HAW
HAMBURG

MASTERTHESIS
Julius Weyl

Developing a generic multi-agent car model to simulate road traffic with MARS

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Julius Weyl

Developing a generic multi-agent car model to simulate road traffic with MARS

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Thomas Clemen
Zweitgutachter: Prof. Dr.-Ing. Marina Tropmann-Frick

Eingereicht am: 19. Juni 2019

Julius Weyl

Thema der Arbeit

Developing a generic multi-agent car model to simulate road traffic with MARS

Stichworte

Verkehrssimulation, Multi-Agenten Modell, MARS

Kurzzusammenfassung

Die vorliegende Arbeit beschreibt die Entwicklung eines generischen, agenten-basierten Auto Modells für das MARS System. Aufbauend auf dem *Intelligent Driver Model* wird eine Reihe von Agentenregeln entworfen die es den autonomen Agenten erlaubt in zwei verschiedenen Teilen der Welt zu fahren. Das erste Szenario liegt in Hamburg Altona wo, basierend auf demografischen Daten über die Bevölkerung, der Verkehr im Verlauf eines Tages simuliert wird. Für das zweite Simulationsszenario wird das Modell and die Verkehrsregeln von Südafrika angepasst. Die Ergebnisse der beiden Simulationsszenarien werden analysiert um herauszufinden ob das Modell wirklich generisch entwickelt wurde.

Julius Weyl

Title of Thesis

Developing a generic multi-agent car model to simulate road traffic with MARS

Keywords

Traffic Simulation, Multi-Agent Model, MARS

Abstract

This thesis describes the development of a generic, agent-based car model for the MARS system. Based on the *Intelligent Driver Model* a set of agent rules is designed to allow autonomous car agents to drive in two distinct parts of the world. The first scenario is located in Hamburg Altona where the traffic over the course of a day is simulated based on demographic data about the people living there. For the second simulation scenario the model is adapted to the traffic rules of South Africa. The results from the two simulation scenarios are examined to assert whether the model has been build truly generic.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Introduction

Traffic research been a vital field of study over decades and reaches back as far as 1934 [7]. As road traffic evolved from horse carriages to cars, its volume climbed steadily over the past centuries. With the rise and growth of large cities, traffic is now gaining even more importance. Today, more people than ever live in urban areas and the trend keeps up [27]. At the same time the amount of car ownership isn't declining but rather growing. The total amount of cars owned in Germany has risen to 46 million. The expansion rate of 2017 alone, accounted for more than 500.000 cars [3]. The amount of traffic that arises from that trend overwhelms the traffic infrastructure increasingly. Especially cities are effected where space is inherently limited.

The town of Hamburg as Germanys second biggest city, experiences these problems on a regular basis. Despite a wide ranging network of public transportation and a growing amount of bike paths, the road traffic exceeds the capacities which leads to constant traffic jams. Solving these complex issues is by no means an easy task but one that cannot be ignored. For now, the existing network of roads has to be used to its maximum. For the future, new ways of traffic research have to be applied so that a modern network can be developed that balances traffic and living quality. While this topic isn't new, the way city planners do this has to change.

One way of planning changes to the road network are simulations. They are an essential part in traffic research and have been applied to traffic planning in various levels of detail. Depending on the use-case, either macroscopic, mesocopic or microscopic models are used to re-create common traffic situations. To this day, most traffic models in use work on a macroscopic scale that aim at emulating real-life measurements on roads as streams of cars. Once the flow of traffic has been measured, these models try to reproduce it by modeling streets as entities with a certain flow capacity. As long as the flow is lower than the capacity no jam occurs, once this limit is exceeded the traffic comes to a halt. This technique works well to study a known traffic network under known conditions as it emulates the measured streams. It is impossible though, to predict future conditions or calculate future traffic volume since the model operates based on a status quo. There is no information about individual behavior involved so that the model only works as long the traffic patterns stay the same.

Microscopic models, on the other hand, try to assess traffic from the bottom up. Instead of modeling traffic as a stream and roads as pipes with flow capacities, every traffic participant is brought to life. The traffic itself is thereby comprised of individuals that act and react to current conditions. Agent based simulations are a natural fit for tasks where intelligent behavior has to be resembled since they allow for dynamic changes during the simulation [11, 21, 2]. For traffic simulations that means that once the conditions change, the traffic participants adapt their behavior and alter the course of the simulation. This dynamic allows to answer questions that couldn't be answered with models that merely recreate bygone conditions [8].

Creating microscopic models requires a lot of information about individual driving behavior in order to resemble such during a simulation. Before an agent can drive on virtual roads, all aspects of driving, including the abidance of traffic code, have to be modeled. For agent-based simulations this means translating driving behavior in a set of rules that the agents can follow. Once these rules have been established, they have to be implemented in a multi-agent framework like the MARS system.

1.1 Traffic research with MARS

MARS is the abbreviation for 'Multi-Agent Research and Simulation' and the name of a research group located at the University of Applied Sciences Hamburg. The groups focus is on agent-based modeling and simulation where they worked on different topics ranging from socio-ecological work, over pedestrian evacuation scenarios to disease spread models [10]. Traffic research has become part of the groups focus in 2017 where the first steps were taken towards implementing a traffic model for Hamburg. Today, the MARS group is working on traffic related issues as part of the *Smart open Hamburg* project (http://www.smartopenhamburg.de/). In the long term the objective is to develop a multimodal model of Hamburgs traffic with pedestrians, bikes, public transport and street traffic. With the framework initially lacking means of simulating streets, new components were developed to enable this. The *Spatial Graph Environment*, a data structure based on a custom graph representation was developed and integrated into the MARS System.

After developing this required component and publishing the first results [26] the project is now becoming due for new extensions.

A prototype was designed as performance test for the newly developed Spatial Graph Environment. Looking into existing models [4, 12] for traffic simulations, a car-following approach was chosen. One of the most renowned ways to simulate individual traffic behavior in terms of driving physics is the so called *Intelligent Driver Model* [23]. It was selected from a group of car-following models capable of reproducing realistic driving behavior. Over the past decades many models were proposed with the IDM currently being the most accurate in relation to the amount of input variables. The first implementation was a simplified version of the *Intelligent Driver Model* with static values for acceleration, breaking and safe headway. When approaching an intersection, the agents chose a random adjacent road and continued driving there. Even though this model didn't reproduce realistic driving, it was detailed enough to be used in benchmarking scenarios for the *Spatial Graph Environment.*

During the design phase of the Smart Open Hamburg project, the goal was set do research around urban mobility. With the initial prototypes in place, the project evolved into aiming at simulating selected parts of Hamburg. This model would include the cities modal split consisting of public transport, pedestrians and street traffic. Since this is a comprehensive undertaking it was decided split the agent-based model three-ways into the described modes of transport. The purpose of this thesis is to develop a generic car model for the MARS system that can be used in the Smart Open Hamburg project to simulate road traffic.

1.2 Hypotheses

The Intelligent Driver Model was developed by Martin Treiber et al. in 2000 [23] and has been used extensively in microscopic traffic simulations. Being a car-following model the main influence is the car driving in front, the velocity, safe headway and individual parameters for acceleration and deceleration. It has its limitations though. Besides the movement in longitudinal direction it doesn't incorporate any rules for lane-changing or traffic code. This leads tho hypothesis one:

H1 The Intelligent Driver Model can be used as foundation to build a rule-based agent model that follows the traffic code

The first hypothesis will test the model in specific scenarios to make sure that the agents behave correctly. For the second hypothesis, the model will be applied to a real setting, namely Altona, where the car-agents will drive on Hamburgs streets. Based on census data, a virtual population is created whose citizens move around the city running errands, working or enjoying their free time. The trips, done by car, will be simulated and the resulting traffic compared to traffic counts from the city of Hamburg. This will be a case study to test whether the combination of data coming from a developed day-plan generator (created by Andreas Löffler) and the implemented traffic model are sufficient to recreate the traffic volume measured through traffic counts.

H2 Using the car-model in conjunction with census data can reproduce traffic volumes as measured in traffic counts

Building a model for a specific area means incorporating certain aspects that only apply to said region. When a generic model is to be developed, it must be easily applicable to different settings. Every country has its own peculiarities like left-hand driving and a custom traffic code that must be taken into account. Therefore the generic car model must be tested in different scenarios with varying conditions.

H3 The model was implemented in a generic way so that it is applicable to simulate the traffic of different countries

To test the third hypothesis, the traffic model will be adapted to the conditions in South Africa. Simulations of the Kruger National Park will show whether the model can easily be adapted to the local traffic code.

1.3 Structure Outline

Chapter 2 (Methodology) extends the introduction in terms of the involved topics and the related work. Before the agent model can be designed, an analysis of the required contents will be performed in chapter 3 (Analysis). Chapter 4 (Experiments) will describe how the model behavior is going to be examined. With the analysis completed and the experiments formulated, the model will be designed and its implementation characterized in chapter 5 (Design and Implementation). Chapter 6 (Results) will present the outcomes of the experiments and discussion will take place in chapter 7 (Discussion). Finally an outlook on potential further development is given in chapter 8 (Future Work).

2 Methodology

This chapter is an in depth introduction to the field of traffic simulations. At first, the history of this discipline will be highlighted to better understand the aspects and especially the level of details. Not all traffic simulation models are equal and therefore differ in their applicability to certain scenarios. This distinction will be part two of this chapter.

2.1 History of traffic research

Understanding traffic has been part of research for many decades now. Starting with studies of traffic capacities in the 1930s [7], this field kept growing as the amount traffic increased. Induction loops hadn't been invented so these first investigations were carried out with stop watches and manual counts. After initially studying the behavior of traffic as a total system [13], the focus began to turn towards the small details of why things like traffic jams occur. These can often be found on roads where the mere capacity cannot account for it as there is more than enough space available. Still, the affects of individual behavior lead to jams that can grow into large obstructions [17]. To further examine the understanding, the first simulation models were created. Over the course of the past decades numerous approaches were developed that could be divided into macroscopic, mesoscopic and microscopic models [19]. Based on these models, multiple frameworks have been build that simulate the traffic in different level of detail with MATSim [5, 9], SUMO [15], PTV Vissim [5] and TRANSIM [20] being the most renowned ones.

2.2 Models used in traffic research

The distinction of macroscopic, mesoscopic and microscopic models is done based on the level of detail the traffic is modeled at [22]. Macroscopic simulations describe traffic as gas kinetic model where streets have certain capacities and the cars ”flow” through them [19]. Actions of individuals like, for example lane changes, are not taken into account. Microscopic models on the other hand incorporate individual traffic participants like cars, buses and pedestrians. The traffic in itself is thereby comprised of a multitude of entities and their actions in contrast to a mathematical description of the overall system. [21]. The third option to describe traffic are so called mesoscopic models. They combine the aforementioned techniques to gain a very detailed view on certain parts of the simulated area through microscopic models while keeping the computational requirements in check by simulating the rest on a macroscopic scale [24].

2.3 Microscopic models

Microscopic simulations can further be distinguished into agent-based models and cellular automata [23]. Both model types simulate each entity contained in the simulation. This leads to very detailed simulation results since the cars don't just drive on their own but also interact with each other as they would in *real traffic*. In the past the cellular automata models were used to a great extend since they require less computational resources. In contrast to multi-agent simulations, they discretize both space and time while agent-based models only discretize time. Recent advances in computational power now allow for bigger scenarios; even with more demanding agent-based models [22, 6, 26].

2.3.1 Cellular automata models

The most prominent example of cellular automata traffic simulations is the so called *Nagel-Schreckenberg* model which is both time and space discrete [16]. This approach which originally has been designed as sinlge-lane model, divides the simulated lane into segments of usually 7.5m length. Each segment can either contain no car or exactly one car. Speed of the cars is given in integers, denoting the amount of cells crossed in each simulation time step. A safe headway gap is kept so that the cars don't collide. Random braking noise is part of the model too so that the simulated cars don't behave perfectly, leading to jams that otherwise wouldn't occur. Even though this is a rather strong simplification the model includes the major aspects of traffic and has been used extensively in the past [5].

2.3.2 Agent-based models

Agent-based models can be thought of as extended cellular automata where each individual agent interacts on a broader scale or farer distance. While CA usually interact with their immediate neighbors, agents explore and interact over bigger distances. When agent-based models are applied to traffic simulations, usually so called car-following models are being used. These models calculate motion equations for each individual car in the simulation. They are influenced by different factors like speed differences, velocity, distance to the car ahead or desired deceleration values. Over the years many models have been developed, all taking into account different stimuli [19, 25, 5].

The first car-following models developed were applied to very specific situations. They were aimed at replicating highway traffic and even there they could only be applied to certain conditions. For free flowing traffic the models weren't able to calculate realistic acceleration if no car was ahead that one could follow. In dense traffic where the speed differences are very small, they didn't advocate to brake or to increase/ decrease the gap. This led to cars keeping a fixed distance even though their speed increased. The models simply couldn't detect it since the speed difference itself was constant which was the most influential factor [23].

2.4 Intelligent Driver Model

The Intelligent Drive Model has been chosen from the class of car-following models and will be the foundation for the rule-based agent model that is being build. It was published by Martin Treiber, Ansgar Hennecke and Dirk Helbing from the University of Stuttgart, Germany in 2000. In their initial paper the state that *"it is simple, has only a few intuitive parameters with realistic values, reproduces a realistic collective dynamics, and also leads to a plausible "microscopicäcceleration and deceleration behaviour of single drivers"* [23]. The IDM was designed as collision-free, single-lane model. Avoiding collisions was made possible by including the relative velocity to the car-ahead which exerts a growing impact with decreasing car-to-car gap. In comparison to other car-following models, the IDM has only a few parameters which make it easy to use.

The main factors of the *Intelligent Driver Model* are the cars own speed (v_n), the gap between cars (s_n) and the speed difference (Δv) to the car ahead [19]. The motion equation has been divided into three parts and is shown in equations 2.2, 2.1 and 2.3.

The change in velocity ($\dot{v}_n$) is calculated in every simulation step by solving the three equations.

$$s_n = x_{n+1} - x_n - l_{n+1} \tag{2.1}$$

$$s^*(v_n, \Delta v_n) = s'_n + s''_n \sqrt{\frac{v_n}{v_0}} + T v_n + \frac{v_n \Delta v_n}{2\sqrt{ab}} \tag{2.2}$$

$$\dot{v}_n = a \left[1 - \left(\frac{v_n}{v_0} \right)^{\delta} - \left(\frac{s^*}{s_n} \right)^2 \right] \tag{2.3}$$

The first equation (2.1) calculates the distance to the car ahead s_n by subtracting the cars current position x_n from the position of the car ahead (x_{n+1}), minus the length of the car in front (l_{n+1}). In the second equation, the desired gap s^* is calculated based on the desired distance in congestion (s'), distance in convoy (s''), velocity difference, safe time headway (T), maximum acceleration a and the comfortable deceleration b. Inserting the results into the third equation finally allows to calculate the optimal change in velocity while also incorporating the acceleration exponent (δ). Table 2.1 shows the default values for the IDM as they have been used in this thesis.

parameter	default value
desired velocity v_0	13.89 m/s
safe time headway T	1.6 s
maximum acceleration a	0.73 $\frac{m}{s^2}$
comfortable deceleration b	1.67 $\frac{m}{s^2}$
acceleration exponent δ	4
distance in congestion s'	2 m
distance in convoy s''	0 m
car length l	5 m

Tabelle 2.1: Default values for the *Intelligent Driver Model* [19]

3 Analysis

This chapter covers the analysis and collects requirements for the agents and the overall traffic simulation. First the requirements for the simulated environment will be analyzed, the second part is about the agents that make up the traffic. To resemble real driving, the car-model will have to take a multitude of aspects into account. These facets will be collected and shall serve as basis for the implementation. Third the required data for running the model will be specified.

3.1 Environment requirements

Before an agent model can be build that resembles cars driving, the environment has to be set up. The requirements for doing so will be collected in this section.

3.1.1 Road network

Integral part of every environment is the road system otherwise the cars wouldn't have something to drive on. The information needed to make the road network usable by an agent can be divided into several groups. Basics are the streets in themselves where for each road the start and endpoints have to be known with their respective geo location. This would be the equivalent of connecting intersections by straight lines wherever there is a road. Letting the agents drive on such a rudimentary road system would lead to cars driving through houses or other obstacles when printing the agent positions on a map. To solve this issue, the curvature of the road has to be taken into account too. Combining all of this information leads to a basic road network that, on its own, can be used to build a traffic simulation.

This would be the minimal set to represent the road network considering its physical layout. What that network is missing, is detailed information about things like speed limits, width, traffic signs and turning lanes. Without these details, the simulation will only be able to reproduce traffic to a certain degree. If these limitations should be overcome, the required information about conditions found in cities has to be included as well. When extending the road information, the lanes are one of the most important aspects since they determine the streets capacity. Knowing how many lanes exist per road is of great importance. If a major road in the simulation would have only one lane while it has three lanes in reality, this surely would change the simulation results.

3.1.2 Intersections

Depicting intersections is by no means an easy task. There is a wide variety of shapes they come in, each bearing their peculiarities. When combining intersections with traffic signs or lights, the amount of types grows extensively. This section will remove the aspect of lights and signs and will only discuss the intersections itself to clarify the challenges involved. In the next sections, these left out topics will be discussed in detail.

To start off the intersection topic, one has to look at the way streets and crossings are modeled in the MARS system. The *Spatial Graph Environment* is a custom graph implementation which is used to represent the street data downloaded through OpenStreetMap while managing the agents on top of it. Basic graph specifics apply in the way that the graph is composed of edges (streets) and nodes (intersections). At this point an approaching car can perceive the outgoing roads and continue its journey. The intersection itself has no spatial extend and each node in the simulation specifies the centre of a crossing. Each outgoing road has a starting point (the intersection) and an end point where it usually meets the next intersection. If one calculates the bearing and compares that with the current driving direction one is able to make out what cardinal direction these streets lead. This allows to distinguish the various directions for standard situations with up to eight streets per intersection. Judging the direction from an incoming perspective is a basic requirement to plan and execute the agents driving maneuvers. The cardinal direction aside, the agents miss another fundamental piece of information: the connections between the current road and the adjacent ones on the intersection. When the agent approaches the crossing, it can only 'see' that there is number of n streets it can continue driving on. What it doesn't know, is whether the current lane leads to the desired road.

3.1.3 Traffic lights

Traffic lights are tailored to the very crossing they are standing on. Factors like the amount of incoming and outgoing roads, the shape/ angles at which the crossing is set up, signals for various traffic participants, the amount of passing traffic etc. varies from intersection to intersection. This leads to the first requirement: The implemented traffic lights have to represent the circumstances of the intersection and take incoming and outgoing roads into consideration. One after another the different streets have to be signaled so that cars can cross the intersection.

For the first scenario set in Hamburg, the traffic light details of Germany have to be included. These traffic lights can generally be operated in two modes: The first one are fixed schedules where every signal is timed following a rigid scheme. For the programs duration the involved lights have their individual schedule that prescribes when they switch from green, to yellow, to red and back. Secondly the lights can operate dynamically thereby reacting to influences like the amount of incoming traffic. They still operate in bounds but they have room to maneuver to adapt their behavior to the current circumstances. Both modes of operation have a schedule that sets how long different light phases last and how they are coordinated. So for every phase for every individual light signal there is a clearly defined program that determines when and how long they show which signal. Inside these programs the process repeats over and over again until a different program is being executed. Typically there is different programs for rush-hour traffic, regular traffic, weekend traffic and one for the lights at night. This requires that the traffic lights adapt their behavior over the course of the day when different programs are run. Rush hour traffic in the mornings and afternoon has to be treated different from the rest of the day. For weekends, the traffic lights will also have to act different than during nights.

3.1.4 Traffic signs

Traffic signs play a major role in guiding the traffic. Besides traffic lights, they are the main way of telling drivers how fast they are allowed to go, who has right of way or where they are and aren't allowed to drive. Therefore the traffic signs have to be incorporated into the model so that it contains as much data on the traffic code as possible. In an ideal scenario, every traffic sign that has been placed in the city should be part of the simulation. Additionally the traffic signs must be inserted into the simulation in a way that the agents can sense them and adjust their behavior. For them, all the signs should be visible when they are driving around without any group of signs being prioritized. After sensing the information, the agents must determine what is important or not on their own.

3.2 Agent requirements

Microscopic simulation models are build by giving the agents a set of rules that define their behavior [21]. Formulating these rules is a complex process that includes multiple steps of refinement until the agents behave in the desired way. This section will collect the required high-level goals of agent behavior so that the reader can understand the dynamics involved in the model.

One general requirement for all following agent behaviors is that they must be integrated in a modular and dependency-free fashion. The car-agents must be able to interpret unknown traffic situations and still be able to act accordingly. The agents will have to sense their surroundings to get a clear picture of the current situation. This exploration muss reach far enough to include all information that might be relevant to the decision making process of the agent. At the same time it must be just big enough to catch all relevant information while staying small enough to keep the performance at a reasonable level All information has to be sensed as part of the agents tick/ reason method. In a second step the agent processes the input and decides (on its own) what to make of it until finally an action is decided on and performed. The agents actions might consist of multiple steps at times but for every part of it, the decision on doing it must come from the agent.

3.2.1 Longitudinal movement

Most basic part of driving is the movement in longitudinal direction. Accelerating to gain momentum, driving straight and finally breaking are the fundamentals of this kind of movement. A most basic version of a microscopic traffic model would be a car following a route without minding other traffic participants or any traffic rules. The functional requirement arising is the agents capability to perform this task so that a real drivers behavior is resembled. This includes accelerating in a realistic way as human drivers would do as well as decelerating appropriately. Carrying out these actions requires information about the current circumstances/ the environment.

3.2.2 Lateral movement

In addition to moving straight, the agents have to perform movements in lateral direction. This is composed of two aspects: following the course of roads by cornering and changing lanes if appropriate. The first one, means following a road by cornering when the course of the road dictates. If the road has a curvature, the agent must follow its path so that the agent actually drives the whole distance of the route. In respect of nodes and edges of a road network this forbids going straight from intersection to intersection. Obligation number two are lane changes and bypassing maneuvers. These are necessary to include so that cars don't make turns from the leftmost lane to right or vice versa.

Lane changing

The agents will have to plan their movement on the streets so that they choose lanes based on their route. Without this planning they won't be able to move to the right lane prior to turns for example. It is therefore necessary to incorporate lane-planning that takes care of the agents lane changing behavior. This involves everything from complying with the german 'Rechtsfahrgebot' to timely pulling in the correct lane for turning or drive-off maneuvers.

Turning maneuvers

When agents reach an intersection they can go straight or perform a turning maneuver that either points them to the right, left or back. While the agent can go straight without reducing its speed, this isn't realistic for changing directions. Depending on the turning angle, the agents have to adapt their velocity to a safe speed prior to executing the movement. Braking must be executed in a controlled way so that the maneuver as a whole resembles realistic driving. Once the agent approaches an intersection it must reduce its speed in time so that it can either come to a stop, should the need arise, or to make its turn. After turning the agent can proceed with its regular behavior.

3.2.3 Traffic code

With the aforementioned requirements in place, the driving agent will be equipped with the basic movement schemes to follow a set of rules. These rules consist mainly of the traffic code which has to be respected by the agents. These traffic rules have to be incorporated in the model and therefore real-world situations have to be translated into code. Depending on the location (meaning city or country) which the model resembles, these might differ heavily. This can be rudimentary differences like right-hand versus left-hand driving and spans to most diverse right-of-way scenarios found in countries around the world. It is important that the traffic code can be switched out so that the model is applicable to different scenarios.

Traffic signs

When driving around the simulated area the agents have to react to traffic signs. These might give them information about the right of way, warn about dangers, limit the speed or tell them where they are allowed to drive. All this information has to be considered when deciding on the next action to perform.

Traffic signs that belong to the right of way category are of great importance as they dictate the behavior at intersections. Agents have to follow their guiding rigidly as negligence would lead to accidents in the simulation. These regulations differ in the simulated areas so there must be dedicated logic for dealing with it based on the simulated country.

Roads have speed limits that determine how fast traffic participants are allowed to go. This varies from country to country but the overall concept is the same. For the simulation model this means that speed limits have to be respected by the agents. They have to *sense* the current limits and adapt their speed accordingly. If new speed limits apply, the agents have to change their driving parameters and act according to the new maximum velocity. For scenarios where the new limits are higher, the agents can accelerate until they reach the now higher speed limit. If the limit gets reduced, the agents have to slow down.

Intersection behavior

When approaching intersections the agents first have to determine the type of intersection they are facing. In general this is one of two situations: Either there is a traffic light, or there isn't. This subsection starts with the former and in the seconds half deals with the latter.

When an intersection has traffic lights, the agents have to follow their signaling and ignore other traffic signs. Not all traffic signs, but those that deal with the right of way as they only apply if the traffic lights are switched off. The basics in dealing with traffic lights is that cars stop at red lights and drives when signaled a green light. If the light turns yellow, the agents have to decide whether there is enough space to come to a halt in time or if they are too close already. If they can stop, they must do so, otherwise they continue at their current velocity and cross the intersection.

The second type of intersections (those without traffic lights) requires a more complex treatment since there are many factors that have to be considered. In situations where no traffic signs are present, the car must execute a default behavior, based on its current location. In Germany, that would mean left yields to right, while other countries have different regulations. For example in South Africa and the US it is customary to come to a complete stop and then proceed in the order of arrival (4 way stop). If traffic signs regulating the right of way are present, the car must follow their ruling. As before these signs are subject to local legislation and vary from country to country so that there is no one-fits-all solution to it. Since this model is meant as a generic model, the behavior must be adaptable depending on the simulated environment.

3.2.4 Further regulations

Road behavior beyond these requirements depends on the simulated country. The car agent must be usable in a variety of scenarios that include changing local restrictions as well as modes of operation. For this thesis there will be two countries of interest since the two proof-of-concept scenarios are located there. Germany will be looked at first, with the second scenario being situated in South Africa. To incorporate such divers traffic codes, the simulation model will have to be build in a way that allows for swopping of components. Depending on the research question, the model must be easily adaptable so that the respective question can be answered. In addition to the requirement of being interchangeable, these traffic-rule components must be build in a modular way so that parts can be reused. For example four-way stop crossings can be found both in the US and in South Africa. If one were to create a new scenario in North America, then it must be possible to use the already existing module for four-way stops from South Africa instead of programming it a second time.

3.2.5 Routing

The last requirement in terms of driving is routing. This defines which roads the agents follow to reach their destination. Before the agents can head off towards their destination, they will have to plan how to get there. During the drive the agents have to manage that route in a way that they always know their current position on it. Depending on the encountered traffic conditions the agents might re-plan and change their route as well.

3.2.6 Non-functional agent requirements

The model developed in this thesis will establish a baseline and will be extended over the years. It is therefore critical to choose an extensible architecture capable of evolving with future requirements. Every aspect of the model whether it is calculating the current speed or deciding on right-of-way has to be easily manageable, reusable and extensible.

3.3 Model requirements

The agents in themselves must be able to follow a set of rules, i.e. the traffic code, in order to be used in a traffic simulation. The closer this behavior resembles that of a *real* driver, the more accurate the results and therefore more significant. These agent-requirements have been formulated in the last sections since they lay the foundation before the bigger picture can be taken into consideration. Besides technical aspects like routing, the agents actually have to go somewhere. If the goal is to recreate the traffic of Hamburg, the car agents must follow the movement patterns of actual human beings. Even though cars don't drive around on their own, at least not officially here in Germany, the model must do exactly that.

3.3.1 Virtual population

Running simulations of Altona not only requires data about the area, but also about its population. Each citizen and/ or commuter makes trips throughout Altona which make up its total traffic. If the simulated traffic should resemble the real system, it must depict these trips made through its traffic participants. For simulations on a microscopic level this means that each individual trip of each individual traffic participant has to be modeled. This includes all the trips made, like going to work, driving home, running errands, going to the gym and many more. As this thesis specifically looks at car traffic, only the trips made by car have to be recreated.

3.3.2 Creating agents during simulation

When running a traffic simulation there will be an initial set of agents that gets created and inserted into the simulation at startup. This typically involves the desired amount of cars that should be driving around at that points in time. What each car does, is up to the specified logic but they will have an origination and a destination once the simulation starts. Over the course of the simulation the agents will eventually reach that destination which leads to the question what to do then? There are many ways of handling that but for this thesis the desired behavior is as followed: Car agents are on the road because it was derived from the virtual population that a real human being would, statistically speaking, take its car to go somewhere. Once this task has been completed, the car doesn't matter anymore and must be removed from the simulation. At some point in the simulation all agents would have finished their trip and the simulation would be empty since there were no agents added during runtime. In the real world, there are always people who get into their car and start driving somewhere. The virtual population must do so too. Throughout the day this substitute must produce trips that resemble the real dynamics seen in everyday traffic. Streets are more crowded in the morning hours and in the afternoon (rush hour). Mid-day and in the evening the traffic density decreases until it is at its lowest during the night hours. Based on these dynamics, new agents must be created and added to the simulation over the course of the day so that the simulation never runs out of agents.

4 Experiments

To test the formulated hypothesis, a range of experiments is devised. This chapter collects these experiments and describes each of the simulated scenarios as well as the expected outcomes. The results will be shown in chapter 6 (Results) and discussion will take place in chapter 7 (Discussion).

The experiments are carried out in two geographic locations: Setting one is Hamburg, Germany where the models capabilities to comply with the german traffic code and the models validity will be tested. Hamburgs location is shown in figure 4.1 on the left. The second location, the Kruger National Park (shown on the right of figure 4.1), will be used to examine the third hypothesis. Both sites are described further below.

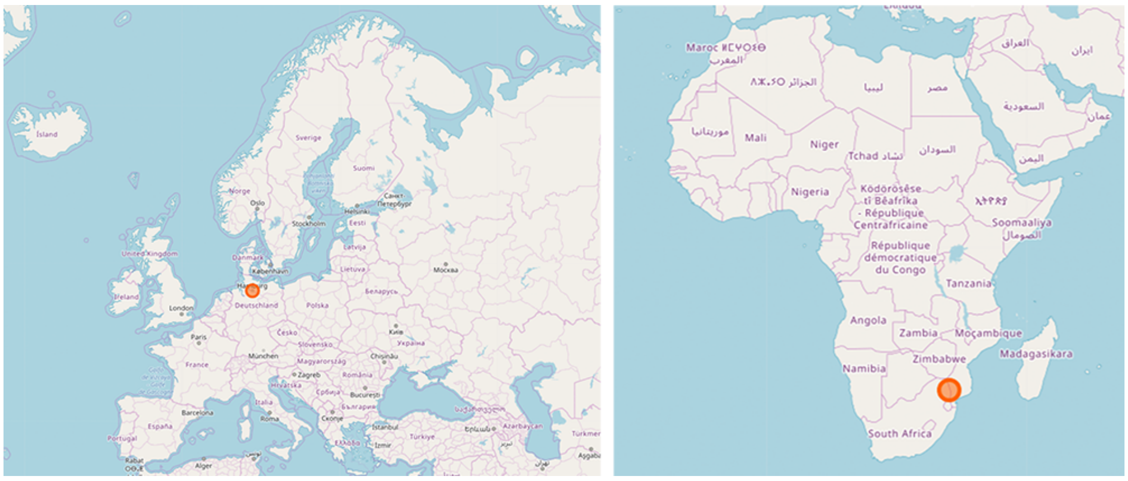


Abbildung 4.1: Locations of Hamburg (left) and the Kruger National Park (right) on their respective continent (source: OpenStreetMap)

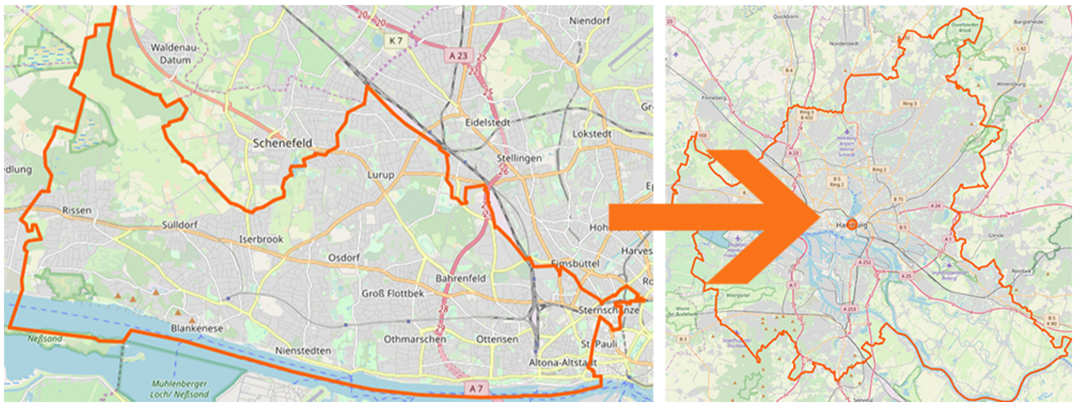


Abbildung 4.2: Location of Altona in Hamburg (source: OpenStreetMap)

4.1 Setting One: Hamburg Altona

Scenario number one is located in Altona, a borough of Hamburg (Germany). This district which is located in the west of the city, spans an area of 77 square kilometers and has a population of 270263 people [1]. Out of all Hamburgs neighborhoods, this one was chosen because of its current changes. There is a significant amount of construction zones where the cities traffic system is reshaped to fit its modern demands. Additionally new boroughs are created and new public transport stations are build from scratch to extend the network. There is a lot going on so that the SmartOpenHamburg project decided to make it one of its study sites. Population-wise there are 270.263 citizens living in Altona (effective 31.12.2016) [1].

4.2 Setting Two: Kruger National Park

The second scenario was chosen to test the model in a completely different surrounding. Instead of Hamburgs busy streets this setting comprises the tar and gravel roads of one of Africas most famous national parks. This setting was chosen to test the models adaptability to different traffic rules. The parks roads come by without traffic lights and rely mostly on four way stops. On of the biggest differences between Hamburg and South Africa is that road-users drive on the left-hand side.

As part of the EMSAfrica project this traffic model is being developed to provide decision support in the Kruger National Park. Managing visitors becomes a very important topic

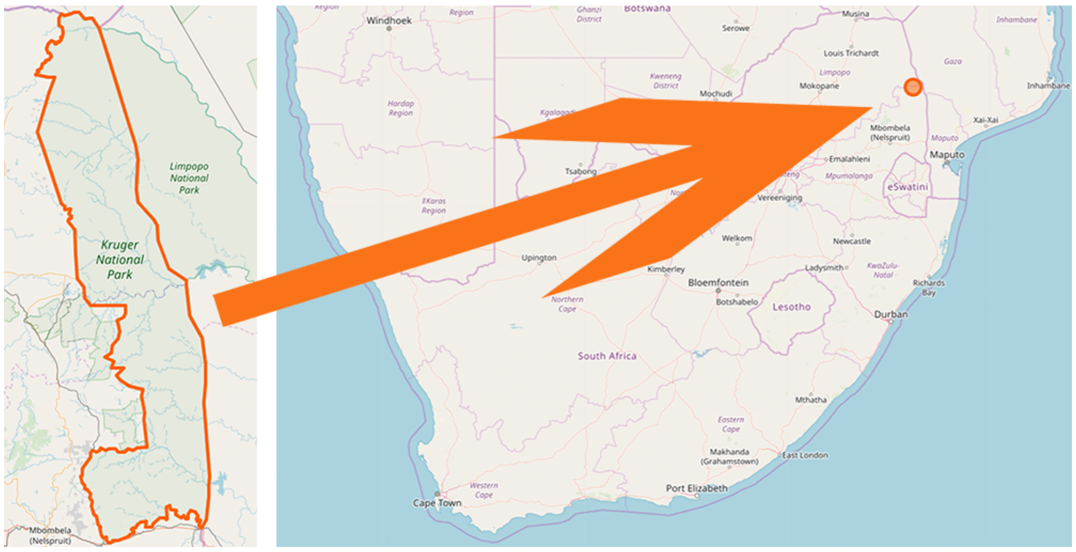


Abbildung 4.3: Location of the Kruger National Park in South Africa (source: OpenStreetMap)

for the park since their numbers are increasing steadily [18]. Especially during peak times, traffic amounts inside the park grow and will turn into a problem if the main roads get congested. Developing an agent-based simulation model to investigate road traffic in the park will help dealing with these issues. With this thesis, a foundation should be established where the model developed for Hamburg is being adapted to the parks rules.

4.3 Experiments for Hypotheses One

The first hypotheses states that *The Intelligent Driver Model can be used as foundation to build a rule-based agent model that follows the traffic code*. Examination will take place through simulating situations, typically found in urban street traffic. These scenarios will test the agents capabilities of correctly recognizing the current circumstances. In a second step, their reactions to these situations will be examined. Since the implementation of the car model extends the *Intelligent Driver Model*, it is also tested if the original model still produces the correct results in situations where the added rules don't apply or interfere.

4.3.1 Basic driving

The first group of experiments will test the basic driving capabilities to make sure that there is a solid foundation to build the rule-based agent model. Setting for the following experiments is the *Veddeler Damm*, a perfectly straight road in *Kleiner Grasbrook*, a borough in the south of Hamburg. This street has been extracted from OpenStreetMap and the resulting graph has been modified so that only required data is contained. Some of the following experiments make changes to this setup which will be stated there.

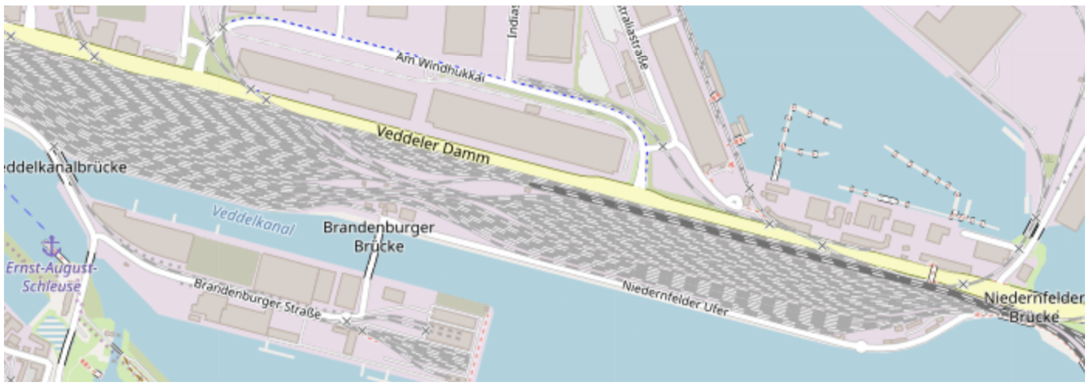


Abbildung 4.4: Veddeler Damm in the south of Hamburg (source: OpenStreetMap)

Acceleration and breaking

This agent model uses the *Intelligent Driver Model* as foundation. To make sure that it has been implemented correctly, the following experiments will examine acceleration and deceleration values calculated through the IDM implementation.

Experiment: "Accelerating"

This experiment will test the car-agents acceleration behavior from a standing position to the maximum speed of 13.89 m/s (50 km/h). The setting is the Veddeler Damm which has no curvature so that the cars behavior can be examined clear of any influence. No agent rules concerning the traffic code apply and there are no other cars, traffic lights, traffic signs. etc. involved in the experiment.

Expected outcome: With every simulation step the car will incrementally increase its velocity until the maximum speed is reached. During the acceleration the increase per step will start with the maximum acceleration value ($0.73\ \frac{m}{s^2}$) as defined through the IDM's parameters. From there, the acceleration will decrease slowly until the maximum velocity is reached in simulation step 38 at which the car stops accelerating. The acceleration seen will be the exact values as calculated through the IDM since none of the agents other rules apply here.

Experiment: "Regular deceleration"

Basic deceleration has to be tested in order to ensure the cars braking, using a realistic deceleration behavior. This experiment tests a situation where the agent has enough time and space available to come to a complete stop without the need to brake very hard. The car agents starts with an initial speed of 13.89 m/s (50 km/h) and an obstacle at 50 m distance.

Expected outcome: The car will instantly begin to reduce its velocity and keep the deceleration high with up to $-1.67\ \frac{m}{s^2}$ as defined through the IDM *comfortable deceleration* parameter. This continues until the speed has decreased enough so that the remaining distance can be used to reduce the remaining velocity by minor brake application. Finally the car comes to a complete stop 2m before the obstacle since this is the safety distance kept by the IDM.

Experiment: "Intense deceleration"

To test the cars deceleration behavior in a scenario where severe breaking is mandatory, a stop for an obstacle within 25m is being simulated. The car will use the same setting as in the acceleration experiment. The starting velocity will be 13.89 m/s (50 km/h) and the task is to stop in time so that there is no collision with the obstacle that is 25 m away. This distance was calculated following the default formula taught in driving school where the braking distance is calculated by dividing the velocity by ten multiplied with the velocity divided by ten ($\frac{velocity}{10} * \frac{velocity}{10}$). For a car driving at 13.89 m/s this solves to a breaking distance of 25 m while not executing an emergency braking maneuver. Since the IDM takes a 2m safety distance into account, the actual distance to the obstacle is set to 27m.

Expected outcome: The car starts with an initial speed of 13.89 m/s (50 km/h) and will immediately decelerate strongly. High deceleration values will be seen for the whole experiment that will exceed the *comfortable deceleration* parameter specified in the IDM. After braking strongly for a couple of seconds, the car will come to a stop within the 25 meter. There will be no collision since there is enough space to brake.

Experiment: "Driving on a road"

Once the car is able to accelerate and brake, the next step is to test its capability to drive on a road. The main objective is to see if the car follows the course of the road and doesn't cut corners or drives through buildings. As setting a road with a l-curve has been chosen (Vogt-Groth-Weg, Altona) where the car has to follow the streets geometry. This experiment was added since the cars initially drove straight from the edges origin to the destination without following the actual road.

Expected outcome: The car will start with increasing its velocity to the specified maximum speed at which it will keep its velocity throughout the simulated scenario. Right at the beginning, the agent will read the roads geometry to get a clear picture of the environment. After 150m the curve is reached whose geometry is available through the SGE. As the agents progresses, it keeps adapting its bearing and follows the actual geometry of the road. The experiment results will be judged visually.

4.3.2 Speed Limits

The speed limits of streets are set through the authorities and are shown to the drivers either through traffic signs or road markings. Each agent derives its own desired top speed from that and constantly tries to reach it as long as the conditions allow for it. In the default configuration each agent has a desired top speed of 50 km/h (13.89 m/s) as it is common on german roads. For the following speed limit tests this is changed.

Experiment: "Increasing speed limits"

The setting is again the *Veddeller-Damm* since it has no curves. A car agent drives on that road which has sections with different speed limits. At the intersection in the middle of the road the speed limit is increased. The intersection itself is being ignored so that the agents behavior isn't altered by other rules. The scenario is configures so that the speed limit increases from 30 km/h to 50 km/h. The agent starts from stand with its maximum velocity set to 30 km/h so that the increase can be monitored once it enters the zone with higher speed limits.

Expected outcome: The agent accelerates to the initial speed limit of 30 km/h. After reaching this limit it continues driving until the intersection is reached. Once the agent has passed, it should detect the new speed limit. Incorporating the now changed maximum speed into its movement calculation, the agent should accelerate until the new limit is reached. To determine whether the experiments were successful, the velocity of the car, extracted from the result data will be compared to the speed limits. It will be examined whether the car agent followed the instructions, came below the limits or exceeded them.

Experiment: "Decreasing speed limits"

As with the previous experiment the setting is a street with no curves (*Veddeler Damm*). This time the opposite is tested with speed limit decreasing from 50 km/h to 30 km/h. The car starts from standstill with its initial speed limit set to 50 km/h. After 1150 m the intersection will be crossed at which point the limit decreases.

Expected outcome: After driving at a speed of 50 km/h, the agent reaches the 30 km/h zone. The car calculates the desired deceleration and reduces its speed to comply with the new speed limit. The measured velocities and deceleration maneuvers will be extracted from the result data and compared to the ideal values derived from the Intelligent Driver Model. It will be examined whether the car agent sensed the changing speed limits and how it obeyed them.

4.3.3 Traffic lights

The third group of experiments will test the agents ability to deal with traffic lights. It will be examined if the three phases are recognized and how the agents react to them. The scenarios include the stopping in front of red lights and the proper behavior in green light situations. Yellow phases are treated separately to determine whether the agents make the right decisions about coming to a halt or continuing. For these scenarios

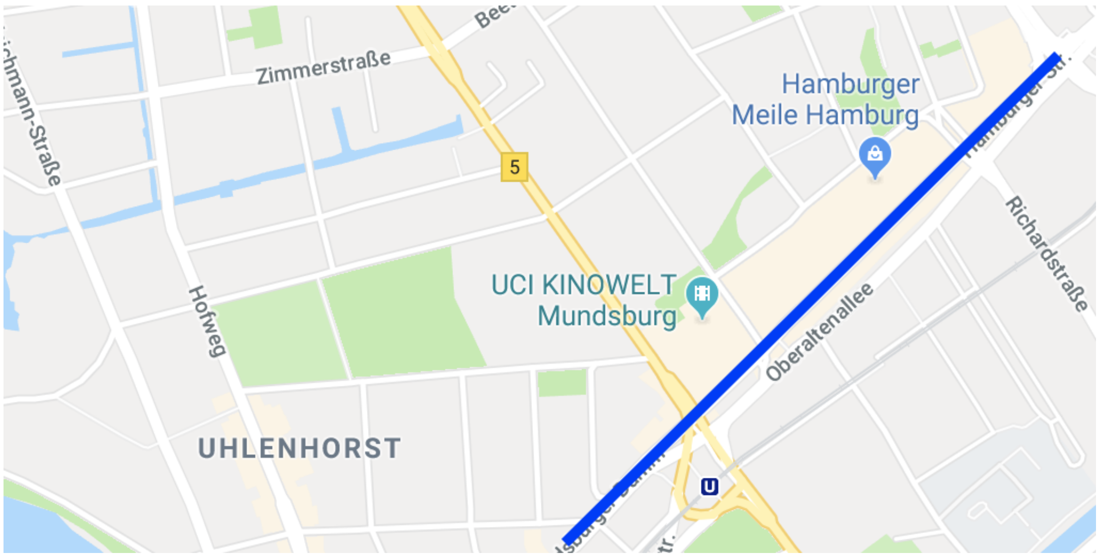


Abbildung 4.5: Setting for the traffic light experiments in Mundsburg (source Google maps)

a special graph was created that includes three crossings in Hamburg-Wandsbek. The agent starts out at the crossing of Hamburger Strasse and Wagnerstrasse before driving to the intersection on Winterhuder Weg and Hamburger Strasse (south-west direction).

Experiment: "Stopping at red lights"

The car-agent will drive on the *Hamburger Straße* which has a traffic light at the first intersection. The agent will approach the traffic light after 200 meters where it will encounter a red signal, forcing it to stop.

Expected outcome: The agent will accelerate according to its individual driving parameters until it approaches the traffic light. The field of view allows it to see the light in time to decide on a proper action to perform. It checks the state of the traffic light which is red and then starts decelerating until coming to a complete stop, two meters before the traffic light. The two meters derive from the IDM where a certain safety distance is kept from obstacles.

Experiment: ”Green lights”

As before with the experiment to test the red light behavior, the agent will start with an initial speed of 0 km/h and drive on *Hamburger Straße*. The difference to the aforementioned scenario is, that it will encounter a green traffic light which allows the agent to continue.

Expected outcome: The agent starts with accelerating to its maximum speed of 50 km/h. It approaches the intersection after 200 meters and perceives the traffic light. Checking its status reveals the green light so the agent proceeds and crosses the intersection without decelerating.

Experiment: ”Yellow light - cross intersection”

The setting stays the same as with the experiments before. When the agent approaches the intersection, the traffic light turns yellow. A german yellow phase lasts 3 seconds for intersections where 50 km/h is set as speed limit. In that time a car traveling at 13.89 m/s (50 km/h) will have driven 41.67 meters. In the simulated scenario, the agent will be less than 41.67 meters out so that there is enough time to cross the intersection.

Expected outcome: The agent drives towards the intersection and perceives the traffic light in its initial green phase. As the agent gets closer, the light turns yellow. Using its current velocity to calculate the position of the next ticks the agent decides to continue driving since the speed is high enough and the traffic light close enough. The intersection is crossed without stopping or changes in velocity.

Experiment: ”Yellow light - stop”

The second yellow light experiment will test whether the car comes to a stop in a scenario where the car is more more than 41.67 meters away from the traffic light. As it turns yellow, the car should check its current speed, calculate that there is enough time/ space to come to a halt and then start to decelerate. The setting is the same as before.

Expected outcome: Driving towards the intersection the agent perceives the traffic light in its initial green status. During the approach this changes to yellow. As the light turns, the agent calculates that he is too far out to cross the intersection without running a red light. After calculating the remaining distance to the traffic light, a controlled braking maneuver is performed and the agent comes to a stop.

4.3.4 Intersections

At the time of writing this thesis there was no detailed data available on the design of intersections. The only information that could be obtained through Hamburg's open data portal where the positions of traffic lights. For intersections it is therefore known whether there are lights or not, but it is not known how they are operated. If an intersection has no traffic lights, usually traffic signs display the right of way regulations. Unfortunately there was no information available about traffic signs so they aren't part of the model. The last detail about intersections are road markings which mark the direction one is allowed to drive. These couldn't be acquired either. As a result the experiments for intersections are limited as the cars don't have much information available. What is known, is the amount and direction of incoming and outgoing roads as well as their orientation. Detailed information about the agent rules at intersections are described in the next chapter (Car agent modeling).

For intersections without traffic signs or lights it is customary to give way to cars coming from the right. At least on german roads the *left yields to right* rule applies. The following experiments test this kind of intersection in varying scenarios. Setting is an intersection in Winterhude, one of Hamburgs boroughs in the north. Figure 4.6 shows the crossing with its four outgoing and incoming roads. In reality the roads have more than one lane per driving direction but this was changed. For each incoming and outgoing road there is exactly one lane available and the length is equally set to 100m.

Experiment: "Approaching alone"

The first simulation scenario depicts an empty intersection with only one car arriving. It will be examined if the car reduces its velocity prior to crossing the intersection. Since no other cars are present, the agent is allowed to proceed without coming to a stop.

Expected outcome: After initial accelerating the car approaches the intersection. The starting distance to the intersection is 100m so the car will not reach its maximum speed of 50 km/h. Instead it will sense the crossing and will start to reduce its velocity to an adequate level. During the approach the agent will explore all incoming roads to check

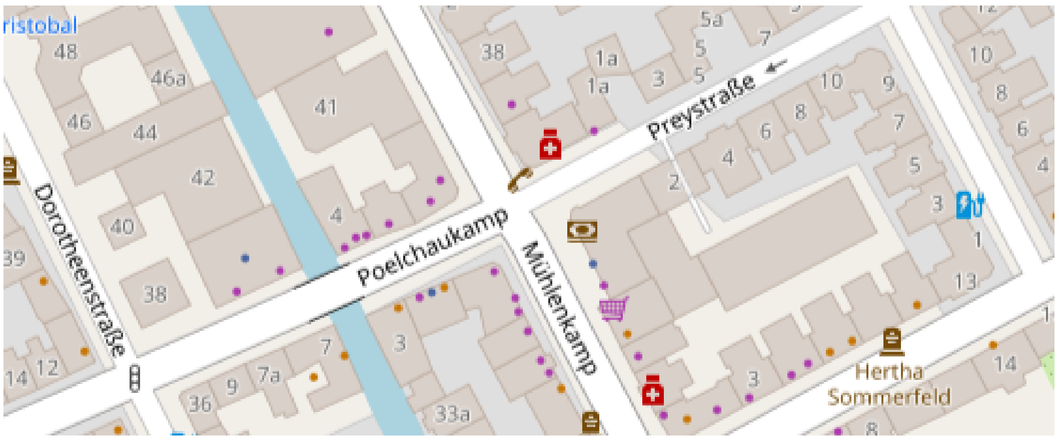


Abbildung 4.6: Four way intersection in Winterhude, Hamburg

if it has to give way. After determining that there is no car approaching, the agent will proceed to accelerate after crossing.

Experiment: "Traffic from the right"

In the second experiment for intersections there will be two cars approaching at the same time. The first agent comes from the south and wants to go north while the second one comes from the east and wants to go west. For the first agent the second one comes from the right so that one has right of way.

Expected outcome: After their initial acceleration phase, both cars sense the intersection and reduce their speed so they could come to a stop. The agent coming from the east checks the other incoming roads, detects agent number one and ignores it since it has the right of way. Agent number two has right of way so he proceeds to cross. At the same time the car coming from the south senses agent number two and realizes that he has to yield. As a result the first agent reduces its velocity further until the second car has crossed. Only then he proceeds on its way north.

Experiment: "Turning left with approaching traffic"

Besides the rule to give way to traffic coming from the right there is another regulation making left turns. These are allowed as long as there is no traffic coming from the front. If so, that incoming traffic has precedence and the car must stand down. To test if this behavior was implemented correctly, two agents will approach from opposing directions. Agent one comes from the south and wants to turn left (west) while the second agent is southbound.

Expected outcome: Both agents start their trip with accelerating until they approach the intersection. In order to be able to come to a complete stop, should the need arise, the cars reduce their speed. The car coming from the north registers the incoming agent but since he is headed south that doesn't bother it. For the other car wanting to go left, the incoming car crosses its path. The southbound car crosses the intersection while the other one lets him pass. Only then the car turns left and crosses the intersection.

Experiment: "Deadlock turning left with approaching traffic"

Yielding to incoming traffic from ahead can lead to a deadlock situation if both cars want to turn left. In that situation, each car must give way to the other one which leads to no one driving. Turning lights would solve this issue in reality but that is not part of the model. To test the agents behavior, this experiment puts two of them in such a situation.

Expected outcome: After reducing their speed while approaching the intersection each agent senses the other incoming cars. Both agents register the opposing car and since wanting to turn left, come to a complete stop to give way to the other agent. After waiting at the intersection with no one moving, one agent starts driving to resolve the deadlock.

Experiment: "Complex deadlock situation with four directions"

To test the worst case scenario, a gridlock will be simulated. In that situation cars approach from all four sides with one going right, one going left two cars going straight. Even though the car going right could resolve this issue by making its turn, it mustn't since all cars arrived at the same time. For such scenarios the german traffic code dictates the use of hand signals so that one car can proceed, thereby resolving the deadlock.

Expected outcome: After arriving at the intersection all cars come to a complete stop. To solve the issue, one car will cross the intersection. After this step the process continues until the intersection has been cleared. Only one car will cross the intersection during the simulation step.

4.3.5 Multilane roads

The last experiment for hypothesis one is dedicated to multilane traffic. It will be examined how agents deal with lane choices prior to turning maneuvers.

Experiment: "Turning lanes"

When cars make turns in reality they don't cross from the leftmost lane to the right or vice versa. The usual way to do this is to go right from the right lane and to go left from the left lane. Road markings indicate this but unfortunately there is no data available on them. In this experiment it will be tested if two car-agents make turns from the correct lane. A road with three lanes is simulated so the first agent should go right from the rightmost one and the second one left from the left lane.

Expected outcome: Once the cars are on the road they will determine the ideal lane based on their route. They calculate that based on the angle and then proceed to the desired lane. For the car going right, this will be the right lane and for the other car the left lane. In the end, the agents make their turns from the correct lane

4.4 Experiments for Hypotheses Two

With the model in place and examined, the next step is to broaden the view from an individual level to the bigger picture. If one agent can follow the rules, what happens if many of them come together? In this section the synergy will be examined to see how the agents interact. More importantly it will be examined whether the interaction of many individuals resembles the traffic patterns found in Hamburg.

4.4.1 Altona traffic counts

The city of Hamburg uses traffic counts in a variety of ways to gain insights into the situation on the roads. The first way of gathering are permanently installed counts on the major roads. These have data available for every hour of the day, both on weekdays and the weekend. Additionally there are yearly counts that get repeated as regular as possible. This doesn't necessarily work every year so there are gaps in the data. The

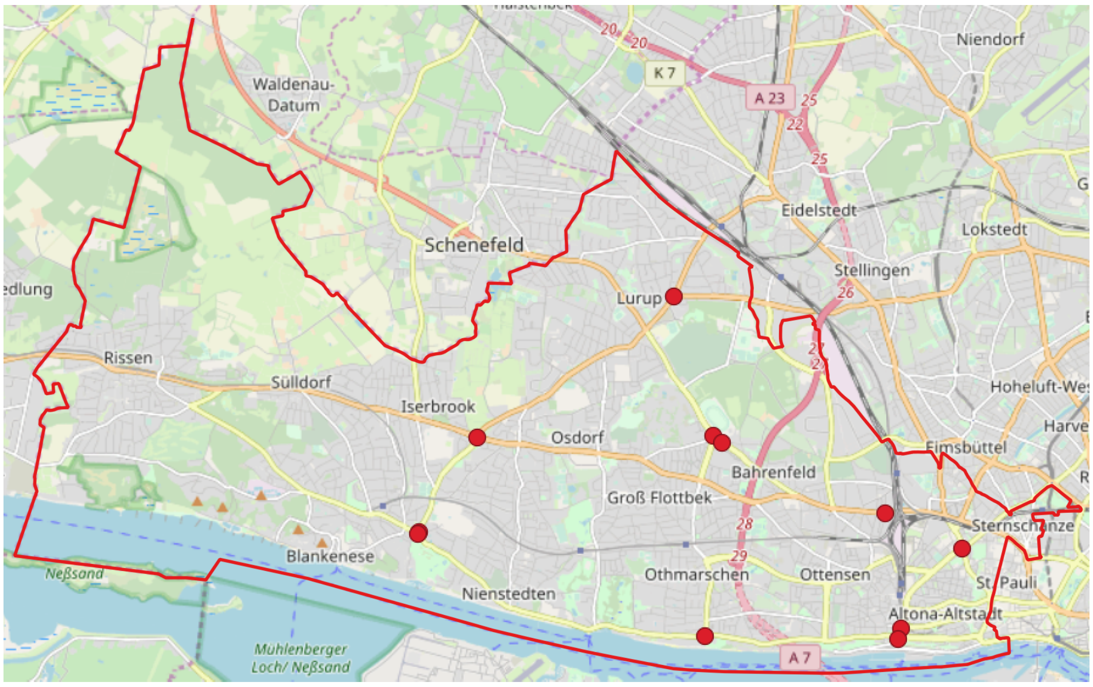


Abbildung 4.7: Locations of traffic counts in Altona

last kind of counts are being done on demand when certain roads are to be examined in detail. This only happens when there is a need for it.

For the following experiments the data of years 2016 - 2018 will be used as combining them covers the simulated area. The resolution is 15 minutes for all counts, no matter if permanently installed or on demand. It is to be noted that the counts from the same years weren't necessarily collected on the same day as this would exceed the staff. Besides the days/ dates of counting differing, there isn't always data for every day of the week. There might be data on road one from a count in March on a Monday while the next count for road two might be from a Wednesday in October. Permanently installed counts have data available for every day while the on-demand counts usually only counted on a specific day. All counts used for the experiments were from weekdays and if possible from Mondays. If these days weren't available, another weekday was chosen.

These traffic counts are used as reference for the simulation. Hamburgs *LSBG* provided these counts for various intersections throughout the simulated area. Based on the calculated trips for the virtual population, the agents will go about their days while making trips throughout Altona. During the simulation, the movements of agents is monitored by artificial traffic counters just as in the real system. After simulating the area multiple

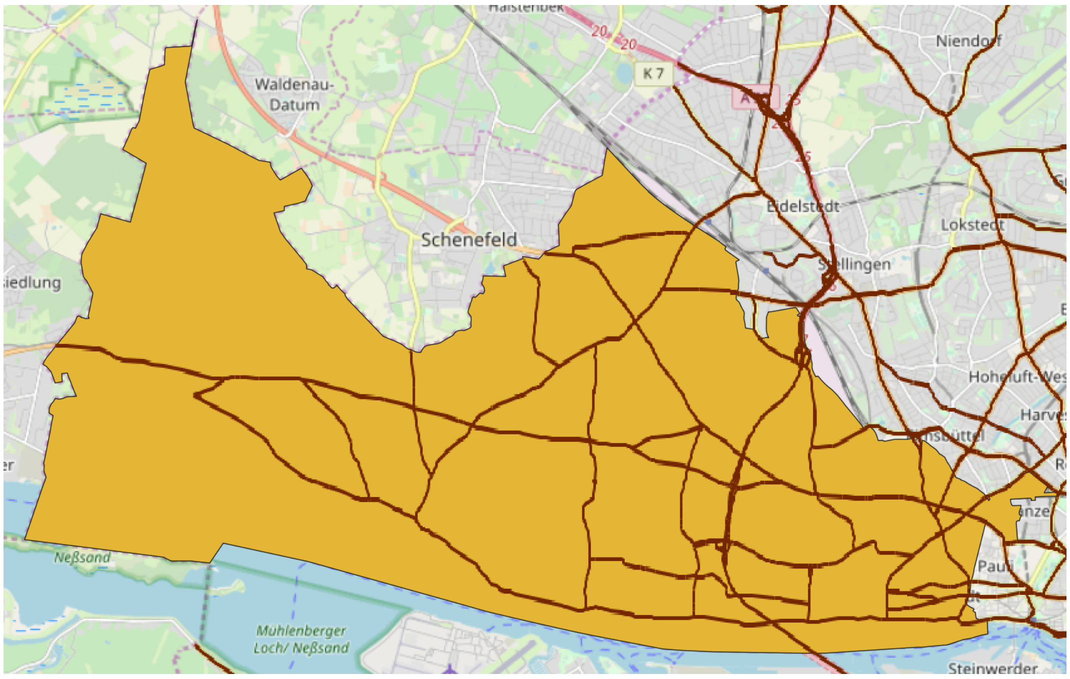


Abbildung 4.8: Main roads in Altona

times these counts will be compared to the actual traffic counts conducted in Altona. Figure 4.7 shows the locations of ten selected counting stations, extracted from the available set. There were more counts available but only the ones on major roads were chosen. Figure 4.8 shows these roads for the simulated area. Also the counting positions at the borders of Altona were neglected since these mainly count traffic entering or leaving the borough which isn't part of the simulation.

4.4.2 Simulating Altona's virtual population

How the virtual population of Altona is created is part of the next chapter that deals with the implementation. For now, it is sufficient to know that the virtual population resembles that of Altona, based on demographic data. In that population, citizens make trips throughout the city and if these trips are made by car, they will be simulated. The simulation covers the hours from 6am to 7pm which where chosen because there are traffic counts available for that period. During the simulation, the day-plan generator is used to derive trips with origins and destinations in Altona for 89705 citizens and execute these trips. This number was chosen based on the amount of cars registered in Altona

[1]. From a result perspective, the flow of traffic will be examined and compared to the actual traffic counts described above. This basic but yet very effective comparison will examine whether the measured simulated traffic corresponds to the amounts of traffic, measured in real counts.

4.5 Experiments for Hypotheses Three

For the last hypothesis, the model will be examined in a completely different scenario. Instead of Hamburgs streets, the Kruger National Park will be simulated. Four experiments are conducted to see if the model has been implemented in a generic way so that only a few changes have to be made. At first, the basic driving behavior will be tested since the new setting requires a couple of changes to the agent model. Once the basic experiments have been carried out, the Kruger National Park will be simulated with its roads, gates and camps.

Experiment: ”Left hand driving in South Africa”

Since the traffic code in South Africa dictates that cars drive on the left side of the road, the car agents will have to do so too. In terms of the road network this doesn’t alter the overall behavior of the car. At intersections the behavior will change though. For this experiment the cars will be configured through a config file to drive on the left side of the road. Additionally the *traffic code* parameter is set to ”south-africa” which tells the model to replace the german traffic code with the South African one. This experiment will show if the model is truly generic so that these changes can be made solely through configuration instead of by code.

Expected outcome: The car model has input parameters that decide about the used traffic code and the side of the road they drive on. Using a configuration file, these parameters are set during the simulation initialization. A simulation, based on said config file shows that the parameter has been set correctly and that the agents follows the traffic rules of South Africa.

Experiment: "Implementing a new intersection type"

Since the location changes from Germany to South Africa, the traffic code changes too. During the second experiment a new intersection type will be added to the code. It will be examined how much effort this takes and where these changes need to be made. This will determine if the car-agents have successfully been implemented as a generic model or if they only work on german roads.

Expected outcome: The model itself has been build in a modular way so that the new rules for the South African traffic code can be added with minimal changes to the original model. After implementing the desired behavior at intersections, this set of new instructions can be added to the model without changing anything but the part where the parameter for the traffic code is checked to determined which one to use. Once this is done, the model will ignore the german traffic code and work with the South African one.

Experiment: "Four way stop"

After examining the amount of changes needed to add a new traffic code, that of South Africa will be tested in this experiment. A four way stop will be simulated and the cars behavior will be examined similar to the experiments of hypothesis one. At this intersection, four cars approach from four directions so that they reach the intersection one after the other. The traffic code dictates that each car comes to a halt and then proceeds in the order of arrival.

Expected outcome: Each car approaches the intersection and reduces its speed before coming to a complete stop. During the approach each car monitors the order of arrival. After coming to a complete stop, the cars cross the intersection one after the other in the order they arrived there.

Experiment: "Simulating the whole park"

The last experiment will simulate the complete national park. Cars are spawned at either camps or gates and drive around looking for wildlife. Even though this is listed as experiment, it should be considered a showcase. It will be visually judged whether the cars drive around where they are supposed to.

5 Design and Implementation

This chapter will describe how the different parts of the model have been implemented. At first the environment with its road network, traffic lights etc. is covered while the second part deals with the car agent. Lastly the process of creating a virtual population will be described which is necessary to run the simulation.

5.1 Environment modeling

This part elaborates how the agent environment is set up and how things like the road network and traffic lights are made available to the agents during runtime.

5.1.1 Road network

To represent the road network, a custom graph implementation has been build. This was done as part of the *Grundprojekt* and has been described in detail there. This section will summarize this and will describe how the so called *Spatial Graph Environment* is being used in the simulation.

In general, the custom graph implementation has been split in two projects: the *Abstract Graph Environment* and the *Spatial Graph Environment* (SGE) targeting different use-cases. To this day the *Abstract Graph Environment* has only been used as basis for the spatial extensions but it was build with idea of having a non-specific/ generic graph implementation that could be used immediately. Simulating social network where agents are connected through graphs that represent relationships was one of the envisioned scenarios. Figure 5.1 shows the *Abstract Graph Environment's* UML class diagram. The basics to comprise a directed graph are nodes and edges as represented through the abstract classes *AbstractEdge* and *AbstractNode*. Both have their respective interfaces to specify the attributes and methods. In order to identify each entity in the graph, both

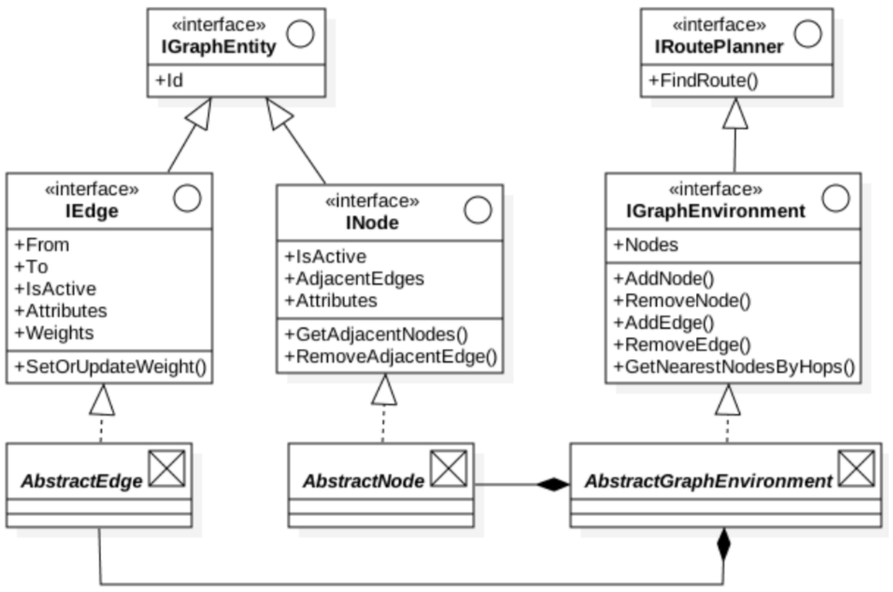


Abbildung 5.1: Abstract graph environment

classes additionally implement the *IGraphEntity* interface which gives them a GUID each. Edges cannot exist without nodes as they can be seen as mere connections of these. So the most important parameters of an edge are its *From* and *To* nodes. To denote the current status of edges and nodes an *IsActive* flag has been added which could later be used to activate/ deactivate certain connections (edges) or nodes. For each entity a dictionary of attributes (*Attributes*) is available too, which can be used to represent a variety of information about the edge/ road. For route finding purposes the edges also implement a weight attribute and the *SetOrUpdateWeight* function to modify them. The *INode* interface includes a list of all adjacent edges for the node and methods to access and modify these. This has been build with exploration and graph modification in mind so that agents can sense their environment and to allow the graph to be changed at runtime. Managing the nodes and edges is the purpose of the *Abstract Graph Environment*. Through the *IGraphEnvironment* interface it is equipped with the methods to build, alter and delete a graph which includes adding and removing nodes. Since edges are connections of nodes, these are being removed if either the origin or destination is deleted. The last component of the environment is the *IRoutePlanner* interface which is implemented by the *AbstractGraphEnvironment* as well. This allows for route finding based on various algorithms like for example breadth first search or Dijkstra. Since they were implemented by Daniel Glake, they will not be discussed further.

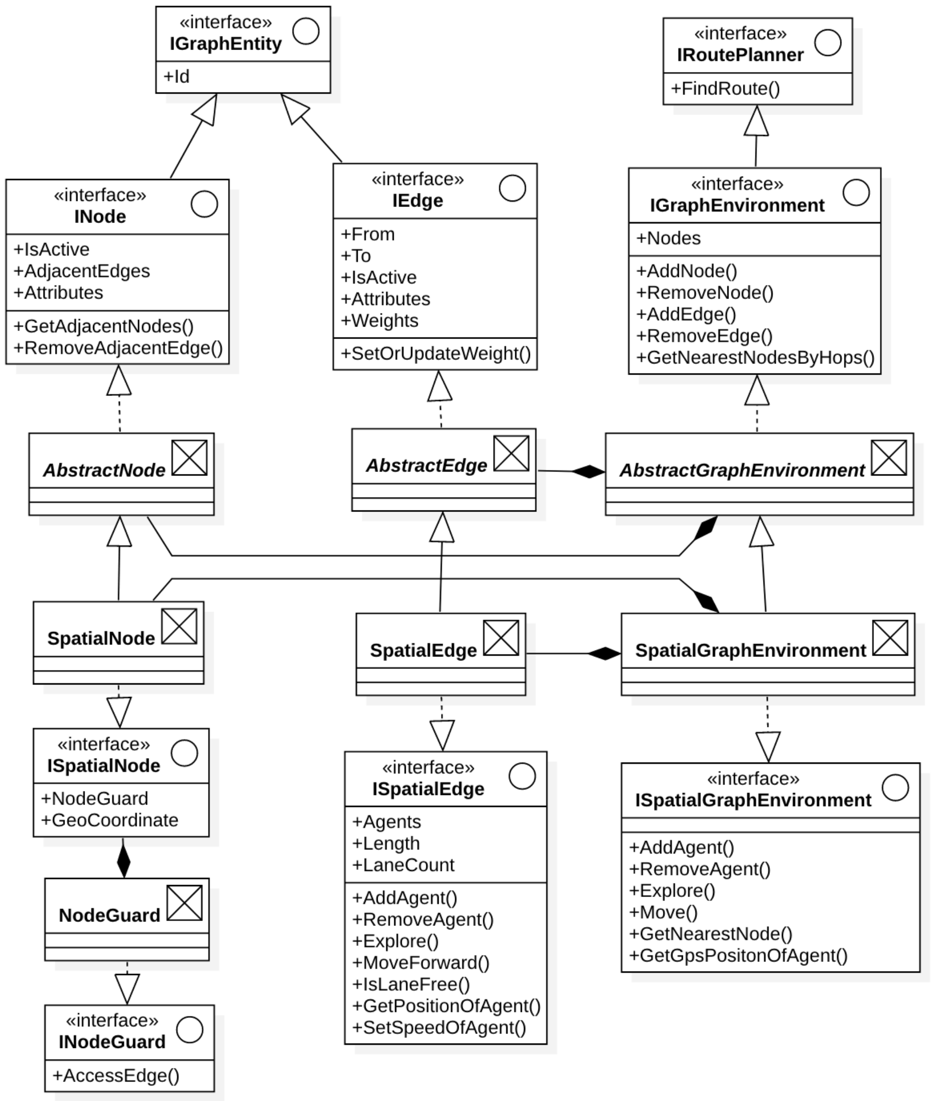


Abbildung 5.2: Spatial graph environment

The *Spatial Graph Environment* extends the abstract graph implementation by spatial components. Figure 5.2 shows the UML class diagram of it. Main implementation purpose was to depict streets so that it can be used to build traffic simulations. Therefore the *ISpatialGraphEnvironment* interface extends the existing implementation with new functionality to manage agents.

Information about the road network is taken from OpenStreetMap. Daniel Glake build an integration tool that enriches this data with information from other sources to collect as much data as possible. This so called *OpenData Discovery and Integration tool* uses OWS services and OData endpoints to query information. In a second step this data is being transformed into a MARS compatible format (GraphML). This file is stored on disk as input data and gets parsed during simulation initialization. From there a *Spatial Graph Environment* is being constructed that represents the data during the simulation and manages the agents.

The *Spatial Graph Environment* extends the base implementation by agent functionality which is needed for simulations. Agents can be added (*AddAgent*) to the environment and then use its methods *Explore* and *Move* to sense their surrounding and move on it. Additionally the *ISpatialGraphEnvironment* interface includes convenience functions like *GetNearestNode* which finds the closest contained node in the graph, based on a GPS position. This is used to make sure that the agents are inserted at the correct position. Querying an agents position in the SGE is possible too, through the *GetGpsPositionOfAgent* method. The SGE itself forwards the agent responsibility to the *Spatial Edges* where they are being managed. Each *Spatial Edge* contains a list of agents that are currently on said edge. If agents move through the road network, they are either moved on an edge or from edge to edge. Crossing an intersection means being removed from the first edge and getting inserted into the next. The SGE is implemented as collision free road network so the agents can position and spatial extend are stored as well. When moving an agent it is checked if the move would result in collisions and if so, it is prohibited. Agents request to be moved, based on their understanding of the environment which has been explored prior to moving. If that move is valid, the agent gets moved, otherwise it remains at its position. In both cases the agent is informed about that in the return arguments.

Intersections are represented through the *Spatial Node* class which now also have a GPS position. Additionally they contain a so called *Node Guard*. This construct was invented so that the flow of traffic can be managed by a controlling entity; for example a traffic light. When agents explore the environment this entity gets asked if the intersection can be crossed in the desired direction. If the intersection has no traffic lights this query will always be true, otherwise the traffic light checks its current status and then responds with the current light phase.

5.1.2 Intersections

Following up on the analysis part about intersections (3.1.2) this passage will demonstrate how the described challenges have been solved to meet the model requirements. Since this is first and foremost a data problem, a couple of simplifications had to be made. The city of Hamburg is providing much information about the city as part of their open data initiative but the road markings aren't part of it. Since detailed lane data won't be available anytime soon (this affects both streets and intersections), the following simplifications apply: Incoming roads at intersections are connected to every outgoing street except for the one that leads in the direction where the car came from (backwards). The lane count itself is known but not the direction where these lanes lead. As a result, the cars are allowed to continue their way on every adjacent road if the street has only one lane. For multi-lane roads the model is refined further. Due to the lack of data on the direction of lanes the following rules will apply: Cars are allowed to turn right only from the right lane. Same goes for left turning maneuvers which are only permitted from the left lane. Going straight ahead is allowed from any lane.

Before the agents could use the intersections properly, new functionality to explore incoming traffic had to be added. The spatial graph environment only allowed to explore along the desired route so that other traffic participants could be sensed. If the agent waited at an intersection, there was no way to obtain information about incoming cars at crossings. To account for this rather basic but very much need functionality, a new explore method was added to the SGE interface. This allows to explore an edge at an intersection and reveals the distance and speed of any incoming cars. Before this functionality could be added, the SGE had to be extended though since there was no concept like incoming edges at an intersection. As described above, the SGE only manages nodes, not edges. If information about and edge is needed, this has to be retrieved through the starting node since that one has a list of all outgoing edges. The other node (incoming node/ destination) had no such list and therefore only knew about its own outgoing edges. To make the exploration possible, every node now holds two list for its incoming and outgoing edges. This extension allows the car to explore both in their driving direction to sense other cars around them as well as to sense incoming cars at intersections.

5.1.3 Traffic counts

Counting the agents that move through the simulation is required in order to compare the results to real traffic counts. The measurements from the city of Hamburg are divided into 15 minute intervals so it was decided to follow the same approach. Each traffic count from Altona specifies the road it was taken on as well as the streets cardinal direction. Since the traffic in the simulation must be counted for the exact same road, first the representation in the SGE had to be found. The data used during its initialization contains so called OSM ID's that are contained when extracting data from OpenStreetMap. These ID's are used to uniquely identify each entity like roads, intersections trees and everything else there. In order to match the real traffic counts to the simulated street, for each counted road the corresponding OSM ID had to be found. This was a manual step since there was no way of doing in in an automated way. Once these ID's were known, the input file for the SGE had to be modified. Each edge that is supposed to count traffic was extended by an attribute called *TrafficCount*. During the initialization process these attributes are parsed and if said attribute was present the edge's counting functionality was activated. Counting agents is then performed by the *Spatial Edge* class which has a counter for passing agents. Each time a new agent is added during a *move* operation the counter gets incremented. The results of these counts are collected by the *RoadLayer*. This layer is also the base layer for the cars as they *live* on it. After 15 minutes in the simulation, all counts are collected and the result persisted to a CSV file. For each counting point this file contains a line with the timestamp, OSM ID and the number of counted agents. Once the result have been persisted to disk, the counters are reset and the process starts over again.

5.1.4 Traffic signs

Information about traffic signs is not widely available at the moment. At least not through Open Data sources like the GeoPortal of Hamburg or OpenStreetMap. For Germany the traffic signs include a wide variety of categories to warn of danger, state the right of way, regulate the speed and tell people where to drive and not to drive. Of all these categories, only the speed limits could be obtained for the simulated areas as they are contained in the OpenStreetMap data. The remaining signs therefore have been neglected and the cars have no rules to react to other restrictions besides speed limits.

OpenStreetMap collects a few key aspects of every street with the speed limit being on the most important (besides location, length and amount lanes). This information is available for most streets in Hamburg Altona and is automatically being extracted through the ODDI tool. As a result these details are contained in the metadata for edges. During the initialization process of the *Spatial graph environment* that data is loaded into the *attribute* collection of each spatial edge. This is not available for all edges as they could be as short as a couple of meters but for the most of Hamburg. Agents can directly access this information for each edge in their surrounding or their route by checking for the attribute *maxspeed*.

5.1.5 Traffic lights

At the moment of writing the this thesis the MARS group didn't have access to detailed data on traffic lights and their schedules. This will change in the future though. To prepare the simulation for that case, this model will contain a traffic light prototype that includes the described behavior. The traffic light system that operates an intersection has to comprise schedules for the adjacent roads so that the individual signals steer the incoming traffic over the intersection.

The traffic lights were implemented, following the requirements, formulated in the analysis section. A dynamic entity is needed to perform the task of signaling the incoming traffic so that each road gets their turn to cross. For these turns, schedules have to be made and executed that are tailored to the individual intersection. Throughout the day, different programs have to run so that rush-hour is treated different from regular traffic.

In MARS simulations, dynamic behavior can be achieved in two ways, either through agents or layers. On the layer sidee there are a couple of possible ways to incorporate dynamics. The first layer type are *active layers* where the layers have a *tick* method that can be used to execute code. This is similar to agents who use this method to go through the *sense*, *reason* and *act* stages. As the agents and layers aren't triggered in order there is no way of knowing when the signaling actually takes place. For the simulation this means that the first executed agents in a simulation step that arrive at an intersection could see a different signal than 'later' agents because the *tick* method of the layer was executed in the meantime and the lights have been switched. This is very fuzzy and

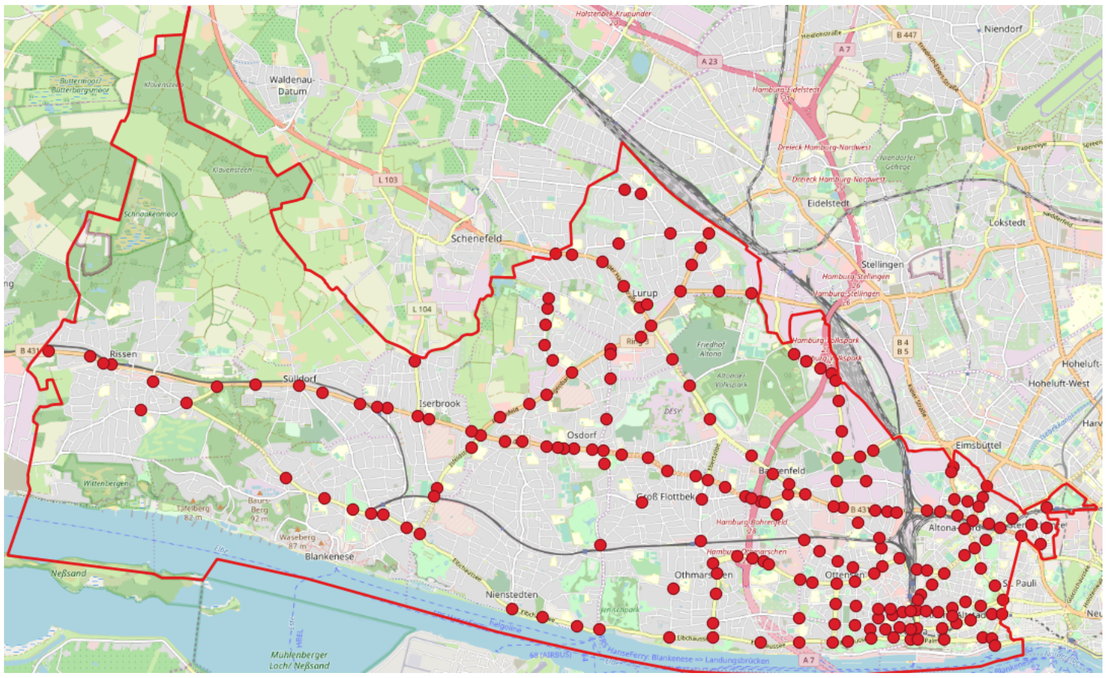


Abbildung 5.3: Traffic light positions in Altona

could lead to problems like car accidents when different roads are shown the same signal, therefore this approach isn't valid and will not be used.

If the traffic lights were implemented as agents, this problem would arise too. Even though dynamic behavior in a simulation like the switching of lights would be a perfect fit for the agent concept, the fact that these agents would perform their actions at the same time as car-agents would blur the simulation. It would be impossible to tell simulation errors from the dynamics evolving through in-tick changes of the lights.

The next layer option, so called *stepped active layers*, extend the *active layer* concept in the way that they have additional *pre-tick* and *post-tick* methods. The order of execution is 1) *pre-tick* methods, 2) *tick* methods of both layers and agents and finally 3) *post tick*. One of these methods can be used to perform signal changes prior or after the agents have executed their actions thereby resolving the 'fuzziness' issue.

The last layer option for dynamic behavior are GIS layers which come in two forms, GIS-raster layer and GIS-vector layer. They have the ability to represent either rasterized or vector-based information and thereby expose it to agents. Vector-based data can be points, lines or polygons while rasterized data incorporates grid-based information. For the traffic-lights, a GIS-vector layer can be used to represent the coordinates/ positions of traffic lights and initialize them in the simulation. Additionally the GIS-layers implement time-series capabilities that allow to swap information at runtime. This 'information' can be of any type, as long as it is linked to a geo-referenced position. In the past this has been used to include temperature and precipitation data in a simulation but it can also be used to include switching times for traffic lights. Swapping these files at runtime would mean switching light programs depending on the time of day.

To sum up the available solutions on the layer site: Stepped active layers can be used to perform actions before or after the agents are executed so that fuzzy behavior can be avoided. The GIS-layers have the ability to represent data with spatial reference and can also exchange that information at runtime. If parts of these two concepts were combined, the requirement of running different schedules throughout the day could be met. Additionally the GIS-layers spatial capabilities can be used to import the traffic light positions into the simulation.

At this point the actual light switching capability is still missing. There is neither an agent nor a layer that can manage and change the lights according to the schedule. Each intersection has to be treated as individual entity with its own schedule that is tailored to that very crossing. Within that management, the system has to map incoming roads to outgoing roads and assign them an actual light. When car-agents arrive at an incoming road they check the current status of it and then continue with their activity planning.

Whether intersections have traffic lights or not is determined through data from Hamburgs *Geo-Portal* where open data is made available to the public. The positions of all traffic lights as of 2018 have been downloaded as shown in figure 5.3. To be more precise, the geographic locations of all intersections that have traffic lights have been downloaded. The data allows for a clear distinction between intersections that have traffic lights and those who don't. There is no data available on the schedules. To connect this information with the road system in the simulation, the *Spatial graph environment* has to be aware of it since the agents 'drive' on roads, that are managed there. When building the SGE the concept of *node guards* was introduced to resemble controlling entities on intersections. Node guards can be anything ranging from a barrier, over a traffic policeman to a traffic light. Agents exploring the environment are automatically informed about the status of the intersection's node guard. This object is being asked for its current status and thereby, whether the car may pass in the stated direction. The *INodeGuard* interface was created to allow objects of different types to implement this behavior.

Traffic light design

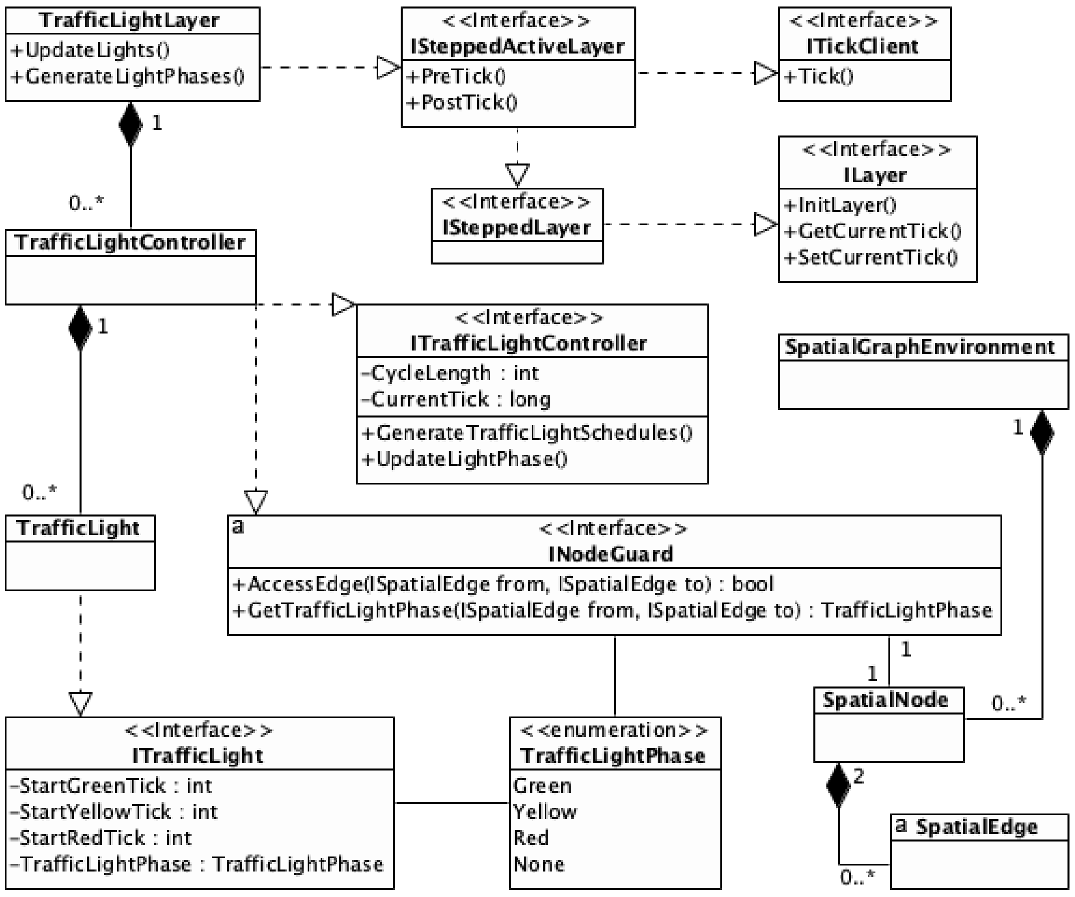


Abbildung 5.4: Class diagram of the traffic light layer with the involved interfaces

The traffic lights have be realized as a new custom layer. It was created inside a new module so that it can be reused in different models. Figure 5.4 shows the class diagram of the layer. As described above, the traffic light layer is an *ISteppedActiveLayer* so that it has *PreTick* and *PostTick* methods where the traffic light changes will be executed. For completeness the overlying interfaces where the three *tick* methods originate are shown too. Per intersection there will be an *TrafficLightController* that manages all lights on that crossing. Different lights signal the various incoming roads and the traffic light controller manages all of them, thereby resembling the control system. This controller implements the two interfaces *ITrafficLightController* and *INodeGuard*. By implementing the latter it will gain the required signature to be inserted into the *Spatial graph environment* as a *node guard*. On the bottom right of figure 5.4 the *Spatial Graph Environment* and its components are depicted to show the relationships. Attributes and operations were left out on purpose since they aren't of relevance here. The *ITrafficLightController* interface includes the *UpdateLightPhase* method which will be used to update the currently shown light phases in the simulation step. Each controller has a cycle length field that represents the the intersections turnaround time as described in the analysis chapter. With the *currentTick* field, the progress is tracked and the value reset, once the cycle has completed so that the light program runs in a loop. Since data on schedules is currently not available, the *GenerateTrafficLightSchedule* method will forge these and populate the individual traffic lights with an appropriate schedule. At meetings with employees of Hamburgs *LSBG* (Landesbetrieb Straßen, Brücken, Gewässer) we learned that the turnaround time of most traffic lights in Hamburg is 90 seconds. Since more detailed information about individual intersections is currently not available, the default value is set to that. During this time, the traffic light controller gives every incoming street the signal for driving one at a time. The length of each directions driving time depends directly on the amount of adjacent roads as the turnaround time is divided by their number. The last component of the layer are the traffic lights themselves. For every connection between an incoming and an outgoing road there is an individual traffic light that signals the incoming agents. Each light stores the schedule in which tick it changes the status from green to yellow to red as well as the current status. The *current tick* is determined through the traffic light controller's current step of program execution. Once the turnaround time of the overall traffic light system is through (90 seconds), it starts from the beginning. The individual traffic lights are thereby repeated in a loop until the traffic light controller switches to a different program.

Since the traffic light system is not the main focus of this thesis, a couple of simplifications have been made that allow to include the system without over-complicating it. In a later version of the Altona simulation, a more precise version can be implemented. Hopefully there will be ground data on light schedules by then.

1) Incoming roads receive the same signal for all directions so there is no separate lights for right, left and straight ahead. 2) The traffic light system of an intersection gives way to the incoming streets one at a time so that every road has the same priority. 3) The turnaround time of each crossing is set to 90 seconds, as described in the analysis chapter.

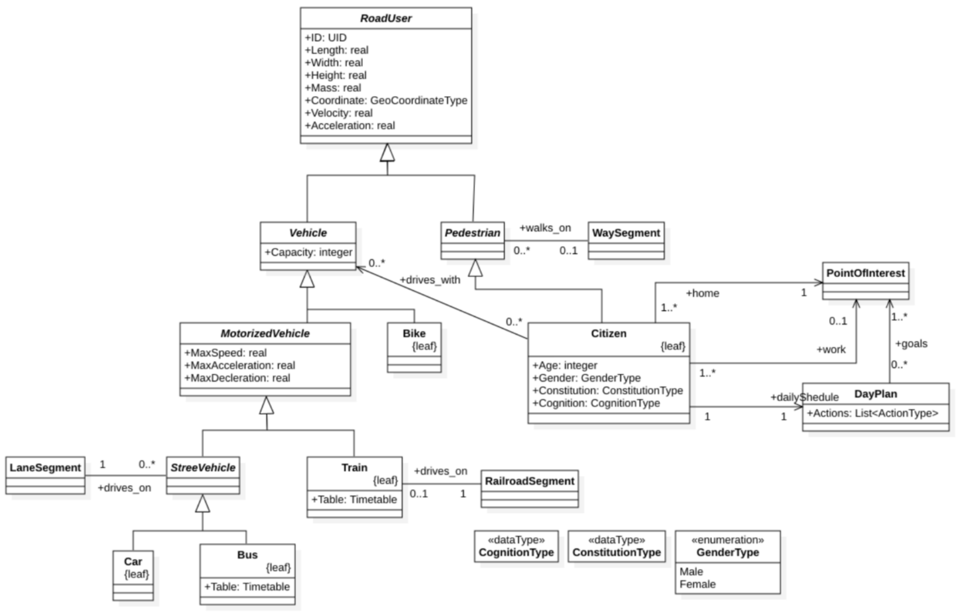


Abbildung 5.5: UML diagram of the agents classes for SmartOpenHamburg

5.2 Car agent modeling

In this section the car agent implementation is described. Before the agents get to drive around they have to be initialized properly. Since this model has been developed as generic as possible, there are many ways to configure the agents through input data. The agent (*Car*) class will be part of the SmartOpenHamburg model for which an inheritance hierarchy has been set up. Dividing all traffic participants into various categories allows for an easier implementation later since there is a common set of attributes. It starts with the most generic form *Road User* as shown in figure 5.6 which collects the most common attributes each individual will have. These include spatial extend (length, width, height), mass, a position and the current movement parameters velocity and acceleration. From there the right side of figure 5.6 covers the pedestrians which are developed by Andreas Löffler and the vehicles. These were further divided into motorized vehicles and non-motorized vehicles of which the bike is the only representative. For the motorized vehicles there is a distinction between street vehicles and trains, the latter being the basis for the public transport implementation by Daniel Glake. The cars directly inherit from the street vehicle class and extend this by the following parameters.

5.2.1 Constructor

Since the model has been implemented with a wide variety of purposes in mind there are a few required parameters and even more optional ones. Before these get discussed, first the MARS specific, standard parameters have to be set. The basic parameters for all MARS agents are the *layer* on which the agent will be managed and the *environment* the agent uses to move around the simulated area. These vary from simulation to simulation and can be either located in Hamburg or in the Kruger National Park, based on the scenario. Before and agent is part of any simulation it also received two delegates to register itself and to unregister itself from the simulation core. If an agent unregisters itself, it gets removed from the simulation core and will be deleted after the current simulation step has been completed. This delegate is invoked once the car has reached its destination. Following up are all available parameters including a description what they are being used for and whether they are required or optional.

length Denotes the cars length in meters. This is used by in the *Spatial Graph Environment* as physical size. When agents move in there this length gets checked so that cars don't bump into each other.

height, mass, width These parameters describe the spatial extend of each car. For the cars driving they don't play a role as their main goal is to describe the outer shape.

maxAcceleration is one of the most important parameters for the driving physics. This setting is passed to *Intelligent Driver Model* to calculate the acceleration/ deceleration in each simulation step. Increasing the value leads to faster acceleration while lowering it mitigates the cars capability to reach its maximum speed.

maxDeceleration states how hard the car should brake in regular situations. This is mapped to the IDM's *comfortable deceleration* parameter. Based on this the acceleration/ deceleration is calculated.

maxSpeed specifies initially how fast the car is allowed to drive. This changes during the simulation when the cars react to speed limits.

velocity (optional) When the cars are created, they are inserted into the environment with a velocity of 0 km/h. If cars should be inserted at higher speeds, this parameter can be used and set to the desired velocity.

driveMode sets the desired mode of operation for the agent. The next section describes each drive mode in detail. In short: the car can operate in different modes that allow to either drive around randomly, use route finding, start at a designated location and or with a preset destination.

startLat (optional) The start position is coded as GPS with latitude and longitude components. As the mapping only allows for primitive types the coordinate was split in two parts with this parameter being the latitude.

startLon (optional) is the second part of the starting position. The longitude is coded as double and inserted into the constructor.

destLat (optional) Latitude part of the destination GPS coordinate. This sets where the agent is supposed to go in in conjunction with the *destLon* parameter

destLon (optional) describes the longitude where the agent is supposed to go. To be used alongside *desLat* as part of the GPS coordinate.

startingEdge (optional) This parameter can be used to specify a starting edge. The agent still starts at a node but his first edge in the route will be set to the desired edge.

capacity (optional) of the vehicle meaning how many people fit into the car. Not used at the moment but important for the later SmartOpenHamburg model.

stableId (optional) For testing purposes the agents can be outfitted with a stable ID that stays the same in every test. This allows to assert if a certain agent behaves the desired way and was used a lot in the tests for the intersection behavior.

leftHandDriving determines on which side of the road the agents drive. This can either be set to 0 meaning right hand driving or to 1 which results in the agents driving on the left. In the default configuration the cars drive on the right side.

trafficCode tells the model which country it is driving in and thereby which traffic doe to use. For Germany this has to be set to *german* while South Africa is coded as *south-african*. In the default configuration the cars use the german traffic code.

route (optional) This parameter can be used to pass a pre-computed route to the model which is only being used in drive mode six. The string must contain a list of OSM ID's separated by commas.

Drive modes

In order to run the agent model in various circumstances, a couple of different drive modes were implemented. These modes allow to use and instantiate the cars in the following configurations:

- Random driving (1): the car randomly chooses a location in the simulated area and starts driving there. There is not routing involved and the agent picks its way randomly at every intersection. This is very useful for benchmarking scenarios where the route finding shouldn't influence the results as calculation this for multiple thousand agents takes its time.
- Random start and goal with routing (2): As before, the car agents starts at a randomly chosen position in the simulated area. Instead of driving around without a goal, a random destination is picked and a route calculated that the agent follows. Once it reaches its destination, the car is being removed from the simulation. This mode of operation is helpful when data about realistic origins and destinations is missing. Including route finding between the random start and goal leads to realistic usage of higher order streets due to the route finding. This mode was used for the simulations in the Kruger National Park.
- Drive mode three (3): When start and goals are known prior to the agents creation. This mode was used to run all simulations in Altona as the day-plan generator provided start and destinations. Route finding takes place in the constructor so that the agent follows a realistic route. To use this mode the *startLat, startLon, destLat* and *destLon* parameters have to be set while creating the agent.
- Mode number four (4) is a combination of the aforementioned. Based on a known starting location the agent picks a random goal and uses route finding to get there. This route is then followed until the goal is reached and the agent removed from the simulation. This mode was created to insert commuters at the simulated area which would drive from the known influx point to a random destination. These experiments will be executed in the future and aren't part of this thesis.
- Another drive mode (5) was developed to spawn agents in the simulated area. For scenarios where agents should start at an intersection and head out in a specific direction (on a specific road) this drive mode expects the starting point and the desired edge as input parameters. From there, the agent is inserted into the desired

edge, a random destination is picked and the car starts driving based on a calculated route.

- The last driving mode (6) is targeted at situations where the start, destination and the route is known. The route is coded as a list of OpenStreetMap ID's which is then translated to an actual in-memory route. This is done through the *Spatial Graph Environment* which parses the edge attributes to derive the actual route. This mode was mainly used for testing purposes where the cars had to follow a certain route.

5.2.2 Intelligent driver model

Following the rules in the next section, the agents will at some point calculate the distance they want to move (drive). Foundation for all movement/ distance calcuations is the *Intelligent Driver Model* as described in the methodology chapter. This model is used to calculate the overall driving behavior in latitudinal direction. More specifically, it encapsulates the differential equation, used to calculate acceleration and deceleration. The car agents call this model component with their individual parameters for max acceleration, desired gaps etc. and receive instructions on how to adapt their velocity. The IDM itself has been implemented as separate, static class with a single function to calculate the acceleration based on input parameters. As such is being used by all agents. Since the *Intelligent Driver Model* has been implemented as separate class, it can easily be replaced. This allows to add other modules or car-following models like the Wiedemann equation.

5.2.3 Agent rules

Following the high-level goals defined in section 3.2 (Agent requirements) this section will list the precise agent rules. These rules are derived step by step from the coarser defined actions. The simulation model consists of multiple functions working together to allow the cars to process inputs from the environment and decide on the next actions. Figure 5.6 shows an activity diagram of the agents reason method which determines its behavior in every tick.

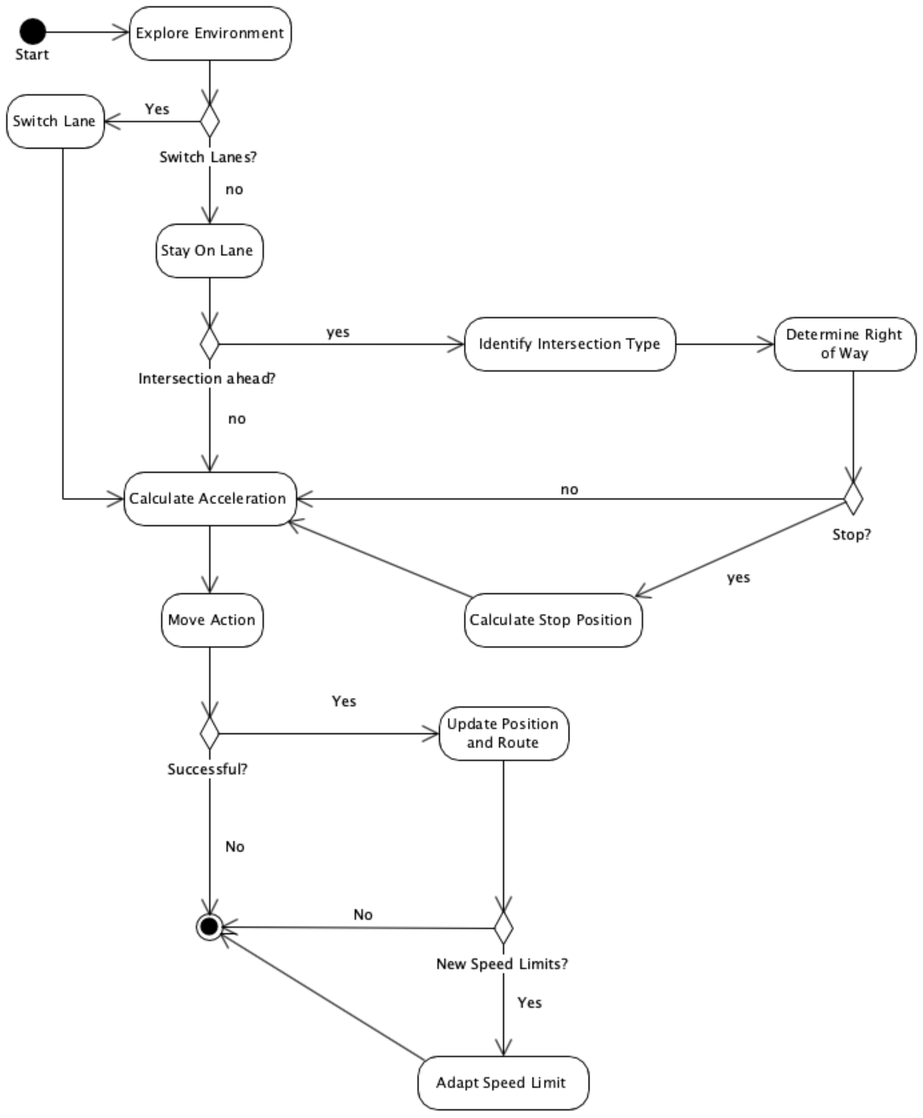


Abbildung 5.6: Activity diagram of car-agent

Explore and agent field of view

The agents have to sense their environment before any actions can be performed. In every simulation step the agent starts by exploring the road around itself. This is done by using the *Spatial Graph Environment's Explore* method which returns information about the cars on either side of the agent and in front. The sensing has to obey the non-functional requirement of not affecting the performance beyond unjustifiable means so the required distance of exploration had to be calculated. Initially it was set to 30 meters which limited the field of view at higher speeds too much. The agent had to perform harsh braking maneuvers as it ”saw” obstacles very late. To overcome this, a dynamic field of view has been implemented which changes with the current speed. The agent calculates this field of view by multiplying its current velocity in m/s by six which lets him ”see” far enough even at high speeds. The minimum field of view is kept at 30 meters as the agents don't need to see too far when they are standing still or at an intersection.

The exploration reveals the cars in front, the next intersections and details about the road the car is currently driving on. This is the basic information used to sense the current circumstances. Analyzing the explore results starts with calculating the remaining distance on the road which is used later in the acceleration/ deceleration process. Next, the intersections up front are stored in a list with the remaining distance to it, the node which represents the crossing and the current light phase, should there be a traffic light. After this the same is done for cars around the agent each with the distance, position and current speed. This concludes the interpretation/ sensing process as the results are now sorted into categories that the agent uses to plan its next action.

Route and lane management

Routing is a key influence for the car-agents behavior as it defines both the road to take as well as the lanes to drive in. While the route itself is important, the lanes are the bigger challenge. Cars are allowed to drive in any lane as long as they go straight at the next intersection. The turns introduce complexity as the car has to detect them and switch on the correct lane prior to making its turn. Following this initial step the agent then decides whether it is going to stay on the same lane or switch based on the exploration results. If the current lane is valid the agent continues and won't check the lanes again until it has progressed to the next road. Should the route require a lane change, the agent will check the space next to it on the desired lane to see if there is enough space to pull over.

With enough space available, the lane is changed and the car proceeds with its rules. This process is repeated in every simulation step until the lane change was successful or the end of the edge is reached. In that case, the agent proceeds in the desired direction anyway which isn't ideal but was necessary to prevent deadlocks. One of the next steps for the agent model, the implementation of a dedicated lane changing is planned. This should allow for more complex maneuvers like overtaking as well.

Biggest deceleration

Before describing the next processes for dealing with intersections, a fundamental part of the model has to be explained: the so called *biggestDeceleration*. In each simulation step, each agent uses a double field with the specified name to track the hardest braking maneuver. Following the agent rules there are multiple values for acceleration/ deceleration calculated, based on different data and for different use-cases. This can be explained best with an example: if an agent was to drive on an empty road, there are no other cars it has to slow down for. Calculating the desired speed for the current tick based on the IDM would suggest that the car keeps driving at full speed. At the same time the agent might be closing in on an intersection which requires him to slow down, leading to a lower value for acceleration. Comparing these two acceleration/ deceleration values shows that the second value is lower than the first. If the agent chooses to use the second one to calculate how far it moves in this step, it automatically does the right thing. The agent naturally slows down instead of following the guidance of the higher (wrong) suggestion calculated through the car-following model. Whenever an agent calculates a value for deceleration it compares that to the currently lowest one and sets the new value as lowest, should it be even smaller. This way the various agent rules are balanced without the need for nested if/ else statements.

Intersection behavior

There are many types of intersections, all comprising an individual set of rules. When agents approach an intersection, they have to find out what to do. Based on the collected exploration results, the upcoming intersections are processed first. When an agent approaches an intersection it will slow down as long as the light isn't showing a green signal which is the first thing checked in every tick. Then, based on the agents current road and the next one in the route, the turning angle is calculated to determine the the optimal speed. The angle is sorted into one of the following categories: backwards, sharp left, left, wide left, straight, wide right, right or sharp right. For sharp turns (right or left) the agent reduces its speed to 2.7 m/s, for regular turns to 4.1 m/s and for wide turns to 5.5 meters per scond. If the agents drives straight to the next road, the velocity will not be reduced (by this part of code). Once the value is known, the required deceleration/ acceleration is calculated and compared to currently biggest deceleration changing it if the turning value is lower. The agent reduces its speed over multiple ticks so that the speed is reduced incrementally.

In the next step the intersection itself is checked to see if it has a traffic light or not. For those crossings with traffic lights, the current light phase is extracted from the exploration results. If the light is green the agent proceeds with its basic behavior since the he doesn't have to stop. For yellow traffic lights the agent asserts whether the required deceleration exceeds the configured maximum deceleration. This equates to the car checking if it can come to a stop in time without performing an emergency stop. Should this be possible, the agent starts braking by setting the *biggest deceleration* parameter which leads to a full stop over the next ticks. If the car is too close already, it proceeds at its current speed and crosses the intersection during the yellow phase. When the agent approaches red lights it calculates how much it has to brake in order to come to a stop in time and does so. Should the traffic light status change over the course of the next simulation steps, then the agent adapts its behavior to the new situation. If the agent was to approach a red light which then turns green, it would stop braking and start accelerating and vice versa.

Should there be no traffic light on the intersection, the chosen traffic code is checked and the respective method used to determine the correct action. For the simulation in South Africa, the same car-model is used so at this point the differing rules have to incorporated. Depending on the location and the simulation config this is either the German traffic code or the one from South Africa. The constructor parameter *trafficCode* sets this at initialization.

German traffic code

Before reaching an intersection the speed is reduced so that the car can come to a stop, should the need arise. After this the incoming traffic at the intersection is checked to sense other cars approaching the intersection. Traffic from the right has precedence over the currents agents movements so the agent will stop if an agent approaches from there. Once the agents from the right have passed, the car continues its drive. The second rule that applies to left yields to right intersections is that left turning maneuvers are only allowed if there is no opposing traffic whose path would be cut. If there is coming from the front, these cars have priority and the car won't continue until they are out of the way. For situations where two opposing cars wan't to go left this would lead to a deadlock situation that had to be resolved by allowing one car to proceed. Following the described rules for left turns and incoming traffic from the right, the german intersection type of left yields to right has been implemented. After going through these rules and deciding on the correct action, the agent uses the IDM to calculate the required acceleration. This is then checked against the *biggestDeceleration* parameter to ensure safe behavior without collisions.

South African traffic code

As described in the simulation scenarios, the traffic in South Africa drives on the left side in contrast to Germany. Additionally the default behavior at intersections changes from left yields to right to so called four way stops. At such intersections the traffic has to come to a complete stop and then proceed in the order of arrival at the intersection. The first thing every agent does to follow this rule is to come to a stop, even if there are no other agents around. During the approach, the incoming cars are tracked by each car to determine the order of arrival. After coming to a stop the car checks this order to determine if it is allowed to drive. Based on the exploration the car updates this list in every simulation step and removes the cars that arrived earlier after they have crossed the intersection. If it is the cars turn to cross the intersection it does so by calculating the acceleration through the *Intelligent Driver Model*. Once the intersection has been crossed the list with incoming cars is emptied.

Interacting with other cars

After checking the intersections, the next step is to react to cars on the surrounding lanes. During the initial exploration the positions of the cars left, right and in front of the agent have been determines as well as their speed. If there is a car ahead, the agent calculates its next move by incorporating the distance and speed of the agent ahead into the acceleration/ deceleration calculation. When there is no agent in front or the gap is big enough the agent drives with up to its maximum speed. Should the next agent be too close, the car reduces its velocity to prevent the collision. Again the *biggest deceleration* mechanism is used to select the maximum required deceleration to behave correctly. The check for other cars concludes the agents planning phase.

Moving the agent

Now that the agent has checked all relevant details about its surroundings, the perfect amount of acceleration/ deceleration has already been chosen. This is now added to the current velocity (deceleration is negative acceleration so this value is reduced) which is then used as input for the *move* command. If the calculated distance to drive is equal to zero the agent stops, otherwise the move command is used to calculate the new position. Input for the move command is the agent itself as entity to be moved, the desired driving distance and the route including the lanes. If the move succeeds, the method returns information about the new position which is processed using another method (*ProcessMoveResult*). The move does not necessarily succeed as another car could have moved to the desired position so there is always the possibility of having to brake. However this only happens if another car that is being simulated in another thread does its move on the same road in between the explore and move method of car one. For positive movements the returned information includes the new the position and lane it currently occupies. Should the agent cross an intersection, the new edge is returned as well. Based on this information the car then calculated its current GPS position and sets this in its parameters so it gets persisted after the simulation step. The GPS calculation incorporates the edges curvature which is stored as list of intermediary points. Based on these points, the new position is determined so that the agent drives on the actual road.

If the car changes the road (crossing an intersection) the following, additional steps are made: Information about the allowed maximum speed is contained in the data from OpenStreetMap. The agents can access this information directly through the *Spatial Graph Environment* where the downloaded data is being stored in-memory. By reading the associated attributes of the current edge, the agent adapts its parameter for maximum speed to the current limit. This parameter is then being used for the acceleration/ deceleration calculation in the next simulation steps. Additionally, the car updates its route by removing the already driven roads.

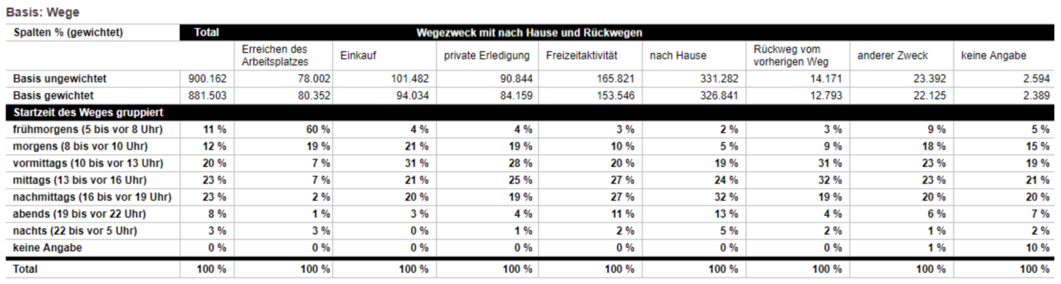
5.3 Scenario modeling

Now that the environment has been set and the agent rules are in place, the simulated scenario has to be modeled. For Altona this means creating a virtual population whose combined travel activity make up the traffic there.

5.3.1 Virtual population and day plans

Where people come from and where they are going is not the focus of this thesis but it plays a major role in the simulation. A colleague, Andreas Löffler developed a so called *day-plan generator* that is based on demographic data and uses statistics to create travel patterns. These patterns represent a real humans trips, made throughout a typical day. Combining these travel patterns from thousands of citizens creates a virtual population who's joint travel activities make up the city's traffic. To simulate traffic in Altona, the agents have to drive from one point to another. In the beginning these origins and destinations were picked at random but that doesn't depict the measured traffic flows. As a next step it had to be figured out when the agents would drive to which location to resemble real human behavior. This has been done by Andreas Löeffler as part of his master research in which he developed mechanism to create day plans for each individual living in a simulated area. The agents can spend time at home (*hometime*), go out to work, eat, run errands or enjoy their free-time. Based on demographic data he determines the probability for each of those categories and thereby derives behavior schemes for the agents. Main source for his day-plans is the study "Mobilität in Deutschland"[3] which covers a wide variety of topics from socio-demographic information about citizens to their average daily travel patterns. This study that is repeated regularly, published their results in form of an online tool that allows to directly extract relevant information about the travel patterns as shown in figure 5.7.

The columns represent the activity categories and the rows are the time slots in these the activities take place. This shows for example that for the early morning hours from 5am to 8am, 60% of all trips to work are made while only 3% of free-time activities take place. The columns are independent from each other and only show how many of the trips belonging to this category (out of 100%) are being made in the respective time slot. This data is the basis to determine when agents start their activities throughout the day.



Basis: Wege

Spalten % (gewichtet)	Total	Wegezweck mit nach Hause und Rückwegen							
		Erreichen des Arbeitsplatzes	Einkauf	private Erledigung	Freizeitaktivität	nach Hause	Rückweg vom vorherigen Weg	anderer Zweck	keine Angabe
Basis ungewichtet	900.162	78.002	101.482	90.844	165.821	331.282	14.171	23.392	2.594
Basis gewichtet	881.503	80.352	94.034	84.159	153.546	326.841	12.793	22.125	2.389
Startzeit des Weges gruppiert									
frühmorgens (5 bis vor 8 Uhr)	**11 %**	**60 %**	**4 %**	**4 %**	**3 %**	**2 %**	**3 %**	**9 %**	**5 %**
morgens (8 bis vor 10 Uhr)	**12 %**	**19 %**	**21 %**	**19 %**	**10 %**	**5 %**	**9 %**	**18 %**	**15 %**
vormittags (10 bis vor 13 Uhr)	**20 %**	**7 %**	**31 %**	**28 %**	**20 %**	**19 %**	**31 %**	**23 %**	**19 %**
mittags (13 bis vor 16 Uhr)	**23 %**	**7 %**	**21 %**	**25 %**	**27 %**	**24 %**	**32 %**	**23 %**	**21 %**
nachmittags (16 bis vor 19 Uhr)	**23 %**	**2 %**	**20 %**	**19 %**	**27 %**	**32 %**	**19 %**	**20 %**	**20 %**
abends (19 bis vor 22 Uhr)	**8 %**	**1 %**	**3 %**	**4 %**	**11 %**	**13 %**	**4 %**	**6 %**	**7 %**
nachts (22 bis vor 5 Uhr)	**3 %**	**3 %**	**0 %**	**1 %**	**2 %**	**5 %**	**2 %**	**1 %**	**2 %**
keine Angabe	**0 %**	**0 %**	**0 %**	**0 %**	**0 %**	**0 %**	**0 %**	**1 %**	**10 %**
Total	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**

Abbildung 5.7: Purpose of making a trip in relationship to the time of day

Individual factors like sex, occupation, place of residence and availability of a car are taken into account next to further refine the model. Occupation is one of the main factors since the travel patterns of full-time workers and part-timers differs strongly from unemployed people or those who work at home. These three categories (full-time, part-time, unemployed) are coupled with demographic data about the total inhabitants to generate the initial virtual population. The data used for this distinction is the "Statistisches Jahrbuch Hamburg", a report about Hamburgs demographics and socio-economic status that gets published yearly [1]. At first the occupation distribution among the agents is being calculated and divided into the three categories full-time, part-time, unemployed based on the report. Full-time means 39.5 hours per week and part-time is 26 hours per week on average. The trips made throughout the day are divided into 36% free-time, 37% errands and 27% work and have been calculated for weekdays, not weekends. Once each citizen has been assigned its work status, the next step is to generate a chain of actions that depicts that agents travel pattern. Each action gets a starting time assigned so that the agents stay at the destination for an appropriate amount of time before making the next trip [14].

When the amount of trips made by car is to be determined, the biggest influence is car ownership. Car sharing is increasingly popular and especially citizens in larger cities like Hamburg have accounts for the ride-sharing services. As of 2017 up to 14% percent of the large city population is user of car-sharing [3]. Since the data supplied through the traffic counts and the "Statistisches Jahrbuch Hamburg" are based on data from 2016, car sharing will be neglected. For Hamburg Altona there were 89.705 private cars registered in the beginning of 2017. This equates to 332 cars per 1.000 residents [1]. Citizens, use cars to get around town so the cars will only be actively driving when such a trip is to be made. Based on the total amount of cars registered in Altona, trips for 89.705 citizens have been created. Each agent follows his plan, look up a location where his assignment can be fulfilled and then drives there. When the simulations are initialized, the first agents are created in the simulated area. These are the agents that are present there at the the first simulation time step. After this, more agents are inserted based on the calculated day-plans which contain the time of day they are to be created.

5.4 Implementation anecdotes

5.4.1 Figuring out directions

What is left and what is right? This rather simple issue in real life wasn't as easy in the model when it comes to intersections. Every adjacent road either incoming or outgoing has a direction. The unknown part is their spatial relationship to each other. So when the agents arrive at an intersection they may know where their route leads, but they didn't know whether this involved turning or if it meant going straight.

This implementation detail is being used both by the traffic lights as well as by the car agents. Car agents, in most situations, have a route that they follow. When they get to an intersection they naturally don't know in which cardinal direction the next road leads. What they know as a route is a list of roads without any information about their angles. So turning right is the same as going straight ahead. This doesn't resemble the real world though and had to be changed. Taking into account the bearing of arrival at the crossing, the angle to the bearing of the outgoing road is calculated. This angle is then used to classify the direction to one of eight directions. These are straight on, wide right, right, sharp right, backwards, sharp left, left and wide left. Depending on these directions the agents change their behavior and most prominently the speed at which they perform driving maneuvers. As a result the agents drive with unchanged speed straight ahead over a crossing with a green light while they decelerate before taking a sharp left. Details about the car agents driving behavior can be found above.

The traffic lights use the same implementation for a similar purpose. Since there is not data available that states which roads at an intersection are connected, the traffic light has to figure that out on its own. Determining the angle between incoming and outgoing roads, the traffic lights connect only those that would be connected in reality. For example there would never be a traffic light for making a U-turn while there most likely will be one for going straight ahead.

6 Results

This chapter collects the results from the experiments for hypothesis one to three. The discussion will take place in the next chapter.

6.1 Hypotheses One

The experiments for hypothesis one aim at answering the question whether the *Intelligent Driver Model* can be used as basis to build a rule-based traffic model. Additionally they allow to assess the agents behavior in typical situations found in urban traffic.

6.1.1 Basic driving

This section collects the results from the basic driving experiments. These test the cars acceleration and deceleration behavior.

Experiment: ”Accelerating”:
For the first experiment, the agent was tasked to drive on an empty road in order to assess its acceleration behavior. Figure 6.1 shows both the velocity and the acceleration over the course of the simulation. As described in the expected outcome part, the cars velocity increases steadily in the beginning with up to 0.73 $\frac{m}{s^2}$. Towards the maximum speed, the acceleration declines as the car reaches its desired top speed of 13.89 m/s (50 km/h). The results represent the exact values as calculated through the IDM. Comparing them to manual calculations shows no deviation.

Experiment: ”Regular deceleration”
Figure 6.2 shows the progress of velocity and acceleration during the second experiment. This time the cars deceleration behavior is simulated where the task is to come to a complete stop within 100 meters. The car starts with an initial speed of 13.89 m/s (50 km/h) and immediately reduces its velocity. After braking with up to -0.87 $\frac{m}{s^2}$ the

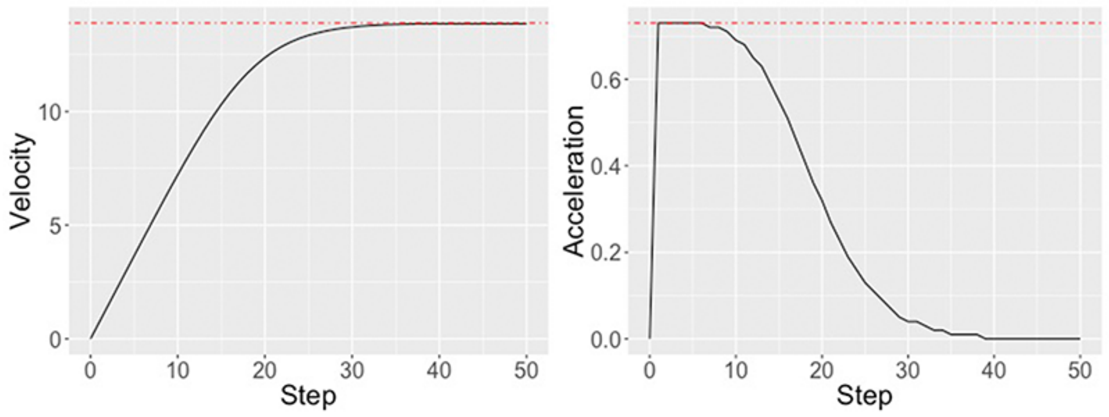


Abbildung 6.1: Car acceleration from 0 m/s to 13.89 m/s

deceleration is slightly reduced to -0.72 $\frac{m}{s^2}$ in tick three before braking even harder. In tick nine a maximum deceleration of -1.26 $\frac{m}{s^2}$ can be seen which is lower than the so called *comfortable acceleration* parameter specified in the IDM. From tick ten onwards the deceleration is constantly reduced until the car comes to a halt in tick 22. Calculating the IDM manually as reference shows no deviation.

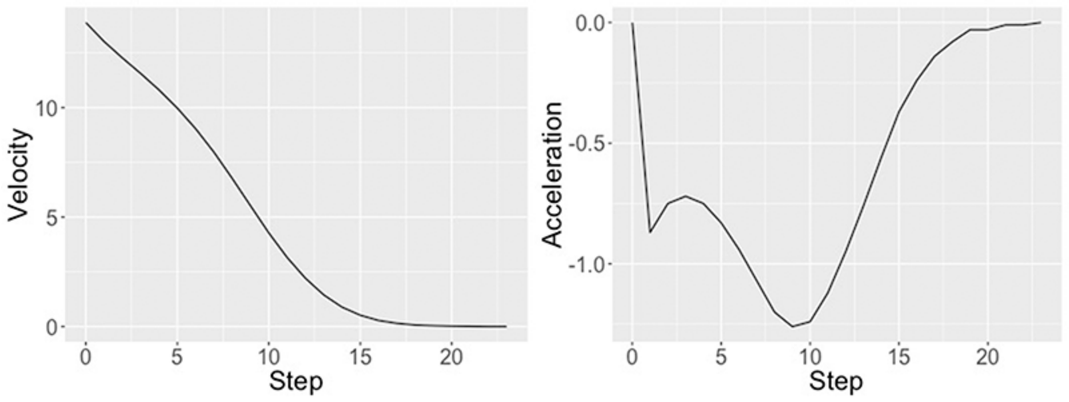


Abbildung 6.2: Car braking from 13.89 m/s to 0 m/s within 100m

Experiment: "Intense deceleration"
In the last deceleration experiment the cars reaction to unforeseen events is tested where the task is to come to a halt within 25 meters. Since the IDM keeps a safety distance to obstacles of two meters, the overall distance was increased to 27 meters. Running this scenario reveales a problem with the *Intelligent Driver Model.* Looking at figure 6.3 shows that the agent decelerates from 13.89 m/s to 1.42 m/s within the first simulation step (1 second). The deceleration is as high as -12.47 $\frac{m}{s^2}$ which points to the IDM not calculation

realistic values anymore. After almost coming to a complete stop in the first simulation step, the car then accelerates again before stopping when reaching the obstacle after 18 ticks. Since calculating the same scenario manually showed the same results, this shows a problem with the IDM. The *comfortable deceleration* value of 1.6 $\frac{m}{s^2}$ exceeded may times over. Instead of constantly applying the brakes until coming to a stop, the IDM plans an emergency braking maneuver for one second. That there is enough space left to accelerate afterwards, shows that deceleration calculated through the IDM is too high in such situations.

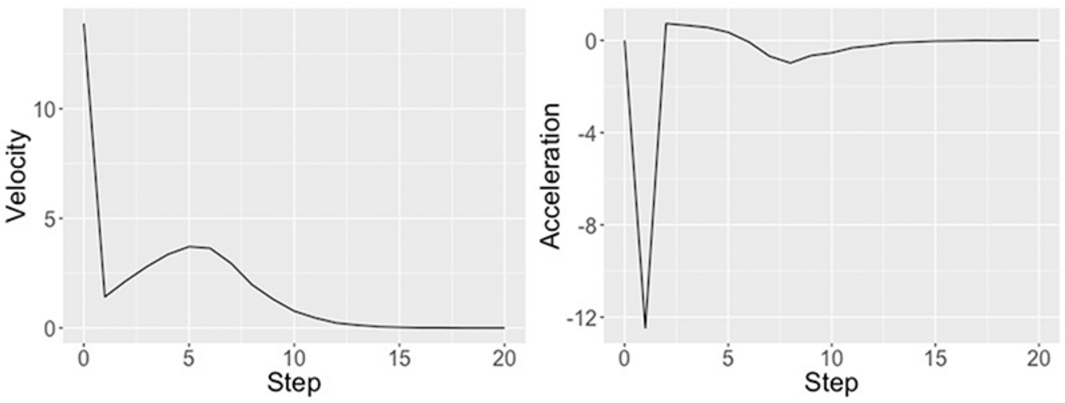


Abbildung 6.3: Car braking from 13.89 m/s to 0 m/s within 27m

Experiment: ”Driving on a road”
In the last experiment of testing the cars basic driving skills, its ability to follow the curvature of a road is tested. Figure 6.4 shows the GPS positions of the car driving on Vogt-Groth-Weg in Altona. Starting on the top right corner, the drive connects to the bottom left one in an L-shape. The dots show the cars positions over the course of the simulation which align with the shape of the road. It follows the curvature perfectly, passes the test and concludes the experiment successfully. Figure 6.5 confirms this for another road with a unique curvature in S-shape.

6.1.2 Speed limits

In this section, the results from the speed limit experiments are be presented. This includes those simulating an increase as well as those simulating a decrease in the allowed maximum speed of a road.

Experiment: ”Increasing speed limits”

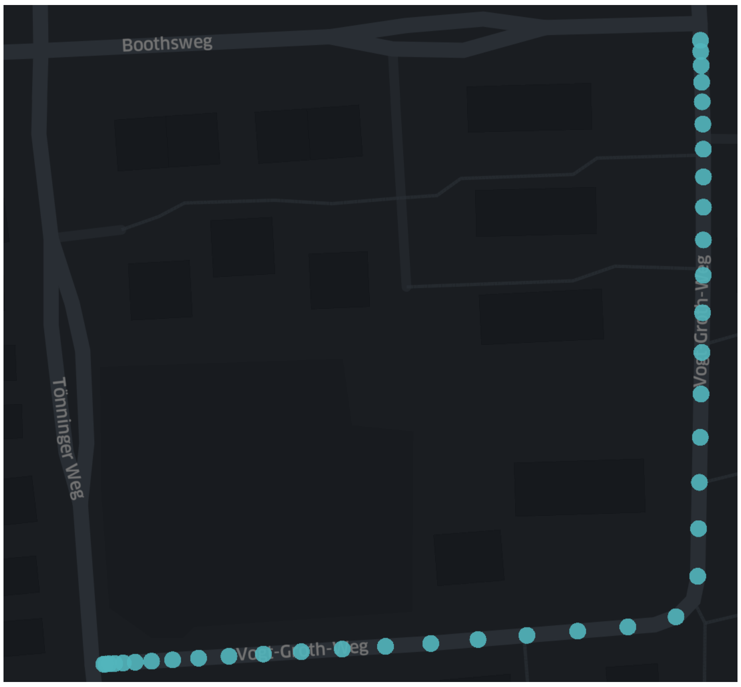


Abbildung 6.4: Car following the curvature of Vogt-Groth-Weg in Altona

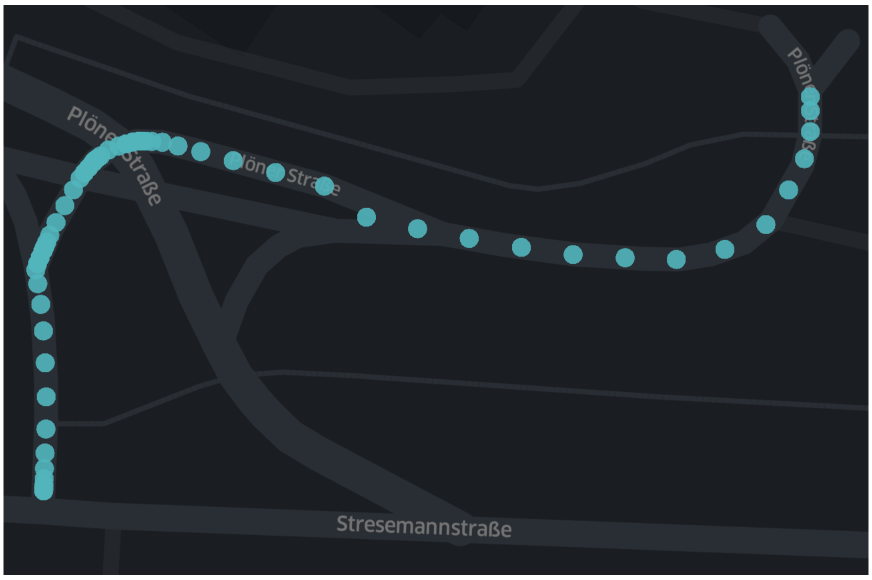


Abbildung 6.5: Car following the curvature of Plöner Straße in Altona

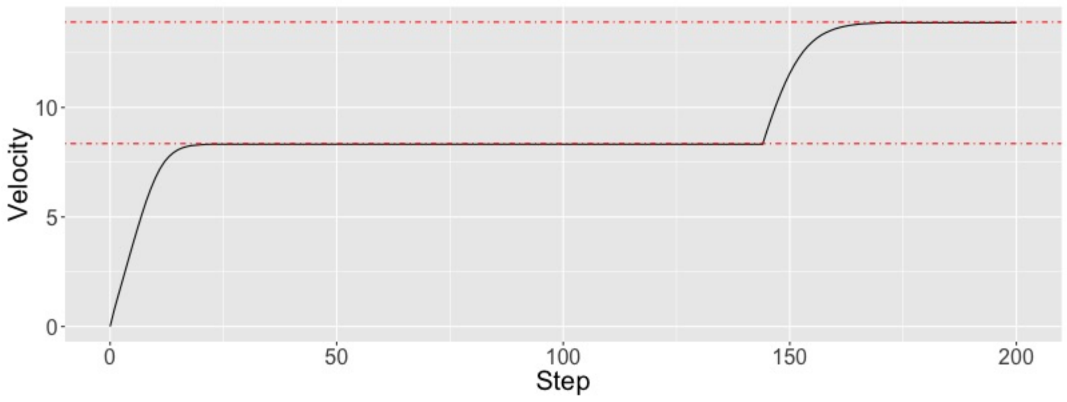


Abbildung 6.6: Car adopting its velocity to increasing speed limits.

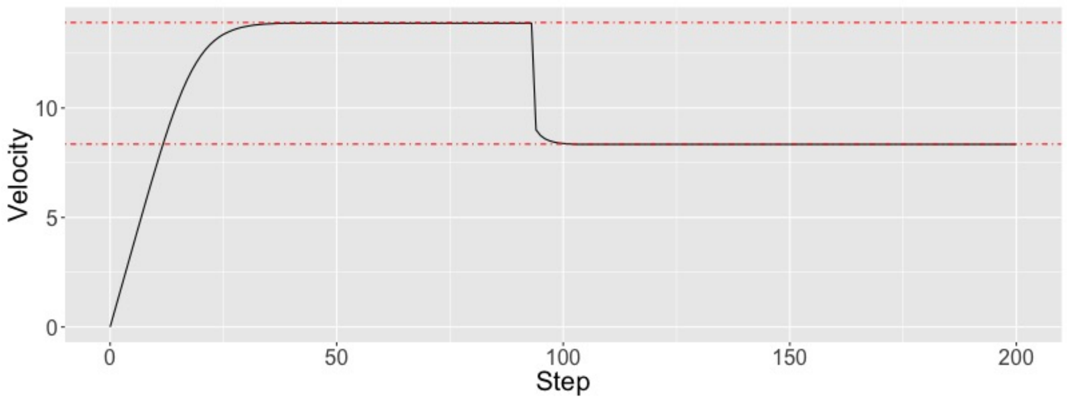


Abbildung 6.7: Car adopting its velocity to reduced speed limit.

Figure 6.6 shows a the velocity curve for the car from the increasing speed limit experiment. Initially the car accelerates to the currently allowed maximum speed of 8.33 m/s (30 km/h) as denoted by the lower dotted red line. At simulation step 144 the car crosses the intersection where the speed limit changes to 13.89 m/s (50 km/h). This is being picked up by the car (sensing) which then starts accelerating until the new speed limit, denoted by the upper red line, is reached. The limits are neither exceeded, nor undercut which ends the experiment successfully.

Experiment: ”Decreasing speed limits”

Figure 6.7 shows the results from the decelerating speed limit test. In the beginning the car accelerates to the maximum speed of 50 km/h as denoted by the upper dotted line. Once the intersection is crossed and the speed limit decreased, the car agent incorporates these new driving parameters and decreases its velocity. After decelerating to 30 km/h

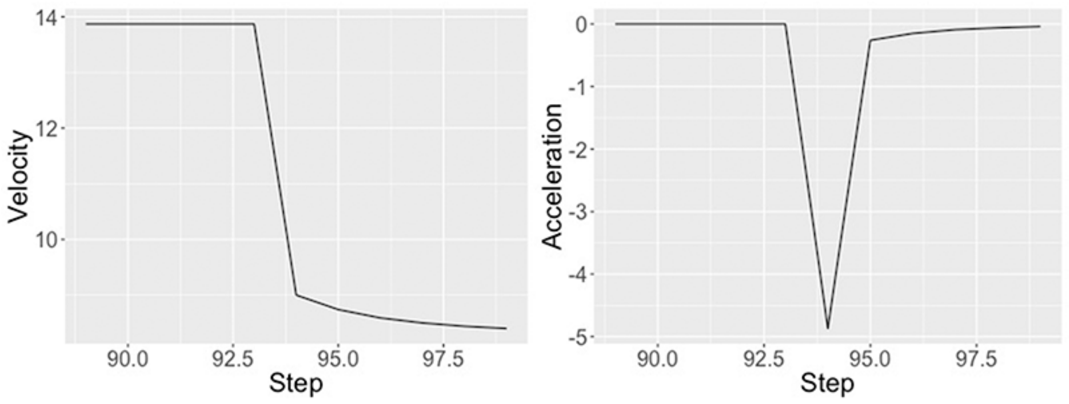


Abbildung 6.8: Harsh braking to comply with reduced speed limit

(lower dotted line) it continues driving at that speed. Obeying the changing speed limit works, however the amount of deceleration used to do so is very high. As with the previous experiments for deceleration, the car reduces its velocity by -4.87 $\frac{m}{s^2}$ in the first tick after sensing the new limits. Figure 6.8 shows an excerpt from the other figure which demonstrates this. After an initial very harsh deceleration, the car proceeds to reduce its speed with minimal brake application until the speed limit is met.

6.1.3 Traffic lights

Experiment: "Stopping at red lights":
The results of experiment 4.3.3 are shown in figure 6.9. On the left side the velocity is depicted in relationship to the ticks of the simulation. The figure on the right shows the acceleration for the same period. The agent starts with an initial velocity and acceleration of zero and then increases its speed following the depicted acceleration values until it reaches $10.8\frac{m}{s}$. At this point the car hasn't reached its maximum speed of $13.89\frac{m}{s}$ but since it is approaching a traffic light, it starts reducing the velocity. Tick 20 marks the start of the deceleration process where the acceleration changes from positive to negative values. The deceleration is at its maximum at tick 27 where it reaches $-1.25\frac{m}{s^2}$. At tick 39 the car has come to a complete stop with a remaining distance of $2m$ to the traffic light.

Side-note on the initial results which are shown in figure 6.10. Looking at the course of the velocity one can see a harsh drop in tick 23 where the agent suddenly decelerates before slowly coming to halt. This heavy braking is caused by a too low field of view.

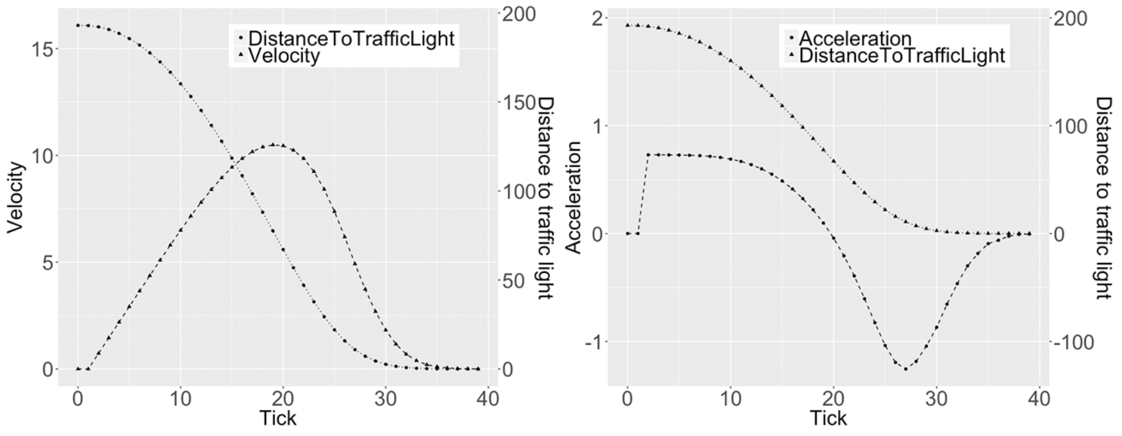


Abbildung 6.9: Velocity and the distance to the red traffic light in relationship to the current simulation tick on the left and acceleration/ distance to the traffic light in relation to the simulation step on the right. Velocity in $\frac{m}{s}$, distance in m, acceleration in $\frac{m}{s^2}$.

The agent 'sees' the traffic light very late and instantly performs an emergency breaking maneuver to avoid running the red light. Deceleration is as high as $-4.37\frac{m}{s^2}$ at this point. When encountering this behavior the necessity of a bigger field of view was eminent. See section 5.2.3 for details about this.

Experiment: "Green lights"

Figure 6.11 shows the agents velocity and acceleration over the course of simulating the green light scenario. As before the agent starts out with an initial speed and acceleration of zero. In contrast to the red light experiment, the car doesn't reduce its velocity while approaching the intersection. This time the traffic light is sensed and then green light is processed as a go signal. The car continues accelerating and increases its speed to the maximum velocity of $13.89\frac{m}{s}$. In tick 25 the car passed the light, not having slowed down in any way.

Experiment: "Yellow light - cross intersection"

The results for the first yellow light experiment look exactly like the ones for the green light experiment so figure 6.11 is reused. When the agent approaches the intersection, the traffic light turns yellow as the agent is 40 meters out. The car calculates that the intersection will be crossed within three seconds at the current velocity so there is no deceleration needed. As a result, the intersection is crossed without reducing the velocity

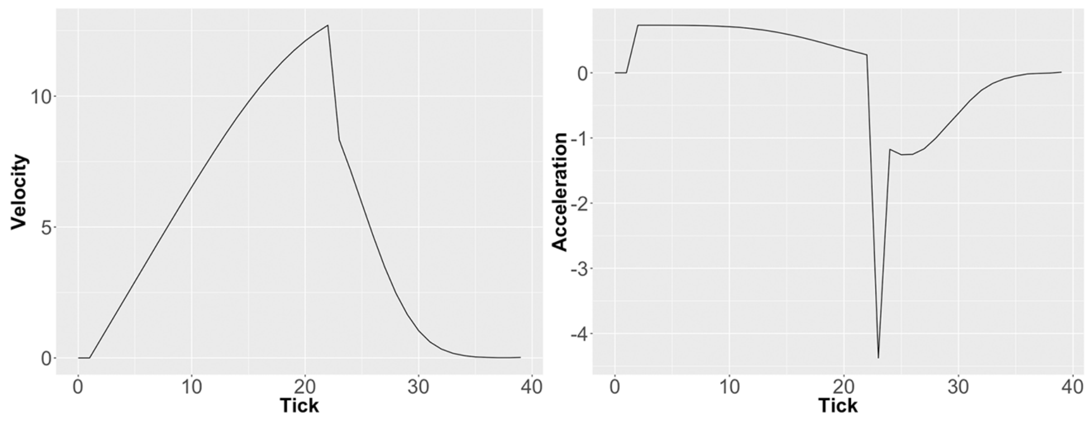


Abbildung 6.10: Velocity and the distance to the red traffic light in relationship to the current simulation tick on the left and acceleration/ distance to the traffic light in relation to the simulation step on the right. Velocity in $\frac{m}{s}$, distance in m, acceleration in $\frac{m}{s^2}$.

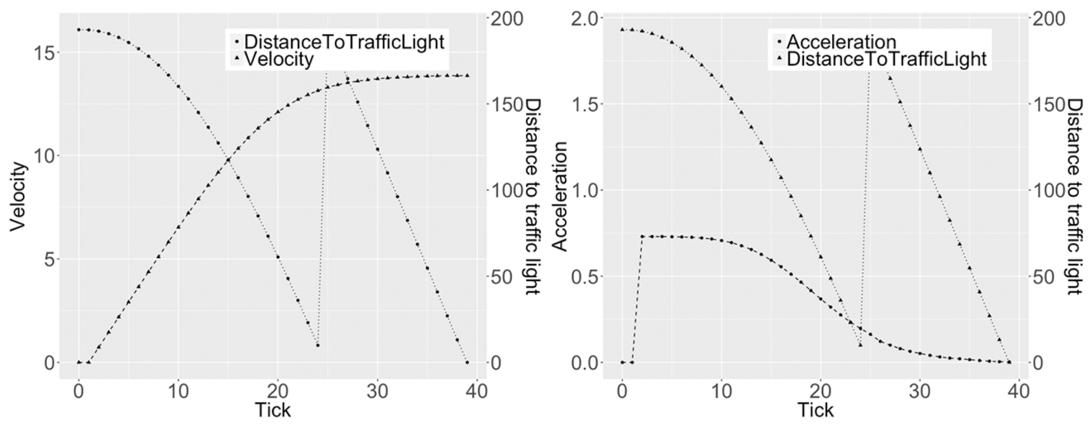


Abbildung 6.11: Velocity and the distance to the green/ yellow traffic light in relationship to the current simulation tick on the left and acceleration/ distance to the traffic light in relation to the simulation step on the right. Velocity in $\frac{m}{s}$, distance in m, acceleration in $\frac{m}{s^2}$.

and the car keeps accelerating as the maximum speed of 13.89 m/s has not been reached yet.

Experiment: ”Yellow light - stop”

Figure ? shows the the cars velocity and acceleration/ deceleration during the approach. In simulation step 23 the traffic light turns from green to yellow as the car is 42 meters out. The car senses this change and after calculating that there is enough space/ time to come to a stop, starts to decelerate. This deceleration is as high as -3.38 $\frac{m}{s^2}$ which again shows the IDM’s tendency to calculate high values for situations with sudden changes. In simulation step 39 the car comes to a complete stop, two meters from the intersection just as expected.

6.1.4 Intersections

Experiment: ”Approaching alone”

The car approaches the intersection with an initial speed of 13.89 m/s. As the intersection is sensed the car reduces its velocity incrementally to 2.7 m/s so that it could stop at any time. Since the exploration shows no incoming cars, the intersection is crossed and the car starts accelerating again. Analyzing the performed simulation showed that the agent behaves correctly.

Experiment: ”Traffic from the right”

For the second intersection simulation, car one comes from the south and is headed north while the second car comes from the east and is traveling east. Both cars reduced their velocity to 2.7 m/s as they approach the intersection where they checked the incoming traffic. As car two doesn’t have to give way it proceeds and crosses the intersection just as in the last experiment. The other car senses traffic from the right and further reduces its velocity to let that car pass. In the next simulation step after car two has passed, car one starts accelerating again and crosses the intersection as well. Again, both agents behave in the desired way.

Experiment: ”Turning left with approaching traffic”

The third simulation is a slight variation of experiment two but this time the first car comes from the south and wants to go east while the second car comes from the north and wants to cross southbound. Again both cars reduce their speed to 2.7 m/s during the approach before the second car crosses the intersection. The first car has successfully sensed that his left turn crosses the path of car two and that it therefore has to wait until it passed. Once this has happened, the first car crossed the intersection as well which concludes the experiment as success.

Experiment: ”Deadlock turning left with approaching traffic”

In the fourth simulated scenario the first car comes from the south, heading west while the second car comes from the north headed east. In the simulation both cars come to a complete stop at the intersection as they both want to make a left turn. As the concept of turning lights is not part of the model, the cars can’t figure out that they both want to turn left which would work perfectly fine. Both cars wait for a complete second as no one moves before they start driving again as desired.

Experiment: ”Complex deadlock situation with four directions”

The last simulation includes cars approaching from four directions. Car one comes from the south and wants to go east while car two comes from the east and wants to go west. Car three comes from the north and wants to go south while car four comes from the west and wants to go north. During the first part of the simulation, all cars approach, sense the other incoming cars and come to a stop as no one has the right of way. This is the deadlock situation. To solve this issue, a mechanism has been included in the cars intersection routine that checks for lowest OSM ID. The car on that road then proceeds, thereby resolving the deadlock. In the simulation this was car three that crosses the intersection which leads to car two having no traffic from the right which allows it to cross as well. Now that car one has no cars to its right it makes its turn before finally car number four can cross the intersection. This concludes the intersection experiment successfully.

6.1.5 Multilane roads

The experiment has been realized as simulation test where the two car agents have been used in drive mode number six as described in the implementation chapter. Both cars start at the same intersection driving north. The route of car one dictates to go right on the next intersection while second car is tasked to go to the left. During the first simulation step, both cars change to the correct lanes so that car one drives on the rightmost lane while car number two uses the leftmost one. In the last step before making the turn, this is confirmed again so that the experiment is completed successfully.

6.2 Hypotheses Two

Simulating Altona's virtual population for the time period of 6am to 7pm took 12 hours on average. The simulation was repeated 10 times. On this scale and with the used agent count this is faster than real time so that the model could potentially be used for predictions into the future. Based on the day-plan generator the agents made their trips throughout the city while the movements were captured by the traffic counts. The results from comparing the simulation counts to the traffic counts from Altona show that the measurements from the simulations are much lower than the real-life counts. Over all traffic counts and all 15 minute time frames the simulated counts make up only 15.75% of the actual counts. The counting point which gets closest to the actual counts is at the intersection of Elbchaussee and Schenefelder Landstraße in the northern direction. The counts make up 29.76% of the actual traffic measured there. At the intersection of Stresemannstraße and and Tasköprüsstraße the eastbound traffic is only 2.6% of the actual counts. This is the biggest deviation seen in the ten counting points that have been monitored.

To further discuss the results, three traffic count positions have been selected whose results represent both extremes and the closest resemblence. The counting position on Elbchaussee and Schenefelder Landstraße is the closest match in terms of amount of agents counted with 29.76% of the actual counts. Figure 6.13 shows these results over the course of the day with the blue bars representing the real traffic counts while the orange bars show the simulated ones. In the morning hours the day-plan generator caused the most traffic around 8 o'clock with another, smaller peak at 9:30h. After this, the amount of traffic stays stable with another slight increase around noon. Towards the end of the day, the amount of cars counted increases slightly without reaching the morning peak. This is conclusive with the amount of trips produced through the day-plan generator. The peaks in the morning are caused by agents traveling from the west of Altona with more residential areas to the northwest. As the counting position is located on a major road that connects both parts, the amount of traffic during the day stays constant.

The intersection Stresemannstraße/ Tasköprüstraße showed the least traffic of all monitored intersections. Also the results deviate strongly from the actual counts as shown in

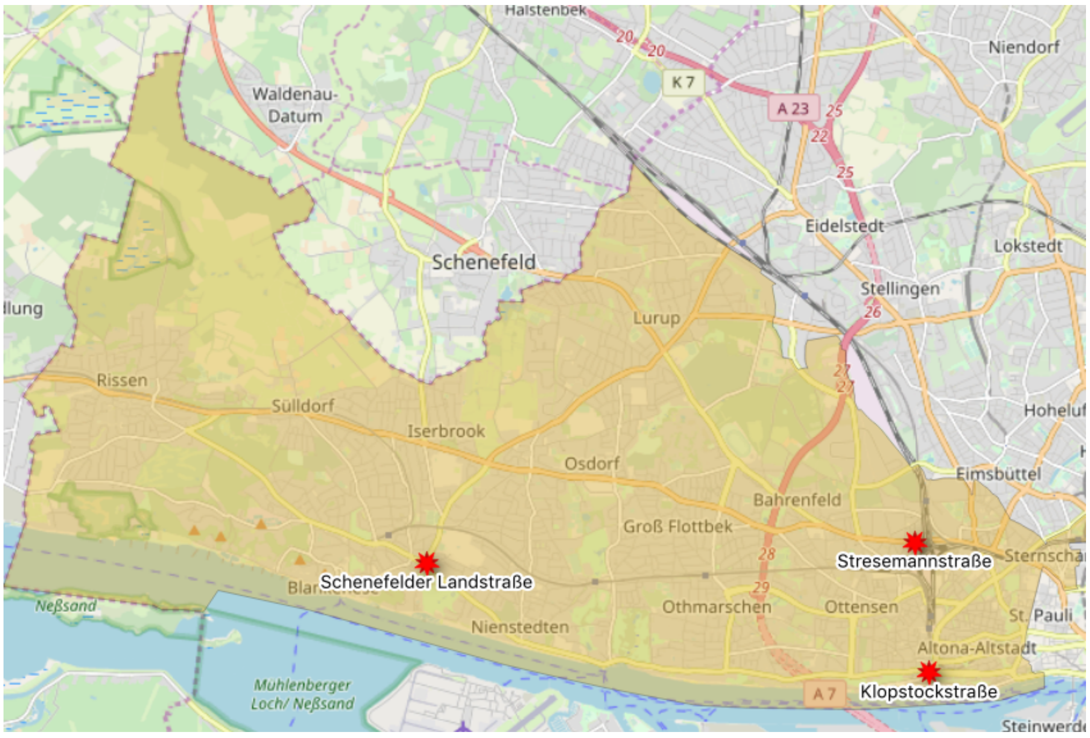


Abbildung 6.12: Locations of the three counting points discussed in detail

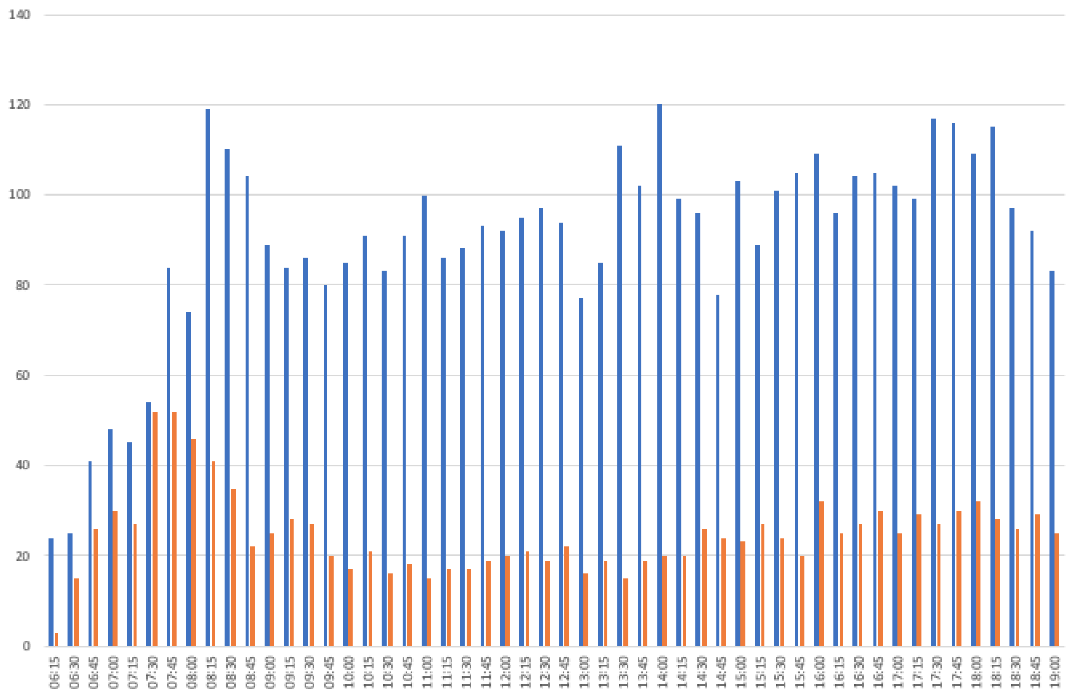


Abbildung 6.13: Schenefelder Landstraße northbound traffic from Elbchaussee

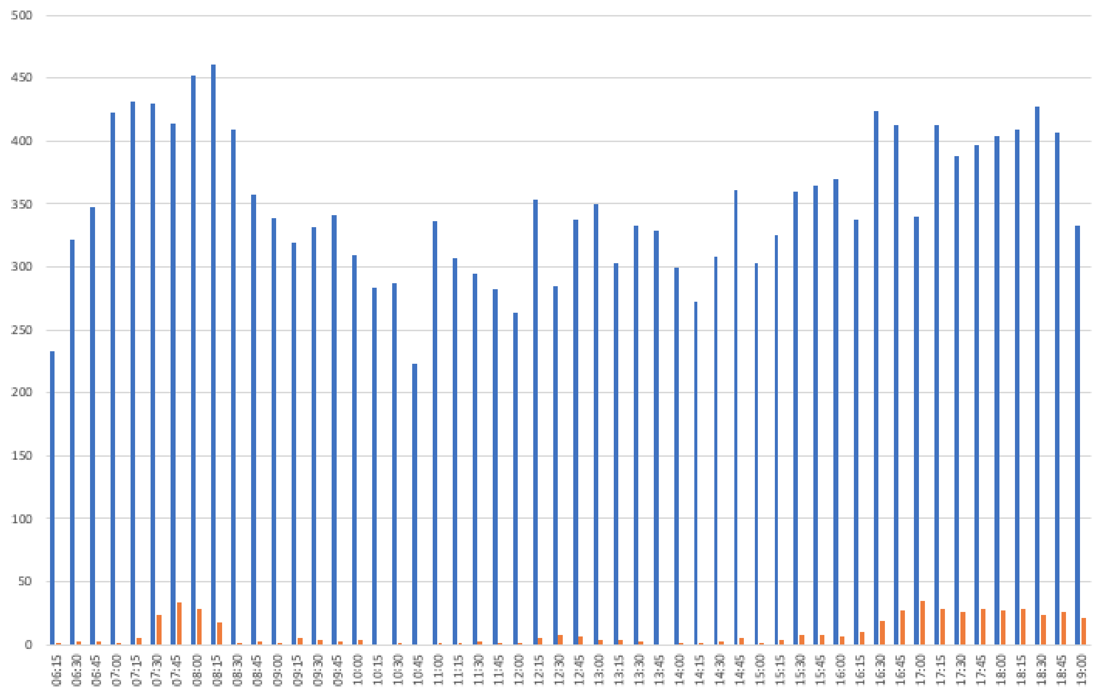


Abbildung 6.14: Stresemannstraße eastbound intersection Tasköprüstraße

figure 6.14 with the blue bars being the actual counts and the orange bars representing the simulated results. Traffic flowing on the road only accounted for 2% of the actual counts in Altona. The simulated traffic (orange bars) is barely visible at the bottom where the counts drop to 0 cars per 15 minute time window on multiple occasions. In the morning and in the afternoon the simulated traffic goes up during rush hours but even then it only accounts for up to 10% of the actual counts. These results are difficult to explain since this is one of the most important roads in Altona with a lot of traffic flowing in both directions. Even though the peaks in the morning and afternoon are resembled, the overall amount of counted cars is suspiciously low. The combination of virtual population, used traffic model and the environment data should be further examined before applying it to real world predicitions.

At the third count on the intersection of Klopstockstraße and Kaistraße, the pattern found in the real traffic is matched closest even though the simulated traffic only accounts for 15.35% of the real counts. Figure 6.15 shows the measured traffic in orange while the blue bars represent the actual counts. Over the course of the simulation, the measured traffic copies the actual counts in their characteristic even though accounting for fewer agents. The peaks in the morning around 8 o'clock as well as those towards the end

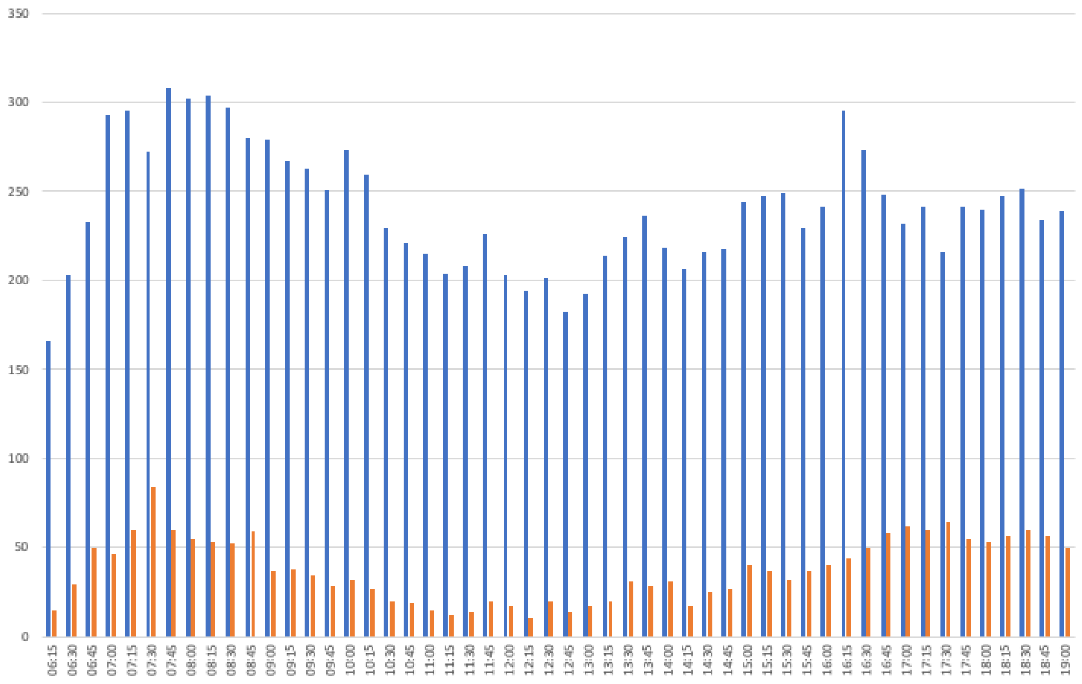


Abbildung 6.15: Klopstockstraße/ Kaistraße in south-west direction

of the day resemble the actual counts very closely. In the morning from 6 o'clock on, the simulation produces the same incline and in the evening towards the end of day the traffic stays constant as in the actual counts. Even the slight increase 13:45 has been reproduced. This does not mean that the simulation is perfectly adjusted or is generally able to reproduce the exact traffic but it sure demonstrates that it is possible to recreate the main dynamics only using publicly available data.

Overall the traffic measured is significantly lower than the actual counts. The closest resemblance can only account for up to 29.76% while the lowest measurements only makes up 2.6%. That the amount of traffic is generally lower than the actual counts can be partially explained by the lack of commuters in the simulation. The day-plan-generator can only generate trips for the people living in the simulated area and even then, only those starting and ending in Altona are simulated. Commuters that leave Altona or those coming from outside aren't part of the simulation so these trips will not occur here. This leads to the overall traffic being significantly lower than the measured volume which is being collected through counts. Secondly the model used and the environment it operates in could be the reason too. The traffic lights are operating on a very simplified schedule since there is no data available on the actual switching times. This might change in the future and the traffic light layer can already handle it but at the moment of finishing this thesis it was not available. As a result, the overall flow of traffic is slowed down throughout the city since the simplified lights give way to only one road at a time instead of allowing straight traffic in opposing directions crossing the intersection at the simultaneously. Also there are no priority roads which receive longer green phases than others as they would in reality. Another problem at intersections is missing information about the lane directions which is missing completely. For any road, only the total amount of lanes is known, but not where they are leading. Whether the traffic is allowed to turn right from three lanes or only from the rightmost lane wasn't contained in the available road data so it had to be simplified too. This was solved in the way that cars can go right only from the rightmost lane and go straight from every lane. For situations where more than one lane of a bigger road leads to the left (or right), the model cannot account for it and naturally produces jams as the amount of cars going in that direction exceeds the lanes capacity. As a result, the overall flow of traffic is slower.

6.3 Hypotheses Three

Experiment: "Left hand driving in South Africa"

The experiment has been tested using a simulation to determine whether the left hand driving parameter gets set correctly. After reading the agent initialization file the values are passed through the agent manager and the new agent is created with the left hand driving enabled which concludes this experiment successfully.

Experiment: "Implementing a new intersection type"

The design chapter describes how the two traffic codes have been incorporated into the model. Based on the input parameter *trafficCode* the desired traffic code is selected. To test this, another simulation run has been made whose results of the four way intersection are described in the next experiment. Adding this new type of intersection only required a minimal change to the model at the point where the currently used traffic code is checked. Depending on this parameter a dedicated function for the traffic code is called where the new intersection has been implemented. Only requiring to change two lines of code proved that the model was implemented in a generic manner so that new types of intersections or a new traffic code can be implemented easily.

Experiment: "Four way stop"

Four cars arrive from four different directions. Car one from the south, car two from the west, car three from the north and car four from the west. The order of arrival is car one, then car two, car three and then car four. During the experiment all cars sensed the intersection and the incoming cars and came to a complete stop. After this, the cars proceeded in the correct order which concludes this experiment successfully.

Experiment: ”Simulating the whole park”

Figure 6.16 shows a simulation of the whole Kruger National Park. Red dots denote the park’s camps while the green dots represent the park’s gates. To better visualize the results, only the tracks of 30 agents have been plotted as they drive around the park. The orange lines represent the cars paths driven over a time period of 30 minutes. Simulating the agents in the park has been deemed correct based on the authors knowledge about the park.

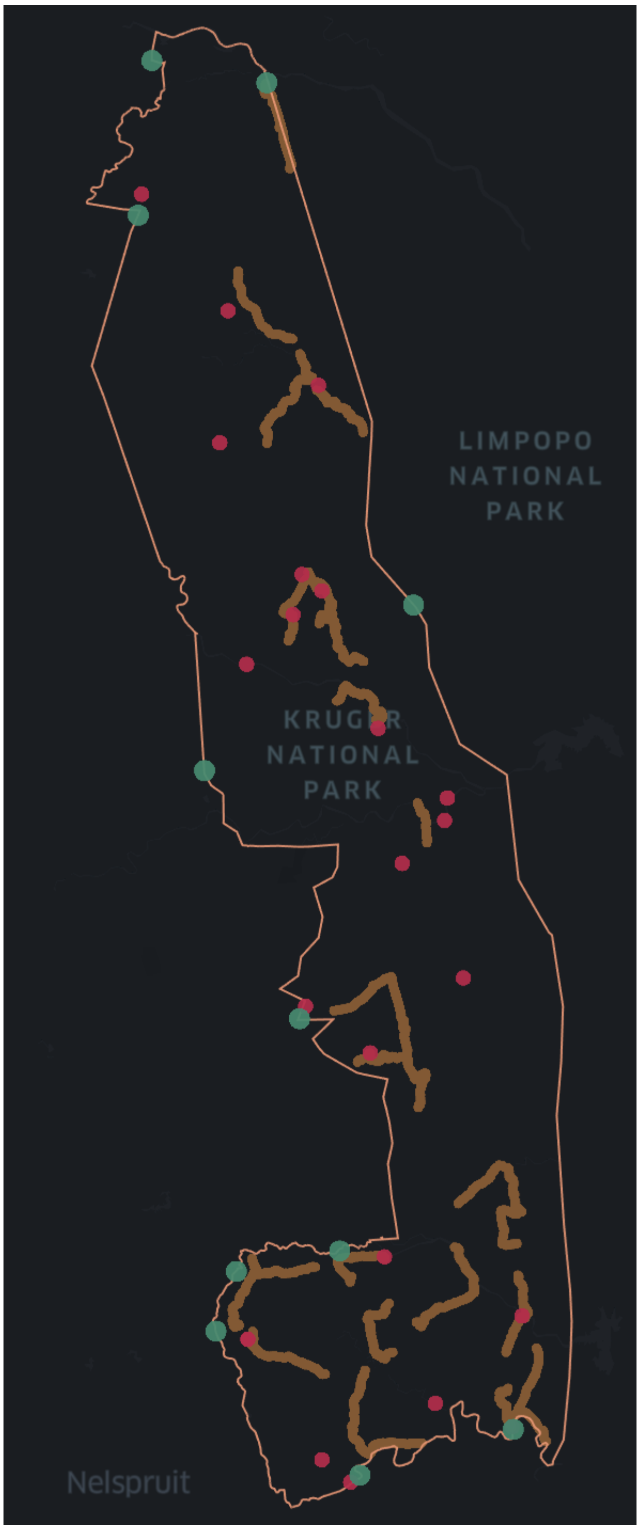


Abbildung 6.16: Simulating the whole Kruger National Park

7 Discussion

The discussion has been divided into multiple parts as there are three separate hypothesis to evaluate: At first the analysis and implementation is concluded in order to establish a baseline of what has been implemented and what hasn't. Once these details have been laid out, the experiment results for hypothesizes one to three are examined. At last, a conclusion is being drawn that summarizes the thesis.

7.1 Hypothesis One

All experiments have been carried out and their results shown in the last chapter. Now the different groups of experiments are discussed with regard to the first hypothesis.

7.1.1 Basic driving

The *Intelligent Driver Model* has been successfully integrated into the agent model and is the basis for all calculations concerning changes in speed. After sensing their environment the agents reason about the information and decide where they go. Based on this decision they use the IDM module to calculate the change in velocity as input for possible move actions. Accelerating as well as decelerating has been covered and plotted with the results for deceleration showing problems as the calculated values are very high. Results from the acceleration experiments show that the *Intelligent Driver Model* produces realistic values for acceleration maneuvers and can be applied without restrictions.

This is not the case for the deceleration experiments though. They show that the IDM works as long as there is enough space/ time for the car to reduce its velocity. If the remaining distance to an obstacle is too small or the speed difference is too high, the IDM calculates unrealistic values for deceleration. When the car agents move within a controlled environment where they don't have to react to sudden changes, the IDM will not be a problem. For scenarios where unexpected things might occur, the IDM will not be sufficient to calculate realistic deceleration behavior. If, in an integrated scenario with pedestrians and cars, there was one human to step in front of the car, the car would stop immediately instead of reacting appropriately. Since the IDM has been build as collision-free model, this mechanism is embedded by design and the car will brake as hard as it has to, in order to avoid the crash.

7.1.2 Speed limits

The speed limit experiments show that the agents can identify changing speed limits by themself and have appropriate rules in place to react to them. Correct reactions to both increasing and decreasing speed limits were proven with the IDM causing to high deceleration values when entering a zone with reduced maximum speed. As a result the current velocity is reduced heavily within the first second instead of slowly decreasing it as human drivers would do. Still, the cars act by either reducing the current speed to a lower speed limits or by accelerating until a higher speed limit is met. It was shown that the agents neither exceed the limits, nor undercut them by more than 1 km/h.

7.1.3 Traffic lights

Traffic lights have been included through the dedicated traffic light layer which manages their light changes. Linking the traffic lights directly into the spatial graph environment allows the agents to access the current situation during the explore phase. The experiments for traffic lights show that the agents pick up the current light situations while approaching the intersection. Based on that information, the agents directly incorporate the presences/ absence of traffic lights and their current status into their planning and react accordingly. The traffic light experiments prove that the agents correctly recognize the three phases red, green and yellow and react accordingly. Red lights are strictly being followed and green lights seen as a go signal. In situations with yellow the cars decide on the correct action based on the current velocity and the distance to the traffic light.

7.1.4 Intersections

The experiments at intersections show that the agents know to handle the various situations found at crossings. When traffic lights are absent, the right of way is determined based on the driving direction and the angle of incoming roads and cars. Situations where cars come from the right can be handled correctly as well as the more complex scenarios where cars come from multiple sides. The only situation that can't be handled ideally is when two cars are turning left with incoming traffic since there is no concept like turning lights. As a result the cars don't know that the other one desires to turn left too. This has been solved by neglecting the rule if no one moves for multiple steps which can be improved in the future. Besides this, even complex deadlock situations can be resolved and the agents behave as desired at left yields to right intersections.

7.1.5 Multi-lane roads

Driving on the right lane when turning right and driving left when turning left has successfully been integrated into the model. The experiments for multi-lane roads demonstrate how the agents move to the correct lane before making the turn. There is no moving to the right from any lane but the rightmost and left turns are only performed from the leftmost lane.

7.1.6 Summary

In conclusion it can be said that hypothesis one holds and that the *Intelligent Driver Model* can be used as foundation for an agent-based traffic model. It can calculate realistic acceleration and deceleration values for scenarios without abrupt changes but has its downsides when it comes to deceleration. In situations where human beings would brake hard and consistent for multiple seconds, the IDM calculates too high deceleration values in order to reduce the speed as fast as possible. Since the IDM has been build collision free, this behavior is by design part of the model and has to be taken into account. Implementing the agent rules on top has been proven possible with the various experiments showing it.

7.2 Hypothesis Two

Recreating traffic patterns found in real traffic is not an easy task as described in the analysis chapter. Even taking into account a multitude of data is not always sufficient which has been proven on other occasions [28]. That the traffic patterns of Hamburg

Altona could only be reproduced to a certain degree is unfortunate but not unexpected. In microscopic models the resulting traffic is based on the trips made of a virtual population. If the population resembles the real situation this could lead to the same measurements at best. Since statistics are only able to depict the reality to a certain level, there is uncertainty involved per definition. But then, the social science aspect wasn't the focus of this thesis. The second variable that has to be taken into account is the agent model itself as well as the data it is operating on. As described in the Design and Implementation chapter there a multiple factors that lead to the seen results. The result section for hypothesis two lists these reason which mainly account to slowing down the overall flow of traffic through missing data on the simulated area and the resulting simplifications made to the model.

As a result the traffic patterns in the simulation can differ strongly from the real system especially if road markings or traffic light schedules apply that contradict the simplification. For small scenarios with a limited amount of intersections this could be corrected manually and is often done so with frameworks like PTV Vissim but for the simulated scenarios in Altona or the Kruger National Park this approach is not feasible anymore. Since large scale simulations require a lot information, they can only be as good as the provided data. Despite the differences in the results, the second hypothesis mustn't be seen as complete failure but rather as first step in the right direction that can be improved upon in the future.

7.3 Hypothesis three

Adapting the model to the South African environment worked with few modifications to the base implementation. The car which has been build for Altona includes a wide variety of drive modes which proved as good starting point for the Kruger model. In the near future the model will be enriched with data about cars traveling through the park so there will be a similar virtual population. At the time of writing the thesis this data wasn't available so drive mode two was used where agents pick random starts and destinations. With the origin-destination covered, the focus was put on implementing the remaining changes which mostly covered the intersection behavior.

Left hand vs right hand driving could be solved through adding an input parameter so that the model can be configured with a config file. This eliminates the need to change code or to recompile the model when running a simulation in another part of the world.

For the required traffic code the same approach was chosen so that there is another parameter (*trafficCode*) to be set from a config file. Using this parameter one can switch between German and South African traffic code as these are the only ones implemented at this time. For future scenarios these fields can be extended so that other areas in the world can be added easily. Since the traffic codes have been separated from the main agent method, new methods for other traffic codes can be added without the need to reprogram the main part of the agent. This concludes hypothesis three and it can be said that the implemented model is generic.

7.4 Final conclusion

Starting out with a rudimentary car agent and the spatial graph environment, a lot of new things had to be developed. The progress of the car model stopped on multiple occasions as new concepts had to be added to either LIFE or the encompassing models for Altona and KNP. Traffic lights were very important for the models but implementing a dedicated *Traffic Light Layer* slowed down the progress. This proved as challenge but also allowed for more realism in the simulation and can hopefully be extended should detailed data on traffic lights become available. The same is true for the *Car Spawner Layer* and the *virtual population* which had to be invented so that new cars could be inserted into the simulation during runtime. Before the agents could drive in Altona, the data on traffic counts had to be obtained as well and a dedicated counting mechanism be developed in order to make the simulation comparable. Each addition to the car model meant going back one step to extend something else or invent it from scratch. As a transition to the future work chapter I must say that building the model was challenging, yet fun but it also showed how much effort it takes to develop a traffic simulation.

The implemented model works and the agents behave correct in the simulated scenarios both in Altona in the Kruger National Park. Hypothesis one can be confirmed based on the seen results with a few remarks to the *Intelligent Driver Model.* For free flowing traffic without out sudden changes this model can be used as basis for traffic simulations without limitations. Should the situation require braking maneuvers that exceed the comfortable braking limit, the model calculates unrealistically high values for deceleration. This poses a severe problem and other models must be considered to either replace the IDM completely or at least in such situations. Even though the data on Altona was limited, the model could reproduce the general dynamics of urban traffic to a certain level. In total, the counts were significantly lower but there are multiple reasons leading to this mainly being the absence of commuters and simplifications in the model. For future simulations, the shortcomings should be examined in detail and more focus been laid on the demographics behind the virtual population. Altona connects the west of Hamburg with the city centre and has a lot of passing traffic, not originating in Altona. This isn't part of the simulations yet and should certainly be included in the future. Still, hypothesis two can be seen as partial success as the overall dynamics where recreated. With more detailed information about the origins and destinations of trips in the future, the model will certainly be able to get closer to the actual traffic counts.

Lastly hypothesis three can be confirmed as well. The agent model, build for Hamburg, could be used in the Kruger National Park with only a few extensions. Once the traffic code had been implemented, the model was usable in the new setting without the need for further code changes. The MARS specific simulation configuration files now allow to populate the agent constructors during simulation initialization so that both the traffic code as well as the parameter for driving on the left side of the road can be configured from outside. The model has been build truly generic so that components to simulate the traffic of other parts of the world can be added gradually as the scenarios change in the future.

8 Future Work

For future simulations there are a couple of topics that can be improved upon. First there is the data aspect which will probably have the biggest impact because most of the simplifications made to the current model where du to the lack of it. Detailed data on road markings, traffic light schedules and traffic signs would allow to increase the degree of realism manifold. Based on such data, a dedicated intersection component could be implemented that represents the crossings actual spatial extend with all its turning lanes, dictating the direction of traffic. This would improve greatly on the current situation where agents can cross intersections in any direction they want. With data on traffic lights, higher order streets would receive longer green phases so that more traffic can flow in certain directions. Concepts like these are especially needed if the traffic of Altona should be recreated more detailed. Once a dedicated intersection module has been added to the simulation, all the various intersection scenarios could be recreated based on traffic signs. This data isn't available at the moment so every intersection has either a traffic light or the right of way is determined based on left yields to right/ four way stop rules.

The second improvements can be made to the agent model itself which has been developed to simulate cars. This implementation will be the basis for busses in public transport but there are even other types of road users like trucks possible. Since the model has been implemented as generic as possible only the driving parameters for acceleration, deceleration, length etc. would have to be altered. In terms of driving physics, the cars curvature should be part of the movement calculation as well so that cars don't speed in places where it isn't appropriate. Additionally the cars/ trucks mass could be incorporated into the movement calculation for even more realism. The IDM showed a couple of disadvantages for the simulation so this model could be replaced with another model for scenarios where intense braking has to be applied. If these challenges aren't enough, one could even set out to include human emotions to incorporate different driving styles which would probably enable the highest level of realism.

Literaturverzeichnis

[1] Statistisches Jahrbuch Hamburg 2017/2018. Hamburg, 2018. – Forschungsbericht

[2] Bazzan, Ana L. ; Klügl, Franziska: A review on agent-based technology for traffic and transportation. In: *Knowledge Engineering Review* (2013). – ISBN 1469-8005

[3] BMVI: Mobilitaet in Deutschland 2017 - Ergebnisbericht / Bundesministerium für Verkehr und digitale Infrastruktur. Bonn, 2018. – Forschungsbericht

[4] Brackstone, Mark ; McDonald, Mike: Car-following: a historical review. In: *Transportation Reserach Part F: Traffic Psychology and Behaviour* 2 (1999), Nr. 4, S. 181–242

[5] Dallmeyer, Jörg: *Simulation des Straßenverkehrs in der Großstadt.* Springer, 2014. – ISBN 978-3-658-05206-5

[6] Foytik, Peter ; Jordan, Craig ; Robinson, R M.: Exploring Simulation Based Dynamic Traffic Assignment With A Large-Scale Microscopic Traffic Simulation Model. In: *ANSS '17 Proceedings of the 50th Annual Simulation Symposium.* Virginia Beach, USA, 2017

[7] Greenshields, B D.: A Study Of Traffic Capacity. In: *Proceedings of the Fourteenth Annual Meeting of the Highway Research Board Held at Washington, D.C. December 6-7, 1934. Part I*, Highway Research Board, 1934

[8] Grignard, Arnaud ; Alonso, Luis ; Taillandier, Patrick ; Gaudou, Benoit ; Nguyen-Huu, Tri ; Gruel, Wolfgang ; Larson, Kent: The Impact of New Mobility Modes on a City: A Generic Approach Using ABM. In: Morales, Alfredo J. (Hrsg.) ; Gershenson, Carlos (Hrsg.) ; Braha, Dan (Hrsg.) ; Minai, Ali A. (Hrsg.) ; Bar-Yam, Yaneer (Hrsg.): *Unifying Themes in Complex Systems IX.* Cham : Springer International Publishing, 2018, S. 272–280. – ISBN 978-3-319-96661-8

[9] Horni, A ; Nagel, K ; Axhausen, Kay W.: *The Multi-Agent Transport Simulation MATSim*. London : Ubiquity Press, 2016

[10] Hüning, Christian ; Adebahr, Mitja ; Thiel-Clemen, Thomas ; Dalski, Jan ; Lenfers, Ulfia ; Grundmann, Lukas ; Dybulla, Janus ; Kiker, Gregory: Modeling & Simulation as a Service with the Massive Multi-Agent System MARS. In: *Spring Simulation Multiconference*, 2016. – ISSN 07359276

[11] Kesting, Arne ; Treiber, Martin ; Helbing, Dirk: Agents for Traffic Simulation. In: Uhrmacher, Adeline M. (Hrsg.) ; Weyns, Danny (Hrsg.): *Multi-Agent Systems: Simulation and Applications*. CRC Press, 2009, Kap. 11, S. 566. – URL http://arxiv.org/abs/0805.0300. – ISBN 978-1-4200-7023-1

[12] Li, Li ; Chen, Xiqun (.: Vehicle headway modeling and its inferences in macroscopic/microscopic traffic flow theory: A survey. In: *Transportation Research Part C: Emerging Technologies* 76 (2017), mar, S. 170–188. – URL https://linkinghub.elsevier.com/retrieve/pii/S0968090X17300141. – ISSN 0968090X

[13] Lighthill, M. J. ; Whitham, G. B.: On Kinematic Waves. II. A Theory of Traffic Flow on Long Crowded Roads. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 229 (1955), Nr. 1178, S. 317–345. – URL http://rspa.royalsocietypublishing.org/cgi/doi/10.1098/rspa.1955.0089. – ISBN 00804630

[14] Löffler, Andreas: *Verfeinerung und Parametrisierung eines Fußgängermodells durch Anreicherung mit geospatialen und demographischen Daten*, Hamburg Universtity of Applied Sciences, Hauptprojekt, 2019. – 49 S

[15] Lopez, Pablo A. ; Behrisch, Michael ; Bieker-Walz, Laura ; Erdmann, Jakob ; Flötteröd, Yun-Pang ; Hilbrich, Robert ; Lücken, Leonhard ; Rummel, Johannes ; Wagner, Peter ; Wiessner, Evamarie: Microscopic Traffic Simulation using SUMO. In: *2005 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, nov 2018. – URL https://elib.dlr.de/124092/

[16] Nagel, Kai ; Schreckenberg, Michael: A cellular automaton model for freeway traffic. In: *EDP Sciences* 2 (1992), Nr. 12, S. 2221–2229

[17] Newell, G. F.: Nonlinear Effects in the Dynamics of Car Following. In: *Operations Research* 9 (1961), Nr. 2, S. 209–229. – URL http://pubsonline.informs.org/doi/abs/10.1287/opre.9.2.209. – ISBN 0030-364X 1526-5463

[18] Sanparks: Kruger National Park - Management Plan / South African National Parks. 2018. – Forschungsbericht. – 243 S

[19] Schadschneider, Andreas: *Physik des Straßenverkehrs.* 2004

[20] Smith, Laron ; Beckman, Richard ; Baggerly, Keith ; Anson, Doug ; Williams, Michael: TRANSIMS: Transportation Analysis and Simulation System / Los Alamos National Laboratory. 1995. – Forschungsbericht

[21] Sokolowski, John A. ; Banks, Catherine M.: *Principles of Modeling and Simulation - A Multidisciplinary Approach.* Wiley, 2009. – 280 S. – ISBN 978-0-470-28943-3

[22] Tchappi Haman, Igor ; Kamla, Vivient C. ; Galland, Stéphane ; Kamgang, Jean C.: Towards an Multilevel Agent-based Model for Traffic Simulation. In: *Procedia Computer Science* 109 (2017), Nr. 2016, S. 887–892. – URL http://dx.doi.org/10.1016/j.procs.2017.05.416. – ISSN 18770509

[23] Treiber, Martin ; Hennecke, Ansgar ; Helbing, Dirk: Congested Traffic States in Empirical Observations and Microscopic Simulations. In: *Physical Review E 62* (2000)

[24] Treiber, Martin ; Kesting, Arne: An Open-Source Microscopic Traffic Simulator. 2 (2010), Nr. 3, S. 1–7. – URL http://arxiv.org/abs/1012.4913{%}0Ahttp://dx.doi.org/10.1109/MITS.2010.939208

[25] Treiber, Martin ; Kesting, Arne: Car-Following Models Based on Driving Strategies. In: *Traffic Flow Dynamics.* URL http://dx.doi.org/10.1007/978-3-642-32460-4{_}11, 2013, S. 181–204. – ISBN 978-3-642-32459-8

[26] Weyl, Julius ; Glake, Daniel ; Clemen, Thomas: Agent-based Traffic Simulation at City Scale with MARS. In: *2018 Spring Simulation Multiconference*, 2018

[27] Zheng, Y U. ; Capra, Licia ; Wolfson, Ouri ; Yang, H A I.: Urban Computing : Concepts , Methodologies , and Applications. In: *ACM Transactions on Intelligent Systems and Technology* (2014)

[28] Ziemke, Dominik ; Nagel, Kai: Development of a fully synthetic and open scenario for agent-based transport simulations–The MATSim Open Berlin Scenario / VSP Working Paper 17-12, TU Berlin, Transport Systems Planning and Transport Telematics. 2017. – Forschungsbericht

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen.“

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: ______________________________

Vorname: ______________________________

dass ich die vorliegende Masterarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Developing a generic multi-agent car model to simulate road traffic with MARS

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original