Master of Science Thesis

Prediction and Simulation Models for Intelligent Vehicle Highway Systems

Assessment, Development, and Calibration

B.J.A de Graaf

March 24, 2009

DCSC

TU Delft

Delft Center for Systems and Control

Delft University of Technology

Prediction and Simulation Models for Intelligent Vehicle Highway Systems

Assessment, Development, and Calibration

Master of Science Thesis

For obtaining the degree of Master of Science in Systems and Control
at Delft University of Technology

B.J.A de Graaf

March 24, 2009

Examining Committee:

prof.dr.ir. B. De Schutter
L.D. Baskar MSc.
prof.dr.ir. J. Hellendoorn
V.L. Knoop MSc.

Faculty of Mechanical, Maritime and Materials Engineering · Delft University of Technology

Delft University of Technology

Abstract

Congestion on highways is becoming a bigger problem every day. With the increasing need for more highway capacity, which cannot always be met by building new highways, opportunities must be found in state-of-the-art technologies for more efficient traffic management. In this thesis traffic control is studied where the traffic consists solely out of intelligent vehicles arranged in platoons, i.e. vehicles that are clustered longitudinally on the highway. With the platooning concept more vehicles can be accommodated on the highway, thus increasing the traffic flow. As of today, no highway simulator is capable of simulating intelligent vehicles. In this thesis, first two existing open-source software packages, MITSIM and FreeSim are assessed regarding their potential for use as a traffic simulator for intelligent vehicles and platoons. As a result of this assessment, FreeSim is selected for further development. FreeSim is transformed such that it is possible to simulate intelligent vehicles. For this, platoon models are implemented. The platoon leaders are equipped with an intelligent speed adaptation (ISA) controller, such that they can follow a reference speed. The followers inside the platoon all have adaptive cruise control (ACC) installed, which makes it possible for them to follow a predecessor vehicle with a safe distance. Since the reference speed for the platoon leaders is not natively provided by FreeSim, FreeSim is adjusted so that a reference speed can be provided in two ways. If the traffic is controlled, the speeds are provided by a controller. For uncontrolled traffic, a METANET model is implemented to compute a reference speed for the platoon leaders. In this thesis we use a so-called "Big Car model" implemented in Matlab as a prediction model. In the Big Car model, a platoon is modeled as one big car with a variable length and has a platoon following behavior. Because FreeSim and the Big Car model use different models, the parameters of the Big Car following model are calibrated. This is done using a nonlinear least squares optimization algorithm. The optimization resulted in a set of parameters that gave a weighted root mean squared error as small as 1.16.

Keywords: intelligent vehicles, traffic simulator, platooning, modeling, calibration.

Acknowledgements

I would like to thank Bart De Schutter and Lakshmi Baskar for supporting me with my thesis project. During the regularly meetings you have always helped me with different kinds of issues and helped me with keeping track of my goal in a pleasant and supporting manner.

Delft, University of Technology
March 24, 2009

B.J.A de Graaf

Table of Contents

List of Figures

List of Tables

Nomenclature

D_i	Distance/length of edge i
γ_{head}	Distance between platoons
K_{l}	Controller gain
K_{v}	Controller gain
K_{x}	Controller gain
L_{v}	Length of a vehicle
ξ	Number of vehicles on an edge
n_{p}	Platoon leader
N_{p}	Number of vehicles inside a platoon
Ω	Weighted root mean squared error
ρ	Traffic density [veh/km]
V_i	Speed for edge i [km/h]
W_i	Weight of edge i [s]
a^+	Upper acceleration bound
a^-	Lower acceleration bound
$J(k)$	Objective function
k	Simulation step counter
q	Traffic flow [veh/h]
s_0	Minimum safe intraplatoon distance
s_{ref}	Reference distance headway
s_{head}	Space headway
T_{headway}	Time headway [s]
v	Average speed [km/h]
v_{ISA}	Reference (ISA) speed provided by roadside controllers [km/h]

v_{n_p}	Speed of the platoon leader [km/h]
x_n	Position of vehicle n measured from the last passed node [km]

Symbols related to METANET

η	Model parameter [km^2/lane]
κ	Model parameter [veh/km/lane]
λ_m	Number of lanes on segment m
$\rho_{m,i}$	Traffic density on segment i of link m [veh/km/lane]
τ	Model parameter [h]
I_n	Set of links leaving node n
L_m	Length of a segment in link m [km]
N_m	Number of segments in link m
O_n	Set of leaving links of node n
$q_{m,i}$	Traffic flow on segment i of link m [veh/h]
T	Simulation time step [s]
$v_{\mathrm{free},m}$	Average free-flow speed on segment m [km/h]
$v_{m,i}$	Mean speed on segment i of link m [km/h]

Acronyms

ACC Adaptive Cruise Control

ISA Intelligent Speed Adaptation

IV Intelligent Vehicles

MPC Model Predictive Control

RMSE Root Mean Squared Error

TTS Total Time Spent

Chapter 1

Introduction

Congested highways due to the ever increasing traffic demand are becoming a bigger problem every day. This has negative impacts on society, such as loss of time and increasing fuel consumption. The increasing need for more highway capacity cannot always be met by building more highways. As the technologies that are implemented in vehicles are getting more and more sophisticated, opportunities to increase the highway performance can be found in more advanced control of vehicles.

1-1 Problem Statement

Currently various control measures are used to control the traffic flow on highways. For instance ramp metering is used to gradually let vehicles access the highways such that the traffic density will not become too high. Or dynamic speed limits are used to control the average speed on a part of the highway. This can for instance be used to slow down the traffic upstream of a traffic jam. All control measures are first thoroughly tested before they are implemented on a large scale. A useful tool to test control measures is highway traffic simulation software. With traffic simulators certain control actions can be analyzed in regards to the way in which they can affect the traffic, without disturbing "real" traffic. All the current traffic simulators are designed to simulate vehicles with human drivers and their interactions with each other and the roadside. However, new vehicles that are introduced to the market are getting more and more sophisticated. For instance, it is now quite common that a vehicle is equipped with an anti-lock braking system and cruise control. But the more luxurious vehicles already have adaptive cruise control and lane-keeping assistance. All these systems that are built into the vehicles are designed to assist the driver in operating the vehicle. These systems will become even more intelligent in the future and will gradually take over the control of the vehicle. In the far future the task of the driver will only consist out of determining the destination of the vehicle.

The traffic simulator software packages that are currently freely available, all have the shortcoming that they are not able to simulate intelligent vehicles that can drive fully autonomously. This means that the current traffic simulators cannot be of use to test control

strategies on traffic that consists solely out of Intelligent Vehicles. The objective of this thesis project is therefore to find and modify an existing traffic simulator so that it can simulate intelligent vehicles.

1-2 Overview

The structure of this thesis is as follows. In Chapter 2 insight is given in technologies that enable intelligent vehicles. In this chapter also the used traffic flow characteristics are discussed as well as common traffic control strategies. This chapter concludes with discussing some models for intelligent vehicles. Chapter 3 presents the assessment made for identifying a traffic simulator that has the most potential to be used for simulating Intelligent Vehicles. In Chapter 4 it is explained how FreeSim is transformed such that it is capable of simulating intelligent vehicles. Chapter 5 consists of a case study that is performed to calibrate a prediction model with data gathered from the modified traffic simulator. In this chapter the experiments are discussed and also the results are discussed. Chapter 6 is the final chapter that contain the conclusions that can be drawn from this thesis as well as some recommendations for future work.

Chapter 2

Traffic Control with Intelligent Vehicles

This chapter gives insight into traffic control where the traffic consists solely of Intelligent Vehicles (IV), to be precise, IV that can drive fully autonomous. For that, a clear understanding of technologies that can be used in IV is needed. Therefore, Section 2-1 discusses the current and future technologies that enable fully automated IV. Some traffic characteristics and control strategies are treated in Section 2-2. Section 2-3 explains how IV can be modeled. These models can for instance be used for simulations or they can act as prediction models.

2-1 IV Technologies

IV systems can be found in many different types of road vehicles like cars, trucks, and military vehicles. These systems are used in IV for obtaining a more efficient driver-vehicle operation, and as such IV can be seen as the "next wave" of Intelligent Transportation Systems [7]. It is important to distinguish IV systems from current active safety systems, e.g. anti-lock braking systems and electronic stability control. IV systems assists the driver in operating the vehicle more efficiently, more safely, and/or with less stress, by assessing risk or sensing the environment [8]. The IV application areas can roughly be divided into three groups depending on the level of support to the driver:

- Advisory systems. These systems provide an advisory/warning to the driver. This is also referred to as *collision warning systems*. Examples are animal warning, side object warning (blind spot), and driver impairment monitoring.
- Semi-autonomous systems. These are systems that take partial control of the vehicle, either for driver assistance or for an emergency intervention to prevent a collision (*collision avoidance*). These systems use haptic measures, i.e. based on the sense of touch, to assist the driver. Semi-autonomous systems include functions such as lane-keeping, Adaptive Cruise Control (ACC), precise maneuvering, and precision docking.

- Fully autonomous systems. This kind of systems take full control of the vehicle (*vehicle automation*). Examples are low speed automation (for congested traffic), autonomous driving, and platooning.

The next subsections explain a few technologies that are needed for fully autonomous IV.

2-1-1 Lane-Keeping Assistance

Many minute steering adjustments made by drivers are a significant source of fatigue. Lane-Keeping Assistance assists the driver in these steering adjustments by adding a torque on the steering wheel. This torque increases as the vehicle nears a lane edge to create a "driving in a bathtub" sensation [8]. Lane-keeping assistance can use machine vision technology and line recognition to detect a lane. The disadvantage is that machine vision requires a good vision of the road, which is not the case with bad weather, e.g. snow or fog, and with bad or no line markings. Another possibility is the use of magnetic lane markers [12]. Then the accuracy is not subjected to weather conditions, but it requires investments in infrastructure, which brings additional costs.

Lane-keeping assistance systems are nowadays implemented in vehicles as convenience products. The lane-keeping assistance cannot be used by the driver as a "hands-off" system, because it requires steering inputs of the driver or else it will disengage. Also at sharp curves the system disengages. Full automatic "hands-off" steering will be commercially available in the future, when more advanced versions of lane-keeping assistance have been developed. In this thesis, it is assumed that IV do have the advanced version of lane-keeping assistance that do allow full automatic steering.

2-1-2 Intelligent Speed Adaptation

Vehicles equipped with Intelligent Speed Adaptation (ISA) systems receive the current speed limit on the road. This speed limit can differ from road to road and can depend on legislations, weather conditions, etc. The ISA system can receive the speed limit from roadside controllers or by using digital maps (with GPS technologies).

The speed limit can be fixed, variable, or dynamic [9]:

- If the speed limit is fixed, than there is one limit that will activate the ISA system.
- With a variable speed limit, the system adjusts the speed limit to the current road.
- With dynamic speed limits, the road environment is taken into account, e.g. bad weather, accidents.

The ISA system receives the maximum driving speed specified by the roadside infrastructure and can give feedback to the driver. There are three ways in which the feedback can be given to the driver:

- Advisory. When the driver exceeds the speed limit, he is warned by an audible and/or visual signal. This is thus only a warning signal and not a control signal.

- Voluntary. This system is like an intelligent gas pedal. The driver needs to use more force on the accelerator when the speed limit is exceeded, but it is still possible to exceed the limit.
- Mandatory. The driver is not able to exceed the speed limit, even if he wants to exceed it.

2-1-3 Cooperative Adaptive Cruise Control

Many vehicles today are equipped with a conventional cruise control. This system enables the vehicle to maintain a constant desired speed set by the driver. The driver regularly has to make sure that the speed is appropriate. ACC differs from the conventional cruise control, because if a vehicle immediately ahead of the equipped vehicle is driving at a slower speed, the ACC system controls the throttle and the brakes such that the equipped vehicle slows down and matches the speed of the vehicle ahead. When the ACC matches the speed of the vehicle ahead it also maintains a predefined driver-selectable time headway or gap. It is possible not to use a user defined time headway, but algorithms to maintain a safe inter-vehicle distance to avoid collisions [13, 14]. When the roadway ahead is unobstructed again, the desired speed is reattained. Possible technologies for monitoring the forward scene are radar or lidar (laser radar). In the future ACC equipped vehicles may also use machine vision instead of radar-based vision.

There are multiple variants of ACC. The most popular are high-speed ACC and low-speed ACC . The first ACC system on the market was a high-speed ACC, designed to operate around 40 km/h and above. This is because at high speeds, the discrimination between the vehicles ahead and stationary objects on the roadside is easier to make. Other vehicles traveling in the same direction will be moving at relatively low velocities, but non-targets have relatively high velocities and can thus be filtered out. Low-speed ACC operates at lower speeds and can also be a Stop-and-Go ACC, so that the system can manage a full stop and reinitiate forward motion afterward.

Although ACC systems can handle many traffic situations reliably, there are situations that ACC systems find difficult to handle, e.g. sudden and strong deceleration of the vehicle ahead. Cooperative ACC is a more advanced system than ACC because Cooperative ACC uses wireless technologies to exchange vehicle states (e.g. speed and acceleration) with the tracked vehicle. This means that in opposite to ACC where the vehicle states have to be measured with a laser, the wireless communication is also used such that the vehicle states are almost instantly known. In this way it is possible to have real-time information available that can be used in controlling the appropriate headway more tightly. Due to the tighter control that can be realized with Cooperative ACC, the safe time headway can be reduced to as small as 0.5 s [8].

2-1-4 Platooning

A very interesting concept to improve traffic flow with IV is platooning. A “platoon” is a cluster of IV arranged longitudinally on the highway [8]. Within a platoon, the vehicles communicate with each other to exchange essential information, e.g. speed and braking. The

first vehicle in a platoon is called the leader and the rest of the vehicles are called followers. Platooning is a far more advanced form of Cooperative ACC. The main difference from Cooperative ACC is that in Cooperative ACC, the vehicle exchanges information only with the one directly ahead. In a platoon every vehicle is aware of the states of all other vehicles in the platoon. This allows dynamic maneuvers like smooth merging of two platoons, splitting of a platoon into two platoons, or a lane change. Platooning is thus a fully automated process.

The spacing between cars in a platoon (intra-platoon) is around 1 m and the spacing between platoons (inter-platoon) is roughly 60 m [11]. With the platooning concept it is possible to maintain short distances between vehicles at high speeds [37, 24]. Because the intra-platoon distance is kept small, more vehicles can be accommodated on the highway, thus increasing the traffic flow on the network.

2-2 IV-Based Traffic Control

2-2-1 Traffic Characteristics

In this section a few basic characteristics of traffic will be explained to provide a basis for the understanding of the possibilities of controlling fully autonomous vehicles in traffic networks for an optimal traffic flow. First an important term called *time headway* and three traffic states (q, v, and ρ) will be defined.

Let us consider vehicles crossing a certain point on a freeway. The time that a vehicle n reaches that point is denoted by t_n^0, and the time that the vehicle has completely passed the measurement point, is denoted by t_n^1 as can be seen in Figure 2-1. The time headway of vehicle n is calculated by $T_{\text{headway}} = t_n^0 - t_{n-1}^1$. The time headway can also be expressed as:

$$T_{\text{headway},n} = \frac{s}{v_n}, \tag{2-1}$$

where s is the distance between vehicle n and its predecessor, and v_n is de speed of vehicle n.

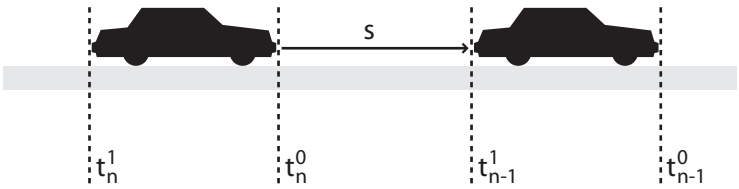


Figure 2-1: Time Headway $T_{\text{headway}} = t_n^0 - t_{n-1}^1 = \frac{s}{v_n}$

If the number of vehicles are counted that cross that point on the freeway during a time interval ΔT, then the traffic *flow* can be calculated as follows:

$$q = \frac{N}{\Delta T}, \tag{2-2}$$

where N is the number of crossing vehicles, and q is the traffic flow expressed in vehicles per hour. The (arithmetic) *average speed* (v) is computed as follows:

$$v = \frac{1}{N}\sum_{n=1}^{N} v_n. \tag{2-3}$$

The traffic *density* can be calculated by using the *theoretical flow formula* [19]:

$$q = \rho v \tag{2-4}$$

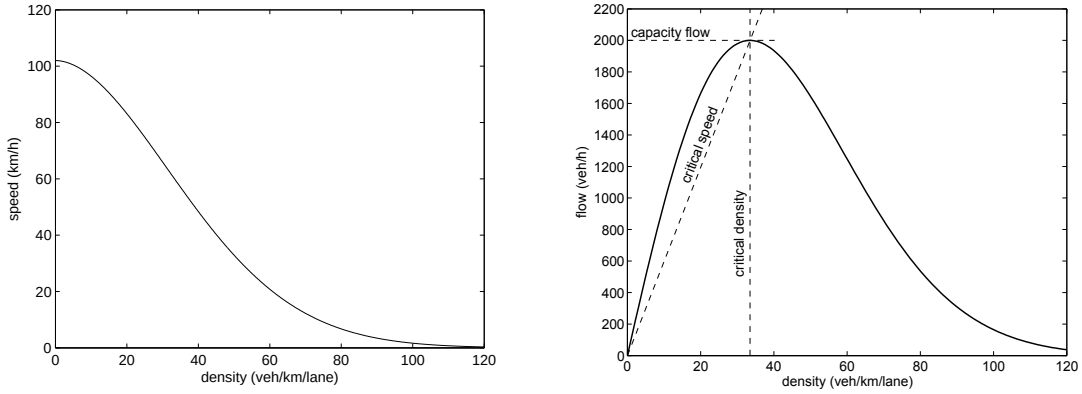
Thus $\rho = q/v$, where ρ is the *vehicle density* expressed in vehicles per kilometer.

If one wants to maximize the number of vehicles that pass a section of a highway during a certain time interval, thus increase the traffic flow q, there are intuitively two possibilities. One possibility is to increase the average speed v of the vehicles. The other possibility is to increase the density. This is justified by the theoretical flow formula (cf. (2-4)).

Theoretically, increasing both the average vehicle speed and traffic density will result in an increased flow, but in practice this is not always an option due to driver behavior. For example, if vehicles are traveling at high speeds, the distances between the vehicles are increased by the drivers for safety reasons. And if the distance between vehicles is reduced due to an increased traffic density ρ, drivers will decelerate to create a safe time headway between their own vehicle and the vehicle directly ahead. Hence, there is an optimum between average speed and vehicle density.

Speed-Density Relations

The theoretical flow formula (2-4) does not display the peculiar feature of traffic flow that the aggregate traffic speed decreases with increasing traffic density. Other functional relations between the traffic flow q, the average speed v, and the vehicle density ρ have been measured. A common used flow-density relation is the *fundamental diagram* [18], see Figure 2-2(b).



(a) Speed-density relation

(b) Flow-density relation

Figure 2-2: An example of the fundamental diagram with the corresponding speed-density relation. The critical speed is the speed that corresponds to critical density and maximum flow.

Assuming that in the fundamental diagram the number of lanes is equal to one, then the vehicle speed can be calculated by dividing the flow with the density, thus $v = q/\rho$. Following this relation, one can see that the vehicle speed is maximum at low densities and almost zero at high densities (cf. Figure 2-2(a)). This is because drivers can drive their desired speed, when traffic is unobstructed. This is called *free-flow*. In free-flow, drivers can maintain their desired

speed, because their time headway is very large. When the traffic density increases, drivers will drive in a so called *car-following* mode, i.e. drivers adjust their speed such that they can follow the vehicle directly in front, while maintaining a safe time headway. In congested traffic, the density has passed a certain *critical density* such that the traffic flow and vehicle speed decrease significantly. If the vehicle speed drops to almost zero, the density has become too large and a traffic jam will occur. The slope of the fundamental diagram starts almost linearly and corresponds to the free-flow speed. In this region the density can increase while the average speed stays the same, thus increasing the traffic flow. With increasing density, the traffic flow increases up to a maximum, i.e. the *capacity flow*, which is referred to as a critical point. The corresponding density and vehicle speed are the *critical density* and the *critical speed*. At higher densities than the critical density, the average vehicle speed is significantly lower than in free-flow traffic. This region corresponds to congested traffic. The expression for the fundamental diagram is as follows:

$$V(\rho) = v_{\text{free}} \exp\left[-\frac{1}{a}\left(\frac{\rho(k)}{\rho_{\text{crit}}}\right)^{a}\right], \tag{2-5}$$

where v_{free} is the free-flow speed, ρ_{crit} is the critical density, and a is a model parameter.

The fundamental diagram displays a smooth curve, but also non-smooth relations are proposed [22], because empirical flow-density data show discontinuities around some critical density. The flow-density plot proposed by [22] looks like a mirror image of the Greek letter lambda (λ). The two branches of this reverse lambda are used to define free low-density traffic and congested high-density traffic. Other relations of traffic flow are proposed but are beyond the scope of this thesis. The interested reader is referred to [19].

2-2-2 Control Strategies

A typical characterization of free-flow is the large average velocity. Congested traffic is the opposite of free-flow, because of the small average traffic speed. A traffic jam is a region where the average speed and flow is negligible compared to the very high density of the traffic. Traffic jams typically move upstream, thus against the direction of the traffic. If there are more vehicles that can leave the traffic jam at the downstream front, than there are vehicles that will enter the traffic jam at the upstream front, the traffic jam will reduce in length. The outflow of a traffic jam is more or less a fixed quantity [26]. If the fixed quantity is denoted by q^*, the traffic jam will thus reduce in width if $q_{\text{inflow}} < q^*$, where q_{inflow} is the traffic flow that enters the traffic jam at the upstream front. Control strategies have been developed with the objective to control this inflow of traffic jams. If the inflow of a traffic jam is controlled by speed limits upstream of the jam, a low-density wave is created that moves downstream. The high-density wave (the traffic jam) merges with the low-density wave created by the speed limits. The high-density and low-density wave can then compensate each other, thus eliminating the traffic jam.

Another control strategy is of course to prevent traffic breakdown when possible. When keeping the fundamental diagram in mind, the traffic can have a breakdown when the density is larger than the critical density. The traffic flow will reduce significantly when this happens. A possible control action is preventing that the traffic density exceeds the critical density. The inflow of the highway must thus be controlled before the capacity reaches its maximum.

This can be accomplished by controlling the highway access (on-ramps). Ramp metering is a general term used for describing techniques for restricting the access through on-ramps [3]. Ramp metering is of course only useful when traffic is not too light, because then there is enough capacity for the inflow of vehicles. Another possible control action is limiting the maximum vehicle speed. When the average vehicle speed is reduced, the traffic flow is decreased also. If the speed limits are lowered upstream of an area where the traffic capacity is at its maximum, the inflow of that area is thus delayed. Speed limits can in this way decrease the inflow of an almost congested area. Speed limits can be combined with ramp metering. Especially when ramp metering is not sufficient, speed limit control can help with regulating the traffic flow.

It is possible, however, that because of the improved traffic flow, vehicles run faster into another congestion downstream. If speed limits are applied at a traffic flow that is near the critical speed, a new high density wave can be created. This is because the average velocity is reduced and the traffic density will become higher than the critical density. This will create a congestion upstream of the speed limit control point. Local control measures can thus have effect on the traffic flow further upstream or downstream. Flows that arrive at a local controller can thus depend on control actions of other local controllers. It can be advantageous to have a control strategy based on the whole traffic network [18].

Traffic control can use Model Predictive Control (MPC) strategies. MPC is a *model*-based control method that computes control signals in order to optimize future process behavior [15]. MPC has already been extended to non-IV traffic management [18, 6]. It is however also possible to use MPC for traffic management and control with IV.

In Figure 2-3 a schematic representation of the MPC structure can be found. Here one can see that the MPC controller uses an explicit prediction model, in contrast to PID-like design methods. By using models in MPC, future output behavior of the system can be predicted on the basis of current system states and inputs applied to the process. In this way the next input signal that minimizes the objective function $J(k)$ can be computed. The objective function reflects the reference tracking error and the control action. A typical control objective is the Total Time Spent (TTS) by vehicles on the highway. The TTS of vehicles in a part of highway or a network is calculated by multiplying the number of vehicles by the time they traveled on that part of the highway or network. Of course the travel time depends partly on the distance drivers have to travel, but at a highway with or without a traffic jam, the total travel distance for drivers stays the same. The TTS at same travel distances is thus a measure for the length and duration of traffic jams and travel delays.

Minimizing the objective function is done with optimization. Solving the MPC optimization problem is the most demanding and time consuming operation in the MPC approach. At each time step k, the prediction model is simulated repeatedly within the optimization algorithm. It is thus important to choose an appropriate prediction model. Too complex models are usually not suited for MPC. A good trade-off must thus be made between the accuracy of the prediction model and the computational complexity of the prediction model. In Section 2-3 different models are discussed that can be used for simulation and as prediction models.

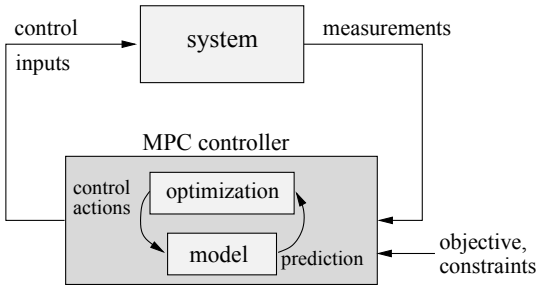


Figure 2-3: Schematic representation of the MPC structure.

2-2-3 Control Architectures

To control IV in an automated highway system, it is too complex for one controller to control all the vehicles in a highway network. Therefore, multiple control architectures have been developed to link roadside infrastructure and IV [16, 32, 34, 37, 4]. All the frameworks consist of multiple layers, which divide the control tasks. In this section two hierarchical control architectures are briefly discussed to give inside in possible hierarchical control strategies.

PATH Architecture

The California PATH program uses a hierarchical architecture composed of five self-organized functional layers [37]. Figure 2-4 gives a block diagram of this PATH architecture. The PATH framework assumes that the traffic is organized in platoons. In the PATH framework, a highway network consists of multiple interconnected highways. A highways in itself can be divided into links of about 5 km long. One link consists of multiple sections that are about 1 km long, and a section consists of lanes. A section contains at most one exit or one entrance ramp. The network and link layer controllers are located in the roadside, whereas the coordination and regulation layer controllers are part of the vehicles. In short, the different layers in Figure 2-4 can be explained as follows. The *network* layer computes the optimal routes of each vehicle in the network, the *link* layer assigns the path, the target platoon size and the target platoon speed, the *coordination* layer selects the maneuvers in coordination with its neighbors to follow the assigned path, and the *regulation* layer implements the maneuvers calculated by the coordination layer.

New Integrated Hierarchical IV-based Framework

The control architecture proposed in [4] distributes the intelligence between roadside infrastructure and vehicles, and uses IV-based control measures, like ISA and Cooperative ACC, to optimize the traffic flow. In Figure 2-5 the hierarchical control structure of the IV-based framework is shown. This framework is based on the platoon concept rather than on the section concept as in the PATH framework. This is because the roadside controllers in the PATH framework can have difficulties with assigning activities when a platoon is located in

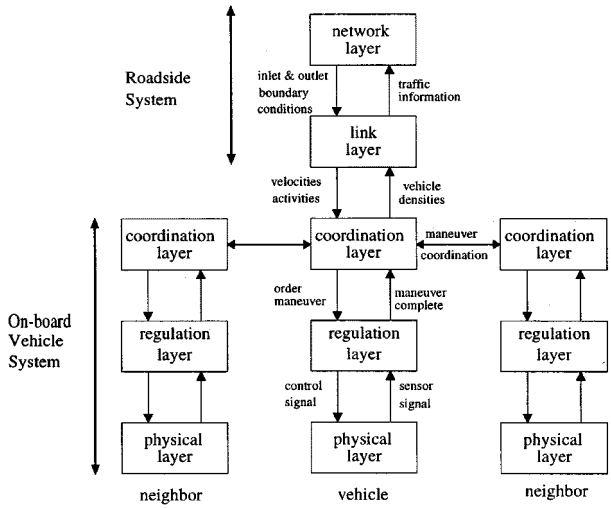


Figure 2-4: PATH architecture [21].

two sections (given that platoons are allowed to be long enough). The framework of [4] enables the integration of in-vehicle IV-based control measures (e.g. ISA) and roadside control measures (e.g. ramp metering), and it is also possible to integrate it with an MPC strategy.

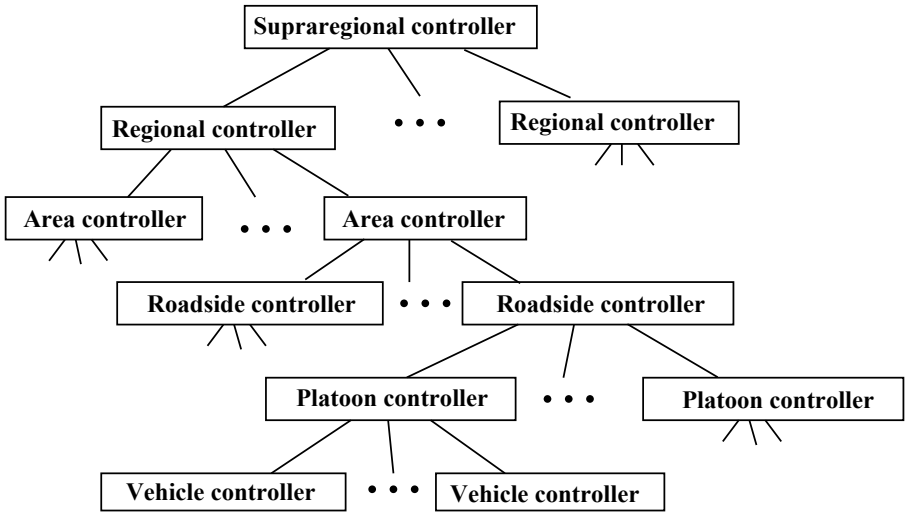


Figure 2-5: Hierarchical IV-based framework as proposed in [4].

2-3 IV Models

2-3-1 Traffic Model Classifications

In this section, various IV-based traffic models that can be used for simulations or that can act as prediction models in MPC are discussed. Traffic models can be classified according to various criteria [20], e.g. type of the independent variables (continuous, discrete), level of detail, representation of the process (deterministic, stochastic). Throughout this thesis, traffic models will be classified according to the level of detail, i.e. microscopic, mesoscopic, or macroscopic.

- *Microscopic* models describe the characteristics of individual vehicles as well as their interactions. From the individual vehicle characteristics and driver behavior, the acceleration, speed and position can be calculated for each vehicle.
- *Macroscopic* models operate on a more aggregate level and describe traffic without distinguishing individual vehicles. Macroscopic models deal with traffic flow in terms of average densities, average speeds, and average flows.
- *Mesoscopic* models describe traffic flow in medium detail level, and can be situated between microscopic and macroscopic models. In mesoscopic models, individual vehicles are not distinguished as in microscopic models, but the behavior is specified in individual terms. Some types of mesoscopic models are based on gas-kinetic theories. The advantage of gas-kinetic models is that the behavior of individual vehicles can be described, without the need to describe their individual time-space behavior.

Macroscopic models are very suited to describe the macroscopic characteristics of traffic flow like average densities and average flows. However, when it is important to describe microscopic characteristics of the traffic, macroscopic models are generally too coarse and thus not very suited. For simulating and controlling IV, microscopic models are the most suited models to use, because microscopic models describe each IV. Also especially for off-line simulation microscopic models are most suitable. Because of the importance of using microscopic models in controlling and simulating IV, the main focus in this section is on microscopic models.

For the sake of simplicity, only PID-type controlled IV models as used in [5] are taken into account, which will result in computationally less demanding simulations. In the models, it is assumed that the IV are controlled with IV-based control measures, like ISA and Cooperative ACC. The models are discretized models, where k denotes the simulation step counter. The leader model and the follower model are based on platoons, where the vehicles in a platoon are numbered such that the platoon leader is $n_{\mathrm{p}}= 1$, the vehicle behind the leader is $(n_{\mathrm{p}}+1)$, etc., and the last vehicle is $n = N_{\mathrm{p}}$. Here N_{p} denotes the number of vehicles inside the platoon.

2-3-2 Leader Model

The leader of the platoon must minimize the difference between the reference speed and the actual speed. The reference speed is the (ISA) speed provided by roadside controllers, and

is denoted by v_{ISA}. The acceleration of the leader is based on the difference between the reference speed and the actual speed, and is calculated by a proportional controller:

$$a_{n_{\text{p}}}(k) = K_1 \left[v_{\text{ISA}}(k) - v_{n_{\text{p}}}(k) \right] \tag{2-6}$$

here is K_1 the proportional gain constant, and $v_{n_{\text{p}}}$ the speed of the platoon leader.

2-3-3 Follower Model

The followers in a platoon must travel at (almost) the same speed as the platoon leader, while maintaining short intra-platoon distances. A following vehicle uses an ACC system to keep a safe distance between its predecessor, and to minimize the speed difference between the preceding vehicle and itself. The ACC controller has thus two tasks:

1. Keeping the headway distance as close as possible to the reference headway distance s_{ref}.
2. Minimizing the speed difference between the vehicle driving directly ahead and itself.

The reference distance headway (s_{ref}) consists of a stationary part, the distance that is to maintained at zero speed, and a variable part, the distance that is speed dependent. The reference speed of vehicle n is calculated as:

$$s_{\text{ref},n}(k) = s_0 + v_n(k)\, T_{\text{head},n} + L_{n-1} \tag{2-7}$$

where s_0 and L_{n-1} are the stationary part, and are respectively the minimum bumber-to-bumber distance, and the vehicle length of vehicle $n-1$ (see Figure 2-6). The speed dependent part is calculated by $(v_n(k)\, T_{\text{head},n})$, where $T_{\text{head},n}$ is the time headway of vehicle n.

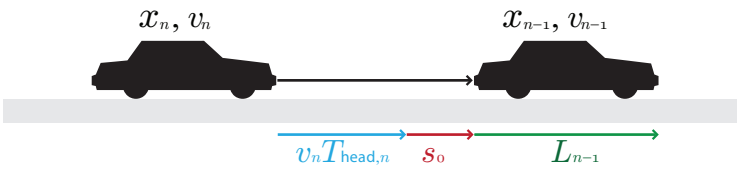


Figure 2-6: $s_{\text{ref},n}(k) = s_0 + v_n(k)\, T_{\text{head},n} + L_{n-1}$

The acceleration of the follower is then calculated with:

$$a_n(k) = K_x \left[s_{\text{ref},n}(k) - (x_{n-1}(k) - x_n(k)) \right] + K_v \left[v_{n-1}(k) - v_n(k) \right] \tag{2-8}$$

where K_x and K_v are controller constants, and x_n denotes the position of vehicle n.

2-3-4 Big Car Model

It is also possible to model platoons on a more aggregate level. This is possible by considering a platoon as a so called "Big Car", i.e. one single entity. The *acceleration* of a big car can be calculated with the leader model. The length of the platoon is variable and is dependent of

the number of vehicles inside the platoon, and the intraplatoon distances controlled by the ACC systems. The platoon length is thus speed depended and can be calculated with:

$$L_{\text{BigCar}}(k) = (N_{\text{p}} - 1)\left(s_0 + s_1 v_{n_{\text{p}}}(k)\right) + \sum_{n=1}^{N_{\text{p}}} L_n \tag{2-9}$$

where $L_{\text{BigCar}}(k)$ is the length of the platoon, s_1 is a model constant, $v_{n_{\text{p}}}$ is the speed of the platoon (leader), and $\left(s_0 + s_1 v_{n_{\text{p}}}(k)\right)$ is the speed-dependent intervehicle spacing.

2-3-5 METANET Model

The models mentioned in the previous sections are microscopic models. In this section a macroscopic model, called *METANET* [28, 17, 23] is discussed. In Chapter 4 and 5 this model is used to compute a reference speed for the platoon leader if the traffic is uncontrolled. The reasoning behind the use of this model in the context of platooning is explained in Chapter 4.

In the METANET model a freeway network is divided into *links*. Each link is a structure of a freeway that has no major changes in characteristics or geometry, e.g. no on-ramps or off-ramps. If a major change occurs in the characteristics of the freeway, a node is placed. So a link connects two nodes. Every link is in turn divided into N_m *segments* of length L_m. The links and segments are indicated by the index m and i respectively (cf. Figure 2-7). The time step used for the simulation of the traffic flow is denoted by T, such that the expression for the time instant is $t = kT$. For every simulation time step T and segment length L_m, the stability criterion:

$$L_m > v_{\text{free},m} T,$$

where $v_{\text{free},m}$ is the average free-flow speed, should be satisfied. Typical values for L_m are in the range of 500–1000 m.

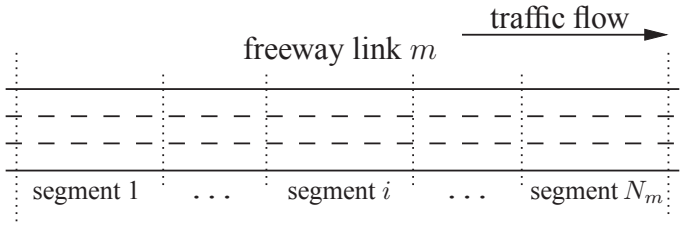


Figure 2-7: The METANET model divides a freeway network into links (m), and each link is divided into segments (i).

There are three quantities that characterize each segment:

- *traffic flow*: $q_{m,i}(k)$ [veh/h],
- *traffic density*: $\rho_{m,i}(k)$ [veh/km/lane],
- *mean speed*: $v_{m,i}(k)$ [km/h].

These three quantities are related to each other by (2-10), where λ_m is the number of lanes on that segment.

$$q_{m,i}(k) = \rho_{m,i}(k)\, v_{m,i}(k)\, \lambda_m \tag{2-10}$$

In the METANET model the average speed at the next simulation step $(k+1)$ is computed by taking the average speed at time instant k plus a *relaxation term*, a *convection term*, and an *anticipation term*. The relaxation term is based on the speed-density relationship, i.e. the fundamental diagram (see Section 2-2), and states that the drivers try to achieve a desired speed $V(\rho)$. The relaxation term is expressed as:

$$\frac{T}{\tau}\Big(V(\rho_{m,i}(k)) - v_{m,i}(k)\Big),$$

where $V(\rho_{m,i}(k))$ is in fact given by (2-5), but now with link and segment indexes (m and i) incorporated:

$$V(\rho_{m,i}) = v_{\text{free},m} \exp\left[-\frac{1}{a_m}\left(\frac{\rho_{m,i}(k)}{\rho_{\text{crit},m}}\right)^{a_m}\right]. \tag{2-11}$$

The convection term is expressed as:

$$\frac{T}{L_m} v_{m,i}(k)\Big(v_{m,i-1}(k) - v_{m,i}(k)\Big),$$

and represents the speed difference caused by the inflow of vehicles, hence the difference between the average speed of the upstream segment $v_{m,i-1}(k)$ and the current segment $v_{m,i}(k)$ is calculated. The anticipation term expresses the speed difference caused by the density increase (or decrease) that drivers experience of the segment downstream:

$$-\frac{\eta T}{\tau L_m}\frac{\rho_{m,i+1}(k) - \rho_{m,i}(k)}{\rho_{m,i}(k) + \kappa}.$$

So the total expression for the mean speed at the simulation step $k+1$, where all three terms are combined, can now be written as:

$$\begin{aligned} v_{m,i}(k+1) = \; & v_{m,i}(k) + \tfrac{T}{\tau}\Big(V(\rho_{m,i}(k)) - v_{m,i}(k)\Big) + \\ & \frac{T}{L_m} v_{m,i}(k)\Big(v_{m,i-1}(k) - v_{m,i}(k)\Big) - \\ & \frac{\eta T}{\tau L_m}\frac{\rho_{m,i+1}(k) - \rho_{m,i}(k)}{\rho_{m,i}(k) + \kappa}, \end{aligned} \tag{2-12}$$

where τ, η, and κ are model parameters. It is important to see that in the METANET model the average speed of segment i on link m at simulation step $(k+1)$ is not only dependent on the current average speed $(v_{m,i}(k))$ and density $(\rho_{m,i}(k))$, but also on the average speed upstream $(v_{m,i-1}(k))$ and the density downstream $(\rho_{m,i+1}(k))$.

As was stated in before, a node is placed between links if there is a major change in the road characteristics or geometry. This means that it could happen that the first segment in a link can have more than one average upstream speed if there are more incoming links. It is as well possible that the road splits, such that the last segment in a link has more than one downstream density. The METANET model deals with this by aggregating multiple segments into one virtual segment.

Upstream Speed

If a node n has multiple incoming links and one leaving link m (as shown in Figure 2-8), the last segments of the entering links are treated as one virtual segment with a virtual upstream speed:

$$v_{m,0}(k) = \frac{\sum_{\mu \in I_n} v_{\mu,N_\mu}(k)\, q_{\mu,N_\mu}(k)}{\sum_{\mu \in I_n} q_{\mu,N_\mu}(k)}, \tag{2-13}$$

where I_n is the set of links leaving node n.

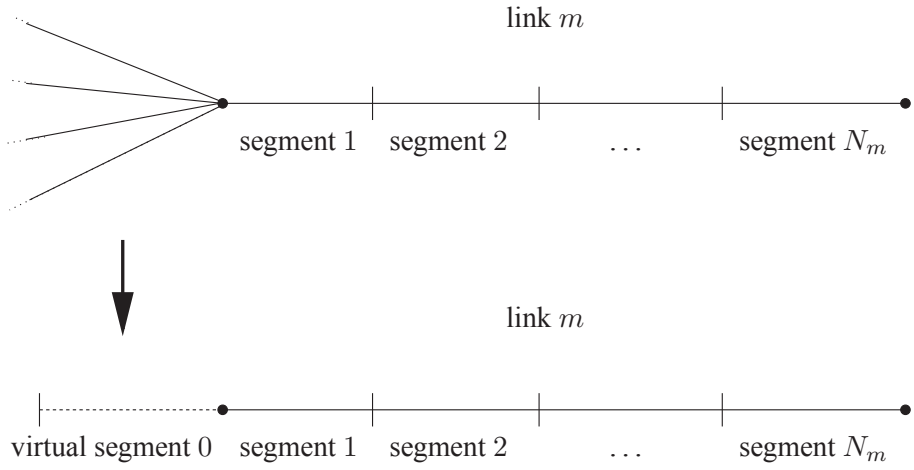


Figure 2-8: The last segments of the links that enter a node are aggregated into one virtual segment with a virtual upstream speed $v_{m,0}(k)$.

Downstream Density

When node n has one entering link, but multiple leaving links (as shown in Figure 2-9), the leaving links are aggregated into one virtual leaving link with a virtual downstream density:

$$\rho_{m,N_{m+1}}(k) = \frac{\sum_{\mu \in O_n} \rho_{\mu,1}^2(k)}{\sum_{\mu \in O_n} \rho_{\mu,1}(k)}, \tag{2-14}$$

where O_n is the set of leaving links of node n.

2-4 Summary

In this chapter a few important technologies are explained that enable the traffic control of fully autonomous IV. An overview is given on the characteristics of traffic flow, including

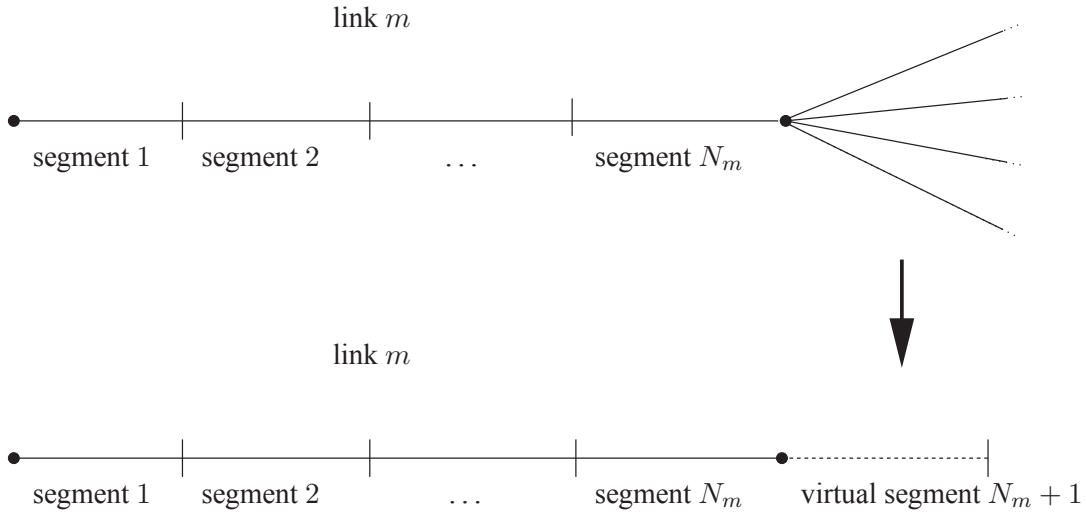


Figure 2-9: The first segments of the links that leave a node are aggregated into one virtual link with a virtual downstream density $\rho_{m,N_{m+1}}(k)$.

relations between average speeds and densities. As can be seen in this chapter, speed limiting control can be an effective control measure to decrease the TTS. Traffic consisting of ISA equipped platoons, can thus increase traffic flow, if controlled well, for instance in a hierarchical control framework with MPC incorporated. The use of appropriate models, i.e. with a good trade-off between the accuracy and the computational complexity of the model, in MPC is necessary, and therefore some IV-based models that can be used in an MPC strategy have been explained. METANET is a macroscopic model that divides a freeway network into links and segments. METANET can be used to compute the average speed of a segment at the next simulation time step.

Chapter 3

Simulation Software

Traffic simulation software is very often used as a tool for traffic control. This is, among other reasons, because a traffic simulator can generate different traffic scenarios, predict traffic states, and it is possible to test (experimental) control measures without disrupting "real" traffic. Such a traffic simulator can also act as a system model or as a prediction model in a Model Predictive Control (MPC) design. In this chapter different traffic simulation software packages will be investigated for their potential use as a traffic simulator for Intelligent Vehicles (IV), and especially in an MPC strategy. There exists a wide variety of traffic simulators, and it is not feasible in this thesis to discuss all of them. So in the first section of this chapter, an overview of the state of the art traffic simulators is given. In Sections 3-2 and 3-3 the two most promising software packages for use in this thesis are analyzed. In Section 3-4 these two simulators are compared with each other. Based on this assessment, the most promising traffic simulator is chosen for further development.

3-1 State of the Art

Each traffic simulation software package often focuses on one type of traffic model. In Section 2-3-1 it was already seen that traffic models could be classified according to the level of detail, and that microscopic models are the most suited for controlling and simulating IV. Therefore, only microscopic traffic simulators are considered in this chapter. Research has been done in comparing microscopic traffic simulators that support Intelligent Transportation Systems [10, 25, 39]. Based on these studies, the choice between all available simulators can be narrowed to:

- TSIS-CORSIM [33, 29]
- FreeSim [25]
- INTEGRATION [36]
- MITSIM [38, 2]

- Paramics [31]

All of the above mentioned traffic simulators are suitable for simulating traffic on freeways. The problem with all of these simulators is that none of them can model IV-based traffic, i.e. no models for Intelligent Speed Adaptation (ISA), Adaptive Cruise Control (ACC) and platooning are currently included. It is therefore necessary to have the possibility to change the traffic models of the simulators. In this way the human traffic models can be replaced with models of IV. The most convenient way of changing models is by editing the source code of the program. However, this is only possible when the source code is freely available, and currently only MITSIM and FreeSim are open source. Another way of customizing the features of a simulator can be done with a so called Application Programming Interface (API) or a plug-in. An API is a set of protocols that is made available by the developer, in which third party programs can interact with the main application. This is a common way to modify or to extend a program, without changing the original (closed) source code. A disadvantage of working with plug-ins, is that the possibilities are restricted by the developer. A second disadvantage is that an API only provides an user interface, but it is not always clear what really happens inside the program. So when creating a plug-in a lot of testing has to be done, to make sure that the modifications work as desired.

Another important aspect to look at, are the costs of a traffic simulator. The price of traffic simulation software can be quite significant, e.g. $ 1,000 for TSIS-CORSIM. Both MITSIM and FreeSim are available free of charge, and the Delft University of Technology does have a license for Paramics, so no additional costs are necessary if one of these three software packages is chosen for further development.

Based on the two criteria, i.e. the freedom of changing the program features and additional costs, both MITSIM and FreeSim have the most potential to be used as a traffic simulator for IV and platoons, because they are free and open source. In the next sections, MITSIM and FreeSim are further investigated to see which is the best option to choose.

3-2 MITSIM

3-2-1 Description

MITSIM stands for MIcroscopic Traffic SIMulator, and was developed at MIT and released in 1993. Its purpose was to provide an environment for testing designs of dynamic traffic management systems [38]. MITSIM simulates individual vehicles using car-following models, lane-changing models, and probabilistic route choice models. MITSIM is coded in C++ using object oriented design. It also contains a graphical user interface for vehicle movement animations.

The simulation is time-driven, as can be seen in the flowchart in Figure 3-1 [38]. The simulation starts with loading all the parameters, the road network, scenario definition, and initializing all the communication channels. Then an iterative procedure starts. Each iteration includes tasks that are performed consecutively. These tasks are for instance, updating of traffic signals and updating of vehicle states.

3-2-2 Disadvantages

MITSIM runs on the Linux operation system, and the user's guide states that the Redhat Linux 7.3 distribution is needed in order to compile the source code. Compiled executables should work on newer Linux distributions, including Fedora Core. Because the source code must be edited in order to the change models, it is necessary to be able to compile the source code. Redhat Linux 7.3 is a quite old distribution, but can still be downloaded from some repositories. It is, however, not compatible with most of the modern day computers. Installing it on an available computer, did not result in a working operation system. It is therefore chosen to try to install MITSIM on Fedora 9, which is a descendant from Redhat Linux.

The installation instructions for installing MITSIM, are rather straightforward. In short, first PVM, which facilitates between the packages in MITSIM has to be installed prior to the installation of MITSIM. Next the GUI Libraries must be installed by running an install script. After the install process some paths to the program has to be set. However, during the installation process, some errors made it impossible to continue. A few errors could be solved by removing non-vital installation steps, but the real problem was due to the fact that the MITSIM is written in C++ code that is dependent on old C++ compilers. The newer C++ compilers which are installed with Fedora 9 do have restrictions for compiling old code. We have tried to install older C++ compilers and to use these to compile the source code, but this did not solve the installation problem, as these gave rise to other new errors. All in all, it proved impossible to install MITSIM in Fedora 9.

MIT does not provide support for the open source MITSIM, but there exists however an user group website [1], where questions can be posted, and other users can try to help. In this user group more users are facing the same problem with installing MITSIM on different Linux distributions than Redhat Linux 7.3, e.g. Ubuntu. The fact that MITSIM is not actively developed anymore, makes it thus unsuitable for modern day computers and operating systems.

3-3 FreeSim

3-3-1 Description

FreeSim is a quite new freeway traffic simulator, as it was made public in 2007 [25]. FreeSim models free-flowing traffic, and was created with the idea of an Intelligent Transportation System that gathers individual vehicle states, by having all vehicles remain in communication with a central server. FreeSim is written in Java and is platform independent, so it can run on any operating system. The graphical user interface is created in Adobe Flash and runs from within a web browser. A MySQL database is used to store all data needed. All third-party applications needed to compile or run FreeSim are freely available and free of charge.

In FreeSim the freeway system is represented by a graph data structure. The data of the graph consist of edges (segments) which are connected with each other by nodes. Prior to beginning a simulation, the freeway system has to be defined and stored in a database. The population of the database is done with another program which is bundled with FreeSim, so

the user only has to create two text files with a list of nodes and a list of edges, respectively. When the graphical user interface is started up, this freeway system is available and can be rendered. The graphical user interface creates a socket connection with a server program to access all the simulator functionalities, see Figure 3-2. During a simulation, multiple events can occur that can be predetermined by the user. This is done in a text file that is processed at the start of the simulation. FreeSim determines the shortest or fastest paths for vehicles, based on the current distances and speeds of the segments. FreeSim contains multiple fastest paths algorithms, but this can easily be extended by other algorithms. In Appendix A a more extensive description how FreeSim works can be found.

3-3-2 Disadvantages

The installation of FreeSim, unlike MITSIM, gives no problem at all. The only problem that can occur, is if the user uses the Adobe Flash 9 plug-in, which needs explicit permission to create a socket connection to a server. As was mentioned earlier, the Flash interface does need this ability to connect to the FreeSim server. This problem can be solved by running a small socket policy file server, which gives permission to the Flash interface to create a socket connection. This solution is described in [35].

The main disadvantage of FreeSim, is that it is currently a rather basic traffic simulator. Firstly, FreeSim only simulates vehicles in free-flow. There is no car-following model implemented and the vehicle speeds are not dependent on the traffic density, thus congestion is not modeled. This is not a major problem, because the intention is to change these models anyway. However, now models have to be created and implemented, instead of adjusted. Secondly, vehicles travel with a constant speed along a segment. If the speed of a segment is adjusted during a simulation, it only affects new vehicles that enter the segment. The vehicle speed is thus only adjusted when the vehicle crosses a node. Whether this is a flaw in the program or intentionally programmed this way is not clear.

To use FreeSim as a system model in an MPC strategy, it must be possible to apply external control measures. This can be done for instance, by creating a socket connection between MATLAB and FreeSim, where the MPC commands from MATLAB are given to FreeSim. It is also possible for MATLAB to request traffic data through this socket connection, but this can also be done by direct accessing the MySQL database from MATLAB. Because FreeSim is object oriented and because Java does support socket connections, and as MATLAB supports Java, this could be done. But to use FreeSim as a prediction model in MPC, the simulation has to be really fast. Currently, a simulation step, where a few hundred vehicles are updated, takes about 3 to 10 seconds to execute. This will probably only take longer if FreeSim is extended with additional models where more computations has to be done during a simulation step. So FreeSim is not suitable to act as a prediction model.

3-4 Comparison

In order to use an existing microscopic traffic simulator to use for simulating IV-based traffic, the options are rather limited. As could be seen in the previous section, the best option is to use a free and open-source program. Only MITSIM and FreeSim qualify for these criteria.

MITSIM exists for quite some time and is capable of describing more “complex” simulations unlike FreeSim, which is very new and rather limited in its capabilities. The major downside of MITSIM is however, that it cannot be used on a modern computer.

This leaves FreeSim as the only microscopic traffic simulator with the potential for using it as an IV-based traffic simulator. But, some major adjustments have to be made, in order to give desired simulations. Firstly, the segment speed and the vehicle speed have to be decoupled, such that vehicles on the same segment can have different speeds and the vehicle speeds can be updated while traveling along an edge and not only when vehicles cross a node. Secondly, the speed of vehicles should be dependent on the capacity of the segment they are traveling along, because now when the traffic becomes very dense, the speed of the vehicles is not affected. And at last, the models for simulating platoons have to be implemented. Fortunately, FreeSim provides the possibility to adjust its source code, such that all of these adjustment can be made.

3-5 Summary

In this chapter some traffic simulation software packages are discussed. It could be seen that none of the existing microscopic traffic simulators are capable of simulating IV and platoons. It is possible to change the features of current traffic simulators, either by using plug-ins or by changing the source code. The most convenient way is by using a free and open-source traffic simulator. Only MITSIM and FreeSim are free and open-source. MITSIM is however not suitable to use, because it needs an old version of Linux, which is not compatible with current computer hardware. FreeSim is thus chosen for further development, but it needs a lot of modifications, because its current set of features are rather limited. FreeSim is not suitable to act as a prediction model in an MPC strategy, however, this does not have to be a problem as will be seen in the next chapters.

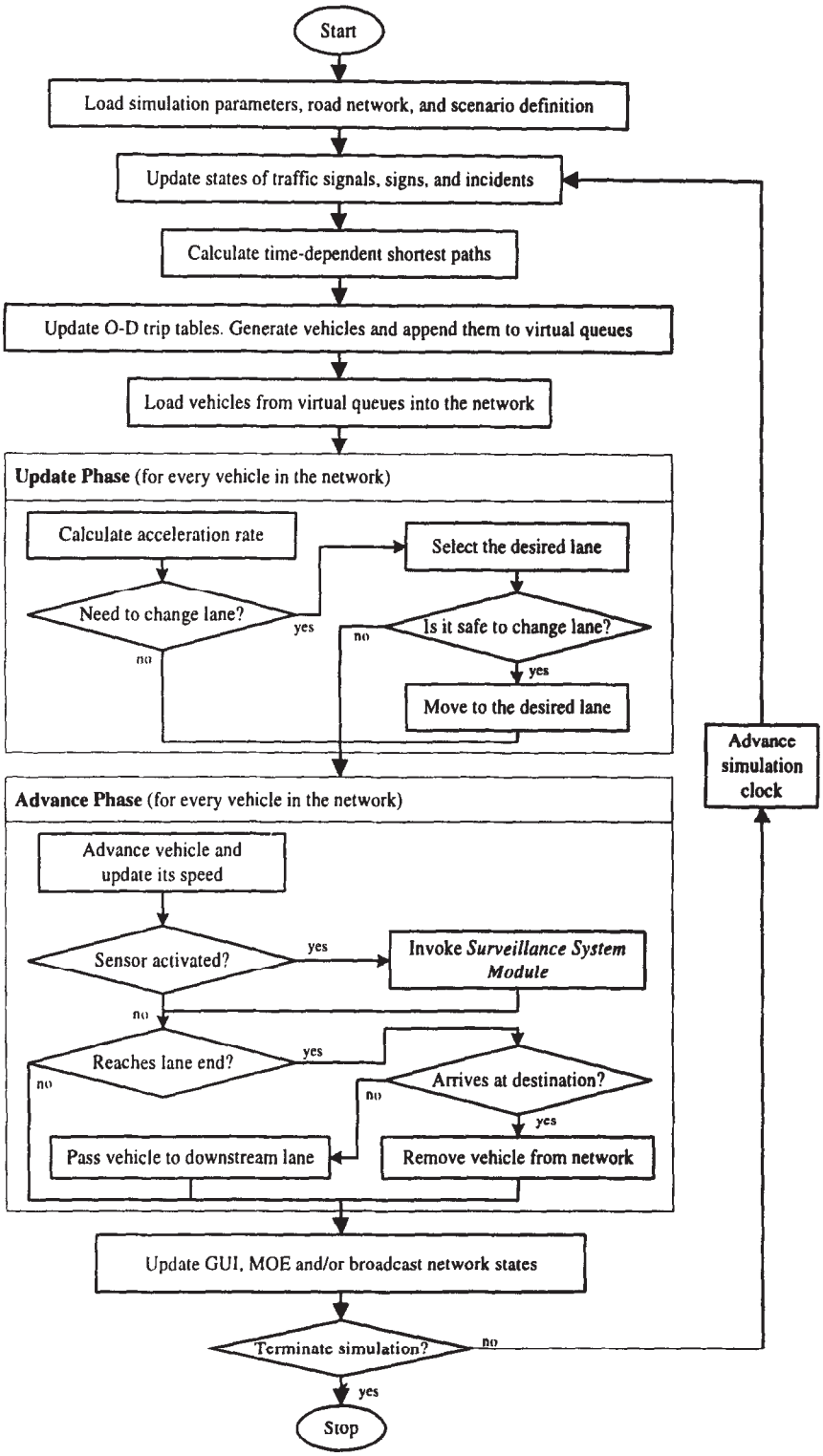


Figure 3-1: Flowchart of the MITSIM simulation model [38]

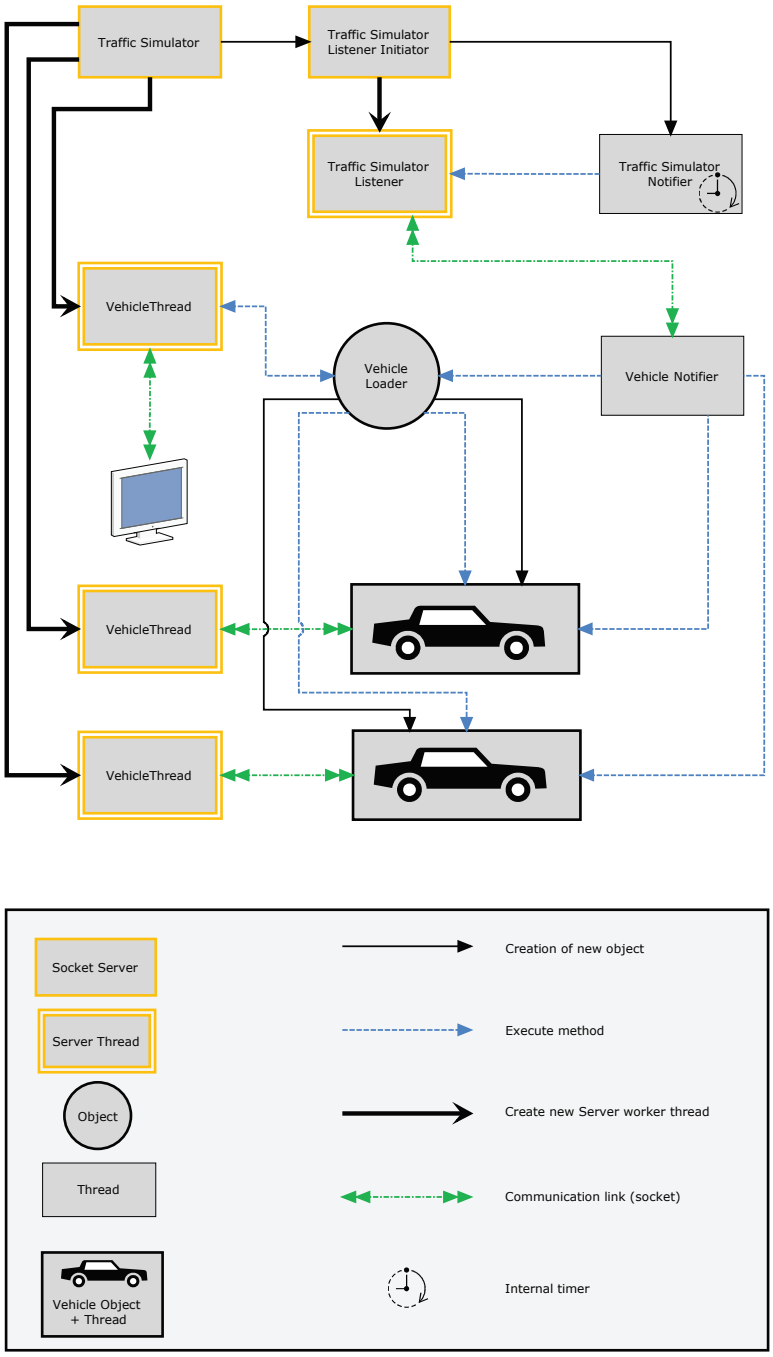


Figure 3-2: FreeSim diagram

Chapter 4

FreeSim

As could be seen in Chapter 3, FreeSim is selected for further development. In this chapter the enhancements that are made to FreeSim are explained as well as the trade-offs and compromises made are discussed. In Appendix A a detailed description of the layout and workings of FreeSim is explained. In this chapter we give enough explanation about the current workings of FreeSim, without going into too much detail about the exact code. If, however, one wants a more detailed overview of the code, Appendix A is a good reference.

In FreeSim the traffic network consists of nodes and edges, where an edge connects two nodes. Note that an edge in FreeSim is the equivalent of a segment in METANET.

In FreeSim all distances are measured in miles and the speeds are expressed in miles per hour. Before any other changes are made to FreeSim, these units are converted into the metric system, thus from now on distances and speeds are expressed in kilometers and in kilometers per hour respectively.

4-1 Vehicle Speed

4-1-1 Current Situation

Currently, each edge is defined with some parameters, i.e. an ID, source node, destination node, distance, speed, and a weight. When an edge is created, which happens prior to a simulation, all parameters are initialized. Only the speed and weight can be adjusted during a simulation. The weight of an edge denotes the time, expressed in seconds, it takes for a vehicle to travel that edge. When, during a simulation, the edge speed is updated the weight is adjusted accordingly:

$$W_i = 3600\frac{D_i}{V_i},$$

where W_i is the weight of edge i, D_i is the distance/length of edge i, and V_i is the speed of edge i.

When a vehicle n is instantiated, it requests from the server the path it must travel, and then it asks for the time it takes to traverse the first edge. The response of the server consists of, among other data, the *travel-time* the vehicle gets to travel that edge and the distance it has to travel, i.e. the edge weight W_i and distance D_i. These two parameters are stored as local variables by the vehicle as ω_n and δ_n respectively. Also the time the vehicle starts traveling on the edge $t_{0,n}$ is stored as a local vehicle variable. At each simulation step, the vehicle checks whether it is time to move to the next edge. This is based on the condition:

$$t_{0,n} + \omega_n \leq t_c, \tag{4-1}$$

where t_c is the current time. So, the vehicle basically is told (once) that it is supposed to travel for W_i seconds, and the vehicle just waits until that time is passed. When criterion (4-1) is met, the vehicle again requests the server for the travel time for the new edge $(i+1)$. The vehicle also sets $t_{0,n} = t_c$. This is a blunt error, because by setting the time the vehicle starts traveling on the new edge equal to the current time, it is assumed that it crosses the node exactly at the current time, which is not necessarily true. This has thus to be corrected (see Section 4-1-2).

After each simulation step, the current states are written to the MySQL database. The position is measured as the distance traveled from the last passed node:

$$x_n = \frac{\delta_n}{\omega_n} \left(t_c - t_{0,n}\right).$$

The current speed of the vehicle is calculated as follows:

$$v_n = 3600 \frac{\delta_n}{\omega_n}.$$

The (relative) position and vehicle speed is thus not a local variable, which is updated every simulation step, but are merely computed every simulation step and written to the database. As stated before, the database only provides data for the users after the simulation is finished.

4-1-2 Modifications

Transition Between Edges

When vehicle n crosses a node, so (4-1) holds true, the time at which this happens is erroneously set to the current time ($t_{0,n} = t_c$). It is possible that a vehicle crosses this point in between two simulations steps, so ($t_{0,n} + \omega_n < t_c$). In this way the vehicle is modeled as if it is standing still on the node, and continues its path when the next simulation step starts. This error is removed by setting the new value of $t_{0,n}$ as follows:

$$t_{0_{\text{new}},n} = t_{0_{\text{old}},n} + \omega_n. \tag{4-2}$$

Now the behavior of the transition between edges is proper modeled and the vehicles cross nodes in a continuous manner.

Decoupling Speeds

In the current method when the speed of an edge is adjusted, the vehicles already traveling on that edge do not notice that the speed is changed. This is due to the fact that the vehicles only interact with the server when the vehicle is instantiated or when it crosses a node. It is important that vehicles have their own speed, such that vehicles on the same edge can have different speeds. Also the position of each vehicle should be known during a simulation, because the speeds of the vehicles inside a platoon should be dependent of the speeds and positions of each other, as will be seen in Section 4-2. So FreeSim is adjusted, such that each vehicle has their position (x_n) and speed (v_n) as a local variable, which must be updated each simulation step. Note that x_n is still measured as the distance traveled from the last passed node. Because each vehicle now has its own speed variable, the speed of an edge can now be used as a speed limit for all vehicles traveling on that edge.

Each vehicle checks whether it is time to move to the next edge by checking whether (4-1) is true. This condition can still hold, but for that it is necessary that ω_n is changed properly when vehicle n adjusts its speed. The initial value of ω_n is set as follows:

$$\omega_n = 3600\frac{\delta_n}{v_n}.$$

If the speed of vehicle n is adjusted, ω_n has to be adjusted accordingly:

$$\omega_n = \left(3600\frac{\delta_n - x_n}{v_n}\right) + (t_c - t_{0,n})\,. \tag{4-3}$$

The first part of (4-3) is the time it takes for vehicle n to travel from its current position to the end of the edge, assuming v_n is kept constant, and the second part is the time it already took for the vehicle to get from the source node to its current position. In this way ω_n still is meaningful as it represents the total travel time needed to travel from the source node to the destination node of the current edge.

Initial Speed

A vehicle (n) is currently instantiated with an initial speed (v_n) equal to the edge speed (V_i), which is now used as a speed limit. This could be a problem, if for instance another vehicle ($n-1$) is traveling ahead on the same edge, but for some reason with a much lower speed than the speed-limit. So $v_n \gg v_{n-1}$. In this case it can happen that these two vehicles collide with each other if the distance between the vehicles is too small for vehicle n to reduce its speed in time such that $v_n \leq v_{n-1}$. This problem can be solved by making sure that the initial speed of vehicle n is dependent of vehicle ($n-1$). To adjust this in FreeSim, first of all the vehicle that last entered the edge should be known. If this vehicle is known, a security check can be performed that consists of measuring the distance between the last vehicle that entered the edge and the vehicle that is instantiated. If this distance is lower that a certain threshold ϵ, the initial speed should be the minimum of vehicle ($n-1$) and the speed-limit. So the initial speed of vehicle n is:

$$v_n\left(t_{0,n}\right) = \begin{cases} \min\left(V_i, v_{n-1}\left(t_{0,n}\right)\right) & \text{if } x_{n-1} \leq \epsilon \\ V_i & \text{if } x_{n-1} > \epsilon \end{cases} \tag{4-4}$$

It is chosen to set ϵ equal to 200 m. This provides a big enough safe clearance, such that a collision is avoided by initializing a vehicle with an appropriate speed.

The most convenient way of keeping track which vehicle last entered a certain edge, is by adding a local variable to each edge object, which contains the last vehicle that entered the edge. When a vehicle n starts traveling on a new edge, it communicates with the server to receive data about that edge. So when that response is processed by the vehicle, it 'logs on' to the new edge by setting the variable of the edge, containing the last entered vehicle, such that it matches vehicle n. In this way a new vehicle that is instantiated can request the last entered vehicle on the edge, and when this vehicle is known the new vehicle can request from the last entered vehicle its position and speed.

4-2 Platooning

In Section 4-1 it is explained how the vehicle speed is decoupled from the edge speed. The original edge speed is now used as a speed limit and each vehicle does now have a variable speed and keeps track of its position. In this section the implementation of platoon models in FreeSim is explained.

In FreeSim a platoon object is created that provides all the necessary data needed for all the vehicles inside a platoon. A platoon contains for instance a table with all the vehicles and their relative positions inside the platoon. In this way a vehicle can look in its platoon to see which vehicle it has to follow, or to see if it is the platoon leader. From now on every vehicle in FreeSim does belong to a platoon. However, a platoon can still consist out of only one vehicle.

In every simulation step, each vehicle will try to maintain an appropriate speed. In FreeSim two functions are implemented (one for the platoon leader and one for followers) that vehicles use to compute a new speed. In each simulation step, each vehicle checks whether it is a platoon leader or a follower. Based on this check, each vehicle uses the correct function to adjust its speed.

4-2-1 Controller Platoon Leader

In this section the function to compute the new speed for the platoon leader is discussed. A platoon leader tries to minimize the difference between the reference speed (v_{ref}) and its actual speed ($v_{n_{\text{p}}}$), as explained in Section 2-3. The reference speed is provided by the edge it is traveling on. For the moment let us assume that the reference speed is equal to the speed limit: $v_{\text{ref}} = V_i$. The acceleration of a platoon leader is modeled by a proportional controller (cf. (2-6)):

$$a_{n_{\text{p}}}(k) = K_{\text{v}}\left[v_{\text{ref}}(k) - v_{n_{\text{p}}}(k)\right] \tag{4-5}$$

In FreeSim the computed acceleration is limited, such that the acceleration stays between a lower and upper bound:

$$a_n = \begin{cases} a^+ & \text{if } a_n > a^+ \\ a^- & \text{if } a_n < a^- \\ a_n & \text{if } a^- \leq a_n \leq a^+ \end{cases} \tag{4-6}$$

Here a^+ and a^- are the upper and lower acceleration bounds respectively. The values of the acceleration bounds are set in Section 5-2-1.

The new speed of a vehicle is computed as follows:

$$v_n(k+1) = v_n(k) + a_{n_\mathrm{p}}(k)\,T, \tag{4-7}$$

where T is the simulation time step. Also the new speed is bounded, such that it is not higher than the reference speed v_{ref} (only when accelerating), and not lower than the lower speed limit v^-, which is typically zero:

$$v_n = \begin{cases} v^{\mathrm{ref}} & \text{if } v_n > v^{\mathrm{ref}} \text{ and } a_n > 0 \\ v^- & \text{if } v_n < v^- \\ v_n & \text{everywhere else} \end{cases} \tag{4-8}$$

4-2-2 Controller Followers

The followers inside a platoon, i.e. all vehicles inside a platoon except the platoon leader, should follow their predecessor with a safe distance and with a minimal speed difference. The safe distance can be given as a certain time headway T_{head} or space headway s_{head}. The distance between two vehicles consists of the safe distance and a stationary distance s_0, i.e. the distance that is maintained at zero speed. Because the positions of all vehicles are measured at the front of the vehicles, the length of the predecessor should be taken into account when calculating the distance between vehicles (see Figure 4-1).

If a follower is controlling a reference time headway (and not a reference space headway), it is important that the controller does not control the difference between the real time headway and the reference time headway, because the controller would then be very sensitive to low speeds. Instead, the real time headway and reference time headway must first be converted to a space headway. The next example illustrates this.

Vehicle n is following vehicle $n-1$ with a reference time headway of $T_{\mathrm{head},n}$. The real time headway $T^*_{\mathrm{head},n}$ can be computed as follows:

$$T^*_{\mathrm{head},n} = \frac{x_{n-1} - x_n - L_{n-1} - s_0}{v_n}.$$

It is clearly seen that in this way $T^*_{\mathrm{head},n}$ goes to infinity when v_n reaches zero. A better strategy is thus to convert the reference time headway to a reference space headway:

$$s_{\mathrm{head},n} = v_n T_{\mathrm{head},n}, \tag{4-9}$$

and to compare the actual space headway with the reference space headway. So there are two options implemented in FreeSim to define the headway a vehicle must try to achieve. A vehicle can follow its predecessor with a given (fixed) space headway *or* with a given time headway, but in the latter case the time headway is converted to a space headway, which is thus dependent on the current speed.

There are also two strategies for choosing the reference speed v_{ref} for follower vehicle n. One option could be to choose the reference speed equal to the speed of the predecessor, so

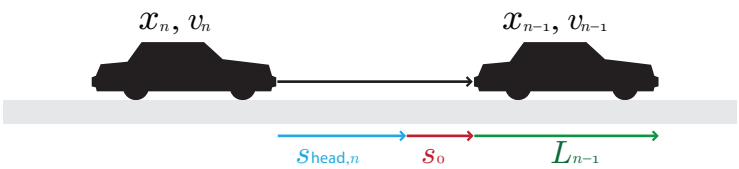


Figure 4-1: $x_{n-1}(k) - x_n(k) = s_{\text{head},n} + s_0 + L_{n-1}$

$v_{\text{ref}} = v_{n-1}$. The other option is to have the reference speed equal to the speed of the platoon leader, so $v_{\text{ref}} = v_{n_\text{p}}$. Choosing to minimize the difference in speed between a following vehicle and its predecessor, can lead to 'errors' that propagate through a platoon, which can lead to instability. This is because the last vehicle in a platoon only reacts to the next-to-last vehicle. It is chosen to minimize the speed of the followers with the speed of the platoon leader, because this should give a more 'rigid' platoon. The algorithm to compute the acceleration of a following vehicle in a platoon is this:

$$a_n(k) = K_\text{x}\Big[s_{\text{ref},n}(k) - (x_{n-1}(k) - x_n(k) - L_{n-1} - s_0)\Big] + K_\text{v}\Big[v_{n_\text{p}}(k) - v_n(k)\Big], \quad (4\text{-}10)$$

where:

$$s_{\text{ref},n}(k) = \begin{cases} v_n(k)\,T_{\text{head},n} & \text{if the safe distance is given as } T_{\text{head}} \\ s_{\text{head},n} & \text{if the safe distance is given as } s_{\text{head}} \end{cases} \quad (4\text{-}11)$$

The acceleration and speed of a follower vehicle n is also limited by (4-6) and (4-8) respectively.

Reaching Destination

If a vehicle reaches its destination it is removed from the memory of FreeSim. If this happens with a vehicle that is not the last vehicle inside a platoon, i.e. it still has followers, its follower cannot compute a new speed anymore. This is because the correct data for this computation is not available anymore, i.e. s_{ref} and v_{ref} is dependent on the vehicle just removed. It is therefore necessary to include a check in FreeSim to make sure that when a platoon leader reaches its destination it gives its leadership to the vehicle right behind it ($n_\text{p} + 1$). It is still possible that in one simulation step the first two vehicles of a platoon reach their destination, but then vehicle $n_\text{p} + 1$ first waits to receive the leadership, after which it gives the leadership to the next vehicle in turn.

4-2-3 Instantiating Platoons

As already mentioned in Section 3-3, it is possible to predefine events that have to occur during a simulation, e.g. instantiating a new vehicle. All these predefined events have to be written in a command text file that is processed by FreeSim before a simulation starts. The command that is put in the text file has to have a special format. For instance, an event to instantiate a new vehicle must be written like this:

```
70|<Vehicle ID>|<Time to Instantiate Vehicle>|<Source Node Name>|<Source
Freeway Name>|<Destination Node Name>|<Destination Freeway Name>
```

Here '70' stands for the *vehicle instantiation* command. All the subsequent data are parameters that are separated with a delimiter '|'. FreeSim was already adjusted such that each vehicle belongs to a platoon, even if the platoon only consists of that one vehicle. So if a single vehicle is instantiated with the previously mentioned command, that vehicle is created as if it were the platoon leader. But in order to instantiate a whole new platoon consisting of multiple vehicles, a new command has to be created.

It is possible to create a new platoon by placing the first vehicle at its source node with $x_{n_\mathrm{p}} = 0$. The vehicle behind the platoon leader should begin with an offset, so $x_{n_\mathrm{p}+1} = -(L_{n-1} + s_0 + s_\mathrm{ref})$. The next vehicle will have an offset twice as big, and so on. If s_ref is given as a space headway, this will give no problem. But if the safe distance is expressed as a time headway, the initial speed has to be know in oder to give the vehicles an appropriate offset. When a new *vehicle* is instantiated, first it is placed at its starting node and subsequently the optimal path is computed. If the path is known, the vehicle starts with an initial speed which is dependent on the first edge it will be traveling along (cf. (4-4)). This first edge is not always known in advance, because it is possible that the source node connects more than one edge. This means that when a new *platoon* is instantiated, the initial speed is not always known in advance, i.e. first the vehicles are placed at a node, and after the path is known the speed is set. The command to create a new platoon should thus contain a parameter for the initial speed in case the platoon starts at a node with more than one edge connected to it. Besides this, three more parameters are needed to successfully instantiate a new platoon, i.e. the number of vehicles in a platoon, whether the vehicles must keep a safe time headway or a safe space headway, and how large the headway must be. It is chosen that the command for instantiating a platoon must look like this:

```
76|<ID Platoon leader>|<Time to Instantiate Platoon>|<Source Node
Name>|<Source Freeway Name>|<Destination Node Name>|<Destination Freeway
Name>|<Number of Vehicles>|<Time Headway or Space Headway>|<Headway>|<Speed>
```

FreeSim recognizes '76' now as the command to create an event for instantiating a new platoon. Only the ID for the platoon leader is needed, because the ID's of following vehicles are automatically set as $n_\mathrm{p}+1, n_\mathrm{p}+2, ..., n_\mathrm{p}+N_\mathrm{p}$. The `<Time Headway or Space Headway>` parameter can either be set to `true` or `false`, which stands for time headway and space headway respectively. The `<Headway>` parameter is expressed in either milliseconds or centimeters. This is because the command file can only contain integers and expressing the headway in seconds or meters can be too coarse. The `<Speed>` is the initial speed, and is thus only used if the platoon is instantiated at a source node that connects multiple edges.

4-3 Reference Speed

FreeSim is modified such that it now can simulate platoons, where the followers are following their predecessor with a safe distance and the platoon leader is trying to travel with a speed equal to the reference speed. Until now, the reference speed for the platoon leader was equal to V_i, where i denotes the edge it is traveling on. This V_i was a speed originally used in FreeSim to compute the travel time all the vehicles needed to travel from the source node to the destination node of the edge. However, this speed is not regulated by FreeSim natively. This means that when the density of an edge is increasing, for instance due to weaving of two edges into one edge, the speed is not affected at all.

The speed limit of an edge can be regulated using control strategies for example a Model Predictive Control (MPC) strategy. Then this (reference) speed is the ISA speed for the platoon leaders, controlled with an MPC controller. However, when the traffic is uncontrolled, so there is no controller to compute an ISA speed, the reference speed should be adjusted properly inside FreeSim, because FreeSim currently does not take care of it. This adjustment must take into account the capacity of an edge, such that the average vehicle speed will decrease when the traffic is becoming congested. Just as the reference speed must be equal to the free-flow speed when the density is low enough that the platoons can travel in free-flow mode. An existing model that incorporates these traffic characteristics is for instance the METANET model. The METANET model uses macroscopic traffic characteristics to compute average speeds (see Section 2-3-5). The computation of the speeds by METANET can be done quite fast as METANET only uses macroscopic quantities. By combining FreeSim and the METANET model, the reference speeds for the platoon leaders are in the uncontrolled case provided by a macroscopic model, however, the vehicles are still simulated individually. The traffic simulation software package called INTEGRATION [36] also bases the vehicle speeds on macroscopic models, while the traffic is simulated microscopically. It is therefore chosen to use the METANET model to adjust the reference speed when the traffic is uncontrolled. Also the *edges* in FreeSim can be used as *segments* in the METANET model, so no extra conversion is necessary. The METANET model, as discussed in Section 2-3-5 is repeated here for convenience:

$$\begin{aligned} v_{m,i}(k+1) &= v_{m,i}(k) + \tfrac{T}{\tau}\Big(V(\rho_{m,i}(k)) - v_{m,i}(k)\Big) + \\ &\quad \frac{T}{L_m} v_{m,i}(k)\Big(v_{m,i-1}(k) - v_{m,i}(k)\Big) - \\ &\quad \frac{\eta T}{\tau L_m}\frac{\rho_{m,i+1}(k) - \rho_{m,i}(k)}{\rho_{m,i}(k) + \kappa}, \end{aligned} \tag{4-12}$$

where:

$$V(\rho_{m,i}) = v_{\text{free},m} \exp\left[-\frac{1}{a_m}\left(\frac{\rho_{m,i}(k)}{\rho_{\text{crit},m}}\right)^{a_m}\right]. \tag{4-13}$$

To let the METANET model work, the density of each segment (edge) is needed, which is not yet provided by FreeSim. It is possible to search in the MySQL database for vehicles traveling on a certain edge and from this data the density can then be computed. But this is computationally very demanding, since this has to be done (for each edge) every time the reference speed is updated. Because no algorithm in FreeSim exists to read the MySQL database (only to write), and because more important it can be computationally very demanding to compute the density of an edge by reading the database, this method is not preferred. An easier method is however possible to accomplish this. In the edge object a local variable ξ is added that contains the number of vehicles on the edge. So each time a vehicle enters an edge, it tells the edge to increment ξ with one. Also when a vehicle leaves an edge it decrements ξ by one. This way the number of vehicles on an edge does not have to be computed whenever it is needed. So each time the METANET model needs the density

on an edge i it can request it from the edge. The density is computed by the edge as follows:

$$\rho_i = \frac{\xi_i}{D_i}. \tag{4-14}$$

It must be noted that the METANET model uses L_m as the length of segment i in link m, and the length of an edge i is denoted by D_i. By using the edges as segments in the METANET model these two variables are in fact the same, so $L_m = D_i$.

The new reference speed $v_{m,i}(k+1)$ computed by the METANET model, is stored as V_i in edge i. To allow the controlling of the edge speed by an external controller, a new variable v_{ISA} in the edge object is created. So each edge can thus have two reference speeds associated with it: V_i and $v_{\text{ISA},i}$. If a platoon leader is requesting from an edge its reference speed v_{ref}, the edge returns the minimum of the two speeds. Thus:

$$v_{\text{ref}} = \min\left(V_i, v_{\text{ISA},i}\right). \tag{4-15}$$

4-4 Summary

In this chapter the modifications are discussed needed in order to use FreeSim as a simulator that can model platoons. The original FreeSim simulates vehicles by computing the travel time each vehicle needs to travel the length of an edge, and when that time is passed then the vehicle crosses a node and a new time is computed. In this way the vehicle speed was fixed along the whole edge. FreeSim is adjusted in such a way that each vehicle has its own (variable) speed and position, which are updated every simulation step. Also a platoon object is created in FreeSim, so now each vehicle is part of a platoon. The platoon leader follows a reference speed that is provided by the edge. All the followers in a platoon try to maintain a safe space headway while trying to travel with the same speed as the platoon leader. When a platoon leader reaches its destination the leadership is transferred to the vehicle behind it. This is done until all vehicles in the platoon have arrived at their destination. A new command is also created such that it is possible to instantiate a new platoon at a preset time.

Because FreeSim natively does not provide a realistic reference speed for the platoon leaders, the METANET model is implemented in FreeSim to compute a reference speed for each edge if the traffic is uncontrolled. This speed is dependent on the density of the edge, which is known by the edge because each edge keeps track of the number of vehicles that have entered the edge. If the traffic is controlled, the minimum of the controlled (ISA) speed and the METANET speed is used as the reference speed for the platoon leaders.

Chapter 5

Case Study

As discussed in Chapter 4, FreeSim is transformed such that it is able to simulate traffic that consists of platoons. However, using FreeSim in Model Predictive Control (MPC) is only feasible if a prediction model is used that is fast while still being accurate enough. FreeSim is not suitable to use as a prediction model as it is not fast enough. The Big Car model (see Section 2-3-4) is a faster model because it models a platoon as one object. In this chapter a case study is discussed in which an MPC scheme is presented where FreeSim acts as the system model and the Big Car model is used as a prediction model.

5-1 Description

In order to use the Big Car model as a prediction model in an MPC strategy, it should have a similar behavior as the system. In this case the system is presented by FreeSim. As already described in the previous chapter, in FreeSim the reference speed for the platoon leaders in uncontrolled traffic is computed by the METANET model. In this way the speed of the platoon leaders is dependent on the traffic density.

In the Big Car model the following behavior is described by an adapted version of a follower model. FreeSim and the Big Car model have a different set of parameters because of the modeling differences. The parameters of the Big Car model should be chosen well, such that it models the system accurately. For this calibration experiments need to be done in order to find the optimal set of parameters such that the Big Car model can be used as a prediction model (see Figure 5-1).

5-2 Calibration Experiments

In order to be able to calibrate the Big Car parameters, data must be gathered with FreeSim that can be used as a reference in the calibration process. To gather data, a simulation setup is designed, such that a wide variety of platoons states are measured. It is chosen to create

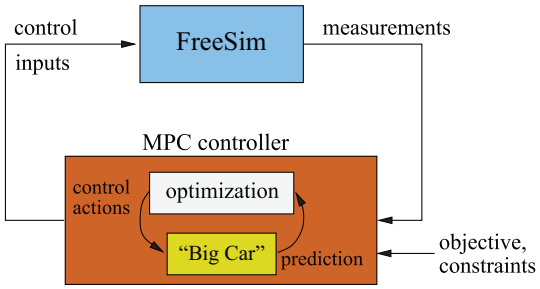


Figure 5-1: Schematic representation of a MPC structure that has FreeSim as the system model and the Big Car model as a prediction model.

one stretch of freeway in FreeSim that consists out of six connected edges, each containing one lane of 1 km. At the origin of the first edge platoons are instantiated at different time intervals. The traffic is uncontrolled, such that the METANET model has to compute the reference speeds for the platoon leaders. The platoons are instantiated in such a way that the traffic will become congested, but such that also free-flowing traffic is included. A Matlab function is created that can read the MySQL database and that stores all the data in an array that can be used for the calibration process that is done in Matlab.

5-2-1 FreeSim Parameters

In this section all the parameters used in FreeSim are chosen. First the controller gains K_l, K_v, and K_x are tuned, such that the platoon leaders follow the reference speed while the followers can follow their predecessors accurately. Secondly, the METANET parameters are chosen in order to be able to compute a reference speed that is based on the density of traffic that consists out of platoons.

Acceleration Boundaries

As explained in the earlier chapters the acceleration of the platoon leader is limited by a^+ and a^-, which are the maximum and minimum acceleration respectively. The maximum acceleration a^+ is set to $3\,\mathrm{m/s^2}$. This means that a vehicle can accelerate from 0 to 100 km/h in approximately 9 s. The maximum deceleration a^- of the vehicles is set to $-7\,\mathrm{m/s^2}$, which is about the breaking deceleration of an average vehicle on dry asphalt [27]. This means that a vehicle that drives with a speed of 100 km/h can brake to a full stop in about 55 m. This takes about 4 s.

Time Headway and Minimal Safe Distance

The vehicles in a platoon can follow their predecessor with a very small space headway. As already explained in Section 4-2-2, the minimum safe distance between two vehicles is s_0. This minimal safe distance is to ensure that there is always a distance between two vehicles even when the speed is zero. This safe distance is set to 1 meter, so $s_0 = 1\,\mathrm{m}$. The headway can either be a fixed space headway or a time headway, i.e. a speed dependent space headway. In

FreeSim either reference headway can be chosen, but in this case study it chosen to use only a reference time headway. This time headway can be set as small as 0.5 s with the current technologies [8]. So from now on the time headway is set to $T_{\text{head}} = 0.5\,\text{s}$.

Controller Gains

Since now a^+, a^-, s_0, and T_{head} have been set, the controller gains K_{l}, K_{x}, and K_{v} can be chosen. It is very important that the controllers are tuned in such a way that there is little to no overshoot. This is because when a follower has to accelerate to decrease the distance between itself and its predecessor, it also has to decelerate in time to end up with the appropriate distance and speed. If the vehicle does not decelerates in time, the distance between the vehicles will become smaller than the reference space headway. This does not necessarily have to be a bad thing, but it can also happen that the distance is smaller than s_0. And this has to be avoided at all times, because s_0 is the safe distance that has to be maintained in all situations.

A good set of parameters is $K_{\text{l}} = 0.04$, $K_{\text{v}} = 0.3$, and $K_{\text{x}} = 0.01$. This is shown in Figure 5-5. Here a trajectory for a reference speed v_{ISA} is chosen such that it starts at 120 km/h, then it drops to 5 km/h and subsequently it is set to 80 km/h. A vehicle ($n = 1$), which acts as a platoon leader, is following this reference speed, using $K_{\text{l}} = 0.04$. Another vehicle ($n = 2$) is following vehicle 1 using the controller gains $K_{\text{v}} = 0.3$ and $K_{\text{x}} = 0.01$. In Figure 5-5(a) the speeds of vehicles 1 and 2 are plotted as well as the reference speed v_{ISA}. In this figure it is seen that both vehicles follow the reference speed quite well. Figures 5-5(b) and 5-5(d) display the error of the speed of vehicle 1 and 2 respectively. It can be seen that the platoon leader adjust its speed smoothly to the reference speed.

Vehicle 2 has a little overshoot in its speed error, but this is due to the fact that vehicle 2 is also controlling its headway. Controlling the headway should have a higher priority, because an overshoot at the headway can result in a collision. In Figure 5-5(c) the positions of both vehicles are plotted. In this figure it can be difficult to see that vehicle 2 follows vehicle 1 with an appropriate headway. Figure 5-5(e) displays this better, because in this figure the reference space headway $s_{\text{ref},2}$ and the actual space headway $s_{\text{head},2}$ are plotted. This figure can be used to check whether vehicle 2 does at some point has a smaller space headway than s_0. If the actual space headway is zero, then the distance between the front bumper of vehicle 2 and the rear bumper of vehicle 1 is exactly s_0. This means that when the actual space headway is negative ($s_{\text{head}} < 0$), the distance is smaller than the minimum space headway. More important is to ensure that $s_{\text{head}} > -s_0$, because otherwise the two vehicles collide with each other. The difference between $s_{\text{ref},2}$ and $s_{\text{head},2}$ is plotted in Figure 5-5(f). This figure shows a smooth curve and no overshoot. So with the chosen controller gains the vehicles are well controlled and at all times the space headway is at least the minimal safe space headway.

It is possible to increase K_{l}, such that the platoon leader follows the reference speed even more accurately, but then the following vehicle has trouble to control the headway. This can be seen in Figure 5-6. Here K_{l} is increased to 0.2. Vehicle 2 has the same controller gains as in the previous example. Clearly it can be seen that vehicle 2 cannot follow vehicle 1 nicely, even more, a collision occurs at 18 s, as $s_{\text{head},2}$ is smaller than s_0 (see Figure 5-6(e)). It is tried to tune the controller gains such that vehicle 2 follows vehicle 1 better when K_{l} is chosen high, but vehicle 2 will almost always react too "nervously" and a collision seems

almost inevitable. Setting the controller gains as $K_l = 0.04$, $K_v = 0.3$, and $K_x = 0.01$ seems to be the best trade-off, as the platoon leader will follow the reference speed quite well and the following vehicle follows its predecessor smoothly without crossing the minimum safe space headway.

METANET Parameters

The METANET model is used to compute the reference speed for the platoon leaders in uncontrolled traffic. METANET was developed to model human traffic in a macroscopic way. METANET uses the fundamental diagram (4-13) and takes into account the inflow of vehicles and the perceived downstream density. The parameters used by the METANET model are normally calibrated by using existing traffic data of a certain freeway. In this way the parameterized METANET model can be used to model that specific freeway. In this case study a fictive freeway is used, so the METANET parameters cannot be calibrated, but they merely have to be chosen. First the parameters $v_{\text{free},m}$, a_m, and $\rho_{\text{crit},m}$ used for the fundamental diagram are chosen. Then the other parameters τ, κ, and η are set.

In Figure 2-2 on page 7 a typical fundamental diagram is plotted. The values of the parameters used in this diagram are $v_{\text{free},m} = 102\,\text{km/h}$, $a_m = 1.867$, and $\rho_{\text{crit},m} = 33.5\,\text{veh/km/lane}$, which are typical values [17]. However, as discussed earlier, when using the platooning concept it is possible to accommodate more vehicles on the freeway. The average speed at high densities can thus be higher when the traffic consists out of platoons than when the traffic has human drivers. As no data is available of an edge where platoons have been traveling along, no fundamental diagram exist to calibrate such data. To make an educated guess as what the parameters should be when an edge is accommodated with platoons, the average speed of an edge is estimated.

To estimate the average speed v_m on an edge that is used in the case study, it is assumed that platoons are equally distributed along the edge (see Figure 5-2). It is also assumed that platoons travel with a constant speed. The length L_m of the edge in the case study is 1 km and this edge has one lane. All vehicles in the case study have the same length L_v= 4 m. Because $L_m = 1\,\text{km}$ the average density ρ_m is equal to the number of vehicles on the edge. The intra platoon distance is s_{head} and the distance between platoons is γ_{head}. The vehicles on the edge should travel with a speed such that s_{head} and γ_{head} is appropriate to the traffic density ρ_m. So the sum of the appropriate space between vehicles and platoons is equal to the length of the edge minus the occupied space by the vehicles:

$$L_m - \sum L_v = \sum s_{\text{head}} + \sum \gamma_{\text{head}} \tag{5-1}$$

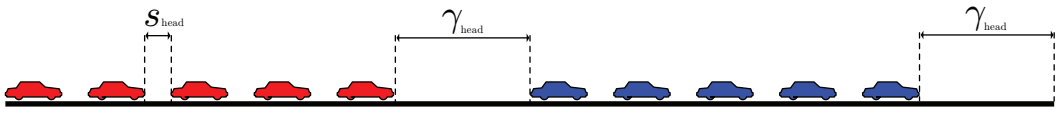


Figure 5-2: Edge with platoons traveling

Since in the case study all platoons consists out of 5 vehicles the number of platoons (ϱ) can

be computed as follows:

$$\varrho = \left\lceil \frac{\rho_m}{5} \right\rceil$$

The ceiling of $\frac{\rho_m}{5}$ is taken, because if there are for instance 6 vehicles on the edge, the number of platoons must be taken as 2, because the sixth vehicle belongs to the second platoon. Now (5-1) can be written as follows:

$$L_m - L_{\mathrm{v}}\rho_m = s_{\mathrm{head}}\left(\rho_m - \varrho\right) + \gamma_{\mathrm{head}}\varrho, \tag{5-2}$$

where:

$$s_{\mathrm{head}} = s_0 + T_{\mathrm{head,v}} v_m \tag{5-3}$$

$$\gamma_{\mathrm{head}} = s_1 + T_{\mathrm{head,p}} v_m \tag{5-4}$$

Here $s_0 = 1\,\mathrm{m}$ and $T_{\mathrm{head,v}} = 0.5\,\mathrm{s}$. The minimum safe distance between platoons is s_1 and the time headway for the platoon leader is $T_{\mathrm{head,p}}$. The values for these platoon parameters are chosen as $s_1 = 20\,\mathrm{m}$ and $T_{\mathrm{head,p}} = 1.2\,\mathrm{s}$, which are common values used for platooning. If s_{head} and γ_{head} are substituted in (5-2), the equation is as follows:

$$L_m - L_{\mathrm{v}}\rho_m = \left(s_0 + T_{\mathrm{head,v}} v_m\right)\left(\rho_m - \varrho\right) + \left(s_1 + T_{\mathrm{head,p}} v_m\right)\varrho \tag{5-5}$$

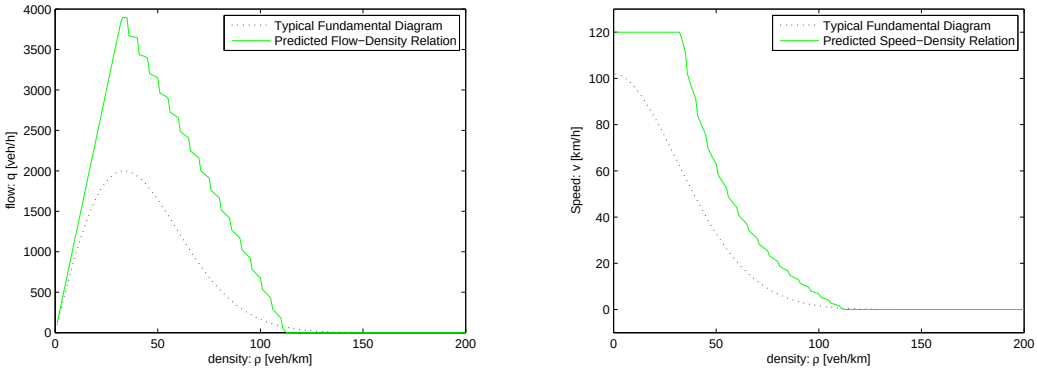
If ρ_m is given, then the only unknown in (5-5) is the speed of the vehicles v_m. This v_m can thus be written as a function of ρ_m, just as the fundamental diagram. So:

$$v_m = \frac{L_m - L_{\mathrm{v}}\rho_m - s_0\left(\rho_m - \varrho\right) - s_1\varrho}{T_{\mathrm{head,v}}\left(\rho_m - \varrho\right) + T_{\mathrm{head,p}}\varrho} \tag{5-6a}$$

$$v_m = \max\left(0, \min\left(120, v_m\right)\right) \tag{5-6b}$$

Of course v_m cannot have any value, so v_m is constrained such that $0 \leq v_m \leq 120\,\mathrm{km/h}$ (cf. (5-6b)). If (5-6) is plotted, it can be seen that the curve will have a triangular shape (see Figure 5-3). This is because the Intelligent Vehicles (IV) arranged in platoons can maintain a free-flow speed at higher densities than human drivers will. In this way the capacity flow will become higher as expected. When the density becomes higher than a certain critical density, the speed will decrease gradually. The little "bumps" in the right half side of the curve are due to the fact that (5-6) accounts for platoons as well as individual vehicles, which have different time headways.

Because the METANET model uses (2-11) and not (5-6), the values for $v_{\mathrm{free},m}$, a_m, and $\rho_{\mathrm{crit},m}$ have to be estimated, such that the estimated fundamental diagram fits the predicted speed-flow relation as good as possible. It is important to note that (5-6) is a function where it is assumed that the vehicles are equally distributed and have a constant speed, so the function can give a too optimistic result. The estimated fundamental diagram should thus fit "inside" the predicted curve. The optimal values for the parameters which give the best estimation are $v_{\mathrm{free},m} = 120\,\mathrm{km/h}$, $a_m = 2.63$, and $\rho_{\mathrm{crit},m} = 40\,\mathrm{veh/km}$. These values are optimized in Matlab, by minimizing the difference between the two curves, where a penalty is given when points of the estimated curve are "outside" the predicted curve. The estimated fundamental diagram is plotted in Figure 5-4. In this figure it can be seen that by using the estimated values, the capacity flow and the critical density are increased, while the estimated curve is still "inside" the predicted curve.



(a) Predicted flow-density relation based on platoons (5-6).

(b) Predicted speed-density relation based on platoons (5-6).

Figure 5-3: The predicated flow-density relationship (5-6) shows a more triangular curve with a higher capacity flow than the typical fundamental diagram.

Now the parameters for the fundamental diagram that are used in the METANET model are estimated, the rest of the parameters, i.e. τ, κ, and η, have to be set. Note that these values also cannot be calibrated as fictive edges are used in the case study. The values are chosen such that they have the same order of magnitude as found in the literature [30]: $\tau = 0.0018\,\text{h}$, $\kappa = 40\,\text{veh/km/lane}$, and $\eta = 20\ \text{km}^2/\text{lane}$. All the parameters that will be used by FreeSim in the case study are listed in Table 5-1.

(a) Vehicle Parameters

Parameter	L_v	T_head	s_0	K_l	K_v	K_x	a^+	a^-
Value	4 m	0.5 s	1 m	0.04	0.3	0.01	3 m/s^2	-7 m/s^2

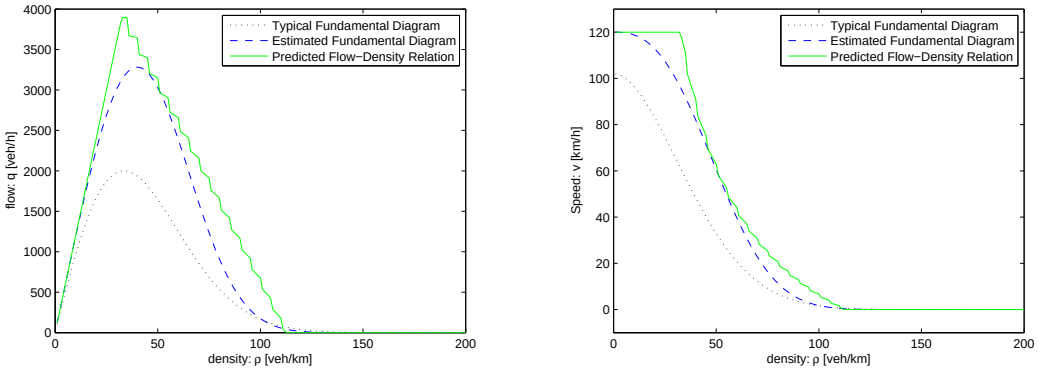
(b) METANET Parameters

Parameter	$v_{\text{free},m}$	a_m	$\rho_{\text{crit},m}$	τ	κ	η
Value	120 km/h	2.63	40 veh/km/lane	0.0018 h	40 veh/km/lane	20 km^2/lane

Table 5-1: All the parameters used in FreeSim

5-2-2 Big Car Setup

It is chosen to implement the Big Car model in Matlab, such that it can be used as a prediction model in an MPC. The Matlab simulation should behave similar as FreeSim, although it uses a different model. The Big Car function in Matlab uses the same Freeway layout (graph) as FreeSim, so all edges and nodes are named the same, and have the same lengths. This graph is saved in Matlab as a structure array containing all the edges with fields that describe each edge as well as all current traffic that travels on the edge. Also all the platoons (in fact Big Cars) that are simulated in the Big Car function are saved in a structure array. Each field in



(a) Estimated fundamental diagram.

(b) Estimated speed-density relation.

Figure 5-4: The estimated flow-density relationship ($v_{\text{free},m} = 120\,\text{km/h}$, $a_m = 2.63$, and $\rho_{\text{crit},m} = 40\,\text{veh/km}$) fits between the predicted speed-flow relationship and the typical fundamental diagram.

this vehicle structure array represents a platoon, and contains all the data of that platoon, i.e. the length, position, and speed at any given time.

The Big Car function in Matlab is written such that it is able to interpret, at the beginning of the simulation, the same command text file as used by the FreeSim simulation. After the command text file has been processed, an array of events is created. This array is used at each time step to look whether a new platoon has to be instantiated. In FreeSim an internal timer is used to give a command at a given interval to process the next iteration step. The Big Car function in Matlab however, uses a continuous loop to process each time step. In each loop, first all the states of the vehicles are updated, then the event array is checked whether new platoons have to be instantiated, and at last a check is performed to see whether the simulation can be stopped. The stopping criteria depend on whether all events are processed and whether there is any traffic still present.

If the traffic is uncontrolled, the behavior of the platoons in the Big Car function is modeled by an adapted version of the follower model (2-8):

$$\begin{aligned} \alpha_n(k) &= K_{\text{x,BG}}\Big[v_n(k)\,T_{\text{BG}} - \Big(x_n(k) - x_{n-1}(k-\sigma) - L_{n-1}(k-\sigma) - s_{\text{BG}}\Big)\Big] + \\ &+ K_{\text{v,BG}}\Big[v_{n-1}(k-\sigma) - v_n(k)\Big]. \end{aligned} \tag{5-7}$$

Here n refers to the index of the platoon under consideration, and $n-1$ is then the index of the preceding platoon. The position of platoon $n-1$ is taken at time step $(k-\sigma)$, where σ is a time delay. The length of the platoon n at time step k is computed by $L_n(k) = \alpha v_n(k) + \beta$ (cf. 2-9). Also in the Big Car model a fixed minimum space headway is used s_{BG}, as well as a time headway T_{BG}. The controller gains are $K_{\text{x,BG}}$ and $K_{\text{v,BG}}$. Also the acceleration of the platoon is bounded by (4-6). To ensure that platoons can travel in free flow when their headway allows it, the platoon *speed* is also bounded. In this way the platoons will travel with the free-flow speed when they have an unobstructed headway, i.e. the controller computes a positive acceleration, but the platoons cannot drive faster than the free-flow speed due to the bounded speed. A platoon will however decelerate when its headway is too small or

when another platoon ahead is traveling too slow. This is because (5-7) computes a negative acceleration in these cases.

Big Car Parameters
α
β
$K_{\mathrm{v,BG}}$
$K_{\mathrm{x,BG}}$
T_{BG}
s_{BG}
σ

Table 5-2: The parameters of the Big Car model that can be calibrated.

5-2-3 Optimization Setup

To ensure that the Big Car following model has a similar behavior as FreeSim, the parameters that can be varied in the Big Car model (see Table 5-2) have to be calibrated. In order to do so, data is needed from FreeSim. Therefore a case study is created to gather data from FreeSim that can be used for calibrating the Big Car model in Matlab.

It is possible to calibrate all the parameters at once, but this can be time consuming. Therefore the calibration is done in multiple steps. The first step is to find the optimal α and β. In this way the length of the platoons will be similar to the length of the platoons in FreeSim. This length influences the computation of the time headway in the Big Car model, since the space headway is directly related to the length and position of a preceding platoon. The second step is to calibrate the parameters of the controller, i.e. $K_{\mathrm{x,BG}}$, $K_{\mathrm{v,BG}}$, T_{BG}, and s_{BG}. The integer σ is varied manually at the end of the second step to reduce the number of parameters at the calibration process.

Optimizing Step 1

In order to find the optimal α and β, simulations have been done with FreeSim to provide data of vehicles traveling in free flow as well as in congestion. The data also contains the states of accelerating and decelerating vehicles. In this way a broad set of data points is created that can be used to calibrate α and β. To calibrate these parameters, an optimization has been done in Matlab that solves this objective function:

$$\min_{\alpha,\beta} \sum_{i=1}^{m} \left[L_i - (\alpha v_i + \beta)\right]^2, \tag{5-8}$$

where m is the length of the dataset. To solve this objective function, the function `lsqlin` can be used in Matlab. This function solves linear least squares data fitting problems. However, because this optimization is unconstrained it is also possible to use the `slash` command in Matlab. In fact, if `lsqlin` is used without constraints then Matlab uses the `slash` command.

So this optimization can thus be done in Matlab by using:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} v_1 & 1 \\ v_2 & 1 \\ \vdots & \vdots \\ v_i & 1 \end{bmatrix} \setminus \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_i \end{bmatrix} \tag{5-9}$$

The input for `lsqlin` are vectors containing the data of the speed of the platoon leaders and the corresponding lengths of the platoons. The results of this optimization are discussed in Section 5-3.

Optimizing Step 2

To calibrate the other parameters of the Big Car model, a dataset is taken from FreeSim where the traffic was uncontrolled such that the METANET model was used to compute reference speeds. From this dataset all the positions and speeds of the platoon leaders are taken and used for calibration. The calibration consists of optimizing the following objective function:

$$\min_{K_{\mathrm{x,BG}},K_{\mathrm{v,BG}},T_{\mathrm{BG}},s_{\mathrm{BG}}} \sum_{i=1}^{m} [x_i - \hat{x}_i]^2 + \lambda \sum_{i=1}^{m} [v_i - \hat{v}_i]^2 , \tag{5-10}$$

where m is the length of the dataset and λ is a scaling term used to ensure that the position errors are in the same order of magnitude as the speed errors. The positions and speeds of the platoon leaders in the dataset are x and v respectively. The positions and speeds of the Big Cars are $\hat{x}$ and $\hat{v}$.

Because (5-10) is a nonlinear objective function the method `lsqnonlin` is used in Matlab. This function can solve a nonlinear least-squares data fitting problem, such as (5-10). The optimization is set to medium-scale since the number of variables is much less than the number of provided data points. The algorithm used for optimizing the parameters is Levenberg-Marquardt. Levenberg-Marquardt is preferred over Gauss-Newton as this will often be faster when the residual of the objective function is not small.

Because the two models (i.e. FreeSim and the Big Car model) do not necessarily have to coincide, the position and speed vectors of the two models do not always have the same lengths. This is because a vehicle n can for instance have a higher average speed in the FreeSim simulation than it had in the Big Car simulation, thus reaching its destination earlier in FreeSim. Because (5-10) can only be used when all vectors have the same lengths, it is chosen to use the constant length of the FreeSim data. This means that the vectors of a Big Car simulation, which can thus vary, have to be adjusted. When a vector of the Big Car simulation is shorter than in FreeSim, it has to be extended. A position vector $\hat{x}$ has the same initial value as x, since a vehicle starts in both simulations at the same point. The extension has thus to be added at the end of vector $\hat{x}$. It makes sense to add the destination position

to the extended area. To give an illustrative example where two extra entries are added:

$$\hat{x} = \begin{bmatrix} 0 \\ 0.1 \\ 0.2 \\ \vdots \\ 0.9 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 0.1 \\ 0.2 \\ \vdots \\ 0.9 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

The same has to be done with the speed vector $\hat{v}$, thus adding the speed when the vehicle reached its destination to the extra entries. When a vector is shorter in the Big Car simulation than in the FreeSim simulation, the vector has to be shortened. Adding entries to the FreeSim vectors will result in a larger output in the objective function, since more data is used, so this must be omitted. It is chosen to delete the last entries in $\hat{x}$ and $\hat{v}$, since again the first values do coincide with the FreeSim data and have to be left intact. In this way the output of the objective function will not be dependent on different lengths of vectors of the Big Car simulations. The results of the calibration of the controller parameters are discussed in Section 5-3.

5-3 Results

5-3-1 Results of Optimization Step 1

The optimization for finding α and β is quite straightforward. There are only two parameters that have to be optimized, and these two parameters describe a linear behavior between the speed and length of a platoon. Optimizing α and β is done by Matlab in a few seconds. The optimal values that have been found are $\alpha = 0.5536$ and $\beta = 24.0186$ (see Table 5-3). The function $L = \alpha v + \beta$ is plotted in Figure 5-7 as well as all the data points gathered from FreeSim. In this figure it can be seen that the average error between the length of a big car and of a platoon in FreeSim is slightly more than 1 %. However, the highest measured error is 20 %, which is due to the deceleration of a platoon. The highest measured error due to acceleration is 7 %.

If the platoons would not have accelerated or decelerated throughout the simulation in FreeSim, the platoon lengths would have been constant. The "steady state" length can also be calculated by using (2-9). If the values of the parameters (see Table 5-1) are used in (2-9), the equation would be:

$$\begin{aligned} L_{\text{BigCar}}(k) &= (N_{\text{p}}-1)\left(s_0 + s_1 v(k)\right) + \sum_{n=1}^{N_{\text{p}}} L_n \\ &= (5-1)\left(1 + 0.5\frac{v(k)}{3.6}\right) + 5\cdot 4 \\ &= \frac{0.5}{3.6}v(k) + 24 \\ &\approx 0.5556 v(k) + 24 \end{aligned} \tag{5-11}$$

As can be seen, the optimized values are almost the same as the steady state values. However, the calibrated values for α and β are used in the optimization of step 2.

Calibrated Values Set 1	
α	0.5536
β	24.0186

Table 5-3: Optimal values found after optimizing (5-8)

5-3-2 Results of Optimization Step 2

In step 2 the controller gains ($K_{\text{x,BG}}$ and $K_{\text{v,BG}}$), the minimum safe space headway (s_{BG}), as well as the reference time headway (T_{BG}) are optimized. The values for α and β that are used in this optimization step are the values as found in Section 5-3-1 (see Table 5-3). The time delay σ is kept constant throughout optimization step 2 to reduce the number of parameters. The integer value of σ is varied at the end of the optimization step and the σ that gives the best result is then taken. For now σ is set to 0.

In (5-10) the scaling term λ is used to set the speed errors in the same order of magnitude as the position errors. The error in speeds are roughly 50 times higher than the error in the positions, because the position error is measured in km and the speed error in km/h. So λ is set to $\left(\frac{1}{50}\right)^2 = 0.0004$.

The optimal parameters found during an optimization run are dependent on the starting points. It is therefore necessary to use multi-start optimization in this step. This means that various starting points are chosen from which Matlab starts searching for parameters such that (5-10) is minimized. This multi-start optimization approach is very time consuming, since each optimization run can take several hours to a day. After each optimization run the weighted Root Mean Squared Error (RMSE) is computed:

$$\Omega = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left[x_i - \hat{x}_i\right]^2} + \sqrt{\frac{\lambda}{m}\sum_{i=1}^{m}\left[v_i - \hat{v}_i\right]^2} \tag{5-12}$$

The weighted RMSE is used to indicate the performance of the optimization run. The weighted RMSE is used instead of the output of the objective function (5-10), because the

outcome of the objective function is a very large number due to the large dataset, and Ω shows a smaller performance index that is easier to interpret. So after each optimization run Ω_{new} is computed and compared against the smallest $\hat{\Omega}$ to check whether the new parameters are better. If $\Omega_{\text{new}} > \hat{\Omega}$ then the new parameter set is discarded. If however $\Omega_{\text{new}} < \hat{\Omega}$ then the new parameter set is saved as the new optimal data set, so $\hat{\Omega} = \Omega_{\text{new}}$. The new $\hat{\Omega}$ with the corresponding parameter set is used for the next comparison. It is too cumbersome to show all the chosen starting points with the parameters found during the optimization runs including their weighted RMSE. Therefore only the interesting results will be discussed.

What was seen was that the optimal values found for $K_{\text{v,BG}}$ were roughly between 0.01 and 0.09, even if the starting values were chosen much higher or lower than this range. This could be expected since the best value for K_{l} found in Section 5-2-1 is also in the same order of magnitude. The controller gains K_{l}, and $K_{\text{v,BG}}$ are both parameters that are used for minimizing the difference between speeds, so it is not strange that the values are comparable.

The optimal values for $K_{\text{x,BG}}$, however, were mostly around the starting values but could also be very small (almost zero) or as large as 1. The optimized values for s_{BG} and T_{BG} were always more or less the same as the starting points. To investigate what the reason could be that the optimal values for these parameters are around the starting points, weighted RMSE are computed by varying only one parameter and keeping the rest of the parameters constant. With this 'slicing' technique more insight is given how the objective function looks like around a certain point. During this slicing now also σ is changed to see what the influence is of σ. The point at which the parameters are sliced, is the point that has the lowest Ω found. The parameters that correspond to this point are listed in Table 5-4.

Parameter	$K_{\text{v,BG}}$	$K_{\text{x,BG}}$	s_{BG}	T_{BG}	σ
Value	0.0209	0.3033	10.1872	1.6003	0
			m	s	

Table 5-4: Optimized parameters that correspond to $\Omega = 1.1639$

In Figure 5-8 the weighted RMSE is plotted when $K_{\text{v,BG}}$, $K_{\text{x,BG}}$, s_{BG}, and T_{BG} are varied independently. In Figure 5-8(a) $K_{\text{v,BG}}$ is varied between 0.01 and 0.06. Clearly it can be seen that the error increases when $K_{\text{v,BG}}$ is very small or when $K_{\text{v,BG}}$ is becoming too large. More interesting to see is Figure 5-8(b). In this figure it can be seen that when $K_{\text{x,BG}}$ is getting larger than 1.5, the error is not affected at all. This is due to the fact that at very high gains the headway is controlled (too) rigidly, and the error will thus be only depend on the chosen headway. In Figure 5-8(c) it can be seen that s_{BG} does not affect the error very much for values lower than 10. This is due that s_{BG} is a constant term that is not dependent on the speed or position of the big cars. The minimal safe distance s_{BG} will thus affect the error less than the controller gains, which control how the distance is kept, and less than the time headway, which states how large the speed dependent distance must be. As can be seen in Figure 5-8(d), T_{BG} must not be chosen too small or large, because then the error will be too large. The effect of the time delay σ can clearly be seen in Figure 5-8(e). In this plot the error will increases sharply when increasing values for σ are chosen. This indicates that the time delay should be chosen zero. It must be noted however, that when σ is taken into account in the optimization, the other parameters are dependent on this, so a different result could follow.

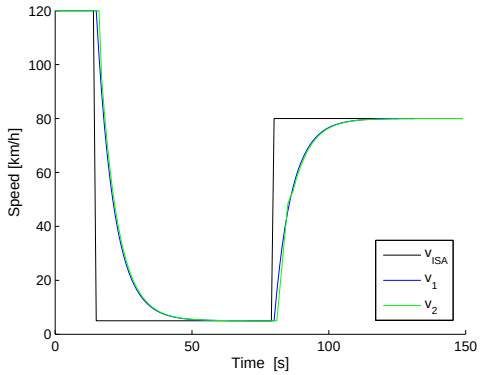
It was seen that during all the optimizations, the optimized values for $K_{x,BG}$, s_{BG} and T_{BG} were more or less the same as the starting values. For $K_{x,BG}$ and s_{BG} this could be explained with slicing, because Figure 5-8(b)–(c) show quite flat curves. However, Figure 5-8(d) does only show a flat curve between 1.4 s and 1.6 s. The reason is probably due to the modeling differences between FreeSim and the Big Car model. The speed of the platoons in FreeSim are dependent on the average density of an edge, whereas the speed of a big car is dependent on the preceding big car. Of course the speed of the preceding big car is in turn dependent on his predecessor. So all big cars influence each other, but in another way than the platoons in FreeSim, i.e. the speed of a big car is only dependent on the big cars ahead and in FreeSim the speed of the platoons are dependent on all platoons on the edge plus to a lesser extend on the platoons on the surrounding edges. Because the speed of a platoon in FreeSim reacts to macroscopic quantities, the reaction will be somewhat slower than the reaction of big cars that react more quickly to density changes. This indicates that speed adjustments based on macroscopic quantities can be valid, but comparing the platoons in FreeSim with big cars on a microscopic level results in errors due to modeling differences. It could be that the average densities, average speeds and traffic flow in the Big Car model is modeled the same as in FreeSim, but on a microscopic level the exact positions and speeds at each time step of the modeled platoons does not necessarily have to coincide. Because the big cars in the Big Car model react more quickly to density changes than the platoons in FreeSim do, it is possible that too optimistic travel times are measured in the Big Car model compared to FreeSim simulations.

5-4 Summary

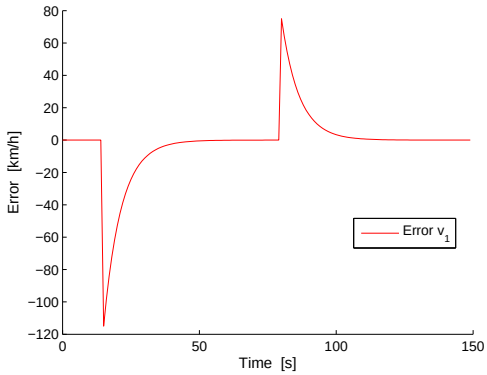
In this chapter a case study is performed where data is gathered from FreeSim that is used to calibrate the Big Car model. In this case study a stretch of road of 6 km long is simulated. First all the parameters used in FreeSim are chosen. The controller gains, i.e. K_l, K_x, and K_v, are tuned in such a way that the platoon leaders can follow a reference speed accurately while their followers are able to follow their predecessors smoothly. Also the METANET parameters have been chosen. These METANET parameters cannot be calibrated since no actual data of traffic that consists of IV is available. The METANET model uses a fundamental diagram in the relaxation term. The parameters of this fundamental diagram are estimated by assuming a constant number of vehicles in a platoon as well as assuming equal distributions of vehicles and platoons on the edges. In this way the fundamental diagram displays a curve with an increased capacity flow.

The parameters of the Big Car model are optimized in multiple steps. In the first step are α and β are calibrated. This can be done quite fast as these parameters describe a linear behavior of the length of a big car. The calibrated values are almost the same as the steady state values, i.e. when the platoons do not accelerate or decelerate. In the second optimization step the controller parameters, i.e. $K_{v,BG}$ and $K_{x,BG}$, as well as the headway parameters, i.e. s_{BG} and T_{BG}, are calibrated using a multi-start nonlinear least squares optimization. The optimal values found are $K_{v,BG} = 0.02$, $K_{x,BG} = 0.3$, $s_{BG} = 10.2$ m, and $T_{BG} = 1.6$ s. After this optimization step, σ was varied to investigate the influence of σ on the performance. It was seen that σ deteriorates the error when σ is increased. It is however possible that when σ is taken into account in the optimization that different optimal parameters will follow, which

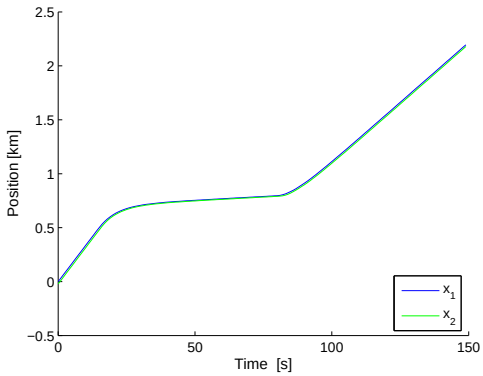
can have an even smaller Ω.



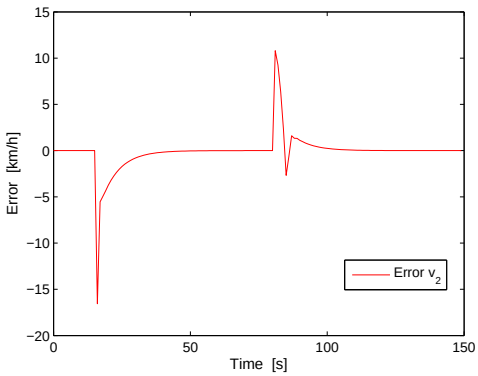
(a) The trajectory of the reference speed v_{ISA} with the speeds of vehicle 1 and 2



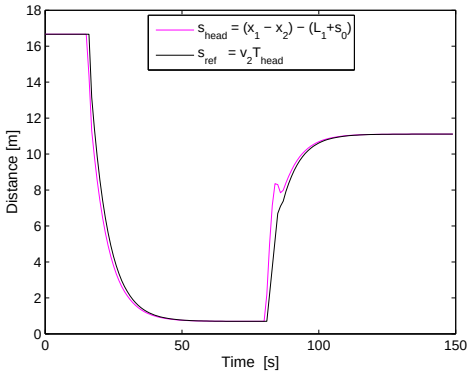
(b) Error $v_1 = v_{\mathrm{ISA}} - v_1$



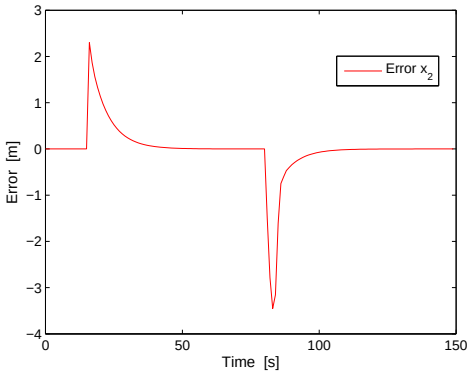
(c) Positions of vehicles 1 and 2



(d) Error $v_2 = v_1 - v_2$

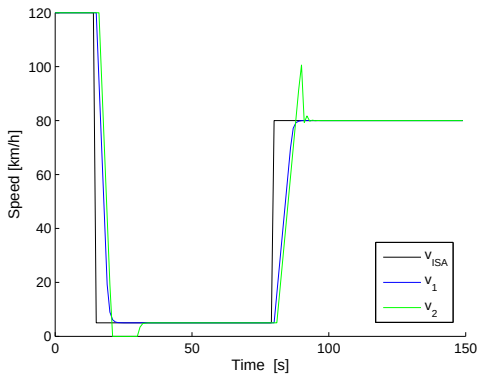


(e) The reference headway s_{ref} and the actual space headway s_{head}

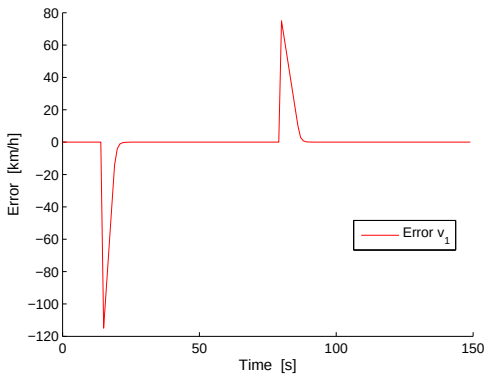


(f) Error $x_2 = x_1 - x_2$

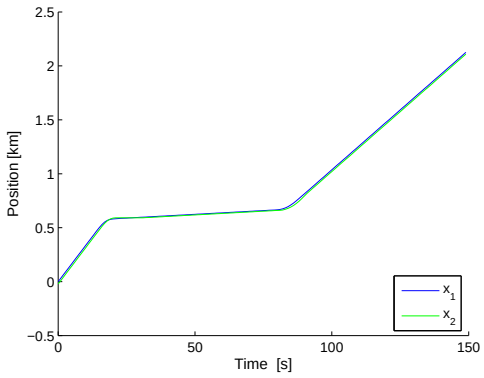
Figure 5-5: Vehicle 1 follows a reference headway v_{ISA} with a controller gain $K_{\mathrm{l}} = 0.04$ and vehicle 2 follows vehicle 1 with controller gains $K_{\mathrm{v}} = 0.3$ and $K_{\mathrm{x}} = 0.01$



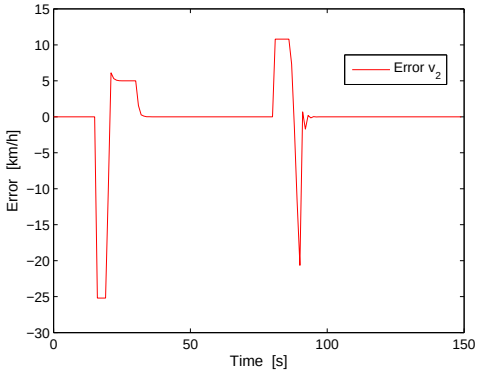
(a) The trajectory of the reference speed v_{ISA} with the speeds of vehicle 1 and 2



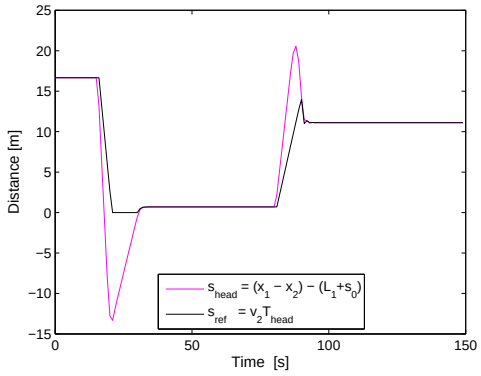
(b) Error $v_1 = v_{\mathrm{ISA}} - v_1$



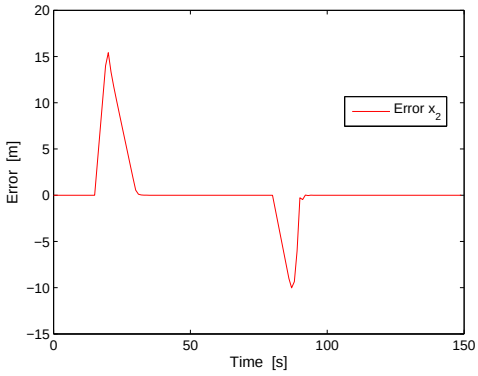
(c) Positions of vehicles 1 and 2



(d) Error $v_2 = v_1 - v_2$



(e) The reference headway s_{ref} and the actual space headway s_{head}



(f) Error $x_2 = x_1 - x_2$

Figure 5-6: Vehicle 1 follows a reference headway v_{ISA} with a controller gain $K_1 = 0.2$ and vehicle 2 follows vehicle 1 with controller gains $K_{\mathrm{v}} = 0.3$ and $K_{\mathrm{x}} = 0.01$. The controller gain K_1 is too big, since vehicle 2 cannot follow vehicle 1 smoothly

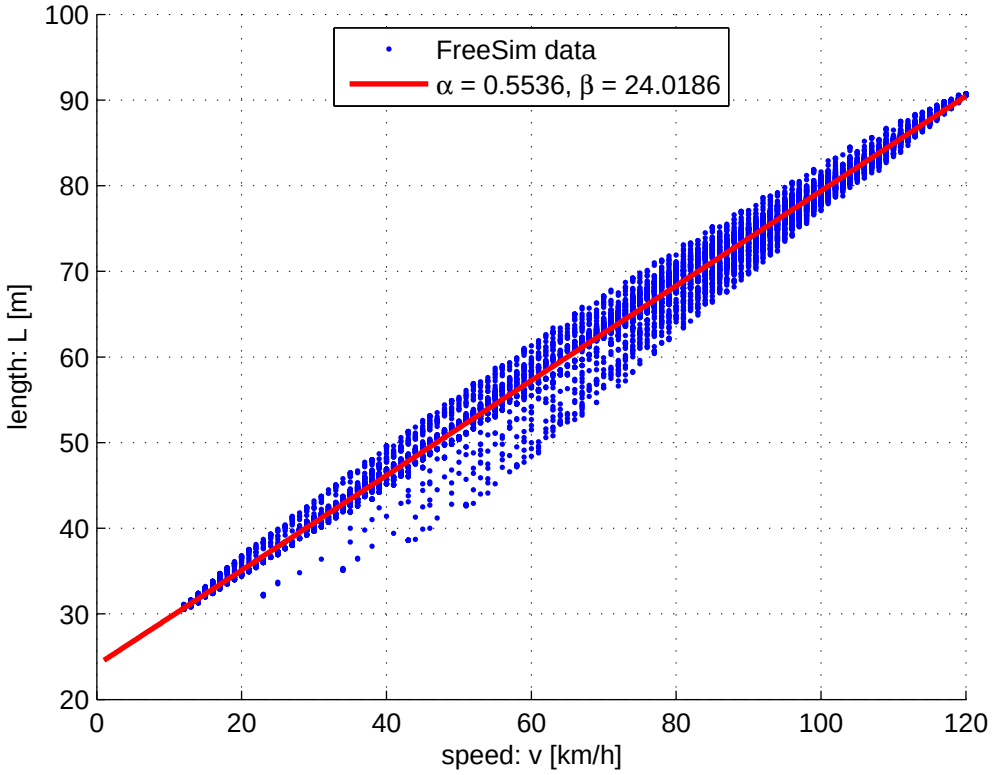
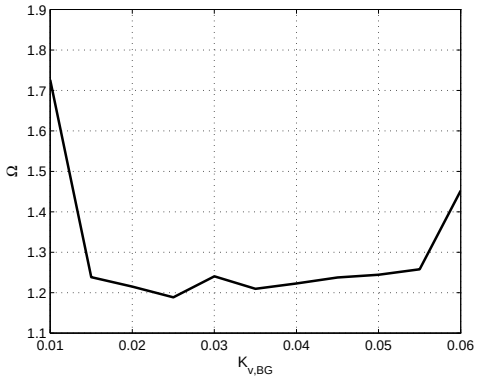
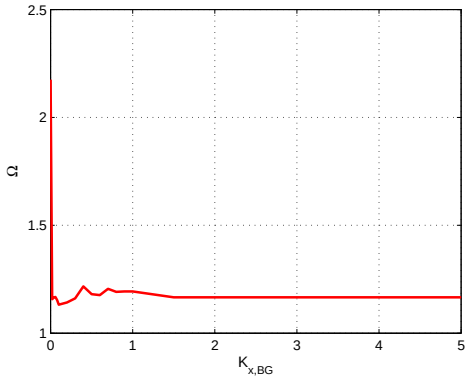


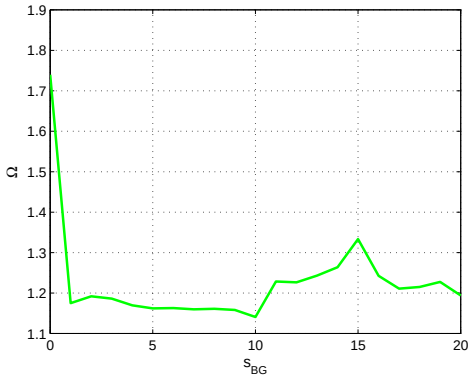
Figure 5-7: The dataset $\{v, L\}$ from FreeSim is plotted as a scatter plot. The line $L = \alpha v + \beta$ is plotted using the calibrated values $\alpha = 0.5536$ and $\beta = 24.0186$.



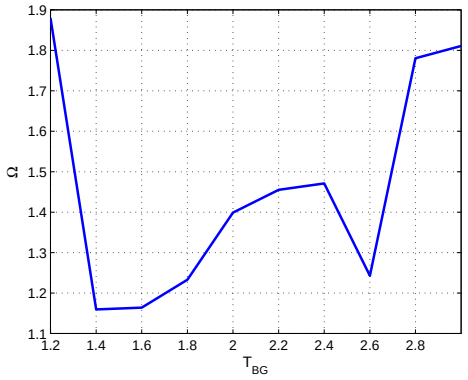
(a) Slicing $K_{\mathrm{v,BG}}$



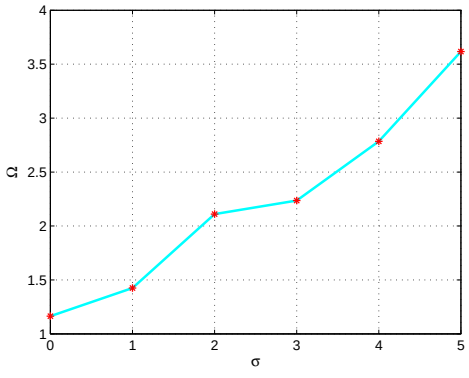
(b) Slicing $K_{\mathrm{x,BG}}$



(c) Slicing s_{BG}



(d) Slicing T_{BG}



(e) Slicing σ

Figure 5-8: Slices of the Big Car model around the point where $\Omega = 1.1639$, i.e. $K_{\mathrm{v,BG}} = 0.0209$, $K_{\mathrm{x,BG}} = 0.3033$, $s_{\mathrm{BG}} = 10.1872$, $T_{\mathrm{BG}} = 1.6003$, and $\sigma = 0$

Chapter 6

Conclusions and Recommendations

6-1 Conclusions

The recent developments in the control of Intelligent Vehicles (IV) indicate that by using platoons, the average traffic flow on freeways can be improved. This means that the Total Time Spent (TTS), which is an often used performance indicator, can be reduced. An effective control measure to improve the performance in traffic flow is speed limit control. An hierarchical control strategy in combination with Model Predictive Control (MPC) can be used for controlling all of the IV in a freeway network. However, no freely available traffic simulator exists that can model a freeway network consisting of IV and platoons in a fast and efficient manner. This means that no current traffic simulator can be used as a system model or as a prediction model in an MPC strategy. Therefore an assessment has been done to find a suitable microscopic traffic simulator that can be modified.

Based on the assessment described in this thesis, both MITSIM and FreeSim seemed to have high potential for modifications, such that it can be used as a microscopic traffic simulator for simulating IV-based traffic. Although MITSIM has more features than FreeSim, due to the lack of development in the last years and because there is no support, MITSIM is incompatible with most of the modern day computers. This makes MITSIM not suitable for further development.

FreeSim is public available since 2007, and was developed with Intelligent Transportations Systems in mind. It proved possible to implement IV models and platoons in FreeSim. Also variable speeds for vehicles have successfully been added in FreeSim. However, no car-following model is present in FreeSim as well as no reference speed for the platoon leaders is provided by FreeSim natively. This makes the traffic unresponsive for density changes. Therefore modifications have to be implemented such that reference speeds can be provided for the platoon leaders when the traffic is uncontrolled.

The METANET model is used in uncontrolled traffic to compute reference speeds for the platoon leaders based on the traffic states. The parameters of the fundamental diagram, which is part of the relaxation term in METANET, can be estimated by assuming a constant

number of vehicles in a platoon as well as assuming equal distributions of vehicles and platoons on the edges. Such an estimation provides a curve with an increased capacity flow, which would be expected at traffic consisting out of platoons.

Due to the increased complexity of FreeSim as a result of the implemented models and speed limiting control possibilities, the simulation speed is decreased. Therefore, FreeSim is too slow to be used as a prediction model in MPC. The Big Car following model that is implemented in Matlab has a much faster simulation speed.

The Big Car model can use a following model that is currently not implemented in FreeSim. Because of the modeling differences between the Big Car following model and FreeSim, the parameters of the Big Car model have to be calibrated in order that the simulated output of both models are comparable. The calibration of the Big Car models can be done in Matlab using multi-start nonlinear least squares optimization techniques.

Optimizing the parameters of the Big Car model can be divided in multiple steps. The first step is optimization the parameters that are needed to compute the length of a big car, i.e. α and β. The other step is optimizing the parameters of the Adaptive Cruise Control (ACC), i.e. $K_{\mathrm{v,BG}}$ and $K_{\mathrm{x,BG}}$, and the values for safe headway, i.e. s_{BG} and T_{BG}.

Optimization α and β can be done very fast. The optimal values prove to be almost the same as the values that can be calculated as if platoons have a constant speed, i.e. thus not acceleration or decelerating. The average error between the length of a big car and of a platoon in FreeSim is slightly more than 1 %. However, the highest measured error is 20 %, which is due to the deceleration of a platoon. The highest measured error due to acceleration is 7 %. The length of a big car can thus be modeled quite well with the linear speed dependent function $L = \alpha v + \beta$. However, at large accelerations and especially at large decelerations, the errors can be quite significant.

The calibration of the second step is very time consuming as one optimization run can take up to ten hours. It is necessary to use multi-start optimization, because the optimal values found during the optimizations are dependent on the starting point. Most of the time the optimal value found for $K_{\mathrm{v,BG}}$ was around the same order of magnitude. The optimal value for $K_{\mathrm{x,BG}}$ could differ per optimization run. By using slicing, it was seen that the mean squared error is not affected at all for high values of $K_{\mathrm{x,BG}}$. This is logical, as a high value for $K_{\mathrm{x,BG}}$ means that the headway is controlled very rigidly. But if the chosen values for the headway are wrong, then better controlling a wrong headway does not result in a smaller error.

The optimization found almost always optimal values for s_{BG} and T_{BG} close to the starting values. This is due to the modeling differences between FreeSim and the Big Car model. The speed of the platoons in FreeSim is mostly affected by the average density of the edge, whereas the speed of a big car is mainly influenced by the big car directly ahead. This indicates that speed adjustments based on macroscopic quantities can be valid, but comparing the platoons in FreeSim with big cars on a microscopic level results in errors due to the modeling differences.

The Big Car following model can be used as a prediction model in MPC since the Big Car model proves to be a lot faster than FreeSim. However, the big cars in the Big Car model react more quickly on increased densities than the platoons in FreeSim will. This can lead in too optimistic travel times in the Big Car model compared to FreeSim simulations.

6-2 Recommendations

Now that a microscopic traffic simulator has been developed as well as a fast model that can be used as a prediction model, simulations can be performed with speed limit controlled traffic. Using the combination FreeSim and the Big Car model, it can be investigated what the result in increased performance of a freeway network can be if traffic consists out of platoons that have MPC controlled dynamic speed limits. Different scenarios can be created to test the influence of speed limit control in different circumstances.

The chosen parameters of the METANET model in FreeSim are based on estimations and on calibrations done on data of freeways with human drivers. These parameters cannot be calibrated for platoon-based traffic, since this data is simply not available. The validation of the use of a METANET model that is developed for modeling human based traffic can thus not be guaranteed. It should be better to investigate whether it is possible to implement a following model for platoons in FreeSim. This also solves the problem that collisions in FreeSim can happen unnoticed, because the platoons are not aware of each other.

One of the reasons that FreeSim is not very fast is due to the fact that originally FreeSim was modeled such that all vehicles are simulated as separate threads. At each simulation time step all vehicles update their states synchronously. But now the vehicles inside a platoon are dependent on each other, so the platoons are models such that first the platoon leader updates its states, then the vehicle behind it and so on. So all platoons update their states in parallel, but the vehicles per platoon do this consecutively. This contributes in a slower process of updating all vehicle states in a platoon. This can maybe be solved by using a (hash-)table per vehicle where the states of that vehicle are inserted at each simulation time step. In this way all vehicles can adjust their accelerations synchronously, since the states of the predecessor can be looked up in the table of that vehicle. It can be investigated whether this really results in a speed increase or that FreeSim only uses more memory or maybe even the speed decreases due to fact that now also all tables need to be updated each time step.

FreeSim is now a fairly basic traffic simulator. It is now only possible to use speed limits as a control measure. FreeSim can be extended with for example ramp metering and more intelligent route algorithms. The current routes are based on either shortest or fastest paths. For instance an MPC can be used for controlling the optimal routes of the vehicles by predicting future traffic states.

Another useful modification to FreeSim are queue models. Currently the command file must be created wisely to avoid unrealistic instantiations of platoons. With a queue model, platoons that are instantiated can be put inside a queue and added to the freeway network when the traffic allows this. It is also interesting to investigate whether the queue model can take care of the initial number of vehicles in each platoon instead of predefining this in the command file.

Since the number of studies in the field of IV control will probably only increase in the future, the importance of good microscopic traffic simulators that can cope with platoon based traffic will also increase. FreeSim can be extended with more models and control measure capabilities, but FreeSim already is not very fast. Maybe it is feasible to develop a totally new microscopic traffic simulator. The best option is to develop this simulator with a modular approach, such that it can be easily extended with different types of models. This simulator must be created with the platooning concept in mind and also that a traffic network

will be controlled in a hierarchical framework. In this way the simulator will be optimized for IV-based traffic and it will not be a modified simulator that was intentionally not designed for these purposes. The best results can be made by using the same approach as FreeSim and MITSIM, that is by making it freely available for other users, e.g. universities. In this way the development can be done in a collaboration and for instance models can be validated by different people. This will speed up the process of the development and this traffic simulator can be acknowledged by other researchers.

Bibliography

[1] http://groups.yahoo.com/group/mitsimlab/. Last visited on August 1, 2008.

[2] http://mit.edu/its/mitsimlabosnew.html. Last visited on Februari 12, 2009.

[3] A. Alessandri, A. Di Febbraro, A. Ferrara, and E. Punta. Optimal control of freeways via speed signalling and ramp metering. *Control Engineering Practice*, 6(6):771–780, 1998.

[4] L.D. Baskar, B. De Schutter, and H. Hellendoorn. Hierarchical traffic control and management with intelligent vehicles. In *Proceedings of the 2007 IEEE Intelligent Vehicles Symposium (IV'07)*, pages 834–839, Istanbul, Turkey, June 2007.

[5] L.D. Baskar, B. De Schutter, and H. Hellendoorn. Model predictive control for intelligent speed adaptation in intelligent vehicle highway systems. In *Proceedings of the 17th IEEE International Conference on Control Applications*, pages 468–473, San Antonio, Texas, September 2008.

[6] T. Bellemans, B. De Schutter, and B. De Moor. Model predictive control for ramp metering of motorway traffic: A case study. *Control Engineering Practice*, 14(7):757–767, July 2006.

[7] R. Bishop. A survey of intelligent vehicle applications worldwide. In *Proceedings of the IEEE Intelligent Vehicles Symposium, IV 2000*, pages 25–30, Dearborn, MI, USA, 2000.

[8] R. Bishop. *Intelligent Vehicle Technology and Trends*. Artech House, 2005.

[9] J. Blum and A. Eskandarian. Managing effectiveness and acceptability in intelligent speed adaptation systems. In *Proceedings of the IEEE Intelligent Transportation Systems Conference, ITSC '06*, pages 319–324, Toronto, Canada, 2006.

[10] S.A. Boxill, L. Yu, Research Center for Transportation Training and, University Texas Southern, and Center Southwest Region University Transportation. *An Evaluation of Traffic Simulation Models for Supporting ITS Development*. Center for Transportation Training and Research, Texas Southern University; Available through the National Technical Information Service, 2000.

[11] M. Broucke and P. Varaiya. The automated highway system: A transportation technology for the 21st century. *Control Engineering Practice*, 5(11):1583–1590, 1997.

[12] C.Y. Chan and H.S. Tan. Evaluation of magnetic markers as a position reference system for ground vehicle guidance and control. Technical report, 2003. *California Partners for Advanced Transit and Highways (PATH). Research Reports:* Paper UCB-ITS-PRR-2003-08.

[13] S. Darbha and K. R. Rajagopal. Intelligent cruise control systems and traffic flow stability. *Transportation Research Part C: Emerging Technologies*, 7(6):329–352, 1999.

[14] L.C. Davis. Effect of adaptive cruise control systems on traffic flow. *Physical Review E*, 69(6):066110, 2004.

[15] C.E. Garca, D.M. Prett, and M. Morari. Model predictive control: Theory and practice–a survey. *Automatica*, 25(3):335–348, 1989.

[16] S. Halle and B. Chaib-draa. A collaborative driving system based on multiagent modelling and simulations. *Transportation Research Part C: Emerging Technologies*, 13(4):320–345, 2005.

[17] A. Hegyi. *Model predictive control for integrating traffic control measures.* Trail Thesis Series T2004/2. Netherlands TRAIL Research School, Delft, 2004.

[18] A. Hegyi, B. De Schutter, and H. Hellendoorn. Model predictive control for optimal coordination of ramp metering and variable speed limits. *Transportation Research Part C*, 13(3):185–209, June 2005.

[19] D. Helbing. Traffic and related self-driven many-particle systems. *Reviews of Modern Physics*, 73(4):1067–1141, December 2001.

[20] S.P. Hoogendoorn and P.H.L. Bovy. State-of-the-art of vehicular traffic flow modelling. *Proceedings of the Institution of Mechanical Engineers, Part I-Journal of Systems and Control Engineering*, 215(I4):283–303, 2001.

[21] R. Horowitz and P. Varaiya. Control design of an automated highway system. *Proceedings of the IEEE*, 88(7):913–925, July 2000.

[22] M. Koshi, M. Iwasaki, and I. Ohkura. Some findings and an overview on vehicular flow characteristics. In *Proceedings of the 8th International Symposium on Transportation Traffic Theory*, pages 403–426, Toronto, Canada, 1983.

[23] A. Kotsialos, M. Papageorgiou, C. Diakaki, Y. Pavlis, and F. Middelham. Traffic flow modeling of large-scale motorway networks using the macroscopic modeling tool metanet. *Intelligent Transportation Systems, IEEE Transactions on*, 3(4):282–292, Dec 2002.

[24] L. Kun and P. Ioannou. Modeling of traffic flow of automated vehicles modeling of traffic flow of automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 5(2):99–113, 2004.

[25] J. Miller and E. Horowitz. Freesim – a free real-time freeway traffic simulator. In *Proceedings IEEE Intelligent Transportation Systems Conference ITSC 2007*, pages 18–23, Seattle, Washington, USA, 2007.

[26] K. Nagel, P. Wagner, and R. Woesler. Still flowing: Approaches to traffic flow and traffic jam modeling. *Operations Research*, 51(5):681–710, 2003.

[27] R.C. Nicklin. Kinematics of tailgating. *The Physics Teacher*, 35(2):78–79, 1997.

[28] Technical University of Crete and A. Messmer. *METANET - A simulation program for motorway networks*. Technical University of Crete, Dynamic Systems and Simulation Laboratory and A. Messmer, Nov. 2000.

[29] L.E. Owen, Yunlong Zhang, Lei Rao, and G. McHale. Traffic flow simulation using corsim. In *Proceedings Winter Simulation*, volume 2, pages 1143–1147, Orlando, FL, 2000.

[30] Kumud K. Sanwal, Karl Petty, Jean Walrand, and Youssef Fawaz. An extended macroscopic model for traffic flow. *Transportation Research Part B: Methodological*, 30(1):1 – 9, 1996.

[31] M. Smith, G. Duncan, and S. Druitt. Paramics: microscopic traffic simulation for congestion management. In *Proceedings IEE Colloquium on Dynamic Control of Strategic Inter-Urban Road Networks*, pages 8/1–8/3, London, England, 23 Feb 1995.

[32] S. Tsugawa, S. Kato, T. Matsui, H. Naganawa, and H. Fujii. An architecture for cooperative driving of automated vehicles. In *Proceedings of the IEEE Intelligent Transportation Systems. Symposium*, pages 422–427, Dearborn, MI, USA, 2000.

[33] http://mctrans.ce.ufl.edu/featured/tsis/. Last visited on August 1, 2008.

[34] http://www.auto21.ca/home_e.html. Last visited on June 12, 2008.

[35] http://www.adobe.com/devnet/flashplayer/articles/socket_policy_files.html. Last visited on March 14, 2009.

[36] M. Van Aerde, B. Hellinga, M. Baker, and H. Rakha. INTEGRATION: An overview of traffic simulation features. In *Transportation Research Board 75th Annual Meeting*, Washington DC, Januari 1996.

[37] P. Varaiya. Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control*, 38(2):195–207, 1993.

[38] Q.I. Yang and H.N. Koutsopoulos. A microscopic traffic simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C: Emerging Technologies*, 4(3):113–129, 1996.

[39] Y. Zhang and L.E. Owen. An advanced microscopic traffic simulation approach for modeling its applications. In *Proceedings IEEE International Conference on Systems, Man, and Cybernetics*, volume 4, pages 3228–3233, 1998.

Appendix A

FreeSim Description

In this appendix a more extensive description on how FreeSim works is given, by explaining what all objects in Figure 3-2 are. It is not the intention to give a full manual of FreeSim, but merely to present a brief pointwise description, such that it is more clear which actions are performed when FreeSim is started. All names that are used in this appendix are the same as the one used in the code of FreeSim.

A-1 TrafficSimulator

- When the TrafficSimulator program is started, it opens a TrafficDatabase-Proxy, such that it can read the freeway networks from the database. It stores each freeway system in a so called graph. These graphs are used from within FreeSim.
- The TrafficSimulator starts a TrafficSimulatorListenerInstantiator, see Section A-2.
- The TrafficSimulator creates a *serversocket* listening on port: *iport*.
- When a connection is received on *iport*, a VehicleThread (see Section A-4) is created, which acts as a server for that connection. The *serversocket* stays listening for new connections on *iport*.

A-2 TrafficSimulatorListenerInstantiator

- When a TrafficSimulatorListenerInstantiator is created, it creates a Traffic-SimulatorNotifier, see Section A-3.
- The TrafficSimulatorListenerInstantiator creates a *serversocket* listening on port: *trafficSimulatorInstantiatorPort*.

- When a connection is received on the *trafficSimulatorInstantiatorPort*, a TrafficSimulatorListener is created. This TrafficSimulatorListener is added at the *vectTrafficSimulatorListeners* in TrafficSimulatorNotifier. The *serversocket* stays listening for new connections on *trafficSimulatorInstantiatorPort*.
  - The TrafficSimulatorListener acts as a server, and sends a message when the TrafficSimulatorNotifier gives the command.

A-3 TrafficSimulatorNotifier

- The TrafficSimulatorNotifier contains a vector *vectTrafficSimulatorListeners*, where all TrafficSimulatorListeners, created by the TrafficSimulatorListenerInstantiator are stored.
- Every *imilliseconds* milliseconds, the TrafficSimulatorNotifier notifies all TrafficSimulatorListeners, with the time.

A-4 VehicleThread

A VehicleThread acts as an important server that handles a connection between the GUI (user interface) and the main program, but it also handles the connection between a vehicle and the main program.

- A VehicleThread is created when a connection is established on *iport*. This is done for instance when the GUI is started. The GUI thus has a connection with a VehicleThread.
- A VehicleThread is also created when a vehicle is instantiated, because each vehicle needs to establish a connection on *iport*, see Section A-7.
- When a simulation is started by executing a command file, the VehicleThread that has a connection with the GUI creates a VehicleLoader, see Section A-5.
- A VehicleThread handles all requests from the GUI or vehicles, such as requesting of fastest routes or termination of the simulation.

A-5 VehicleLoader

- When the VehicleLoader is created, it first creates a VehiclesNotifier and subsequently adds itself in *vectVehicleListeners* in the VehiclesNotifier.
- The VehicleLoader processes the command text file by putting timed events in a hashtable *htTimedEvents*. When the VehicleLoader receives a signal from the VehiclesNotifier it looks for timed events in *htTimedEvents* that corresponds to the current time, to check whether an event has to be executed. Such an event can for instance be to create a new vehicle.

A-6 VehicleNotifier

- When a VehicleNotifier is created it establishes a connection on *trafficSimulatorInstantiatorPort*, such that it receives periodically a signal with the time.
- The VehicleNotifier contains a vector *vectVehicleListeners*, where all vehicles and the VehicleLoader are stored. All vehicles and the VehicleLoader receive a signal to process the next simulation step when the VehiclesNotifier receives a signal with the time.

A-7 Vehicle

- When a vehicle is created, it establishes a connection on *iport*, such that a communication link is created with a VehicleThread.
- After the connection is established, the vehicle adds itself in the *vectVehicleListeners* in the VehiclesNotifier. Now it can periodically receive a signal from VehiclesNotifier to process the next simulation step.
- Because a vehicles has a connection with a VehicleThread it can for instance request whether there is a faster path available.

Index