Institutionen för systemteknik
Department of Electrical Engineering

Examensarbete

Driver-truck models for software-in-the-loop simulations

Examensarbete utfört i Fordonssystem
vid Tekniska högskolan vid Linköpings universitet
av

Oskar Daniels

LiTH-ISY-EX–14/4799–SE

Linköping 2014

Linköpings universitet
TEKNISKA HÖGSKOLAN

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

Driver-truck models for software-in-the-loop simulations

Examensarbete utfört i Fordonssystem
vid Tekniska högskolan vid Linköpings universitet
av

Oskar Daniels

LiTH-ISY-EX–14/4799–SE

Handledare: **Chih Feng Lee**
ISY, Linköpings universitet
Samuel Wickström
Scania

Examinator: **Erik Frisk**
ISY, Linköpings universitet

Linköping, 9 oktober 2014

Avdelning, Institution Division, Department Vehicular systems Department of Electrical Engineering SE-581 83 Linköping	**Datum** Date 2014-10-09

Språk Language ☐ Svenska/Swedish ☒ Engelska/English ☐ ________	**Rapporttyp** Report category ☐ Licentiatavhandling ☒ Examensarbete ☐ C-uppsats ☐ D-uppsats ☐ Övrig rapport ☐ ________	**ISBN** — **ISRN** LiTH-ISY-EX–14/4799–SE **Serietitel och serienummer** Title of series, numbering **ISSN** —

URL för elektronisk version

http://www.ep.liu.se

Titel Title
Förar- och fordonsmodeller för software-in-the-loop-simuleringar
Driver-truck models for software-in-the-loop simulations

Författare Author
Oskar Daniels

Sammanfattning
Abstract

By using vehicle-to-vehicle communication, vehicles can cooperate in many ways by sending positions and other relevant data between each other. One popular example is platooning where many, especially heavy vehicles, drive on a trail with short distances resulting in a reduction of air resistance. To achieve a good efficiency of the platooning it is required that vehicle fleets are coordinated, so that the percentage of time for driving in platoon is maximized without affecting the total driving time and distance too much. For large fleets, this is a complex optimization problem which would be difficult to solve by only using the real world as the test environment.

To provide a more adaptable test environment for the communication and platooning coordination, an augmented reality with virtual vehicles (“Ghost trucks") with relevant communication abilities are developed. In order to realise the virtual testing environment for trucks, Scania initiated a project that could be divided into the workload of three master theses. This thesis involved the part of developing the virtual vehicles, which include the development of a truck model and a driver model.

The developed truck model consists of a single track vehicle model and several powertrain models of different complexity provided by Scania. Additionally, the driver model consists of steering wheel and speed controls in order to keep the truck on a safe distance from the lead truck and stay on a preferred lane. The key feature of the driver-truck model is its modular design, which provides great flexibility in selecting the level of detail for each component. The driver-truck model can be duplicated and simulated together in real time and performs platooning with each other in a road system based on the real world. As the driver-truck model is module based, it can easily be extended for future purposes with more complex functions.

The driver-truck model is implemented in Simulink and the simulation performance for different model complexity is evaluated. It is demonstrated that the flexibility of the developed model allows a balanced decision to be made between realistic truck behavior and simulation speed. Furthermore, multi-truck simulations are performed using the model, which demonstrate the effectiveness of the model in the evaluation of truck platooning operations.

Nyckelord
Keywords vehicle models, driver models, trucks, platooning

Abstract

Genom att använda kommunikation mellan fordon, kan dessa samarbeta genom att skicka positioner och annan relevant data mellan varandra. Ett populärt exempel är platooning där många, i synnerhet tunga fordon, kör i en kolonn med korta avstånd som resulterar i en minskning av luftmotståndet. För att uppnå en god effektivitet med platooning krävs att fordonsflottor är koordinerade så att andelen tid för körning i kolonn maximeras utan för mycket påverkning av den totala körtiden. För stora flottor är detta naturligtvis ett komplext optimeringsproblem som skulle vara svårt att lösa genom att endast använda den verkliga världen som testmiljö.

För att ge en mer anpassningsbar testmiljö för kommunikation och koordinering av platooning, utvecklades en förstärkt verklighet med virtuella fordon (“ Ghost trucks ") med relevanta kommunikationsförmåga. För att förverkliga den virtuella testmiljön för lastbilar inledde Scania ett projekt som kunde delas upp i tre examensarbeten. Detta examensarbete involverade delen för att utveckla de virtuella fordonen, vilket inkluderar implementation av en förar- och lastbilsmodell.

Den utvecklade lastbilsmodellen består av en tvåhjulig fordonsmodell med flera drivlinemodeller av olika komplexitet från Scania. Dessutom består förarmodellen av ratt- och hastighetskontroller för att hålla lastbilen på ett säkert avstånd från den framförvarande lastbilen samt hålla sig inom en önskad vägfil. Ett viktigt inslag i modellen är dess modulkonstruktion som ger stor flexibilitet vid val av detaljnivå för varje komponent. Förar- och fordonsmodellen kan dupliceras för att sedan simuleras tillsammans i realtid och utföra platooning med varandra på ett vägsystem baserat på den verkliga världen. Tack vare att förar- och fordonsmodellen är modulbaserad kan den enkelt utvecklas för framtida ändamål med mer komplexa funktioner. Relevansen av dynamiken för fordonsmodellen bevisades med hjälp av fysiska beräkningar, och resultatet utvärderades med tillfredsställande resultat.

Fordonsmodellen är implementerad i Simulink och prestandan för de olika modellernas komplexitet har simulerats och utvärderas . Det har visats att flexibiliteten i modellen möjliggör ett balanserat beslut mellan realismen i lastbilens beteende och dess simuleringshastighet. Dessutom har flera lastbilar kunnat simuleras samtidigt med hjälp av modellen, vilket visart effektiviteten av modellen i utvärderingen för koordinering av platooning.

Abstract

By using vehicle-to-vehicle communication, vehicles can cooperate in many ways by sending positions and other relevant data between each other. One popular example is platooning where many, especially heavy vehicles, drive on a trail with short distances resulting in a reduction of air resistance. To achieve a good efficiency of the platooning it is required that vehicle fleets are coordinated, so that the percentage of time for driving in platoon is maximized without affecting the total driving time and distance too much. For large fleets, this is a complex optimization problem which would be difficult to solve by only using the real world as the test environment.

To provide a more adaptable test environment for the communication and platooning coordination, an augmented reality with virtual vehicles ("Ghost trucks") with relevant communication abilities are developed. In order to realise the virtual testing environment for trucks, Scania initiated a project that could be divided into the workload of three master theses. This thesis involved the part of developing the virtual vehicles, which include the development of a truck model and a driver model.

The developed truck model consists of a single track vehicle model and several powertrain models of different complexity provided by Scania. Additionally, the driver model consists of steering wheel and speed controls in order to keep the truck on a safe distance from the lead truck and stay on a preferred lane. The key feature of the driver-truck model is its modular design, which provides great flexibility in selecting the level of detail for each component. The driver-truck model can be duplicated and simulated together in real time and performs platooning with each other in a road system based on the real world. As the driver-truck model is module based, it can easily be extended for future purposes with more complex functions.

The driver-truck model is implemented in Simulink and the simulation performance for different model complexity is evaluated. It is demonstrated that the flexibility of the developed model allows a balanced decision to be made between realistic truck behavior and simulation speed. Furthermore, multi-truck simulations are performed using the model, which demonstrate the effectiveness of the model in the evaluation of truck platooning operations.

Acknowledgments

I would like to thank Olivier and Sina for the good cooperation, and also the other guys at Scania for nice discussions during the coffee breaks. I would also like to thank both my supervisors, Samuel and Lee, for all the help during the thesis. I thank my father, mother, sister and brother for a long time of love and support. Finally a special thanks to all the brothers and sisters in OL4an. Thanks to you, these last five years have been absolutely amazing.

Stockholm, August 2014
Oskar Daniels

Contents

1 Introduction

1.1 Background

Transportation is and will always be an important part of the society. As more and more goods are needed to be moved long distances, higher demands for heavy transports are in place. In the latest decades different intelligent transportation systems (ITS) have been developed to improve both safety and efficiency. Modern vehicles have access to a number of sensors and advance features that can be used to support the driver, for example navigation, adaptive cruise control and emergency brakes and anti-lock braking systems.

A new addition to the ITS is the vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication. Fleet Management Systems (FMS) has been created, providing companies with the possibility to organize their vehicle fleets for a better effiency. V2V will not only be useful for when vehicles should drive automatically, as a solution for the absence of the driver's vision. It can also be used to support the driver with a visual demonstration of the surrounding environment, and thereby adding information that can't be provided from windows or mirrors.

By using V2V the vehicles also can cooperate in many ways by sending positions and other relevant data between each other. One popular example is platooning where many, especially heavy vehicles, drive on a trail with short distances resulting in a reduction of air resistance. To achieve a good efficiency of the platooning it is required that vehicle fleets are organized so that the percentage of time for driving in platoon is maximized without affecting the total driving time and distance too much. Recent studies have been done (Liang [2014]) for how to coordinate the vehicles and plan their routes to use platooning in a more efficient way, by adding more intelligence in the fleet management systems. However for large fleets, this will of course be a complex optimisation problem that would be difficult to develop by only using the real world as the test environment.

1.2 Problem formulation

To provide a more adaptable test environment for the communication and platooning coordination, an augmented reality with virtual vehicles (“Ghost trucks") with relevant communication abilities can be developed in software and demonstrated by a visualisation software. The augmented reality can provide information that can be used to test the platooning coordination, but also to test other transportation systems, where the real truck responds to simulated sensor data. A primary test scenario can be a real truck catching up a virtual truck and use this to perform platooning or other adaptive speed control.

When tests are performed in an augmented world, the development can be both cheaper and safer. However, it is crucial that the virtual vehicles have dynamics that respond to internal and external disturbances similar to real trucks and also behaves as if it is controlled by a real driver.

1.2.1 The project

In order to realise the virtual testing environment for trucks, Scania initiated a project that can be divided into workload of three master theses. The project involves developing a system consisting of virtual trucks, a server based environment and a visualisation software. Each part are developed by one thesis student.

The server based environment were to use data from Open Street Map to create the virtual world consisting of multiple virtual trucks and provide a link between the connected trucks and the visualisation software. The server were also expected to handle the communication between the virtual trucks. The environment can be seen as a simulation of V2V communication. The server would access the truck models as dynamic-link libraries, were the formats of input and output were to be defined in the project.

The visualisation software provides a 3D-visualisation of the road system and traffic on a screen on board of the real truck or other clients connected to the server. The server and the virtual trucks would provide information that can be used to test the implementation of the software.

1.3 Objectives

The goal of this thesis was to develop the virtual truck models in a MATLAB/Simulink platform with dynamics that respond to internal and external disturbances similar to real trucks. The truck models were expected to automatically follow a provided route in real world roads (primary highways) and generate data in real time. Information such as positions, velocity and heading would be simulated and sent to a simulated environment which receives, collects and sends information between multiple virtual trucks. The trucks would then use the information given from the environment to follow the roads correctly, keep the speed limit, and avoid driving into other vehicles connected to the server. Other information such as fuel consumption would also be good to simulate, for example test scenarios when slip streaming effects are investigated. The virtual trucks were expected to be modeled to drive with the same inputs as for a real truck, i.e. the steering wheel and throttle.

For achieving the overall goal a set of subtasks were introduced. These were:

- Implement different powertrain models with different complexities based on a powertrain model developed by Scania CV AB.
- Implement a dynamic model for lateral and longitudinal motion.
- Implement a driver model that follows the road and drives legally.
- Compile the model as an dynamic library link for easy access from the server.
- Implement a driver model that reacts on and avoids collision with other vehicles connected to the server.

1.4 Related literature

Virtual vehicles had been used in many implementations before. One example is driving simulators where different vehicles with artificial intelligence is used for providing the driver with different challenges. One example is Han-wu et al. [2006]. However these implementations only need to mimic the physics in a way that the driver perceive as natural, rather than actually simulating the true dynamics of vehicles. Also, these vehicles are used for predefined driving routes, which results in different demands on the intelligence compared to the thesis.

For actually modeling dynamics of vehicles there are a lot of relevant literature and implementations. For the longitudinal dynamics, Scania have already done some implementations of realistic vehicle models for simulations in Simulink. However, there are some deficiencies with these models. Firstly the models are not restricted in computational complexity, because most of the simulations are done offline and often only uses one vehicle in the simulation. In this project a cluster of trucks is simulated in real time which add a lot more computational restrictions. Therefore simpler models are needed. Different methods of longitudinal modeling is described by Eriksson and Nielsen [2014].

For the lateral dynamics, Thommyppillai et al. [2009] describes a method where the axles of the vehicle is simplified to single wheels. This is a well used approach that is relevant for the thesis. The lateral forces on the tires is well investigated by Pacejka [2006].

There are also many different approaches for developing driver models for the steering of the vehicle. One popular method is by using the control perspective, where the driver is considered to be a controller which makes it easy to implement mathematical functions directly from input to output. Ljung and Glad [2003] describes the most popular of the general control strategies. There are infinite ways of using different states in the heading and positions for finding an optimal angle of the steering wheel. The methods by Chatzikomis and Spentzas [2009] and Sharp and Valtetsiotis [2001] are only a few examples. However the methods for finding the reference path are seldom described in these articles.

An other part of the driver model is the platooning intelligence. This is also an area that is well investigated, where different types of control methods can be used. The approach described by Alam [2014] is one example.

1.5 Contribution

By combining existing knowledge of vehicle and driver modeling, a complete driver-truck model is developed. Also a method for generating a reference path from arbitrary waypoints is implemented, which provides a driver-truck model that can follow any path that are defined in Open Street Map. The model can therefore be used to create an augmented reality from real word data where real and virtual trucks can drive together and interact in real time. This provides the possibility to test platooning coordination and other different control strategies with the virtual trucks as references. As the model is developed by modular implementation with simulink, it can be extended for other purposes as well.

1.6 Thesis outline

Chapter 2 describes the overall system that has been developed and the interaction between the subsystems.

Chapter 3 describes the modeling of the dynamics of the trucks, and how it has been implemented.

Chapter 4 describes how the driver has been modeled for following the route and behave in traffic.

Chapter 5 presents the results achieved in the thesis.

Chapter 6 presents an analysis of the results and describes possible future work.

2 The simulation environment

This chapter describes the developed system and presents an overview of the driver-truck model. Interactions between models and their respective inputs and outputs are described. Algorithms for handling the received data and transformations between coordinate systems are also presented.

2.1 Overview

The system that have been developed in cooperation with other master students in a project initiated by Scania CV AB consists of three parts: The server, the clients and the driver-truck models, as presented in Figure 2.1. The server collects information about the world to create a simulation environment, based on real world data with extensions from the simulated driver-truck models. The server also creates a database using the information from both the driver-truck models and from real trucks. It also uses Open Street Map and the Advanced Driver Assistance Systems Interface Specifications (ADASIS) to create routing data for the driver-truck models. In this project, the clients are real trucks that receives the information from the server and use it to extend the driver's knowledge of the surrounding augmented environment. However, a client could also be an observer that receives information for analyses.

2.2 Driver-truck model

The driver-truck model developed in this thesis emulates the behavior of a real truck and is able to interact with other clients (real trucks). It receives data from the server to be used by the driver model, which are described in Chapter 4. The

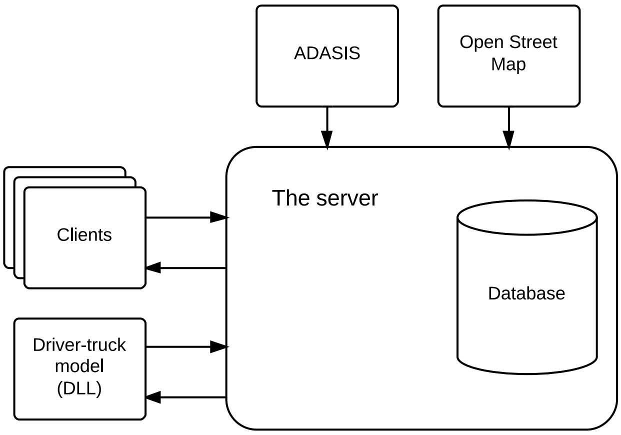

Figure 2.1: *Overview of the system*

driver model then controls the inputs to the vehicle model which are described in Chapter 3. The outputs from the vehicle model will then be generated and sent back to the server. A diagrammatic representation of the model can be seen in Figure 2.2. The driver-truck model is developed in Simulink and is compiled as a Dynamic-Link Library (DLL), using the generated C-code based on the model. The DLL then provides data types and functions for reading outputs, writing data to the inputs and set the model parameters. The server can then execute the model for generating and updating the states and outputs, which is calculated from the previous states. Each execution simulates the model for one time step of 0.01 seconds.

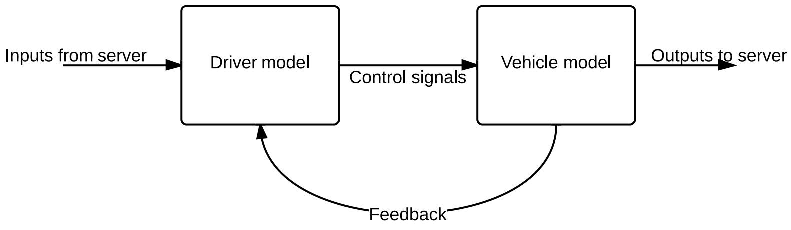

Figure 2.2: *Diagrammatic representation of the driver-truck model*

2.3 Inputs to the driver-truck model

The server can send data to the Simulink model by using the provided functions mentioned in Section 2.2. The data that is received from the server can be divided into four parts, which are the model parameters, the initial states, the routing data, and the surrounding traffic.

2.3.1 Model parameters

The server can set the parameters that define the dynamics of the vehicle which are presented in Table 3.1. The control parameters in the driver model will then be calculated as described in Chapter 4.

2.3.2 Initial states

The initial states include the initial position as an array consisting of the latitude and longitude in the geodetic coordinate system, the initial altitude in cm, and the initial heading as a scalar in radians from $-\pi$ to π, where 0 and $\frac{\pi}{2}$ represents north and east respectively.

Table 2.1: *Input data for the initial states*

Input	Structure
`initialPosition[2]`	$\{lat_0, long_0\}$
`initialHeading`	scalar
`initialAltitude`	scalar

2.3.3 Routing data

The routing data provides information about the route that the vehicle should follow. The server uses Open Street Map (Haklay and Weber [2008]) and the ADASIS protocol (Ress et al. [2008]) to send relevant data to the vehicle. The routing data includes latitude, longitude, altitude, number of lanes, speed limit, and road type for each waypoint that defines the desired route. The data is structured into arrays consisting data for the next ten desired waypoints and are updated each time a waypoint is passed. This provides the vehicle enough information about its future path, but still allows the server to dynamically change the route within a short time interval. The number ten is chosen for limiting the amount of data that shall be sent, and still provide sufficient data to the driver model. However, no investigation for finding an optimal number of waypoints are done within the time period of the thesis.

As the routing data are given in discrete positions, interpolation is needed to generate information between these points. The interpolation method for doing so is described in Section 2.6.

2.3.4 Surrounding traffic

The server provides information about the vehicles that surrounds the receiving vehicle. The server will find the four vehicles that are closest ahead of the receiving vehicle, and send the lateral and longitudinal distances to these vehicles in the local coordinates of the receiving vehicle. The received data will consist of an array which is presented in Table 2.3 where dx_i and dy_i denotes the relative longitudinal and lateral distances.

Table 2.2: *Input data for the routing information*

Input	Structure
`latitude[10]`	$\{lat_1, lat_2, ..., lat_{10}\}$
`longitude[10]`	$\{lon_1, lon_2, ..., lon_{10}\}$
`altitude[10]`	$\{alt_1, alt_2, ..., alt_{10}\}$
`nrOfLanes[10]`	$\{nr_1, nr_2, ..., nr_{10}\}$
`speedLimit[10]`	$\{speed_1, speed_2, ..., speed_{10}\}$
`routeType[10]`	$\{type_1, type_2, ..., type_{10}\}$

Table 2.3: *Input data for the surrounding traffic*

Input	Structure
`relativeDistances[8]`	$\{dx_1, dy_1, dx_2, dy_2, ..., dx_4, dy_4\}$

2.4 Outputs from the driver-truck model

Whenever the server calls the step function, the model will update its states based on the current inputs. The new states will then be accessible by reading the provided output data type. The output signals is presented in Table 2.4, which also includes the parameters that define the size of the vehicle.

Table 2.4: *Output data*

Output	Structure
`latitude`	scalar
`longitude`	scalar
`speed`	scalar
`heading`	scalar
`width`	scalar
`length`	scalar
`height`	scalar

2.5 Transformations between coordinate systems

The route data are received in the geodetic coordinate system used by the Global Position System (GPS), i.e. World Geodetic System 1984 (Smith [1987]). Hence, transformations between Smith [1987] and the vehicle model's inertial system, defined by the north-east coordinate system are needed. The relationship between the differential of the distances in northern-eastern directions and the angles for an ellipsoid is

$$dx_N = R_M(\phi)d\phi \tag{2.1}$$

$$dx_E = R_N(\phi)\cos(\phi)d\lambda, \tag{2.2}$$

where dx_N, dx_E are the differentials of the distances in northern and eastern directions respectively, and $d\lambda$, $d\phi$ are the differential of the longitude and latitude respectively. R_M and R_N is called “Radius of Curvature in the meridian” and “Radius of Curvature in the prime vertical” respectively. Both are functions of the current latitude and are defined as

$$R_M = \frac{a_{sm}(1-e^2)}{(1-e^2\sin^2\phi)^{3/2}} \tag{2.3}$$

$$R_N = \frac{a_{sm}}{\sqrt{1-e^2\sin^2\phi}}, \tag{2.4}$$

where a_{sm} is the semi-major axis and e is the first eccentricity which can be derived from the flattening, f, that defines the shape of the ellipsoid. The relationship is defined by

$$e^2 = 2f - f^2 \tag{2.5}$$

The Smith [1987] defining values for the semi-major axis of the earth and the reciprocal of flattening is presented in Table 2.5. From (2.1) and (2.2) the velocities in the northern and eastern directions can be transformed to the angular velocities in Smith [1987] by

$$\dot{\phi} = \frac{\dot{x}_N}{R_M(\phi)} \tag{2.6}$$

$$\dot{\lambda} = \frac{\dot{x}_E}{R_N(\phi)\cos(\phi)}, \tag{2.7}$$

where $\dot{x}_N$ is the vehicle velocity in the northern direction and $\dot{x}_E$ is the velocity in the eastern direction.

Table 2.5: *Smith [1987] defining parameters*

Parameter	**Symbol**	**Value**	**Unit**
Semi-major axis	a_{sm}	6378137.0	m
Reciprocal of flattening	$1/f$	298.257223563	-

2.6 Filtering of the routing data

The routing data only gives information at the actual waypoints that defines the route. When the vehicle is between two waypoints, the values for the number of lanes, speed limit and road type is assumed to equal the latest value until the next waypoint is passed. However for the actual path and slope, some filtering is needed before the data can be handled by the driver-truck model.

2.6.1 Path generation

The path is generated using the uniform Catmull-Rom spline (Catmull and Rom [1974]), seen in Figure 2.3. The Catmull-Rom spline is a special case of Beizer curve that generates a path that passes through each waypoint. The path will also be C1-continuous in each point, therefore avoiding having sharp corners resulted by a simpler linear interpolation. The algorithm that defines the spline is given by

$$\mathbf{S}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \mathbf{M_C} \begin{bmatrix} \mathbf{P_0} \\ \mathbf{P_1} \\ \mathbf{P_2} \\ \mathbf{P_3} \end{bmatrix}, \qquad 0 \leq t \leq 1, \tag{2.8}$$

with

$$\mathbf{M_C} = \frac{1}{2} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \tag{2.9}$$

where $\mathbf{S}(t)$ describes the spline between the points $\mathbf{P_1}$ and $\mathbf{P_2}$, with $t = 0$ at $\mathbf{P_1}$ and $t = 1$ at $\mathbf{P_2}$. In the special case where $\mathbf{P_1}$ is the first waypoint in the route, the unknown point $\mathbf{P_0}$ are replaced by $2\mathbf{P_1} - \mathbf{P_2}$, based on a assumption that there are no curvature before the starting point. Likewise, $\mathbf{P_3}$ are replaced by $2\mathbf{P_2} - \mathbf{P_1}$, in the case where $\mathbf{P_2}$ is the last known waypoint in the route.

2.6.2 Slope calculation

The slope is estimated as the average slope between two points, given by

$$\beta = \arctan \frac{H_{n+1} - H_n}{||\mathbf{P_{n+1}} - \mathbf{P_n}||}, \tag{2.10}$$

where H_n denotes the altitude in the latest passed waypoint, H_{n+1} is the altitude in the next waypoint, and $||\mathbf{P_{n+1}} - \mathbf{P_n}||$ equals the ground distance between these waypoints. The slope, β, is defined as an angle between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$, where a positive value represents an uphill slope.

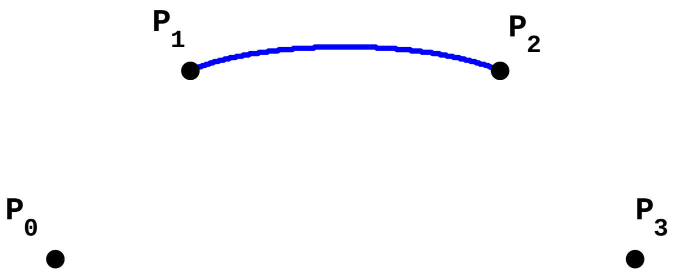

Figure 2.3: *A Catmull-Rom spline between* $\mathbf{P}_1$ *and* $\mathbf{P}_2$.

3 Development of vehicle models

The implementation of the vehicle model is described in this chapter. The basics of a powertrain from Scania is described as well as the theory behind the model for dynamics and kinematics. Parameters that have been used for the model are also summarized in Section 3.3.1. The vehicle model is simplified or extended depending on the desired complexity.

3.1 Engine and powertrain

The model of the engine and powertrain has been based on a model adapted from Scania CV AB. The basic purpose of the model is to simulate the longitudinal engine force, F_{eng}, from a desired speed reference. The parts that are included in the model are cruise controller, engine, gearbox and the final drive. These models will have behavior that is similar to the engines and powertrains in real Scania trucks. The models of the cruise controller, gearbox and engine are exchangeable to fit the purpose of simulation scenario, where more functionality results in higher computational effort.

3.1.1 The cruise controller

The cruise controller will generate the throttle based on a desired speed. There are three different variations of the cruise controllers with different levels of complexity, which include: a simple PID-controller, a standard cruise controller implemented by Scania, and a "Scania Active Prediction" cruise controller. The latter uses future slope of the road to decide which throttle level needed for following the reference speed.

3.1.2 The gearbox

The gearbox is modeled to automatically control which of the twelve different gears that are used. There are two different versions of the gear shift logic. The "Scania Active Prediction" cruise controller will always be used with a corresponding gear shift logic, which similar to the cruise controller will base the selected gear from the future slope of the road. When another cruise controller model is in use, a simpler gear shift logic are used. This is based on a static mapping, where each gear has a corresponding up- and down-shifting limit for the current speed.

3.1.3 The engine

The engine is modeled as an internal combustion engine (ICE), and generate the torque of the driveshaft based on the throttle from the cruise controller. The throttle will control how much fuel that are combusted, and the resulted torque are based by a static fuel map received from Scania. Finally the user is able to choose to calculate consumed fuel or not, and therefore there are two possible versions of the engine model. One where the used fuel are integrated over the simulation time, and one without those calculations.

3.2 Dynamics and kinematics

The dynamics and kinematics of the vehicle are based on a bicycle model, where the axles of the vehicle are simplified to single wheels. It is a model with three degrees of freedom and involves longitudinal, lateral and yaw motion. The forces and states in this model are shown in Figure 3.1.

The kinematics of the vehicle are described by

$$v_N = u \cos \psi - v \sin \psi \tag{3.1}$$

$$v_E = u \sin \psi + v \cos \psi, \tag{3.2}$$

where v_N and v_E are the northern and eastern velocities respectively. The longitudinal velocity and and the lateral velocity are denoted with u and v respectively, while ψ denotes the heading of the vehicle, i.e. the angle between the north vector and the vehicle's longitudinal vector.

Using the vehicle model presented by Thommyppillai et al. [2009], and merging the lateral forces for each axle, the dynamics of the vehicle is given by

$$m(\dot{u} - v\dot{\psi}) = F_x - F_{yf} \sin \delta \tag{3.3}$$

$$m(\dot{v} + u\dot{\psi}) = F_{yr} + F_{yf} \cos \delta \tag{3.4}$$

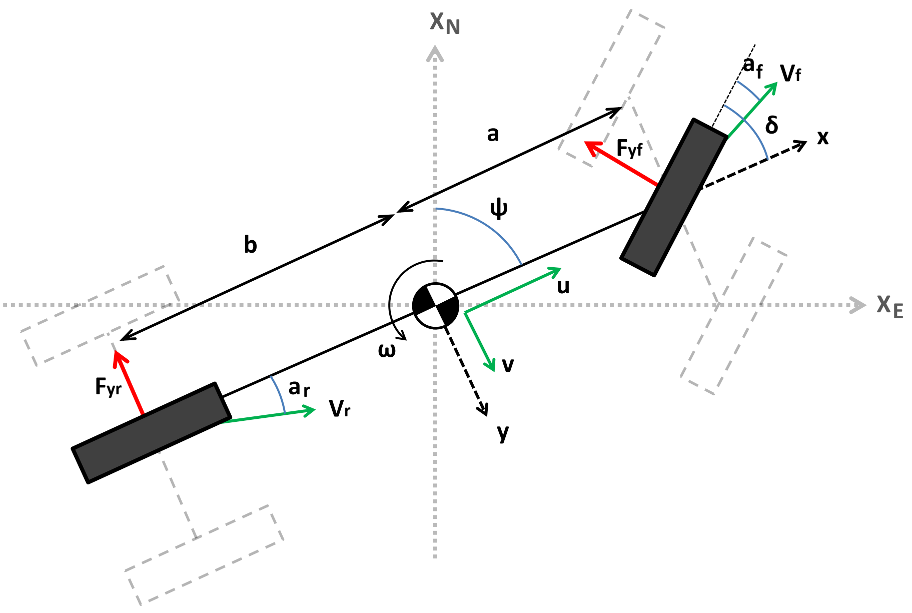

Figure 3.1: The bicycle model describing the basic dynamics and kinematics of a vehicle

$$I_z\ddot{\psi} = aF_{yf}\cos\delta - bF_{yr}, \tag{3.5}$$

where F_x is the total longitudinal force acting on the vehicle and F_{yf} and F_{yr} are the lateral forces on the front and rear tires. The length from the front axle to the center of gravity is denoted with a, and I_z is the moment of inertia around the vehicle's up vector which is defined as:

$$I_z = \frac{1}{12}m(w^2 + l^2), \tag{3.6}$$

where w and l are the width and length of the truck, which is approximated as a solid cuboid.

3.2.1 Longitudinal forces

The total longitudinal force, F_x, as described in (3.7), are resulted from the environment and the powertrain. The forces are shown in Figure 3.2 and F_x is given by:

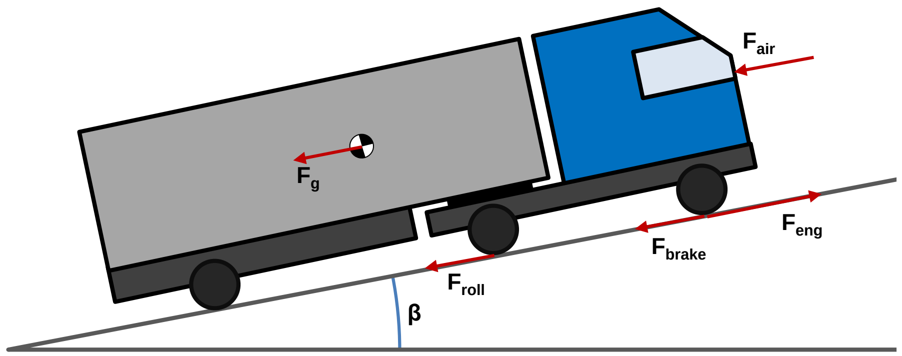

Figure 3.2: The longitudinal forces on the vehicle

$$F_x = F_{eng} - F_{brake} - F_{air} - F_{roll} - F_g, \tag{3.7}$$

where F_{eng} is the force from the powertrain described in Section 3.1, F_{brake} is the braking force induced by the driver, F_{air} is the force from the air drag, F_{roll} is the total force from the rolling resistance for all the wheels and F_g is the gravitational force on the vehicle.

Air drag force

The air drag force is caused by pressure from the air in front of the vehicle. However, the pressure are reduced when the vehicle is driving close to another vehicle. From empirical studies (Wolf-Heinrich [1998]) there have been shown that the air drag reduction is dependent not only on the distance to the vehicle ahead, but also on the distance to the second vehicle ahead and in some cases also on the distance to a following vehicle. The total formula for calculating the air drag resistance is given by

$$F_{air} = \frac{1}{2} c_D(d) A \rho v^2 \tag{3.8}$$

where A is the front area of the vehicle and ρ is the density of the air. The air drag coefficent c_D is given in Figure 3.3, and d is the distance to the vehicle ahead.

Rolling resistance

The rolling resistance force is caused by the tire deformation. The resulting force is given by:

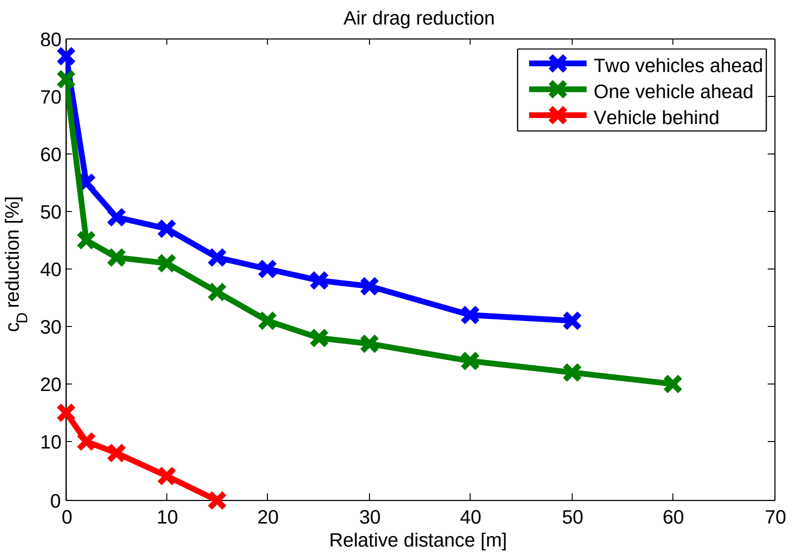

***Figure 3.3:** Mapping of the air drag coefficient $c_D(d)$ (Wolf-Heinrich [1998])*

$$F_{roll} = c_{roll} \cdot mg\cos(\beta) \tag{3.9}$$

where c_{roll} is a dimensionless constant depending on the properties of the tires and the ground surface.

Gravitational force

The gravitational force on the vehicle acts when the vehicle is driven uphill or downhill. From Figure 3.2 the force can be derived as:

$$F_g = mg\sin(\beta) \tag{3.10}$$

3.2.2 Lateral forces

The lateral rear and front forces, F_{yr} and F_{yf} are calculated by the *magic formula* (**?**) and are given by

$$F_{yi} = 2D\sin\{C\arctan[B(1-E)\alpha_i + E\arctan(B\alpha_i)]\} \tag{3.11}$$

where a_i is the slip angle. From Thommyppillai et al. [2009] the slip angle for the front axle is given by

$$\alpha_f = \frac{u \sin \delta - (v + a\dot{\psi}) \cos \delta}{u \cos \delta + (v + a\dot{\psi}) \sin \delta} \tag{3.12}$$

and the slip angle for the rear axle is given by

$$\alpha_r = \frac{b\dot{\psi} - v}{u} \tag{3.13}$$

where b is the length from the rear axle to the center of gravity.

The parameters B, C and E in (3.11) are called stiffness factor, shape factor and curvature factor respectively. These describe the characteristics of the tires and affect how the force will depend on the slip angle. The relationships are well described in **?** and typical values for the parameters are presented in Table 3.1. The last parameter, D, is called the peak factor that is the maximum lateral force that will act on the tires. It is sometimes estimated using μF_z where μ is the friction coefficient, which in this case are for rubber on asphalt. F_z is the normal force on the tire, which in the case of a bicycle model is the normal force for the whole axle. With the assumption that both axles have equal normal force, the peak factor for each axle can be estimated as

$$D \approx \frac{1}{2}\mu m g. \tag{3.14}$$

3.2.3 Assumptions used in the model

Due to various assumptions, the model is not a perfect representation of the reality. As alluded in Section 3.2 and Figure 3.1 the single track model employs a single (imaginative) wheel in the middle of every axle to represent forces experienced by the actual wheels. In reality, the forces may differ between each wheel which would affect the resulting motion. However, these differences would be not be significant in conditions where all the tires have similar properties and the ground surface are consistent, which both are assumed in this thesis.

In the graph for the air drag reduction in Figure 3.3, it is assumed that there are only Heavy-Duty Vehicles ahead or behind of the vehicle. If the surrounding vehicles were smaller, the air drag reduction would be lower.

Other error sources include model parameter discrepancies. Some values, for example tire parameters, would be difficult to obtain. Most parameters are estimated to values that are realistic but not explicitly given from a certain truck. Also the mass and center of gravity for the vehicle are dependent on the loading which could have a significant difference between different travels. However, the parameters can be changed by the user before the model is initialised.

3.3 Simulation models and parameters

The vehicle model is implemented in simulink, and a diagrammatic representations is shown in Figure 3.4. The inputs to the vehicle model are controlled by the driver model and include the brake, steering wheel and speed reference to the cruise control. The outputs that are generated are position, speed, heading and yaw rate.

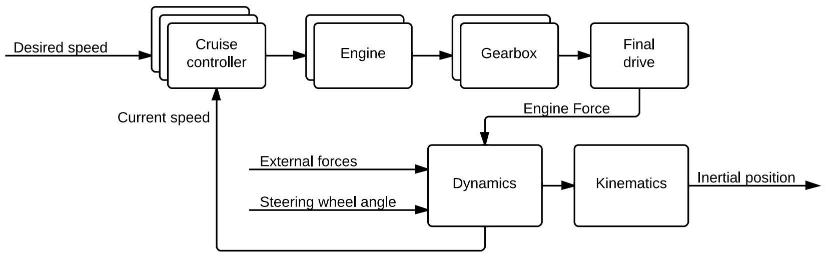

Figure 3.4: Diagrammatic representation of the vehicle model

The blocks representing the cruise controller, engine and gearbox can vary between the different models as described in Section 3.1. Thus, there are six different combinations with different complexities and behavior.

3.3.1 Model parameter values

The parameters in the model can differ between different vehicles. Most of the parameters are difficult to obtain for a certain vehicle, as it will require detailed knowledge, such as tires and load. However the range of parameters can be estimated for a typical truck. Those intervals are described in Table 3.1.

Table 3.1: *Typical truck parameters*

Parameter	**Symbol**	**Typical interval**	**Unit**
Vehicle mass	m	[10000 , 80000]	kg
Front area	A	[5 , 20]	m^2
Length from front to mass center	a	$[10, 20]$	m
Length from rear to mass center	b	$[5, 15]$	m
Total length of the truck	l	$[15, 35]$	m
Width of the truck	w	$[2, 3]$	m
Coefficient of rolling friction	c_{roll}	$[0.005, 0.01]$	-
Tire stiffness factor	B	$[0, 100]$	-
Tire shape factor	C	$[1.0, 2.0]$	-
Tire-road friction coefficient	μ	$[0.5, 1.0]$	-
Tire curvature factor	E	$[-50, 1]$	-
Air density	ρ	[1.0 , 1.4]	$\mathrm{kg\ m}^{-3}$
Gravitational acceleration	g	[9.780 , 9.832]	$\mathrm{m\ s}^{-2}$

4 Development of driver models

The implementation of driver model is described in this chapter. The purpose of driver model is to generate steering wheel and speed reference such that the vehicle follows a desired route and behaves legally in traffic. The methods for how the data from the server is used together with internal feedback for controlling the inputs to the vehicle model is presented in this chapter.

4.1 Steering wheel control

The major objective of the driver-truck model is to follow a path that is described by waypoints received from the server. The path generation is described in Section 2.6. A feedback controller is designed to calculate the steering wheel angle reference based on the error between the path and on estimated future positions of vehicle. An overview of the logic to control the steering wheel is presented in Figure 4.1

4.1.1 Preview Points

As the driver model needs information about the future of the road to react on the changes of curvature, so called "preview points" are calculated. The distance that is traveled between each preview point equals uT where u is the longitudinal velocity and T is the time difference between each preview point. The distances from the preview points to the path are then calculated as described in Section 4.1.2. The preview points are calculated assuming that the vehicles heading is constant as seen in Figure 4.2, thus the resulting equation for the points is

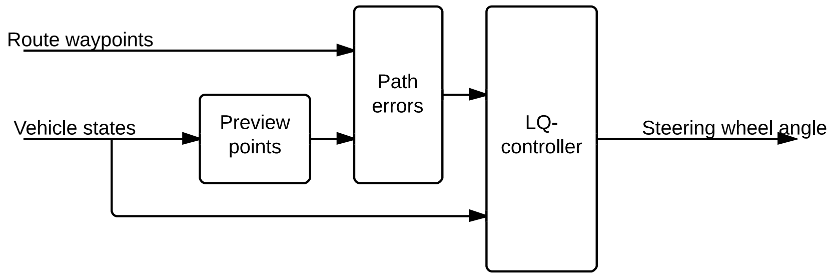

Figure 4.1: Overview of the logic for deciding the steering wheel angle

$$\mathbf{P_{T,n}} = \begin{bmatrix} x_{N,n} \\ x_{E,n} \end{bmatrix} = \mathbf{P_{T,0}} + n \cdot uT \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix} \tag{4.1}$$

where $\mathbf{P_{T,n}}$ is the expected position at nT seconds in the future, and $x_{N,n}$ and $x_{E,n}$ is the corresponding northern and eastern position. The current position is notated with $\mathbf{P_{T,0}}$.

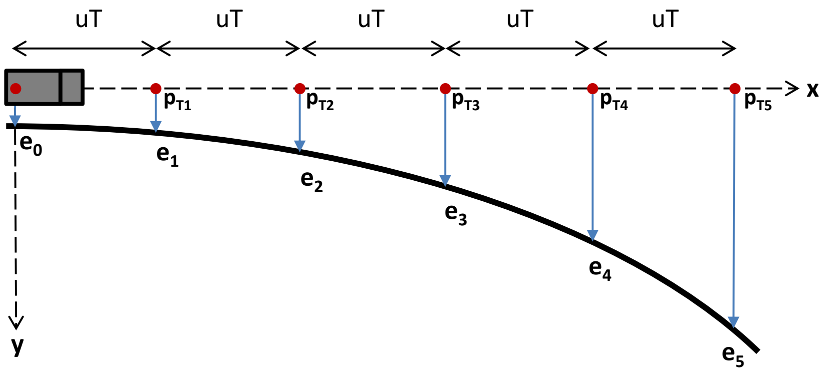

Figure 4.2: Corresponding errors for five preview points and the current position

4.1.2 Error calculation

To follow the road correctly, the driver-model needs information about the deviation from the desired path. For this the closest point on the path is needed, which can be adapted with the Newton-Raphson method.

Finding the relevant waypoints

Firstly, as the spline described in Section 2.6.1 is piecewise-defined between each waypoint, the four waypoints that defines the spline that includes the closest point must be found. For doing this, the closest waypoint is calculated using

$$\mathbf{W_0^*} = \mathbf{min_{W_i}} \|\mathbf{W_i} - \mathbf{P_T}\|^2, \tag{4.2}$$

where $\mathbf{W_i}$ denotes the waypoints received from the server, and the closest waypoint is notated with $\mathbf{W_0^*}$. Then the two neighboring waypoints, $\mathbf{W_{-1}^*}$ and $\mathbf{W_1^*}$ are used to create the vector $\mathbf{v} = (\mathbf{W_1^*} - \mathbf{W_{-1}^*})$ which is used as an estimate of the tangent vector in $\mathbf{W_i}$. By then calculating the scalar product of the vector from the target point to the closest waypoint and the estimated tangent vector with

$$\mathbf{C} = (\mathbf{W_1^*} - \mathbf{W_{-1}^*}) \bullet (\mathbf{W_0^*} - \mathbf{P_T}), \tag{4.3}$$

the searched waypoints can be found with the following criteria: If $\mathbf{C} < 0$, the closest point is assumed to belong between the closest waypoint and next following waypoint, and if $\mathbf{C} > 0$ it is assumed to belong between the previous waypoint and the closest waypoint. The four waypoints that defines the sought Catmull-Rom spline, $\mathbf{P_0}, \mathbf{P_1}, \mathbf{P_2}, \mathbf{P_3}$ will then be:

$$\begin{bmatrix}\mathbf{P_0} & \mathbf{P_1} & \mathbf{P_2} & \mathbf{P_3}\end{bmatrix}^T = \begin{cases} \begin{bmatrix}\mathbf{W_{-2}^*} & \mathbf{W_{-1}^*} & \mathbf{W_0^*} & \mathbf{W_1^*}\end{bmatrix}^T, & \mathbf{C} > 0 \\ \begin{bmatrix}\mathbf{W_{-1}^*} & \mathbf{W_0^*} & \mathbf{W_1^*} & \mathbf{W_2^*}\end{bmatrix}^T, & \mathbf{C} < 0 \end{cases} \tag{4.4}$$

First approximation of the closest point

In the normal case, the path will most often have a low curvature. Therefore, by creating a vector between $\mathbf{P_1}$ and $\mathbf{P_2}$ and project the target point on that vector, a good approximation of t in (2.8) for the Catmull-Rom can be found with

$$t^{*,0} = \frac{(\mathbf{P_T} - \mathbf{P_1}) \bullet (\mathbf{P_2} - \mathbf{P_1})}{|\mathbf{P_2} - \mathbf{P_1}|^2} \tag{4.5}$$

which equals the distance between $\mathbf{P_1}$ to the projection of the target point on the line between $\mathbf{P_1}$ to $\mathbf{P_2}$, divided by the length of that line. By then computing the Catmull-Rom formula for $t^{*,0}$ a good approximation of the closest point are adapted. In the special case when driving on a path with zero curvature, this estimation will equal the true closest point. For paths with more curvature, the Newton-Raphson method must be used.

The Newton-Raphson method

Firstly the function, $\mathbf{D}(t)$, describing the squared distance from the target point $\mathbf{P_T}$ and a point on the Catmull-Rom spline $\mathbf{S}(t)$, is created with

$$\mathbf{D}(t) = |\mathbf{S}(t) - \mathbf{P_T}(t)|^2 = \sum_{i=x,y} (\mathbf{S_i}(t) - \mathbf{P_{T,i}}(t))^2. \tag{4.6}$$

By minimizing $\mathbf{D}(t)$, a more precise approximation of the closest point can be found. The function can be minimized using the Newton-Raphson method, which is computed by repeatedly calculating a new t^* based on the previous result with

$$t^{*,n+1} = t^{*,n} - \frac{\mathbf{D}'(t)}{\mathbf{D}''(t)} \tag{4.7}$$

For a sufficiently well-chosen initial approximation, t^* will converge quadratically to an optimal t. By using the approximation from (4.5), the error can be expected to be insignificant after only a couple of iterations when the vehicle is driving on a highway, as the curvature are close to zero.

The derivatives $\mathbf{D}'(t)$ and $\mathbf{D}''(t)$ can be calculated from (4.6), which are given by

$$\mathbf{D}'(t) = \sum_{i=x,y} \frac{d}{dt}(\mathbf{S_i}(t) - \mathbf{P_{v,i}}(t))^2 = 2\sum_{i=x,y} \mathbf{S_i}'(t)(\mathbf{S_i}(t) - \mathbf{P_{v,i}}(t)), \tag{4.8}$$

and

$$\mathbf{D}''(t) = 2\sum_{i=x,y} \frac{d}{dt}\mathbf{S_i}'(t)(\mathbf{S_i}(t) - \mathbf{P_{v,i}}(t)) = 2\sum_{i=x,y} \mathbf{S_i}''(t)(\mathbf{S_i}(t) - \mathbf{P_{v,i}}(t)) + \mathbf{S_i}'(t)^2, \tag{4.9}$$

where $\mathbf{S}'(t)$ and $\mathbf{S}''(t)$ can be found by differentiating the equation for the Catmull-Rom spline (2.8)), given by:

$$\mathbf{S}'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \mathbf{M_C} \begin{bmatrix} \mathbf{P_0} \\ \mathbf{P_1} \\ \mathbf{P_2} \\ \mathbf{P_3} \end{bmatrix} \tag{4.10}$$

and

$$\mathbf{S}''(t) = \begin{bmatrix} 6t & 2 & 0 & 0 \end{bmatrix} \mathbf{M_C} \begin{bmatrix} \mathbf{P_0} \\ \mathbf{P_1} \\ \mathbf{P_2} \\ \mathbf{P_3} \end{bmatrix} \tag{4.11}$$

In the control algorithms described in Section 4.1.3 only the lateral devation in the vehicles local coordinate system are used. The lateral deviation can be calculated with

$$e = \hat{\mathbf{y}} \bullet (\mathbf{S}(t^*) - \mathbf{P_T}) \tag{4.12}$$

where $\hat{\mathbf{y}}$ is a normalized vector representing the vehicles lateral direction, i.e

$$\hat{\mathbf{y}} = \begin{bmatrix} -\sin(\psi) \\ \cos(\psi) \end{bmatrix} \tag{4.13}$$

where ψ is the heading of the vehicle.

For the preview points it is considerable to not use the iterations in (4.7), and thereby trusting that the first estimated point caculated in (4.5) is good enough for controlling the vehicle. This would remove computational complexities, which could be necessary if a large amount of preview points is used. However, this method may not be robust, as there are no proof that the first estimate of the closest point is close enough to the true closest point. Also note that the current error must be calculated with at least a few Newton-Raphson iterations, as it is important that the estimation of the current lateral error is similar to the true error, as it are used for reviewing the control algorithm.

4.1.3 Control method

To actually control the steering wheel to follow the path from the calculated errors there are number of different methods to choose between. In the classical control theory there are three main strategies that could be used for the current problem, as described in Ljung and Glad [2003]. These are the PID-, LQR-, and MPC-controllers. In Naranjo et al. [2005], fuzzy logic have been used with good result, and there exists numerous other methods that are similar to the ones that have been mentioned here.

In this thesis two different methods have been examined. Firstly, a PID-controller has been used, which is a simple method where the parameters are chosen ad hoc. Secondly a more complex method with a LQ-regulator is implemented, which is less time consuming than a MPC-controller and have a more straight-forward use of the preview points compared to fuzzy logic.

PID-controller with preview points

The PID-controller calculates the steering wheel angle depending on the proportional, integral and deravative values of the feedback signals. The feedback will contain the current lateral error, the current heading error, and the lateral errors in the preview points. The output value of the steering wheel is

$$\begin{aligned}\delta(t) = K_{P,\psi} e_{\psi}(t) + K_{I,\psi} \int_0^t e_{\psi}(\tau)\, d\tau + K_{D,\psi} \frac{d}{dt} e_{\psi}(t) + \\ \sum_{i=0}^{n} \left(K_{P,i} e_i(t) + K_{I,i} \int_0^t e_i(\tau)\, d\tau + K_{D,i} \frac{d}{dt} e_i(t)\right)\end{aligned} \tag{4.14}$$

where $K_{P,i}$, $K_{I,i}$, $K_{D,i}$ are parameters for the different proportional, integral and derivative controllers respectively, and n is the number of preview points. The output is also saturated to avoiding unrealistic angles on the steering wheel.

The parameters is chosen ad hoc based on basic knowledge of the functionality of each parameter. The number of preview points for this method should preferably be very low to be able to make correct assumptions for each parameter. Hence, the implemented PID-controller only uses one preview point. However, to few preview points may negatively effect the performance of the method. Therefore the LQ-regulator has been implemented which is able to use more of the preview points.

Linear-quadratic regulator with preview points

A linear-quadratic regulator is an optimal controller that uses the systems dynamics to minimize a cost described by a quadratic function containing the different errors and input energy. The quadratic function can be weighted in such way that prioritized errors are more significant. In this thesis a method inspired by Sharp and Valtetsiotis [2001] has been used, where the state of the vehicle's motion is used together with the preview errors, to compute an optimal angle for the steering wheel that minimizes the error in heading and lateral position.

To be able to use LQR a linear model of the system is needed. By linearising the lateral forces and slip angles a linear model is adapted for slip angles, lateral velocities, yaw rate and the steering wheel angle close to zero. The linearisation is done as follows:

The first degree Maclaurin expansion of the slip angle for the front axle is defined in (3.12):

$$\alpha_f \approx \frac{u \sin 0 - (v + a\dot{\psi}) \cos 0}{u \cos 0 + (v + a\dot{\psi}) \sin 0} +$$

$$(1 - \frac{(u \sin 0 - (v + a\dot{\psi}) \cos 0)(-u \sin 0 + (v + a\dot{\psi}) \cos 0)}{(u \cos 0 + (v + a\dot{\psi}) \sin 0)^2}) \cdot \delta \tag{4.15}$$

$$= -\frac{(v + a\dot{\psi})}{u} + (1 + \frac{(v + a\dot{\psi})^2}{u^2})\delta \tag{4.16}$$

and by linearising at $v = 0$ and $\dot{\psi} = 0$

$$\alpha_f \approx \delta - \frac{(v + a\dot{\psi})}{u}. \tag{4.17}$$

The linearised equation for slip angle for the rear axle is

$$\alpha_r = \frac{b\dot{\psi} - v}{u} \tag{4.18}$$

which is equal to (3.13), as it already is linear. The fact that $\arctan(x) \approx x$ and $\sin(x) \approx x$ when x is close to zero, can be used to linearise the the Pacejka formula (3.11) with

$$\begin{aligned} F_{yi} &= 2D \sin\{C \arctan[B(1 - E)\alpha_i + E \arctan(B\alpha_i)]\} \\ &\approx 2D \sin\{C \arctan[B(1 - E)\alpha_i + E(B\alpha_i)]\} \\ &= 2D \sin\{C \arctan[B\alpha_i]\} \\ &\approx 2D \sin\{BC\alpha_i\} \\ &\approx 2BCD\alpha_i \\ &= K\alpha_i, \qquad (K \equiv 2BCD), \end{aligned} \tag{4.19}$$

for small slip-angles. By combining (3.4) and (3.5) for the vehicle's lateral dynamics, together with the linearised equations for the lateral forces (4.19),

$$\dot{v} = \frac{K\alpha_f}{m} + \frac{K\alpha_r}{m} - u\dot{\psi} \tag{4.20}$$

and

$$\ddot{\psi} = a\frac{K\alpha_f}{I_z} - b\frac{K\alpha_r}{I_z} \tag{4.21}$$

By adding the linear equations for the slip angles, (4.17) and (4.18), a complete linear model of the vehicle's lateral motion can be derived:

$$\dot{v} = -(\frac{K(a-b)}{mu} + u) \cdot \dot{\psi} + \frac{K}{mu} \cdot \delta \tag{4.22}$$

$$\ddot{\psi} = -\frac{K(a+b)}{I_z u} \cdot \dot{y} - \frac{K(a^2+b^2)}{I_z u} \cdot \dot{y} + \frac{Ka}{Iz} \cdot \delta \tag{4.23}$$

The linearised model described by (4.22) and (4.23) can then be written as a linear state space model, $\dot{\mathbf{X}}_\mathbf{v} = \mathbf{A}_\mathbf{v}\mathbf{X}_\mathbf{v} + \mathbf{B}_\mathbf{v}\delta$, with

$$\mathbf{A}_\mathbf{v} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -(\frac{K(a-b)}{mu} + u) \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{K(a+b)}{I_z u} & 0 & -\frac{K(a^2+b^2)}{I_z u} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ \frac{K}{mu} \\ 0 \\ \frac{Ka}{Iz} \end{bmatrix} \tag{4.24}$$

and

$$\mathbf{X}_\mathbf{v} = \begin{bmatrix} y & v & \psi & \dot{\psi} \end{bmatrix}^T \tag{4.25}$$

which can be converted to the discrete form $\mathbf{X}_\mathbf{d}(k+1) = \mathbf{A}_\mathbf{d}\mathbf{X}_\mathbf{d}(k) + \mathbf{B}_\mathbf{d}\delta(k)$ with the MATLAB function `c2d`.

The preview points errors can be written as a discrete state space model $\mathbf{X}_\mathbf{p}(k+1) = \mathbf{A}_\mathbf{p}\mathbf{X}_\mathbf{p}(k) + \mathbf{E}_\mathbf{p}\mathbf{n}_\mathbf{p}(k)$ with

$$\mathbf{A}_\mathbf{p} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \quad \mathbf{E}_\mathbf{p} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{4.26}$$

and

$$\mathbf{X}_\mathbf{p} = \begin{bmatrix} e_0 & e_1 & \dots & e_n \end{bmatrix}^T \tag{4.27}$$

if the discrete sampling time is the same as the preview time. This will result in that the second preview point at instant k are shifted to the first preview point at instant $(k+1)$, and similarly for the rest of the preview points. The lateral error for the last preview point at instant $(k+1)$, are an unknown input at instant k and is therefore handled as gaussian noise, notated with n_p.

By combining the discrete state space models for the vehicle's lateral motion and the previewed lateral errors, a complete model for the control problem can be adapted as

$$\begin{aligned}\mathbf{z}(k+1) &= \mathbf{A}\mathbf{z}(k) + \mathbf{B}\delta(k) + \mathbf{E}\mathbf{n_p}(k)\\ \mathbf{\Gamma}(k) &= \mathbf{C}\mathbf{z}(k)\end{aligned} \tag{4.28}$$

with

$$\mathbf{A} = \begin{bmatrix}\mathbf{A_d} & \mathbf{0}\\ \mathbf{0} & \mathbf{A_p}\end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix}\mathbf{B_d}\\ \mathbf{0}\end{bmatrix}, \qquad \mathbf{E} = \begin{bmatrix}\mathbf{0}\\ \mathbf{E_p}\end{bmatrix}, \tag{4.29}$$

$$\mathbf{C} = \begin{bmatrix}1 & 0 & 0 & 0 & -1 & 0 & 0 & \dots & 0\\ 0 & 0 & 1 & 0 & -1/uT & 1/uT & 0 & \dots & 0\end{bmatrix}^T, \tag{4.30}$$

and

$$\mathbf{z} = \begin{bmatrix}y & v & \psi & \dot{\psi} & e_0 & e_1 & \dots & e_n\end{bmatrix}^T \tag{4.31}$$

By setting $\mathbf{\Gamma}(k) = \mathbf{C}\mathbf{z}(k)$ a connection between the path and the vehicle model is adapted, where the first row in $\mathbf{\Gamma}(k)$ describes the current lateral deviation between the path and the road and the second row describes the current heading error.

The quadratic criteria

$$\mathbf{J} = \sum_{i=x,y}(\mathbf{\Gamma^T}(k)\mathbf{R_1}\mathbf{\Gamma}(k) + \mathbf{r_2}\delta^2(k)), \tag{4.32}$$

with

$$\mathbf{R_1} = \begin{bmatrix}r_{11} & 0\\ 0 & r_{12}\end{bmatrix} \tag{4.33}$$

provides the possibility to set a balance between lateral deviation, heading error, and the angle of the steering wheel, by varying r_{11}, r_{12} and r_2. By setting

$$\mathbf{Q} = \mathbf{C^T}\mathbf{R_1}\mathbf{C}, \tag{4.34}$$

the criteria can be rewritten with

$$\mathbf{J} = \sum_{i=x,y} (\mathbf{z}^{\mathbf{T}}(k)\mathbf{Q}\mathbf{z}(k) + \mathbf{r}_2\delta^2(k)), \tag{4.35}$$

and the optimal $\delta(k)$ that minimizes $\mathbf{J}$ is then given by

$$\delta(k) = -\mathbf{F}\mathbf{z}^{\mathbf{T}}(k) \tag{4.36}$$

where the optimal gain, $\mathbf{F}$ is adapted from

$$\mathbf{F} = (\mathbf{B}^{\mathbf{T}}\mathbf{P}\mathbf{B} + \mathbf{R}_2)^{-1}\mathbf{B}^{\mathbf{T}}\mathbf{P}\mathbf{A} \tag{4.37}$$

and $\mathbf{P}$ can be adapted by solving the discrete algebraic Ricatti Equation:

$$\mathbf{P} = \mathbf{Q} + \mathbf{A}^{\mathbf{T}}(\mathbf{P} - \mathbf{P}\mathbf{B}(\mathbf{B}^{\mathbf{T}}\mathbf{P}\mathbf{B} + \mathbf{r}_2)^{-1}\mathbf{B}^{\mathbf{T}}\mathbf{P})\mathbf{A} \tag{4.38}$$

which can be solved by iterating the dynamic Ricatti Equation

$$\mathbf{P}_{\mathbf{n+1}} = \mathbf{Q} + \mathbf{A}^{\mathbf{T}}(\mathbf{P}_{\mathbf{n}} - \mathbf{P}_{\mathbf{n}}\mathbf{B}(\mathbf{B}^{\mathbf{T}}\mathbf{P}\mathbf{B} + \mathbf{r}_2)^{-1}\mathbf{B}^{\mathbf{T}}\mathbf{P}_{\mathbf{n}})\mathbf{A} \tag{4.39}$$

for $\mathbf{P}_{\mathbf{0}} = \mathbf{Q}$ until it converges.

Implementation of the LQ-controller

The calculation of the optimal gain, **F**, would be too time consuming to be done for each time step. Therefore it are calculated once when the model is initialized. However, as the optimal gain depends on the current speed, it is calculated for a number of different speeds to create a look-up table for each element in **F**. By then adding the current speed as an input to the controller, the optimal gain can be interpolated and multiplied with the state feedback to compute the steering wheel angle as in (4.36). Finally the output are saturated for avoiding unrealistic values on the steering wheel.

4.2 Speed control

The speed reference that are sent to the cruise controller are based on a number of factors. Firstly all heavy trucks must drive beneath a maximum velocity, V_{max}, which in the Europe region is 90 km/h. The vehicle speed will then also be limited by the actual speed limit of the road that it is driving on. The third limitation is the traffic, as the vehicle must keep a speed that is slow enough for not driving into other vehicles. From all these criteria the lowest limit are passed as the desired speed. Also, a low-pass filter is added to the speed reference, which prevents overshoots in the cruise controller. An overview of the logic for deciding the speed reference is presented in Figure 4.3.

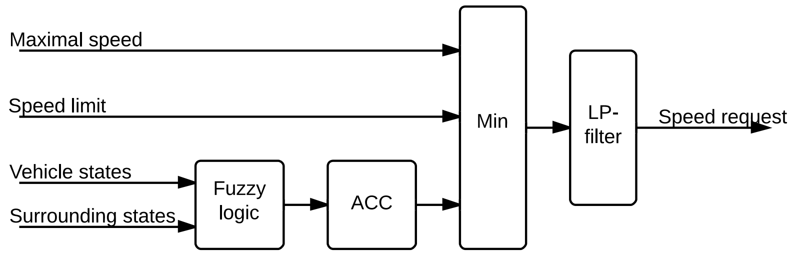

Figure 4.3: *Overview of the logic for deciding which speed reference that should be sent to the cruise controller*

4.2.1 Speed limit

The model will get information about the speed limit for each of the ten waypoints that is received from the server. As the vehicle will need time to decelerate, the lowest speed limit for these points are used. This will result in that the vehicle aregin to decelerate before the a waypoint with lower speed is passed, but begin to accelerate exactly when a waypoint with higher speed is passed.

4.2.2 Vehicle following

Another important objective for the model is avoiding driving into other vehicles. In this thesis the model will solve this by following possible slower vehicles ahead with a predefined gap distance. Another possible method could be to overtake such vehicles. However more information about the surrounding traffic is needed to make such decisions. Therefore the overtaking scenario has not been investigated in this thesis. The basics of the used method for vehicle following is to estimate which vehicles that needs to be considered, and then adaptively choose a speed reference for the cruise controller depending on the position and speed of the targeted vehicle.

Transformation to path coordinates

As described in Section 2.3.4 the server provides the vehicle with relative positions to the four closest vehicles ahead. The distances are defined in the ego vehicle's local coordinate system where dx represents the longitudinal distance, and dy represents the lateral distance, with positive values at the left of the vehicle. However, the ego vehicle should only track the vehicles that drives in the same traveling path. Therefore the local distances are transformated to the traveling path coordinates, to be able to decide which vehicles which is relevant.

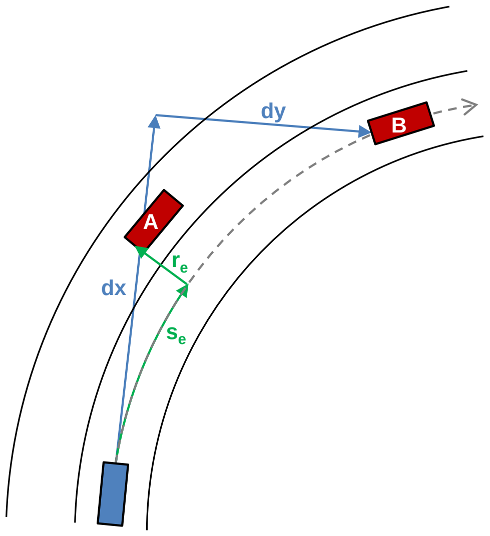

Figure 4.4: *Distances to vehicle A in the ego vehicle's traveling path coordinates and distances to vehicle B in the ego vehicle's local coordinate system*

The current curvature, C_p, of a traveling vehicle can be calculated from the vehicle's yaw-rate and longitudinal velocity with

$$C_p = \frac{\dot{\psi}}{u} \tag{4.40}$$

By assuming that the vehicle drives with a constant curvature at each time step, the traveling path can be estimated using a circle with a radius of

$$R_p = \frac{1}{C_p} = \frac{u}{\dot{\psi}} \tag{4.41}$$

By creating an angle, ζ, between the origin of the circle and the two vehicles, the distances s_e along the path, and r_e out from the path, can be calculated. The angle is defined as

$$\zeta = \tan^{-1}(\frac{dx}{\chi}) \tag{4.42}$$

where

$$\chi = \begin{cases} -(R + dy) & , R < 0 \\ R + dy & , R > 0 \end{cases} \tag{4.43}$$

which, together with the fact that dx always is positive, and $\tan^{-1}$ is monotonic, means that

$$\zeta = \begin{cases} -\tan^{-1}(\frac{dx}{R+dy}) & , R < 0 \\ \tan^{-1}(\frac{dx}{R+dy}) & , R > 0 \end{cases} \tag{4.44}$$

the distance along the path can then be calculated with

$$s_e = |R|\zeta = R\tan^{-1}(\frac{dx}{R + dy}) \tag{4.45}$$

The distance along the normal of the path is from the figure given from the definition of sinus as following

$$\cos(\zeta) = \begin{cases} \frac{\chi}{R+r_e} & , R < 0 \\ \frac{\chi}{-R-r_e} & , R > 0 \end{cases} \tag{4.46}$$

and thanks to that χ is an odd function, the resulting $\cos(\zeta)$, which are equal for all R, thus can be simplified to

$$\cos(\zeta) = \frac{R + dy}{R + r_e}, \tag{4.47}$$

where

$$r_e = \frac{R + dy}{\cos(\zeta)} - R. \tag{4.48}$$

By using the fact that ζ is an odd function and $\cos(\zeta)$ is an even function,

$$r_e = \frac{R + dy}{\cos[\tan^{-1}(\frac{dx}{R+dy})]} - R. \tag{4.49}$$

Note that for the special case where $\dot{\psi} = 0$, which is equivalent to $R \to \infty$, the vehicle's traveling path is straight, which means $s_e = dx$ and $r_e = dy$. This can be proven by calculating the limits of (4.45) and (4.49) for $R \to \infty$.

Lane probability from fuzzy logic

By using r_e and s_e the vehicle can characterize which of the surrounding vehicles it should be tracking. The intuitive method is to compare r_e with a limit value, and from that estimate if the targeted vehicle is driving in another lane than the ego vehicle, and if it can be passed by continue to drive in the current lane. Unfortunately, it is not possible to define an exact value for the limit on r_e. Of course r_e must be larger than the width of the vehicles, and extra safety distance between the vehicles is needed. In addition, the estimation of r_e will have some uncertainty, and finally, one or more vehicles may be in the progress of changing lanes. This results in a more complicated problem. In this thesis, fuzzy logic has been used to estimate the probability for the targeted vehicle being in the same lane as the ego vehicle. The method is similar to Moon et al. [2009], but with some different logic rules and use of the probability value.

Table 4.1: *Rule set for the fuzzy logic, where the structure for each rule is:* `IF C1 AND C2 THEN O(Ri)`

Rule	$C_1 = r_e$	$C_2 = d\|r_e\|/dt$	$O(R_i)$(in lane)
R_1	VERY HIGH		NO, 0.0
R_2	VERY LOW		YES, 1.0
R_3	HIGH	NEGATIVE	MAYBE, 0.5
R_4	HIGH	ZERO	NO, 0.0
R_5	HIGH	POSITIVE	NO, 0.0
R_6	LOW	NEGATIVE	YES, 1.0
R_7	LOW	ZERO	MAYBE, 0.5
R_8	LOW	POSITIVE	NO, 0.0

The method is based on a number of logical rules listed in 4.1. Each rule has two conditions, the first condition is on the lateral distance, and the second condition is on the derivative of the lateral distance. However, instead of using discrete values, continuous values between 0 and 1 are computed depending on how much the conditions are satisfied. The resulting value is called the "truth value", $\mu(R_i)$, which for each rule R_i can be calculated by minimum inference where the truth value will be equal to the minimum of the mapped values for each condition. The inference can be described with

$$\mu(j) = min\{M(C_1, j), M(C_2, j)\} \tag{4.50}$$

where $M(C_i, j)$ decl ares the mapped value for condition C_i in rule R_j. The mapped values for lateral distance and its derivative are presented in Figure 4.5. Each rule will also have an output value, $O(R_i)$, which are 0 if the rule says that the vehicle is outside the lane, 1 if it says that the vehicle is outside the lane, and 0.5 if there is an uncertain case. By using so called centroid defuzzication, the truth and output values for each rule can be combined to compute a single value that describes the position of the target vehicle. The centroid defuzzication is computed by calculating the "center of gravity" with

$$P_{lane} = \frac{\sum_{R_i} \mu(R_i)O(R_i)}{\sum_{R_i} \mu(R_i)} \tag{4.51}$$

The resulting value, P_{lane}, is then used to represents the probability for the target vehicle being in the same lane as the ego vehicle.

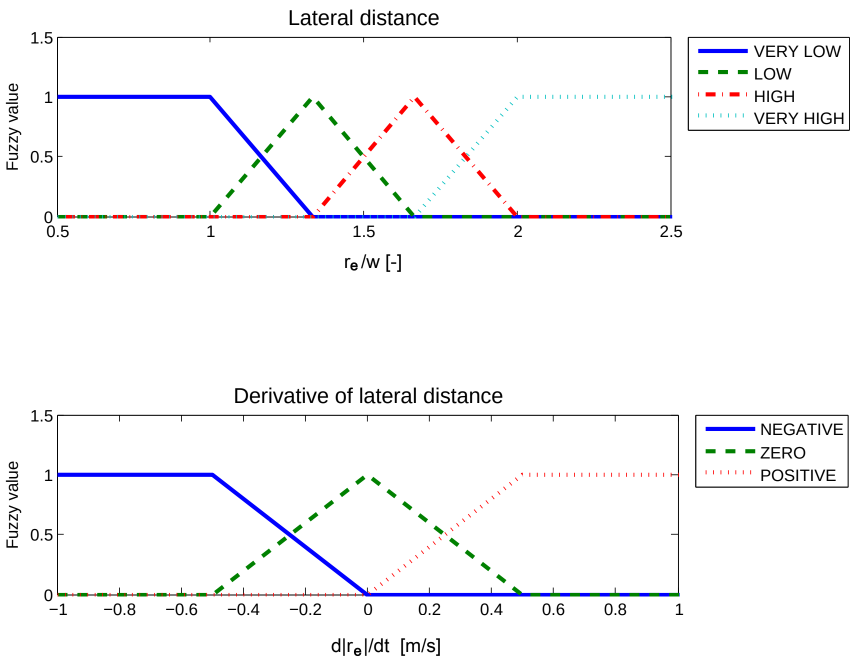

Figure 4.5: *Fuzzy values*

Adaptive cruise controller based on fuzzy logic

The adaptive cruise controller (ACC) are based on the relative velocity and position of the target vehicle and the ego vehicle. The reference for the relative velocity are 0, thus making the vehicle drive in the same speed as the preceding vehicle. The relative distance will have predefined gap distance in seconds as a reference, T_{ref}, that can be chosen by the server. The current gap time is found by dividing the relative distance, s_e, by the current velocity, u, while the relative velocity is calculated by deriving s_e in time. The errors is then computed by subtract the reference values, and then be multiplied by the in lane probability. The resulting formula for the errors is:

$$e_{u,i} = P_{lane,i}\dot{s}_{e,i} \tag{4.52}$$

for the velocity and

$$e_{s,i} = P_{lane,i}(\frac{s_{e,i}}{u} - T_{ref}) \tag{4.53}$$

for the longitunal gap time. The errors are then given to a proportional controller with following structure

$$u_{acc,i} = \begin{cases} u + K_u e_{u,i} + K_s e_{s,i} & , s_e < s_{max} \\ V_{max} & , s_e > s_{max} \end{cases} \tag{4.54}$$

where s_{max} represent a maximum distance when the target vehicle is handled by the ACC. The final speed reference from the ACC is

$$U_{acc} = min\{u_{acc,i}\}. \tag{4.55}$$

5 Results and discussions

This chapter presents the results in the different computational time when varying the complexities for the relevant parts of the model. There are also results in the driver's ability to follow the route and behave legally together with the surrounding traffic.

5.1 Evaluation in simulink

The simulation models are evaluated in Simulink. The Simulink models provide all simulated data which has been sent to the Matlab workspace for further analysis. It is also possible to measure the elapsed time for each simulation, which have been used for analyzing the differences in computational efficiency for the different vehicle models.

5.1.1 Vehicle model

When the vehicle model is evaluated there are two properties that are investigated. Firstly, it is important that the vehicle model reacts in the same way as a real truck. This could have been validated by testing different inputs on a real truck and comparing the results with the behavior of the implemented models for the same inputs. This method is not used, as it would be very time consuming and is out of the scope of this project. However, since the vehicle models are developed based on tested models from Scania and can be verified using the laws of physics, it is believed that the models developed here are well posed. The second property is simulation time which varies between the different combinations in the powertrain and cruise controller (CC) described in Section 3.1. By testing different combinations simulation time, it is possible to receive an approximation of the maximal amount of trucks that are possible to simulate together in real time.

The resulted approximations is presented in Table 5.1. Note that the approximations are far from exact values, but still provides good comparison. There are also reasons to be believe that these numbers would be higher when the models are compiled and executed by the server, however it is noted that the Scania Active Prediction model is too complex to be used for the purpose of the thesis.

Table 5.1: *Approximation of maximal numbers of vehicles possible to simulate for different powertrain combinations*

CC and gear shift logic	Fuel consumption	Number of trucks
Scania Active Prediction	Yes	2
Scania Active Prediction	No	2
Simple Scania CC	Yes	22
Simple Scania CC	No	23
Simple own CC	Yes	26
Simple own CC	No	28

5.1.2 Driver model

The driver model has been tested by feeding the model with data that follows the same format as the data that is received from the server when the whole system is in use. The model have been tested in its ability to follow the road by using the different steering control methods, and also in how it handles the speed according to the speed limit and the surrounding traffic.

The test route

The driver model been simulated to drive on the highway (E4) between Södertälje and Nyköping by being fed with GPS-waypoints describing that road. As described in Chapter 2 the received route data should also include the altitude, number of lanes, speed limit and road type. These have however not been adapted from the reality for the simulink simulations, but have been randomly generated or chosen directly for intended purposes.

Steering wheel controller

In Figure 5.1 the lateral deviation has been plotted for the first 200 seconds in a simulation where one version of the LQ-regulator has been in use. The target is obviously to always stay inside the driving lane. For a common case on a highway where the lane has a width of 4 meters and the truck has a width of 2.5 meters, the critical value for the deviation is at 0.75 meters, which are used for validating the methods in the thesis. As seen in the figure the deviation varies between 0 and 1 meter, which is not really as good as preferred.

In the histogram in Figure 5.2 the distribution of the lateral deviation is presented for the same simulation as in Figure 5.1. However, the histogram has been computed from the whole simulation of 1000 seconds, i.e. 100 000 samples. By looking in the histogram it's evident that, even though the vehicle sometimes deviates

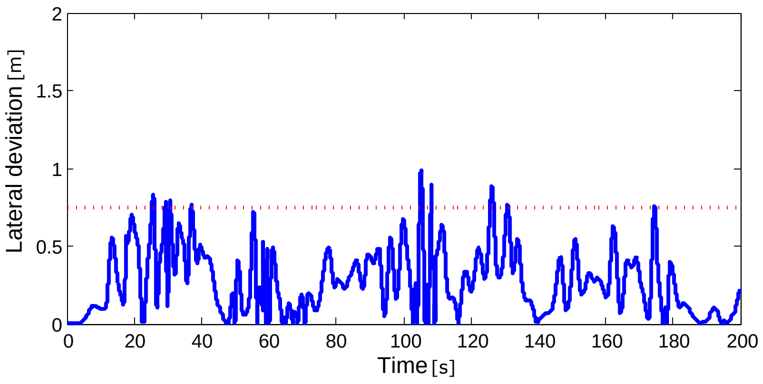

Figure 5.1: *Lateral deviation for a simulation of 200 seconds*

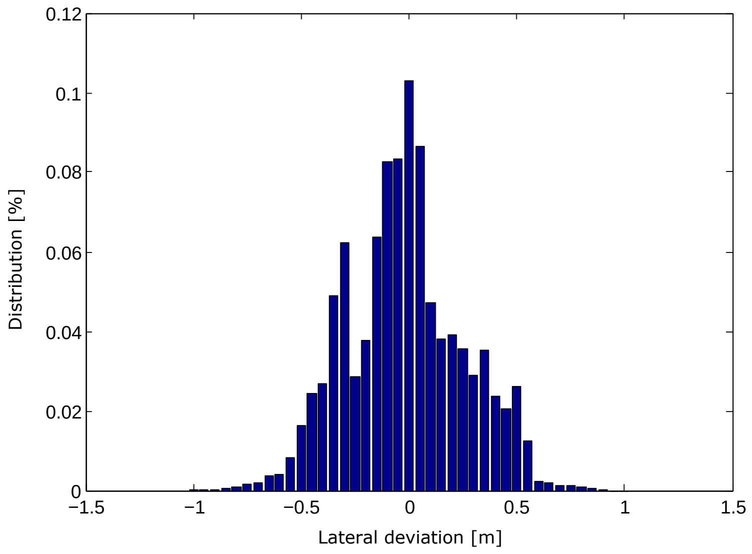

Figure 5.2: *Distribution of the lateral deviation for a simulation of 1000 seconds (100 000 samples), normalized to percentage of time.*

more than what is defined as critical, it only happens for a few samples. By looking on the percentage of when the vehicle deviates more than 0.75m, a suitable performance measurement value is received. Also, it is interesting to evaluate the mean square error (MSE) and root mean square (RMS).

By varying the tunable parameters in both the PID-controller and the LQ-regulator, it is possible to analyze which parameters yield the best performance by observing the performance measurement values presented earlier. The parameters and its simulated performance is presented for both the PID-controller and LQ-regulator in Appendix A.

By comparing the performance for the two different control methods, some observations can be done. Because of the high amount of parameters in the PID-controller, it is difficult to observe an obvious pattern for how the parameters affect the performance. In this study, PID controller with only one preview point is considered, as the number of tuning parameters increase drastically with increasing number of preview points. It is clear from Table A.1 that the LQ regulator performs better than the PID controller. The best tuned LQ only deviates above the critical value for less than 1% of the time, compared to above 4% for the best tuned PID. It is noted that the worst deviations for PID controlled vehicles are typically lower than the LQ regulator. However, the deviations of the PID decreases slower than the LQ. Furthermore, MSE and RMS penalize large deviation more heavily. Hence, in some cases, the MSE and RMS of the PID and LQ are comparable.

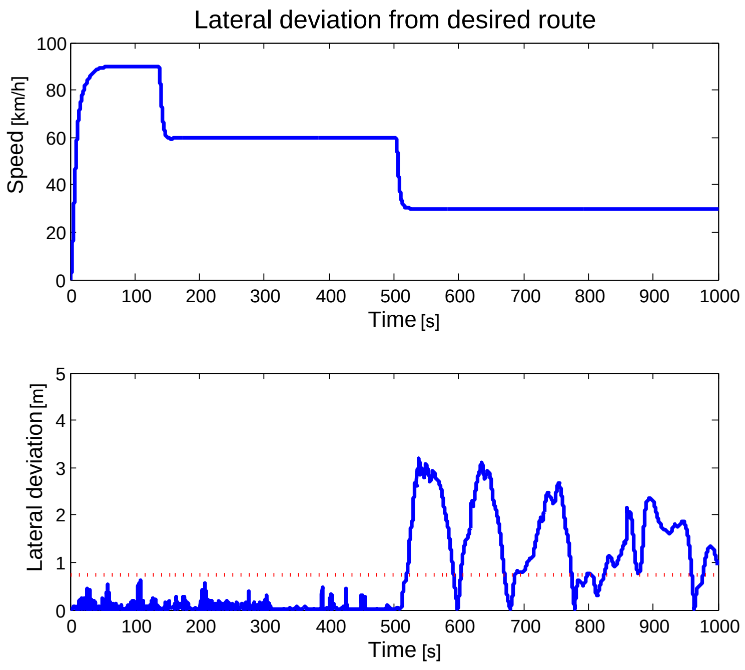

Figure 5.3: *Plot of speed profile and lateral deviation for a simulation with a LQ-regulator with $T_P = 0.4$*

For the LQ-regulator some interesting patterns can be found. When the preview time is low (0.4 seconds) the performance is bad. By looking closer, and plot the deviation together with the speed profile, an explanation can be found. In Figure 5.3 it is clear that the deviation is fine until the speed reaches 30 km/h, where the deviation suddenly reaches several meters. The reason is probably that the resulted preview distance is too short, and don not provide enough information of the future curvature of the road. In the same way the performance for long preview time is worse when driving in high speed, where the preview distance is too long. This may be solved by using the distance rather than time when calculating preview points.

Speed limit following

The speed limit following has been tested by generating a speed limit profile where the limit changes by arbitrary time intervals. The speed limit profile may not be realistic for a normal highway, but suits the purpose of the analysis. The resulted driving speed for a 1000 seconds simulation is plotted together with the actual speed limit in Figure 5.4.

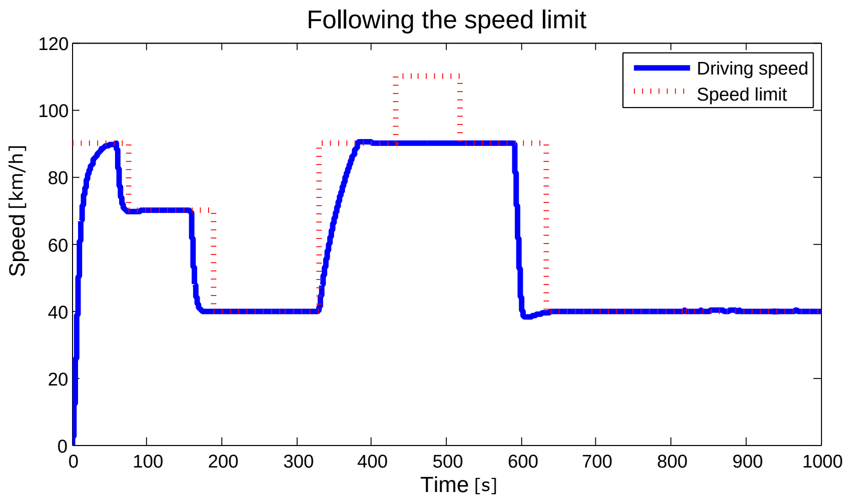

Figure 5.4: *Performance for following the speed limit*

As seen the driving speed is always close to or below the actual speed limit. However the speed profile is not optimal, as the deceleration time depends on the current speed and the step size for the desired speed. This results in that the driver-truck model drives in the lower speed too early. This is obvious when the speed limit changes from 90 km/h to 40 km/h, where the actual speed are 40 km/h approximately 20 seconds before it is supposed to be so. This is not a major problem as the driver-truck model still drives in a legal speed profile. However it results in a longer driving time than what is preferred. The error is caused by shortcoming of the speed limit following logic. The expected time to reach the speed limit is not taken into account, and the time for changing the speed is directly affected by the number of waypoints received. As a result, excessive number of waypoints received can lead to earlier deceleration and larger error.

It can also be noted that the driver does not reach the speed limit for the case of a limit on 110 km/h, which is consistent with the rule on the driver-truck model for not being allowed to drive in higher speed than 90 km/h.

Vehicle following

As the vehicle interaction is dependent on the server, it is not optimal to test the vehicle following without incorporating the server. However the Matlab/Simulink-environment can still provide a useful platform for data analysis, therefore a simple test scenario without the server has been used. The scenario is created by first simulating one vehicle, driving alone with a speed reference of 70km/h, and then recording its positions as a Matlab-timeseries. Then a second vehicle has been simulated alone, but by using the recorded positions, the longitudinal and lateral differences have been generated and fed to the second vehicle. In that way a simple version of the server interaction is simulated. The resulting driving profile is presented in Figure 5.5.

As seen in the figure the performance of the ACC is satisfying. The vehicle is driving faster than the preceding vehicle in the beginning, but slows down as it is getting closer. In the end it is keeping a gap time of the desired one second and performs platooning. The resulted air drag reduction is presented in Figure 5.6. The ACC could however have a little faster performance. In the figure it is clear that the following vehicle notices the other vehicle 90 seconds into the simulation but the desired gap time is not reached until 200-400 seconds into the simulation. This is partially intended, as the model is primarily designed for minimizing the risk of driving into other vehicles rather than perform platooning. It can also be noted that the lane probability is bouncing between 0 and 1 when the vehicle are getting closer. This may look strange, but can be explained by the fact that the distance at that moment is too big for reasonable results in the lane probability calculations.

5.2 Multiple driver-truck models with interactions

For the final implementation that is compiled and used in the actual system with the server interactions, the vehicle model with the fastest simulation time is used (simple PID cruise controller without fuel consumption calculations). This provided a light model that could be duplicated multiple times and therefore be used to create a good proof of concept of a platooning coordination system. The model is module based and implemented in Simulink and can be extended with more complex functions in the future. For the driver model, the LQR controller is used because of the better results in the evaluation.

From a stress-test by the server, a new approximation of the maximal amount of trucks simulated together in real time could be provided. In this way the result is around 50 trucks. It is not clear if the limit is on the memory or the computational power of the used computer. However the result proved that the model could be used for the intended purpose of simulating a big amount of virtual vehicles to create a simulation environment for platooning coordination.

For the final test, a scenario with two or three virtual trucks is used. They start at the same place, but at different time and with different speed. The intention is that the vehicles shall reach each other and then perform platooning. The result is evaluated by watching the real time visualisation that is developed as the third part of the project. It is clear that the vehicles follow the roads as intended, even though in some cases some vehicles deviate from the road. That can however be explained by the fact that the roads in the visualisation are linearly interpolated between the waypoints, unlike the solution used by the driver model. The speed controller also works as in the Simulink simulations, even though the real time visualisation makes it more obvious that the vehicle following controller is too slow to be perfectly satisfied. A test scenario with a real truck were also planned for the future, but could not be done within the limited time period of the thesis.

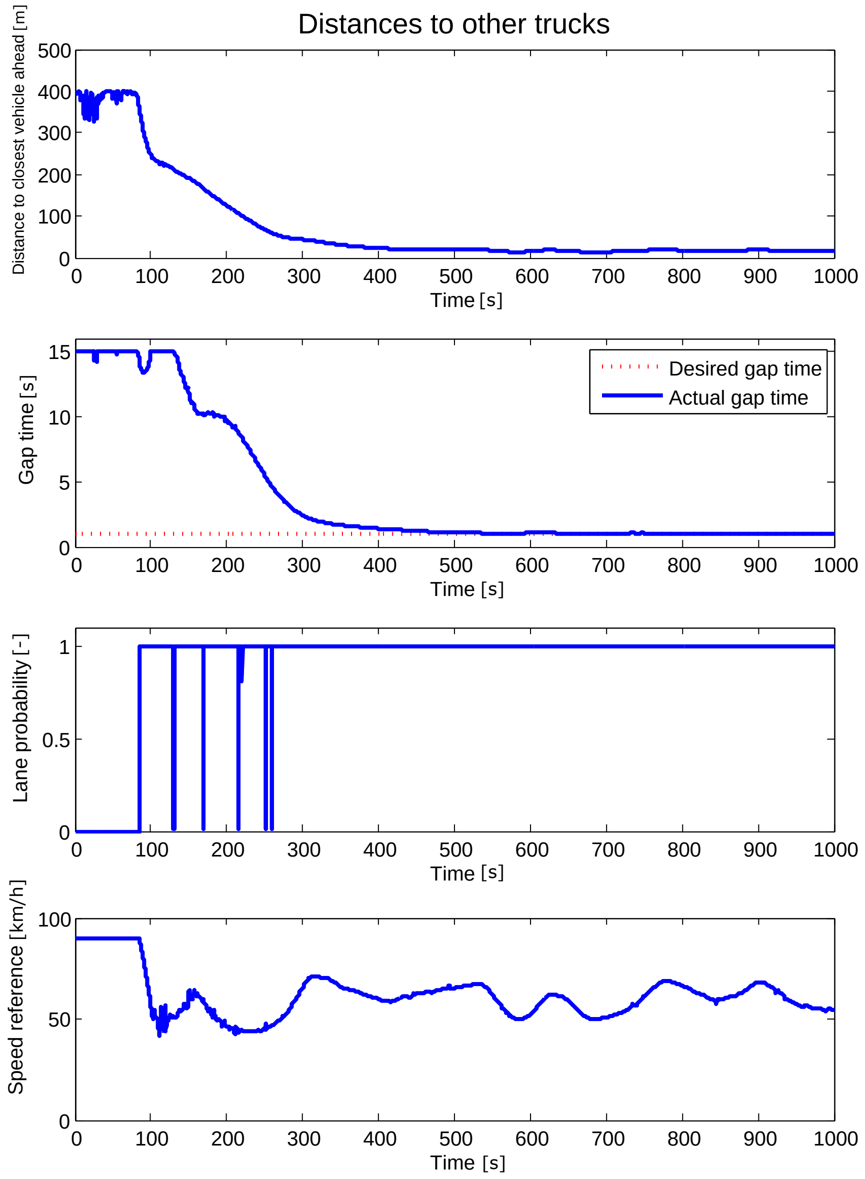

Figure 5.5: *Distance and time gap to closest vehicle ahead, value of lane probability and the resulted speed reference from the ACC*

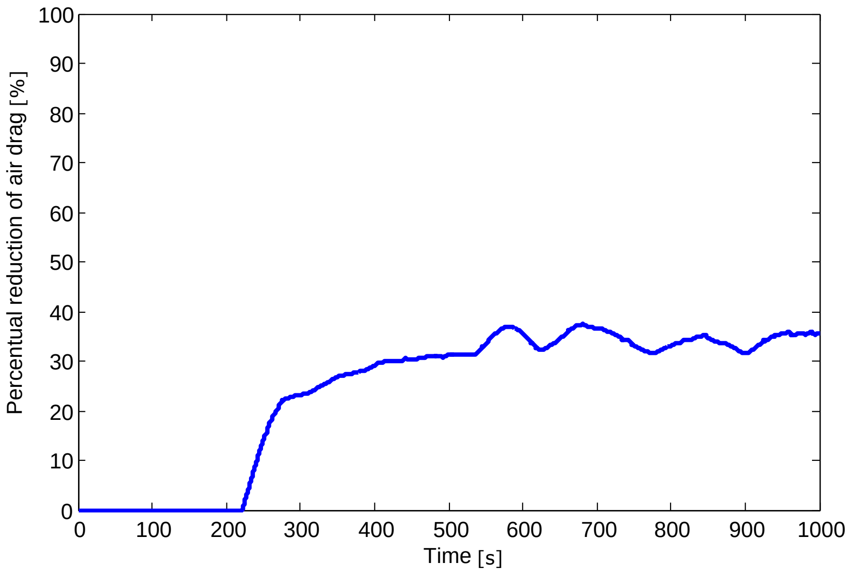

__Figure 5.6:__ Resulted air drag reduction for the same simulation as Figure 5.5

6 Conclusions and future work

In this chapter the drawn conclusion from the thesis is presented together with suggestions on work that can be done in the future.

6.1 Conclusions

By combining physical models for single-track vehicle dynamics, powertrain models developed by Scania, and different control strategies, a driver-truck model are developed in Simulink. The driver-truck model could then be duplicated and simulated together in real time and perform platooning with each other in a road system based on the real world. The developed model is used in a project in Scania to demonstrate the concept of coordinating platooning. As the driver-truck model are module based and implemented in simulink, it can easily be extended for future purposes with more complex functions. The relevance of the dynamics of the vehicle model is proved by physical calculations, and the performance was evaluated with satisfying results.

6.2 Future work

There are parts in the driver-truck model that can be improved, and possibilities for extensions that have not been researched in the thesis. One way to improve the performance of the model's behavior is to implement the same logic that are implemented in real trucks. This will also be useful to test such logic.

Additionally, the control algorithms can also be optimized. Even though the vehicles follow the path with a deviation of less then 1 meter, it is preferable to have a controller that ensures that the vehicle stays in the preferred lane.

In future tests, platooning may be implemented in both real and virtual trucks. The performance of ACC will be very important for such tests. Moreover, platooning logic may be included in real trucks.

It is also possible to implement an intelligent algorithm for automatic braking. There are many active research activities on automatic emergency braking, and some are even commercially available. Such systems can easily be added to the driver-truck model. Also braking is needed for driving according to the speed limit, as shown in Section 5.1.2

The model could also be extended for adding more functionality. One example could be to add environmental disturbances such as weather, which can be modeled to affect the friction of the road. Another interesting addition is to simulate the sending of warning messages, and to provide other vehicle's information of accidents, or dangerous road hazards. The driver-truck model could also include logic for overtaking vehicles. However, all of these improvements would need additions in the server as well.

Finally, there can be a lot of future work of the actual system that are developed in the project. The system can be improved and extended to perform tests with more virtual and real vehicles, and in the future be used for coordinating the vehicle fleet for an optimal use of platooning.

Appendix

A
Control parameters for path following

Table A.1 and Table A.2 presents the parameters and its simulated performances for the LQ-regulator and PID-controller respectivly.

Table A.1: *Performance of the LQ-regulator for different parameters*

Parameters			**Performance**		
T_p	R_{11}/R_2	R_{12}/R_{11}	**MSE**	$> 0.75m(\%)$	**RMS**
0,4	0,01	1	0,40	25	0,63
0,4	0,01	2	0,62	29	0,79
0,4	0,01	5	1,8	41	1,35
0,4	1	1	10	44	3.1
0,4	1	2	1,9	40	1,4
0,4	1	5	1,5	39	1,2
0,6	0,01	1	0,10	0,6	0,31
0,6	0,01	2	0,11	1,0	0,33
0,6	0,01	5	0,15	7,1	0,39
0,6	1	1	0,16	7,1	0,40
0,6	1	2	0,07	0,5	0,27
0,6	1	5	0,08	0,5	0,29
0,8	0,01	1	0,18	12	0,43
0,8	0,01	2	0,18	12	0,43
0,8	0,01	5	0,18	12	0,43
0,8	1	1	0,23	12	0,48
0,8	1	2	0,20	12	0,44
0,8	1	5	0,19	12	0,43

Table A.2: *Performance of the PID-controller for different parameters*

Parameters					Performance		
K_{P0}	K_{D0}	K_{Pp}	K_{Dp}	$K_{P\psi}$	**MSE**	> 0.75**m**(%)	**RMS**
0,1	1	0	10	0,1	1,6	31	1,27
0,1	10	0	10	0,1	2,8	58	1,6
1	1	0	10	0,1	0,11	5,3	0,32
1	10	0	10	0,1	0,21	7,6	0,46
0,1	1	0	100	0,1	20	98	4,5
0,1	10	0	100	0,1	17	98	4,1
1	1	0	100	0,1	1,5	30	1,2
1	10	0	100	0,1	1,5	31	1,23
0,1	1	0,1	10	0,1	0,63	23	0,80
0,1	10	0,1	10	0,1	1,14	26	1,1
1	1	0,1	10	0,1	0,09	4,3	0,31
1	10	0,1	10	0,1	0,17	5,6	0,41
0,1	1	0,1	100	0,1	11	98	3,3
0,1	10	0,1	100	0,1	9,5	98	3,1
1	1	0,1	100	0,1	1,2	30	1,1
1	10	0,1	100	0,1	1,4	31	1,2
0,1	1	0	10	1	1,6	31	1,3
0,1	10	0	10	1	3,8	60	1,9
1	1	0	10	1	0,10	5,3	0,32
1	10	0	10	1	0,19	6,7	0,44
0,1	1	0	100	1	19	98	4,3
0,1	10	0	100	1	17	98	4,1
1	1	0	100	1	1,4	30	1,2
1	10	0	100	1	1,4	30	1,2
0,1	1	0,1	10	1	0,59	22	0,77
0,1	10	0,1	10	1	1,4	31	1,2
1	1	0,1	10	1	0,09	4,4	0,30
1	10	0,1	10	1	0,21	7,6	0,46
0,1	1	0,1	100	1	9,4	98	3,1
0,1	10	0,1	100	1	9,4	98	3,1
1	1	0,1	100	1	1,2	30	1,1
1	10	0,1	100	1	1,4	30	1,1

Bibliography

Assad Alam. *Fuel-Efficient Heavy-Duty Vehicle Platooning*. PhD thesis, KTH, Automatic Control, 2014. QC 20140527. Cited on page 4.

E. Catmull and R. Rom. A class of local interpolating spline. *Computer Aided Geometric Design*, pages 317–326, 1974. Cited on page 10.

C.I. Chatzikomis and K.N. Spentzas. Apath-following driver model with longitudinal and lateral control of vehicle's motion. *Forschung im Ingenieurwesen*, 73 (4):257–266, 2009. Cited on page 4.

Lars Eriksson and Lars Nielsen. *Modeling and Control of Engines and Drivelines*. John Wiley & Sons, 2014. Cited on page 3.

Mordechai (Muki) Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, October 2008. ISSN 1536-1268. doi: 10.1109/MPRV.2008.80. URL `http://dx.doi.org/10.1109/MPRV.2008.80`. Cited on page 7.

He Han-wu, Lu Yong-ming, and Lou Yan. Virtual reality based intelligent vehicle modeling in driving simulation system. In *Computer-Aided Industrial Design and Conceptual Design, 2006. CAIDCD'06. 7th International Conference on*, pages 1–5. IEEE, 2006. Cited on page 3.

Kuo-Yun Liang. Coordination and routing for fuel-efficient heavy-duty vehicle platoon formation. 2014. Cited on page 1.

L. Ljung and T. Glad. *Reglerteori : Flervariabla och olinjara metoder*. 2003. Cited on pages 4 and 25.

Seungwuk Moon, Kyongsu Yi, HyongJin Kang, and Paljoo Yoon. Adaptive cruise control with collision avoidance in multi-vehicle traffic situations. *SAE International Journal of Passenger Cars-Mechanical Systems*, 2(1):653–660, 2009. Cited on page 34.

J.E. Naranjo, C. Gonzalez, R. Garcia, T. de Pedro, and Haber R.E. Power-steering control architecture for automatic driving. 2005. Cited on page 25.

H. Pacejka. *Tyre and Vehicle Dynamics*. Butterworth-Heinemann, 2006. Cited on page 3.

Christian Ress, Dirk Balzer, Alexander Bracht, Sinisa Durekovic, and Jan Löwenau. Adasis protocol for advanced in-vehicle applications. In *15th World Congress on Intelligent Transport Systems*, 2008. Cited on page 7.

RS Sharp and V Valtetsiotis. Optimal preview car steering control. *Vehicle System Dynamics*, 35, 2001. Cited on pages 4 and 26.

Randall W Smith. *Department of Defense World Geodetic System 1984: its definition and relationships with local geodetic systems*. Defense Mapping Agency, 1987. Cited on pages 8 and 9.

M. Thommyppillai, S. Evangelou, and R. S. Sharp. Car driving at the limit by adaptive linear optimal preview control. *Vehicle System Dynamics*, 47(12): 1535–1550, 2009. Cited on pages 3, 14, and 17.

Hucho Wolf-Heinrich. *Aerodynamics of Road Vehicles: From Fluid Mechanics to Vehicle Engineering*. SAE, 1998. Cited on pages 16 and 17.

Linköpings universitet

Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida `http://www.ep.liu.se/`

Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for his/her own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: `http://www.ep.liu.se/`

© Oskar Daniels