SUMO4AV: An Environment to Simulate Scenarios for Shared Autonomous Vehicle Fleets with SUMO Based on OpenStreetMap Data

Reichsöllner E.¹, Freymann A.², Sonntag M.¹, and Trautwein I.²

¹ University of Applied Sciences Esslingen, Germany
emanuel.reichsoellner@hs-esslingen.de,
mirko.sonntag@hs-esslingen.de
² Fraunhofer IAO, Stuttgart, Germany
andreas.freymann@iao.fraunhofer.de, ingo.trautwein@iao.fraunhofer.de

Abstract

In the past years the progress in the development of autonomous vehicles has increased tremendously. There are still technical, infrastructural and regulative obstacles which need to be overcome. However, there is a clear consent among experts that fully autonomous vehicles (level 5 of driving automation) will become reality in the coming years or at least in the coming decades. When fully autonomous vehicles are widely available for a fair trip price and when they can easily be utilized, a big shift from privately owned cars to carsharing will happen. On the one hand, this shift can bring a lot of chances for cities like the need of less parking space. But on the other hand, there is the risk of an increased traffic when walking or biking trips are substituted by trips with shared autonomous vehicle fleets. While the expected social, ecological and economical impact of widely used shared autonomous vehicle fleets is tremendous, there are hardly any scientific studies or data available for the effects on cities and municipalities. The research project KI4ROBOFLEET addressed this demand. A result of the project was SUMO4AV, a simulation environment for shared autonomous vehicle fleets, which we present in this paper. This simulation tool is based on SUMO, an open-source traffic simulation package. SUMO4AV can support city planners and carsharing companies to evaluate the chances and risks of running shared autonomous fleets in their local environment with their specific infrastructure. At its core it comprises the mapping of OpenStreetMap entities into SUMO objects as well as a Scenario Builder to create different operation scenarios for autonomous driving. Additionally, the simulation tool offers a recursive execution with different fleet sizes and optimization strategies evaluated by economic and ecologic parameters. As oval-uation of the toolset a simulation of an ordinary scenario was performed. The workflow to simulate the scenario for shared autonomous vehicle fleets was successfully processed with the SUMO4AV environment.

1 Introduction

Within the last decades, the automotive industry has enhanced their development with many automation levels ending up in autonomous vehicles (AVs) [12]. Today that field has got a wide interest and is further expanding. The German parliament has recently passed a regulation that facilitates the operation of autonomous driving in February 2022 [3]. However, critical topics such as safety or trust in autonomous systems still needs to be further improved [12]. It is expected that in future shared autonomous vehicles (SAVs) will be one building block of

¹Url: https://www.openstreetmap.org/
mobility in cities. Thus, the definition of autonomous systems and scenarios and how they can positively influence the mobility for individuals and the society need to be further investigated [12]. One challenge is the lack of scientific data about the operation of shared autonomous vehicle fleets (SAV fleets). The simulation of AVs is one method to obtain realistic data. It helps to get deeper insights into influence factors in a very complex setup, e.g., fleet sizing, pricing, or changes in the individual mobility behavior [13].

To address this issue, we developed a toolset for the simulation of SAV fleets, called SUMO4AV, that we describe in this paper and which was developed within the project KI4ROBOFLEET2. SUMO4AV extends the Simulation of Urban Mobility (SUMO) package [10]. We use the scenario of transportation between points of interest using SAV fleets (called TPOI-Scenario) as running example. To be precise, the TPOI-Scenario simulates people who need a ride after work from office or from home to other amenity locations such as restaurants, parks or cinemas.

Basically, SUMO enables the simulation of vehicles on roads by representing the roads as edges and the junctions as nodes in a network. The vehicles can move along the edges and turn to another edge/road at the junctions. However, this representation as a network alone does not provide a satisfactory implementation of SAV fleets and their behavior in distinct use cases. The implementation of the TPOI-Scenario and other complex scenarios requires the simulation model to reflect different points of interest, which cannot be modeled directly by the SUMO network. This is where SUMO4AV comes into play. It enables SUMO to simulate SAV fleets that operate according to defined scenarios and that reflect points of interest. SUMO4AV comprises several steps. In the first step, infrastructure data such as roads and points of interests are analyzed and processed. This data can be extracted from OpenStreetMap and will be transferred into a simulation model. The OSMWebWizard [6] tool from SUMO is used to perform the import. The second step enables to connect the map entities (e.g., POIs) with the SUMO objects like edges and parking areas. To enable interactions with the map entities, a projection of the map entities on the network is provided. With this realization, vehicles can now navigate to the map entities that are located on an edge. The third step allows to create customized scenarios. Finally, the fourth step offers a choice of predefined routing algorithms to simulate the customized scenarios.

The developed SUMO4AV environment is evaluated using map data from Mannheim, a city in Germany. Through an iterative process, the simulation was executed with multiple fleet sizes and optimization algorithms. The results show that the simulations give insights into the operation of SAV fleets. Using such a tool, cities or companies in the mobility sector are enabled to evaluate local opportunities and risks for the operation of SAV fleets reflecting local characteristics, like infrastructure, POIs or living and business areas [13].

The rest of the paper is organized as follows: Section 2 provides a review of relevant literature. Section 3 represents the workflow of how to generate the SUMO model to simulate the TPOI-Scenario. This comprises, firstly, how a static simulation model is created using OpenStreetMap data. Secondly, it includes how the required entities are modeled as SUMO objects for the TPOI-Scenario. Thirdly, it involves how the TPOI-Scenario is configured and which optimization algorithms are provided. The extended simulation environment is evaluated in Section 4 using a practical example, and the resulting findings and ideas for further work initiatives are finally presented in Section 5.

---

2 Url: https://www.keim.iao.fraunhofer.de/de/projekte/KI4ROBOFLEET.html
2 Related Work

An observable recurring challenge is the development of reliable and usable mobility scenarios to evaluate new mobility concepts especially in the context of AVs. Several publications present simulation scenarios that deal with the simulation of fleet behavior patterns.

Simulations of autonomous and non-autonomous fleets: Wang et al. [18] extend an approach of Autonomous Intersection Management at an intersection with the focus on connected AVs by considering different challenges such as pedestrian road crossings. Vakayil et al. [17] describe a model for the allocation of user demand between an autonomous mobility-on-demand-system and mass transit. They developed a framework to support an operator’s fleet management planning. Spieser et al. [15] use rebalancing strategies to enable a fleet operator to achieve a favorable balance between customer satisfaction and corporate goals. Different rebalancing strategies are evaluated in a simulation. The last two approaches do not reflect different types of POIs for the simulation of SAV fleets. From our point of view, this aspect is important for running more precise scenarios in the context of autonomous mobility and, hence, to show its potential. Although the information about the characteristics of POIs is extracted via the OSMWebWizard, this information is only used to display buildings and other map entities in the SUMO GUI. In contrast to that, our approach is to integrate information about the POIs into SUMO by a new component, so that the individual AVs can access them in a simulation.

Simulations of conventional fleets in SUMO: Malinverno et al. [11] developed framework to simulate vehicle-to-infrastructure. That framework supports different communication protocols. Codeca et al. [4] run mobility scenarios that support different kinds of traffic demands or free-flow patterns, scenario dimensions and road categories. Additionally, multi-modal traffic evaluations and traffic scenarios like rush hours are considered. The SUMO simulation also contains POIs, which are extracted from OpenStreetMap. As opposed to our approach, the POIs are differentiated binarily: into buildings and parking lots. In another work by Codeca et al. [5] the impact of vulnerable road users on road traffic is investigated. The goal is to optimize traffic and reduce traffic jam through Cooperative Intelligent Transport System applications. Bautista et al. [2] investigate the effects of dynamic route planning in vehicle simulations. They examine the impact of vehicle rerouting capabilities on vehicle mobility and vehicle network connectivity.

Simulations of SAV fleets in SUMO: S. Alazzawi et al. [1] performed a study that included the analysis of traffic count data and mobile phone data to investigate mobility demand and traffic jams. This data was combined with extensive simulations of conventional (classical) cars and self-driving robo-taxis in Milan, Italy. SUMO was chosen as the simulation environment and the map material was imported from OpenStreetMap. The self-driving robo-taxis are offered as on-demand mobility service and have the goal to transport people over a certain route and to pick up other users on the way having the same travel destination. The focus of Schweizer et al. [14] is on the development of travel demand generators that aim to create person-based plans for SUMO. This involves generating populations, activities and associated locations, travel plans, and determining travel time. Based on the calculated travel times, people can modify their travel plans. In Li et al. [9], the open-source simulator for autonomous driving research CARLA is combined with SUMO to create a traffic environment for training AVs. Another approach is followed by Kusari et al. [8]. Here, the background traffic is approximated to reality by adjusting the parameter distribution of well-known car-following

3Url: https://carla.org/
models from driving databases. The second difference is that scenarios are abstracted by taking the strengths of the SUMO simulator and combining them with those of OpenAI Gym. The work of Gasper et al. [7] deals with the use of autonomous shuttles. Here, the autonomous shuttles move autonomously in an area shared with pedestrians. The simulation in SUMO is intended to derive further requirements for the development of AVs. The vehicles as well as the pedestrians both move on the streets and certain situations are simulated, such as a shuttle passing pedestrians without collisions.

3 Creating the SUMO Model

The workflow and its activities for extracting map entities and generating scenarios with SUMO4AV is presented in Figure 1. The figure also sketches the topics of the following subsections. In the first activity, Import Map, the OSMWebWizard extracts the map material from OpenStreetMap and stores it in individual XML files. The file osm.poly.xml contains information about the POIs, which is further processed in the self-developed component SUMO4AV (grey box) that also comprises the other activities of the workflow. Basically, converting data from OpenStreetMap using the OSMWebWizard provides a convenient, reliable and fast way to create a running SUMO model from scratch. To be able to access map entities like POIs and buildings in the SUMO model created with the OSMWebWizard a workaround must be performed, which is implemented in the second activity, Process Map Entities (Section 3.2). The SUMO4AV environment provides GUI based functions to process POIs and make them accessible for AVs. In the next activity, Generate Scenarios (in Section 3.3), the Scenario Builder is used to generate the considered TPOI-Scenario. The last activity, Run Simulation (3.4), describes the flow of the simulation, which includes on the one hand the input file (osm.sumocfg) and static parameters, which go directly into the SUMO simulation and on the other hand the dynamic routing of the selected strategies of the AVs, which are transmitted to the simulation via the interface TraCI4.

3.1 Import Map

The first activity aims to create a SUMO model from OpenStreetMap by selecting a map area within the OSMWebWizard. For the TPOI-Scenario we used map data from Mannheim, Germany. The output comprises several files, which are shown in Figure 1. They are necessary for an executable SUMO model and are listed in the following:

- osm.net.xml contains all essential map entities for the simulation like roads, lanes, rails and traffic lights.
- osm.poly.xml contains polygons and points representing specific areas, buildings and POIs. This file is not required to run the simulation but improves the map appearance in the SUMO GUI by displaying buildings, POIs and other map entities in different colors.
- osm.view.xml contains the SUMO GUI settings.
- osm.[x]trips.xml: contains a random base traffic specification (see Section 3.4).
- routes.xml: contains base traffic specification from traffic counting data (see Section 3.4).
- osm.sumocfg: is the SUMO master file which includes the files above.

4Url: https://sumo.dlr.de/docs/TraCI.html
3.2 Process Map Entities

As described in Section 3.1, the `osm.poly.xml` file is one of the output files of the first activity and serves as input file for the second activity, which contains the mapping of POIs to SUMO accessible objects.

Every entity in the file contains a unique Id and further information (e.g., the POI type like `amenity.restaurant`), which was transferred from the OpenStreetMap data. In the SUMO GUI these entities can be inspected, and their view settings can be changed, but there is no interaction possible between these map entities and the simulation. For closing this gap, we developed the SUMO OSM POI-Tools, which are part of SUMO4AV (Figure 2). They provide a set of functions to create a workaround that meets the aforementioned demands of interacting with map entities for more detailed and realistic simulations and scenario analysis. Furthermore, they comprise functions to analyze, process, map, and display POIs and other map entities.

Figure 2 shows how different OpenStreetMap entity types (e.g., `building.office`) are imported and colored in a customizable way. Each entity type consists of a main type or primary feature (e.g., building) and a sub type (e.g., office). The usage of types and subtypes follow a certain principle described in the OpenStreetMap documentation\(^5\). The first step in the SUMO OSM POI-Tools is to select the map entity types that should be considered for the simulation. Optionally, the colors for these map entities on the map can be set.

After the selection, the following functions (scripts) for further processing are available:

- **Create Edge Positions**: With this script each entity instance of the selected entity types gets a position in the SUMO model. The entity instances are mapped on a SUMO edge (by the edge Id) and given an edge position to make them accessible for the SUMO routing algorithm. The edge position is a float value between zero and the length of the edge. To identify which end of the edge equals to the zero position the edge definition has to be considered. The result is the file `POIsEdges.xml`.

---

\(^5\)Url: https://wiki.openstreetmap.org/wiki/Map_features
• **Convert Parking Areas**: With this script the entity instances with type `amenity.parking` are converted to parking areas (with several parking lots) which can be utilized by the SUMO routing algorithm. The routing strategies, described in Section (3.4), use these parking areas to park AVs that have no trip request at hand. In some `amenity.parking` entities the number of parking lots are specified. We use this data to create the correct number of parking lots for the simulation. If the number is not available, we take a default value. With this script, also the type `amenity.car_sharing` can be converted to parking areas, because it is considered that today’s carsharing companies will be operators of SAV fleets in the future. The output file `parkingAreas.xml` is created and can be included into the SUMO model, i.e., the `osm.sumocfg` file.

• **Apply View Settings**: This script modifies the `osm.poly.xml` input file according to the entity type selection and color settings in the SUMO4AV GUI to apply them in the SUMO GUI.

• **Create POI Statistics**: This script creates the `POI_Statistics.csv` file with statistics of the occurrence for each map entity type (e.g., building.school) of the selected map area.

• **Create Map Legend**: This script creates the `POI_Legend.png` file providing a map legend with all selected entity types and their current color settings. The polygons are represented as colored squares and the POIs as colored points.

### 3.3 Generate Scenario

The **Scenario Builder** module of SUMO4AV, shown in Figure 3, is a tool to create customized lists of random requests which define certain simulation scenarios. For each scenario a list of pickup and target map entity types has to be specified as input data. A use case of a common leisure (after work) scenario could be, for example, to pick up customers at an entity of type `building.office` and to bring them to an entity of type `amenity.restaurant` or `leisure.fitness_centre`. A scenario usually comprises a list of several transportation use cases with separate pickup and
target map entity types. The Scenario Builder randomly picks map entities of the specified pickup and target map entity types and creates a list of customer requests. The submission time is randomly distributed, so the requests arise randomly within the total simulation time specified in the SUMO4AV GUI. For each use case in the list, a certain number of requests is specified. An additional feature is the roundtrip option. Here, for each request a return ride is scheduled after a certain stay time, which can optionally be normal distributed with a given standard deviation to model the customer behavior more realistically. The output of the Scenario Builder is the CustomerRequests.xml file, which contains a list of SUMO readable edge Ids and edge positions for the pickup and target map entities. This file contains also additional meta data for the considered map entities (e.g., restaurant type, opening times and the url), but this data is not used yet by our toolset.

Figure 3: The SUMO4AV Scenario Builder module

3.4 Run Simulation

After creating the SUMO model and the CustomerRequests.xml file, the simulation can be started with three different routing strategies, which define how the customer requests from the CustomerRequests.xml file are processed. In the following Table 1, the developed routing strategies as well as the related conditions and parameters are described. The two parameters LF (lateness factor) and RT (realistic time) for the shared strategy are not self-evident. They are used to determine the length of the detour to pick up a second customer on a similar route. Following calculations are performed in this context: travel_time = sumo_estimate * RT and expected_finish = travel_time * LF where sumo_estimate is the estimated travel time from the SUMO routing algorithm.

The interaction of the running SUMO traffic simulation with AVs and customers is implemented in Python, using the SUMO traffic control interface (TraCI). TraCI provides a large set of functions to access and modify entities of a SUMO traffic simulation during the runtime. In the current implementation of the fleet management algorithm and the routing strategies, TraCI was used to push the data of the Python objects for AVs and customers to the running simulation and to fetch the current state. When a customer request from the CustomerRequests.xml file is submitted, the waiting customer is placed via TraCI at the specified pickup edge id and edge position. Then an AV needs to be dispatched to this customer. SUMO provides a configurable Taxi dispatch algorithm to simulate this case of demand responsive

6Url: https://sumo.dlr.de/docs/TraCI.html
transport (DRT). All strategies described in Table 1 use the SUMO Taxi dispatch algorithm, but they differ in the way how the requests are processed.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Description</th>
<th>Conditions</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>simple</td>
<td>After submitting a customer request, the closest free AV is assigned to this request.</td>
<td>only one customer per AV, fleet size is fixed</td>
<td>fleet size</td>
</tr>
<tr>
<td>look ahead</td>
<td>The fleet is informed about the predicted location of an upcoming customer request in advance, e.g., 10 min (look ahead time) and sends an AV to this location. The idea is to make request predictions based on AI techniques.</td>
<td>only one customer per AV, fleet size is fixed</td>
<td>fleet size, look ahead time [sec]</td>
</tr>
<tr>
<td>shared</td>
<td>The AV can make a detour to pick up a second customer with a similar route.</td>
<td>One or two customers per AV, flexible fleet size</td>
<td>LF (lateness factor)<em>, RT (realistic time)</em></td>
</tr>
</tbody>
</table>

*The length of the detour to pick up a second customer on a similar route can be specified by two parameters called lateness factor (LF) and realistic time (RT).

Optionally, it should be considered to include daily base traffic in order to get a more realistic simulation model. A straightforward way to create a random base traffic is provided by the OSMWebWizard, where different types and flow-rates of traffic (i.e., cars, trucks, buses, motorcycles, bicycles, pedestrians, trams, trains and ships) can be included. For each type of traffic entity, a separate file is created, e.g., osm.bus.trips.xml. However, in order to consider a more realistic base traffic, a more complex approach can be performed by using traffic counting data, which can be converted to XML route files (routes.xml) by using the SUMO routeSampler script.

4 Evaluation

For the evaluation of SUMO4AV and the complete workflow we performed the TPOI-Scenario at the city center of Mannheim (Figure 4). The scenario represents a typical and realistic leisure scenario, which can be observed in cities. Pickup locations are, for example, apartments, offices or residential areas and target locations are restaurants, cinemas, pubs, theatres, fitness centers or beer gardens.

This TPOI-Scenario has been applied to the workflow described in Section 3. Firstly, we imported the map area from the OSMWebWizard as described in the first activity in the workflow (Figure 1). Secondly, we applied the next steps by using SUMO4AV. They comprised the processing of the map entities (activity two in the workflow). For that we used the SUMO OSM POI-Tools to select and to colorize the map entities which were relevant to the TPOI-Scenario. This is shown in Figure 2. Moreover, we generated the TPOI-Scenario with the

7 Url: https://sumo.dlr.de/docs/Simulation/Taxi.html
8 Url: https://sumo.dlr.de/docs/Tools/Turns.html#routesampler.py
Scenario Builder as shown in Figure 3 including several pickup and target map entity types. The output file CustomerRequests.xml of the Scenario Builder contains 300 requests and was processed 14 times with three different routing strategies, each with different parameter sets. With respect to the routing strategies as described in Table 1, the simulations with the simple strategy and with the look ahead strategy were each performed with a fleet size parameter of 10, 25, 50, 100 and 200. Additionally, for the look ahead strategy, the parameter for the look ahead time was set to 900 seconds (15 minutes). The shared strategy was applied with four different pairs of values for the lateness factor (LF) and realistic time (RT): LF=1.2 and RT=1.0, LF=1.4 and RT=1.0, LF=1.2 and RT=4.0, LF=1.4 and RT=4.0. All 300 requests were submitted within the first 30 minutes, but the initial simulation time was set to a much higher value of 10 hours (36000 seconds) to ensure, that even simulation runs with small fleet sizes can complete all 300 requests and subsequently to make the simulation results comparable. When all 300 requests are fulfilled, the current simulation run is aborted and the result file is written. We originally planned to use base traffic derived on traffic counting data. Therefore the city of Mannheim provided traffic counting data of 89 counters in the considered map area, which were converted to XML route files (routes.xml) by using the SUMO routeSampler\textsuperscript{9} script. After we faced severe stability problems by including the (routes.xml) file, we decided to perform the proof-of-concept simulations without base traffic.

The AVs were considered as fully electric vehicles with following boundary conditions for the simulation and the subsequent calculation of key figures:

- Energy consumption per vehicle: 15kWh/100km
- Emissions: 401g/kWh (according to the German energy mix for generating electricity 2019 \textsuperscript{16})
- Energy costs: 0,32€/kWh
- Fleet base costs per vehicle: 3€/h

\textsuperscript{9}Ibid.
Figure 5 shows that the runs for the TPOI-scenario in the SUMO4AV environment were a successful proof-of-concept simulation with comprehensible results, even when the negligence of the base traffic might gloss over the results. The required simulation time depends strongly on the fleet size and varies between 2305 seconds and 26363 seconds. Further simulation results are ecological and economical key figures, e.g., CO\textsubscript{2} emissions, total costs, or waiting times, calculated as average values per request to make these values more descriptive. The results show, that for the given boundary conditions with 300 requests, a minimum fleet size of 50 is necessary to achieve an acceptable waiting time of less than 30 minutes. A fleet size of 25 leads to waiting times of more than one hour and a fleet size of 101 leads to waiting times of more than 3 hours, but with such small fleet sizes the costs per kilometer are less than 0.5€ which can be attractive for some specific use cases. The look ahead strategy seems to be the most efficient strategy concerning waiting time and cost per kilometer. As expected, by transporting more than one passenger at once, the shared strategy causes the lowest emissions, because the average driving distance per request is significantly lower compared to the other strategies. Surprisingly the costs per kilometer are slightly higher, which can be explained by the fact, that the parameter set which specifies the length of the allowed detour to pickup further passengers is not optimized yet. This leads to idle time for some vehicles, causing higher fleet base costs.
5 Conclusions

In this paper we presented SUMO4AV, a simulation environment that facilitates the simulation of SAV fleets in cities. It extends the SUMO package by making OpenStreetMap entities like POIs accessible during simulation runs and by the possibility to define different use case scenarios. We described in detail the complete workflow to create, customize, run and analyze the simulations. SUMO4AV is currently in a prototypical state available on GitHub\(^\text{10}\).

As proof-of-concept, we simulated a leisure scenario where an SAV fleet transports people in the evening after work from office or from home to locations like restaurants or cinemas. The scenario was successfully performed by importing OpenStreetMap data of the city center of Mannheim with the OSMWebWizard, by extracting map entities and creating the scenario with the SUMO4AV environment, by running the simulation with SUMO and by attaining first simulation results. We used three different routing strategies for the SAV fleet and several different fleet sizes. Due to stability problems during runtime the proof-of-concept had to be performed without base traffic.

Future work is planned to improve the usage of the environment and to perform larger studies on the routing strategies, especially for the shared strategy in order to optimize the parameter sets. A further improvement can be attained by implementing a combination of the shared and the look ahead strategy to unite the advantages of both strategies. Also, the stability problems by using base traffic needs to be investigated and solutions must be found. When these problems are solved, we want to apply SUMO4AV for other municipalities, in the ideal case also using further data from mobility studies. It is also planned to implement an additional feature to consider charging states and charging cycles to give SAV fleet operators more accurate results regarding the maximum degree of capacity utilization of an SAV fleet.

References


\(^{10}\)Url: https://github.com/keim-ls-esslingen/k4robofleet


