A Graph-based Framework for Coverage Analysis in Autonomous Driving

Thomas Mühlenstädt and Marius Bause

No Institute Given

1 Abstract

Coverage analysis is essential for validating the safety of autonomous driving systems, yet existing approaches typically assess coverage factors individually or in limited combinations, struggling to capture the complex interactions inherent in traffic scenes. This paper proposes a graph-based framework for coverage analysis that represents traffic scenes as hierarchical graphs, combining map topology with actor relationships. The framework introduces a two-phase graph construction algorithm that systematically captures spatial relationships between traffic participants, including leading, following, neighboring, and opposing configurations.

Two complementary coverage analysis methods are presented. First, a subgraph isomorphism approach matches traffic scenes against a set of manually defined archetype graphs representing common driving scenarios. Second, a graph embedding approach utilizes Graph Isomorphism Networks with Edge features (GINE) trained via self-supervised contrastive learning to project traffic scenes into a vector space, enabling similarity-based coverage assessment.

The framework is validated on both real-world data from the Argoverse 2.0 dataset and synthetic data from the CARLA simulator. The subgraph isomorphism method is used to calculate node coverage percentages using predefined archetypes, while the embedding approach reveals meaningful structure in the latent space suitable for clustering and anomaly detection. The proposed approach offers significant advantages over traditional methods by scaling efficiently to diverse traffic scenarios without requiring scenario-specific handling, and by naturally accommodating varying numbers of actors in a scene.

2 Introduction

In autonomous driving, coverage analysis is a crucial step to ensure the safety and reliability of the system. In most situations, coverage arguments are collected either per coverage factor, or maybe up to 2 or 3 factor interactions. See for example [7] for a production-grade implementation of state-of-the-art coverage analysis. In contrast to existing approaches, this paper proposes a graph-based framework for coverage analysis. While graph-based representations of traffic scenes already exist, they have not been specifically designed for this purpose.

This work bridges that gap by utilizing graph-based traffic scene representations explicitly for coverage analysis.

This paper is structured as follows: In chapter 3, existing coverage and analysis approaches are discussed. The basis for the following chapters is layed in chapter 4. Afterwards, two coverage approaches using the traffic scene graphs are discussed in chapter 5 and chapter 6. Finally, the developed methods are applied to Carla and Argoverse 2.0 data in chapter 7 followed by a short summary and outlook.

3 Existing coverage and analysis approaches

Foundational terminology for automated driving is established by the SAE J3016 taxonomy [13], which defines six levels of driving automation and key concepts such as the Dynamic Driving Task (DDT) and Operational Design Domain (ODD). Building on this, [17] provide precise definitions for the commonly conflated terms *scene*, *situation*, and *scenario*, enabling a consistent vocabulary for test design and coverage arguments. Complementing this work, [8] propose object-oriented ontologies for scenario description with explicit linkage to coverage arguments.

Scenario-based testing has emerged as the dominant paradigm for validating automated driving functions. The PEGASUS project [14] introduces a six-layer model to decompose the driving environment into structured logical scenarios, shifting validation from public road testing toward systematic scenario exploration across simulation and field tests. [11] refine this with three abstraction levels—functional, logical, and concrete scenarios—while noting that existing parameter-selection methods lack a systematic way to determine meaningful test coverage. Motivated by this weak point, [15] present trajectory-based clustering of real-world driving data to structure the scenario space and reduce test-set redundancy.

A central challenge is the *approval trap* identified by [18]: statistically demonstrating superior safety through real-world driving alone would require billions of test kilometres, motivating the need for simulation-based tools and systematic coverage methods. General software testing theory [1] provides foundational coverage concepts (statement, branch, condition coverage), while [7] demonstrate their domain-specific application through coverage buckets, parameterised items, and performance metrics such as time-to-collision. At the standards level, ISO 21448 [16] and UL 4600 [10] both define frameworks for coverage criteria and measurement in the context of automated driving safety.

4 Defining a traffic scene graph

4.1 Datasets

In the following, 2 datasets are used for developing and demonstrating the suggested methods.

Argoverse 2.0 [19] is a large-scale dataset for autonomous driving research, developed by Argo AI. It contains a Dataset of 250,000 scenarios of 11 seconds each, featuring tracked object trajectories for vehicles, pedestrians, cyclists, and other road users.

The dataset contains object classifications, track identities across frames, and semantic map information including lane boundaries, crosswalks, and traffic signal locations across diverse geographic locations. A subset of 17000 scenes from the train partition is used for the analysis.

CARLA (Car Learning to Act, [4]) is an open-source simulator specifically designed for autonomous driving research and development. It provides a realistic urban driving environment with diverse road layouts, weather conditions, and traffic scenarios.

Here, Carla version 0.9.15 is used, because it provides more maps than subsequent versions. Specifically, the following maps were used: Town01, Town02, Town03, Town04, Town05 and Town07.

A script has been generated to simulate multiple vehicle types including trucks, motorcycles, and regular cars with varying probabilities, each exhibiting different behavioral characteristics such as speed preferences, following distances, and lane-changing tendencies. The script incorporates dynamic behavior modifications during simulation, including random slowdowns, periodic behavior changes, and adaptive responses to traffic conditions, resulting in rich and varied traffic scene data across multiple CARLA maps and simulation iterations. The simulation runs have between 20 and 60 vehicles each.

The resulting data consists of 2050 scenes with 11 seconds of simulation time each. The actors are spread over the entire map, so one scene produces multiple connected components, if the actor groups are far apart from each other.

4.2 Map Graph Construction

The map graph is a directed multigraph $G_{\text{map}} = (V_{\text{map}}, E_{\text{map}})$ where each node $v \in V_{\text{map}}$ represents a lane segment, storing geometric information (boundaries, centerline, length) and semantic attributes (intersection status, road and lane type). Edges encode three spatial relationships between lanes: **following edges** connect lanes forming a continuous path in the same direction, **neighbor edges** connect adjacent lanes traveling in the same direction, and **opposite edges** connect lanes traveling in opposite directions. The map graph is constructed by processing map data from Argoverse or CARLA to extract these relationships. Lanes with geometric overlap are marked as intersection lanes.

4.3 Actor Graph Construction

The actor graph $G_{\text{actor}} = (V_{\text{actor}}, E_{\text{actor}})$ represents dynamic relationships between actors at a specific timestep. Each node $v \in V_{\text{actor}}$ represents an actor and stores attributes including primary lane ID, all occupied lane IDs, longitudinal position s along the lane centerline, 3D position, longitudinal speed, actor type, and a lane change indicator. Each edge $e \in E_{\text{actor}}$ stores a actor-actor relation type and a path length along the lane network.

Four relationship types are defined between actors, ordered by semantic hierarchy:

1. **Following/Leading**: Longitudinal relationships where actors share the same lane or are connected entirely by following edges.
2. **Neighbor**: Lateral relationships via paths containing exactly one neighbor edge (actors need not be on immediately adjacent lanes).
3. **Opposite**: Relationships via paths containing exactly one opposite edge (actors need not be on immediately opposite lanes).

4.4 Hierarchical Graph Construction Algorithm

The actor graph is constructed in two phases that separate relation discovery from graph building, enabling hierarchical processing and preventing redundant edges. All parameters are listed in Table 1.

Table 1: Input parameters for actor graph construction

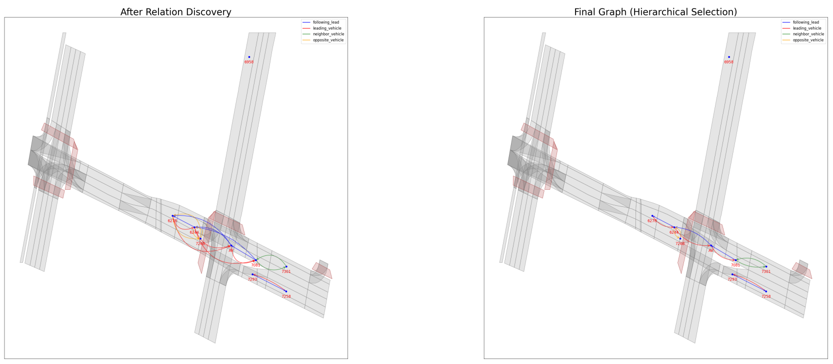
Parameter	**Type**	**Description**	**Value**
Distance limits (discovery phase)			
`max_distance_lead_veh_m`	float	Max distance for leading/following	100
`max_distance_neighbor_fwd_m`	float	Max distance for forward neighbor	50
`max_distance_neighbor_bwd_m`	float	Max distance for backward neighbor	50
`max_distance_opposite_fwd_m`	float	Max distance for forward opposite	100
`max_distance_opposite_bwd_m`	float	Max distance for backward opposite	10
Node distance limits (construction phase)			
`max_node_dist_leading`	int	Max edge count in path for leading/following	3
`max_node_dist_neighbor`	int	Max edge count in path for neighbor	2
`max_node_dist_opposite`	int	Max edge count in path for opposite	2
Timestep configuration			
`delta_timestep_s`	float	Time step increment in seconds	1.0

Phase 1: Relation Discovery: For each actor pair (A, B) at timestep t, the algorithm determines their primary lanes and checks if a connecting path exists in the map graph. The path structure determines the relationship type: paths consisting entirely of following edges yield following/leading relations, paths with exactly one neighbor (or opposite) edge yield neighbor (or opposite) relations. Both lane-based path length and Euclidean distance are checked against the thresholds in Table 1 to filter distant actors.

Phase 2: Hierarchical Graph Construction: Edges are added in hierarchical order—leading/following first (highest priority), then neighbor, then opposite—each sorted by path length (shortest first). Before adding an edge between actors A and B, a breadth-first search checks whether a path of length

$\leq$ max_node_distance already exists in the current graph. If so, the direct edge is skipped, as the relationship is already implicitly encoded. The graph is updated after each edge addition so that subsequent checks reflect the current state.

This redundancy prevention significantly reduces edges while preserving connectivity. For example, three vehicles in a row need only two lead-follow edges; the third pairwise relation is encoded through the intermediate vehicle. Similarly, for a row of vehicles adjacent to an opposite-direction vehicle, only the closest pairwise relation needs an explicit edge (see Figure 1).



(a) After relation discovery (b) After hierarchical selection

Fig. 1: The discovery graph (left) contains all relations within distance limits. Hierarchical selection (right) removes redundant edges representable through existing paths.

5 Create subgraphs for coverage analysis

Traffic scene archetypes from the literature (e.g., lead vehicle following, opposite traffic) can be expressed as small graphs. To determine whether such an archetype appears in a larger traffic scene graph G, we check if any subgraph of G is isomorphic to the archetype graph A. While subgraph isomorphism is NP-hard in general, the graphs considered here are small enough for practical computation using the VF2 algorithm [3].

Our strategy is as follows: (1) Define a set of archetype subgraphs S representing relevant traffic situations. (2) Specify which node and edge attributes are considered for the isomorphism check. (3) For each traffic scene graph G (e.g., from CARLA or Argoverse), check if any subgraph of G is isomorphic to any archetype in S and record the result in a coverage table C.

This bottom-up approach starts from individual situation archetypes and checks their presence across datasets. The resulting coverage table enables follow-up analyses such as visualizing attribute distributions (speed, distance) for matched scenes, cross-tabulating co-occurring archetypes, or computing AV performance metrics conditioned on specific archetypes.

We defined 18 subgraph archetypes covering common traffic scenarios: simple 2-actor patterns (following, opposite, neighbor—used only for isolated pairs), 3-actor patterns (lead vehicle with neighbor, platoons, opposite traffic, lane change/cut-in scenarios with intersection variants), 4-actor patterns (cut-out, multi-vehicle platoons, lead-follow with opposite traffic and intersection variants), and 5-actor patterns combining lead, neighbor, and opposite vehicles with intersection variants.

The last step in the analysis of the subgraphs is to identify coverage differences between real-world and simulation datasets. The archetypes provide an intuitive understanding of what traffic situations are missing from the simulated environment.

These concepts will be applied to CARLA and Argoverse datasets in the application chapter 7.

6 Graph Embeddings for Traffic Scene Analysis

The subgraph isomorphism approach from Section 5 provides a bottom-up methodology for analyzing traffic scenarios through predefined archetypes. However, real-world traffic scene graphs exhibit significantly higher complexity than these patterns, motivating complementary top-down approaches such as graph embeddings.

Embeddings translate raw data into a vector space where distances measure similarity, analogous to the Word2Vec model for words ([12]). For traffic scene graphs, embeddings enable coverage analysis by comparing scenes: identifying similar simulation–real-world scenario pairs, detecting near duplicates, or visualizing structures that would be intractable in the original space of all possible traffic scenes.

A Graph Isomorphism Network with Edge features (GINE) [9] is used to generate embeddings, implemented in PyTorch Geometric ([6]). GINE is chosen because it allows embedding of edge features, so we may encode actor relation type and distance as integral desciptors for a traffic scene (see Section 2).

The architecture is shown in Figure 2. Node features are actor type (one-hot: vehicle, pedestrian, cyclist, motorcycle), speed, intersection presence, and lane change since the last timestep. Edge features are edge type (one-hot, 4 spatial relationship categories) and path length between nodes.

The model was trained on CARLA and Argoverse 2.0 using self-supervised contrastive learning [2], [9]. For each mini-batch, two augmented views were created by perturbing continuous attributes with Gaussian noise ($\sigma = 0.08$) and randomly dropping edges (probability 0.1). A five-layer GINE encoder produced

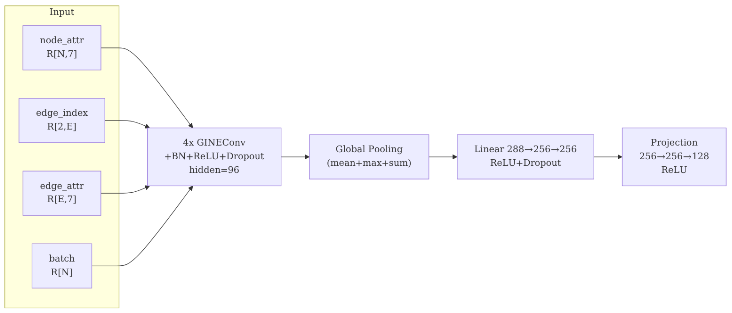

Input
node_attr R[N,7]
edge_index R[2,E]
edge_attr R[E,7]
batch R[N]
4x GINEConv +BN+ReLU+Dropout hidden=96
Global Pooling (mean+max+sum)
Linear 288→256→256 ReLU+Dropout
Projection 256→256→128 ReLU

Fig. 2: Model architecture for the Graph Isomorphism Network with Edge features (GINE).

graph-level representations via concatenated mean, max, and sum pooling, followed by an embedding MLP (ℓ_2-normalized) and a projection head. The contrastive loss maximized agreement between views of the same graph while contrasting against other in-batch graphs (cosine similarity, $\tau = 0.07$). Optimization used AdamW (weight decay 5×10^{-6}), learning rate warmup over three epochs, and exponential decay (initial rate 0.0015, factor 0.85) across 15 training stages [9], [6].

7 Application: Analysing traffic scenes with coverage graphs

Having defined a graph-based traffic scene representation, we can now analyse the coverage of the system. Two methodologies are proposed for this purpose: One is to define archetypes of traffice scenes, and to compare graphs from observed traffic scenes to these archetypes. The second one is to translate graphs to graph embeddings, and then to compare the embeddings of different sets of traffic scenes.

7.1 Subgraph Isomorphism Coverage Analysis

We apply the subgraph isomorphism approach to both the CARLA and the Argoverse data to identify coverage scenarios. The archetypes used here are defined manually and described in table 2. The chosen archetypes are typical traffic situations like e.g. lead vehicle situations, lane change situations or different combinations of opposite direction vehicles and intersections.

While the distributions of scenario characteristics between real-world and simulation data may differ, our primary objective is not to reproduce the exact real-world distribution. Instead, the goal is to systematically identify coverage gaps in the simulation dataset that could lead to insufficient testing of safety-critical scenarios. This gap analysis approach enables targeted improvements of the simulation dataset.

To comprehensively analyze coverage gaps, we employ three complementary methods: (1) identifying gaps in individual scenario archetypes, (2) analyzing gaps in combinations of co-occurring scenarios, and (3) examining gaps in parameter distributions within scenarios. These three perspectives provide a structured framework for understanding where the simulation dataset lacks representation of important real-world traffic situations.

Table 2: Overview of all subgraph archetypes with their structural properties. Edge types: *fl* = following_lead, *lv* = leading_vehicle, *nv* = neighbor_vehicle, *ov* = opposite_vehicle.

Group	**Archetype**	**Actors**	**Edges**	**Edge Types**
Simple (2-actor)	Simple Following	2	2	fl, lv
	Simple Opposite	2	2	ov
	Simple Neighbor	2	2	nv
Complex (3-actor)	Lead + Neighbor (intersection)	3	4	fl, lv, nv
	Cut-in	3	4	fl, lv
	Cut-in (intersection)	3	4	fl, lv
	Platoon (intersection)	3	4	fl, lv
	Opposite Traffic (intersection)	3	4	fl, lv, ov
	Lead + Neighbor at Intersection	3	4	fl, lv, nv
	Triple Opposite (intersection)	3	4	ov
	Lead + Following in Back	3	4	fl, lv
Complex (4-actor)	Lead + Neighbor	3	4	fl, lv, nv
	Cut-out	4	6	fl, lv, nv
	Cut-out (intersection)	4	6	fl, lv, nv
	4-Vehicle Platoon (intersection)	4	6	fl, lv
	4-Vehicle Opposite (intersection)	4	6	fl, lv, ov
Complex (5-actor)	Lead + Neighbor + Opposite	5	8	fl, lv, nv, ov
	Lead + Neighbor + Opposite (inters.)	5	8	fl, lv, nv, ov

Individual Scenario Archetype Coverage Figure 3 presents a comprehensive comparison of scenario archetype coverage between CARLA and Argoverse datasets. The dual-axis visualization displays coverage percentages (horizontal bars) alongside coverage differences (diamond markers). Red diamonds indicate scenarios that are more prevalent in Argoverse, while green diamonds mark scenarios with higher CARLA representation.

The coverage distribution reveals a clear pattern: CARLA achieves higher coverage only for the three simple two-actor scenarios (`simple_following`, `simple_neighbor`, `simple_opposite`), as indicated by green diamond markers. For all complex multi-actor scenarios, Argoverse exhibits substantially higher occurrence rates (red diamond markers), demonstrating that the simulation systematically fails

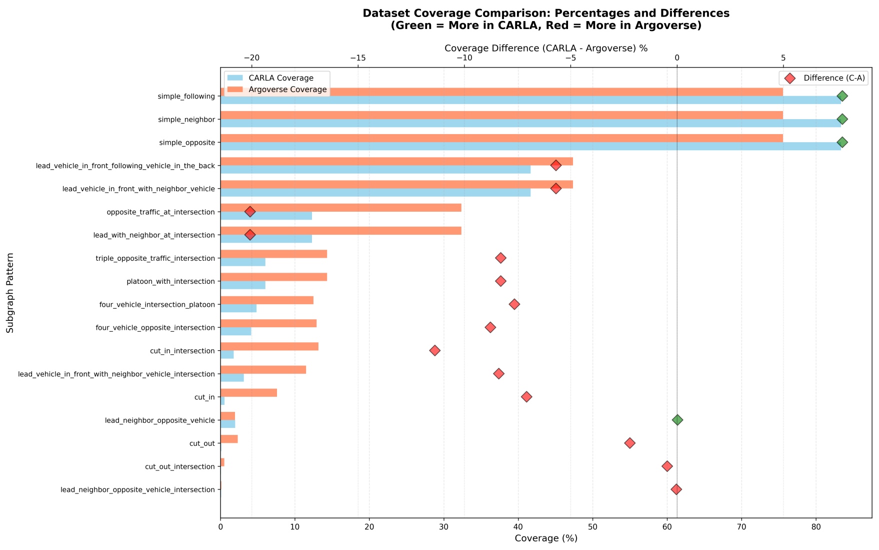


Fig. 3: Scenario archetype coverage comparison between CARLA and Argoverse datasets. Horizontal bars show absolute coverage percentages, while diamond markers indicate coverage differences (CARLA minus Argoverse).

to generate complex traffic situations at the same frequency as observed in real-world data.

The identified coverage gaps in CARLA are substantial. The most critical gaps appear in intersection-related archetypes: `cut_out_intersection` shows nearly complete absence in CARLA (less than 1% coverage) while appearing in approximately 8% of Argoverse scenes, representing a gap of approximately 8 percentage points. Similarly, `lead_neighbor_opposite_vehicle_intersection` appears in roughly 6% of Argoverse data but is virtually absent in CARLA (gap: ∼6 percentage points). The `cut_in_intersection` pattern exhibits a comparable gap of approximately 7 percentage points.

Notably, both `cut_in` and `cut_out` scenarios—fundamental lane change maneuvers—show significant underrepresentation in CARLA regardless of whether they occur at intersections. The `cut_out` pattern appears in less than 1% of CARLA scenes compared to roughly 5% in Argoverse, while `cut_in` shows a gap of approximately 5 percentage points. This systematic absence of lane change scenarios indicates that CARLA's traffic generation lacks the dynamic lateral maneuvers characteristic of real-world driving, limiting its ability to test autonomous systems in scenarios involving merging, overtaking, and lane changes.

Method 2: Parameter Distribution Gaps Even when a scenario archetype is structurally present in both datasets, the distributions of continuous parameters such as speed can reveal additional coverage gaps. To investigate parametric

differences, we analyze role-specific speed distributions within the `lead_vehicle_in_front_with_neighbor_vehi`
scenario, a common highway situation with potential for lane changes.

Since each traffic scene graph contains multiple actors playing different roles within the scenario archetype, we must separate actors by their specific roles in the subgraph to obtain a meaningful comparison. In this three-actor scenario, role ”a” represents the ego vehicle, role ”b” is the leading vehicle in front of ”a”, and role ”c” is the neighbor vehicle adjacent to ”a”. This role-based separation provides a detailed view of how different participants in the same scenario exhibit different parameter distributions.

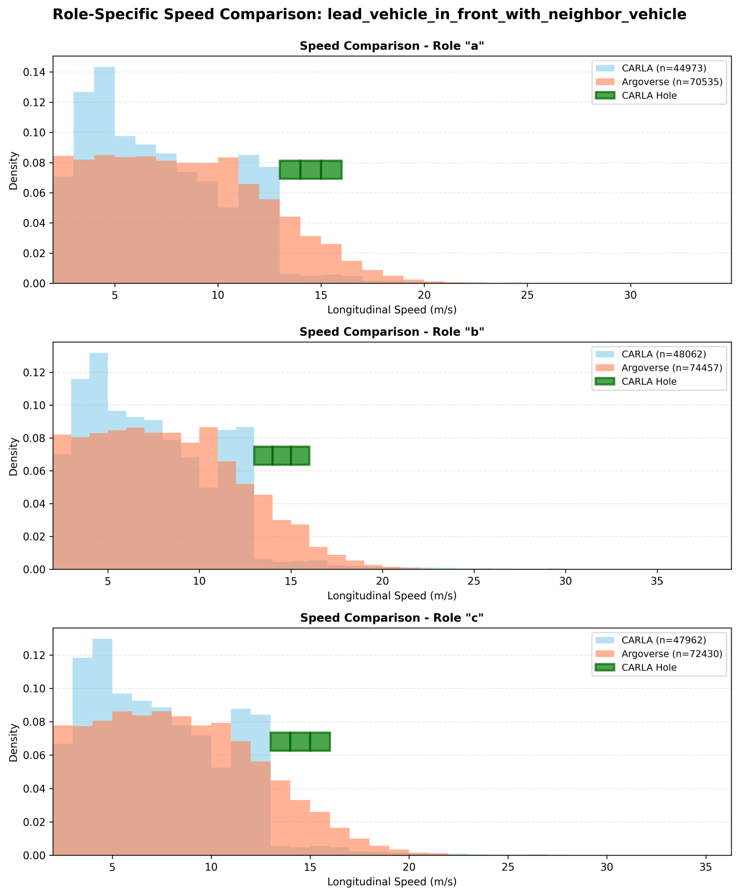


Fig. 4: Role-specific speed distribution comparison for the lead vehicle in front with neighbor vehicle scenario. Green rectangles mark identified coverage gaps where Argoverse shows sufficient density but CARLA is nearly empty. The three panels show distributions for role ”a” (ego vehicle), role ”b” (lead vehicle), and role ”c” (neighbor vehicle).

Figure 4 reveals a systematic pattern: CARLA generally underrepresents higher velocities across all three actor roles. The distributions show that CARLA concentrates actors at lower speeds (peak around 3-5 m/s), while Argoverse exhibits more uniform distributions extending to higher speeds (15-20 m/s and beyond).

Coverage gaps are identified when a speed bin is sufficiently filled in Argoverse (indicating meaningful real-world occurrence) but nearly empty in CARLA (marked as green rectangles in the figure). For role "a" (ego vehicle), gaps appear in the 13-17 m/s range. Role "b" (lead vehicle) shows similar gaps at 13-16 m/s. Role "c" (neighbor vehicle) exhibits gaps at 13-16 m/s. These gaps consistently occur in the moderate-to-high speed range, indicating that CARLA fails to adequately represent highway-speed scenarios with neighboring vehicles, despite the structural scenario being present.

This role-specific parametric analysis demonstrates that coverage gaps exist not only at the structural level (which scenarios are present) and the co-occurrence level (which combinations appear), but also at the parameter level within structurally matching scenarios. The systematic underrepresentation of higher-speed conditions limits CARLA's utility for testing autonomous systems in realistic highway environments where lane changes and overtaking maneuvers typically occur at elevated speeds.

The structured, bottom-up approach presented here demonstrates that subgraph isomorphism can effectively characterize traffic scene coverage for predefined archetypes. However, the manual definition of archetypes and the computational cost of isomorphism checking for large scenario collections motivate the graph embedding approach explored in the following chapter.

7.2 Graph Embeddings Coverage Analysis

The resulting embeddings have been analysed in a number of ways. For plausibility checks, for a number of randomly samples scenarios, the scenario with closest embedding vector (euclidean distance) has been visualized in figure 6. It shows that the embeddings are able to capture the similarity between scenarios.

As a next step, the embedding space has been analysed using dimension reduction techniques, specifically PCA and t-SNE. This is shown in Figure 5. This can be done in a number of different flavors, like for example:

- distinguishing between CARLA and Argoverse scenarios by a color coding,
- visualizing just the CARLA scenarios and color code them by the map, in order to see if the maps deliver different types of scenarios,
- by calculating a density distribution of the embedding space for the Argoverse scenarios, and to check if for a regions with a density about a certain threshold, there exist CARLA scenarios, i.e. do a coverage analysis in the embedded space
- do the same vice versa, i.e. to check for relevance of CARLA scenarios.

As the embedding space is a normal metric space, representing scenarios across a large range of different driving situations, these embeddings can be

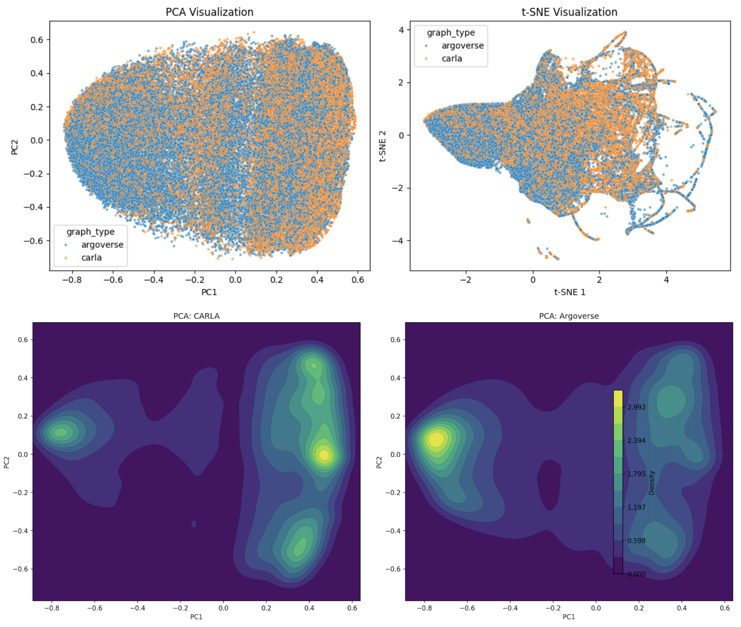


Fig. 5: Top row: PCA and t-SNE visualization of the embedding space for Carla and Argoverse 2.0 scenarios. Bottom row: PCA kernel density plots for Argoverse 2.0 scenarios.

used also for many other tasks, like for example clustering, anomaly detection, similarity search, etc.

And, as the main task considered here is coverage analysis, the embeddings can be used to check if a target distribution is met by a test distribution in the embedded space. Specifically, considering the Argoverse 2.0 scenarios as the target distribution, and the CARLA scenarios as the test distribution, the embeddings can be used to check if the CARLA scenarios cover the Argoverse 2.0 scenarios in the embedded space.

8 Summary

This paper presented a graph-based framework for coverage analysis in autonomous driving that represents traffic scenes as hierarchical graphs combining map topology with actor relationships. Two complementary analysis approaches were developed: subgraph isomorphism using manually defined archetype graphs for interpretable pattern-based coverage metrics, and graph embeddings via GINE networks trained with contrastive learning for similarity-based assessment, clustering, and anomaly detection.

Both approaches were validated on Argoverse 2.0 and CARLA data. It was shown that hierarichal graphs capture complex traffic situtations efficiently and allow the user to define individual specifics on defintions like follow vehicle, opposing vehicle and the set of relevant parameters.

We demonstrated that the graphs capture meaningful traffic scene structure and reveal differences between two datasets (Section 7). First, the bottom-up subgraph approach revealed differences in traffic scnes, combination of traffic scenes and distribution of scenario parameters. Second, the top-down graph embedding approach revealed differences in the embedding space, allowing for further analysis without manual definition of archetypes. This representation can be the foundation for further analysis like similarity search, clustering and anomaly detection.

Compared to approaches like TNO Streetwise [5], the embedding approach scales efficiently without scenario-specific handling rules and naturally accommodates varying numbers of actors.

Future work will incorporate temporal information through multi-timestep graph structures and investigate automatic archetype extraction from real-world data. The source code is publicly available at `https://github.com/tmuehlen80/graph_coverage`.

References

1. Ammann, P., Offutt, J.: Introduction to Software Testing. Cambridge University Press (2008)
2. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International Conference on Machine Learning (ICML). pp. 1597–1607. PMLR (2020)
3. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: A (sub)graph isomorphism algorithm for matching large graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(10), 1367–1372 (oct 2004)
4. 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. pp. 1–16 (2017)
5. Elrofai, H., Paardekooper, J.P., de Gelder, E., Kalisvaart, S., Op den Camp, O.: StreetWise: Scenario-based safety validation of connected and automated driving. Tech. rep., TNO, Helmond, The Netherlands (2018), `https://publications.tno.nl/publication/34626550/AyT8Zc/TNO-2018-streetwise.pdf`
6. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)
7. Foretellix: Av coverage and performance metrics. `https://blog.foretellix.com/2019/09/13/av-coverage-and-performance-metrics/` (September 2019), accessed: 2025-04-04
8. de Gelder, E., Paardekooper, J.P., Saberi, A.K., Elrofai, H., Camp, O.O.d., Kraines, S., Ploeg, J., De Schutter, B.: Towards an ontology for scenario definition for the assessment of automated vehicles: An object-oriented framework. IEEE Transactions on Intelligent Vehicles **7**(2), 300–314 (2022). `https://doi.org/10.1109/TIV.2022.3144803`
9. Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., Leskovec, J.: Strategies for pre-training graph neural networks (2020), `https://arxiv.org/abs/1905.12265`

10. Laboratories, U.: UL 4600: Standard for Safety for the Evaluation of Autonomous Products. UL Standards (2020), standard for autonomous vehicle safety and validation
11. Menzel, T., Bagschik, G., Maurer, M.: Scenarios for development, test and validation of automated vehicles. CoRR **abs/1801.08598** (2018), `http://arxiv.org/abs/1801.08598`
12. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013), `https://arxiv.org/abs/1301.3781`
13. On-Road Automated Driving (ORAD) Committee: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. Standard J3016, SAE International (April 2021). `https://doi.org/10.4271/J3016_202104`, `https://doi.org/10.4271/J3016_202104`
14. PEGASUS: Pegasus method. Available online (2019), `https://www.pegasusprojekt.de/files/tmpl/Pegasus-Abschlussveranstaltung/PEGASUS-Gesamtmethode.pdf`, accessed: February 16, 2026
15. Ries, L., Rigoll, P., Braun, T., Schulik, T., Daube, J., Sax, E.: Trajectory-based clustering of real-world urban driving sequences with multiple traffic objects. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). pp. 1251–1258 (2021). `https://doi.org/10.1109/ITSC48978.2021.9564636`
16. for Standardization, I.O.: ISO 21448:2022 – Road Vehicles – Safety of the Intended Functionality (2022), standard document, ISO/TC 22/SC 32
17. Ulbrich, S., Menzel, T., Reschka, A., Schuldt, F., Maurer, M.: Defining and substantiating the terms scene, situation, and scenario for automated driving. In: 2015 IEEE 18th International Conference on Intelligent Transportation Systems. pp. 982–988 (2015). `https://doi.org/10.1109/ITSC.2015.164`
18. Wachenfeld, W., Winner, H.: The Release of Autonomous Vehicles, pp. 425–449. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). `https://doi.org/10.1007/978-3-662-48847-8_21`, `https://doi.org/10.1007/978-3-662-48847-8_21`
19. Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J.K., Ramanan, D., Carr, P., Hays, J.: Argoverse 2: Next generation datasets for self-driving perception and forecasting. In: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021) (2021)

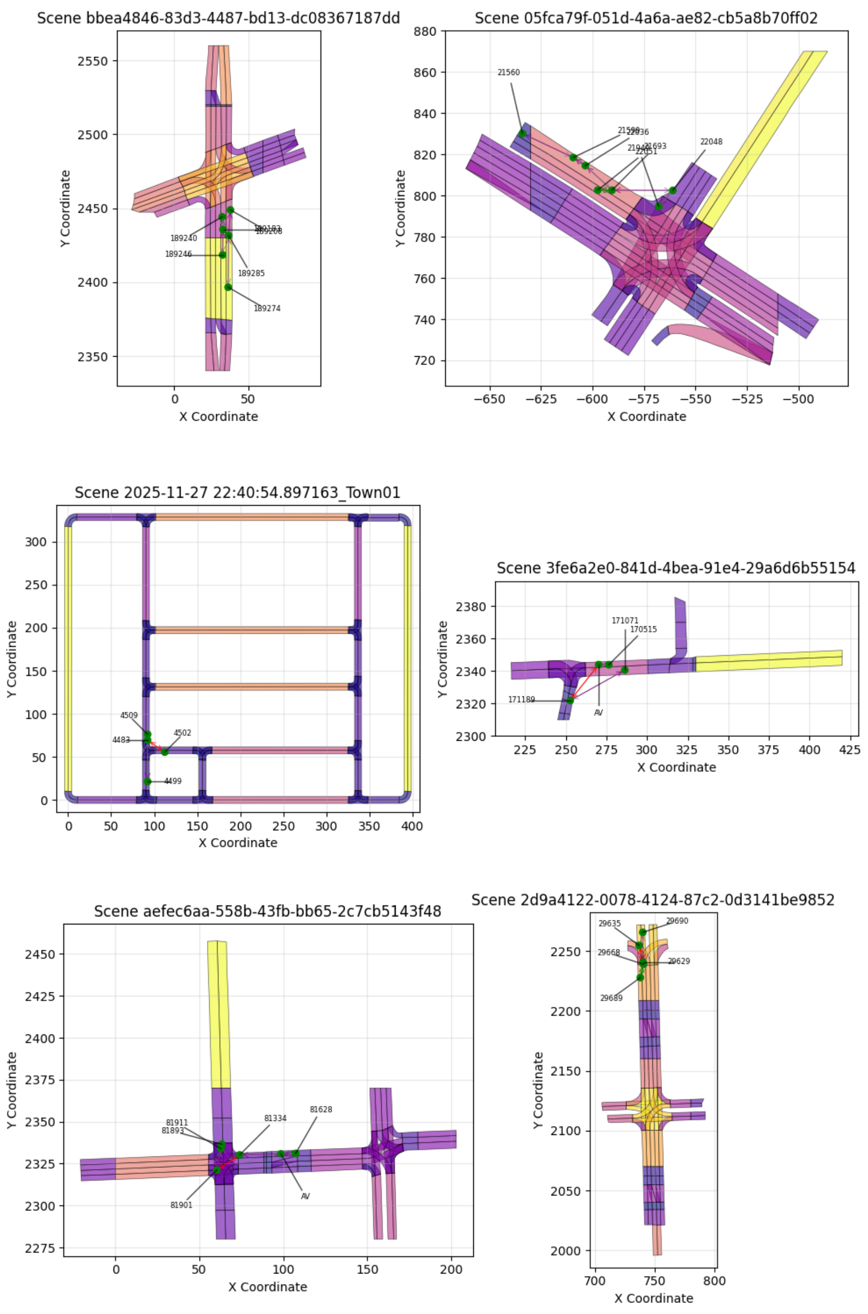

Scene bbea4846-83d3-4487-bd13-dc08367187dd
Scene 05fca79f-051d-4a6a-ae82-cb5a8b70ff02
Scene 2025-11-27 22:40:54.897163_Town01
Scene 3fe6a2e0-841d-4bea-91e4-29a6d6b55154
Scene aefec6aa-558b-43fb-bb65-2c7cb5143f48
Scene 2d9a4122-0078-4124-87c2-0d3141be9852
X Coordinate
Y Coordinate
189240
189246
189285
189274
21560
22048
4509
4483
4502
4499
171071
170515
171189
AV
81911
81893
81334
81628
81901
29635
29690
29668
29629
29689

Fig. 6: Plausibility checks showing graph embedding comparisons. For randomly sampled scenarios, the most similar scenarios based on embedding distance (both Euclidean and cosine) are visualized, including comparisons between CARLA and Argoverse scenarios.