Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2017

Automatic Parking and Path Following Control for a Heavy-Duty Vehicle

Joakim Mörhed and Filip Östman

LINKÖPING
UNIVERSITY

Master of Science Thesis in Electrical Engineering

Automatic Parking and Path Following Control for a Heavy-Duty Vehicle

Joakim Mörhed and Filip Östman

LiTH-ISY-EX--17/5078--SE

Supervisor: **Oskar Ljungqvist**
ISY, Linköpings universitet
Soheil Salehpour
REPF, Scania CV AB

Examiner: **Daniel Axehill**
ISY, Linköpings universitet

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

The interest in autonomous vehicles has never been higher and there are several components that need to function for a vehicle to be fully autonomous; one of which is the ability to perform a parking at the end of a mission. The objective of this thesis work is to develop and implement an automatic parking system (APS) for a heavy-duty vehicle (HDV). A delimitation in this thesis work is that the parking lot has a known structure and the HDV is a truck without any trailer and access to more computational power and sensors than today's commercial trucks.

An automatic system for searching the parking lot has been developed which updates an occupancy grid map (OGM) based on measurements from GPS and LIDAR sensors mounted on the truck. Based on the OGM and the known structure of the parking lot, the state of the parking spots is determined and a path can be computed between the current and desired position.

Based on a kinematic model of the HDV, a gain-scheduled linear quadratic (LQ) controller with feedforward action is developed. The controller's objective is to stabilize the lateral error dynamics of the system around a precomputed path. The LQ controller explicitly takes into account that there exist an input delay in the system. Due to minor complications with the precomputed path the LQ controller causes the steering wheel turn too rapidly which makes the backup driver nervous. To limit these rapid changes of the steering wheel a controller based on model predictive control (MPC) is developed with the goal of making the steering wheel behave more human-like. A constraint for maximum allowed changes of the controller output is added to the MPC formulation as well as physical restrictions and the resulting MPC controller is smoother and more human-like, but due to computational limitations the controller turns out less effective than desired.

Development and testing of the two controllers are evaluated in three different environments of varying complexity; the simplest simulation environment contains a basic vehicle model and serves as a proof of concept environment, the second simulation environment uses a more realistic vehicle model and finally the controllers are evaluated on a full-scale HDV.

Finally, system tests of the APS are performed and the HDV successfully parks with the LQ controller as well as the MPC controller. The concept of a self-parking HDV has been demonstrated even though more tuning and development needs to be done before the proposed APS can be used in a commercial HDV.

Acknowledgments

First of all, we would like to express our sincere gratitude to our supervisors Soheil Salehour and Oskar Ljungqvist. Soheil, you have always been our partner in crime and our many laughs together has been delightful. We would like to wish you a good luck with your newfound parenthood. Oskar, you have probably had your patience tested with us and for that we want to say thank you. We wish you all the luck with the continuation of your Ph.D.

Secondly, we would like to thank all our fellow thesis workers at Scania CV AB, especially Klas Lindsten, Gustav Ling, Magdalena Cederlöf and Nadine Möllberg, for all the enlightening discussions we have had during this spring.

Thirdly, we would direct the attention to our examiner Daniel Axehill for providing us with the foundation of knowledge which this project relies on during our time at Linköping University.

We also want to mention Scania Hälsocenter Gröndal. Here we have found our inner peace during tough times.

Finally, we would like to thank all our colleagues at the department of Autonomous Transport Solution - Fusion & Control group at Scania CV AB in Södertälje for all the help and resources they have spent on us during the project.

Södertälje, June 2017
Joakim Mörhed and Filip Östman

Contents

Notation

Notations

Notation	Description
$\mathbb{N}$	Set of natural numbers
$\mathbb{S}_{+}^{n}$	Positive semidefinite matrix
$\mathbb{S}_{++}^{n}$	Positive definite matrix

Abbreviations

Abbreviation	Description
ADAS	Advanced driver assistance systems
APS	Autonomous parking system
DARE	Discrete time algebraic Riccati equation
FOV	Field of view
GPS	Global positioning system
HDV	Heavy-duty vehicle
LIDAR	Light detection and ranging
LPV	Linear parameter-varying
LQR	Linear quadratic regulator
MPC	Model predictive control
OGM	Occupancy grid map

1 Introduction

This master's thesis project was carried out at the Autonomous Transport Solution - Fusion & Control group at Scania CV AB and resulted in an Automatic Parking System (APS) for Heavy-Duty Vehicles (HDVs), which is a new area of application for this kind of system. The APS is composed of three components performing the following subtasks:

- Generating an Occupancy Grid Map (OGM) based on measurements from LIDAR and GPS sensors.
- Generating a collision free path from an initial position to the parking spot.
- Controlling the truck to follow the generated path.

This chapter will start with a motivation of the work in the thesis with an accompanying problem formulation, specify the thesis assumptions, present related work and give an overview of the thesis outline.

1.1 Motivation and Purpose

Autonomous vehicles have been a very intense area of research during the last ten years but the vision of a driverless vehicle is far from new. As early as 1935, a driverless car was depicted on the screen, *The Safest Place* (1935), and literature with the same theme were written, e.g. *Utopolis* (1930).

During the upcoming decades, the concept of autonomous vehicles became closer to reality and several Advanced Driver Assistance Systems (ADAS) emerged for commercial use, Adaptive Cruise Control and Collision Avoidance System being well known. The two main purposes of ADAS are to increase the safety while also making the vehicle more efficient. In [21] the potential effects of ADAS on accident reduction on rural and urban highways are presented as significant. One crucial factor related to efficiency is fuel consumption which several ADAS have been developed for. An example of such a system is the Look-Ahead Control which uses information of the upcoming road topology to optimize the vehicle's speed profile in order to reduce its fuel consumption [16].

One of the latest ADAS developed for cars is automatic parking; however, this service has not yet been released officially in the field of trucks. One very plausible reason behind this is that a truck usually only stops to park at the end of a shift or when filling up fuel, rendering the use of such a service quite unnecessary because it is very seldom needed compared to the length of the driving time. This reason makes the truck developers rather spend their time and money on other, more useful, systems. However, to succeed in the development of a fully autonomous truck it is a necessity that the truck can park without a driver. Thus, the aim of this thesis is to develop a system that makes it possible to perform automatic parking for an HDV.

1.2 Problem Formulation

Using LIDAR and GPS measurements with the knowledge of the structural properties of a parking lot; is it possible to perform an autonomous parking in that environment? Is it possible to design a path following controller with desired accuracy in forward and backward motion such that automatic parking can be achieved?

1.3 Assumptions

To limit the complexity of this project to a level that is achievable within a master's thesis several assumptions have been made which are presented here:

- The APS is developed for a truck without any trailer. The truck for this system has access to more computational power and is equipped with additional and more advanced sensors compared to a commercial truck.

- The parking lot which the APS is meant to operate in has a known structure with no dynamic obstacles. The ground level of the parking lot is relatively flat.

- Cases when the provided motion planner is not able to plan a route to any of the available parking spots are excluded and not a part of this thesis.

1.4 Related Work

Automatic parking systems have been developed for over a decade by the car industry to help drivers with parking maneuvers such as parallel parking and perpendicular parking. No results have been found on development of a complete APS for a truck. There are several different types of truck and trailer compositions. A single truck is essentially a large car, making the research for parking assistance for cars highly relevant for this thesis. One of the first automatic parallel parkings was performed in 1996 [11] with an experimental car. Today, most of the large car manufacturers deliver car models providing some sort of APS as an optional service.

Common for all autonomous vehicles is the need to collect information from sensors. This is done to get an estimate of the environment and the vehicle's state in that environment for planning future actions. A large amount of applications are dependent on mapping the environment, hence there exists plenty of research in the area and there are several different mapping algorithms compared in [29]. This thesis employs a probabilistic method to update an occupancy grid map to represent the environment as in [29], [10], [28]. In [29] different mapping algorithms are compared; given that the position of the vehicle is known, the occupancy grid map is a good choice.

Motion planning is another part where there exists a vast amount of research and some planning algorithms are presented in [18]. The motion planner, more specifically the Lattice planner [18], [8], [24], is provided by Scania CV AB and is used to compute a feasible path, i.e. a collision free path which is suited for the kinematics of a truck, between waypoints in a given occupancy grid map.

A very important part in the APS is to keep the vehicle close to the desired reference path. This has been a popular topic in the control community for decades and many studies have been done on this topic which have led to several strategies for path following. One of the earlier strategies, which still is popular due to its simplicity is to let the vehicle travel along an arc of a circle to a point on the path which lies at a certain distance in front of the vehicle. This method is referred to as *Pure Pursuit Path Tracking* [17]. Later on, strategies concerning the use of *Linear Quadratic Control* [27] [1] and *Model Predictive Control*[25][3] [12] for vehicle control were studied with successful outcomes. They solve different optimization problems related to the vehicle and the reference path and use the solution as control signal. At the beginning, these studies were mainly directed towards car-like vehicles. Today, studies are carried out on more complicated vehicle compositions, such as an LQ-controller for a truck with a dolly-steered trailer [20] and an MPC-controller for a tractor-trailer system [30]. While more advanced scenarios are being explored the path following capabilities for a car has begun its way into the commercial market for cars. One major car brand, Volvo, states that they will have unsupervised driving cars on the road as early as 2021 [2].

1.5 Outline

This thesis is organized in the following chapters and content:

Chapter 2 - Relevant theory for the reader.

Chapter 3 - An overview of the different parts of the system.

Chapter 4 - The parking spot detection system is explained.

Chapter 5 - A model of the system is derived.

Chapter 6 - The developed controllers are explained.

Chapter 7 - Presents and discusses the result of the different parts of the autonomous parking system.

Chapter 8 - Conclusions from this thesis accompanied with suggested future work.

2 Preliminaries

This chapter will present theory which is used throughout this thesis.

2.1 Path Following and Linearization

A common problem in applications of autonomous vehicles is how to undertake a movement order, i.e. to move along a given geometric curve in the environment of the vehicle. Depending on the priority of the movement this task is categorized differently. In this application, the focus is on accuracy rather than punctuality in time. Hence, there is no pre-specified time constraints for the movement. These kind of problems are referred to as *path-following problems*, while the addition of punctuality in time is referred to as *trajectory-tracking problems*. Consider the nonliner system, which supposedly relates to the path-following problem,

$$\dot{x} = f(x, u) \tag{2.1}$$

where x denotes the states and u denotes the control signals. It is of interest to linearize this relation to enable the framework of linear control theory. The method of linearization is described in, e.g., [14] and will be used here as an inspiration.

Let $x_0(t)$ denote the nominal state trajectory and $u_0(t)$ the nominal input trajectory that satisfies

$$\dot{x}_0 = f(x_0, u_0). \tag{2.2}$$

It is natural to create a representation of the deviation between the current and nominal values on the state and input

$$\begin{aligned} \tilde{x} &= x - x_0, \\ \tilde{u} &= u - u_0. \end{aligned} \tag{2.3}$$

This representation can then, if f is at least one time continuously differentiable in a region around (x_0, u_0), be used to compute a linear parameter-varying system with $\tilde{x}$ and $\tilde{u}$ as the state error and input deviation, respectively. The derivative of $\tilde{x}$ in (2.3) yields

$$\dot{\tilde{x}} = \dot{x} - \dot{x}_0 = f(x, u) - f(x_o, u_o) \triangleq \tilde{f}(x_0, u_0, \tilde{x}, \tilde{u}) \tag{2.4}$$

where $f(x_0, u_0, 0, 0) = 0$, i.e., the origin is an equilibrium point. A first order Taylor series expansion of (2.4) around the origin yields

$$\dot{\tilde{x}} = \underbrace{\left.\frac{\partial \tilde{f}}{\partial \tilde{x}}\right|_{\substack{\tilde{x}=0\\ \tilde{u}=0}}}_{\triangleq A} \tilde{x} + \underbrace{\left.\frac{\partial \tilde{f}}{\partial \tilde{u}}\right|_{\substack{\tilde{x}=0\\ \tilde{u}=0}}}_{\triangleq B} \tilde{u} = A(x_0, u_0)\tilde{x} + B(x_0, u_0)\tilde{u}. \tag{2.5}$$

These calculations lead to a linear parameter-varying (LPV) system which relates the vehicle to the path. Linear control theory can now be used to keep $\tilde{x}$ as close to the origin as possible.

2.2 Discretization of a Continuous System With Input Delay

Consider a continuous time linear system with an input delay,

$$\dot{x}(t) = Ax(t) + Bu(t - \tau), \tag{2.6}$$

where τ denotes the input delay. In most, if not all, applications the controller is discrete which means that the control signal is formed as a piecewise constant control signal

$$u(t) = u(kT_s), \quad kT_s \leq t < (k+1)T_s, \tag{2.7}$$

where T_s is the sampling time. It is commonly occurring that the input delay is significantly larger compared to the sampling time which means that the delay can be assumed to be a multiple of the sampling time

$$\tau = \alpha T_s, \quad \alpha \in \mathbb{N}. \tag{2.8}$$

Using [15] as inspiration, (2.6) can be discretized under the assumption that (2.8) holds as follows. The solution to (2.6) is

$$x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-s)}Bu(s-\tau)\, ds. \tag{2.9}$$

The sampling time is T_s, hence let $t = kT_s + T_s$ and $t_0 = kT_s$ which gives

$$x(kT_s + T_s) = \underbrace{e^{AT_s}}_{\triangleq F} x(kT_s) + \int_{kT_s}^{kT_s+T_s} e^{A(kT_s+T_s-s)}Bu(s-\tau)\, ds. \tag{2.10}$$

Replacing the integration variable s with $\eta = (k+1)T_s - s$ gives

$$x(kT_s + T_s) = Fx(kT_s) + \int_{0}^{T_s} e^{A\eta}Bu(kT_s + T_s - \tau - \eta)\, d\eta. \tag{2.11}$$

Inserting (2.8) into (2.11) gives

$$x(kT_s + T_s) = Fx(kT_s) + \int_{0}^{T_s} e^{A\eta}Bu([k - \alpha + 1]T_s - \eta)\, d\eta. \tag{2.12}$$

Since the input signal is piecewise constant, according to (2.7), the input signal in (2.12) is constant during the integration interval

$$u([k-\alpha+1]T_s-\eta)=u([k-\alpha]T_s)\quad \text{for } 0<\eta<T_s. \tag{2.13}$$

Using (2.13) to move the input signal out from the integral in (2.12) gives

$$x(kT_s+T_s)=Fx(kT_s)+\underbrace{\int_0^{T_s} e^{A\eta}\,d\eta B}_{\triangleq G}\,u([k-\alpha]T_s). \tag{2.14}$$

For simplicity, the following notation will be used throughout the report

$$x_{k+1}=Fx_k+Gu_{k-\alpha}. \tag{2.15}$$

Note that

$$e^{At}=\mathcal{L}^{-1}(sI-A)^{-1}, \tag{2.16}$$

where $\mathcal{L}^{-1}$ is the inverse Laplace transform.

2.3 Linear Quadratic Control for Systems with Input Delay

Consider a system with an input delay of $\alpha\in\mathbb{N}$ samples, resulting in the following LQ-problem:

$$\begin{aligned}\min_{u(\cdot)}\quad & \sum_{k=0}^{\infty} x_k^T Q_1 x_k+u_k^T Q_2 u_k,\\ \text{subject to}\quad & x_{k+1}=Fx_k+Gu_{k-\alpha},\\ & x_0 \text{ given.}\end{aligned} \tag{2.17}$$

The matrices $Q_1\in\mathbb{S}_+^n$ and $Q_2\in\mathbb{S}_{++}^m$, where n and m are the number of states and control signals, respectively. They are called cost matrices and define how the controller should prioritize in the minimization of the deviation in the states and the control signal. They are usually diagonal and are the design parameters of the controller. If the control signal is a scalar it is possible to factorize Q_2 and instead design the matrix $\frac{Q_1}{Q_2}$. The input delay induces the effect that a change in the control signal takes α samples before affecting the states and we can rewrite the problem as

$$\begin{aligned}
\min_{u(\cdot)} \quad & \sum_{k=0}^{\infty} x_{k+\alpha}^T Q_1 x_{k+\alpha} + u_k^T Q_2 u_k, \\
\text{subject to} \quad & x_{k+1} = F x_k + G u_{k-\alpha}, \\
& \alpha \in \mathbb{N}, \\
& x_0 \text{ given.}
\end{aligned} \tag{2.18}$$

Let $\hat{x}_k = x_{k+\alpha}$ be the predicted state forward in time, simulated by the motion model of the system and the optimization problem can be reformulated as

$$\begin{aligned}
\min_{u(\cdot)} \quad & \sum_{k=0}^{\infty} \hat{x}_k^T Q_1 \hat{x}_k + u_k^T Q_2 u_k, \\
\text{subject to} \quad & \hat{x}_{k+1} = F \hat{x}_k + G u_k, \\
& \hat{x}_0 = x_\alpha \text{ given.}
\end{aligned} \tag{2.19}$$

The LQ-problem in (2.19) has the following optimal control law [13]:

$$u_k = -L\hat{x}_k, \tag{2.20}$$

where $L = (G^T S G + Q_2)^{-1} G^T S F$ and S is the unique positive definite solution to the discrete time algebraic Riccati equation (DARE):

$$S = F^T S F + Q_1 - F^T S G (G^T S G + Q_2)^{-1} G^T S F. \tag{2.21}$$

2.4 Model Predictive Control with Input Delay

The Model Predictive Control (MPC) seen in (2.22) is closely related to the LQ-controller described in Section 2.3. The main issue of the LQ-framework is that there is no option to set constraints on any of the related variables. The MPC-controller is an extension of the LQ-controller which allows constraints on control signals and states. This is very useful because it can give the controller insight on several attributes of the controlled system, e.g. physical restrictions on the input signal. Consider the following MPC problem [19]:

$$
\begin{aligned}
\min_{u(\,\cdot\,)} \quad & \sum_{k=0}^{N_p} x_k^T Q_1 x_k + u_k^T Q_2 u_k, \\
\text{subject to} \quad & x_{k+1} = F x_k + G u_{k-\alpha}, \\
& u(\,\cdot\,) \in \mathbb{U}, \\
& x(\,\cdot\,) \in \mathbb{X}, \\
& x_0 \text{ given.}
\end{aligned}
\tag{2.22}
$$

where the system has an input delay of $\alpha \in \mathbb{N}$ samples. The weight matrices $Q_1 \in \mathbb{S}_+^n$ and $Q_2 \in \mathbb{S}_{++}^n$ are the design parameters of the controller while $\mathbb{U}$ and $\mathbb{X}$ defines the set of feasible solutions. As in Section 2.3 this problem can be rewritten in the standard form by introducing the predicted state $\hat{x}_k = x_{k+\alpha}$:

$$
\begin{aligned}
\min_{u(\,\cdot\,)} \quad & \sum_{k=0}^{N_p} \hat{x}_k^T Q_1 \hat{x}_k + u_k^T Q_2 u_k, \\
\text{subject to} \quad & \hat{x}_{k+1} = F \hat{x}_k + G u_k, \\
& u(\,\cdot\,) \in \mathbb{U}, \\
& x(\,\cdot\,) \in \mathbb{X}, \\
& \hat{x}_0 \text{ given.}
\end{aligned}
\tag{2.23}
$$

In contrary to the solution to the LQ problem in (2.19), in general there exists no constant state feedback solution to (2.23) which complicates the implementation of an MPC-controller. The most straight forward way to handle this is to solve this optimization problem at every sample time, referred to as an implicit MPC-controller.

2.4.1 Implementation of Implicit MPC

There exist optimization software to solve quadratic programming problems with linear constraints (2.24).

$$
\min_U \frac{1}{2} U^T H U + f U, \tag{2.24a}
$$

$$
A_u U \leq b_u. \tag{2.24b}
$$

where $H \in S_{++}^n$. The solver used in this thesis is based on an interior point method described in [4]. Hence (2.23) needs to be rewritten on a form that corresponds to (2.24).

Cost function

In [23] the method to create H and f is described, for clarity the method is also described here. Given a prediction horizon, N_p, the model in (2.23) can be used to let any state at time $k+i$ where $i \leq N_p$ be a function of the state at time k and the preceding inputs. That is

$$\begin{aligned}
\hat{x}_{k+1} &= F\hat{x}_k + Gu_{k+1} \\
\hat{x}_{k+2} &= F\hat{x}_{k+1} + Gu_{k+1} \\
&= F(F\hat{x}_k + Gu_k) + Gu_{k+1} \\
&= F^2\hat{x}_k + FGu_k + Gu_{k+1} \\
\hat{x}_{k+3} &= F^3\hat{x}_k + \ldots \\
\vdots \quad & \quad \vdots \qquad \ddots \\
\hat{x}_{k+N_p} &= F^{N_p}\hat{x}_k + F^{N_p-1}Gu_k + \ldots + Gu_{k+N_p-1}.
\end{aligned} \tag{2.25}$$

By introducing

$$\mathcal{X} = \begin{bmatrix} \hat{x}_{k+1} \\ \hat{x}_{k+2} \\ \vdots \\ \hat{x}_{k+N_p} \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N_p-1} \end{bmatrix}, \tag{2.26}$$

(2.25) can be written in the more compact form

$$\mathcal{X} = \mathscr{F}\hat{x}_k + \mathscr{G}U, \tag{2.27}$$

where

$$\mathscr{F} = \begin{bmatrix} F \\ F^2 \\ \vdots \\ F^{N_p} \end{bmatrix} \text{and } \mathscr{G} = \begin{bmatrix} G & & & & \\ FG & G & & & \\ \vdots & & & \ddots & \\ F^{N_p-1}G & F^{N_p-2}G & \cdots & FG & G \end{bmatrix}. \tag{2.28}$$

By introducing the extended cost matrices

$$\bar{Q}_1 = \begin{bmatrix} Q_1 & & \\ & \ddots & \\ & & Q_1 \end{bmatrix} \text{ and } \bar{Q}_2 = \begin{bmatrix} Q_2 & & \\ & \ddots & \\ & & Q_2 \end{bmatrix}, \tag{2.29}$$

the objective function in (2.23) can be rewritten in matrix form using (6.16) and (2.29),

$$\min_U \mathcal{X}^T \bar{Q}_1 \mathcal{X} + U^T \bar{Q}_2 U. \tag{2.30}$$

Using (2.27) in (2.30) gives

$$\min_U (\mathscr{F}\hat{x}_k + \mathscr{G}U)^T \bar{Q}_1 (\mathscr{F}\hat{x}_k + \mathscr{G}U) + U^T \bar{Q}_2 U. \tag{2.31}$$

This can be rewritten by applying the transpose inside the parenthesis which gives

$$\min_U (\hat{x}_k^T \mathscr{F}^T + U^T \mathscr{G}^T) \bar{Q}_1 (\mathscr{F}\hat{x}_k + \mathscr{G}U) + U^T \bar{Q}_2 U. \tag{2.32}$$

By performing the matrix multiplications and rearranging the terms and neglecting constant terms the following equivalent problem is obtained

$$\min_U 2\hat{x}_k^T \mathscr{F}^T \bar{Q}_1 \mathscr{G} U + U^T (\mathscr{G}^T \bar{Q}_1 \mathscr{G} + \bar{Q}_2) U. \tag{2.33}$$

Now let

$$\begin{aligned} H &= \mathscr{G}^T \bar{Q}_1 \mathscr{G} + \bar{Q}_2, \\ f &= \hat{x}_k^T \mathscr{F}^T \bar{Q}_1 \mathscr{G}, \end{aligned} \tag{2.34}$$

which means that the sum in (2.33) can be presented as

$$\min_U \frac{1}{2} U^T H U + f U. \tag{2.35}$$

There also exist a sparse formulation of the MPC problem, see e.g. [5], and quasi-sparse formulations as described in [6] which are relevant to reduce the computational complexity of MPC.

Terminal State Weight

The section above solves the constrained finite horizon problem described in (2.23) and Section 2.3 solves the unconstrained infinite horizon problem. With the linear state feedback $u_k = -L\hat{x}_k$ from (2.20) and S being the solution to the discrete time algebraic Riccati equation (DARE) (2.21), we get the following optimal objective function value for the unconstrained infinite horizon problem [19]

$$\min_{\substack{u \in \mathbb{R}^m \\ \text{s.t } \hat{x}_{j+1} = F\hat{x}_k + Gu_j}} \sum_{j=k+N_p}^{\infty} \hat{x}_j^T Q_1 \hat{x}_j + u_j^T Q_2 u_j = \hat{x}_{k+N_p}^T S \hat{x}_{k+N_p}. \tag{2.36}$$

If the constraints are inactive after the time horizon, N_p, it is possible to solve the constrained infinite horizon problem

$$\min_{\substack{u(\cdot) \\ \text{s.t } u(\cdot) \in \mathbb{U} \\ \hat{x}(\cdot) \in \mathbb{X} \\ \hat{x}_{j+1} = F\hat{x}_k + Gu_j}} \sum_{j=k}^{\infty} \hat{x}_j^T Q_1 \hat{x}_j + u_j^T Q_2 u_j, \tag{2.37}$$

by splitting the sum at the time horizon, N_p,

$$\begin{aligned} &\min_{\substack{u(\cdot) \\ \text{s.t } u(\cdot) \in \mathbb{U} \\ \hat{x}(\cdot) \in \mathbb{X} \\ \hat{x}_{j+1} = F\hat{x}_k + Gu_j}} \sum_{j=k}^{k+N_p-1} \hat{x}_j^T Q_1 \hat{x}_j + u_j^T Q_2 u_j \\ + &\min_{\substack{u \in \mathbb{R}^m \\ \text{s.t } \hat{x}_{j+1} = F\hat{x}_k + Gu_j}} \sum_{j=k+N_p}^{\infty} \hat{x}_j^T Q_1 \hat{x}_j + u_j^T Q_2 u_j, \end{aligned} \tag{2.38}$$

and inserting (2.36)

$$\min_{\substack{u(\cdot) \\ \text{s.t } u(\cdot) \in \mathbb{U} \\ \hat{x}(\cdot) \in \mathbb{X} \\ \hat{x}_{j+1} = F\hat{x}_k + Gu_j}} \sum_{j=k}^{k+N_p-1} \hat{x}_j^T Q_1 \hat{x}_j + u_j^T Q_2 u_j + \hat{x}_{k+N_p}^T S \hat{x}_{k+N_p}. \tag{2.39}$$

The terminal state weight is therefore included in the cost matrix $\bar{Q}_1$

$$\bar{Q}_1 = \begin{bmatrix} Q_1 & & & \\ & \ddots & & \\ & & Q_1 & \\ & & & S \end{bmatrix}. \tag{2.40}$$

The introduction of the terminal state weight is important as it prohibits the controller to place the system in a state which is favorable in the short run but unfavorable in the long run. Even though it is usually not possible to guarantee that no constraints are active after $j = k + N_p$, choosing S as end penalty usually gives good control in practice.

Constraints

To be able to formulate the general constraints in (2.23) as linear constraints in (2.24) none of the constraints are allowed to be nonlinear. Given a set of constraints in the form

$$\begin{aligned} c_{\mathcal{X}_{\text{low}}} &\leq C\mathcal{X} \leq c_{\mathcal{X}_{\text{high}}}, \\ d_{U_{\text{low}}} &\leq DU \leq d_{U_{\text{high}}}, \end{aligned} \tag{2.41}$$

where $\mathcal{X}$ and U are defined in (6.16). Inserting (2.27) in the uppermost equation in (2.41) yields

$$c_{\mathcal{X}_{\text{low}}} \leq C(\mathscr{F}x[k] + \mathscr{G}U) \leq c_{\mathcal{X}_{\text{high}}}, \tag{2.42}$$

which can be written in the form

$$\begin{cases} C\mathscr{G}U & \leq c_{\mathcal{X}_{\text{high}}} - C\mathscr{F}x[k] \\ -C\mathscr{G}U & \leq -\left(c_{\mathcal{X}_{\text{low}}} - C\mathscr{F}x[k]\right) \end{cases} \tag{2.43}$$

The constraints on the control signal gives

$$d_{U_{\text{low}}} \leq DU \leq d_{U_{\text{high}}} \Leftrightarrow \begin{cases} DU & \leq d_{U_{\text{high}}} \\ -DU & \leq -d_{U_{\text{low}}} \end{cases} \tag{2.44}$$

Equation (2.41) can now be written in the form in (2.24)

$$\underbrace{\begin{bmatrix} C\mathscr{G} \\ -C\mathscr{G} \\ D \\ -D \end{bmatrix}}_{A_u} U \leq \underbrace{\begin{bmatrix} c_{\mathcal{X}_{\text{high}}} - C\mathscr{F}x[k] \\ -\left(c_{\mathcal{X}_{\text{low}}} - C\mathscr{F}x[k]\right) \\ d_{U_{\text{high}}} \\ -d_{U_{\text{low}}} \end{bmatrix}}_{b_u} \tag{2.45}$$

3

System Overview

This section will give a short overview of the different components in the APS. A visualization is shown in Figure 3.1. The four subsystems in the APS are:

- Parking Spot Detection
- Motion Planner
- Motion Controller
- Sensors

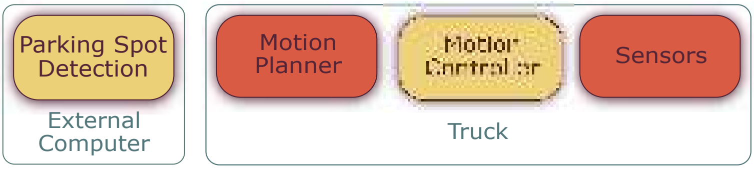


Figure 3.1: *Illustration over the four subsystems in the APS. The Parking Detection runs at an external computer connected to the truck by a Cat 6 cable. The Motion Planner, Motion Controller and signal processing of the sensor data are located on one of the computers on the truck. The systems in yellow are developed in this thesis and the orange systems are provided by Scania CV AB.*

The parking spot detection system is responsible for mapping the environment, deciding where the truck should go next and keeping track over the state of the mission. This system is running at an external computer connected to the truck with a Cat 6 cable and will be further explained in Chapter 4. The motion planner, motion controller and signal processing are located on one of the computers on the truck which has more computational power compared to what a commercial truck is equipped with today. The motion planner and the sensors are briefly described in this chapter. The motion controller can be divided into a longitudinal and a lateral controller, the former is provided by Scania CV AB while the latter is explained in Chapter 6. The systems in yellow are developed in this thesis and the systems in orange are provided by Scania CV AB. Figure 3.2 shows the truck for which the APS has been developed in this thesis and is of type SCANIA R 580.

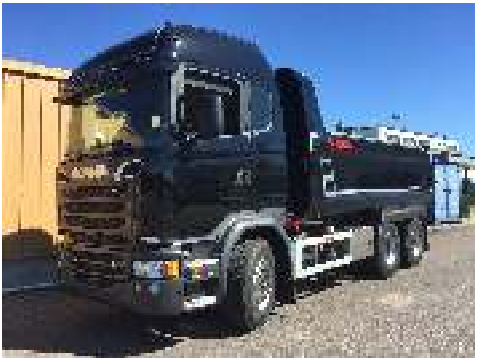


Figure 3.2: *Picture of the truck for which the APS has been developed in this thesis.*

3.1 Sensors

The sensors used to estimate the state of the truck and the surroundings consist of a high performance GPS and two *Light Detection and Ranging* (LIDAR) sensors.

3.1.1 GPS

Global positioning and heading are available from a Real Time Kinematic-GPS (RTK-GPS), which is also referred to as Carrier-Phase Enhancement. In addition to the receiver, the RTK-GPS requires an additional base station transmitter with known position. The receiver utilises the carrier phase from the additional transmitter to determine the position with centimeter precision. This system also provides the heading of the truck relative to the geographical north pole with high accuracy. The position and heading are already filtered and the position is translated to the rear axle and no further signal processing is needed for the system developed in this thesis.

3.1.2 LIDAR

The truck is equipped with two LIDAR sensors; one is mounted in the front of the truck pointing forward and one sensor in the rear of the truck pointing backward, see Figure 3.3. Both LIDAR sensors have a horizontal Field of View (FoV) of 145 degrees. The sensors sample a sequence of measurements over the FoV and deliver 25 measurement sequences each second. The measurements are given in meters and filtering is performed to remove measurements closer than 0.5 m and further away than 100 m.

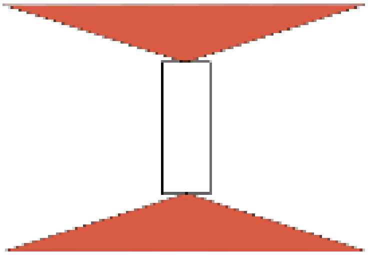


Figure 3.3: *Visualization over the LIDAR sensor placement and FoV. One sensor is mounted in the front of the truck and one is mounted in the rear of the truck. The two sensors have a horizontal FoV of 145 degrees.*

3.2 Planning a Path

The motion planner is provided by Scania CV AB and is used to generate a path between two positions. The motion planner is called from the parking spot detection system when the truck should move to a new position. Input to the motion planner is one, or more, waypoint given in GPS-coordinates and a heading relative to the geographical north pole. The current position and heading are also needed and an obstacle map with information regarding the environment around the truck. Grid cells in the obstacle map can have three different states; occupied, close to an occupied cell or empty. Planning a path through grid cells which are occupied is not allowed, planning a path through cells which are close to other occupied cells induce a higher cost than planning a path through empty grid cells not close to any occupied cells. Given these inputs, the motion planner computes a path from the vehicle's current position to the desired parking spot. A path following controller is then used to execute the planned maneuver by stabilizing the vehicle around the computed path. The motion planner, which does not take the vehicle's size into consideration, can not guarantee that there will not appear a collision and therefore enlarges all obstacles in the grid map as a safety precaution.

The motion planner is a Lattice planner [8] which uses motion primitives to find the path with lowest cost between the current and desired goal position of the truck. A motion primitive is a precomputed and feasible motion for the truck, see Figure 3.4. The path is constructed by combining multiple motion primitives until the two waypoints are connected. As an illustration, by combining the motion primitives shown in Figure 3.4, a path can be constructed as in Figure 3.5. In this example the start position is (0,0) with heading $\pi/2$ and the end position is (1,4) with heading π and the colored line is the path computed by the motion planner. The output is a feasible path with information about desired position, heading and curvature in each point along the path.

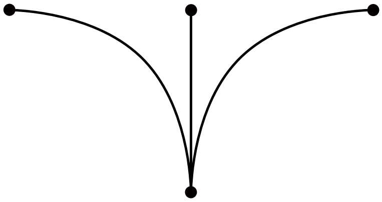


Figure 3.4: *An example of three motion primitives connecting four grid points.*

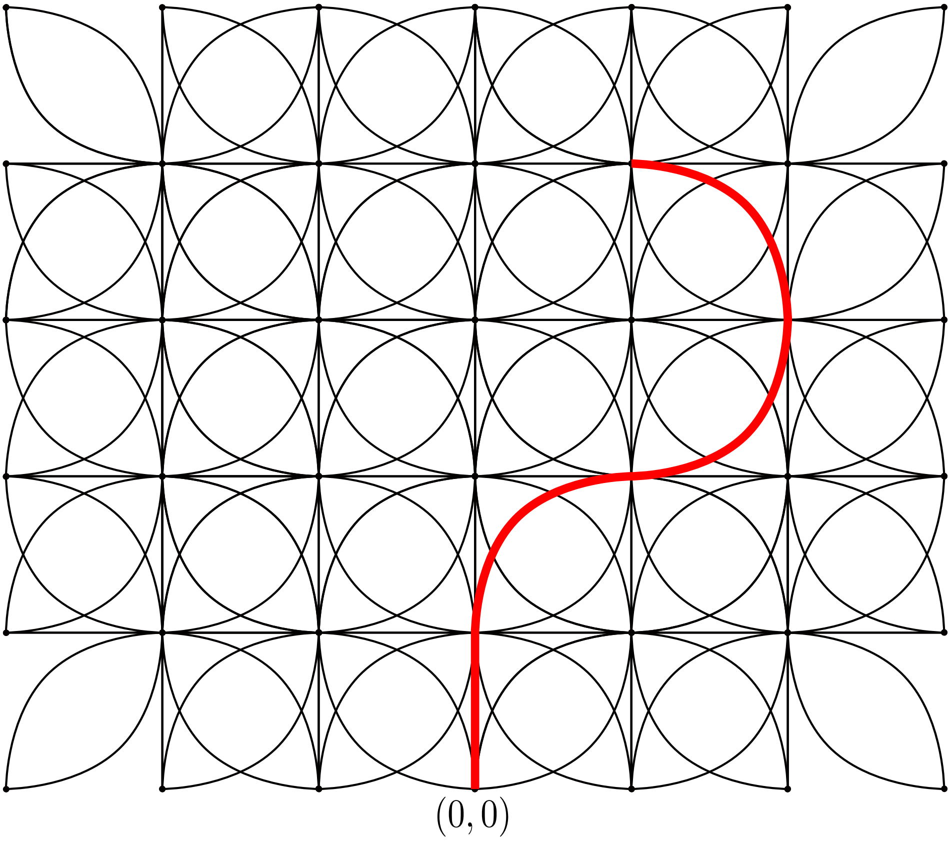


Figure 3.5: *An example with repeated motion primitives from Figure 3.4 and a path marked in red, connecting the start point (0,0) with heading $\pi/2$ and the end position (1,4) with heading π.*

4

Parking Spot Detection

This chapter will describe how to construct a map of the truck's surroundings with sensors mounted on the truck and how to use this map to decide which parking spots that are free or occupied. The parking spot detection system is developed in C++ and runs on an external laptop, connected to the vehicle by a Cat 6 cable for communication with the other parts of the APS.

4.1 Known Structure

One of the assumptions in this thesis is that the parking lot has a known structure, the structure is saved to a configuration file and is loaded in the initialization of the program and contains the following information:

- The size of the parking lot, which is needed to create a map of appropriate dimensions and resolution.
- GPS position of the center of the parking lot, used to initiate the map.
- GPS position and orientation in terms of heading of the possible parking spots.
- Waypoints with GPS position and heading describing how to search the parking lot.

4.2 Occupancy Grid Map

The surroundings of the truck is represented using an Occupancy Grid Map (OGM). The environment is divided into a grid where each grid cell contains a probability of being occupied based on measurements from the LIDAR sensors. A low grid value corresponds to a low probability that the grid cell is occupied and a high value corresponds to a high probability of the grid cell to be occupied.

In the initialization of the grid, every cell is initialized to an unknown state which is equally likely to be free as well as occupied. The probability based method described in [10] is used to update the OGM and is shown in (4.1) and (4.2). The notation used in (4.1) and (4.2) is described in Table 4.1.

Table 4.1: *Notation for the grid cell update.*

Notation	**Description**
C_i	Grid cell C, index i.
OCC	Occupied grid cell.
EMP	Empty grid cell.
$P_{\text{OCC}}^{C_i,1:t-1}$	Current estimate of the state of a cell, C_i, given all observations up to time $t-1$.
$P_{\text{OCC}}^{C_i,1:t}$	Current estimate of the state of a cell, C_i, given all observations up to time t.
$p_{hit\|hit}$	Probability of true hit.
$p_{hit\|miss}$	Probability of false detection.
$p_{miss\|hit}$	Probability of missed detection.
$p_{miss\|miss}$	Probability of true miss.

The last four notations in Table 4.1 are design parameters that determine how fast the OGM will converge. Based on the position and heading estimation and LIDAR measurements, a ray is defined from the grid cell in which the LIDAR sensor is located and the grid cell in which the hit occurred. The cell in which the hit occurred is updated according to (4.1). Furthermore, the cells that the ray passes over are updated according to (4.2).

$$P_{\text{OCC}}^{C_i,1:t} = \frac{p_{hit|hit} P_{\text{OCC}}^{C_i,1:t-1}}{p_{hit|hit} P_{\text{OCC}}^{C_i,1:t-1} + p_{hit|miss}\left(1 - P_{\text{OCC}}^{C_i,1:t-1}\right)}. \tag{4.1}$$

$$P_{\text{OCC}}^{C_i,1:t} = \frac{p_{miss|hit} P_{\text{OCC}}^{C_i,1:t-1}}{p_{miss|hit} P_{\text{OCC}}^{C_i,1:t-1} + p_{miss|miss}\left(1 - P_{\text{OCC}}^{C_i,1:t-1}\right)}. \tag{4.2}$$

To handle dynamic events to some extent, the probability for each cell to be occupied is saturated between a lower and an upper threshold.

4.3 Parking Spot State

By combining the known structure from Section 4.1 and the OGM from Section 4.2, the state of the parking lot can be estimated. This is done by mapping the known structure of the parking spots to the corresponding position in the grid map as in Figure 4.1 where (a) shows a defined parking spot on top of the grid map and the center of the parking spot is marked with a line. In (b) the grid cells which the center of the truck passes through is marked in red. The mean of the values in the marked cells is used to determine the current state of the parking spot. Possible states of a parking spot are occupied, unknown and empty and is decided by comparing the calculated average, ξ, to a lower threshold, α_1, and an upper threshold, α_2 according to

$$\begin{cases} \xi \leq \alpha_1, & \text{Free,} \\ \alpha_1 < \xi \leq \alpha_2, & \text{Unknown,} \\ \xi > \alpha_2, & \text{Occupied.} \end{cases} \tag{4.3}$$

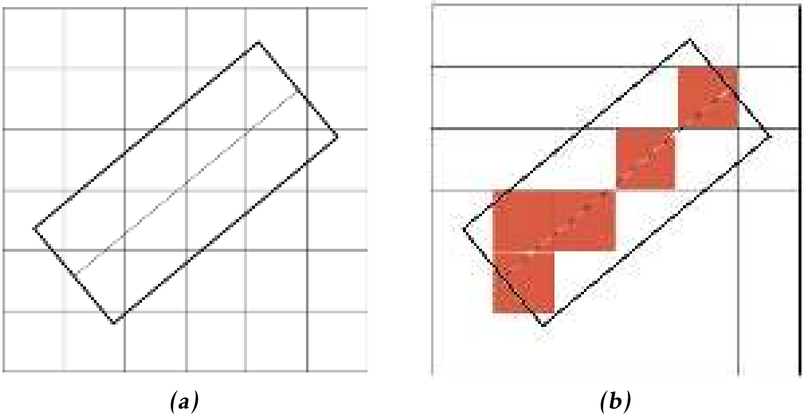

(a) (b)

Figure 4.1: (a) shows a defined parking spot on top of the grid map and the center of the parking spot is marked with a line. In (b) the grid cells which the center of the truck passes through is marked in red. The mean of the values in the marked cells is used to determine the current state of the parking spot.

4.4 Exploration of the Parking Lot

In Section 4.1, it is specified that the configuration file should contain a sequence of waypoints describing how the parking lot should be explored. This is a simple solution to the problem of searching an area which demands knowledge of the parking lot. If the waypoints are properly selected, the map should be able to determine the status of each of the defined parking spots from the configuration file when all waypoints have been visited.

The flowchart in Figure 4.2 shows and explains the different parts which are needed to complete the parking, from the initialization phase to the stop of the APS. If multiple free parking spots are updated as free at the same time, the parking detection system sends the parking spot which is defined first in the configuration file.

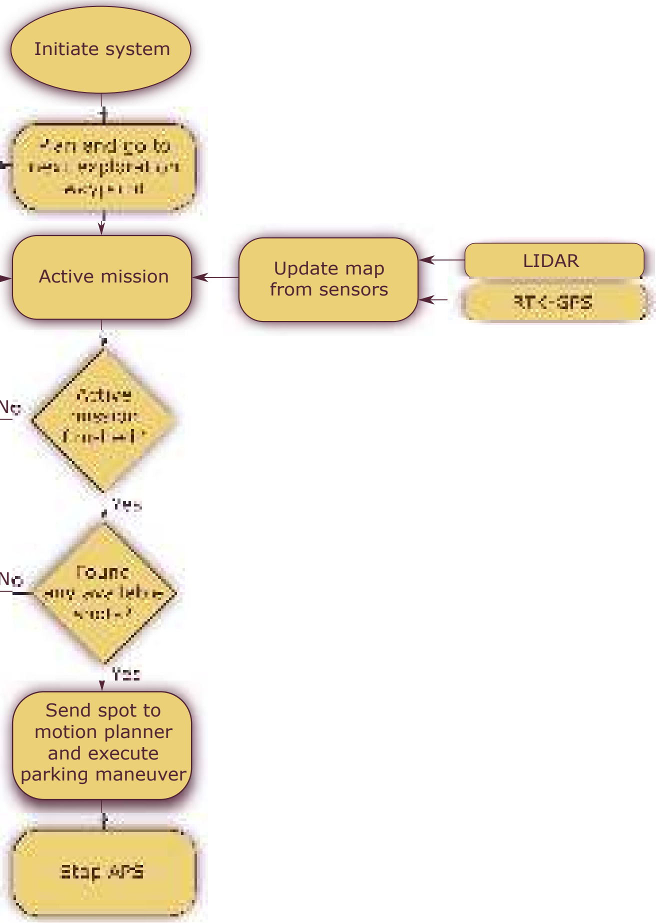

Initiate system
Plan and go to next exploration waypoint
Active mission
Update map from sensors
LIDAR
RTK-GPS
No
Active mission finished?
Yes
No
Found any available spots?
Yes
Send spot to motion planner and execute parking maneuver
Stop APS

Figure 4.2: *Flowchart for the parking spot detection system. In the initialization phase, the system is setup by loading the configuration file, creating an OGM, markers etcetera. Then the first waypoint from the configuration file is sent to the motion planner and the state is now* active mission. *The OGM is updated based on sensor readings as they are received. Once every second it is checked if the active mission is finished, if not, stay in the inner loop, if yes, check if a free parking spot has been located, if not, send the next waypoint from the configuration file to the motion planner and return to the inner loop, if yes, send the free parking spot to the motion planner and stop the system when the parking is finished.*

5 Modeling

In this chapter, a kinematic model of the vehicle will be derived. This model is then represented in a path coordinate system where the lateral error dynamics will be considered. Futhermore, this error model is linearized to enable the use of an LQ-controller and a linear MPC to perform the control of the vehicle.

5.1 Vehicle Model

A car-like nonholonomic vehicle model, also referred to as the bicycle model, is modeled and derived in [22] but will also be described here for sake of completeness:

$$\begin{aligned} \dot{x} &= v\cos\theta, \\ \dot{y} &= v\sin\theta, \\ \dot{\theta} &= \frac{v}{D}\tan\delta, \end{aligned} \tag{5.1}$$

where (x, y) are the global coordinates of the center of the rear axle, D is the length between the front and rear axle of the truck, θ is the heading of the vehicle, δ is the steering angle and v is the longitudinal velocity of the rear axle. A visualization of the model can be seen in Figure 5.1.

The curvature of the path that the vehicle undertakes is defined as

$$\kappa \triangleq \frac{\tan\delta}{D}. \tag{5.2}$$

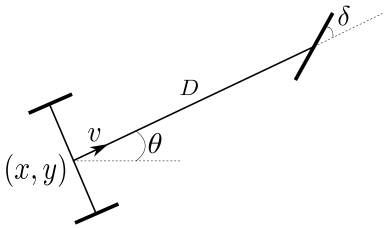


Figure 5.1: Visualization over the vehicle model where (x, y) are the global coordinates of the center of the rear axle, D is the length between the front and rear axle, θ is the heading of the vehicle, δ is the steering angle and v is the speed of the rear axle.

and can be used to simplify the expression in (5.1) to

$$\begin{aligned} \dot{x} &= v\cos\theta, \\ \dot{y} &= v\sin\theta, \\ \dot{\theta} &= v\kappa. \end{aligned} \tag{5.3}$$

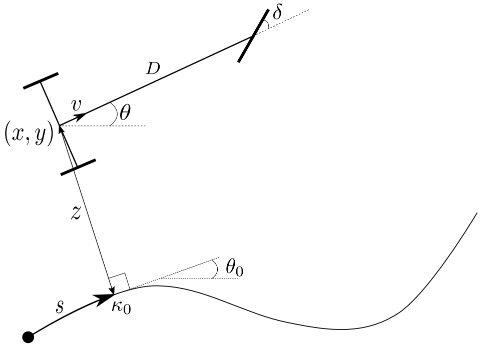


Figure 5.2: Visualization of the vehicle in the Frenet frame *coordinate system.*

An appropriate representation of the vehicle, suited for path following, is to transform the model in (5.3) from global coordinates onto the *Frenet frame*, see [22] for details, which is a moving coordinate system. With reference to Figure 5.2 let the nominal curvature, κ_0, along the reference path be defined as

$$\kappa_0 = \frac{d\theta_0}{ds}, \tag{5.4}$$

where θ_0 is the nominal heading of the path, s is the traveled distance along the path. Equation (5.4) implies that

$$\dot{\theta}_0 = \kappa_0 \dot{s}. \tag{5.5}$$

The reference point on the path is chosen to be such that the vehicle's rear axle is placed on its normal, i.e., the orthogonal projection of the vehicle onto the nominal path. Let

$$\tilde{\theta} = \theta - \theta_0, \tag{5.6}$$

then, according to [22],

$$\begin{aligned} \dot{s} &= v \cos\tilde{\theta} + \dot{\theta}_0 z, \\ \dot{z} &= v \sin\tilde{\theta}, \end{aligned} \tag{5.7}$$

where the second term in the uppermost equation relates the geometric shape of the path to where the vehicle is projected. Combining (5.5) and (5.7) yields

$$\begin{aligned} \dot{s} &= \frac{v \cos\tilde{\theta}}{1 - z\kappa_0}, \\ \dot{z} &= v \sin\tilde{\theta}. \end{aligned} \tag{5.8}$$

Furthermore, differentiating (5.6) yields

$$\dot{\tilde{\theta}} = \dot{\theta} - \dot{\theta}_0 = v\kappa - \kappa_0 \dot{s} = v\left(\kappa - \frac{\kappa_0 \cos\tilde{\theta}}{1 - z\kappa_0}\right). \tag{5.9}$$

The model in the Frenet frame is thus

$$\dot{s} = \frac{v \cos\tilde{\theta}}{1 - z\kappa_0}, \tag{5.10a}$$

$$\dot{z} = v \sin\tilde{\theta}, \tag{5.10b}$$

$$\dot{\tilde{\theta}} = v\left(\kappa - \frac{\kappa_0 \cos\tilde{\theta}}{1 - z\kappa_0}\right). \tag{5.10c}$$

5.2 Linearization Around the Path

The system derived in Section 5.1 describes the relationship between the vehicle and the reference path. To keep the vehicle within the proximity of the path a lateral controller needs to be designed. This controller is supposed to stabilize the lateral error dynamics by controlling the steering angle of the truck. Since there is a static mapping between the steering angle and the curvature, (5.2), the curvature will be considered as control signal. However, this project will use linear controllers which means that the nonlinear model in (5.10) cannot be used as it is. This can be managed by linearizing the model around a suitable field of operation, preferably the origin

$$\begin{aligned} z &= 0, \\ \tilde{\theta} &= 0, \\ u &= \kappa_0. \end{aligned} \tag{5.11}$$

The output from the controller has to be transferred through communication software to the actual actuator for the steering wheel, hence it is reasonable to assume that this signal is influenced by a delay τ. Using the method described in Section 2.1 the model in (5.10) can be linearized around (5.11). However, (5.10a) is disregarded because it is not needed for lateral control. Denote the lateral and angular error states as $w = \begin{bmatrix} z & \tilde{\theta} \end{bmatrix}^\mathsf{T}$. Around the origin $w = 0$ the linearized model becomes

$$\begin{bmatrix} \dot{z} \\ \dot{\tilde{\theta}} \end{bmatrix} = \begin{bmatrix} 0 & v \\ -v\kappa_0^2 & 0 \end{bmatrix} \begin{bmatrix} z \\ \tilde{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ v \end{bmatrix} \tilde{u}(t-\tau) \tag{5.12}$$

since the origin to (5.10b) and (5.10c) is zero and the definition

$$\tilde{u}(t-\tau) = u(t-\tau) - \kappa_0(t). \tag{5.13}$$

is used.

5.2.1 Integrated State

To enable the possibility to further suppress the error in the distance to the path the integrated state of z is added, referred to as z_{int}. In order to have a distance dependend integration, z_{int} is scaled with the speed of the truck

$$\dot{z}_{int} = |v|z. \tag{5.14}$$

The final linearized model then becomes

$$\begin{bmatrix} \dot{z}_{int} \\ \dot{z} \\ \dot{\tilde{\theta}} \end{bmatrix} = \begin{bmatrix} 0 & |v| & 0 \\ 0 & 0 & v \\ 0 & -v\kappa_0^2 & 0 \end{bmatrix} \begin{bmatrix} z_{int} \\ z \\ \tilde{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ v \end{bmatrix} \tilde{u}(t-\tau) \tag{5.15}$$

which is a linear model when either $v \geq 0$ or $v \leq 0$. Let $p = \begin{bmatrix} z_{int} & z & \tilde{\theta} \end{bmatrix}^{\mathsf{T}}$ for future references.

5.3 Discretization of the Linearized Model

The linearized model developed in Section 5.2 will be used to design the lateral controller, therefore it is important that the model itself is discrete because the controller will be discrete. Another difficulty regarding the implementation is the number of variables in the system, as of this moment both the speed v and the reference curvature κ_0 will vary throughout the driving. Some values will have to be implemented as look-up tables where each variable will present a new dimension. One of these look-up tables is the feedback gain matrix from the LQ-control in Section 6.3 which is solved offline and then stored. In Figure 7.42 the feedback gain matrix is visualized parameterized in speed for -1.0 m/s, 0.5 m/s and 2.0 m/s and in reference curvature between [-0.16 0.16] with the following cost matrices

$$Q = \begin{bmatrix} 0.001 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R = 1. \tag{5.16}$$

The gain scheduling error is shown in Table 5.1 where the difference in speed is for $v = 0.1$ m/s and $v = 2.0$ m/s at $\kappa_0 = 0$ m^{-1} and the difference in curvature is for $\kappa_0 = 0$ m^{-1} and $\kappa_0 = \pm 0.16$ m^{-1} for $v = 1$ m/s.

Since the controller primary will work around $\kappa_0 = 0$ and seldom for curvatures larger than $|\kappa_0| > 0.1$ the decision to only parameterize in velocity with $\kappa_0 = 0$ was made to simplify the model. With $\kappa_0 = 0$ in (5.15), the model then becomes

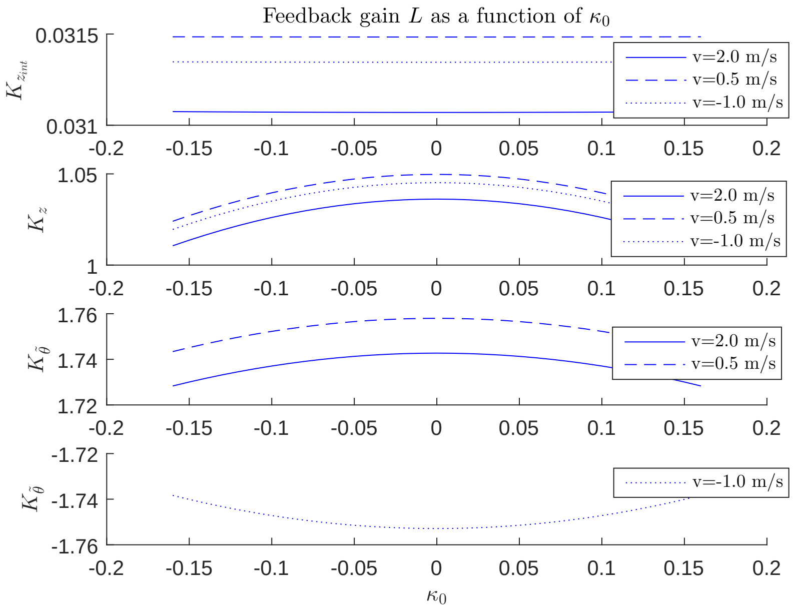


Figure 5.3: *Visualization of the feedback gain matrix for the speed -1.0 m/s, 0.5 m/s and 2.0 m/s as a function of the reference curvature κ_0.*

$$\dot{p} = \begin{bmatrix} 0 & |v| & 0 \\ 0 & 0 & v \\ 0 & 0 & 0 \end{bmatrix} p + \begin{bmatrix} 0 \\ 0 \\ v \end{bmatrix} \tilde{u}(t-\tau). \tag{5.17}$$

By using the theory in Section 2.2, with the assumption that the speed is constant over a sample time interval, the discrete model becomes

$$p_{k+1} = Fp_k + G\tilde{u}_{k-\alpha}, \tag{5.18}$$

where $\alpha = \frac{\tau}{T_s}$, T_s is the sampling time and the matrices F and G are:

$$F = \begin{bmatrix} 1 & |v|T_s & \frac{|v|vT_s^2}{2} \\ 0 & 1 & vT_s \\ 0 & 0 & 1 \end{bmatrix}, \quad G = \begin{bmatrix} \frac{|v|v^2T_s^3}{6} \\ \frac{v^2T_s^2}{2} \\ vT_s \end{bmatrix}. \tag{5.19}$$

Table 5.1: *Gain scheduling error. In order to simplify the model, it was investigated if it was reasonable to assume constant curvature ($\kappa_0 = 0\ m^{-1}$) or constant velocity ($v = \pm 1\ m/s$). The difference in feedback gain is between v = 0.1 m/s and $v = 2.0\ m/s$ for a constant $\kappa_0 = 0\ m^{-1}$ is listed as* ***Difference in speed [%]****.* ***Difference in curvature [%]*** *lists the feedback gain difference between $\kappa_0 = 0\ m^{-1}$ and $\kappa_0 = \pm 0.16\ m^{-1}$ for the constant velocity v = 1 m/s.*

Gain	Difference in speed [%]	Difference in curvature [%]
$K_{z_{int}}$	1.67	0.03
K_z	1.64	2.47
$K_{\tilde{\theta}}$	1.09	0.87

6
Control

This chapter will describe how the lateral control of the vehicle is designed. The control system itself consists of four minor subsystems necessary to obtain a fully working lateral controller. The output from the control system has to be sent over the communication network in the vehicle, where a short delay of τ s occur (classified information). Since there is a delay acting on the output of the controller the system itself tries to reduce its effect by predicting the future vehicle position hence neglecting the delay, described in Section 6.1. The global coordinates of the vehicle is then transformed into the *Frenet frame* by extracting information from the reference path, described in Section 6.2. Once the states of the vehicle have been calculated a regulator will provide the system with a requested curvature. Both an LQ-controller and an MPC-controller have been developed for the steering, they are presented in Section 6.3 and Section 6.4 respectively. The final part of the system is a transformation from curvature to the corresponding angle of the steering wheel on the truck.

6.1 Simulation of State Due to Input Delay

The input delay, τ, has to be taken into account in order to increase the tracking performance. According to Section 2.3 the forward simulated state $\hat{p}_k$, which is the estimated state after the input delay has elapsed, i.e.,

$$\hat{p}_k = p_{k+\alpha}, \tag{6.1}$$

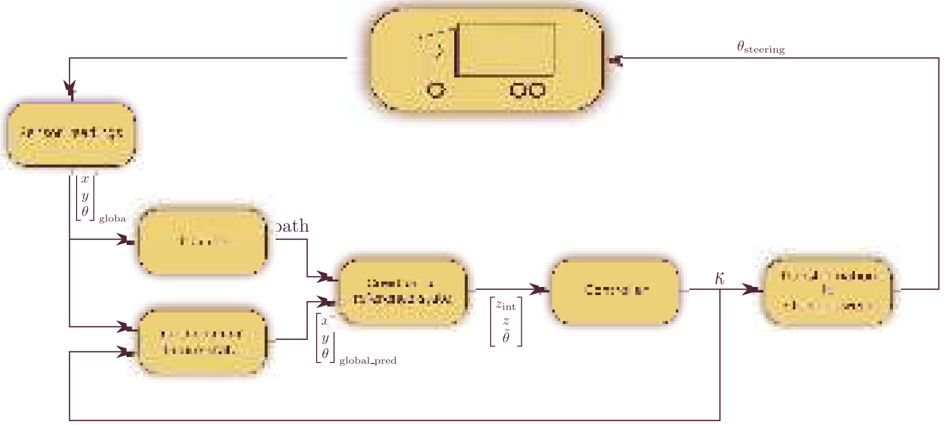

θ_{steering}
$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{\text{globa}}$
path
$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{\text{global\_pred}}$
$\begin{bmatrix} z_{\text{int}} \\ z \\ \tilde{\theta} \end{bmatrix}$
Controller
κ

Figure 6.1: Overview of the lateral control system.

where p is the state vector from (5.15) and α is the input delay in number of sampling periods. From Chapter 5, the control signal is partially composed of the nominal curvature, κ_0. This means that the model in (5.15) cannot solely be used for this simulation since the curvature reference has to be estimated. However, the bicycle model in (5.3) can be used to estimate the vehicle's global position. As this is to be implemented online, (5.3) has to be discretized. The heading at time $T + T_s$ can be calculated as

$$\theta(T + T_s) = \theta(T) + \int_{T}^{T+Ts} \dot{\theta}\, dt \tag{6.2}$$

Using (5.3) once again with the knowledge of the input delay we obtain

$$\theta(T + T_s) = \theta(T) + \int_{T}^{T+T_s} v(t)u(t - \tau)\, dt. \tag{6.3}$$

u and v can be approximated as constants over a sampling period and the delay is known, hence

$$\theta_{k+1} = \theta_k + T_s v_k u_{k-\alpha}. \tag{6.4}$$

The heading θ in the equations for the global coordinates, (5.1), is not constant over a sampling period hence it is better to use Tustins Formula [13] for the discretization of those two equations,

$$d = \frac{2}{T_s}\frac{q^{+1} - 1}{q^{+1} + 1}, \tag{6.5}$$

where d is the derivative operator and q is the shift operator. Applying this formula onto the equations for the global coordinates in (5.3) gives

$$\begin{aligned} x_{k+1} &= x_k + \frac{vT_s}{2}\cos\theta_{k+1} + \frac{vT_s}{2}\cos\theta_k, \\ y_{k+1} &= y_k + \frac{vT_s}{2}\sin\theta_{k+1} + \frac{vT_s}{2}\sin\theta_k. \end{aligned} \tag{6.6}$$

Equation (6.4) and (6.6) can now be used iteratively to predict the global coordinates at τ s into the future using the stored curvatures $u_{k-\alpha}, \ldots, u_{k-1}$ and the measured states θ_k, x_k and y_k. By using the method described in Section 6.2 the reference state τ s into the future can now be extracted and used for the controller.

6.2 Selection and Creation of Reference States

The reference path, provided by the motion planner described in Section 3.2, is composed of a finite amount of points with the additional properties of the path, such as curvature, heading and position. A controller needs some sort of a reference to act desirably upon the system, therefore there should exist logic to extract these references from the path. One solution is to use the reference from the closest point on the path; however, this will create a discontinuous reference as the point which is closest changes. Hence, another method must be used to gain a continuous reference. By projecting the vehicle orthogonally onto the path and then interpolating the reference from the adjacent points a mainly continuous reference is obtained. To proceed with this method the closest point on the path must be extracted. Let a point in the reference path be defined as (x_p, y_p). Then the nearest point is

$$i = \arg\min_k \left(\left(x - x_{p,k} \right)^2 + \left(y - y_{p,k} \right)^2 \right), \tag{6.7}$$

where $k = 1, 2, \ldots, M$ and M is the number of points in the reference path. Once the closest point on the path, with index i, is found the vehicle can be projected onto the preceding path segment using (6.8) and the succeeding path segment using (6.9).

$$\begin{aligned} \bar{v}_1 &= \begin{bmatrix} x_{p,i} - x_{p,i-1} \\ y_{p,i} - y_{p,i-1} \end{bmatrix} \\ \bar{u}_1 &= \begin{bmatrix} x - x_{p,i-1} \\ y - y_{p,i-1} \end{bmatrix} \\ a_1 &= \frac{\bar{u}_1 \cdot \bar{v}_1}{|\bar{v}_1|^2} \end{aligned} \tag{6.8}$$

$$\begin{aligned} \bar{v}_2 &= \begin{bmatrix} x_{p,i+1} - x_{p,i} \\ y_{p,i+1} - y_{p,i} \end{bmatrix} \\ \bar{u}_2 &= \begin{bmatrix} x - x_{p,i} \\ y - y_{p,i} \end{bmatrix} \\ a_2 &= \frac{\bar{u}_2 \cdot \bar{v}_2}{|\bar{v}_2|^2} \end{aligned} \tag{6.9}$$

Here v_j is the vector from the first point in the path segment to the second, $\bar{u}_j$ is the vector from the first point in the path segment to the vehicle and $a_j \bar{v}_j$ is the vector from the first point in the path segment to the area where the vehicle is projected onto the path. This is visualized in Figure 6.2 and Figure 6.3. By using the values on a_1 and a_2 the vehicle's position relative the path can be identified according to Table 6.1. Depending on where the vehicle is, related to the path, the reference state is interpolated between different points on the path, according to Table 6.2. Note that there is a discontinuity when moving from the situation in Figure 6.4 to Figure 6.6; however, the impact of this is negligible due to the curvature of the path being small.

Anti-Windup of Integrated State

To ensure that the integrated state introduced in Section 5.2.1 will not introduce an undesired oscillative movement pattern in the path following control, the

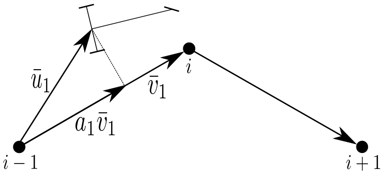


Figure 6.2: *The vector $\bar{u}_1$ projected on the line between $[i-1, i]$.*

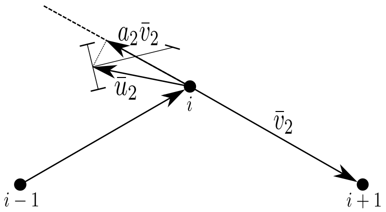


Figure 6.3: *The vector $\bar{u}_2$ projected on the line between $[i, i+1]$.*

Table 6.1: *How to identify the vehicle's relation to the path using the values on a_1 and a_2. Depending on these values the vehicle is paired with one of the situations depicted in Figure 6.4-6.7.*

a_1	a_2	**Vehicle state**
< 1	< 0	Figure 6.4
≥ 1	≤ 0	Figure 6.5
≥ 1	≤ 1	Figure 6.6
≤ 1	≤ 1	Figure 6.7

Table 6.2: *Which indices to interpolate between depending on which one of the four situation depicted in Figure 6.4-6.7 that is active.*

Situation	**Extraction of reference state**
Figure 6.4	Interpolation between $i-1$ and i
Figure 6.5	The unaffected values from i
Figure 6.6	Interpolation between i and $i+1$
Figure 6.7	Interpolation between i and $i+1$

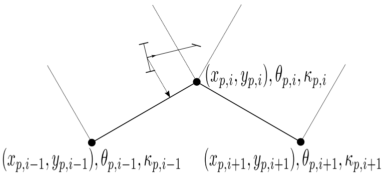

$(x_{p,i},y_{p,i}),\theta_{p,i},\kappa_{p,i}$
$(x_{p,i-1},y_{p,i-1}),\theta_{p,i-1},\kappa_{p,i-1}$
$(x_{p,i+1},y_{p,i+1}),\theta_{p,i+1},\kappa_{p,i+1}$

Figure 6.4: *When the vehicle only can be projected onto the line between $[i-1, i]$ the reference state is interpolated between these states.*

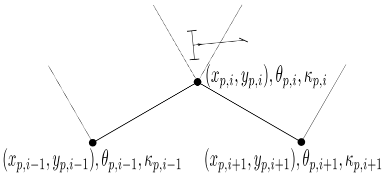

$(x_{p,i},y_{p,i}),\theta_{p,i},\kappa_{p,i}$
$(x_{p,i-1},y_{p,i-1}),\theta_{p,i-1},\kappa_{p,i-1}$
$(x_{p,i+1},y_{p,i+1}),\theta_{p,i+1},\kappa_{p,i+1}$

Figure 6.5: *When the vehicle cannot be projected onto any line the chosen reference state is i.*

model is equipped with an *anti-windup* function. This function monitors the value of z_{int} and restricts it to operate in a certain interval; this means that z_{int}

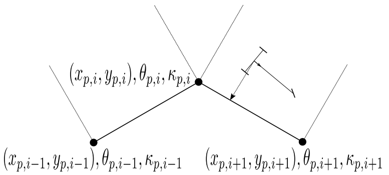

$(x_{p,i}, y_{p,i}), \theta_{p,i}, \kappa_{p,i}$
$(x_{p,i-1}, y_{p,i-1}), \theta_{p,i-1}, \kappa_{p,i-1}$
$(x_{p,i+1}, y_{p,i+1}), \theta_{p,i+1}, \kappa_{p,i+1}$

Figure 6.6: *When the vehicle only can be projected onto the line between $[i, i+1]$ the reference state is interpolated between these states.*

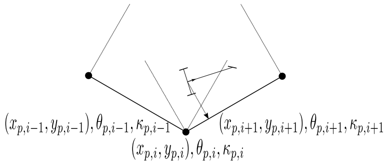

$(x_{p,i-1}, y_{p,i-1}), \theta_{p,i-1}, \kappa_{p,i-1}$
$(x_{p,i+1}, y_{p,i+1}), \theta_{p,i+1}, \kappa_{p,i+1}$
$(x_{p,i}, y_{p,i}), \theta_{p,i}, \kappa_{p,i}$

Figure 6.7: *When the vehicle can be projected onto both line $[i-1, i]$ and $[i, i+1]$ the latter is chosen for better path following performance.*

will not tend to infinity even if z is positive $\forall t$.

Anti-Updating of Integrated State

If the vehicle happens to be far away from the reference path, the control system's reaction to z and $\tilde{\theta}$ should be enough to steer the vehicle back onto the path. Therefore, the integrated state is equipped with *Anti-Updating Logic* to prevent the integrated state from being updated when further than one meter away from the path. This leads to the contribution from the integrated state being bounded when far away from the path, preventing the integrated state from inducing oscillations when moving back to the path.

6.3 LQ-Control

The LQ-controller is a state-feedback controller which tries to minimize a set of states and usage of a control signal described as

$$\begin{aligned} \min_{u(\cdot)} \quad & \sum_{k=0}^{\infty} p_k^T Q p_k + u_k^2, \\ \text{subject to} \quad & p_{k+1} = F p_k + G \tilde{u}_{k-\alpha}, \\ & p_0 \text{ given}, \end{aligned} \tag{6.10}$$

where $Q \in \mathbb{S}_+^3$, u is the curvature of the vehicle, F and G are defined in (5.19) and p is defined from (5.15). It was chosen as a controller for its convenient and practical method to design its parameters. The behavior of the controller is decided by prioritizing the input signal and the different states relatively to eachother by adjusting Q. According to Section 2.3 a system, where there exists a delay in the input signal, the optimal solution to the problem in (6.10) is solved by the predicted state feedback

$$\tilde{u}_k = -L\hat{p}_k, \tag{6.11}$$

where

$$\begin{aligned} L &= (G^T S G + 1)^{-1} G^T S F, \\ S &= F^T S F + Q - F^T S G (G^T S G + 1)^{-1} G^T S F, \end{aligned} \tag{6.12}$$

and where F and G describe the system and are defined in (5.19). From (5.13) we then get the requested curvature as

$$u_k = \tilde{u}_k + \kappa_{0,k+\alpha}. \tag{6.13}$$

The discrete time model is described in Section 5.3 where it is stated that the system depends on the speed. This means that the solution to (6.12) varies with the speed, and hence, cannot be solved only once but several times depending on the speed of the vehicle. However, the solution to this equation is not trivial and in our project the means to solve it online is not available, hence an offline solution is needed. That is, the feedback gain matrix is calculated offline for different speeds and then stored as a lookup-table available for the controller online. The performance of the controller described in this section is presented in Section 7.3.2.

6.4 Model Predictive Control

MPC can be seen as an evolution of the LQ-controller where it now is possible to impose constraints on the states and the control signals. In the development of the LQ-controller we noticed that the steering wheel often moved at relatively high speeds giving the driver a nervous feeling as it did not act human-like. Therefore, one of the ideas behind the MPC was to formulate a constraint which restricted the angular velocity of the steering wheel. There also exists a maximum curvature the vehicle is able to maneuver, which is included as a constraint in the controller optimization problem. The problem the MPC-controller solves is

$$
\begin{aligned}
\min_{u(\cdot)} \quad & \hat{p}_{N_p}^T Q_{N_p} \hat{p}_{N_p} + \sum_{k=0}^{N_p-1} \hat{p}_k^T Q \hat{p}_k + u_k^2, \\
\text{subject to} \quad & \hat{p}_{k+1} = F\hat{p}_k + Gu_k, \\
& u(\cdot) \in \mathbb{U}, \\
& p(\cdot) \in \mathbb{P}, \\
& \hat{p}_0 \text{ given.}
\end{aligned}
\tag{6.14}
$$

In Section 2.4 a method to solve (6.14) was presented. To be able to follow this method the constraints in (6.14) has to be written as

$$
\begin{aligned}
c_{\mathcal{P}_{\text{low}}} &\le C\mathcal{P} \le c_{\mathcal{P}_{\text{high}}} \\
d_{U_{\text{low}}} &\le DU \le d_{U_{\text{high}}},
\end{aligned}
\tag{6.15}
$$

where

$$
\mathcal{P} = \begin{bmatrix} \hat{p}_{k+1} \\ \hat{p}_{k+2} \\ \vdots \\ \hat{p}_{k+N_p} \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N_p-1} \end{bmatrix},
\tag{6.16}
$$

According to Section 2.4 the terminal state weight, S, should be chosen as the solution to the discrete algebraic Riccati equation for the corresponding infinite horizon problem without constraints, see Section 2.3. However, similar to the LQ controller in Section 6.3 difficulties appear for the implementation. The terminal state weight cannot be solved online since suitable software is not available. Hence, the solution is calculated offline, for a set of different velocities, and stored as a look-up table available for the MPC online.

6.4.1 Constraints

The constraints were chosen to adapt the controller more to a human-like behavior and to achieve more stable path-following properties. The constraints on the states are simple minimum and maximum limits which can be written as

$$
c_{\text{low}} \le p_k \le c_{\text{high}}. \tag{6.17}
$$

This can then be written for all indices k as

$$\underbrace{\begin{bmatrix} c_{\text{low}} \\ \vdots \\ c_{\text{low}} \end{bmatrix}}_{c_{\mathcal{P}_{\text{low}}}} \leq \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}}_{C} \mathcal{P} \leq \underbrace{\begin{bmatrix} c_{\text{high}} \\ \vdots \\ c_{\text{high}} \end{bmatrix}}_{c_{\mathcal{P}_{\text{high}}}}. \tag{6.18}$$

The constraints on the control signal are minimum and maximum limits on the amplitude of the control signal and the derivative of the control signal. Notice that the control signal used as an input to (6.14) is the difference between requested and ideal curvature, $\tilde{u}$, and has to be taken into consideration when formulating these constraints. The constraint on the amplitude can be written as

$$u_{\text{min}} \leq u_k \leq u_{\text{max}}. \tag{6.19}$$

Using $u_k = \tilde{u}_k + \kappa_{0,k+\alpha}$ then gives

$$u_{\text{min}} - \kappa_{0,k+\alpha} \leq \tilde{u}_k \leq u_{\text{max}} - \kappa_{0,k+\alpha}, \tag{6.20}$$

which can further be expressed for all indices as

$$\underbrace{\begin{bmatrix} u_{min} - \kappa_{0,k+\alpha} \\ u_{min} - \kappa_{0,k+\alpha+1} \\ \vdots \\ u_{min} - \kappa_{0,k+\alpha+N_p-1} \end{bmatrix}}_{U_{\text{lowAmp}}} \leq \underbrace{I_{N_p \times N_p}}_{D_{\text{amp}}} \tilde{U} \leq \underbrace{\begin{bmatrix} u_{max} - \kappa_{0,k+\alpha} \\ u_{max} - \kappa_{0,k+\alpha+1} \\ \vdots \\ u_{max} - \kappa_{0,k+\alpha+N_p-1} \end{bmatrix}}_{U_{\text{highAmp}}}. \tag{6.21}$$

The constraint on the derivative can be expressed as

$$\Delta u_{\text{min}} \leq u_k - u_{k-1} \leq \Delta u_{\text{high}}. \tag{6.22}$$

Once again, $u = \tilde{u} + \kappa_0$ can be used to rewrite (6.22) to

$$\Delta u_{\text{min}} - \kappa_{0,k+\alpha} + \kappa_{0,k+\alpha-1} \leq \tilde{u}_k - \tilde{u}_{k-1} \leq \Delta u_{\text{max}} - \kappa_{0,k} + \kappa_{0,k-1}. \tag{6.23}$$

Generalizing (6.23) to all indices gives

$$\begin{aligned}
&\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix} \tilde{U} \leq \underbrace{\begin{bmatrix} \Delta u_{\max} + \kappa_{0,k+\alpha-1} + \tilde{u}_{k-1} - \kappa_{0,k+\alpha} \\ \Delta u_{\max} + \kappa_{0,k+\alpha} - \kappa_{0,k+\alpha+1} \\ \Delta u_{\max} + \kappa_{0,k+\alpha+1} - \kappa_{0,k+\alpha+2} \\ \vdots \\ \Delta u_{\max} + \kappa_{0,k+\alpha+N_p-2} - \kappa_{0,k+\alpha+N_p-1} \end{bmatrix}}_{U_{\text{highDeriv}}}, \\
&\underbrace{\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix}}_{D_{\text{deriv}}} \tilde{U} \geq \underbrace{\begin{bmatrix} \Delta u_{\min} + \kappa_{0,k-1} + \tilde{u}_{k-1} - \kappa_{0,k} \\ \Delta u_{\min} + \kappa_{0,k} - \kappa_{0,k+1} \\ \Delta u_{\min} + \kappa_{0,k+1} - \kappa_{0,k+2} \\ \vdots \\ \Delta u_{\min} + \kappa_{0,k+N_p-2} - \kappa_{0,k+N_p-1} \end{bmatrix}}_{U_{\text{lowDeriv}}}.
\end{aligned} \tag{6.24}$$

Combining (6.21) and (6.24) now gives

$$\underbrace{\begin{bmatrix} U_{\text{lowAmp}} \\ U_{\text{lowDeriv}} \end{bmatrix}}_{d_{U_{\text{low}}}} \leq \underbrace{\begin{bmatrix} D_{\text{amp}} \\ D_{\text{deriv}} \end{bmatrix}}_{D} \tilde{U} \leq \underbrace{\begin{bmatrix} U_{\text{highAmp}} \\ U_{\text{highDeriv}} \end{bmatrix}}_{d_{U_{\text{high}}}}, \tag{6.25}$$

From here the method in Section 2.4 is used to obtain the optimal $\tilde{u}_k$ and the optimal curvature u_k can then be extracted as $u_k = \tilde{u}_k + \kappa_{0,k+\alpha}$. The performance of this controller is presented in Section 7.3.3.

7

Results

This chapter presents the results that have been obtained using the proposed APS framework. This section starts by evaluating each component in the APS separately and in the end the complete APS system is evaluated.

7.1 Occupancy Grid Map

Figure 7.1 shows the resulting OGM after manually collecting data at a parking lot by driving a complete lap. Grid cells in the color green corresponds to an unknown state, purple to a low probability that the cell is occupied and red to a high probability that the cell is occupied. Table 7.1 shows the values of the design parameters defined in Table 4.1 and (4.3). The same values are used for all testing in this chapter.

The visualization of the OGM in real-time turned out to be unprecedentedly computationally heavy. The solution to this problem is to visualize the states of the parking spots alone, without the OGM. This is illustrated in Figure 7.2 where (a) shows the state of two parkings and the truck on top of the OGM. (b) shows the same information and also the driven path but without the OGM, making it possible to run the system in real-time. The color red specifies that the parking spot is occupied and the color green that the parking spot is available.

Table 7.1: *Design parameters defined in Table 4.1 and* (4.3) *and their values during the tests in this chapter.*

Parameter	**Value**	**Description**
$P_{\text{OCC}}^{C_i,t=0}$	0.5	Initialization of OGM.
$p_{hit\|hit}$	0.6	Probability of true hit.
$p_{hit\|miss}$	0.4	Probability of false detection.
$p_{miss\|hit}$	0.3	Probability of missed detection.
$p_{miss\|miss}$	0.7	Probability of true miss.
α_1	0.2	Lower threshold for parking spot state.
α_2	0.6	Upper threshold for parking spot state.

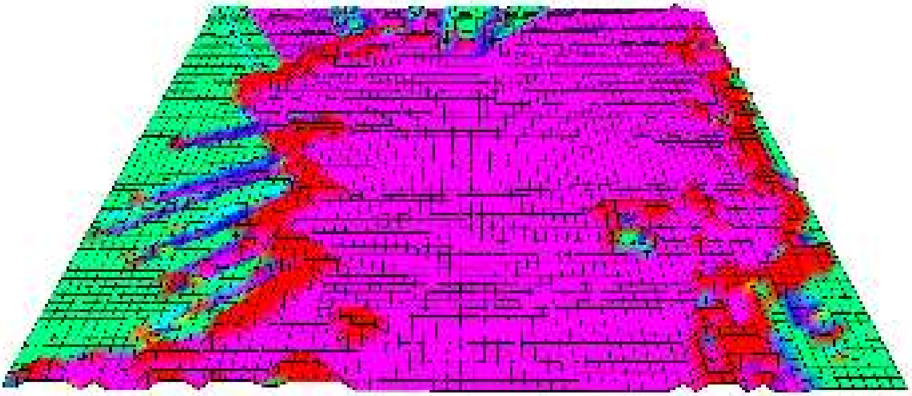


Figure 7.1: *Visualization over a test run at a parking lot. Grid cells in the color green corresponds to an unknown state, purple is a low probability that the cell is occupied and red is a high probability that the cell is occupied.*

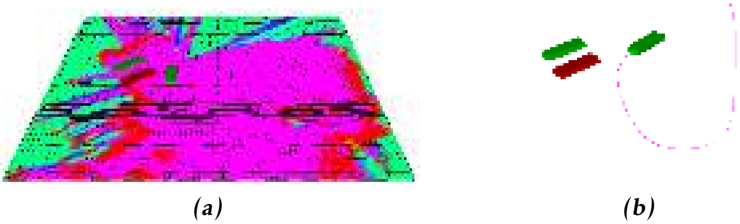
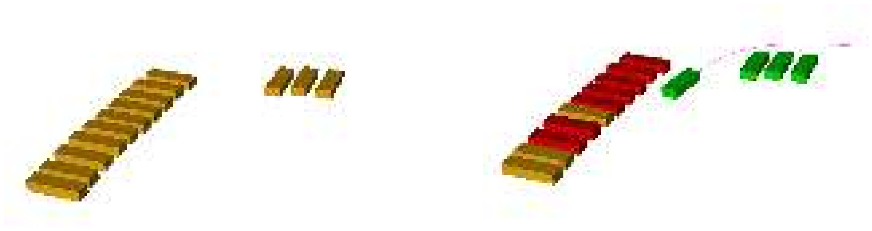


Figure 7.2: *(a) shows the state of two parkings and the truck on top of the OGM. (b) shows the same information and also the driven path but without the OGM, making it possible to run the system in real-time. The color red specifies that the parking spot is occupied and the color green that the parking spot is available.*

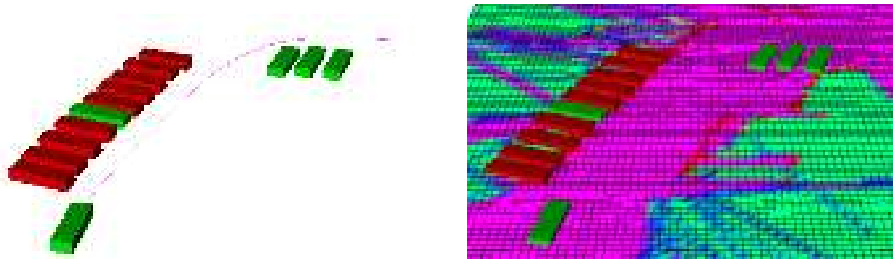
A test in a larger parking lot resulted in Figure 7.3 and Figure 7.4. The configuration file included 13 parking spots, illustrated in Figure 7.3a where the state of each parking spot is unknown (orange). Figure 7.3b illustrates the result halfway through the test and at this time instance three parking spots are classified as free, seven are occupied and three are unknown. The end of the test is shown in Figure 7.4a where the state of each parking spot is determined and Figure 7.4b shows the resulting OGM over the parking lot. The result of the test coincided with reality; hence a successful test with correct classification of the states of the parking spots.

The thresholds α_1 and α_2 defined in (4.3) are design parameters and are used to determine the state of each parking spot. It is essential that α_1 is set low to avoid setting the state as free when it is occupied. The upper limit, α_2, is only relevant for visualization to the operator since the states unknown and occupied belong to the set of parking spots which is not available to park on.



(a) *(b)*

Figure 7.3: *(a) shows the test setup with 13 parking spots in an unknown state (orange). In (b) the truck has travelled along the purple line and three parking spots have been updated with the state free (green), seven parking spots have the state occupied (red) and three parking spots are still of unknown state (orange).*



(a) *(b)*

Figure 7.4: *Continuation from Figure 7.3 when all parking spots are updated with the correct state shown with and without OGM.*

The test in this section shows that it is possible to update an OGM based on GPS- and LIDAR-measurements sufficiently accurate to determine the states of the parking spots based on the OGM and the information in the configuration file. This shows that the parking spot detection system described in Chapter 4 works as intended.

7.2 Model Validation

This section will evaluate the model derived in Section 5.1 by comparing data from a real HDV with the output from the model. The test was performed in manual mode by driving around and collecting data of the position, heading and speed. The input to the model is the curvature, κ, which was calculated according to (5.4) from the ground truth given by the RTK-GPS. To simulate the model the discretized equations in (6.6) are used for the global position. The equation for the heading in (6.4) is exchanged with (7.1) due to being a better approximation when κ is not constant during a sample interval. The model was then simulated with the calculated curvature and the logged speed.

$$\theta[T + T_s] = \theta[T] + \frac{T_s}{2} v u[T] + \frac{T_s}{2} v u[T + T_s] \tag{7.1}$$

The test performed is sufficiently long to validate the performance of the model, it was executed by driving in forward motion at roughly 3.3 m/s with a flying start. The true position compared with the modeled position with knowledge of the initial position can be seen in Figure 7.5. Similarly, the true and modeled heading can be seen in Figure 7.6. From these figures, it can be seen that the model is accurate enough to predict the vehicle's state trajectory within the prediction horizon of the MPC controller. Since this is an open-loop simulation and the system is marginally stable, long predictions will start to deviate from the true trajectory of the vehicle.

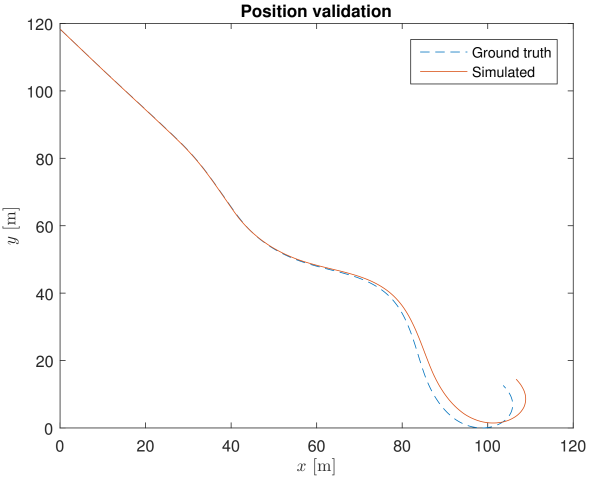


Figure 7.5: *Positions x and y from the RTK-GPS in dotted blue and the simulated position in red.*

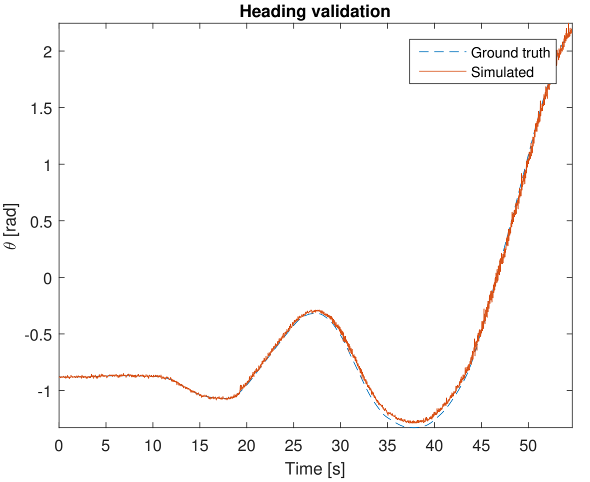


Figure 7.6: *Heading, θ, from the RTK-GPS in dotted blue and simulated heading in red.*

7.3 Lateral Control

This section presents the performance of the lateral controllers described in Chapter 6. Several different scenarios have been carried out, following different paths in forward and backward motion. The purpose of the lateral controller is to keep the vehicle on the reference path, therefore the distance from path, z, is chosen as performance measure. This distance to path is usually presented along the velocity of the vehicle to get a better understanding of the driving process. If nothing else is said the start position of the vehicle is (0,0) in all figures showing the vehicle's position during a test. The tests in the advanced environment and the HDV had a saturated velocity between −1 and 2 m/s.

7.3.1 Test Environments

The goal of this thesis was always to end up with an APS operating in an actual vehicle. During the development, a lot of time was spent in simulation environments to ensure that the system worked before testing it on the full-scale test platform. This was done in a basic and in an advanced simulation environment, to achieve a desired behavior before tests in a real vehicle were considered to be sufficiently safe and profitable. The basic simulation environment only provides the necessary components from the control system while the other two uses the complete system. The complete system is implemented in MATLAB using SIMULINK. However, the system is implemented in MATLAB under certain restrictions which enables code generation to C-code. The generated C-code can then be used to create an executable program with the means of communication with other programs, this was needed for the advanced simulation environment and for tests on the real vehicle.

Basic Simulation Environment

MATLAB and SIMULINK were used to create the basic environment. The environment consists of the bicycle model described in Section 5.1 to model the vehicle in relation to a generated path and the basic parts of the control system from the APS. The environment was created in the development of the APS and provided sufficient means for basic simulation and proof of concepts.

Advanced Simulation Environment

The advanced simulation environment is provided by Scania CV AB and consist of a complete vehicle system, divided in several separate modules as executable programs with means for communication in between them, a vehicle model and a graphical user interface. This environment was used to get a more accurate impression of the performance since it contains a more representative model of the vehicle. It also gave an understanding of different issues which could be encountered when cooperating with other systems.

Heavy Duty Vehicle

Once a reliable controller was designed in the advanced simulation environment it was tested in the actual vehicle. The vehicle was provided by Scania CV AB and is of type SCANIA R 580, see Figure 3.2 for a picture of the vehicle.

7.3.2 Linear Quadratic Control

This section will present results from the control system when using the LQ controller described in Section 6.3. Results from both simulation environments, basic and advanced, will be presented as well as results from the actual vehicle. The LQ controller used in this section uses the cost matrix

$$Q = \text{diag}\left(\begin{bmatrix}0.0001 & 0.1 & 0.1\end{bmatrix}\right),$$

resulting in the feedback gain matrix $L = \begin{bmatrix}0.0097 & 0.3316 & 0.8692\end{bmatrix}^T$ for $v = 2$ m/s. The system was running in 50 Hz.

Basic Simulation Environment

As already stated the basic simulation environment was used for proof of concepts. Therefore, in Figure 7.7-7.9 the vehicle information from a test is shown where the vehicle's state was initialized with errors. The vehicle should follow a circle with a radius of 20 meters with the speed 2 m/s. However, the vehicle started one meter above the path (10,-9) with an offset in the heading by $\frac{\pi}{6}$ radians.

Regarding the fundamental concept of the linear quadratic lateral controller, Figure 7.9 shows that all states are converging to zero. This aligns with the expected behavior of the controller in a perfect world with no noise or model errors presented. Here the effects of the *Anti-Updating* can be seen in action, while the distance from the path, z, is above one meter the integral of the distance from the path, z_{int}, will not update. We can see from Figure 7.8 that the vehicle curvature already is maximized during the major part of the duration when $z > 1$ m, hence the impact of an increased z_{int} would have been neglected. However, when closing in on the path z_{int} is now less than it would have been without the *Anti-Updating*, reducing the probability of oscillations to occur when reaching the origin.

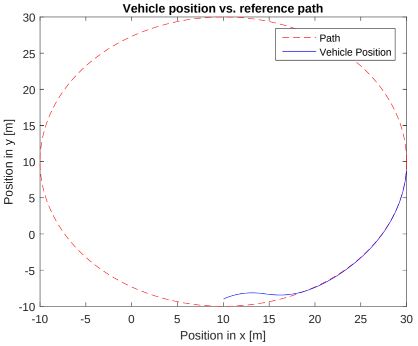


Figure 7.7: Visualization of the vehicle driven path and the reference path.

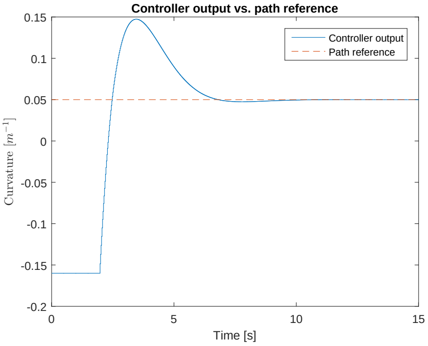


Figure 7.8: Visualization of the controller output with the curvature of the reference path.

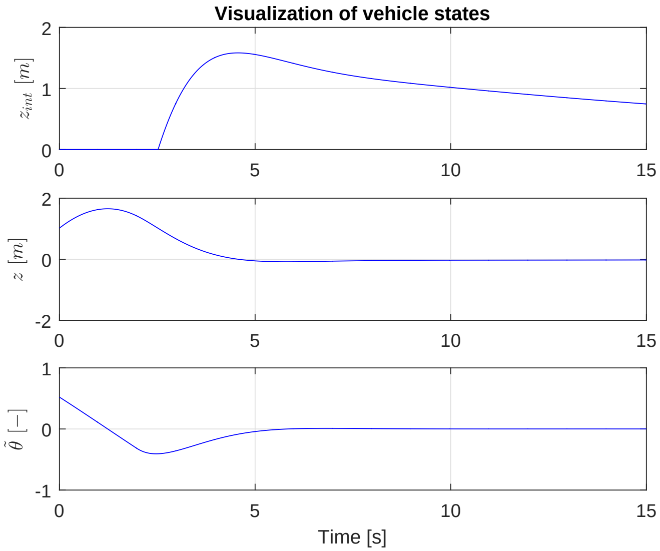


Figure 7.9: Visualization of the vehicle states for forward driving with the LQ-controller in the basic simulation environment.

Advanced Simulation Environment

Once the LQ controller worked in the basic simulation environment test continued in the advanced simulation environment. In Figure 7.10-7.12 the result from a forward driving test with moderate curvature can be seen. Figure 7.10 visualizes the driven path, from it we can see that the path is quite gentle. In Figure 7.10 both the output curvature of the controller, κ, and the feedforward curvature from the path, κ_0, can be seen. When the vehicle is perfectly aligned with the path these will coincide. We can see a significant difference in these the first couple of seconds. This is due to the motion planner not taking the angle of the wheels into consideration, meaning that a reference path can go directly to the right while the wheels are facing left. Hence, give rise to errors in the vehicle states. In contrary to the basic simulation environment the distance from path, z, does not converge to zero anymore. This is due to the vehicle model in the environment is not coinciding with the bicycle model used for the controller design. The mean of the absolute value to the path is 0.0416 m.

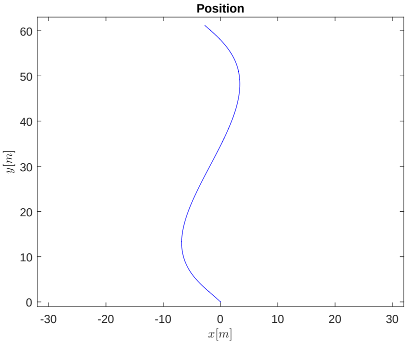


Figure 7.10: *x- and y-position of the driven path during forward driving with the LQ controller in the advanced simulation environment.*

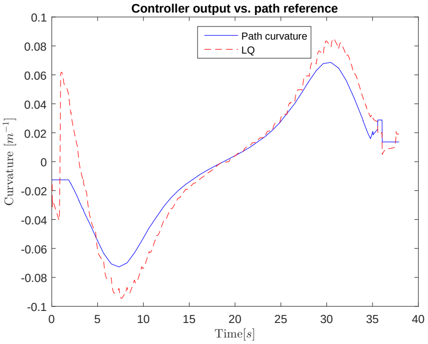


Figure 7.11: *Visualization of the controller output along with the curvature of the reference path.*

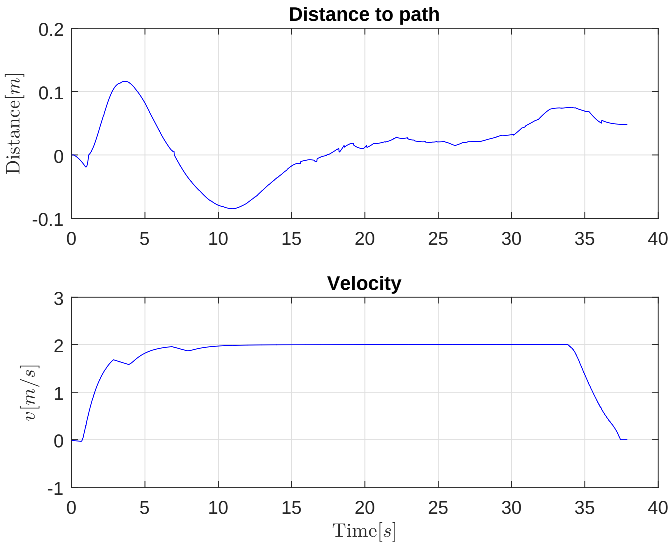


Figure 7.12: *The resulting distance from the path and the velocity of the vehicle. The mean and median of the absolute value are 0.0416 m and 0.0313 m, respectively.*

Heavy Duty Vehicle

Once satisfying results were achieved in the simulation environments, tests were carried out with a real HDV. Several tests were performed using the LQ controller and here we present a test when the vehicle is moving in forward motion and a test when the vehicle is reversing. Information regarding the forward driving can be seen in Figure 7.13-7.15 while the reversing is presented in Figure 7.16-7.18. In Figure 7.13 and Figure 7.16 the vehicle's position throughout the tests is shown, from these plots we can see that both tests are quite similar. We can see in Figure 7.14 that the controller output, κ, when driving forward, is quite aligned with the path reference curvature, κ_0. This means that the vehicle is following the path well, i.e. the states used for the state feedback are close to zero. This can also be seen in Figure 7.15 where the distance from the path, z, is shown where the mean of the absolute value is 0.0176 m. At around 15 s we can see some discontinuities, these probably occur due to the RTK-GPS not being completely consistent. In Figure 7.18 the distance from path while reversing is shown. This figure shows a very sharp oscillating behavior which can not be explained but is probably due to some minor bug in the software which measures the distance to the path. However, the mean of the absolute value is 0.0111 m which is very low.

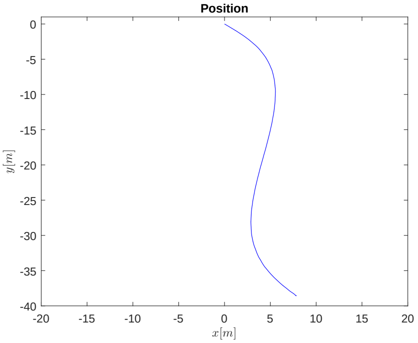


Figure 7.13: *x- and y-position of the driven path during forward driving with the LQ controller in the vehicle.*

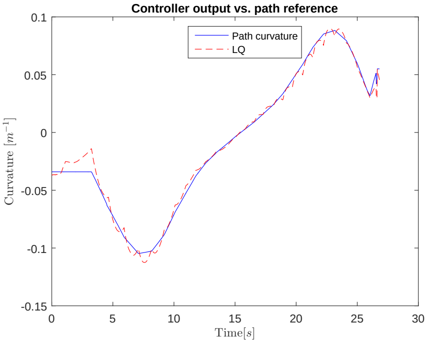


Figure 7.14: *Visualization of the controller output with the curvature of the reference path.*

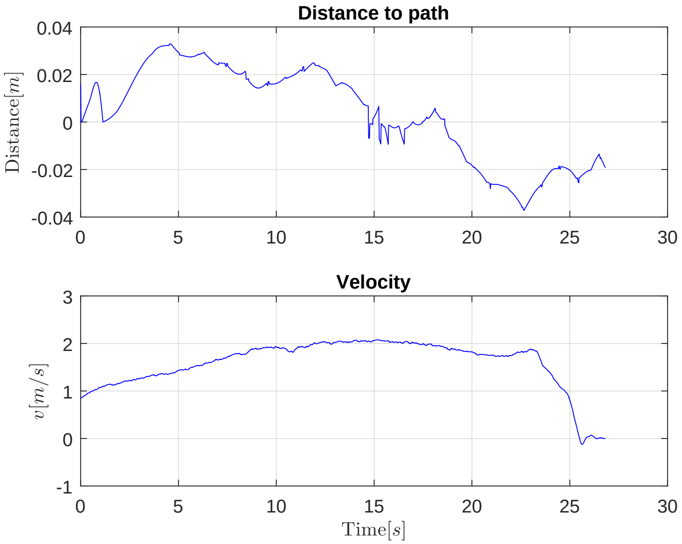
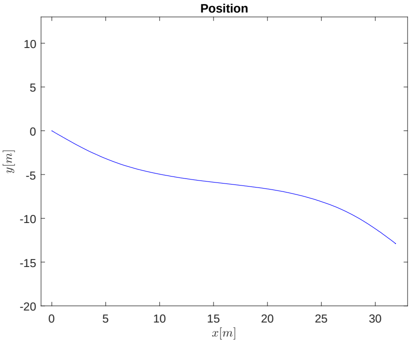


Figure 7.15: *The resulting distance from the path and the velocity of the vehicle. The mean and median of the absolute value are 0.0176 m and 0.0192 m, respectively.*

7.3.3 Model Predictive Control

This section will present results from the control system when using an MPC controller as lateral controller, described in Section 6.4. Results from both simulation environments; basic and advanced, will be presented as well as results



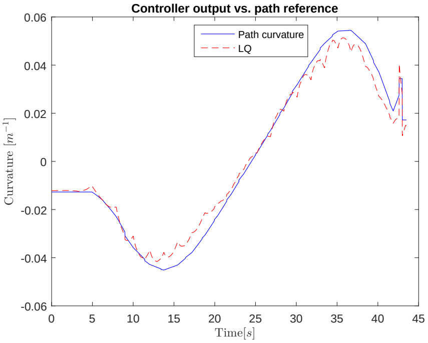


Figure 7.16: *x- and y-position of the driven path during reversing with the LQ controller in the vehicle.*

Figure 7.17: *Visualization of the controller output with the curvature of the reference path when reversing.*

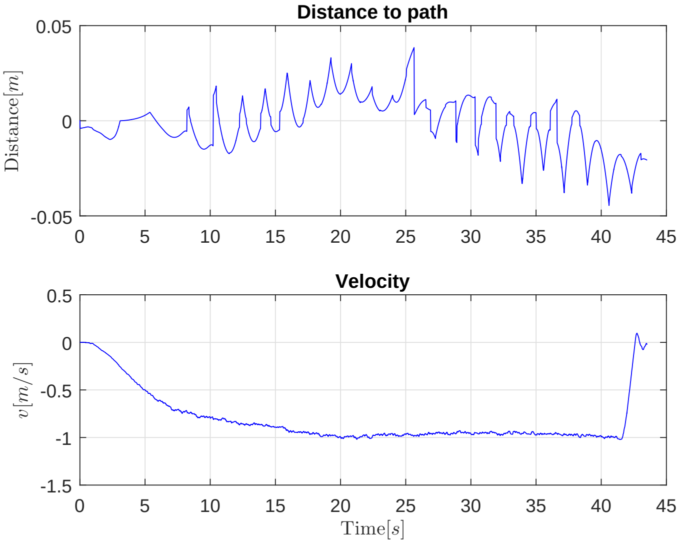


Figure 7.18: *The resulting distance from the path and the velocity of the vehicle when reversing. The mean and median of the absolute value are 0.0111 m and 0.0094 m, respectively.*

from the actual vehicle. Table 7.2 shows the prediction horizon and solver used in this section. *quadprog* is a solver available in MATLAB which it is not possible to generate C-code from, limiting this solver to the basic simulation environment. The solver provided by Scania CV AB can be used to generate C-code and can thus be used in the advanced simulation environment and in the vehicle. The tests for the MPC controller were carried out using the same cost matrix as for the LQ controller in Section 7.3.2,

$$Q = \text{diag}\left(\begin{bmatrix} 0.0001 & 0.1 & 0.1 \end{bmatrix}\right),$$

and the same frequency, 50 Hz. During the prediction both the vehicle velocity, v, and curvature of the path, κ_0, are assumed to be constant. These assumptions are based on the appearance of the path in the basic simulation environment, see Figure 7.19, and the fact that the prediction horizon is very short in the other environments.

Table 7.2: *Prediction horizon and solver used in the MPC controller for the different environments in this chapter.*

Environment	N_p	**Solver**
Basic	200	quadprog
Advanced	4	Scania CV AB
Vehicle	4	Scania CV AB

Basic Simulation Environment

As for the LQ controller, tests were carried out with the MPC in the basic simulation environment for proof of concepts. Using the same test setup as for the LQ controller in Section 7.3.2 with the extension of constraints on z, $\tilde{\theta}$ and Δu measuring 2, $\frac{\pi}{2}$ and 0.003, respectively. The result can be seen in Figure 7.19-7.21.

From the figures we can see that the MPC controller is working as intended, i.e. all the states converge to zero and the constraints are satisfied. In Figure 7.20 the curvature from both the LQ and MPC controller can be seen. The MPC controller starts to increase its curvature before the LQ controller does, with a constant slope, the cause of this is the influence of the constraint in curvature derivative. Which time instance the MPC starts to increase its curvature is most likely dependent on the constraint in maximum curvature output.

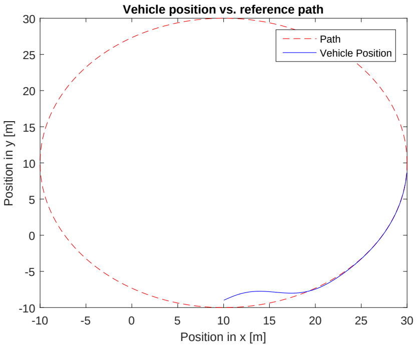


Figure 7.19: *Visualization of the vehicle driven path and the reference path from a basic simulation using the MPC controller.*

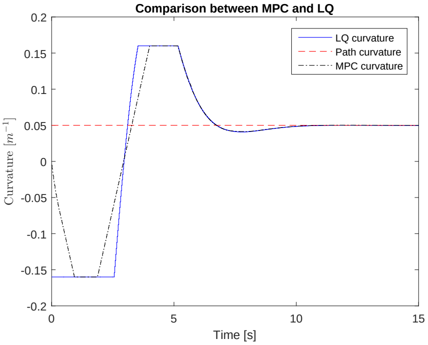


Figure 7.20: *Comparison between the output from the MPC and LQ controller together with the curvature from the path. Note that it is the MPC output that actually steers the vehicle, the output from the LQ controller is only plotted for comparison.*

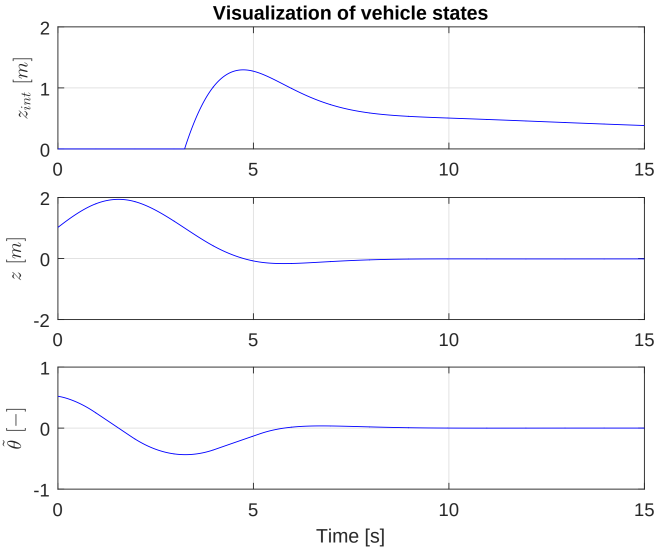


Figure 7.21: *The resulting vehicle states when using the MPC in the basic simulation environment for forward driving.*

Advanced Simulation Environment

Once adequate results were achieved in the basic simulation environment the MPC controller was deployed in the advanced simulation environment. One test was sufficient to verify the implementation. This test was chosen to be performed in backward motion. A reference path with moderate curvature was chosen and the position of the vehicle through the test can be seen in Figure 7.22. The curvature output from the MPC controller, LQ controller and the feedforward curvature from the path, κ_0, can be seen in Figure 7.23. In Figure 7.24 the distance to the path, z, can be seen with the velocity of the vehicle.

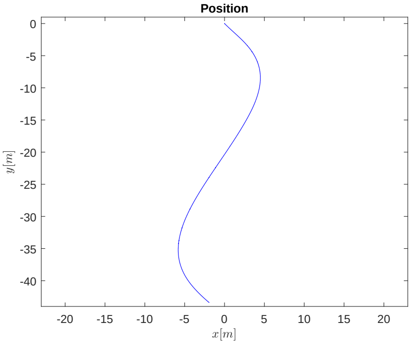


Figure 7.22: *Visualization of the path driven by the vehicle when reversing in the advanced simulation environment.*

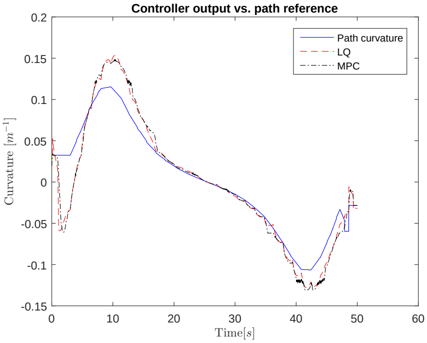


Figure 7.23: *Comparison between the output from the MPC-controller and LQ controller together with the curvature from the path when reversing with moderate curvature. Note that it is the MPC controller output that actually steers the vehicle, the output from the LQ controller is only plotted for comparison.*

Similarly, as for the LQ controller the performance is decreased when changing from the basic simulation environment due to model errors. A keen eye can see slight difference between the controller outputs in Figure 7.23. This is rather unexpected when we are using the same cost function and a prediction horizon of four samples, normally the MPC controller would act as the LQ controller but with a rate-limiter. The reason behind the difference is the change of solver from MATLAB'S *quadprog* function to a solver provided by Scania CV AB, which enabled the necessary possibility to generate C-code, which both the advanced environment and the HDV uses. To get the solver to work in the required frequency the precision had to be lowered. Thus, the performance of the controller will be affected. The prediction horizon had to be lowered to $N_p = 4$ for the same reason. However, the MPC controller still manage to keep the vehicle rather close to the path with a mean value 0.0565 m of the absolute distance from the path.

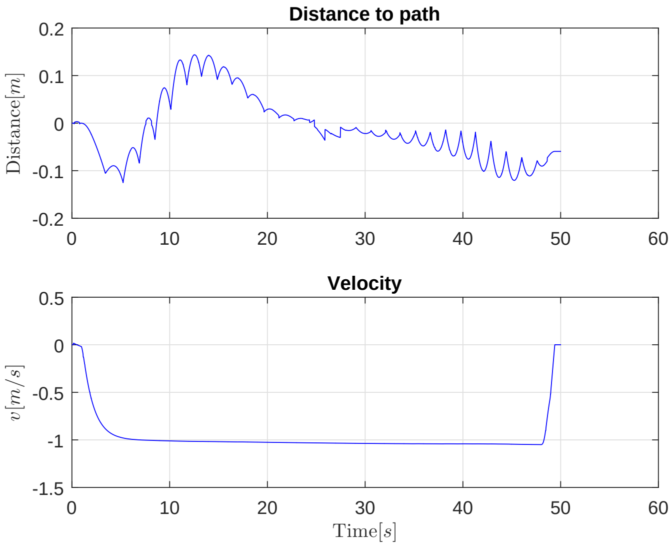


Figure 7.24: *The resulting distance from the path with the velocity of the vehicle. The mean and median of the absolute value are 0.0565 m and 0.0516 m, respectively.*

Heavy-Duty Vehicle

Once satisfying results were achieved in the simulation environments, tests were carried out with a real HDV using the MPC controller. Two tests were performed, one driving forward on a path with high curvature and one reversing on a path with moderate curvature. The result from the forward driving is presented in Figure 7.25-7.27 and the reversing in Figure 7.28-7.30. Figure 7.25 and 7.28 show the position of the vehicle during the two tests. Figure 7.26 and 7.29 show both the MPC controller's and the LQ controller's output along with the curvature of the path. Figure 7.27 and 7.30 visualize the distance to the path along with the vehicle's velocity during the tests.

As for the MPC controller in the advanced simulation environment, Section 7.3.3, the solver is not completely accurate, see the difference in output in-between the MPC controller and the LQ controller in Figure 7.26 and 7.29. However, the resulting absolute distance from the path is still very precise with a mean of 0.0270 m for forward driving and 0.0317 m for reversing.

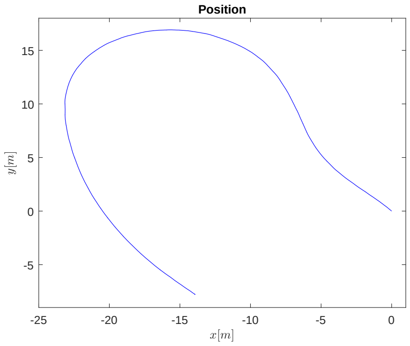


Figure 7.25: *x- and y-position of the driven path during forward driving with the MPC in the vehicle.*

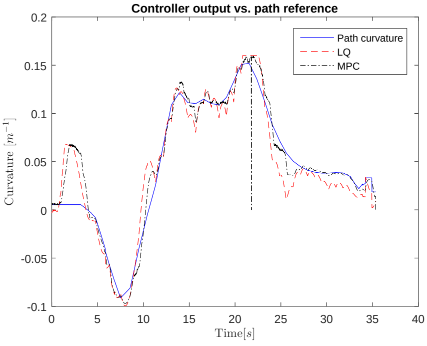


Figure 7.26: *Visualization of both the MPC controller's and LQ controller's output along with the curvature of the reference path when driving forward. Note that it is the MPC controller output that actually steers the vehicle, the output from the LQ controller is only plotted for comparison.*

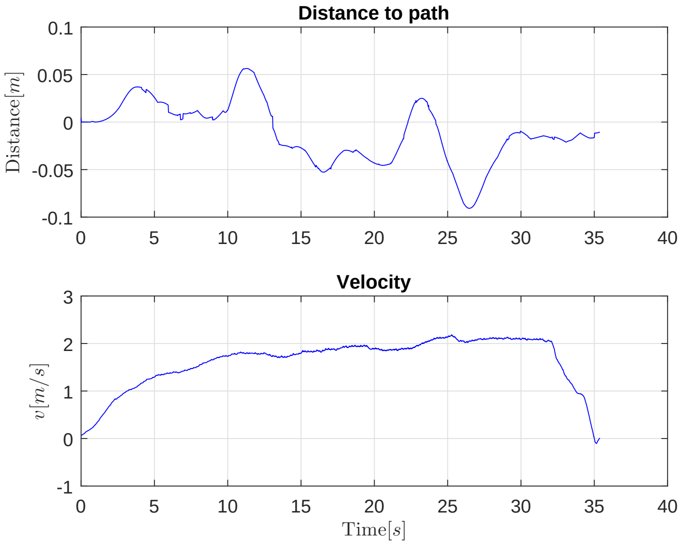


Figure 7.27: *The resulting distance from the path and the velocity of the vehicle when driving forward. The mean and median of the absolute value are 0.0270 m and 0.0228 m, respectively.*

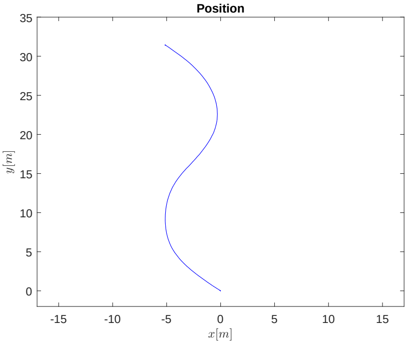


Figure 7.28: *x- and y-position of the driven path when reversing using the MPC controller.*

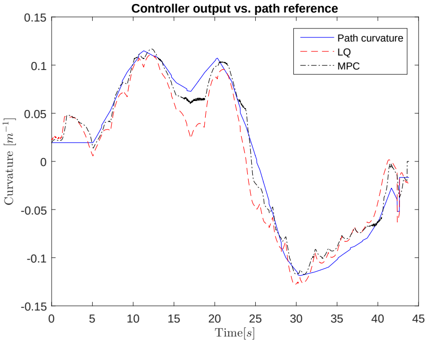


Figure 7.29: *Comparison between the output from the MPC controller and LQ controller together with the curvature from the reference path when reversing. Note that it is the MPC controller output that actually steers the vehicle, the output from the LQ controller is only plotted for comparison.*

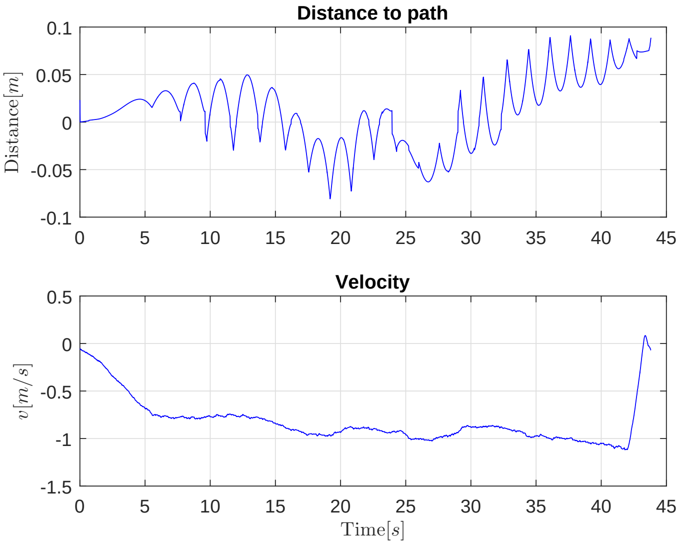


Figure 7.30: *The resulting distance from the path along with the velocity of the vehicle when reversing. The mean and median of the absolute value are 0.0317 m and 0.0272 m, respectively.*

7.4 System Test

Finally, a test of the complete system is presented. Due to limited amount of time available for the complete system test only the MPC controller was chosen to be tested with the whole system. This is motivated by the fact that the MPC controller has shown to have worse performance compared to the LQ controller, i.e., if the MPC controller is sufficient to park the vehicle then the LQ controller will work as well. The system test was performed at a large parking lot with three defined waypoints and three parking spots. A graphical illustration of the test is shown in Figure 7.31-7.38, and the distance from the path, vehicle velocity, controller output and vehicle position are visualized in Figure 7.39-7.41.

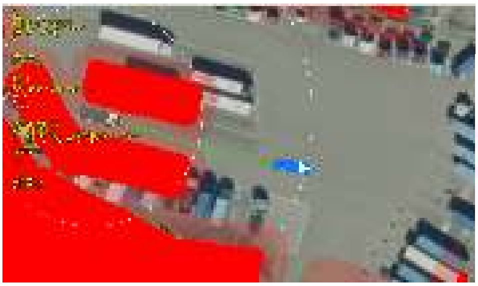


Figure 7.31: *The APS starts off by being clueless about any parking spot states. There is no available parking spot at its current sight, hence its new mission is to travel to the first waypoint.*

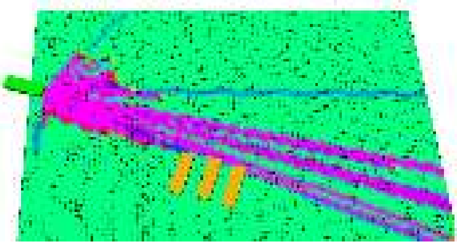


Figure 7.32: *Visualization from the parking spot detection system. The three defined parking spots are currently of unknown state.*

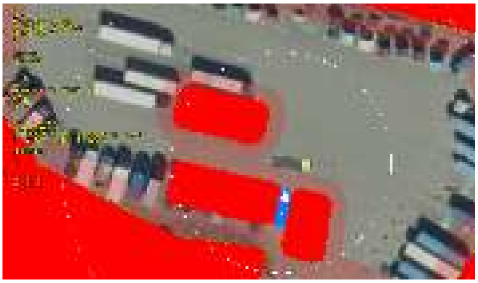


Figure 7.33: *When the vehicle has reached the waypoint it checks if any unoccupied parking spots have been detected. This is the case, hence the new mission will be to park in this available parking spot.*

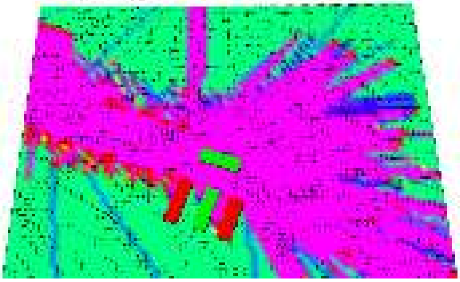


Figure 7.34: *At the first waypoint, one parking spot has been determined as free and is sent to the motion planner.*

In Figure 7.31 the reference path from the initial position to the first waypoint

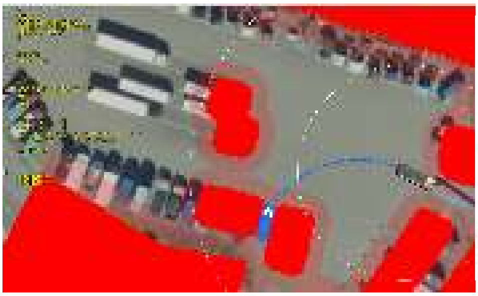


Figure 7.35: *After some minor corrections in the vehicle's position, it starts reversing towards the parking spot.*

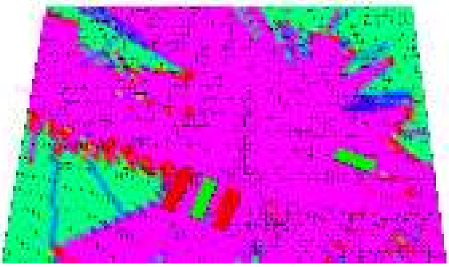


Figure 7.36: *Visualization of the parking spot detection system at the same time instance as in Figure 7.35.*

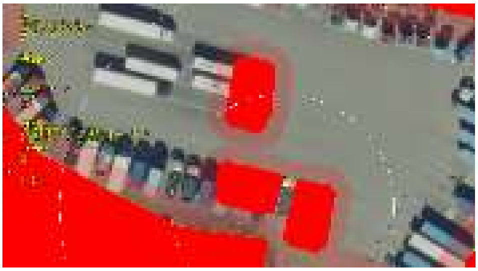


Figure 7.37: *Finally, the APS has managed to park the vehicle.*

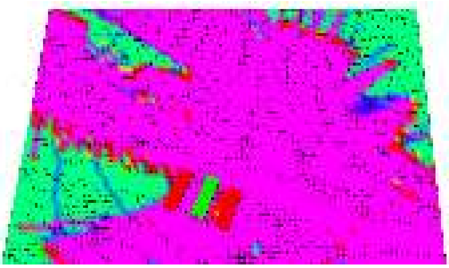


Figure 7.38: *Image from the parking spot detection system when the vehicle has parked successfully.*

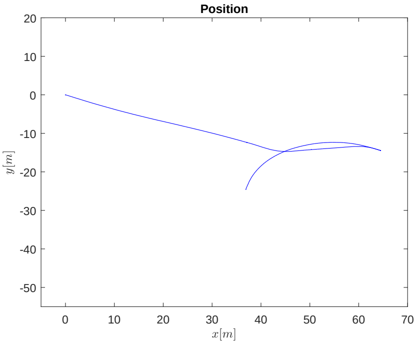


Figure 7.39: *x- and y-position of the driven path during the system test with the MPC controller steering the vehicle.*

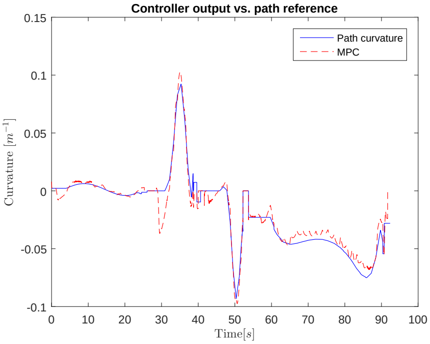


Figure 7.40: *Output from the MPC controller together with the curvature from the path.*

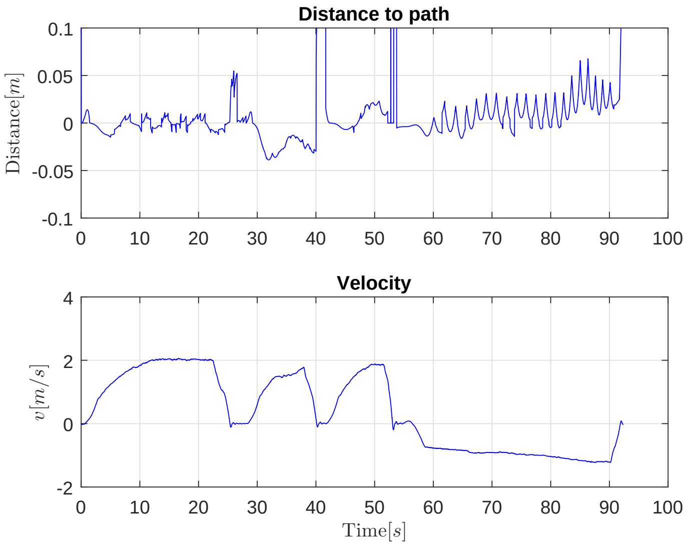


__Figure 7.41:__ The resulting distance from the path with the velocity of the vehicle for the complete system test. The mean and median of the absolute value, when outliers larger than 0.3 m are removed, are 0.0164 m and 0.0085 m, respectively.

is shown. In the corresponding visualization over the OGM (see Figure 7.32), we can see that all three parking spots have an unknown state. Once the vehicle has traveled to the first waypoint (see Figure 7.33), we can see that the one of the three parking spots is detected as available, see Figure 7.34. Hence, the next requested vehicle position will be in that parking spot, see Figure 7.33. After further movement forward we can see that the planner sends a reference path into the parking spot, see Figure 7.35. The MPC controller manage to follow this planned path with enough accuracy and the final position of the vehicle can be seen in Figure 7.37.

In Figure 7.39 the driven path from the initial position to the parked position is presented. In Figure 7.40 we can see that major part of the output from the MPC controller is the feedforward curvature from the path, this is however expected when the vehicle is for the most part very close to the planned path. In Figure 7.41 the vehicle's distance from path and velocity are presented. The graph over the velocity shows that it takes approximately 27 s to reach the first waypoint, depicted in Figure 7.33, after additional 15 s another stop was reached. After 55 s the vehicle starts to reverse towards the parking spot, as can be seen in Figure 7.35. Finally, after around 90 s the parking is completed. In the upper graph of Figure 7.41 several phenomena are seen. The resolution of the RTK-GPS is clearly seen in the first 30 s. The odd shape that occurs when reversing can also be seen in-between 60-100 s. There are also several spikes in the signal, occurring at 27 s, 40 s and 55 s. These happen when a waypoint is reached and can not be explained but is due to some software bug.

The system test resulted in a successful parking using the APS. The mean of the absolute distance from the path was 0.0164 meters and the median as good as 0.0085 meters (when the outliers at 40 and 55 s was removed) which is extremely good. However, the paths given by the planner were quite *nice* which influenced the result for the better.

7.5 Discussion

This section will discuss the performance of the APS based on the results presented in Section 7.

Parking Spot Detection System

The result from the Occupancy Grid Map in Section 7.1 determined the correct state of all defined parking spots. The system does however determine the wrong state in some cases, for example if there is an object occupying only a part of the parking spot and the mean value of the grid cells involved is smaller than the lower threshold in (4.3). This is easily solved by checking if any of the grid cells are larger than some higher threshold and in that case, give the parking spot the state occupied. This has however not been a problem since the parking spots usually are completely empty or occupied.

In Section 3.2, it was mentioned that the motion planner needs an obstacle map for planning the path between waypoints. The OGM produced by the parking spot detection system did not reach the level of the OGM provided by Scania CV AB, hence the motion planner does not use the OGM developed in this thesis.

Simplified Model

The simplification to set $\kappa_0 = 0$ in (5.15) seemed rational at the time, though during development the cost matrix Q changed due to tuning without validating that this parametrization still was reasonable. The final cost matrix in the LQ control design was chosen as:

$$Q = \begin{bmatrix} 0.001 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}. \tag{7.2}$$

In Figure 7.42 the feedback gain matrix is visualized parameterized in speed for -1.0 m/s, 0.5 m/s and 2.0 m/s and in reference curvature between [-0.16 0.16].

Using the cost matrix in (7.2) the variations in the feedback for gain scheduling with respect to v and κ_0 are shown in Table 7.3. In the first column the nominal curvature $\kappa_0 = 0\ \mathrm{m}^{-1}$ is fixed and the variations in the feedback gains for the velocities $v = 0.1$ and 2.0 m/s are compared. In the second column the velocity is fixed to $v = 1$ m/s and the variations in the feedback gain for $\kappa_0 = 0$ and $\pm\kappa_0 = 0.16\ \mathrm{m}^{-1}$ are presented. From this table we can see that variations in the nominal curvature affects K_z with up to 8 %. Based on this it would therefore be motivated to parametrize the feedback gain in the nominal curvature and not in the speed. However, this was not done due to lack of time.

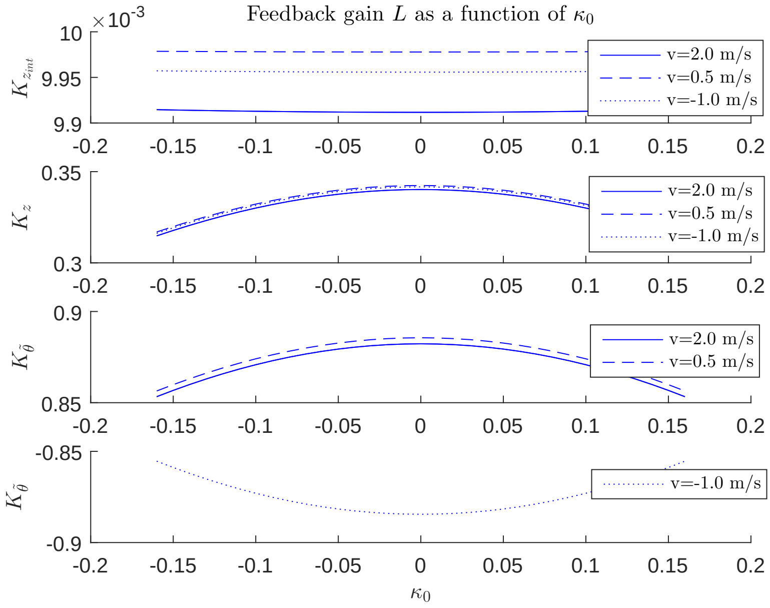


__Figure 7.42:__ Visualization of the feedback gain matrix for the speed - 1.0 m/s, 0.5 m/s and 2.0 m/s as a function of the reference curvature κ_0.

7.5.1 Lateral Controller

From the four tests (counting both controllers) performed in the vehicle and presented in Section 7.3 the summary can be seen in Table 7.4. Even if it is hard

Table 7.3: *Gain scheduling variations. In order to simplify the model, it was investigated if it was reasonable to assume constant curvature (κ_0 = 0 m^{-1}) or constant velocity (v = ±1 m/s). The difference in feedback gain is between v = 0.1 m/s and v = 2.0 m/s for a constant κ_0 = 0 m^{-1} is listed as* ***Difference in speed [%].*** ***Difference in curvature*** *[%] lists the feedback gain difference between κ_0 = 0 m^{-1} and κ_0 = ±0.16 m^{-1} for the constant velocity v = 1 m/s.*

Gain	Difference in speed [%]	Difference in curvature [%]
$K_{z_{int}}$	0.85	0.03
K_z	0.82	8.07
$K_{\tilde{\theta}}$	0.48	3.40

to compare individual tests, due to slight different initial states and reference paths, we can draw the conclusion that the LQ controller track the reference path better compared to the MPC controller. This is however expected due to the constraint in Δu. For both controllers the mean of z is around 0.01-0.03 m which is more than sufficient for parking maneuvers.

Table 7.4: *Result from the LQ controller and MPC controller in the vehicle. The mean and median of the absolute value of the distance to path for the rear axle is shown.*

Situation	Mean [m]	Median [m]
LQ forward driving	0.0176	0.0192
LQ reversing	0.0111	0.0094
MPC forward driving	0.0270	0.0228
MPC reversing	0.0317	0.0272

As presented, z is satisfactory low, but is that enough to perform a parking maneuver? In Figure 7.43 we can see the distance from both the rear and front axle when driving the path depicted in Figure 7.25. It is clear from the figure that the front axle is generally further away than the rear axle. This is of course expected when it is only the distance from the rear axle that is controlled. However, both distances are equally important to prevent the vehicle from colliding with other obstacles. However, this issue is typically handled in the planning level where the motion planner does not plan paths that are too close to obstacles.

Due to the necessity to let the prediction horizon be very short the constraints did not have the intended effect. The first derivative of the curvature is still constrained but the same result would have been achieved by applying a *Rate Lim-*

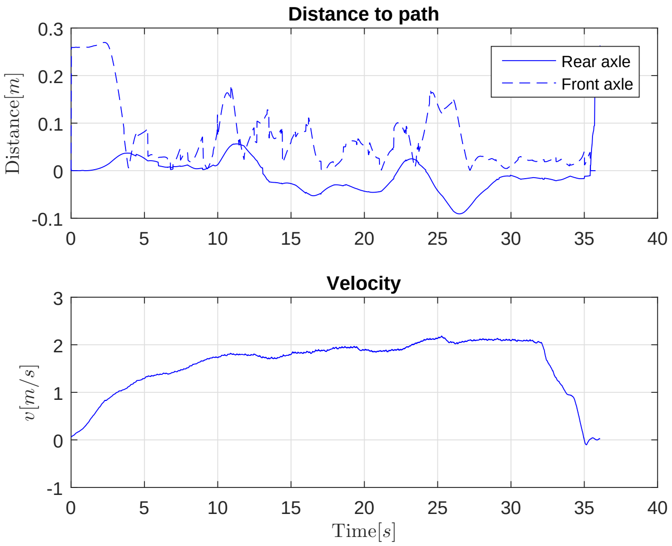


Figure 7.43: *The figure presents the distance from both rear and front axle to the path as well as the velocity.*

iter to the LQ controller. However, there are several possibilities to increase the prediction horizon. For instance, Δu could be moved from being a constraint to be included in the cost function. That would enable the possibility to use *Move Blocking* [7] for decreased computational complexity. It would also be wise to use a more optimized solver than the one which was used.

8

Conclusions and Future Work

This thesis has developed an automatic parking system with a path following controller for a heavy-duty vehicle and an automatically generated OGM. This chapter will present the conclusions from the results in this thesis and propose how to proceed with future work to continue developing the automatic parking system.

8.1 Conclusions

The purpose with this thesis was to develop an APS for an HDV. One of the most critical decisions that has to be adressed is whether a parking spot is occupied or available. We have shown that by creating an OGM based on measurements from GPS and LIDAR sensors the OGM can be used to determine the state of the parking spot. The APS commands the vehicle to travel between pre-stated waypoints until an unoccupied parking spot has been detected. The motion planner, provided by Scania CV AB, then computes a path from the vehicle's current position to the desired position at the parking spot. Both an LQ controller and an MPC controller have been developed for path following control. They have been proven to, according to Table 7.4, be able to keep the vehicle sufficiently close to the desired path, with an absolute mean lateral control error of about $1-3$ cm through the different tests. Hence, we can come to the conclusion that we have designed two different path following controllers that both have a sufficient tracking accuracy to enable automatic parking. The LQ controller can be seen as a PID controller with feedforward due to $\dot{z} = v \sin \tilde{\theta} \approx$

$v\tilde{\theta}$, hence a PID controller would have been sufficient as path following controller. Finally, the complete APS is tested and the result is that the APS succeeds to park the HDV, completely autonomously.

8.2 Future Work

The future work proposed in this section aims to remove the exclusions introduced in Section 1.3 to deliver a more robust and general parking system that works in unknown parking lots.

We have experienced issues with reliability of the RTK-GPS and would like to remove the dependence of the RTK-GPS by introducing some sort of *Simultaneous Localization and Mapping* method (SLAM) and work in local coordinates as proposed in [9]. A camera could also be integrated for locating free parking spots by identifying the lane markers on the ground [26]. Another improvement would be to integrate additional LIDAR and RADAR sensors to enhance the performance of the OGM.

The current way of exploring the parking lot with hard coded waypoints defined in global coordinates should be changed to some sort of active sensing, where the parking detection system figures out where to go next to explore the parking lot based on the OGM. The truck should investigate areas with a lot of unknown states in the OGM to expand the map.

The OGM would have needed some more work if it were to be used by the motion planner and the parking detection system would need some more oversight for a more robust performance that can handle special cases.

As mentioned in Section 7.5.1 the solver used for the MPC was not powerful enough to have a long prediction horizon. Hence, one possibility could be to look into other, more optimized, solvers to see if the prediction horizon can be increased. If this is not possible, then a suggestion would be to remove the constraint on Δu from (6.25) and add this Δu to the cost function in (6.14) instead. This would enable the opportunity to use *Move-blocking*[7] for decreased computational complexity.

It appeared in Section 7.5.1 that even if the rear axle was close to the path it was not always the same for the front axle. It would be interesting to explore the option to also include that position in the control problem. However, one might argue for that it is the motion planner's responsibility to plan paths which are easily followed by only taking the rear axle into consideration.

Bibliography

[1] Path following of an omni-directional four-wheeled mobile robot. *Artificial Intelligence and Robotics (IRANOPEN), 2016*, page 36, 2016. ISSN 978-1-5090-2169-7. Cited on page 4.

[2] Autonomous driving, July 2017. URL `http://www.volvocars.com/intl/about/our-innovation-brands/intellisafe/autonomous-driving`. Cited on page 4.

[3] A. Alessandretti, A. P. Aguiar, and C. N. Jones. Trajectory-tracking and path-following controllers for constrained underactuated vehicles using model predictive control. In *2013 European Control Conference (ECC)*, pages 1371–1376, July 2013. Cited on page 4.

[4] Martin Andersen, Joachim Dahl, Zhang Liu, and Lieven Vandenberghe. Interior-point methods for large-scale cone programming. `http://www.seas.ucla.edu/~vandenbe/publications/mlbook.pdf`. Accessed: 2017-05-25. Cited on page 11.

[5] Daniel Axehill. *Integer Quadratic Programming for Control and Communication*. PhD thesis, Linköping University, Automatic Control, The Institute of Technology, 2008. Cited on page 12.

[6] Daniel Axehill. Controlling the level of sparsity in MPC. 76:1–7, 2015. doi: 10.1016/j.sysconle.2014.12.002. Cited on page 12.

[7] R. Cagienard, P. Grieder, E. C. Kerrigan, and M. Morari. Move blocking strategies in receding horizon control. In *2004 43rd IEEE Conference on Decision and Control (CDC)*, volume 2, pages 2023–2028, Dec 2004. doi: 10.1109/CDC.2004.1430345. Cited on pages 70 and 72.

[8] M. Cirillo, T. Uras, and S. Koenig. A lattice-based approach to multi-robot motion planning for non-holonomic vehicles. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 232–239, Sept 2014. doi: 10.1109/IROS.2014.6942566. Cited on pages 3 and 18.

[9] Andrew J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings IEEE International Conference on Computer Vision*, pages 1403–1410, 2003. Cited on page 72.

[10] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989. ISSN 0018-9162. doi: 10.1109/2.30720. Cited on pages 3 and 22.

[11] Igor E.Paromtchik and Christian Laugier. Autonomous parallel parking of a nonholonomic vehicle. `http://emotion.inrialpes.fr/~paromt/infos/papers/paromtchik:laugier:iv:1996.pdf`, 1996. Cited on page 3.

[12] Paolo Falcone, H. Eric Tseng, Francesco Borrelli, Jahan Asgari, and Davor Hrovat. Mpc-based yaw and lateral stabilisation via active front steering and braking. *Vehicle System Dynamics*, 46(sup1):611–628, 2008. doi: 10.1080/00423110802018297. Cited on page 4.

[13] Torkel Glad and Lennart Ljung. *Reglerteori : flervariabla och olinjära metoder.* Lund : Studentlitteratur, 2003, 2003. ISBN 9144030037. Cited on pages 9 and 37.

[14] Torkel Glad and Lennart Ljung. *Reglerteknik : grundläggande teori.* Lund : Studentlitteratur, 2006, 2014. ISBN 978-71-44-02275-8. Cited on page 5.

[15] M. Haraguchi and H.Y. Hu. Using a new discretization approach to design a delayed {LQG} controller. *Journal of Sound and Vibration*, 314(3–5):558 – 570, 2008. ISSN 0022-460X. doi: http://doi.org/10.1016/j.jsv.2008.01.047. Cited on page 7.

[16] Erik Hellström, Maria Ivarsson, Jan Åslund, and Lars Nielsen. Look-ahead control for heavy trucks to minimize trip time and fuel consumption. *Control Engineering Practice*, 17(2):245 – 254, 2009. ISSN 0967-0661. doi: http://dx.doi.org/10.1016/j.conengprac.2008.07.005. Cited on page 2.

[17] CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST. and R.C. Coulter. *Implementation of the Pure Pursuit Path Tracking Algorithm.* Carnegie Mellon University, the Robotics Institute, 1992. URL `https://books.google.se/books?id=tkAkpwAACAAJ`. Cited on page 3.

[18] S. M. LaValle. *Planning Algorithms.* Cambridge, U.K.: Cambridge University Press, 2006. Cited on page 3.

[19] Johan Löfberg. *Linear Model Predictive Control : Stability and Robustness.* PhD thesis, Linköping University, Automatic Control, The Institute of Technology, 2001. Cited on pages 9 and 13.

[20] Oskar Ljungqvist, Daniel Axehill, and Anders Helmersson. Path following control for a reversing general 2-trailer system. In *2016 IEEE 55TH CONFERENCE ON DECISION AND CONTROL (CDC) :*, IEEE Conference on

Decision and Control, pages 2455–2461. IEEE, 2016. ISBN 978-1-5090-1837-6. doi: 10.1109/CDC.2016.7798630. Cited on page 4.

[21] W. J. R. Louwerse and S. P. Hoogendoorn. Adas safety impacts on rural and urban highways. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 887–890, June 2004. doi: 10.1109/IVS.2004.1336502. Cited on page 2.

[22] Alessandro De Luca, Giuseppe Oriolo, and Claude Samson. Feedback control of a nonholonomic car-like robot. 2009. Cited on pages 27, 28, and 29.

[23] Joachim Lundh. Model predictive control for active magnetic bearings. Master's thesis, Linköping University, Automatic Control, The Institute of Technology, 2012. Cited on page 11.

[24] M. Pivtoraiko and A. Kelly. Kinodynamic motion planning with state lattice motion primitives. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2172–2179, Sept 2011. doi: 10.1109/IROS.2011.6094900. Cited on page 3.

[25] M. Salazar, A. Alessandretti, A. P. Aguiar, and C. N. Jones. An energy efficient trajectory tracking controller for car-like vehicles using model predictive control. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 3675–3680, Dec 2015. doi: 10.1109/CDC.2015.7402789. Cited on page 4.

[26] M. Schreiber, F. Poggenhans, and C. Stiller. Detecting symbols on road surface for mapping and localization using ocr. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 597–602, Oct 2014. doi: 10.1109/ITSC.2014.6957755. Cited on page 72.

[27] T. Sugimachi, T. Fukao, J. Yoshida, Y. Hirata, Y. Suzuki, and K. Aoki. Autonomous driving based on lq path following control and platooning with front and rear information. In *17th ITS World Congress*, number 17th ITS World Congress, Department of Mechanical Engineering, Kobe University, 2010. Cited on page 4.

[28] Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15(2):111–127, Sep 2003. ISSN 1573-7527. doi: 10.1023/A:1025584807625. Cited on page 3.

[29] Sebastian Thrun. Exploring artificial intelligence in the new millennium. chapter Robotic Mapping: A Survey, pages 1–35. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. ISBN 1-55860-811-7. Cited on page 3.

[30] T. Wu and J. Y. Hung. Path following for a tractor-trailer system using model predictive control. In *SoutheastCon 2017*, pages 1–5, March 2017. doi: 10.1109/SECON.2017.7925337. Cited on page 4.