Integration of Driving- and Trafficsimulation

Martin Scheuchenpflug

BACHELORARBEIT

eingereicht am

Automotive computing

Universal Computing

in Hagenberg

im Juni 2024

Advisor:

FH-Prof. Dr. Gerald Ostermayer

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere. This printed copy is identical to the submitted electronic version.

Hagenberg, June 25, 2024

Martin Scheuchenpflug

Contents

Preface

Abstract

This should be a 1-page (maximum) summary of your work in English.

Kurzfassung

An dieser Stelle steht eine Zusammenfassung der Arbeit, Umfang max. 1 Seite. ...

Chapter 1

Introduction

1.1 Motivation

1.1.1 Driving simulation

The use of driving simulators is common practice among researchers for studying driving behaviors in scenarios that may include dangerous and unlikely situations[42, 54]. Utilization of driving simulators is primarily driven by the significant reduction in time and financial resources for engineers and developers, as evidenced by reference[42]. Additionally, it facilitates expeditious research and evaluation of innovative human-machine-interface designs (HMI) within the automotive industry[35]. One such simulator represents the “me”-perspective by focusing the simulation on the vehicle that is driven by a human in the simulator, thereby ensuring that this vehicle is modeled with great precision. One illustrative example of a well-known driving simulator is CARLA, which is an open-source, free-to-use driving simulator with a feature-rich Python API[11].

1.1.2 Microscopic traffic simulation

The use of microscopic traffic simulators is also a common methodology among researchers studying the dynamics of vehicular traffic, particularly in the context of intelligent transportation systems (ITS)[43]. One such microscopic traffic simulator employs (among other models like: lane-change, fuel-consumption, . . .) car-following models to regulate the acceleration and, consequently, the movement of each vehicle[47]. A variety of car-following models are available to use for modelling different types of vehicles. One notable example is the IDM (Intelligent Driver Model) (see Section 2.1.3) described by Springer et al., which assembles a complete and accident free models, that produces plausible acceleration profiles[47]. Moreover, as described in [47] these models are derived from a set of assumptions about real drivers such as keeping a “safe distance” from the leading vehicle, driving at a desired speed or preferring acceleration to be within a comfortable range, etc. As all the mentioned Parameters are like sensor-values to a modern ACC (adaptive cruse control) System with the model being the control system. Since these models are fully deterministic, they don‘t model the “errors” of human drivers.

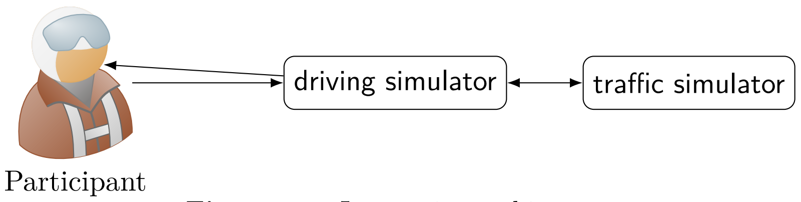


Figure 1.1: Integration architecture

1.1.3 Integrated simulation environment

The goal of investigating the impact of human drivers on traffic flow can be achieved by modeling those imperfections of a human driver. Using so called meta-models on top of typical vehicle-following models which modify the behavior to closely match a real human driver. Example for meta-models are the human driver model (HDC)[48] or the extended human driver model (EHDC)[23]. An additional potential methodology for the observation of human behavior is to engage directly in a scenario, which offers the benefit of eliminating the necessity for a model of the human driver. As a result of the direct participation of the human in the scenario, the impact of that behavior on the overall traffic flow can be observed. The aforementioned "me"-perspective is transformed into a holistic "we"-perspective through the utilization of an integrated simulation environment [35]. This kind of integration has proven very useful[31].

There are two types of integrations:

- **Offline integration:** For this integration a driving simulator and a traffic simulator are used seperatly. For example one could build the same scenario in both simulators and a human driver could participate in the scenario inside the driving simulator. Afterward the behavior of the human driver is recorded and imported into the traffic simulator, so the impact of the recorded behavior can be observed. The key disadvantages of this integration are, that it lacks the possibility of observing the impact of the driving behavior in real time and the model-controlled vehicles are not able to adapt to the human driver's behavior during the simulation.
- **Online integration:** The aforementioned disadvantages are rendered obsolete by implementation of an online integration. The most straightforward structure for such an integration would be to have the participant interact with the driving simulator, which is then synchronized to the traffic simulator. This can be achieved by connecting a driving simulator with a microscopic traffic simulator, as illustrated in Figure1.1. By running the simulators in sync, it is possible for the simulation controlled (by the traffic simulator) vehicles to react to the driving actions of the vehicle controlled by a human driver.

The remaining work focuses on the online integration of the simulators, unless otherwise specified.

1.2 Objective of the work

In this thesis, an integrated Simulation Environment is proposed, consisting of the driving simulator CARLA[11] and the microscopic traffic simulator TraffSim[1, 2]. The primary objective of this integration is to achieve a robust and synchronized environment, where the real-time interactions between human-driven and vehicular-follow-model-controlled vehicles unfold seamlessly.

The key challenges of the proposed integrated environment are:

- **Synchronisation of integration components:** One of the primary objectives of this work is to ensure seamless synchronisation between the driving and traffic simulators. This synchronisation involves real-time data exchange, particularly for online integration, where the actions of a human driver in the driving simulator should immediately impact vehicle behaviours in the traffic simulator. A significant challenge in this regard is the minimisation of latency, which could cause inconsistent or delayed responses between simulators.
- **Road network compatibility and flexibility:** Synchronizing road networks across the simulators is essential for a coherent simulation environment, as inconsistencies can lead to inaccuracies in traffic flow or navigation. Therefore, a procedure for exporting, importing and exchaning road networks between the simulators was implemented by the use of the OpenDrive**??** Standard.
- **Handling human driving behavior:** Human drivers often exhibit unpredictable behaviors, creating a need for adaptable models that can respond effectively to erratic or non-standard actions. This objective was partially archived by extrapolating the acceleration and speed of human-driven vehicle, for further use, this objective has to be readressed. One potential solution for this problem could be to use a vehicle-following, which models a human driver[23], to predict the next action taken by the human in charge of the vehicle.

1.3 Structure

This thesis is structured into multiple chapters, each addressing discrete aspects of the integration of driving and traffic simulators and the associated challenges.

The initial chapter, entitled “Introduction”, provides an overview of the subject, outlining the historical development of driving and traffic simulation, the individual functionalities of each, and the rationale for integrating them into a unified environment. Furthermore, it delineates the objectives and scope of the work.

The second chapter, entitled “Basics”, identifies the foundational studies of the field of traffic research, that are related to the discussed topic.

The third chapter, entitled “Related Work”, presents a review of existing research on driving and traffic simulation systems. It focuses on previously explored methods for integration, identifying both successful techniques and common obstacles encountered.

The fourth chapter, entitled “Methods”, provides a detailed account of the architectural design of the integrated simulation environment. It describes the technical framework and design decisions that have been taken in order to achieve synchronisation of the simulators, to facilitate real-time data exchange and to ensure compatibility between simulator platforms.

The fifth chapter, Results, presents the outcomes of the implemented integration, including a discussion of performance metrics, an analysis of the observed behaviours of the simulation components, and an evaluation of the solutions proposed for specific integration challenges.

The sixth chapter, entitled “Future Work”, identifies areas for further research and development. It mainly discusses future use cases for the proposed integrated environment in addition to a description of the necessary steps to transform the already working proof of concept into a usable tool for traffic research.

Chapter 2

Basics

The following chapter provides an overview of research and state-of-the-art in the domain of modelling and simulation of traffic systems. Section 2.1 examines foundational studies that have shaped current understanding of modelling traffic systems, focusing on the methodologies and findings that are most relevant to this thesis. Section 2.2 focuses on studies that are more closely related to the proposed simulation environment and the challenges that had to be solved in order to create one such integration. By mapping out the state of the art, this chapter establishes the context for the subsequent discussion of the proposed approach and methodology in Chapter 4.

2.1 Foundational studies

The field of traffic science is divisible into multiple different sub-categories by the time span of the observed events. As described by Treiber et al. [47] there are vehicle dynamics, traffic flow dynamics and transportation planning as the three major topics(see Table 2.1). These fields are divisible even further as stated in table 2.1. The proposed simulation environment would fall into the category of traffic flow dynamics using microscopic car following models. According to Treiber et al. this kind of simulation environments are best used to model reaction times, time gaps and the acceleration/breaking behaviors of vehicles in a traffic scenario. Driving behavior of vehicles in traffic flow dynamics are usually described by vehicle following (VF) models, which will be discussed further in the following sections.

Time scale	Field	Models	Aspects of traffic (examples)
$\leq 0.1s$	Vehicle dynamics	Sub-microscopic	Control of engine and breaks
1 s	Traffic flow dynamics	Car-following modells	Reaction time, time gap
10 s			Acceleration and deceleration
1 min		Macroscopic models	Cycle period of traffic lights
10 min			Stop-and-go waves
1 h	Transportation planning	Route assignment traffic demand	Peak hour
1 day			Daily demand pattern
1 year			Building/changing infrastructure
5 years		Statistics age pyramid	Socioeconomic structure
50 years			Demographic change

Table 2.1: Delimitation of traffic flow dynamics from vehicular dynamics and transportation planning[47]

2.1.1 Gazis-Herman-Rothery model

Vehicle following models have been researched for more than 70 years [29] and many such models have been developed since. One of the first widely known VF models is the Gazis-Herman-Rothery (GHR) model (see Eq 2.1), which described the acceleration of a vehicle n in a driving scenario with respect to the difference in speed Δv to the leading vehicle $n-1$ and the distance to the leading vehicle Δx, at a point earlier in time, with T being the reaction time of the driver[7].

$$a_n(t) = cv_n^m(t)\frac{\Delta v(t-T)}{\Delta x^{-1}(t-T)} \tag{2.1}$$

2.1.2 Gipps' model

Gipps states in the paper proposing his own VF model (Gipps´ model), that most VF models up until then (1981) have generally been in the form of EQ 2.2, where $\tau = T$. One example for this general form is the GHR model, some more of those models are found in publications dating back to that time: [5, 21, 27].

$$a_n(t+\tau) = l_n\frac{[v_{n-1} - v_n]^k}{[x_{n-1} - x_n]^m} \tag{2.2}$$

As pointed out by Philip A. Seddon, the fact, that the time interval between subsequent calculations was given by the reaction time was an undesired characteristic of these models[38], which could be overcome by storing a considerable amount of historical data, which was undesired at the time. The second pain point of these models, by today's standards maybe the more relevant point, was the existence of parameters l_n,k,m (EQ 2.2), that have no identifiable connection to driver or vehicle characteristics[13].

To address these deficits, Gipps proposed a new vehicle following model describing the speed of a vehicle at a point in time, in contrast to describing the acceleration of a vehicle at a point in time. The following form of the Gipps‘ model is not the form of the original publication, but the form from [47], which introduces a save speed v_{save} for simplification. As stated by Treiber et al. the model is conseptually unchanged. The GHR model calculates the speed of a vehicle with respect to the desired acceleration a, deceleration b, Desired speed v_0 and minimum distance s_0.

$$v_{save}(s, v_l) = -b * \Delta t + \sqrt{b^2 \Delta t^2 - v_l^2 + 2b(s - s_0)} \tag{2.3}$$

$$v(t + \Delta t) = min[v + a\Delta t, v_0, v_{save}(s, v_l)] \tag{2.4}$$

With the concept of the save speed depending on the distance to the front vehicle s and its speed v_l, the gipps‘ model assembles one of the simplest complete and accident-free models possible. The accident-free characteristic of the model is guaranteed with the assumptions that deceleration and reaction times are constant [47]. In essence the gipps‘ model chooses the minimum between the desired speed, the safe speed or the speed that max acceleration would result in the next time step. Problem with this model is, that as stated by Treiber et al. it produces an unrealistic acceleration profile.

2.1.3 Intelligent driver model

The time-continuous Intelligent driver model produces a realistic acceleration profile, since one of the requirements for forming the IDM is that the acceleration function $\dot{v}(s, v, v_l)$ is continuously differentiable in every three variables and thus producing smooth transitions between eg. breaking and acceleration phases. Another design criteria of the IDM is that the equilibrium distance[1]has to be larger than the "safe" distance $(v * T + v_0)$, with v being the current speed, T being the desired distance between vehicles in seconds and v_0 representing the "bumper-to-bumper" distance (min distance kept between vehicles), which ensures the accident-free characteristic of that model.

$$s^*(v, \Delta v) = s_0 + \max\left(0, vT + \frac{v\Delta v}{2\sqrt{ab}}\right) \tag{2.5}$$

$$\dot{v}(s, v, v_l) = a * \left[1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s}\right)^2\right] \tag{2.6}$$

The IDM (see EQ 2.6)is constructed from 2 pieces, the first part is comparing the current speed to the desired speed: $\left(\frac{v}{v_0}\right)^{\delta}$. The second part is comparing the current distance to the desired distance s^* (see EQ 2.5): $\left(\frac{s^*(v,\Delta v)}{s}\right)^2$ [44, 47].

The advantage of using an intuitive model like Gipps‘ model or the IDM, is that they are easy to configure by some simple understandable parameters (including vehicle and driver specific numbers). For example the configuration of the IDM consists of six

[1]The Distance, that is required between vehicles to stay in a steady-state equilibrium, which means, in a homogenous convoy the distance between all vehicles and the speed is the same. Furthermore, the acceleration has to be 0 for every vehicle to be in a steady-state eqilibrium. [44, 47]

parameters controlling the behavior of vehicles, with default values used by TraffSim (see Table 2.2) [1, 2, 47]. The Implementation in TraffSim includes two additional parameters, determining the maximum possible acceleration a_{max} and deceleration b_{max}, which act as a hard limit for the acceleration function. The value of b_{max} for example models the physical limit of the breaks of a car.

Parameter	TraffSim default Value	Acceptable Range
Comfortable acceleration a	2m/s^2	$0 < a < a_{max}$
Comfortable deceleration b	-2m/s^2	$b_{max} < b < 0$
Desired Speed v_0	25m/s	$v_0 > 0$
Minimum bumper-to-bumper distance s_0	2m	$s_0 > 0$
Time gap T	1.5s	$T > 0$
Acceleration exponent δ [2]	4	$\delta > 0$

Table 2.2: Default Values for IDM

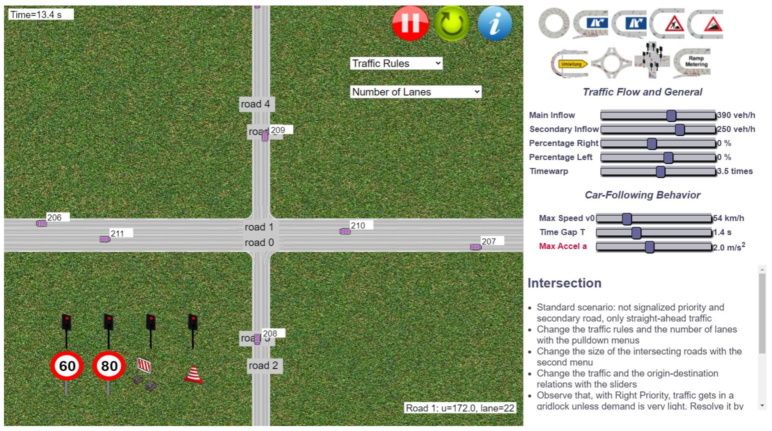
2.1.4 Micro traffic simulation

The aforementioned vehicle following models are an integral part of a traffic simulator as they are used to control the movement of vehicles along the road network. Since one such simulator is a key building block of the desired simulation environment, the next section focuses on the micro traffic simulation environments available in literature. Special focus will be placed onto the micro traffic simulator TraffSim [1, 2], developed in house at Hagenberg.

MovSim

The first notable example is MovSim [46], which is an open source, free to use traffic simulator developed by a team at TU Dresden. MovSim implements the IDM (see Section 2.1.3) among other vehicle following model which is responsable for the longitudinal movement of vehicles. For transversal movement of vehicles, MovSim utilizes the MOBIL model [20], which is a model for lane changing working alongside a variety of VF models. MovSim is primary source for sample implementations of VF models described by Treiber et al. (in [47]). A picture of the web based user interface is depicted in Figure 2.1.

[2]Dimensionless factor determining the rate at wich the vehicle‘s acceleration decreases as it approaches its desired velocity



Traffic Flow and General

Main Inflow 390 veh/h
Secondary Inflow 250 veh/h
Percentage Right 0 %
Percentage Left 0 %
Timewarp 3.5 times

Car-Following Behavior

Max Speed v0 54 km/h
Time Gap T 1.4 s
Max Accel a 2.0 m/s^2

Intersection

- Standard scenario: not signalized priority and secondary road, only straight-ahead traffic
- Change the traffic rules and the number of lanes with the pulldown menus
- Change the size of the intersecting roads with the second menu
- Change the traffic and the origin-destination relations with the sliders
- Observe that, with Right Priority, traffic gets in a gridlock unless demand is very light. Resolve it by

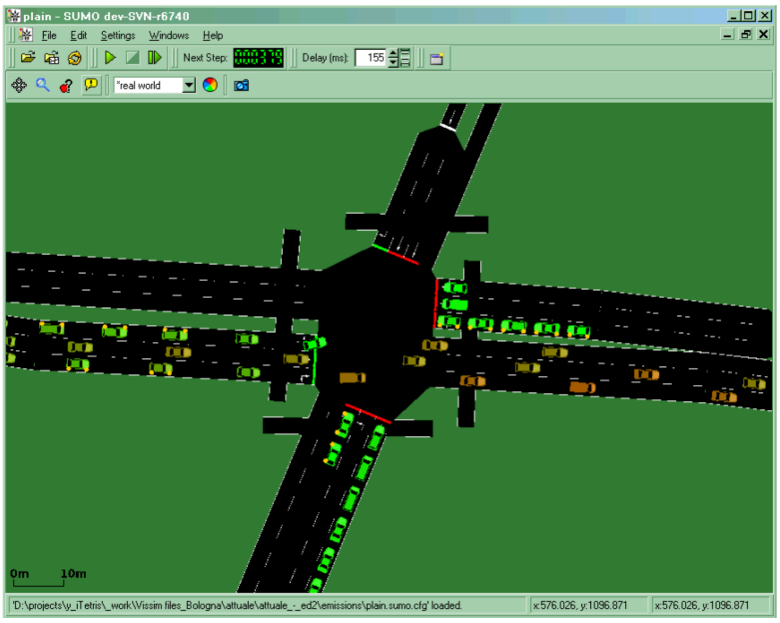
Figure 2.1: MovSim user interface [56]

Eclipse SUMO

One of the most popular traffic simulation environments is Eclipse SUMO [4], which is a suite of applications to model and simulate traffic scenarios. SUMOs development started in 2001 and has been made publicly available and open source in 2002. The simulation suite includes two applications used to import/generate road networks from a variety of sources.

1. *netgen*: The *netgen* tool is able to generate a fictional road network based on a set of input parameters. Different kinds of road networks can be generated by *netgen*: Grid Networks, Spider Networks and Random networks [57].
2. *netconvert*: This tool converts a road network, described by a variety of input formats, into a SUMO usable road network. Among other input file formats, netgen supports road networks from other simulators like Vissim [25] or MATsim [3, 49]. Furthermore, the importing filetypes like OpenStreetMap [16] and OpenDRIVE [28] is supported [4].

Many other applications are part of the SUMO suite that can be used for e.g. modelling V2X vehicles, generating routes from inputs like Source/Destination matrices or Loop detector statistics [4]. SUMO can be used in command-line mode for efficient simulation of a large number of parallel scenarios or in graphical UI mode for viewing a simulation while running (see Figure 2.2). During the runtime of simulations its possible to record statistics, that can be analyzed in post-processing.



'D:\projects\y_iTetris_work\Vissim files_Bologna\attuale\attuale_-_ed2\emissions\plain.sumo.cfg' loaded.

Figure 2.2: “Screenshot of the graphical user interface coloring vehicles by their CO2 emission.”[4]

TraffSim

TraffSim is a microscopic traffic simulator develeoped by a Team at University of Applied Scieneces Upper Austria. It‘s initial purpose, was to investigate the impact of intelligent traffic management on traffic flow, by using V2V-technology to dynamically reroute vehicles and thus reduce traffic jams. Since then TraffSim has gone through a major rewrite and getting the name TraffSim-B. TraffSim-A has been an Application developed in Java with a conventional desktop GUI, whereas TraffSim-B switched to Spring Boot[50] and Vaadin[15] as its Frontend, thus the new version is used in a web browser. The following work, if not otherwise stated, is referring to the TraffSim-B version.

TraffSim is built to use OpenStreetMap data as input for its road network, which is loaded and transformed into a working Scenario. To allow for computation of routes, a directed weighted cyclic graph is generated, that enables TraffSim to use a pathfinding Algorithm[12]. Furthermore, the road network is transformed into TraffSims own datastructures, using SUMOs *netconvert* tool. The transformed network is then stored into a local MongoDB instance In figure 2.3 the datastructures, used in TraffSim, to store the Road Network are visualized. In essence, a RoadSemgnet consists of one or more LaneSegments. RoadSegments can be connected directly to each other (1:1) or by a Junction. One such junction consists therefor of one or more Junction Connectors. Any DrivableArea is represented by the width and a reference line, which is composed of multiple points that are connected by straight lines. Which implies, that turns in the road network are also represented by small straight segements and thus do not appear to be smooth. The consequences of this will be discussed in more detail in Section

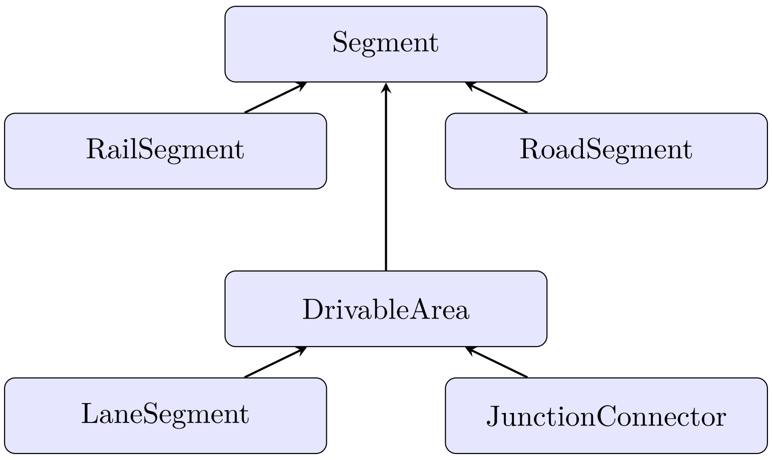


Figure 2.3: Class diagram of road network representation in TraffSim

The position of vehicles, moving along the road network, are represented by the DrivableArea they are on and by the distance, along the current segment, of their front relative to the start of the Segment. This implies, that vehicles in TraffSim esentially behave like trains on a railroad moving only longitudinal along the street and having no latitudinal movement.

TraffSim belongs to the class of time-discrete simulation environments, meaning, all changes in simulation state changes at an explicit point in time. The vehicle following models implemented in TraffSim are mostly given in time-continuous form, modelling the acceleration a_α of a vehicle α [26]. These time-continuous models have to be augmented by a numerical integration technique for usage, calculating the resulting speed and distance traveled by the vehicle with the newtonian equations of movement. Multiple algorithms are valid candidates for usage in traffic simulations according to [45] the default method for numerical integration of Ordinary Differential Equations (like the newtonian equations of movement) is the fourth order Range-Kutta scheme). For example the Range-Kutta integration scheme is used in. Although Range-Kutta is used in some examples like [19, 39], most simulators implement a simpler approach or do not state the used numerical integration at all[45]. Commonly either Euler‘s method is used for integration (e.g used in AIMSUN [8] or Eclipse SUMO [4]). Or a ballistic approach is used, this means, the acceleration during calculation of a subsequent timestep is assumed constant. This ballistic method is implemented in TraffSim, which results in the following calculations for speed v_α (see Equation 2.7)and position on the road s_α (see Equation 2.8) [26].

$$v_\alpha(t + \Delta t) = v_\alpha(t) + a_\alpha(t) * \Delta t \tag{2.7}$$

$$s_\alpha(t + \Delta t) = s_\alpha(t) + v_\alpha(t) * \Delta t + \frac{1}{2} v_\alpha(t) * (\Delta t)^2 \tag{2.8}$$

This distance calculated is than added to the vehicles position on the road. If the calculated distance added to the vehicles current position is less than the road length it is simply added, if it is greater, the vehicle is placed on the next road segment of the

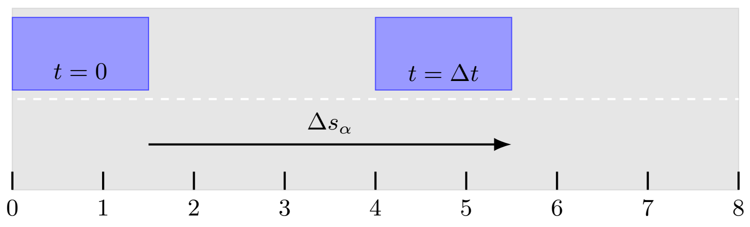


Figure 2.4: Example of vehicle position update

route. In the example shown in Figure 2.4, a Vehicle is at time $t = 0$ at position $s = 1.5$ on the road, in the next time step, the Vehicle moves by Δs_α. By doing this process over and over again, the Vehicles are "driving" along the roadnetwork in TraffSim.

The architecture of TraffSim (see Figure 2.5) is designed, that a great number of simulations, from the same scenario or different, can be run in parallel. Therefor TraffSim strictly separates the logic during a Simulation from the input data describing the Scenario. For each Scenario the input data is kept in the Simulation Model, which is stored in the local MongoDB instance. "It contains all the entities that are configurable, which include the road network, the route graph, repositories for (e.g. vehicles, routes, RSUs[3]), traffic generators and much more" [26]. Every Simulation Run[4] gets a copy of the Simulation Model, that is used to carry out the simulation. Since the model is copied into every instance, the same scenario can be run multiple times in parallel. This is especially useful when using batch simulations, which allow the user to vary parameters and investigate the impact of one/many parameter variations onto the traffic. Since it‘s impractical to visually track multiple simulations, when using batch mode the UI is disabled and just the statistics are generated and can be analyzed in post processing. The Statistics are exported using the HDF (.h5) file format and can be analyzed in Matlab or Python using the, in TraffSim included, libraries.

[3]Road Side Units

[4]A SimulationRun is a running instance of a Scenario

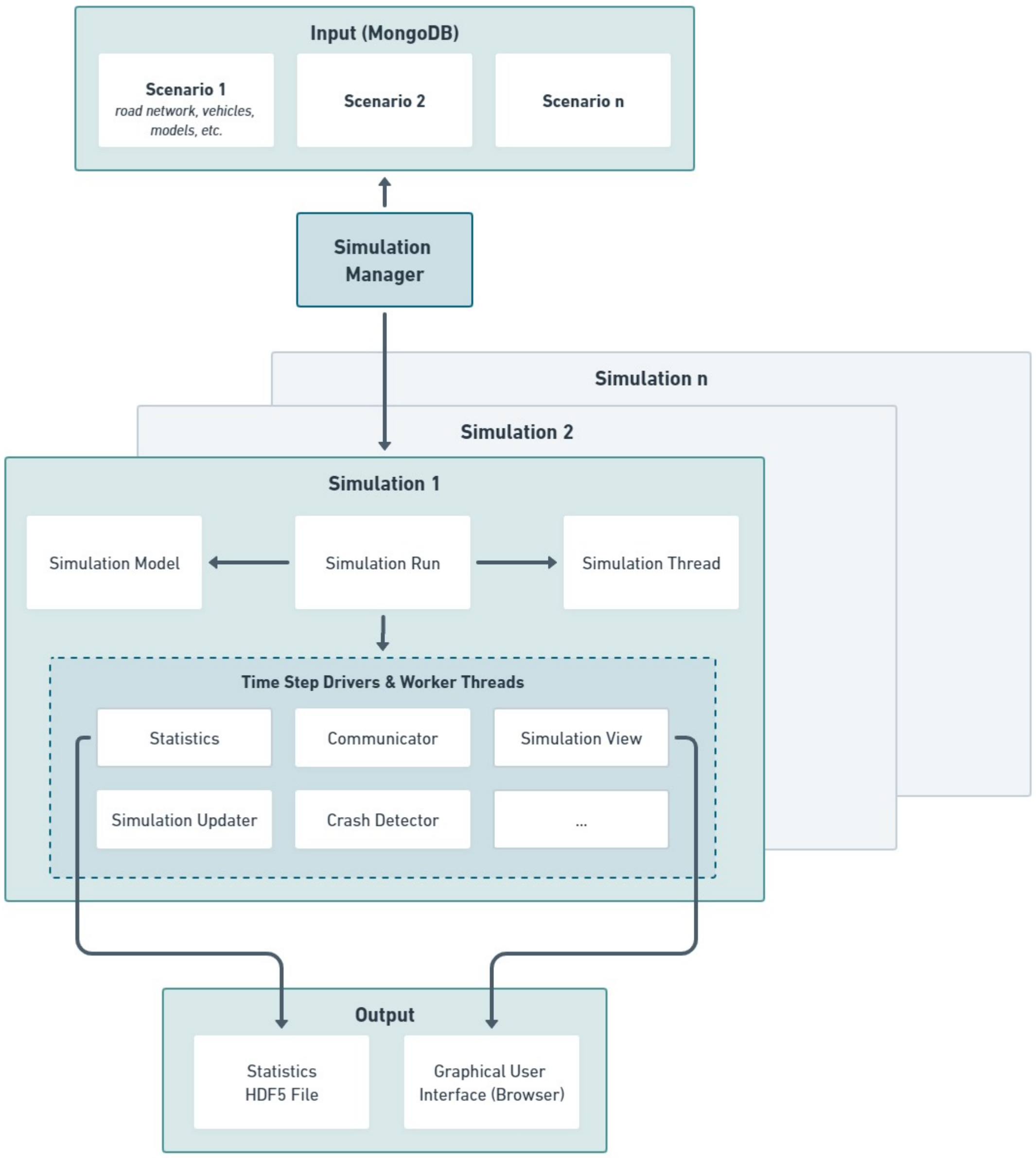


Figure 2.5: TraffSim Archtecture[26]

The control logic of many features of TraffSim is implemented in Time Step Divers, which is a mechanism inside TraffSim that allows the developer to register Components that get called repeatedly during the simulation. This could eiter be a fixed time interval (in simulation time) or during the calculation of every simulation step. There is a defined order, in wich the individual Time Step Drivers are called. One notable example of one such Time Step Driver is the SimulationUpdater, this module invokes the calculation of the acceleration of every vehicle on the RoadNetwork using its associated vehicle following models.

2.1.5 Driving simulation

Driving simulators and traffic simulators follow distinct but complementary objectives within the broader field of transportation research. While traffic simulators generally focus on modeling the aggregate or microscopic flow of vehicles across a network, driving simulators emphasize individual driver experience, behavior, and interaction with the vehicle or environment. This section delineates the main differences between these two categories of simulation tools and highlights reasons why both are integral to modern transportation research.

Level of Abstraction

In the category of traffic simulators, as discussed earlier, there are different levels of detail in simulations: microscopic-, mesoscopic- and macroscopic simulation. Where each of these kind of simulators adds a layer of abstraction, from microscopic models, tracking each vehicle and the interaction with others to macroscopic models, which represent the flow of vehicles as continuous fluid-like streams. Driving simulators are located somewhere below the microscopic traffic simulators, because they are modeling one vehicle‘s physical properties and behavior to a great extent. For example, a traffic simulator usually reduces the model of a vehicle‘s engine to some primitive parameters like max acceleration and efficiency, a driving simulator models the engine in much greater detail, including torque curves, power curves and many more engine specific parameters. In contrast to most traffic simulators, driving simulators do allow lateral movement of vehicles on the road. That means, that vehicles are not only bound to the center of the lane but are able to move free in 3 dimensions.

Fidelity and Realism

Driving simulators are designed to deliver realistic visual and physical environments. Many use game engines or high-end graphics libraries to simulate lighting, weather conditions, and road surfaces with high fidelity. Some simulators also support hardware-in-the-loop (HIL) configurations, which utilize real steering wheels, pedals, and motion platforms to replicate the physical sensations associated with driving [40]. These features are essential for various research applications, including human factors studies, driver assistance testing, and the prototyping of autonomous vehicle algorithms that depend on camera or LiDAR[10] data.

In contrast, traffic simulators typically utilize simplified graphical interfaces or purely abstract representations of road networks [46]. While visualization has evolved over time, it continues to prioritize monitoring flow patterns, such as queue formation or travel-time distributions, over replicating photorealistic driving experiences [11].

Driver in the loop

Driving simulators typically enable driver-in-the-loop experiments, in which a human or advanced AI driver interacts with the virtual environment in real time. This setup is vital for evaluating driver behavior under different road, traffic, or weather conditions and for testing advanced driver-assistance systems (ADAS) before deployment. Such real-time interactions yield deeper insights into cognitive and perceptual aspects that standard offline simulations can overlook. Coupled with advanced data collection techniques, it becomes possible to measure factors such as driver workload, reaction times, and overall situational awareness. In addition, driver-in-the-loop experiments provide a controllable environment for investigating emergency braking maneuvers, crash avoidance strategies, and the human–machine interface in partially automated vehicles [35, 36].

CARLA

In addition to some other notable examples like Madras [37] or Torcs [53], CARLA [11] is a free to use and open source driving simulator. CARLA is built using the widely adopted Unreal Engine [22] and thus developed in the programming language C++. The simulator is available on Windows and Linux as desktop application. CARLA is designed in a classical Client-Server architecture, as depicted in Figure 2.6, a server is started and multiple clients can be connected to one such server. Typically, these clients are implemented in Python, since CARLA provides a feature rich API Library to interact with it‘s server. When starting the carla Server one is provided a game window with a camera that is movable through the game world. The actual vehicle control has to be done via a python client application, there are some default examples for such client programs. The key advantage of this architecture is, that CARLA can be used in a vast variety of scenarios, the vehicles in the simulation can be controlled by humans, machine learning algorithms [24], static pathfinding algorithms or for example by real world vehicles in need for a digital twin [41]. A common use case of carla is to train neuronal networks to control a vehicle in the simulation environment [9, 34].

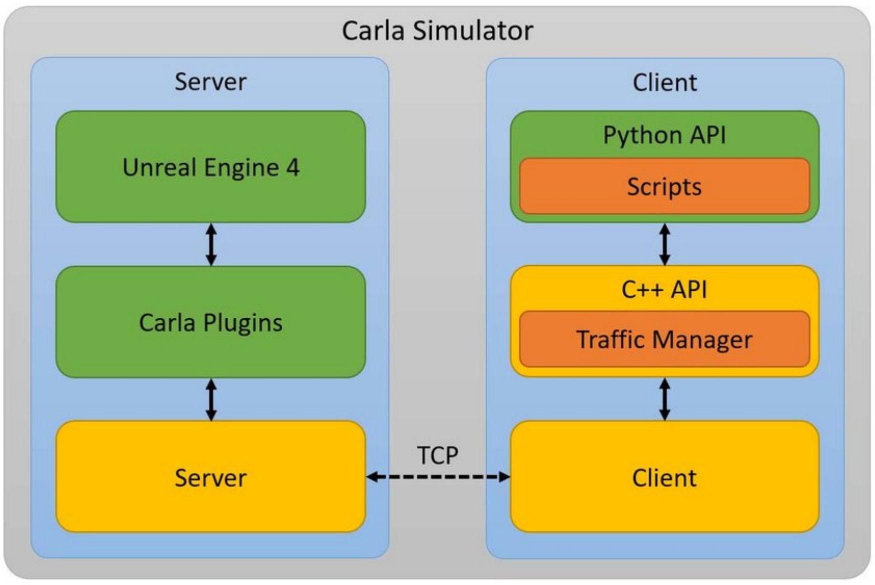


Figure 2.6: Carla client-server architecture [30]

2.2 Studies closely related to an integrated simulation environment

2.2.1 OpenDrive/OpenScenario

OpenDRIVE and OpenSCENARIO represent complementary standards for modeling and exchanging data in traffic simulation and autonomous driving research (see Figure 2.7). OpenDRIVE focuses on the precise description of road networks, including lane geometry, signage, and signal systems, while OpenSCENARIO targets the definition of traffic maneuvers, actor behaviors, and event sequences. Together, these formats enable consistent scenario creation and simulation across different software platforms, thereby reducing integration challenges and facilitating reproducible experiments. Through standardized road network specification and scenario scripting, researchers and developers can more efficiently collaborate on validation efforts for automated driving systems. These Standards use XML Files to represent data of road networks and driving maneuvers, the files are typically using the file ending ".xodr".

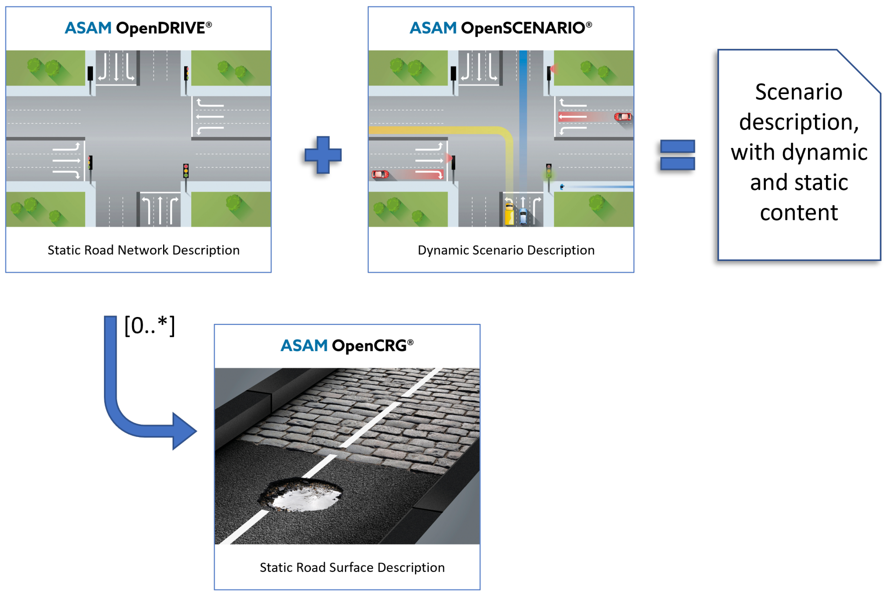

ASAM OpenDRIVE®
Static Road Network Description
ASAM OpenSCENARIO®
Dynamic Scenario Description
Scenario description, with dynamic and static content
[0..*]
ASAM OpenCRG®
Static Road Surface Description

Figure 2.7: Relationship between OpenDRIVE and OpenSCENARIO [28]

An OpenDRIVE file mainly consists of Junctions and Roads, described by the Listing 2.1 is a example of a described road in OpenDRIVE format. As visible from the example file, a road is modelled by many parameters like id, name, length, predecessor, successor, ... The tags "<planView>", "<lane>" and "<elevationProfile>" will be discussed in the following sections.

Listing 2.1: Example OpenDRIVE File [55]

```xml
road name="Road 0" length="3.95e+1" id="0" junction="-1">
    <link>
        <predecessor elementType="junction" elementId="106"/>
        <successor elementType="junction" elementId="281"/>
        </link>
            <type s="0.00e+0" type="town">
                <speed max="55" unit="mph"/>
            </type>
        <planView>
            <geometry s="0.00" x="2.11e+2" y="3.09e+2" hdg="-1.02e-2" length="3.95e+1">
                <line/>
            </geometry>
        </planView>
        <elevationProfile>
            <elevation s="0.00e+0" a="4.34e-3" b="0.00e+0" c="0.00e+0" d="0.00e+0"/>
        </elevationProfile>
        <lanes>
            <laneOffset s="0.00e+0" a="0.00e+0" b="0.00e+0" c="0.00e+0" d="0.00e+0"/>
            <laneSection s="0.000+0">
                <left>
                    <lane id="1" type="driving" level="false">
                        <width sOffset="0.00e+0" a="3.50e+0" b="0.00e+0" c="0.00e+0" d="0.00+0"/>
                        <roadMark sOffset="0.00e+0" type="none" material="standard" color="white" laneChange="none"/>
                        <roadMark sOffset="6.12e+0" type="solid" material="standard" color="white" width="1.25e-1" laneChange="none"/>
                        <userData>
                            <vectorLane travelDir="backward"/>
                        </userData>
                    </lane>
                </left>
                <center>
                    <lane id="0" type="none" level="false">
                        <roadMark sOffset="0.00e+0" type="none" material="standard" color="white" laneChange="none"/>
                        <roadMark sOffset="6.12e+0" type="broken" material="standard" color="yellow" width="1.25e-1" laneChange="none"/>
                    </lane>
                </center>
                <right>
                </right>
            </laneSection>
        </lanes>
    </road>

```

Reference line system

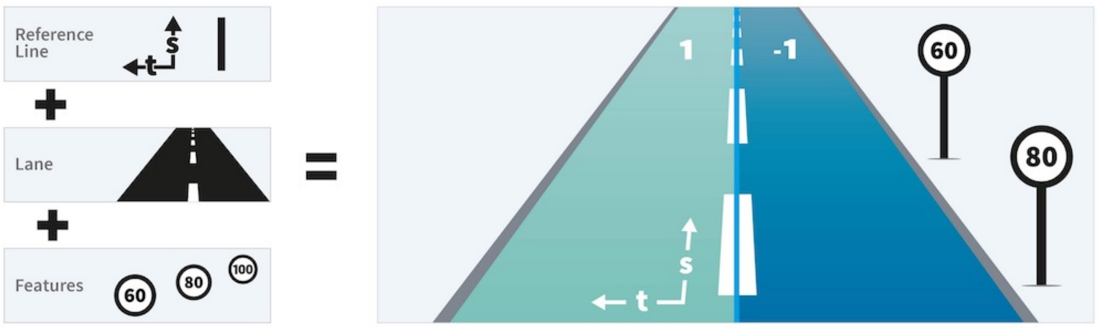


Figure 2.8: Reference Line System [28]

As depicted in the Figure 2.8, a road is modelled in 3 Layers.

1. Reference line *<planView>*: The course of a road segment is described by a line in the center of the Road. Since the curse of roads can have a variety of shapes, this line can be described with one of the following geometric elements[28]:

   - Straight line
   - Spirals
   - Arcs with constant curvature
   - Parametric cubic polynomials
   - Cubic polynomials

   This reference line is always placed at the base plane of the map. Therefore, it describes the road in 2 Dimensions, the elevation profile of the road is modelled by usage of the *<elevationProfile>* tag, that describes the elevation of a road segment using a 3rd order polynomial. Any mix of the geometric elements can be used to describe any possible course of a road as depicted in Figure 2.9.

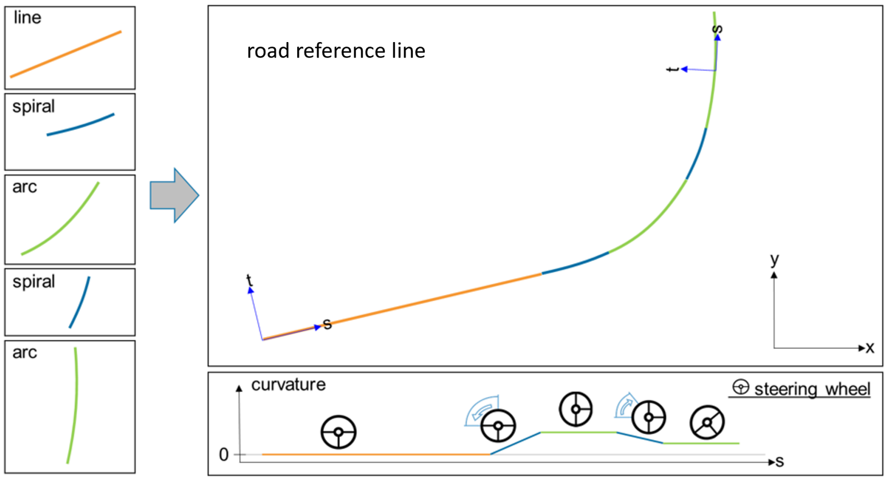


Figure 2.9: Combined geometrics [28]

2. Lane *<lane>*: One road can have multiple lanes on the right and left side, with each lane having an identifier. The lanes to the right of the center line are identified by negative numbers and the lanes on the left side with positive numbers. Lanes are added along the reference line and thus make the infinitely thin line into a surface with specific length and width. In addition to the width, lanes can be decorated with road markings.
3. Features: A modelled road can be enhanced with additional features like speed limits, traffic signals, etc.

For editing and viewing of openDRIVE files, tools like OpenDRIVE viewer (https://odrviewer.io/) or Mathworks' RoadRunner can be used. OpenDRIVE is also utilized by CARLA to import road networks into scenarios.

2.2.2 Map matching

Map matching is the process of aligning observed vehicle positions, typically recorded via GPS or other location-tracking sensors, to the correct roads or paths on a digital map. This step is essential in navigation, route planning, and autonomous driving applications, as it mitigates positioning errors and ensures that location data remain consistent with the underlying road network [17]. Various methodologies, including topological [14, 18, 52], geometric [6], and probabilistic approaches [32], have been proposed to address different accuracy and computational demands [33]. By anchoring vehicle positions to a known map, map matching facilitates reliable trajectory tracking, route deviation detection, and sensor fusion for advanced driver assistance systems.

In the field of geometric map matching algorithms, 3 categories can be distinguished:

1. Point-to-Point matching: the most straight forward way of matching an arbitrary position to a road network would be to find the closest node of the network. In most cases the Euclidean distance (see Equation 2.9), usually in $\mathbb{R}^2$, is used to determine the closest node, by iteratively checking the given position with each

node of the road network and storing the lowest value [6].

$$d(p,q) = \sqrt{(p_0 - q_0)^2 + (p_1 - q_1)^2 + \ldots + (p_n - q_n)^2} \tag{2.9}$$

This method works well when the road network consists of lots of nodes and the connection between those nodes are fairly short. If dealing with long straight roads, the method stops working.

2. Point-to-Curve matching: To conquer the mentioned problem, one might attempt to find the road segment N nearest to the given position instead of searching for the nearest node. In order to calculate the minimum distance of a point p to a road, the distance to each line segment A of the road has to be computed. Using Equation 2.10 (Line A is described by its Endpoints a and b), the minimum perpendicular distance between p and A can be calculated (see Figure 2.10,blue case) [6].

$$d_{line}(p,A) = \sqrt{\frac{((a_y - b_y) * c_x + (b_x - a_x) * c_y + (a_x * b_x - b_x * a_y))^2}{(a_y - b_y)^2 + (b_x - a_x)^2}} \tag{2.10}$$

If the intersection q of the perpendicular line of p and A is be not between a and b (see Figure 2.10, red case), the minimum distance of between (p,a) and (p,b) is taken as distance to the line [6].

$$D(p,A) = \begin{cases} d_{line}(p,A) & \max(d(q,a), d(q,b)) < d(a,b) \\ \min(d(p,a), d(p,b)) & otherwise \end{cases} \tag{2.11}$$

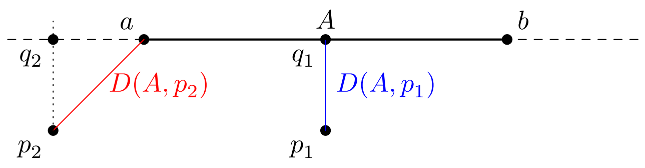


Figure 2.10: Point-to-curve matching [6]

3. Curve-to-curve mapping: In real world applications GPS positions often contain outliers due to a lack of satellites available for position. By using the previous method, these outliers would be mapped to another road segment and thus create impossible paths. The solution to this problem is to not match each one point to an arc, but to map the whole trajectory leading to the current point onto a valid arc of the road network.

Chapter 3

Related Work

Chapter 4

Methods

4.1 Architecture of the Integrated Simulation Environment

The proposed integration consists of the open-source Driving Simulator CARLA[11] and the microscopic traffic simulator TraffSim[1, 2], which was developed at the University of Applied Sciences Upper Austria. To connect the two simulators into one integrated environment, a gRPC[51] API was added to the microscopic traffic simulator, as well as a REST (Representational State Transfer) Endpoin. The gRPC-API is used to transfer data between the simulators during a running Simulation, as well as to transfer control signal (such as: Pausing the Simulation). The REST-Endpoint is used by the driving simulator to request the map used for the scenario. This map is requested in OpenDrive Format (see Section4.5).

4.1.1 Vehicles in TraffSim

In TraffSim there are a few different Types of Vehicles (see Figure

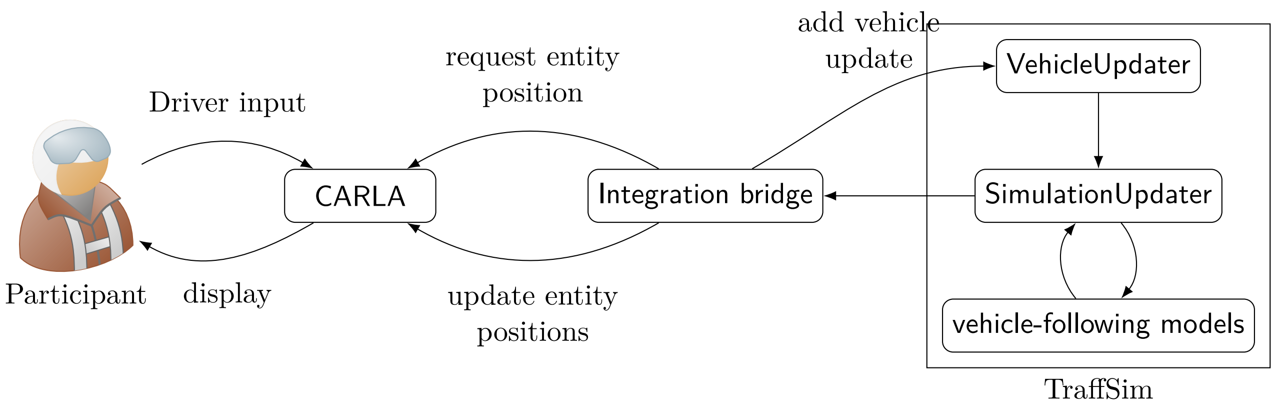


Figure 4.1: Data roundtrip

4.1.2 Simulation step synchronisation

The human driver employs the Carla interface to engage with the integrated simulation environment (see Figure4.1). It is essential that the driving behaviour of the simulation-controlled vehicle appears natural to the human driver. Therefore, the time between driver input and the visual output of the driving simulator must be minimised (data round-trip time). It is essential that the driver's input is accurately captured by Carla and that the human-driven vehicle is precisely simulated in order to create a realistic driving experience for the driver. As the two simulators are not synchronised with regard to their tick rates, the integration bridge requests the current position, speed and acceleration of the vehicle and a vehicle update is added to the update queue of the HumanDrivenVehicleUpdater within the traffic simulator. Upon calculation of the subsequent simulation step in the traffic simulator, all vehicle updates that have been queued are executed, thereby updating the human-driven vehicle. Therefore, the delta time between simulation steps should be selected as low as possible, in order to minimise the data round-trip time.

4.1.3 Real time simulation

Although t must not be lower than the time it takes for the simulation step to be calculated T. To bind the simulation time to the real time, the easiest way would be to set the time step size to the time duration it took for the last simulation step to compute:

$$t_x = T_{x-1}$$

The problem with that approach is, that during a simulation in TraffSim it should be avoided to change the simulation step t, so another approach is used, by adding a delay after each simulation step to compensate for the difference between t and T. It‘s important for this technique to work, the condition $t \geq T$ has to be met. In every simulation step the added delay d has to be calculated and thus t can be static:

$$d_x = t - T_x$$

4.2 TraffSim API

To interact with the microscopic traffic simulator TraffSim, a gRpc[51] API was specified and developed, which allows the client to call the functions described in Listing4.1. The Api was designed to allow the client to control all the functions that are important during a simulation session and also to sync the two simulators.

Listing 4.1: TraffSim API proto

```
service TraffSimController {
    rpc requestVehicleData (ActiveSimulationRun)
        returns (VehicleData);
    rpc notifyFreeDrivingVehicle(FreeDrivingVehicleUpdateRequest)
        returns(VoidMessage){}
    rpc simulationAllowTswConnection (ActiveSimulationRun)
        returns (BoolMessage) {}
    rpc getAvailableFreeDrivingVehicles(ActiveSimulationRun)
        returns(AvailableFreeDrivingVehicles){}
    rpc getAvailableScenarios (VoidMessage)
        returns (AvailableScenarios){}
    rpc getOpenSimulations (VoidMessage)
        returns (ActiveSimulationRuns){}
    rpc getSimulationState (ActiveSimulationRun)
        returns (SimulationState){}
    rpc pauseSimulation (ActiveSimulationRun)
        returns (VoidMessage){}
    rpc startSimulation (ActiveSimulationRun)
        returns (VoidMessage){}
    rpc setSimulationResolution (SimulationResolutionRequest)
        returns (VoidMessage){}
    rpc setSimulationDelay (SimulationDelayRequest)
        returns (VoidMessage){}
}
```

4.3 Mapped dummy vehicles

In TraffSim, there are several different types of vehicles (see Figure).The human-driven vehicles are able to traverse the entire map and are not constrained to the road network, whereas the simulation-controlled vehicles are constrained to the road network.

In order for the vehicle-following models to react to the human driver, it is necessary to represent the human-driven vehicle on the “rail-like” road network. Dummy vehicles are placed on the road network at each time step to enable the model-controlled vehicles to interact with the human-controlled vehicle. A map-matching algorithm is employed to identify the locations where dummy vehicles should be placed.

4.3.1 Map matching

To determine the position, a dummy vehicle has to be placed on the road network, a map-matching algorithm is used.

4.4 Speed and Acceleration projection

In order to allow the vehicle-following models to work, the human driven vehicle‘s speed $\vec{v_h}$ and acceleration $\vec{a_h}$ need to be mapped onto the dummy vehicle. Therefore $\vec{a_h}$ and $\vec{v_h}$ are projected onto the tangent vector of the nearest road segment $\vec{R}$. Since a mapped vehicle is bound to the road network and thus traveling “on rails”, the speed and acceleration can be expressed as scalars (a_m and v_m), which act like the magnitude of the

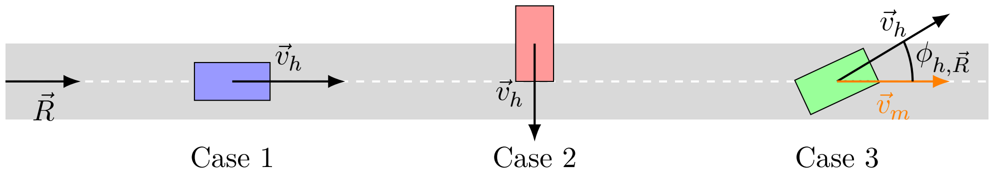


Figure 4.2: Cases of vehicles traveling on road network

speed vector in the direction of the current Road segment. The idea behind this mapping is to allow model-controlled vehicles to react to human-controlled vehicles, even if they are not traversing the road network as expected. To achieve the desired behavior some cases have to be considered, the cases described below can be seen in Figure4.2 :

Case 1: Vehicle driving on the nearest road. In this case the speed and acceleration vector of the mapped vehicle should be equal to the speed and acceleration of the human-controlled vehicle.

$$\vec{v_h} \times \vec{R} = 0 \Rightarrow v_m = |\vec{v_h}|$$
$$\vec{a_h} \times \vec{R} = 0 \Rightarrow a_m = |\vec{a_h}|$$

Case 2: Vehicle crossing the nearest road. In the case of a vehicle crossing the nearest road, the speed of the mapped vehicle should be zero. This is used to block the road for upcoming vehicles when one such human-controlled crosses the road.

$$\vec{v_h} \cdot \vec{R} = 0 \Rightarrow v_m = 0$$
$$\vec{a_h} \cdot \vec{R} = 0 \Rightarrow a_m = 0$$

Case 3: Vehicle moving on the road on an angle. Like in the 2 cases before the speed and acceleration of the mapped vehicle should correspond to the component of the speed and acceleration vector of the human-controlled vehicle (see Figure4.2) In this case the angle between the road and the vehicles speed and acceleration vectors has to be calculated.

$$\cos(\phi_{h,\vec{R}}) = \frac{\vec{v_h} \cdot \vec{R}}{|\vec{v_h}||\vec{R}|}$$

To combine all the cases above the speed and acceleration are projected onto $\vec{R}$:

$$v_m = |\vec{v_h}| * \cos(\phi_{h,\vec{R}}) = \frac{\vec{v_h} \cdot \vec{R}}{|\vec{R}|}$$
$$a_m = |\vec{a_h}| * \cos(\phi_{h,\vec{R}}) = \frac{\vec{a_h} \cdot \vec{R}}{|\vec{R}|}$$

4.5 Road network export and sync

Chapter 5

Results

Chapter 6

Future work

References

Literature

[1] Christian Backfrieder, Christoph F Mecklenbräuker, and Gerald Ostermayer. “TraffSim–A Traffic Simulator for Investigating Benefits Ensuing from Intelligent Traffic Management”. In: *2013 European Modelling Symposium*. IEEE. 2013, pp. 451–456 (cit. on pp. 3, 8, 22).

[2] Christian Backfrieder, Gerald Ostermayer, and Christoph F Mecklenbräuker. “TraffSim-A traffic simulator for investigations of congestion minimization through dynamic vehicle rerouting”. *ResearchGate* 15.4 (2014), pp. 38–47 (cit. on pp. 3, 8, 22).

[3] Michael Balmer et al. “MATSim-T: Architecture and simulation times”. In: *Multi-agent systems for traffic and transportation engineering*. IGI Global, 2009, pp. 57–78 (cit. on p. 9).

[4] Michael Behrisch et al. “SUMO - Simulation of Urban MObility An Overview”. In: 2011. URL: https://api.semanticscholar.org/CorpusID:433701 (cit. on pp. 9–11).

[5] JG Bender and RE Fenton. “On the flow capacity of automated highways”. *Transportation Science* 4.1 (1970), pp. 52–63 (cit. on p. 6).

[6] David Bernstein, Alain Kornhauser, et al. “An introduction to map matching for personal navigation assistants” (1996) (cit. on pp. 19, 20).

[7] Mark Brackstone and Mike McDonald. “Car-following: a historical review”. *Transportation Research Part F: Traffic Psychology and Behaviour* 2.4 (1999), pp. 181–196. DOI: https://doi.org/10.1016/S1369-8478(00)00005-X (cit. on p. 6).

[8] Jordi Casas et al. “Traffic simulation with aimsun”. *Fundamentals of traffic simulation* (2010), pp. 173–232 (cit. on p. 11).

[9] Felipe Codevilla et al. “Exploring the limitations of behavior cloning for autonomous driving”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9329–9338 (cit. on p. 15).

[10] RTH Collis. “Lidar”. *Applied optics* 9.8 (1970), pp. 1782–1788 (cit. on p. 14).

[11] Alexey Dosovitskiy et al. “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the 1st Annual Conference on Robot Learning*. Ed. by Sergey Levine, Vincent Vanhoucke, and Ken Goldberg. Vol. 78. Proceedings of Machine Learning Research. PMLR, Nov. 2017, pp. 1–16. URL: https://proceedings.mlr.press/v78/dosovitskiy17a.html (cit. on pp. 1, 3, 14, 15, 22).

[12] Daniel Foead et al. “A systematic literature review of A* pathfinding”. *Procedia Computer Science* 179 (2021), pp. 507–514 (cit. on p. 10).

[13] Peter G Gipps. “A behavioural car-following model for computer simulation”. *Transportation research part B: methodological* 15.2 (1981), pp. 105–111 (cit. on p. 6).

[14] Joshua S Greenfeld. “Matching GPS observations to locations on a digital map”. In: *Transportation research board 81st annual meeting*. Vol. 22. 2002, pp. 576–582 (cit. on p. 19).

[15] Marko Grönroos. *Book of vaadin*. Lulu. com, 2009 (cit. on p. 10).

[16] Mordechai Haklay and Patrick Weber. “Openstreetmap: User-generated street maps”. *IEEE Pervasive computing* 7.4 (2008), pp. 12–18 (cit. on p. 9).

[17] Zhenfeng Huang et al. “Survey on vehicle map matching techniques”. *CAAI Transactions on Intelligence Technology* 6.1 (2021), pp. 55–71 (cit. on p. 19).

[18] Rajashri R Joshi. “A new approach to map matching for in-vehicle navigation systems: the rotational variation metric”. In: *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 01TH8585)*. IEEE. 2001, pp. 33–38 (cit. on p. 19).

[19] Jevgenijs Kaupužs et al. “Applications to traffic breakdown on highways”. In: *Progress in Industrial Mathematics at ECMI 2002*. Springer. 2004, pp. 133–138 (cit. on p. 11).

[20] Arne Kesting, Martin Treiber, and Dirk Helbing. “General lane-changing model MOBIL for car-following models”. *Transportation Research Record* 1999.1 (2007), pp. 86–94 (cit. on p. 8).

[21] Gentry Lee. “A generalization of linear car-following theory”. *Operations research* 14.4 (1966), pp. 595–606 (cit. on p. 6).

[22] Michael Lewis and Jeffrey Jacobson. “Game engines”. *Communications of the ACM* 45.1 (2002), p. 27 (cit. on p. 15).

[23] Manuel Lindorfer, Christoph F. Mecklenbräuker, and Gerald Ostermayer. “Modeling the Imperfect Driver: Incorporating Human Factors in a Microscopic Traffic Model”. *IEEE Transactions on Intelligent Transportation Systems* 19.9 (2018), pp. 2856–2870. DOI: 10.1109/TITS.2017.2765694 (cit. on pp. 2, 3).

[24] Xiruo Liu et al. “Synthetic dataset generation for adversarial machine learning research”. *arXiv preprint arXiv:2207.10719* (2022) (cit. on p. 15).

[25] Nicholas E Lownes and Randy B Machemehl. “VISSIM: a multi-parameter sensitivity analysis”. In: *Proceedings of the 2006 Winter Simulation Conference*. IEEE. 2006, pp. 1406–1413 (cit. on p. 9).

[26] Research Group Networks and Mobility. *TraffSim Documentation*. University of Applied Sciences Upper Austria. Dec. 2024 (cit. on pp. 11–13).

[27] Gordon Frank Newell. “Nonlinear effects in the dynamics of car following”. *Operations research* 9.2 (1961), pp. 209–229 (cit. on p. 6).

[28] *OpenDRIVE*. Association for Standardization of Automation and Measuring Systems (cit. on pp. 9, 16, 18, 19).

[29] Louis A Pipes. "An operational analysis of traffic dynamics". *Journal of applied physics* 24.3 (1953), pp. 274–281 (cit. on p. 6).

[30] Patrick Pirri et al. "Towards cooperative maneuvering simulation: Tools and architecture". In: *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE. 2021, pp. 1–6 (cit. on p. 15).

[31] Vincenzo Punzo and Biagio Ciuffo. "Integration of driving and traffic simulation: Issues and first solutions". *IEEE transactions on intelligent transportation systems* 12.2 (2010), pp. 354–363 (cit. on p. 2).

[32] Mohammed A Quddus. "High integrity map matching algorithms for advanced transport telematics applications". PhD thesis. Imperial College London London, 2006 (cit. on p. 19).

[33] Mohammed A Quddus, Washington Y Ochieng, and Robert B Noland. "Current map-matching algorithms for transport applications: State-of-the art and future research directions". *Transportation research part c: Emerging technologies* 15.5 (2007), pp. 312–328 (cit. on p. 19).

[34] Francisco Ramos. "Autonomous Kart Racing: End-to-End Self-Driving with Behavioral Cloning in CARLA Simulator" () (cit. on p. 15).

[35] Andreas Riegler, Andreas Riener, and Gerald Ostermayer. "From Me to We: Combining Driving Simulation and Traffic Simulation for Holistic Usability and Safety Research". In: *Adjunct Proceedings of the 15th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. 2023, pp. 294–296 (cit. on pp. 1, 2, 15).

[36] Andreas Riener et al. "Driver in the loop: Best practices in automotive sensing and feedback mechanisms". *Automotive user interfaces: creating interactive experiences in the car* (2017), pp. 295–323 (cit. on p. 15).

[37] Anirban Santara et al. "Madras: Multi agent driving simulator". *Journal of Artificial Intelligence Research* 70 (2021), pp. 1517–1555 (cit. on p. 15).

[38] Philip A Seddon. "A program for simulating the dispersion of platoons of road traffic". *Simulation* 18.3 (1972), pp. 81–90 (cit. on p. 6).

[39] Daisuke Shamoto et al. "Car-following model with relative-velocity effect and its experimental verification". *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 83.4 (2011), p. 046105 (cit. on p. 11).

[40] Gregor Sievers et al. "Driving simulation technologies for sensor simulation in sil and hil environments". In: *Proceedings of the DSC*. 2018 (cit. on p. 14).

[41] Charles Steinmetz et al. "Digital Twins modeling and simulation with Node-RED and Carla". *IFAC-PapersOnLine* 55.19 (2022), pp. 97–102 (cit. on p. 15).

[42] Thomas Nguen That and Jordi Casas. "An integrated framework combining a traffic simulator and a driving simulator". *Procedia - Social and Behavioral Sciences* 20 (2011), pp. 648–655. DOI: https://doi.org/10.1016/j.sbspro.2011.08.072 (cit. on p. 1).

[43] Tomer Toledo et al. “Microscopic traffic simulation: Models and application”. In: *Simulation Approaches in Transportation Analysis: Recent Advances and Challenges*. Springer, 2005, pp. 99–130 (cit. on p. 1).

[44] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. “Congested traffic states in empirical observations and microscopic simulations”. *Phys. Rev. E* 62 (2 Aug. 2000), pp. 1805–1824. DOI: 10.1103/PhysRevE.62.1805 (cit. on p. 7).

[45] Martin Treiber and Venkatesan Kanagaraj. “Comparing numerical integration schemes for time-continuous car-following models”. *Physica A: Statistical Mechanics and its Applications* 419 (2015), pp. 183–195 (cit. on p. 11).

[46] Martin Treiber and Ame Kesting. “An Open-Source Microscopic Traffic Simulator”. *IEEE Intelligent Transportation Systems Magazine* 2.3 (2010), pp. 6–13. DOI: 10.1109/MITS.2010.939208 (cit. on pp. 8, 14).

[47] Martin Treiber and Arne Kesting. *Traffic flow dynamics*. Vol. 227. Springer, 2013, p. 228 (cit. on pp. 1, 5–8).

[48] Martin Treiber, Arne Kesting, and Dirk Helbing. “Delays, inaccuracies and anticipation in microscopic traffic models”. *Physica A: Statistical Mechanics and its Applications* 360.1 (2006), pp. 71–88 (cit. on p. 2).

[49] Kay W Axhausen, Andreas Horni, and Kai Nagel. *The multi-agent transport simulation MATSim*. Ubiquity Press, 2016 (cit. on p. 9).

[50] Craig Walls. *Spring Boot in action*. Simon and Schuster, 2015 (cit. on p. 10).

[51] Xingwei Wang, Hong Zhao, and Jiakeng Zhu. “GRPC: A communication cooperation mechanism in distributed systems”. *ACM SIGOPS Operating Systems Review* 27.3 (1993), pp. 75–86 (cit. on pp. 22, 23).

[52] Christopher E White, David Bernstein, and Alain L Kornhauser. “Some map matching algorithms for personal navigation assistants”. *Transportation research part c: emerging technologies* 8.1-6 (2000), pp. 91–108 (cit. on p. 19).

[53] Bernhard Wymann et al. “Torcs, the open racing car simulator”. *Software available at http://torcs. sourceforge. net* 4.6 (2000), p. 2 (cit. on p. 15).

[54] Rachael A. Wynne, Vanessa Beanland, and Paul M. Salmon. “Systematic review of driving simulator validation studies”. *Safety Science* 117 (2019), pp. 138–151. DOI: https://doi.org/10.1016/j.ssci.2019.04.004 (cit. on p. 1).

Online sources

[55] URL: https://github.com/carla-simulator/carla (cit. on p. 17).

[56] Gabriel Plassat. *MovSim User Interface*. Feb. 2022. URL: wiki.lafabriquedesmobilites.fr (cit. on p. 9).

[57] *Sumo User Documentation*. Dec. 2024. URL: https://sumo.dlr.de/ (cit. on p. 9).

Check Final Print Size

— Check final print size! —

width = 100mm
height = 50mm

— Remove this page after printing! —