vehicles

MDPI

Article

RKGRScen: Road-Network Knowledge Graph Retrieval-Based Scenario Generation for Autonomous Driving Testing

Zhe Wang [1], Yaqing Shi [1,*], Tongtong Bai [1,*], Song Huang [1], Changyou Zheng [1], Kui Yao [2], Kunyuan Li [1] and Yao He [3]

[1] College of Command and Control Engineering, Army Engineering University of PLA, Nanjing 210007, China; 18945870795@163.com (Z.W.); huangsong@aeu.edu.cn (S.H.); zhengchy@aeu.edu.cn (C.Z.); stephanielee@aeu.edu.cn (K.L.)
[2] Department of Foundation Studies, Army Engineering University of PLA, Nanjing 211101, China; yaokuinju@aeu.edu.cn
[3] Research and Development Department, Sichuan Koma Smart Mobility Technology Co., Ltd., Suining 629001, China; m19980841101@163.com
* Correspondence: shiyaqing@aeu.edu.cn (Y.S.); baitongtong@aeu.edu.cn (T.B.)

Abstract

Simulation-based scenario testing is central to the safety validation of autonomous driving systems (ADS), where abstract scenarios must be instantiated into executable concrete scenarios. A key challenge in this process is road matching, which aims to identify a concrete site on a map that genuinely satisfies the road-semantic requirements of a scenario. Existing methods often reduce this step to template-based mapping or coarse geometric filtering, which capture coarse road types but fail to meet the finer structural and topological constraints of a scenario, thereby undermining the executability and behavioral fidelity of the generated scenarios. To address this, we propose RKGRScen (Road-Network Knowledge Graph Retrieval-Based Scenario Generation), which formalizes road matching as a semantic location retrieval problem over a road-network knowledge graph. Its indexing module constructs a road-network knowledge graph from OpenDRIVE maps and partitions it into topologically coherent communities annotated with LLM-generated semantic summaries and violation tags, organizing them into a searchable semantic community index. Its instantiation module then grounds each scenario onto concrete road sites through two-level global-to-local retrieval and resolves executable parameters and conflict-point timing with a constraint solver. In a scenario-quality evaluation on a CARLA Town01–Town05 map pool comprising 2649 scenarios, RKGRScen achieves an executability rate of 93.88%, an end-to-end behavior reproduction rate of 72.59%, and a road–environment matching rate of 93.43%. Therefore, RKGRScen can reliably ground high-level scenarios into executable scenarios that satisfy complex road and topological constraints, providing effective support for the safety validation of autonomous driving systems.

Keywords: autonomous driving testing; violation scenario generation; road-network knowledge graph

Check for updates

Received: 30 July 2026
Revised: 15 September 2026
Accepted: 15 September 2026
Published: 24 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

With the continuous development of core technologies such as artificial intelligence, sensor fusion, and high-precision localization, automated driving systems (ADS) have become an important research direction in both the automotive industry and computer science. Safety validation is a prerequisite for the large-scale deployment of ADS. Since

real-world road testing is costly, time-consuming, and struggles to cover rare safety-critical scenarios at scale, simulation-based scenario testing has become a mainstream approach for evaluating ADS behavior. The safety standard ISO 21448 [1], also known as Safety of the Intended Functionality (SOTIF), further emphasizes the need to systematically identify and reproduce safety-critical scenarios.

Existing studies commonly adopt the scenario hierarchy proposed by Menzel et al. [2], which classifies scenarios into three abstraction levels: functional scenarios, logical scenarios, and concrete scenarios. A functional scenario describes a driving situation in natural language. A logical scenario further specifies road requirements, relationships among traffic participants, behavioral objectives, and parameter ranges without fixing map locations or exact parameter values. A concrete scenario assigns deterministic values to all parameters, making the scenario executable in a simulator. In instantiating scenarios into concrete scenarios, existing methods primarily focus on parameter sampling and search [3–7], which assign or optimize continuous parameters such as participant speeds, positions, and trigger timings on a given road structure to generate numerous executable variants. However, these methods generally assume that the road structure is already given and simplify map selection to a lookup process based on road types, thereby overlooking the real complexity of road requirements in scenarios. Such requirements often involve not only a straight road segment or an intersection but also specific lane configurations, topological connections, and combinations of multiple road elements, necessitating the retrieval of road structures in real road networks whose structural and topological properties precisely satisfy the specified constraints. Existing studies that specifically address scenario grounding [8–10] often reduce this step to template-based mapping or coarse geometric filtering, which cannot perform semantic-level location retrieval over the graph structure of road networks and thus fail to satisfy position-level, multi-hop topological, and compound road constraints. This road-matching step has long been insufficiently addressed, directly limiting the executability and behavioral fidelity of generated scenarios. Moreover, even when a suitable road location is successfully identified, it remains necessary to further solve for participant spawn coordinates, speeds, and interaction timings to actually trigger the target violation behavior, yet existing methods likewise lack effective mechanisms for this parameter-solving step.

To address this gap, we propose RKGRScen, a Road-Network Knowledge Graph Retrieval-Based Scenario Generation framework that formulates road matching as semantic location retrieval over a road-network knowledge graph and integrates Leiden community detection [11], structured retrieval, and constraint solving into a logical-to-concrete pipeline. RKGRScen extracts structured logical scenarios from accident reports and constructs a road-network knowledge graph from OpenDRIVE maps. The graph is partitioned into topologically coherent communities annotated with LLM-generated semantic summaries and violation tags. Scenario grounding then proceeds through global community retrieval followed by local subgraph matching against road and topological constraints. Finally, a constraint solver determines participant positions, speeds, lane assignments, and interaction timing, and the generated scenarios are evaluated using a rule-based detector and Apollo [12].

The main contributions of this paper are summarized as follows:

- We propose a road-network indexing method that constructs a knowledge graph from OpenDRIVE maps, partitions it into topologically coherent communities, and annotates the communities with LLM-generated semantic summaries and violation tags to support location-level retrieval.
- We propose a scenario instantiation method that expands logical scenarios into structured road constraints, retrieves concrete positions through global-to-local graph matching, and solves executable participant parameters and interaction timing through constraint solving.

- We evaluate RKGRScen on 2649 scenarios over CARLA Town01–Town05 and further assess cross-stack simulation transfer using Apollo 8.0 as the full-stack ADS under test.

The remainder of this paper is organized as follows. Section 2 introduces the background on scenario-based ADS testing and road matching in scenario instantiation. Section 3 presents the RKGRScen methodology in detail. Section 4 describes the experimental design, including the research questions and the evaluation setup. Section 5 reports and analyzes the experimental results. Section 6 discusses the findings and limitations. Section 7 concludes the paper and outlines future work.

2. Background

2.1. Scenario-Based ADS Testing

Simulation-based scenario testing has been widely used to validate the safety of automated driving systems (ADS). Existing scenario generation methods can be broadly divided into two categories. Search-based methods aim to expose hazardous interactions by optimizing simulation parameters or behavioral features. AV-FUZZER uses a genetic algorithm to fuzz the behavioral parameters of non-player characters (NPCs) [3]. AutoFuzz formulates scenario generation as a constrained optimization problem [4]. BehAVExplor discovers adversarial driving scenarios through behavior-feature-guided search [5]. Data-driven methods reconstruct or generalize safety-critical scenarios from real-world accident data or driving trajectories. AC3R mines scenario structures from accident reports [13]. CRISCO reconstructs high-risk scenarios using real-world trajectory data [14]. DeepCrashTest generates test scenarios from collision videos through deep learning [15]. These methods have made important progress in improving scenario realism and risk exposure capability. However, their optimization objectives are usually defined in terms of collision probability or risk indicators.

Existing scenario generation methods mainly focus on scenario mining, reconstruction, and parameter-space search. Most studies assume that the road environment is predefined and concentrate on parameter-level sampling and optimization. However, instantiating a logical scenario into a concrete scenario involves not only assigning parameter values, but also identifying concrete locations in a real road network whose structural and topological properties satisfy the logical requirements. Existing methods still show clear limitations in the road-matching stage.

2.2. Road Matching in Scenario Instantiation

LeGEND proposes a top-down scenario generation framework that uses a large language model (LLM) to progressively decompose functional scenarios into logical-scenario DSLs and then applies NSGA-II multi-objective search to optimize participant parameters on a fixed map [8]. Based on ChatScene, ARISE retrieves scenario fragments using textual similarity and combines them with Scenic geometric filters to ground scenarios on maps. It also introduces a test-repair loop to improve scenario execution success rates [16]. TARGET designs a dedicated DSL to describe traffic-rule scenarios and generates executable test cases through predefined road structures [10]. Scenic, as a probabilistic scenario description language, supports map-based position sampling through geometric condition filters and has been used as a grounding mechanism by multiple methods [17]. AVUnit designs a domain-specific language to define scenario structures and execution conditions and supports mapping scenarios to multiple simulators [18].

Although these methods have achieved important progress in scenario instantiation, the map-grounding step is often simplified as template mapping based on road types or coarse-grained geometric filtering. As a result, they cannot perform semantic location retrieval over the graph structure of road networks, making it difficult to satisfy location-level, multi-hop topological, and compound road constraints. The lack of road-matching

capability directly limits the executability and behavioral fidelity of generated scenarios. In contrast, RKGRScen formalizes road matching as a semantic location retrieval problem over a road-network knowledge graph. Rather than assigning a logical scenario to a predefined road type or relying only on flat text similarity, RKGRScen constructs a road-network graph from OpenDRIVE maps and partitions the graph into topologically coherent semantic communities through community detection. It then uses an LLM to summarize community-level road semantics and to enrich the road and interaction constraints of logical scenarios. During instantiation, RKGRScen retrieves and locates concrete positions satisfying topological and compound road constraints through a global-to-local search process over the graph structure, and finally employs a constraint solver to generate parameter configurations that satisfy both physical feasibility and behavioral reproducibility.

3. Methodology

3.1. Overview

Figure 1 presents the overall architecture of RKGRScen. The framework first constructs a semantic community index from OpenDRIVE maps, then extracts and expands logical scenarios from accident reports, retrieves constraint-compliant road locations, and solves executable scenario parameters. The same underlying LLM supports three task-specific modules. LLM_{index} annotates road-network communities, $\text{LLM}_{extract}$ extracts logical scenarios, and LLM_{expand} converts road requirements into graph-queryable constraints.

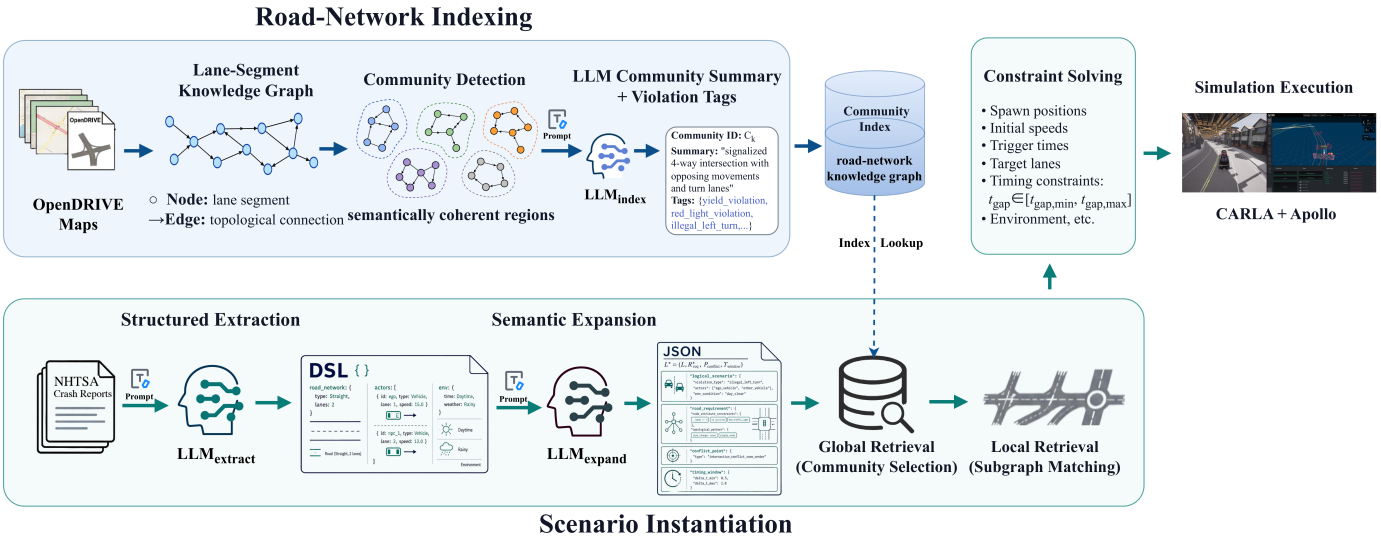


Figure 1. Overview of the RKGRScen framework.

3.2. Road-Network Indexing

3.2.1. Road-Network Graph Extraction

RKGRScen constructs a deterministic road-network knowledge graph from OpenDRIVE maps. CARLA waypoints are sampled at fixed 20-m intervals along drivable lanes and grouped by their `(road_id, section_id, lane_id)` triplets into contiguous lane-segment nodes. Node attributes capture lane configuration, road geometry, traffic-rule information, junction membership, and lane-change permissions. Edges follow waypoint successor/predecessor relations and junction connectivity and are labeled as forward or junction connections. Formally, the graph is $G = (V, E, \mathcal{F})$, where each node $v \in V$ has attribute vector $\mathcal{F}_V(v)$ and each edge $e = (v_i, v_j) \in E$ has type $\mathcal{F}_E(e) \in \{\text{forward}, \text{junction}\}$. Table 1 summarizes the resulting graph and community statistics for Town01–Town05.

Table 1. Road-network graph construction and community detection statistics for Town01–Town05. Community size denotes the number of lane-segment nodes contained in a community.

Map	Nodes $\|V\|$	Edges $\|E\|$	Communities m	Max Size	Min Size	Avg Size
Town01	124	129	12	23	6	10.3
Town02	88	97	10	19	5	8.8
Town03	402	445	17	83	6	23.6
Town04	461	445	14	170	9	32.9
Town05	486	510	22	82	8	22.1

3.2.2. Community Detection and Semantic Partitioning

To organize the road-network graph for efficient retrieval, RKGRScen partitions each graph into topologically coherent communities using the Leiden algorithm [11]. Leiden combines local moving and refinement phases while optimizing modularity. Before community detection, the directed road graph is converted to an undirected graph, and edge weights are recomputed to reflect topological affinity based on shared road, section, lane, junction, lane-count, and edge-type information:

$$w_{ij} = \alpha_0 + \alpha_{road}\mathbf{1}_{road}(i,j) + \alpha_{section}\mathbf{1}_{section}(i,j) + \alpha_{lane}\mathbf{1}_{lane}(i,j) + \alpha_{junction}\mathbf{1}_{junction}(i,j) + \alpha_{count}\mathbf{1}_{lane_count}(i,j) + \alpha_{type}\mathbf{1}_{edge_type}(e_{ij}), \quad (1)$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function, taking value 1 when the condition is satisfied and 0 otherwise, α_0 is the base connectivity weight, and α_{road}, $\alpha_{section}$, α_{lane}, $\alpha_{junction}$, α_{count}, $\alpha_{type} \geq 0$ control the affinity contributions of shared road, section, and lane identity, junction membership, lane-count consistency, and edge type, respectively. In all experiments these coefficients are fixed to $\alpha_0 = 1.0$, $\alpha_{road} = 2.5$, $\alpha_{junction} = 1.5$, $\alpha_{section} = \alpha_{lane} = 1.0$, and $\alpha_{count} = \alpha_{type} = 0.5$, applied uniformly across all five maps. The largest weights are assigned to shared road identity and joint junction membership, so that lane segments belonging to the same road or forming a common junction are more likely to be grouped into the same community, while the remaining terms provide finer affinity adjustments; these values are part of the released configuration.

Because Leiden operates on the weighted graph, community quality is evaluated using weighted modularity

$$Q = \frac{1}{2W}\sum_{i,j}\left[w_{ij} - \frac{s_i s_j}{2W}\right]\delta(c_i, c_j), \quad (2)$$

where $s_i = \sum_j w_{ij}$ is the weighted strength of node v_i, $W = \frac{1}{2}\sum_{i,j} w_{ij}$ is the total edge weight, c_i is the community assignment, and $w_{ij} = 0$ for non-adjacent nodes. Leiden iteratively optimizes this objective until convergence.

For each community C_i, a structural statistics vector $\mathbf{s}_i$ is computed deterministically, comprising node and junction counts, traffic-light and shoulder statistics, the maximum lane count, and derived ratios and indicators such as the junction ratio and a complex-intersection indicator, which together characterize the road composition of the community.

To organize the partitioned communities into a semantically searchable index, RKGRScen employs LLM_{index} to generate natural-language summaries and applicable-violation tags for each community. The community's structural statistics and derived features are formatted into a structured text prompt, which is provided to LLM_{index} along with the predefined set of seven violation types: inattention to the road ahead, failure to yield, wrong-way driving, failure to maintain safe following distance, illegal lane change, illegal overtaking, and speeding. LLM_{index} is constrained to output strictly formatted JSON containing two fields: a natural-language summary summary_i describing the road characteristics of the community, and a list

of applicable violation types $\text{tags}_i \subseteq \mathcal{V}$ selected exclusively from the predefined set $\mathcal{V}$. If the LLM output is malformed or contains illegal tags, the request is retried up to two times, and persistent failures are logged and excluded from the index. The summary text is then encoded into an embedding vector $\mathbf{e}_i \in \mathbb{R}^d$ using a sentence-embedding model:

$$\mathbf{e}_i = \text{Embed}(\text{summary}_i), \tag{3}$$

where d is the embedding dimension. The final community index table $\mathcal{I}$ contains, for each community, its identifier, structural statistics $\mathbf{s}_i$, semantic summary summary_i, applicable-violation tags tags_i, embedding vector $\mathbf{e}_i$, and the list of node identifiers nodes_i:

$$\mathcal{I} = \{(i, \mathbf{s}_i, \text{summary}_i, \text{tags}_i, \mathbf{e}_i, \text{nodes}_i) \mid C_i \in \mathcal{C}\}, \tag{4}$$

where $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ is the community set. This index is serialized to JSON files and serves as the substrate for global-level semantic retrieval during scenario instantiation.

3.3. Scenario Instantiation

3.3.1. Structured Extraction from Functional Scenarios

RKGRScen begins scenario instantiation from NHTSA crash narratives recorded between 2006 and 2015. Reports are retained when they contain a complete pre-crash description, involve one of the seven target violation types, and describe a road environment compatible with the available CARLA map structures. After initial filtering, 2052 candidate reports entered the structured extraction stage. Forty-seven reports (2.3%) failed schema validation after all allowed retries and were excluded, leaving 2005 successfully extracted logical scenarios that form the NHTSA-derived portion of the frozen benchmark.

$\text{LLM}_{extract}$ is prompted to extract accident reports into structured logical scenarios L. RKGRScen adopts the scenario description DSL proposed by TARGET [10] as the target extraction format, which provides a structured grammar to capture scenario elements including environment conditions, road network characteristics, actor configurations, and behavioral oracles. The logical scenario L is formalized as a four-tuple $L = (V_{type}, \mathcal{A}, R_{req}, E_{cond})$. Here $V_{type} \in \mathcal{V}$ is the target violation type mapped to one of the seven predefined categories; $\mathcal{A} = \{a_1, \ldots, a_n\}$ is the set of actors, each specified by a triple $\langle \text{type}, \text{behavior}, \text{position} \rangle$ with behaviors extracted from the narrative such as go forward or change lane; R_{req} is the coarse road requirement describing road types, traffic signs, and road markers at the accident location; and E_{cond} specifies the weather and time conditions mentioned in the report.

The extraction prompt is designed to translate free-form descriptions into standardized DSL fields while preserving causal relationships and temporal sequences from the narrative. $\text{LLM}_{extract}$ is constrained to output strictly formatted JSON conforming to a predefined schema that enforces field types, value ranges, and cross-field consistency rules. Malformed outputs or schema violations trigger up to three retries. Reports that still fail schema validation after all allowed retries are recorded as persistent extraction failures and excluded from the benchmark, as quantified above. This structured extraction ensures that logical scenarios retain safety-critical elements from NHTSA functional descriptions while organizing them into a machine-readable format suitable for subsequent semantic expansion.

3.3.2. Semantic Expansion

To enable graph-based road retrieval, LLM_{expand} transforms the coarse road description R_{req} into a structured constraint specification R^*_{req}. Road descriptions such as a four-lane intersection cannot be directly evaluated against graph attributes and topology and are therefore converted into explicit attribute and structural constraints.

The expanded specification is $L^* = (L, R^*_{req}, P_{conflict}, T_{window})$. Here, R^*_{req} contains node-attribute and topological-pattern constraints, $P_{conflict}$ specifies the geometric conflict location, and T_{window} defines the admissible arrival-time difference for interaction violations.

Node attribute constraints directly correspond to attribute fields of nodes in the road-network graph, including lane count lower bound $n_{lanes} \geq k$, junction membership is_junction $=$ true, traffic-light control has_traffic_light $=$ true, lane-change permission direction allows_lane_change $\in \{\text{left}, \text{right}, \text{both}, \text{none}\}$, and opposing-lane existence has_opposing_lane $=$ true; for example, a four-lane intersection requires $n_{lanes} \geq 4 \wedge \text{is_junction} = \text{true}$.

Topological pattern constraints describe the connection relationships of nodes in the graph, including single-node patterns suitable for intersection conflict scenarios, multi-hop sequence patterns, such as a curved exit to straight road to intersection, requiring a 3-hop path, and merge-point patterns requiring two lanes merging into the same node. $P_{conflict}$ specifies the geometric location type where the violation behavior occurs, such as a lane merge point for illegal lane change, an intersection conflict-zone center for failure to yield, or a front-hazard position for inattention to the road ahead, providing a reference point for distance calculation and arrival timing in subsequent constraint solving.

For interaction violations, semantic expansion derives a scenario-specific timing constraint window $T_{window} = [\Delta t_{min}, \Delta t_{max}]$ defining the hazardous range of the signed arrival-time difference $\Delta t = t_{ego} - t_{npc}$ between the ego and the interacting NPC at the conflict point, where a positive value means the ego arrives later than the NPC and a negative value means the ego arrives earlier. For example, in a failure-to-yield scenario, $T_{window} = [0.5, 2.0]$ seconds requires the ego to arrive at the conflict point 0.5 to 2.0 s later than the NPC. LLM_{expand} is guided by violation-type-specific mapping rules, with the prompt including natural-language-to-graph mapping examples, such as T-junction with traffic lights mapping to junction $=$ true $\wedge$ traffic_light $=$ true. LLM_{expand} is constrained to output structured queries conforming to a fixed JSON schema, prohibiting generation of attribute fields that do not exist in the graph. If the output contains fields that cannot be mapped to the graph schema, the system provides the graph attribute dictionary as reference and requests regeneration; if retry fails, the system falls back to a default query template based on the violation type.

3.3.3. Two-Level Global-to-Local Retrieval

Given L^*, RKGRScen retrieves concrete road locations in two stages. Global retrieval ranks semantically relevant communities using the community index $\mathcal{I}$, whereas local retrieval performs deterministic attribute and subgraph-pattern matching within the selected communities. The first stage reduces the search space, while the second verifies the exact road attributes and topology required by R^*_{req}.

The global retrieval level filters and ranks communities in the index $\mathcal{I}$. First, RKGRScen applies a tag filter that retains only communities whose applicable-violation tags contain the target violation type, i.e., $V_{type} \in \text{tags}_i$, discarding regions that are structurally unsuitable for the target violation. For each retained community C_i, the expanded specification L^* is converted into a canonical textual representation through a deterministic serialization function $\phi(\cdot)$, which renders the node attributes, topological pattern, conflict-point type, and interaction requirements in a fixed field order. This text is encoded into a query embedding $\mathbf{q} = \text{Embed}(\phi(L^*))$ using the same sentence-embedding model as in indexing, and its semantic relevance to the community is scored by the cosine similarity between the query embedding and the community summary embedding $\mathbf{e}_i$:

$$\text{sim}(\mathbf{q}, \mathbf{e}_i) = \frac{\mathbf{q} \cdot \mathbf{e}_i}{\|\mathbf{q}\| \|\mathbf{e}_i\|}. \tag{5}$$

The communities are ranked by $\mathrm{sim}(\mathbf{q}, \mathbf{e}_i)$ in descending order, and the top-K communities are retained as the candidate region set $\mathcal{C}_{cand} = \{C_{(1)}, C_{(2)}, \ldots, C_{(K)}\}$.

The local retrieval level performs subgraph matching within the candidate regions to identify concrete positions satisfying R^*_{req}. For each community $C_{(k)} \in \mathcal{C}_{cand}$, RKGRScen examines every node $v \in C_{(k)}$ and its topological neighborhood against two kinds of constraints in R^*_{req}. The node attribute constraints are verified directly on the attribute vector $\mathcal{F}_V(v)$, checking predicates such as the lane-count lower bound, junction membership, traffic-light control, lane-change permission, and opposing-lane existence; a node is discarded as soon as any attribute predicate fails. The topological pattern constraints are verified over the neighborhood of v according to the pattern type: a single-node pattern is satisfied by v itself, a multi-hop sequence pattern is verified by searching for a path of the required length whose nodes successively satisfy the per-hop attribute predicates, and a merge-point pattern is verified by checking whether two distinct predecessors connect into the same node. For single-node patterns the match is an anchor node, whereas for multi-hop and merge-point patterns the match is a connected subgraph whose nodes jointly satisfy the per-hop and convergence constraints; in all cases the anchor node of the match is recorded in the candidate location set. A node v is added to the candidate location set $\mathcal{L}$ only when both its attribute constraints and the associated topological pattern are satisfied:

$$\mathcal{L} = \{v \mid v \in C_{(k)},\ C_{(k)} \in \mathcal{C}_{cand},\ \mathrm{Attr}(v) \wedge \mathrm{Topo}(v)\}, \tag{6}$$

where $\mathrm{Attr}(v)$ denotes satisfaction of all node-attribute constraints and $\mathrm{Topo}(v)$ denotes satisfaction of the required topological pattern. Each candidate location retains its community retrieval score, and $\mathcal{L}$ is passed to the constraint solver in descending relevance order.

3.3.4. Constraint Solving and Scenario Instantiation

Given the candidate location set $\mathcal{L}$, the constraint solver determines executable parameters for each actor $a_i \in \mathcal{A}$, including spawn position pos_i, initial speed v_i, and heading θ_i. For interaction violations with a defined timing window T_{window}, the expected arrival time t_i at the conflict point $P_{conflict}$ is additionally considered. Speeds and candidate spawn positions are discretized for tractable enumeration. When the timing constraint is active, arrival time is computed from path distance and speed:

$$t_i = \frac{\mathrm{dist}(\mathrm{pos}_i, P_{conflict})}{v_i}, \tag{7}$$

where $\mathrm{dist}(\mathrm{pos}_i, P_{conflict})$ is the path distance along the actor's travel route accumulated over the waypoint sequence.

Let $\mathcal{C}(L^*) = \{\mathrm{C1}, \mathrm{C2}, \mathrm{C4}\} \cup \{\mathrm{C3} \mid T_{window} \neq \varnothing\}$ denote the active hard-constraint set for a scenario. The parameter solving is formulated as a constraint-satisfaction problem with a soft objective:

$$\Theta^* = \arg\min_{\Theta} f_{soft}(\Theta) \quad \text{s.t.} \quad \Theta \models \mathcal{C}(L^*), \tag{8}$$

where Θ always contains $\{\mathrm{pos}_i, v_i, \theta_i \mid a_i \in \mathcal{A}\}$ and additionally contains the relevant arrival-time variables t_i when $T_{window} \neq \varnothing$. The hard constraints are

$$\text{(C1)} \quad d(\mathrm{pos}_i, \mathrm{pos}_j) \geq d_{min}, \quad \forall i \neq j, \tag{9}$$

$$\text{(C2)} \quad \mathrm{pos}_i \in \mathrm{Lane}_{legal}(\ell), \quad \forall i, \tag{10}$$

$$\text{(C3)} \quad \Delta t = t_{ego} - t_{npc} \in T_{window} = [\Delta t_{min}, \Delta t_{max}], \quad \text{if } T_{window} \neq \varnothing, \tag{11}$$

$$\text{(C4)} \quad v_i \in [v_{min}, v_{max}], \quad \forall i, \tag{12}$$

Here, d_{min} is the minimum spawn distance, $\text{Lane}_{legal}(\ell)$ denotes the legal waypoints at location ℓ, and $\Delta t = t_{ego} - t_{npc}$ is the signed arrival-time difference at the conflict point. C1 prevents actor overlap, C2 enforces legal spawn lanes, C3 applies only when T_{window} is defined, and C4 bounds feasible speeds.

Algorithm 1 summarizes the procedure. Candidate locations are examined in retrieval order, infeasible spawn and speed assignments are discarded, and the feasible assignment minimizing f_{soft} is retained. The solver returns the first location with a feasible solution and reports failure if no candidate is feasible.

Algorithm 1: Constraint-Based Scenario Parameter Solving

Input: Candidate location set $\mathcal{L}$, expanded specification L^*, actor set $\mathcal{A}$
Output: Executable concrete scenario S or $\varnothing$
foreach *location* $\ell \in \mathcal{L}$ *(ranked by retrieval score)* **do**
  $\mathcal{W}_{spawn} \leftarrow$ candidate spawn waypoints derived from ℓ;
  $\Theta_{best} \leftarrow \varnothing$; $f_{best} \leftarrow +\infty$;
  foreach *spawn assignment* $\{pos_i\} \subseteq \mathcal{W}_{spawn}$ **do**
    if $\exists i \neq j : d(pos_i, pos_j) < d_{min}$ **then**
      continue ; // violates C1
    if $\exists i : pos_i \notin Lane_{legal}(\ell)$ **then**
      continue ; // violates C2
    foreach *speed assignment* $\{v_i\}$, $v_i \in [v_{min}, v_{max}]$ *(discretized)* **do**
      $\Theta_{cand} \leftarrow \{\text{pos}_i, v_i, \theta_i \mid a_i \in \mathcal{A}\}$;
      if $T_{window} \neq \varnothing$ **then**
        Compute $t_i = \dfrac{\text{dist}(\text{pos}_i, P_{conflict})}{v_i}$ for the ego and interacting NPC;
        $\Delta t \leftarrow t_{ego} - t_{npc}$;
        if $\Delta t \notin T_{window}$ **then**
          continue ; // violates C3
        $\Theta_{cand} \leftarrow \Theta_{cand} \cup \{t_i\}$;
        $f \leftarrow \left| \Delta t - \dfrac{\Delta t_{min} + \Delta t_{max}}{2} \right|$;
      else
        $f \leftarrow 0$;
      if $f < f_{best}$ **then**
        $f_{best} \leftarrow f$; $\Theta_{best} \leftarrow \Theta_{cand}$;
  if $\Theta_{best} \neq \varnothing$ **then**
    return $S = (\ell, \Theta_{best})$;
return $\varnothing$; // instantiation failed

The soft objective is likewise applied only when an interaction-timing window is defined. For such scenarios, feasible parameter assignments closer to the center of the timing window are preferred; for scenarios without T_{window}, all feasible assignments receive zero timing cost and the deterministic enumeration order resolves ties:

$$f_{soft}(\Theta) = \begin{cases} \left| \Delta t - \dfrac{\Delta t_{min} + \Delta t_{max}}{2} \right|, & T_{window} \neq \varnothing, \\ 0, & T_{window} = \varnothing. \end{cases} \tag{13}$$

3.4. Simulation Execution

The concrete scenario $S = (\ell, \Theta^*)$ is translated into a CARLA-executable configuration containing the ego vehicle, NPC vehicles, conflict point, environmental settings, and

expected violation parameters. The solver determines the ego spawn state and destination, while driving decisions remain under the ADS under test. NPC trajectories and timing follow the solved configuration, and CARLA configures weather, lighting, traffic signals, and static obstacles.

During execution, RKGRScen records complete trajectory frames every 0.2 s at approximately 5 Hz. A CARLA health check and spawn-point precheck precede execution, and all actors are cleaned up afterwards. All selected scenarios remain in the experimental accounting. In RQ1 and RQ2, an incomplete run is recorded as $\mathrm{exec}(s) = 0$. In RQ3, failures before scenario logic begins are recorded as pre-execution aborts and remain in the all-scenario ER denominator but not the ER_{att} denominator. Runs that begin scenario execution but fail to complete are recorded as post-attempt execution failures. Completed traces are evaluated by the unified violation detector and semantic match score defined in Section 4.1.2.

4. Experiments

We implement RKGRScen and use CARLA [19] as the primary scenario execution and testing environment, with Apollo 8.0 introduced for validation as a full-stack open-source ADS evaluated in co-simulation. To comprehensively evaluate the effectiveness of RKGRScen, we design experiments around three research questions:

RQ1. *How accurately can RKGRScen match functional scenarios to road-network structures and generate executable violation scenarios?*

RQ2. *How does each component of RKGRScen, including community detection, semantic expansion, and constraint solving, contribute to the overall scenario generation quality?*

RQ3. *How effectively can the scenarios generated by RKGRScen trigger target violations on a full-stack ADS in co-simulation?*

4.1. Experiment Settings

4.1.1. Experimental Environment

We conducted experiments on Ubuntu 20.04.6 with an AMD Ryzen 9 3950X CPU, 62 GB RAM, and an NVIDIA RTX 3090 GPU. Simulation experiments were conducted in CARLA 0.9.13 using the Town01 through Town05 maps, which together provide diverse road structures including straight roads, curves, intersections, T-junctions, roundabouts, and ramps. For RQ1 and RQ2, we use the built-in Traffic Manager as the ADS under test. For RQ3, we additionally deploy Apollo 8.0 as a full-stack ADS under test in co-simulation, connected to CARLA 0.9.13 through the CARLA–Apollo Bridge, which enables Apollo to control the ego vehicle in CARLA during scenario testing, while CARLA controls the NPC background vehicles and environmental elements such as weather, traffic lights, and static obstacles. All three LLM modules, LLM_{index}, $\mathrm{LLM}_{extract}$, and LLM_{expand}, use DeepSeek-V3 [20] (version 2024-12-25) with task-specific JSON schemas. The decoding parameters are temperature 0.2, top-p 0.9, and a requested random seed of 42 where supported by the hosted API. Hosted-API decoding is not assumed to be bit-reproducible across independent executions. Community summaries and serialized road-requirement queries are encoded using all-MiniLM-L6-v2 [21] with embedding dimension $d = 384$. Across the full frozen benchmark of 2649 scenarios, semantic expansion completed normally for 2523 scenarios, while 126 scenarios (4.76%) invoked the violation-type-specific fallback query template.

4.1.2. Evaluation Metrics

Let $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$ denote the evaluated scenario set with $|\mathcal{S}| = N$. For each scenario $s \in \mathcal{S}$, we introduce three binary predicates that characterize its outcome: $\mathrm{exec}(s) = 1$

if s is instantiated and executed to completion in the simulator without runtime failure; $\mathrm{trig}(s) = 1$ if the target violation is reproduced in the execution trace as judged by the unified detector; and $\mathrm{match}(s) = 1$ if the grounded road site of s satisfies all road-environment constraints of R^*_{req}. Building on these predicates, we define four metrics that assess road matching, executability, and behavioral fidelity from complementary perspectives.

To assess whether a generated concrete scenario is syntactically and physically valid enough to run to completion, we define the executability rate (ER) as the fraction of scenarios that execute without runtime failure:

$$\mathrm{ER} = \frac{1}{N} \sum_{s \in \mathcal{S}} \mathbf{1}[\mathrm{exec}(s)], \tag{14}$$

where $\mathbf{1}[\cdot]$ is the indicator function. A higher ER indicates that more scenarios are grounded into executable configurations that survive simulation without spawn failures, physical infeasibility, or runtime crashes.

Unless otherwise specified, the denominator of ER is the entire evaluated scenario set $\mathcal{S}$. In RQ1 and RQ2, which are conducted in the CARLA-only environment, this denominator coincides with the selected scenario set, and unsuccessful completed runs are recorded as $\mathrm{exec}(s) = 0$. In the RQ3 Apollo–CARLA co-simulation, 49 of the 1260 sampled runs do not enter scenario execution. These pre-execution aborts are primarily caused by occasional Apollo-module crashes during prolonged co-simulation operation, with a small number attributable to other setup-level failures. The affected runs comprise 10, 18, and 21 cases in Batches 1–3, respectively. We therefore report ER over all sampled scenarios as the primary metric and additionally report ER_{att} over the 1211 runs that successfully enter scenario execution.

To assess the extent to which the generated scenarios elicit the intended violation behavior, we define the behavior reproduction rate (BRR). Since a scenario can fail either by not executing or by executing without triggering the violation, we decouple end-to-end capability from trigger quality through two complementary variants, an unconditional rate over the full set and a rate conditioned on successful execution:

$$\mathrm{BRR}_{all} = \frac{\sum_{s \in \mathcal{S}} \mathbf{1}[\mathrm{exec}(s) \wedge \mathrm{trig}(s)]}{N}, \qquad \mathrm{BRR}_{exec} = \frac{\sum_{s \in \mathcal{S}} \mathbf{1}[\mathrm{exec}(s) \wedge \mathrm{trig}(s)]}{\sum_{s \in \mathcal{S}} \mathbf{1}[\mathrm{exec}(s)]}. \tag{15}$$

BRR_{all} characterizes the end-to-end capability of transforming a logical scenario into an executable scenario that reproduces the target behavior, whereas BRR_{exec} isolates the trigger quality among the scenarios that are already executable.

To assess the accuracy of the road-matching step itself, independently of downstream execution and triggering, we define the road-environment matching rate (REM) as the fraction of scenarios whose grounded site satisfies every road-environment constraint of R^*_{req}. Writing $R^*_{req} = \{r_1, \ldots, r_{|R^*_{req}|}\}$ for the set of node-attribute predicates and topological-pattern constraints of scenario s, and $\ell(s)$ for its grounded site, REM is

$$\mathrm{REM} = \frac{1}{N} \sum_{s \in \mathcal{S}} \mathbf{1}\Big[\bigwedge_{r \in R^*_{req}(s)} r\big(\ell(s)\big)\Big], \tag{16}$$

where $r(\ell(s)) = 1$ iff the grounded site satisfies constraint r, so a scenario contributes to REM only when all road constraints are jointly satisfied. REM is evaluated at the grounding stage before simulation by checking R^*_{req} against the static OpenDRIVE-derived graph attributes and topology of $\ell(s)$. It is therefore independent of $\mathrm{exec}(s)$ and $\mathrm{trig}(s)$ and isolates road-matching accuracy from downstream execution and behavior reproduction.

Because the structured requirement R^*_{req} is used both to retrieve candidate road sites and to judge matching correctness in Equation (16), REM could in principle be self-confirming. To rule this out, we validated REM against an independently annotated

road-matching ground truth on a stratified subset of 350 scenarios (50 per violation type). The same three experts who conducted the extraction audit judged, working only from the original functional-scenario description and a visualization of the grounded segment and without access to R^*_{req} or the system's REM decision, whether each grounded site was correctly matched. On this subset the system-side REM is 93.7% (328/350) and the independent human ground truth is 92.0% (322/350), differing by only 1.7 percentage points and both consistent with the full-benchmark REM of 93.43%; the two decisions agree on 98.3% of scenarios (Cohen's $\kappa = 0.87$), with disagreements limited to six false positives and no false negatives. Since this assessment never sees R^*_{req}, REM tracks an independent human judgment rather than merely re-confirming its own retrieval constraints. The full protocol and per-type results are given in Appendix B.

To quantify how faithfully a generated concrete scenario preserves the semantics of the input functional scenario, we define the semantic match score (MS) as a weighted aggregation of eight binary check items spanning the semantic, topological, role, and behavioral levels:

$$\mathrm{MS} = \sum_{m \in \mathcal{M}} \lambda_m c_m, \qquad \lambda_m \geq 0, \qquad \sum_{m \in \mathcal{M}} \lambda_m = 1, \tag{17}$$

where $\mathcal{M} = \{road, env, role, lane, action, auto, noctrl, context\}$ indexes the eight items, each $c_m \in \{0, 1\}$: road-type match (c_{road}), environmental-condition match (c_{env}), actor–role match (c_{role}), lane-geometry match (c_{lane}), action–intent match (c_{action}), whether the ego is under autonomous control (c_{auto}), whether there is no direct manual control of the ego (c_{noctrl}), and whether community semantic context is available ($c_{context}$). The weights λ_m are determined by the Analytic Hierarchy Process (AHP) [22]: the eight items are pairwise compared according to their relative importance to scenario fidelity, and the priority vector is derived from the principal eigenvector of the resulting comparison matrix after passing the consistency test (consistency ratio $CR < 0.1$). The pairwise comparisons were elicited from three researchers in autonomous-driving testing, the same panel that conducted the benchmark extraction audit, and the resulting matrix passed the consistency check with $CR \approx 0.001$; the full comparison matrix, the derived priority vector, and a sensitivity analysis under alternative weightings are provided in Appendix A. The obtained weights are $\lambda_{road} = \lambda_{lane} = 0.18$, $\lambda_{role} = 0.16$, $\lambda_{action} = \lambda_{auto} = 0.12$, $\lambda_{env} = \lambda_{noctrl} = 0.10$, and $\lambda_{context} = 0.04$, assigning the largest importance to road-type and lane-geometry consistency, which most directly reflect scenario fidelity. The MS is independent of violation detection: the detector answers whether the scenario triggers the target violation, whereas the MS answers whether the generated scenario is faithful to the input functional scenario. This independence lets the experiments distinguish failure modes; for instance, a high MS with an untriggered violation indicates semantic alignment without behavioral induction, whereas a triggered violation with a low MS indicates behavioral success with semantic deviation. The MS is reported as the average over successfully executed scenarios. It is therefore a conditional fidelity measure of how faithfully an executable scenario preserves its functional semantics, and is read together with ER, BRR_{all}, and REM rather than as a standalone criterion, since end-to-end capability and road-level correctness are measured over the full benchmark by these metrics.

Violation reproduction, i.e., the predicate $\mathrm{trig}(s)$, is judged by a unified rule-based detector applied uniformly to the seven violation types. The detector operates on the frame-level execution trace, where each frame carries the ego state, the NPC states, the conflict point, and lane and signal information, enabling the assessment of multi-actor interactions rather than single-vehicle behavior alone. It combines threshold constraints with temporal and sequential constraints, scanning the trace frame by frame to verify the

triggering conditions of the target violation. The detector thresholds are fixed judgment criteria applied uniformly across all scenarios and are therefore distinct from the scenario-specific timing window T_{window} used in constraint solving: the former decides whether an executed trajectory constitutes a violation, whereas the latter guides parameter solving toward a critical configuration. The basic physical quantities are the time-to-collision $\mathrm{TTC} = d/v$, where d is the distance from the actor to the target point and v its speed; the time headway $\mathrm{THW} = \mathrm{gap}/v_{ego}$, where gap is the longitudinal distance between the ego and the lead vehicle; and the responsibility-sensitive safety (RSS) distance:

$$d_{RSS} = v_{ego}\rho + \frac{1}{2}a_{max}\rho^2 + \frac{(v_{ego} + a_{max}\rho)^2}{2b_{min}} - \frac{v_{lead}^2}{2b_{max}}, \tag{18}$$

where ρ is the response time, a_{max} the maximum acceleration, and b_{min}, b_{max} the minimum braking deceleration of the ego and the maximum braking deceleration of the lead vehicle, respectively. In all experiments these parameters are fixed to $\rho = 1.0$ s, $a_{max} = 2.0$ m/s^2, $b_{min} = 4.0$ m/s^2, and $b_{max} = 8.0$ m/s^2, consistent with the values used in the released detector implementation. The trigger conditions and key thresholds of the seven violation types are summarized in Table 2. For RQ1 and RQ2, we report 95% percentile bootstrap confidence intervals using 10,000 scenario-level resamples for the two primary metrics BRR_{all} and REM. When two methods or configurations are evaluated on the same scenarios, between-method differences are assessed using paired bootstrap resampling, in which the paired scenario outcomes are resampled jointly. McNemar's exact test is additionally applied to the small illegal-overtaking category. ER, BRR_{exec}, and MS are otherwise reported as point estimates in RQ1 and RQ2. For RQ3, we additionally report pooled 95% bootstrap confidence intervals for ER, BRR_{exec}, REM, and MS. All bootstrap analyses use 10,000 resamples with a fixed statistical resampling seed of 42. These intervals quantify scenario-level sampling uncertainty conditional on the recorded pipeline outputs and do not incorporate additional run-to-run variability arising from hosted-LLM decoding or simulator/co-simulation execution. Percentage-point differences are computed from the underlying integer counts or unrounded scenario-level rates and are rounded only after subtraction; therefore, they may differ by 0.01 percentage points from differences computed directly from the displayed rounded percentages.

Table 2. Trigger conditions and key thresholds of the seven violation types.

Violation Type	Trigger Condition	Key Thresholds
Speeding	Ego speed exceeds the limit for $\geq n_c$ consecutive frames	$n_c = 3$
Wrong-way driving	Small inter-vehicle distance, opposing heading, both moving	$\delta_w = 12.5$ m, $\psi_{op} = 120^\circ$, $v_\epsilon = 1.0$ m/s
Failure to maintain safe following distance	$\mathrm{THW} \leq \tau_h \wedge \mathrm{gap} \leq \delta_f$, or $\mathrm{gap} < d_{RSS}$	$\tau_h = 1.0$ s, $\delta_f = 8.0$ m
Inattention to the road ahead	Front target within hazardous TTC/RSS window while ego advances	$\delta_a = 15.0$ m, $\tau_t = 3.0$ s
Failure to yield	Arrival-time difference to conflict point within window and small gap	$[0.5, 2.0]$ s, $\delta_y = 8.0$ m
Illegal lane change	Source/target lane differ, direction match, entered target lane, dangerous gap	$\delta_l = 8.0$ m
Illegal overtaking	NPC passes behind $\prec$ changing $\prec$ ahead with sustained small gap	$\delta_b = \delta_h = 3.0$ m, $\delta_o = 10.0$ m

4.1.3. Benchmark Extraction Audit

Because the 2005 NHTSA-derived scenarios are produced by $LLM_{extract}$ and the 644 synthetic scenarios are deterministically derived from them, we conduct a human audit to verify the quality of the original extraction. From the 2005 NHTSA-derived scenarios we draw a stratified sample of 350 scenarios, with 50 scenarios per violation type. Because the NHTSA-derived benchmark contains only 52 illegal-overtaking scenarios and 55 illegal-lane-change scenarios, the 50-scenario audit covers 96.2% and 90.9% of these two strata, respectively. These two audit rows should therefore be interpreted as near-census assessments rather than small subsamples of the corresponding categories. The sampled report identifiers are fixed by a random seed and released in the repository. Three experts in autonomous-driving testing independently compare each extracted scenario against its source narrative and judge three fields, namely the violation type, the actor behavior, and the road requirement. The violation type is a single label scored as correct or incorrect, whereas the actor behavior and the road requirement are scored on a three-level scale of correct, partial, and incorrect, where partial denotes correct primary content with a missing or deviating secondary attribute. The three experts independently evaluate the finalized extracted scenarios against their source narratives, without access to one another's judgments or to the LLM's intermediate generations. The final ground truth for each field is decided by majority vote among the three experts. Per-field extraction accuracy is reported as the proportion of correct judgments over the individual expert labels (150 per violation type and 1050 in total). Majority-vote labels over the 50 scenarios per violation type are used as the adjudicated scenario-level audit outcomes and are included in the released audit records. Inter-expert agreement is quantified by Fleiss' κ. The results are summarized in Table 3.

Table 3. Human audit of $LLM_{extract}$ on 350 sampled NHTSA scenarios, 50 per violation type, judged independently by three autonomous-driving-testing experts. Accuracy is the proportion of correct judgments over the individual expert judgments (150 per row, 1050 in total), rather than over the 50 majority-vote decisions. Majority-vote labels are retained as the adjudicated scenario-level audit outcomes and are included in the released audit records. Fleiss' κ measures inter-expert agreement. VT is violation type, AB is actor behavior, RR is road requirement. The Partial column reports the actor-behavior field; the road-requirement Partial rate is given in the released audit records. The Overall/Weighted row is shown in bold.

Violation Type	n	Accuracy (%)			Fleiss' κ			Partial (%)
		VT	AB	RR	VT	AB	RR	
Inattention to the road ahead	50	97.3	92.7	90.7	0.91	0.84	0.81	5.3
Failure to yield	50	95.3	90.7	87.3	0.89	0.81	0.78	8.7
Speeding	50	98.7	93.3	91.3	0.94	0.84	0.81	4.7
Failure to maintain safe following distance	50	96.7	89.3	87.3	0.90	0.80	0.77	8.7
Wrong-way driving	50	93.3	87.3	85.3	0.86	0.79	0.76	10.7
Illegal lane change	50	92.7	87.3	83.3	0.84	0.78	0.75	11.3
Illegal overtaking	50	91.3	85.3	83.3	0.83	0.77	0.74	12.7
Overall/Weighted	**350**	**95.0**	**89.4**	**87.0**	**0.89**	**0.80**	**0.78**	**8.9**

The extraction attains an overall accuracy of 95.0% on violation type, 89.4% on actor behavior, and 87.0% on road requirement, with per-field Fleiss' κ of 0.89, 0.80, and 0.78, all indicating substantial to almost perfect inter-expert agreement. Accuracy is highest on single-vehicle categories such as speeding and inattention and slightly lower on the rarer interaction-oriented categories such as illegal overtaking and illegal lane change, whose road requirement is more compositional. In addition, $LLM_{extract}$ stayed within the allowed retries in the vast majority of cases, and its persistent-failure rate that triggers exclusion

was 2.3%. Because the same extracted logical scenarios are used by all three methods, the shared benchmark reduces input-related confounding in the relative comparison. However, different methods may exhibit different sensitivities to residual extraction errors.

4.1.4. RQ1 Settings

RQ1 is evaluated on a frozen shared benchmark containing 2649 logical scenarios. Among them, 2005 are original logical scenarios extracted from the NHTSA accident reports, and the remaining 644 are synthetic fine-grained scenarios derived from the original scenarios through deterministic topology extension. The per-type breakdown of the two sources is given in Table 4; every violation type retains at least 50 NHTSA-derived scenarios, which is what makes the stratified extraction audit of 50 scenarios per type in Section 4.1.3 feasible. These synthetic scenarios are not random perturbations. Each is deterministically constructed from a distinct qualified original scenario under a fixed seed by appending one connected road segment, turning a single-segment requirement into a two-hop topological one while preserving the actors, conflict point, and violation type; the results are deduplicated. Because appending a segment changes the road-matching requirement, a synthetic scenario and its source are grounded to different sites and thus constitute distinct test inputs rather than duplicates. The benchmark spans the seven violation types, with the distribution shown in Table 4.

Table 4. Distribution of the 2649 benchmark scenarios across the seven violation types, separated into NHTSA-derived and synthetic scenarios. The Total row is shown in bold.

Violation Type	NHTSA	Synthetic	Total
Inattention to the road ahead	646	217	863
Failure to yield	517	140	657
Speeding	349	90	439
Failure to maintain safe following distance	302	100	402
Wrong-way driving	84	60	144
Illegal lane change	55	22	77
Illegal overtaking	52	15	67
Total	**2005**	**644**	**2649**

RQ1 further partitions the benchmark into 1854 coarse and 795 fine scenarios using a deterministic structural-complexity rule applied before LLM processing. A scenario is labeled fine if it contains a multi-hop topology, at least three coupled node attributes or an equivalent compound road specification, or a conflict involving multiple role-and-lane participants. Otherwise, it is labeled coarse. This rule yields 1854 coarse and 151 fine NHTSA-derived scenarios, while all 644 topology-extended synthetic scenarios are fine by construction. Each scenario is instantiated over the CARLA Town01–Town05 candidate map pool.

To evaluate road-matching capability, RQ1 compares RKGRScen against two baselines representing the prevailing map-grounding paradigms. Template Mapping grounds each logical scenario by assigning it to a predefined road type and placing it on a matching map fragment, representing the type-lookup paradigm common in existing scenario generation methods. ARISE, built on ChatScene, retrieves matching Scenic snippets from a knowledge base by textual similarity and grounds scenarios through Scenic geometric filters with a test-and-repair loop, representing the flat text-similarity paradigm that reduces map grounding to snippet retrieval and geometric filtering. The two baselines thus contrast directly with the graph-structured semantic retrieval of RKGRScen, highlighting the contribution of road-network graph retrieval to road matching.

All three methods use the same frozen scenarios, Town01–Town05 map pool, detector, and CARLA environment. RKGRScen retains the top-3 communities and top-3 locally matched nodes and then applies its deterministic constraint solver once without a repair loop. Template Mapping assigns each violation type to a fixed map and selects a single matching node without a constraint solver. ARISE is evaluated using the original Scenic pipeline released by its authors without reimplementation. It uses top-2 snippet retrieval and the original test-and-repair procedure with at most ten repairs and five attempts. The comparison therefore controls the scenarios, detector, and simulator while preserving the native grounding procedure of each method.

RQ1 reports four metrics, namely the executability rate (ER), the behavior reproduction rate (BRR), the road-environment matching rate (REM), and the match score (MS), whose definitions are given in Section 4.1.2. On the full benchmark, RQ1 reports ER, BRR_{all}, BRR_{exec}, MS, and REM to assess the overall scenario-generation quality; on the coarse and fine subsets, RQ1 further reports BRR_{all} and REM to compare the road-matching capability of the three methods as structural complexity increases. In addition, RQ1 conducts a sensitivity analysis on the number of retained communities K in the global retrieval level, varying K over $\{1, 2, 3, 4, 5, 6\}$ to examine its effect on road matching and downstream scenario quality and to justify the default setting of RKGRScen. All methods share the same logical scenarios, the same unified detector, and the same simulation environment to ensure a fair comparison.

4.1.5. RQ2 Settings

RQ2 conducts an ablation study to quantify the independent contribution of each component of RKGRScen to scenario-generation quality. Because the distribution of the full benchmark across the seven violation types is uneven, performing ablation directly on it would allow frequent categories to dominate the aggregate metrics and mask the true contribution of each component across the full spectrum of violation types. RQ2 therefore uniformly samples 60 scenarios per violation type from the shared benchmark, forming a balanced subset of 420 scenarios that weights every violation type equally and attributes the ablation conclusions to the components themselves rather than to the data distribution. Unlike RQ1, which examines capability retention under increasing complexity, RQ2 aims to isolate the contribution of each component, an effect that is already evident in the aggregate metrics over the type-balanced subset; RQ2 therefore does not introduce difficulty stratification to avoid further partitioning the samples and weakening the statistical stability of the ablation attribution.

RQ2 compares the full RKGRScen against three ablated variants, each removing one core mechanism. The w/o Community variant drops community detection and retrieves directly over the flat node set. The w/o Expansion variant drops the LLM-based semantic expansion and matches the coarse road description R_{req} against the graph without deriving the structured constraints R^{*}_{req}. The w/o Constraint variant drops the constraint solver and assigns participant parameters heuristically. The three variants isolate the contributions of community-level retrieval organization, query-side semantic expansion, and parameter instantiation, respectively.

RQ2 reports the executability rate (ER), the average match score (MS), the road-environment matching rate (REM), and two variants of the behavior reproduction rate, BRR_{all} and BRR_{exec}, to distinguish the end-to-end reproduction capability from the trigger quality among successfully executed scenarios and to separate road-matching accuracy from downstream effects. All variants share the same balanced subset, the same unified detector, and the same CARLA Town01–Town05 simulation environment under identical conditions to ensure a fair comparison.

4.1.6. RQ3 Settings

RQ3 evaluates the cross-stack simulation transfer of the generated scenarios to Apollo 8.0, a full-stack open-source autonomous driving system evaluated in CARLA–Apollo co-simulation. Validation on physical roads or a full real-world ADS deployment is beyond the scope of this co-simulation study and is left as future work. RQ3 randomly samples a balanced subset of 420 scenarios from the shared benchmark, with 60 scenarios per violation type. To account for sampling variability, the 420-scenario subset is drawn three times. Each selected scenario is deployed in the CARLA–Apollo joint simulation, where Apollo controls the ego vehicle and receives only the start and end positions together with the heading, with its driving decisions made fully autonomously, while CARLA controls the NPC background vehicles and the environmental elements, including weather, traffic-light states, and static obstacles.

Violation judgment reuses the unified detector adopted in RQ1 and RQ2, operating on the ego trajectory returned by Apollo, so that the detection criterion remains consistent across platforms. RQ3 reports ER over all sampled scenarios, ER_{att} over the runs that enter scenario execution, BRR_{exec} over valid executions, REM, and the average MS to assess the effectiveness of the generated scenarios on a full-stack ADS in co-simulation.

5. Results

5.1. RQ1 Results

Table 5 reports the performance of RKGRScen and the two baseline methods on the full benchmark, and Table 6 reports the road-matching results on the coarse- and fine-grained subsets. On the full benchmark of 2649 scenarios, RKGRScen achieves an executability rate (ER) of 93.88%, an end-to-end behavior reproduction rate (BRR_{all}) of 72.59%, a behavior reproduction rate among successfully executed scenarios (BRR_{exec}) of 77.32%, a match score (MS) of 0.969, and a road-environment matching rate (REM) of 93.43%. Among the three methods, RKGRScen obtains the highest BRR_{all}, BRR_{exec}, MS, and REM. Compared with Template Mapping, RKGRScen improves BRR_{all} by 17.37 percentage points, MS by 0.086, and REM by 5.81 percentage points. ARISE attains the highest ER of 94.90%, exceeding RKGRScen by 1.02 percentage points; however, its BRR_{all}, BRR_{exec}, MS, and REM are lower than those of RKGRScen by 5.25 percentage points, 6.36 percentage points, 0.113, and 11.29 percentage points, respectively. This contrast confirms that a high executability rate does not imply correct road matching and that the two must be measured separately, since ARISE executes marginally more scenarios yet grounds far fewer of them onto road sites satisfying the required road-environment constraints.

Table 5. RQ1 results on the 2649-scenario benchmark. Brackets give 95% bootstrap confidence intervals for BRR_{all} and REM. The best value in each column is shown in bold.

Method	ER	BRR_{all}	BRR_{exec}	MS	REM
RKGRScen	93.88	**72.59** [70.9, 74.3]	**77.32**	**0.969**	**93.43** [92.4, 94.4]
Template Mapping	77.65	55.23 [53.3, 57.1]	71.12	0.883	87.62 [86.3, 88.9]
ARISE	**94.90**	67.35 [65.5, 69.1]	70.96	0.856	82.14 [80.7, 83.6]

Paired-bootstrap comparisons over the same 2649 scenarios confirm the advantage of RKGRScen on the two primary metrics. Relative to Template Mapping, RKGRScen improves BRR_{all} by 17.37 percentage points (95% paired-bootstrap CI [15.6, 19.1]) and REM by 5.81 points (95% CI [4.7, 6.9]). Relative to ARISE, the corresponding improvements are 5.25 points in BRR_{all} (95% CI [3.4, 7.1]) and 11.29 points in REM (95% CI [9.8, 12.8]). All four paired confidence intervals lie entirely above zero, showing that the advantages on the two primary metrics are robust to scenario-level sampling variability.

Table 6. Road-matching capability across difficulty levels, with 95% bootstrap confidence intervals below each value. The fine subset is further split by source into NHTSA-derived fine scenarios (151) and synthetic fine scenarios (644). BRR_{all} and REM are in percent; the best value in each column is in bold.

Method	Coarse (1854)		Fine: NHTSA (151)		Fine: Synthetic (644)	
	BRR_{all}	**REM**	**BRR_{all}**	**REM**	**BRR_{all}**	**REM**
RKGRScen	**74.70**	**95.79**	**72.85**	**92.72**	**66.46**	**86.80**
	[72.7, 76.7]	[94.9, 96.7]	[65.6, 80.1]	[88.1, 96.7]	[62.7, 70.0]	[84.2, 89.4]
Template Mapping	63.65	91.15	41.06	80.79	34.32	79.04
	[61.5, 65.9]	[89.8, 92.4]	[33.1, 49.0]	[74.8, 86.8]	[30.7, 37.9]	[75.8, 82.1]
ARISE	73.09	84.95	56.29	79.47	53.42	74.69
	[71.1, 75.1]	[83.3, 86.6]	[48.3, 64.2]	[72.8, 86.1]	[49.5, 57.3]	[71.3, 78.0]

The advantage of RKGRScen increases with structural complexity. On the coarse subset, RKGRScen achieves a BRR_{all} of 74.70% and REM of 95.79%. On the combined fine subset, these values are 67.67% and 87.92%, compared with 35.60% and 79.37% for Template Mapping and 53.96% and 75.60% for ARISE. The fine-subset BRR_{all} advantage therefore widens to 32.07 percentage points over Template Mapping and 13.71 points over ARISE. This trend is consistent with the explicit attribute and topology verification used by RKGRScen as road requirements become more compositional.

The same ranking holds when the fine subset is separated by source. RKGRScen achieves BRR_{all}/REM values of 72.85%/92.72% on the 151 NHTSA-derived fine scenarios and 66.46%/86.80% on the 644 synthetic fine scenarios, outperforming both baselines in each case.

To determine whether the aggregate advantage is concentrated in a small number of categories, we next examine behavior reproduction at the violation-type level. Table 7 reports BRR_{all} for all seven violation types.

Table 7. Per-violation-type BRR_{all} (%) of the three methods. The best value in each row is in bold.

Violation Type	RKGRScen	Template Mapping	ARISE
Failure to maintain safe following distance	84.08	80.85	**96.02**
Failure to yield	**70.78**	42.77	1.83
Inattention to the road ahead	81.92	80.30	**94.32**
Illegal lane change	**50.65**	15.58	19.48
Illegal overtaking	**25.37**	11.94	13.43
Wrong-way driving	56.25	87.50	**95.14**
Speeding	62.87	4.10	**93.62**

RKGRScen reproduces the target behavior in all seven violation categories and is the only method that attains a reproduction rate above 25% on every interaction-oriented violation. Its clearest advantage is on failure to yield, illegal lane change, and illegal overtaking, reaching 70.78%, 50.65%, and 25.37%, respectively, whereas Template Mapping obtains only 42.77%, 15.58%, and 11.94% and ARISE only 1.83%, 19.48%, and 13.43% on the same three categories. These three violations require the simultaneous satisfaction of conflict geometry, actor roles, lane connectivity, and temporal ordering, which flat similarity and template lookup cannot ground reliably. ARISE's especially low failure-to-yield rate is consistent with the difficulty of satisfying a signal-controlled conflict geometry together with a two-actor arrival-time ordering using snippet retrieval and geometric filtering alone. Both baselines also remain low on illegal lane change and illegal overtaking, which require appropriate multi-lane structure and coordinated relative motion. On illegal overtaking specifically, RKGRScen reproduces the target violation in 17 of 67 scenarios against 9 of 67 for ARISE; a McNemar exact test on the paired outcomes gives $p = 0.077$, so although

RKGRScen is higher in the expected direction, the difference does not reach significance at the 0.05 level, consistent with the small number of overtaking scenarios in the benchmark. Conversely, ARISE leads on failure to maintain safe following distance, inattention, wrong-way driving, and speeding, which are governed mainly by local road attributes or single-vehicle motion states and are well served by its executability-oriented repair loop.

Table 8 shows a clear improvement from $K = 1$ to $K = 3$, with BRR_{all} increasing from 62.78% to 72.59%, REM from 83.58% to 93.43%, and MS from 0.884 to 0.969. Performance then reaches a practical plateau. Relative to $K = 3$, the maximum deviations for $K = 4$–6 are 0.30 percentage points in ER, 0.75 in BRR_{all}, 0.38 in REM, and 0.005 in MS. Each K setting is based on the frozen scenario-level outputs produced by its previously completed full-pipeline run. The aggregate metrics in Table 8 were recomputed directly from these frozen records; no additional hosted-LLM or simulator reruns were performed for the numerical corrections. The small residual differences across K settings may therefore contain run-to-run variability already present in the recorded pipeline outputs and should not be interpreted as systematic degradation with increasing K. We therefore use $K = 3$, the smallest and least costly setting on this plateau.

Table 8. Effect of the number of retained communities K in the global retrieval level on RKGRScen performance in RQ1, with the local top-K fixed to 3. ER, BRR_{all}, and REM are in percent; the best value in each column is in bold.

Global-K	ER	BRR_{all}	REM	MS
1	85.13	62.78	83.58	0.884
2	91.05	70.22	90.11	0.946
3	93.88	**72.59**	93.43	**0.969**
4	**93.92**	72.44	**93.58**	0.968
5	93.77	72.10	93.28	0.966
6	93.58	71.84	93.05	0.964

We additionally replace DeepSeek-V3 with Qwen2.5-72B-Instruct for all three LLM modules on the 420-scenario RQ2 subset while keeping the remaining pipeline fixed. As shown in Table 9, the two backbones differ by at most 2.02 percentage points across ER, BRR_{all}, BRR_{exec}, and REM, and by 0.0121 in MS. REM remains 87.86% with Qwen2.5-72B-Instruct compared with 89.52% with DeepSeek-V3, indicating limited sensitivity to the backbone model.

Table 9. LLM sensitivity analysis on the balanced subset of 420 scenarios. Only the backbone of the three LLM invocations is changed, with all other components fixed. ER, BRR_{all}, BRR_{exec}, and REM are in percent.

Backbone LLM	ER	BRR_{all}	BRR_{exec}	MS	REM
DeepSeek-V3	88.57	68.81	77.69	0.9113	89.52
Qwen2.5-72B-Instruct	89.05	67.38	75.67	0.8992	87.86

Overall, the RQ1 results show that RKGRScen achieves the best road-environment matching, end-to-end behavior reproduction, and semantic matching performance while maintaining competitive executability. Its advantage is most pronounced on fine-grained scenarios and on interaction-oriented violations that involve complex road topology and multi-actor coordination, precisely the cases where template mapping and flat text-similarity retrieval break down.

5.2. RQ2 Results

Table 10 reports the ablation results on the balanced subset of 420 scenarios. The complete RKGRScen configuration achieves an ER of 88.57%, a BRR_{all} of 68.81%, a BRR_{exec} of 77.69%, an MS of 0.9113, and an REM of 89.52%, outperforming all three ablated variants on every metric.

Table 10. Ablation results on the balanced subset of 420 scenarios, with 95% bootstrap confidence intervals (10,000 resamples) shown below BRR_{all} and REM. ER, BRR_{all}, BRR_{exec}, and REM are in percent; the best value in each column is in bold.

Configuration	ER	BRR_{all}	BRR_{exec}	MS	REM
Full RKGRScen	**88.57**	**68.81** [64.3, 73.1]	**77.69**	**0.9113**	**89.52** [86.4, 92.4]
w/o Community	50.71	31.67 [27.1, 36.2]	62.44	0.8162	69.76 [65.5, 74.0]
w/o Expansion	48.81	29.52 [25.2, 34.0]	60.49	0.7793	81.43 [77.6, 85.0]
w/o Constraint	49.76	23.33 [19.3, 27.4]	46.89	0.7298	82.86 [79.3, 86.4]

The absolute results in Table 10 quantify the performance loss caused by each ablation. To make the relative degradation patterns easier to compare across metrics, Figure 2 visualizes the decrease from the full configuration for the same three variants. The figure complements the absolute values and confidence intervals in Table 10 rather than introducing an additional evaluation metric.

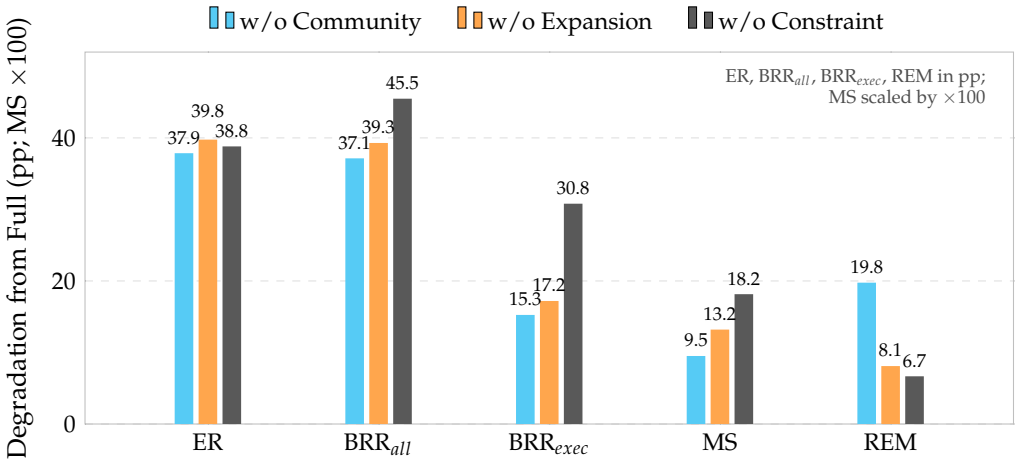


Figure 2. Per-metric performance degradation of the ablated variants relative to the full RKGRScen. MS degradation is reported in points (×100) for comparability.

Removing community retrieval lowers ER to 50.71%, BRR_{all} to 31.67%, BRR_{exec} to 62.44%, MS to 0.8162, and REM to 69.76%, corresponding to drops of 37.86, 37.14, and 15.25 percentage points in the three rates, 0.0951 in MS, and 19.76 percentage points in REM. This is the largest REM drop among the three variants because, without community-level organization, retrieval degenerates into a flat node search that can no longer locate semantically relevant regions, grounding far more scenarios onto sites that violate the road-environment constraints.

Removing semantic expansion yields an ER of 48.81%, a BRR_{all} of 29.52%, a BRR_{exec} of 60.49%, an MS of 0.7793, and an REM of 81.43%. It causes the largest ER drop of 39.76 percentage points, together with drops of 39.29 and 17.20 percentage points in BRR_{all} and BRR_{exec}, 0.1320 in MS, and 8.10 percentage points in REM. The pronounced ER drop shows that, without expanding R_{req} into the structured constraints R^{*}_{req}, the query cannot be matched precisely against graph node attributes, and the imprecise grounding propagates into lower executability.

Removing constraint solving yields an ER of 49.76%, a BRR_{all} of 23.33%, a BRR_{exec} of 46.89%, an MS of 0.7298, and an REM of 82.86%. It causes the largest drops in behavior reproduction and semantic matching, namely 45.48 percentage points in BRR_{all}, 30.80 in BRR_{exec}, and 0.1815 in MS, while ER falls by 38.81 percentage points. Its REM drops by only 6.67 percentage points, the smallest among the three variants, which matches the design of RKGRScen, where road matching precedes parameter solving. Thus, removing the solver leaves the matched location largely intact, while the heuristic parameter assignment fails to turn it into coordinated positions, speeds, and timings, causing behavior reproduction to collapse.

Overall, the RQ2 results show that community retrieval, semantic expansion, and constraint solving all contribute substantially to the performance of RKGRScen. Community retrieval has the largest effect on realized road-environment matching (REM), semantic expansion has the largest effect on executability (ER), and constraint solving has the largest effect on behavior reproduction and semantic matching, confirming that the three components act on complementary stages of the pipeline. Paired-bootstrap comparisons over the same 420 scenarios further confirm the contribution of all three components. Removing community retrieval reduces BRR_{all} by 37.14 percentage points (95% paired-bootstrap CI [31.9, 42.4]) and REM by 19.76 points (95% CI [14.5, 25.2]). Removing semantic expansion reduces BRR_{all} by 39.29 points (95% CI [33.8, 44.8]) and REM by 8.10 points (95% CI [3.8, 12.4]). Removing constraint solving reduces BRR_{all} by 45.48 points (95% CI [40.2, 50.7]) and REM by 6.67 points (95% CI [2.6, 10.7]). All six paired confidence intervals lie entirely above zero, demonstrating that the degradation associated with each ablation is robust to scenario-level sampling variability.

We further examine semantic-expansion fallback behavior over the full frozen benchmark. Of the 2649 scenarios, 2523 complete normal semantic expansion and 126 scenarios (4.76%) invoke the violation-type-specific fallback query template. As shown in Table 11, the normal-expansion group achieves an ER of 94.13%, a BRR_{all} of 73.13%, a BRR_{exec} of 77.68%, an MS of 0.972, and an REM of 93.82%. The fallback group achieves an ER of 88.89%, a BRR_{all} of 61.90%, a BRR_{exec} of 69.64%, an MS of 0.905, and an REM of 85.71%. The lower values of the fallback group across all five metrics indicate that the fallback mechanism preserves pipeline continuity but does not fully retain the quality of successful semantic expansion.

Table 11. Performance of normal semantic expansion and fallback expansion on the full 2649-scenario benchmark. MS is averaged over successfully executed scenarios. ER, BRR_{all}, BRR_{exec}, and REM are in percent.

Expansion Mode	N	ER	BRR_{all}	BRR_{exec}	MS	REM
Normal expansion	2523	94.13	73.13	77.68	0.972	93.82
Fallback	126	88.89	61.90	69.64	0.905	85.71
Overall Full	2649	93.88	72.59	77.32	0.969	93.43

We additionally examine the cases in which semantic expansion falls back to the default query template. On the balanced RQ2 subset, 20 of 420 scenarios (4.76%) invoke the fallback mechanism. As shown in Table 12, the fallback subset achieves an ER of 80.00%, a BRR_{all} of 40.00%, a BRR_{exec} of 50.00%, an MS of 0.870, and an REM of 80.00%, all below the corresponding values of the normally expanded subset. The degradation is consistent in direction with the w/o Expansion ablation, although the magnitudes are not directly comparable because the fallback subset contains only 20 cases that specifically failed normal semantic expansion, whereas the ablation is evaluated over all 420 scenarios. The subset MS values are rounded for presentation, whereas the overall MS is computed directly from

the unrounded scenario-level records. These results indicate that the fallback mechanism provides a conservative recovery path rather than silently preserving the quality of normal semantic expansion.

Table 12. Performance of normal semantic expansion and fallback expansion on the balanced RQ2 subset. MS is averaged over successfully executed scenarios. ER, BRR_{all}, BRR_{exec}, and REM are in percent.

Expansion Mode	N	ER	BRR_{all}	BRR_{exec}	MS	REM
Normal expansion	400	89.00	70.25	78.93	0.913	90.00
Fallback	20	80.00	40.00	50.00	0.870	80.00
Overall Full	420	88.57	68.81	77.69	0.9113	89.52

5.3. RQ3 Results

Table 13 reports the three Apollo–CARLA co-simulation batches. Across the three batches, an average of approximately 404 scenarios per batch enter execution, 364 produce valid executions, and 186 reproduce the target violation. When pooled, the evaluation contains 1260 sampled scenarios, including 1211 attempted runs, 1091 valid executions, and 557 detected violations. Forty-nine of the 1260 sampled runs do not enter scenario execution. These pre-execution aborts are primarily caused by occasional Apollo-module crashes during prolonged co-simulation operation, with a small number attributable to other setup-level failures; they comprise 10, 18, and 21 runs in Batches 1–3, respectively.

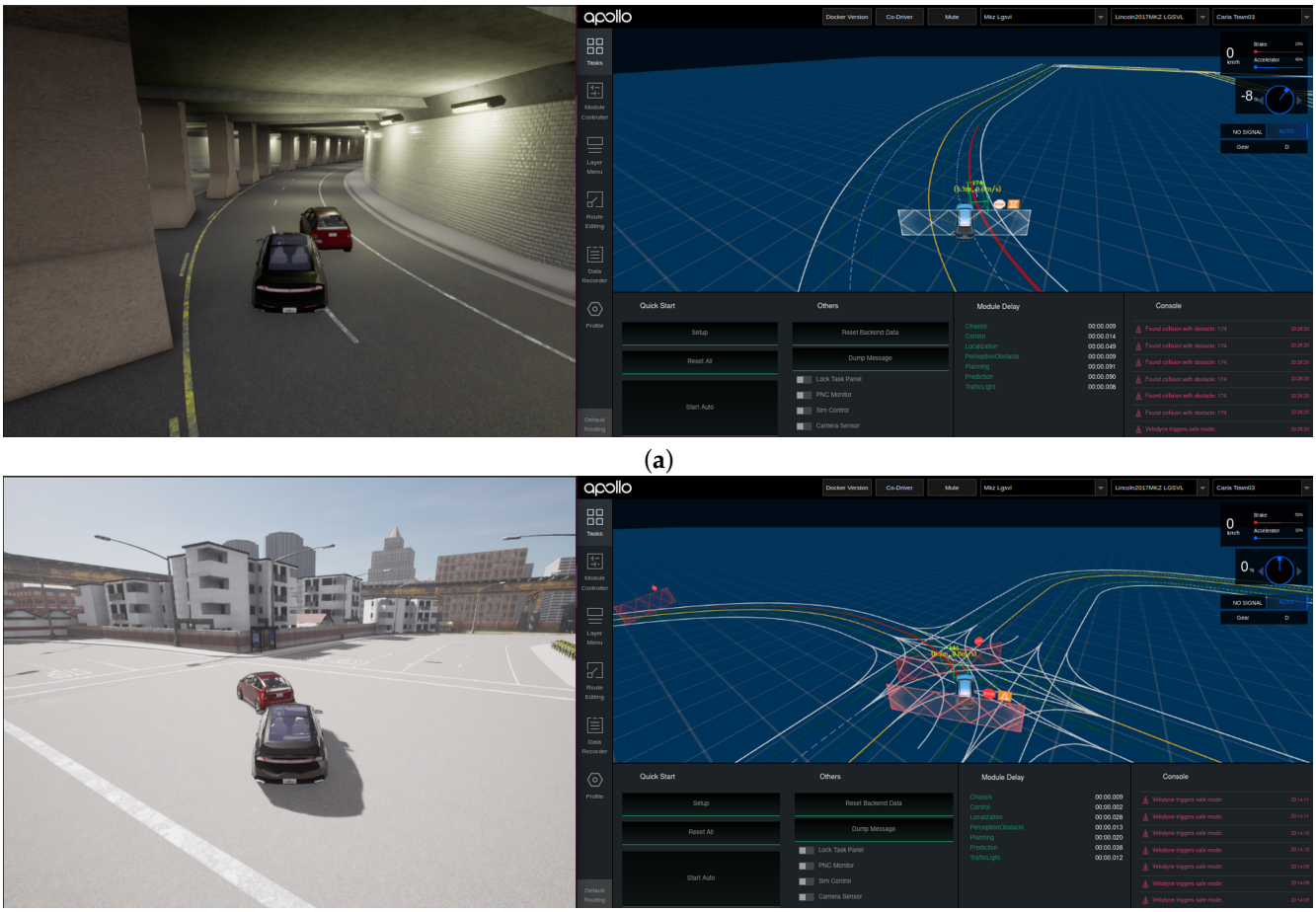
Table 13. RQ3 results on Apollo 8.0. Three balanced batches of 420 scenarios are evaluated. The Pooled column aggregates all 1260 sampled runs; 95% bootstrap confidence intervals for the pooled estimates are shown in brackets. MS is averaged over successfully executed scenarios; other rates are in percent. The pooled estimates are shown in bold.

Metric	Pooled (95% CI)	Batch 1	Batch 2	Batch 3
Total sampled scenarios	1260	420	420	420
Pre-execution aborts	49	10	18	21
Attempted scenarios	1211	410	402	399
Post-attempt execution failures	120	38	42	40
Valid executions	1091	372	360	359
Detected violations	557	179	192	186
Road-matched scenarios	1101	363	367	371
ER (%)	**86.59** [84.7, 88.4]	88.57	85.71	85.48
ER_{att} (%)	**90.09**	90.73	89.55	89.97
BRR_{exec} (%)	**51.05** [48.1, 54.0]	48.12	53.33	51.81
REM (%)	**87.38** [85.6, 89.2]	86.43	87.38	88.33
MS	**0.9023** [0.899, 0.906]	0.8972	0.9012	0.9087

The pooled all-scenario ER is 86.59%, while ER_{att} over the 1211 runs that enter scenario execution is 90.09%. Among the 1091 valid executions, 557 reproduce the target violation, yielding a pooled BRR_{exec} of 51.05%. The pooled REM is 87.38% (1101/1260), and the pooled MS is 0.9023. The pooled MS is computed directly from the 1091 unrounded scenario-level MS values rather than from the rounded batch means. Across the three batches, the largest spread among the percentage metrics is 5.21 percentage points for BRR_{exec}, while MS ranges from 0.8972 to 0.9087. The pooled bootstrap intervals in Table 13 quantify scenario-level sampling uncertainty, whereas the observed between-batch spread provides empirical context for the combined sampling and complete-pipeline variability of the co-simulation experiments.

The pooled REM of 87.38% remains close to the 89.52% obtained by the full RKGRScen configuration in the CARLA-only RQ2 evaluation, a difference of only 2.14 percentage points. In contrast, BRR_{exec} decreases from 77.69% in RQ2 to 51.05% on Apollo, a difference of 26.64 percentage points. This divergence indicates that road grounding transfers comparatively well across the two simulation stacks, whereas reproducing the intended behavioral violation is more difficult against Apollo's full-stack planning and control.

Beyond the aggregate statistics, Figure 3 illustrates two representative violation types reproduced by RKGRScen in the CARLA–Apollo co-simulation. Figure 3a shows a following-distance violation in a tunnel scenario, whereas Figure 3b shows an interaction violation at an open intersection. In both examples, Apollo autonomously plans the ego trajectory, while the generated road configuration and background-vehicle interactions establish the corresponding violation scenario.



(**a**)

(**b**)

Figure 3. Qualitative examples of RKGRScen-generated scenarios executed on Apollo 8.0. (**a**) Following-distance violation executed in a tunnel scenario; (**b**) interaction violation executed in an open intersection scenario. In the CARLA views, the black vehicle is the ego vehicle and the red vehicle is the interacting vehicle. In the Apollo views, the blue grid is the visualization background, colored curves represent road and trajectory overlays, and green boxes indicate detected obstacles. The negative steering-wheel-angle readout in the upper-right corner of panel (**a**) denotes clockwise (rightward) steering. The visualization colors do not encode quantitative experimental results.

Overall, the RQ3 results show that RKGRScen-generated scenarios transfer to a full-stack ADS in co-simulation, maintaining high executability, stable road-environment matching, and semantic consistency in the CARLA–Apollo co-simulation environment while reproducing the target violation in approximately half of the valid executions.

6. Discussion

The experiments indicate that the three main components of RKGRScen address different failure modes. Community retrieval primarily affects whether an appropriate road region is found, semantic expansion determines whether natural-language road requirements can be expressed as graph constraints, and constraint solving determines whether the grounded scenario can be instantiated into a coordinated interaction. The lower Full RKGRScen values in RQ2 reflect the type-balanced sampling, which gives greater weight to the harder interaction-oriented categories than the full RQ1 benchmark.

Cross-stack evaluation shows that road grounding is more stable than behavioral reproduction when scenarios are transferred from CARLA-only execution to Apollo–CARLA co-simulation. Successful road matching therefore does not guarantee that a full-stack planner will exhibit the intended violation. The 49 pre-execution aborts also show that co-simulation reliability affects end-to-end executability.

The current implementation samples OpenDRIVE waypoints at 20 m intervals, which may under-represent short merge lanes or tight urban geometry. Future work will investigate adaptive graph sampling, continuous or SMT-based parameter solving, richer environmental constraints, larger road networks, additional ADS platforms, and physical-road and full real-world ADS validation beyond CARLA–Apollo co-simulation.

7. Conclusions

This paper presented RKGRScen, a road-network knowledge graph retrieval framework for grounding logical driving scenarios into executable concrete scenarios. RKGRScen combines semantic community retrieval, deterministic subgraph matching, and constraint-based parameter solving. On 2649 scenarios, it achieves an ER of 93.88%, BRR_{all} of 72.59%, and REM of 93.43%, with its largest advantages observed on scenarios involving compound road constraints and multi-hop topology. Ablation experiments confirm complementary contributions from community retrieval, semantic expansion, and constraint solving, while Apollo–CARLA experiments show that the generated scenarios retain strong road-matching quality under cross-stack simulation transfer. These results support graph-structured retrieval as an effective approach to scenario grounding for ADS testing.

Author Contributions: Conceptualization, Z.W., Y.S. and T.B.; methodology, Z.W. and Y.S.; validation, Z.W., K.L. and Y.H.; formal analysis, Z.W. and S.H.; investigation, Z.W. and C.Z.; data curation, Z.W. and K.L.; writing—original draft preparation, Z.W.; writing—review and editing, Y.S., T.B., S.H., C.Z., K.Y., K.L. and Y.H.; supervision, Y.S., T.B. and S.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable. The study did not recruit human subjects or collect personal or sensitive information. Expert judgments were used solely for methodological annotation and validation.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code and experimental data supporting the findings of this study are available in the public repository at https://github.com/Wz716/RKGRScen (accessed on 14 September 2026). The original accident-report data were obtained from the publicly available NHTSA crash investigation database, and the simulation maps used in this study are available in CARLA.

Acknowledgments: The authors thank the developers of CARLA and Apollo for the open platforms used in this study.

Conflicts of Interest: Author Y.H. was employed by Sichuan Koma Smart Mobility Technology Co., Ltd. The remaining authors declare no conflicts of interest.

Appendix A. AHP Weights for the Semantic Match Score

The eight MS weights were obtained from pairwise comparisons provided by the same three autonomous-driving-testing experts who conducted the extraction audit. The aggregated AHP matrix is reported in Table A1. Its principal eigenvector yields weights of 0.18 for road and lane, 0.16 for role, 0.12 for action and auto, 0.10 for env and noctrl, and 0.04 for context. The matrix satisfies the consistency criterion with $\lambda_{\max} = 8.007$, $CI = 0.001$, and $CR \approx 0.001$.

Table A1. AHP pairwise comparison matrix over the eight semantic-match items and the derived priority weights ($\lambda_{\max} = 8.007$, $CI = 0.001$, $CR \approx 0.001$). Rows and columns follow the order road, lane, role, action, auto, env, noctrl, context.

	Road	Lane	Role	Action	Auto	Env	Noctrl	Context	Weight
road	1.00	1.00	1.13	1.50	1.50	1.80	1.80	4.00	0.18
lane	1.00	1.00	1.13	1.50	1.50	1.80	1.80	4.00	0.18
role	0.89	0.89	1.00	1.33	1.33	1.60	1.60	4.00	0.16
action	0.67	0.67	0.75	1.00	1.00	1.20	1.20	3.00	0.12
auto	0.67	0.67	0.75	1.00	1.00	1.20	1.20	3.00	0.12
env	0.56	0.56	0.63	0.83	0.83	1.00	1.00	2.50	0.10
noctrl	0.56	0.56	0.63	0.83	0.83	1.00	1.00	2.50	0.10
context	0.25	0.25	0.25	0.33	0.33	0.40	0.40	1.00	0.04

To verify that the MS-based ranking does not hinge on the specific AHP weights, we recomputed MS on the full RQ1 benchmark under two alternative weightings: equal weighting ($\lambda_m = 1/8$ for all items) and a flattened scheme that reduces the spread between the largest and smallest AHP weights ($\lambda_{road} = \lambda_{lane} = 0.15$, $\lambda_{role} = 0.14$, $\lambda_{action} = \lambda_{auto} = 0.12$, $\lambda_{env} = \lambda_{noctrl} = 0.11$, $\lambda_{context} = 0.10$). As shown in Table A2, the method ranking is identical under all three schemes, with RKGRScen ranking first, Template Mapping second, and ARISE third, and each method's MS shifts by at most 0.007 between the AHP weights and equal weighting. The MS-based conclusions are therefore robust to the choice of weights.

Table A2. Sensitivity of the match score (MS) to the weighting scheme, on the full RQ1 benchmark. The method ranking is invariant across all three schemes.

Method	AHP (Default)	Equal (1/8)	Flattened
RKGRScen	0.969	0.962	0.964
Template Mapping	0.883	0.881	0.882
ARISE	0.856	0.856	0.856

Appendix B. Independent Validation of REM

REM is independently validated on 350 stratified scenarios, with 50 scenarios per violation type. The same three autonomous-driving-testing experts judge each grounded site using only the original functional-scenario description and a visualization of the grounded segment, without access to R^*_{req} or the system REM decision. Labels are match, partial, or incorrect. Partial is treated as non-match, and majority vote defines the human ground truth.

As shown in Table A3, the system-side REM on this subset is 93.7% (328/350), while the independent human ground truth marks 92.0% (322/350) of the sites as correctly matched, so the two estimates are within 1.7 percentage points and both agree with the full-benchmark REM of 93.43%. The REM decision and the human judgment coincide on 98.3% of the scenarios (344/350) with a Cohen's κ of 0.87, and the disagreements are confined to 6 false positives, where REM accepts a site the experts reject, with no false negatives. Because this external assessment never sees R^*_{req}, the high agreement indicates that REM is

not merely re-confirming its own retrieval constraints but tracks an independent human judgment of road-matching correctness.

Table A3. Independent validation of REM on 350 stratified scenarios. System REM and Human report matched sites according to RKGRScen and the three-expert majority vote, respectively. The inter-expert Fleiss' κ column quantifies agreement among the three expert annotators, whereas the Cohen's κ reported in the text quantifies agreement between the system REM decision and the majority-vote human ground truth. The Overall row is shown in bold.

Violation Type	n	System REM	Human	Agree.	Agree. (%)	Inter-Expert Fleiss' κ
Inattention to the road ahead	50	48	48	50	100.0	0.88
Failure to yield	50	47	46	49	98.0	0.81
Speeding	50	48	47	49	98.0	0.86
Failure to maintain safe following distance	50	47	46	49	98.0	0.79
Wrong-way driving	50	46	45	49	98.0	0.75
Illegal lane change	50	46	45	49	98.0	0.74
Illegal overtaking	50	46	45	49	98.0	0.72
Overall	**350**	**328**	**322**	**344**	**98.3**	**0.79**

References

1. *ISO 21448:2022*; Road Vehicles—Safety of the Intended Functionality, 1st ed. International Organization for Standardization: Geneva, Switzerland, 2022. Available online: https://www.iso.org/standard/77490.html (accessed on 14 September 2026).
2. Menzel, T.; Bagschik, G.; Maurer, M. Scenarios for Development, Test and Validation of Automated Vehicles. In *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV)*; IEEE: New York, NY, USA, 2018; pp. 1821–1827. [CrossRef]
3. Li, G.; Li, Y.; Jha, S.; Tsai, T.; Sullivan, M.; Hari, S.K.S.; Kalbarczyk, Z.; Iyer, R. AV-FUZZER: Finding Safety Violations in Autonomous Driving Systems. In *Proceedings of the 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*; IEEE: New York, NY, USA, 2020; pp. 25–36. [CrossRef]
4. Zhong, Z.; Kaiser, G.; Ray, B. Neural Network Guided Evolutionary Fuzzing for Finding Traffic Violations of Autonomous Vehicles. *IEEE Trans. Softw. Eng.* **2023**, *49*, 1860–1875. [CrossRef]
5. Cheng, M.; Zhou, Y.; Xie, X. BehAVExplor: Behavior Diversity Guided Testing for Autonomous Driving Systems. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*; Association for Computing Machinery: New York, NY, USA, 2023; pp. 488–500. [CrossRef]
6. Ji, X.; Xue, L.; He, Z.; Luo, X. Autonomous Driving System Testing via Diversity-Oriented Driving Scenario Exploration. *Acm Trans. Softw. Eng. Methodol.* **2025**, *34*, 201. [CrossRef]
7. Tian, H.; Jiang, Y.; Wu, G.; Yan, J.; Wei, J.; Chen, W.; Li, S.; Ye, D. 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*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 94–106. [CrossRef]
8. Tang, S.; Zhang, Z.; Zhou, J.; Lei, L.; Zhou, Y.; Xue, Y. LeGEND: A Top-Down Approach to Scenario Generation of Autonomous Driving Systems Assisted by Large Language Models. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*; Association for Computing Machinery: New York, NY, USA, 2024; pp. 1497–1508. [CrossRef]
9. Guo, A.; Zhou, Y.; Tian, H.; Fang, C.; Sun, Y.; Sun, W.; Gao, X.; Luu, A.T.; Liu, Y.; Chen, Z. SoVAR: Building Generalizable Scenarios from Accident Reports for Autonomous Driving Testing. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*; Association for Computing Machinery: New York, NY, USA, 2024; pp. 268–280. [CrossRef]
10. Deng, Y.; Tu, Z.; Yao, J.; Zhang, M.; Zhang, T.; Zheng, X. TARGET: Traffic Rule-Based Test Generation for Autonomous Driving via Validated LLM-Guided Knowledge Extraction. *IEEE Trans. Softw. Eng.* **2025**, *51*, 1950–1968. [CrossRef]
11. Traag, V.A.; Waltman, L.; van Eck, N.J. From Louvain to Leiden: Guaranteeing Well-Connected Communities. *Sci. Rep.* **2019**, *9*, 5233. [CrossRef] [PubMed]
12. Apollo Auto. Apollo Open Platform, Release 8.0. GitHub Release. 2022. Available online: https://github.com/ApolloAuto/apollo/releases/tag/v8.0.0 (accessed on 29 April 2026).
13. Gambi, A.; Huynh, T.; Fraser, G. Generating Effective Test Cases for Self-Driving Cars from Police Reports. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 257–267. [CrossRef]

14. Tian, H.; Wu, G.; Yan, J.; Jiang, Y.; Wei, J.; Chen, W.; Li, S.; Ye, D. Generating Critical Test Scenarios for Autonomous Driving Systems via Influential Behavior Patterns. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 1–12. [CrossRef]
15. Bashetty, S.K.; Ben Amor, H.; Fainekos, G. DeepCrashTest: Turning Dashcam Videos into Virtual Crash Tests for Automated Driving Systems. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*; IEEE: New York, NY, USA, 2020; pp. 11353–11360. [CrossRef]
16. Poddubnyy, K.; Vozniak, I.; Burmistrov, I.; Lipp, N.; Hovhannisyan, D.; Müller, C.; Slusallek, P. ARISE—Adaptive Refinement and Iterative Scenario Engineering. In *Proceedings of the 2026 IEEE Intelligent Vehicles Symposium (IV)*; IEEE: New York, NY, USA, 2026; pp. 823–830. [CrossRef]
17. Fremont, D.J.; Kim, E.; Dreossi, T.; Ghosh, S.; Yue, X.; Sangiovanni-Vincentelli, A.L.; Seshia, S.A. Scenic: A Language for Scenario Specification and Data Generation. *Mach. Learn.* **2023**, *112*, 3805–3849. [CrossRef]
18. Zhou, Y.; Sun, Y.; Tang, Y.; Chen, Y.; Sun, J.; Poskitt, C.M.; Liu, Y.; Yang, Z. Specification-Based Autonomous Driving System Testing. *IEEE Trans. Softw. Eng.* **2023**, *49*, 3391–3410. [CrossRef]
19. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An Open Urban Driving Simulator. In Proceedings of the 1st Annual Conference on Robot Learning, Mountain View, CA, USA, 13–15 November 2017; Volume 78, pp. 1–16. Available online: https://proceedings.mlr.press/v78/dosovitskiy17a.html (accessed on 14 September 2026).
20. DeepSeek-AI. DeepSeek-V3 Technical Report. *arXiv* **2024**, arXiv:2412.19437. [CrossRef]
21. Sentence Transformers. all-MiniLM-L6-v2. Hugging Face Model Card. Available online: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2 (accessed on 14 September 2026).
22. Saaty, R.W. The Analytic Hierarchy Process—What It Is and How It Is Used. *Math. Model.* **1987**, *9*, 161–176. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.