, China, in 2015, and the PhD degree in electrical and computer engineering from University of California at Los Angeles, Los Angeles, CA, USA, in 2022. She is currently a postdoctoral researcher at Department of Automation, Tsinghua University, Beijing, China. Her research focuses on generative models and safe autonomous driving.



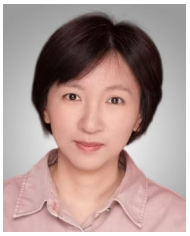
Meng Ma received the bachelor degree in control engineering from Nankai University, Tianjin, China, in 2020, and Master degree from University of California, Irvine, CA, USA, in 2024. He is currently a PhD student at Rensselaer Polytechnic Institute, Troy, NY, USA. His research interest focuses on scenario generation in autonomous driving.



Zheng Wang received the PhD degree in computer application technology from East China Normal University, Shanghai, China, in 2012. After graduation, he worked at China Aerospace, Beijing, China, where he was successively engaged in spacecraft software development and spacecraft system design. During his tenure, he won the First Prize of Beijing Municipal Science and Technology Progress Award and the First Prize of National Defense Science and Technology Progress Award. In 2021, he joined Beijing DiDi Infinity Technology and Development Co., Ltd., Beijing, China, undertaking work related to robotaxi system design and safety assessment.



Wei Lu received the PhD degree in transportation engineering and computational science from The University of Tennessee, Knoxville, TN, USA, in 2013. He has worked at the Oak Ridge National Lab, Oak Ridge, TN, USA, and multiple private companies in the autonomous vehicle industry. He works in Beijing DiDi Infinity Technology and Development Co., Ltd., Beijing, China. His interests concentrate on AI and robotics safety, simulation, and evaluation.



Mingyue Ma received the PhD degree in transportation engineering from Beihang University, Beijing, China, in 2014. She is currently an associate researcher at Research Institute for Road Safety of the Ministry of Public Security, Beijing, China. Her research interests include testing technology for compliance of autonomous driving with traffic regulations, autonomous vehicle safety verification, and vehicle information security.



Guangming Zhao received the PhD degree in mechanical engineering from Wuhan University of Technology, Wuhan, China, in 2013. He is currently a researcher at the Research Institute for Road Safety of the Ministry of Public Security, Beijing, China. He has obtained eight invention patents, drafted ten vehicle management polices and formulated six technical standards. Currently he serves as an expert reviewer of Ministry of Public Security for the two national intelligent connected vehicle pilot programs: market access & road operation and “vehicle-road-cloud integration” applications. Related research achievements have been widely applied to in road traffic safety practices. His research focuses on compliance testing for AVs with traffic rules.



Shuo Feng received the bachelor and PhD degrees from Tsinghua University, Beijing, China, in 2014 and 2019, respectively. He was a postdoctoral researcher and also an assistant researcher at Department of Civil and Environmental Engineering, the University of Michigan Transportation Research Institute (UMTRI), University of Michigan, Ann Arbor, MI, USA. He is currently an associate professor at Department of Automation, Tsinghua University, Beijing, China. His research interests lie in the development and validation of safety-critical machine learning, particularly for connected and automated vehicles. He is a member of the IEEE. He was a recipient of the Best PhD Dissertation Award from the IEEE Intelligent Transportation Systems Society in 2020 and the ITS Best Paper Award from the INFORMS TSL society in 2021. He is an associate editor of *IEEE Transactions on Intelligent Vehicles* and an academic editor of *Automotive Innovation*.