A Graph-based Framework for Coverage Analysis in Autonomous Driving

xxx

October 6, 2025

1 Abstract

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 [13] for an 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.
- There are already other graph-based approaches for analysing and representing traffic scenes, see for example [5].
- However, the work in that paper is not specifically focused on coverage analysis.
- Hence in this paper, graph based traffic scene representations are utilized for coverage analysis.
- This paper is structured as follows:
  - In the first section, xxx

[27]
[3]

3 exsting coverage and analysis approaches

add Master thesis Johannes
[1]
[24]
[21]
[25]
[18]
[2]
[13]
[9]
[26]
[6]
[12]
[17]

4 Defining a traffic scene graph

[5]
[20]
[4]
[23]
[14]
[22]

4.1 time based graph representations

5 Analysing a traffic scene with agraph

- 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.

6 Create subgraphs for coverage analysis

- There is a lot of knowledge in the literature on how to define archetypes of traffic scenes.
- Once an archetype is defined, a special property of graphs can be used.

- Two graphs are isomorphic if they have the same structure, regardless of the node and edge labels.
- As the archetypes are not necessarily are involving a lot of actors, these are more like subsets of actual traffic scenes.
- A very simple example might be 2 vehicles on the same lane, driving in the same direction and another vehicle driving on a neighboring lane.
- This situation can be represented by a graph with 3 nodes and 2 edges.
- In most real traffic situations however, there will be additional actors present, so that we are not searching for isomorphic graphs, but rather want to check if any subgraph of G is isomorphic to the archetype graph A.
- This is an example of a subgraph isomorphism problem.
- While this problem is NP-hard, the graphs considered here are rather small, so the computational time is reasonable.
- One such algorithm is the VF2 algorithm, which is implemented in the NetworkX library (see [8]).
- The strategy we are then applying is the following:
- 1. Define a set of subgraphs S that are considered to be archetypes.
  2. Define which node and edge attributes are considered for the isomorphism check.
  3. Create an empty dataframe C with a column for each subgraph in S
  4. Define the set of traffice scenes (e.g. from Carla or Argoverse) defined as graphs G
  5. For each graph G, check if any subgraph of G is isomorphic to any subgraph in S and note the result in a new row in table C
- This strategy can be described to some degree as a bottom up approach: Starting from a detail level, individual situations are defined.
- Then going upwards to different datasets, it is checked, if the archetype is present.
- Also, follow up analysis of the created coverage dataframe can be performed. For example,
- - The distribution of numeric attributes like speed and and distance to other actors can be visualized for the subset of all traffic scenes which are subgraph isomorphic to an archetype.
  - It can be cross tabulated, which combinations of archetypes are jointly present in a traffic scene.

– Pass Fail rates or other AV performance metrics can be calculated for the subset of all traffic scenes which are subgraph isomorphic to an archetype.

7 Implementation of Graph Embeddings for Traffic Scene Analysis

This section describes the concepts, implementation and application of graph embeddings to traffic scene graphs. The implementation follows a comprehensive approach to learning graph representations through self-supervised contrastive learning.

Embeddings are a widely used method to translate raw data like images or text into an embedding space in order to be able to perform machine learning tasks on them. One well known example of this is the Word2Vec model, which is used to translate words into a vector space, where the distance between vectors can be used to measure the similarity between words ([19]).

In the context of traffic scene graphs, embeddings are used to translate the graph structure into a vector space, where the distance between vectors can be used to measure the similarity between traffic scenes. This is useful for coverage analysis, as it allows to compare traffic scenes among each other. For example, two traffic scenes can be considered similar if the distance between their embeddings is small. This enables to search for a most similar simulation scenario given a real world scenario, to identify areas with near duplicates or to easily visualize structures in the embedding space, which in the original space of all possible traffic scenes would not be possible.

Graph neural networks (GNNs) are a class of neural networks that are designed to process graph-structured data and have gained a lot of popularity in the last years, see for example (add references).

In this paper, a network architecture using a Graph Isomorphism Network with Edge features (GINE) as described in [15] is used to generate embeddings for traffic scene graphs as implemented in the pytorch geometric library ([11]). Main reason for using this specific architecture is that is capable of learning embedding representations not only on the graph structure itself but on both node and edge attributes. Other network architectures like GraphSAGE or GAT are not capable of this. (TODO: check if this is true)

The exact architecture of the model is shown in Figure **??**. The features used are the actor type (as a one hot encoding), the actor speed (float), if the actor is on an intersection (boolean) and if the actor changed its lane since the last timestep (boolean) for the nodes. For the edges, the edge type (as a one hot encoding) and the path length (float) between the two nodes are used.

The core component is the `TrainableGraphGINE` class, which implements a Graph Isomorphism Network with Edge features (GINE) architecture for generating graph-level embeddings. The model consists of multiple GINE convolutional layers, each containing:

- Sequential multi-layer perceptrons (MLPs) with batch normalization and ReLU activations
- Edge-aware message passing incorporating both node and edge features
- Dropout regularization to prevent overfitting

The model employs a combination of three graph pooling strategies - mean, max, and sum pooling - to create a comprehensive graph-level representation. A projection head enables contrastive learning through InfoNCE loss, while an optional classification head supports supervised learning tasks.

7.1 Data Processing and Augmentation

The `GraphDataset` class handles the conversion from NetworkX graphs to PyTorch Geometric format through the `networkx_to_pyg` function. This conversion includes:

- One-hot encoding of categorical node features (actor types: VEHICLE, PEDESTRIAN, CYCLIST, MOTORCYCLE)
- One-hot encoding of edge types (neighbor_vehicle, opposite_vehicle, same_lane, adjacent_lane, following, intersection)
- Preservation of continuous features such as longitudinal speed and path length

Data augmentation is implemented through the `augment_graph` function, which adds Gaussian noise to continuous features to create augmented versions for contrastive learning.

7.2 Training Methodology

The training process utilizes self-supervised contrastive learning with the following characteristics:

- InfoNCE contrastive loss implemented in the `contrastive_loss` function
- Adaptive learning rate schedule with exponential decay (starting at 0.02, decaying by factor 0.75)
- Multi-epoch training with 15 epochs per learning rate stage
- Data from both CARLA simulator (4,442 graphs) and Argoverse 2.0 dataset (3,588 graphs)

7.3 Embedding Analysis and Visualization

The trained model generates 256-dimensional embeddings for each traffic scene graph. Analysis includes:

- Dimensionality reduction using Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE)
- Visualization of embedding space distinguishing between CARLA and Argoverse data sources
- Similarity-based scene retrieval using squared Euclidean distance in embedding space
- Integration with visualization functions (`plot_lane_map_advanced`, `add_actors_to_map`, `add_actor_edges_to_map`) to display similar traffic scenes

The implementation demonstrates successful learning of traffic scene representations, with the embedding space capturing meaningful similarities between scenes from different data sources while maintaining distinction between simulation and real-world data.

8 Application

8.1 Argoverse 2.0

[28]

8.2 Carla

CARLA (Car Learning to Act, [10]) is an open-source simulator specifically designed for autonomous driving research and development. It provides a highly realistic urban driving environment with diverse road layouts, weather conditions, and traffic scenarios. The simulator features a comprehensive sensor suite simulation, flexible API for scenario creation, and supports both learning-based and traditional autonomous driving approaches. CARLA enables researchers to test and validate autonomous vehicle systems in a safe, controllable environment before real-world deployment.

The simulator has gained widespread adoption across both academic and industrial settings. In research, CARLA serves as a standard platform for developing and benchmarking autonomous driving algorithms, including reinforcement learning approaches for vehicle control and sensor fusion techniques [7]. Industry applications include virtual testing of production autonomous vehicle systems, scenario-based validation pipelines, and integration with hardware-in-the-loop testing frameworks [16]. CARLA is also extensively used in autonomous driving competitions and challenges, providing a common evaluation environment for comparing different approaches across research groups worldwide.

Here, Carla version 0.9.15 is used. The CARLA version 0.10.0 is not used, because it had only 2 maps and Mine_1 (which is not really normal roads) at the start of this project. Specifically, the following maps were used: Town01, Town02, Town03, Town04, Town05 and Town07. Plots of these maps are shown in Figure 1.

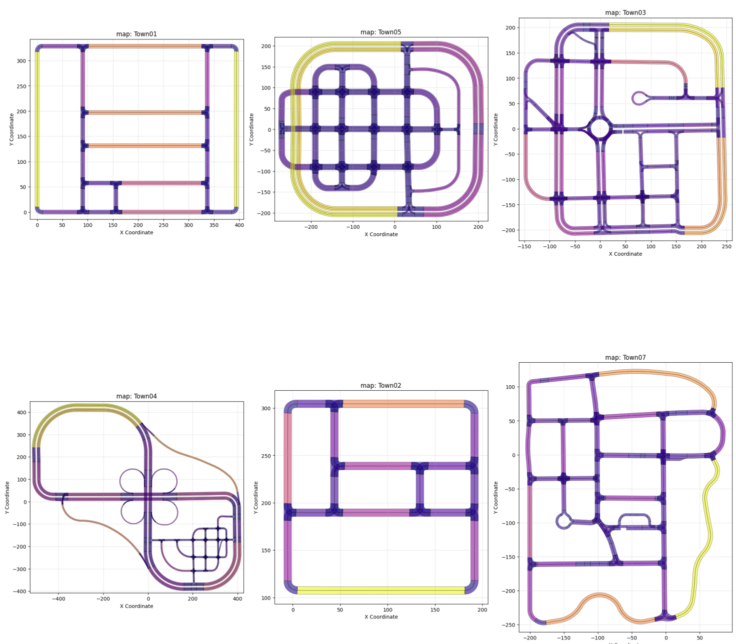

Figure 1: Overview of CARLA maps used in the simulation study: Town01, Town02, Town03, Town04, Town05, and Town07. These maps provide diverse urban driving environments with varying road layouts, intersections, and traffic patterns.

The data generation script implements sophisticated behavior control mechanisms to create diverse and realistic traffic scenarios. Multiple vehicle types including trucks, motorcycles, and regular cars are spawned 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 xxx scenes with 11 seconds of simulation time each, in order to have a similar data size as the Argoverse 2.0 dataset.

9 Summary

References

[1] Pegasus method. Available online, 2019. Accessed: October 6, 2025.

[2] Paul Ammann and Jeff Offutt. *Introduction to Software Testing*. Cambridge University Press, 2008.

[3] Artem Arzamasov, Martin Lauer, and Klaus Bengler. Data-driven testing of autonomous driving systems: A survey. *Journal of Field Robotics*, 38(8):1063–1087, 2021.

[4] Mojtaba Bagheri, Thomas Menzel, and Markus Maurer. Ontology-based scene creation for the development of automated vehicles. *CEAS Aeronautical Journal*, 11(3):661–673, 2020.

[5] Felix Behr, Thomas Menzel, Gerrit Bagschik, and Markus Maurer. A graph-based model for representing and analyzing traffic scenarios. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1458–1465. IEEE, 2021.

[6] Christian Berger, Bernhard Rumpe, Julian Stier, et al. A survey on scenario-based verification and validation of autonomous vehicles. *IEEE Access*, 8:87456–87477, 2020.

[7] Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving, 2019.

[8] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1367–1372, oct 2004.

[9] Erwin de Gelder, Jan-Pieter Paardekooper, Arash Khabbaz Saberi, Hala Elrofai, Olaf Op den Camp, Steven Kraines, Jeroen Ploeg, and Bart De Schutter. 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.

[10] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[11] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[12] International Organization for Standardization. ISO 21448:2022 – Road Vehicles – Safety of the Intended Functionality, 2022. Standard document, ISO/TC 22/SC 32.

[13] Foretellix. Av coverage and performance metrics. `https://blog.foretellix.com/2019/09/13/av-coverage-and-performance-metrics/`, September 2019. Accessed: 2025-04-04.

[14] Daniel J. Fremont, Tommaso Dreossi, and Sanjit A. Seshia. Scenic: A language for scenario specification and scene generation. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 118–132. ACM, 2020.

[15] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks, 2020.

[16] Bernhard Jaeger, Kashyap Chitta, and Andreas Geiger. Hidden biases of end-to-end driving models, 2023.

[17] Underwriters Laboratories. *UL 4600: Standard for Evaluation of Autonomous Products*. UL Standards, 2020. Standard for autonomous vehicle safety and validation.

[18] Till Menzel, Gerrit Bagschik, and Markus Maurer. Scenarios for development, test and validation of automated vehicles. *CoRR*, abs/1801.08598, 2018.

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[20] M.E.J. Newman. *Networks, An Introduction*. Oxford University Press, 2010.

[21] 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.

[22] Carlos Pek, Gerrit Bagschik, Thomas Menzel, and Markus Maurer. Generating traffic scenario variants using ontologies and graph transformations. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1394–1401. IEEE, 2019.

[23] Stefan Riedmaier, Christian Bock, Anselm Stahl, and et al. Realistic modeling of traffic scenarios for the safety validation of automated vehicles. *Technische Sicherheit*, 2020. Technical report from PEGASUS and VVM projects.

[24] Lennart Ries, Philipp Rigoll, Thilo Braun, Thomas Schulik, Johannes Daube, and Eric Sax. Trajectory-based clustering of real-world urban driving sequences with multiple traffic objects. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1251–1258, 2021.

[25] Simon Ulbrich, Till Menzel, Andreas Reschka, Fabian Schuldt, and Markus Maurer. Defining and substantiating the terms scene, situation, and scenario for automated driving. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 982–988, 2015.

[26] Walther Wachenfeld and Hermann Winner. The release of autonomous vehicles: A safety argumentation and its validation. In *Proceedings of the 22nd ESV Conference*, 2016.

[27] Stephan Wagner, Thomas Menzel, Gerrit Bagschik, and Markus Maurer. Operational design domain verification and validation: Challenges and approaches. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 948–955. IEEE, 2022.

[28] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. 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.