A Graph-based Framework for Coverage Analysis in Autonomous Driving

xxx

October 12, 2025

1 Abstract

test2

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 [**?**] 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.
- 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

3 exsting coverage and analysis approaches

add Master thesis Johannes
[?]
[?]
[?]
[?]
[?]
[?]
[?]
[?]
[?]
[?]
[?]

4 Defining a traffic scene graph

still to be added: https://sagroups.ieee.org/adwg/wp-content/uploads/sites/661/2024/10/ADWG$_S TV2_w hitep$
still to be added: https://www.vvm-projekt.de/securedl/sdl-eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ey
[?]
[?]
[?]

4.1 time based graph representations

5 Analysing a traffic scene with a graph

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 [**?**]). The strategy we are then applying is the following:

1. Define a set of subgraphs S that are considered to be archetypes of traffic scenes, e.g. unprotected left turns with opposite traffic or lead vehicle following situations.
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 ([**?**]).

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 [**?**] is used to generate embeddings for traffic scene graphs as implemented in the pytorch geometric library ([**?**]). 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 model has been trained on the CARLA and Argoverse 2.0 datasets using self-supervised contrastive learning.

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 as well as cosine distance) has been visualized. This is shown in Figure **??**. This includes comparison between CARLA and Argoverse scenarios.

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

- distinguishing between CARLA and Argoverse scenarios by a color coding,
- visualiizing 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 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.

The results for this approach are shown in Section xxx.

8 Application

8.1 Argoverse 2.0

[**?**]

8.2 Carla

CARLA (Car Learning to Act, [**?**]) 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 [**?**]. Industry applications include virtual testing of production autonomous vehicle systems, scenario-based validation pipelines, and integration with hardware-in-the-loop testing frameworks [**?**]. 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 **??**.

The data generation script implements sophisticated behavior control mechanisms to create diverse and realistic traffic scenarios. Multiple vehicle types

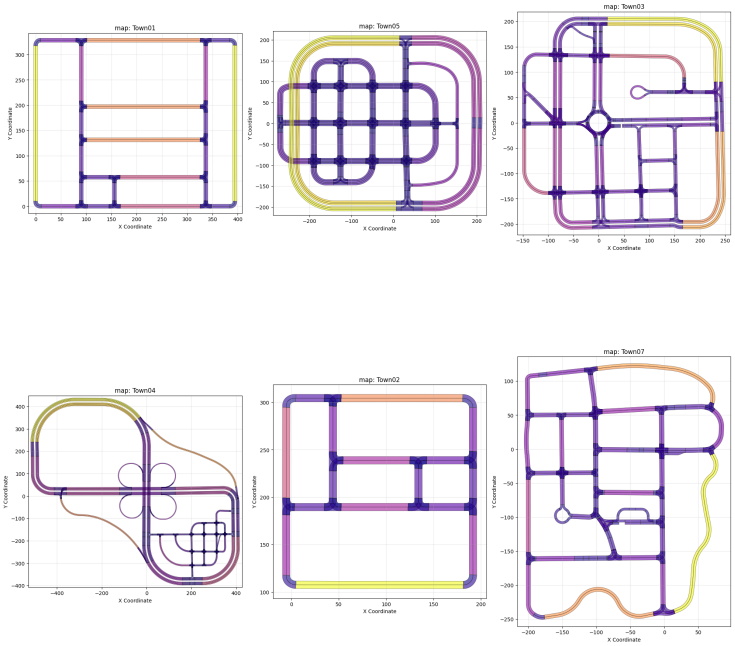

Figure 1: Overview of CARLA maps used in the simulation study: Town01, Town02, Town03, Town04, Town05, and Town07. These maps provide diverse urban driving environments with varying road layouts, intersections, and traffic patterns.

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