AALBORG UNIVERSITY
STUDENT REPORT

Autonomous Navigation in Urban Environments

Student Report
Master's Thesis
Group 934
Control & Automation
2018

Control & Automation
Fredrik Bajers Vej 7C
DK-9220 Aalborg Ø

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Autonomous Navigation in Urban Environments

Theme:
Master's Thesis

Project Period:
Autumn 2017 - Spring 2018

Project Group: 934

Page Numbers: 144

Date of Completion:
7^{th} of June 2018

Supervisor(s):
Jan Dimon Bendtsen
Jesper Abildgaard Larsen

Participant(s):

Joan Calvet Molinas

Himal Kooverjee

David Romanos

Abstract:

A future transportation service for Aalborg University (AAU) is imagined as Autonomous Mobility on Demand (AMoD). This thesis intends to build a solid foundation to such a service by addressing the basics of autonomous driving. The problem is derived specifically for the available platform, which is a golf cart. A first-principles model of a cart is derived for simulation purposes. Furthermore, a simplified model is derived to ease the controller design. To bind this model with the actual platform, the models for the actuators are derived. Nonlinear controllers are designed to fulfil the path-following problem and are tested in simulation. The actuator models are used to extend these controllers to implement them in the actual platform. As the controllers needs feedback, the measurements provided by the equipped sensors are fused together using an Extended Kalman Filter (EKF). The combination of the controllers and the EKF is tested in the platform. An exhaustive review of the state-of-the-art path-planners is performed. Based on this, an optimal Rapidly-exploring Random Trees algorithm (RRT*) with kinodynamic constraints is implemented and tested in simulation. The environment for it is constructed as an occupancy-map. This is performed with two LIDARs using the Cartographer^{TM} library for Robot Operating System (ROS).

Preface

The focus of this Master's Thesis is to lay the foundation for a future project for Aalborg University (AAU), providing an Autonomous Mobile on Demand (AMoD) service. This foundation consists of designing a solution to autonomously navigate a golf cart between two points in an urban environment . In an engineering sense, this is to design a control system for an autonomous vehicle.

This thesis has been written to fulfil the completion of a Master's in Control and Automation at AAU, ending spring 2018. It has been supervised by Jan Dimon Bendtsen and Jesper Abildgaard Larsen, both, associate professors at the Department of Electronic Systems at AAU.

The authors would like to acknowledge Simon Jensen, Jesper Dejgaard Pedersen and Frank Høgh Rasmussen for their constant assistance in instrumenting and making modifications to the golf cart. Furthermore, the authors acknowledge Karl Damkjær Hansen for sharing his knowledge on a wide range of topics explored through this thesis. Finally, John-Josef Leth is acknowledged for his invaluable assistance with the controller design.

The reader is expected to have a knowledge within physics and mathematics domain, as well as in modelling and control theory.

Reading Instructions

- The report is divided in three parts. Part I deals with the analysis of the system, which introduces the AMod problem and scopes it down into two distinct problems, namely autonomous driving and navigation. Part II includes the modelling, control structure and sensor fusion designed to solve the problem of autonomous driving. Part III addresses the problem of navigation with its focus on path-planning. In Part IV, the conclusion and discussion of the project are presented.
- The report also includes appendixes that contain supplementary information.
- The bibliography is written using ISO 690, noted as [x], and it is included at the end of the report, prior to the appendices.

Contents

Part I

Pre-analysis

1 Introduction

In the past years, the evolution of the impact produced by vehicles in society has been widely studied. It is estimated that there is one vehicle per seven persons [1] and also that more than half of the population inhabiting the Earth live in urban areas [2]. This increment of population in urban areas together with the increasing amount of vehicles in use have brought a challenge for the future of these environments. The consequences of the human and vehicle population rapid growth can be expected to intensify already existing urban problems such as road congestion, parking availability and pollution [3].

A relatively new suggestion to find a solution to such problems is the Mobility-on-Demand (MoD) service. Explained shortly, MoD services aim to decrease the effect of the vehicle-related problems present in urban environments. Like public transportation, MoD services like taxis or vehicle-sharing, intend to reduce the amount of privately owned vehicles in these environments. If the amount of vehicles is decreased, congestion, parking and pollution problems will decrease. Unlike taxi and vehicle-sharing services, some MoD transportation systems are meant to address the so called "first and last mile" problem [1], meaning transport between the transportation hub to the final destination. By doing this, these MoD services offering door-to-door service want to complement and encourage the use of public transportation [1]. An example of urban problems where a door-to-door service adds to the problem can be seen in Figure 1.1.

Figure 1.1: In the figure, an example of a traffic jam from Times Square is shown. It can be seen that there are multiple taxis in the figure, showing that this kind of door-to-door service does not complement public transport and increases congestion [4].

However, the fundamental difference between MoD solutions and existing transportation services is the demand-responsiveness of the system. The biggest disadvantage of public transportation

against private transportation is the scheduled behaviour of the service. Adapting the transportation to the user's needs and not the opposite way should be prioritised. Furthermore, not having available vehicles is a common situation even with door-to-door services. On-demand systems operate based on the demand of the service, and thus, do not have a scheduled system. Not having vehicles running without being used promises to be also a more sustainable solution.

In spite of the mentioned advantages of MoD systems, some problems with it can arise. The biggest concern of MoD services is having an unbalanced system, meaning that there is a problem with the availability of the service. This would bring back the problem of other mentioned door-to-door services. Therefore, one of the biggest challenges for MoD is guaranteeing the balance of the vehicles, ensuring minimal waiting time for the users. A potential explored solution for availability problems is the usage of vehicles with self-driving capabilities.
The design of self-driving vehicles is a broadly treated topic within the field of study of control. Automating vehicles has long been researched and many solutions to the problem are in existence. These have shown that it is possible to automate a vehicle such that it is self-driven to a desired location. Nevertheless, there is not a single established solution, and challenges within the field of study still exist.

In conclusion, merging MoD with autonomous driving into Autonomous Mobility-on-Demand systems (AMoD) could redistribute the service vehicles to satisfy the demand. By interconnecting of the autonomous vehicles, a better service could be achieved even in terms of the time of usage of the service, routing for example through less-congested roads, and therefore, reducing the ecologic impact [2]. Furthermore, having autonomous vehicles introduces the MoD service to people unable or unwilling to drive. The automation of MoD services would increase productivity, accessibility, road efficiency and improve the environmental impact [1].

1.1 AMoD service in Aalborg University

This thesis is born from imagining the future of Aalborg University (AAU). Looking into the capabilities of transportation services, the idea of introducing an AMoD service at the Aalborg campus of AAU is highly appealing.

Regarding the implementation of AMoD services within closed environments, some inspiring and ground-braking work has been developed by the Singapore-MIT Alliance for Research and Technology (SMART) within the Future Urban Mobility (FM) research department. SMART-FM was born to develop a new paradigm for the planning, design and operation of future urban mobility systems [5].
The idea of implementing an AMoD service around campus, mirrors the achievements of the SMART autonomous vehicle collaborative project [6], and specifically the achievements of the driverless golf carts running on the National University of Singapore (NUS) campus [1].

The objective of an AMoD service in a university campus is straightforward: supply autonomous vehicles able to transport people around campus based on their demand. Users of such service should be able to order a vehicle to drive themselves from their location to a desired end-location in campus.
In addition, implementing a service of vehicles fully equipped is expected to require a high investment. Not just the cost of the vehicles themselves, but the hardware to be mounted is expensive. It is decided that the overall creation of the service must try at all times to reduce the cost. To address the price challenge, inspiration has been found on the development of driverless

golf carts from [1]. For this thesis, a golf cart with the necessary hardware including actuator and sensor systems to carry on with the project is provided, based on the ones developed by the SMART autonomous vehicle collaborative project. An exhaustive description of the used platform is presented in Chapter 2.

It is clear that the development of such ambitious service is not implementable from a single project. Bringing such a challenging idea to life requires work from a wide range of areas. For this, the pursued AMoD service is thought as the result of an addition of different projects carried out throughout the years that are built upon each other.

1.2 Problem formulation

Since no work has been carried out to implement an AMoD service in AAU, it is deemed a reasonable first step to have a well-functioning fleet of autonomous vehicles. The aim of this thesis is to give the opening shot to their design, and more specifically, to address the basic concept of autonomous driving: bringing a vehicle from one point to another. Furthermore, this problem should be undertaken by using the provided platform, without adding new hardware that would increase the cost.

The problem of bringing the golf cart from the initial location to the destination can be divided into subproblems presented in detail in the sections of this thesis. The overview of the presented solution is given in Chapter 3. First, a model of the cart is needed such that its behaviour can be simulated accurately and the controller of the cart can be designed. This is explained in Chapter 4 and Chapter 5 in detail. Afterwards, the control of the cart making it capable of moving autonomously needs to be derived. The control design is presented in Chapter 6. In order to drive autonomously, the controller needs to be aware of the vehicle's behaviour, and therefore, sensor fusion is needed. The sensor fusion implemented is shown in Chapter 7. Last, the navigation problem of finding how to go from one point to another needs to be studied. This is addressed in its entirety in Chapter 8.

2 Platform description

The vehicle provided for this project is the Yamaha Drive2-E-DC [7], which has been used as a base platform to build an autonomous vehicle by incorporating necessary actuation, sensing and processing systems. One of the considerations when these subsystems have been chosen is that the overall cost of the vehicle must be kept low. Thus, expensive sensors such as a Velodyne LIDAR, currently used in other autonomous driving research [8], are not used. The vehicle with all the connected subsystems is shown in Figure 2.1.

Figure 2.1: Overview of the components that have been installed in the vehicle.

The vehicle is a complex system comprised of many subsystems. It is decomposed into three main subsystems: the processing system, actuation system and sensor system. The sensor system is responsible for sensing the environment around the vehicle. This information is given to the processing system which has algorithms that analyse the data and, based on the analysis, sends commands to the actuator system to manipulate the vehicle's movement.

In this chapter, the components in each subsystem and their functionality are discussed. Further technical information about these is given in Appendix A.

2.1 Processing system

The processing system is comprised of three main components providing different functionality to the system: a Low Level Computer (LLC), a High Level Computer (HLC) and a microcontoller. All the communication between these components is via Ethernet through a switch.

Figure 2.2 gives an overview of the processing system and depicts the interconnectivity between the components.

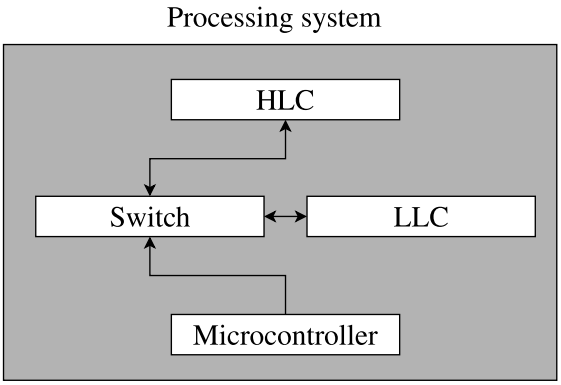


Figure 2.2: Diagram showing the interconnectivity between the components of the processing system.

Low Level Computer

The LLC's main responsibility is to interface to the actuation system of the vehicle. The LLC is able to control the entire actuation system and also to receive data from some of the sensors attached.

It includes many safety factors, including logic for an emergency stop button to mitigate risk related to autonomous driving. Furthermore, each actuation subsystem has an enable button. These buttons are available to the user via the touch screen display depicted in Figure 2.1.

High Level Computer

The HLC is a general purpose computer where all algorithms and work developed through the project is implemented, in Robot Operating System (ROS). The HLC communicates with the LLC and the microcontroller.

Microcontroller

The purpose of the microcontroller is to expand the possibility of adding more sensors to the vehicle. Using this instead of connecting experimental sensors to the LLC ensures that the safety protocols implemented in the vehicle are not compromised.

2.2 Actuation system

The actuation system is divided in three subsystems: the steering system, the braking system and the drive-train system. In this section, these three subsystems and their functionalities are discussed.

In Figure 2.3, an overview of the actuation system can be seen. The interconnectivity between itself and the processing system is also shown. As mentioned in Section 2.1, all the communication for the actuation system is carried out through the LLC.

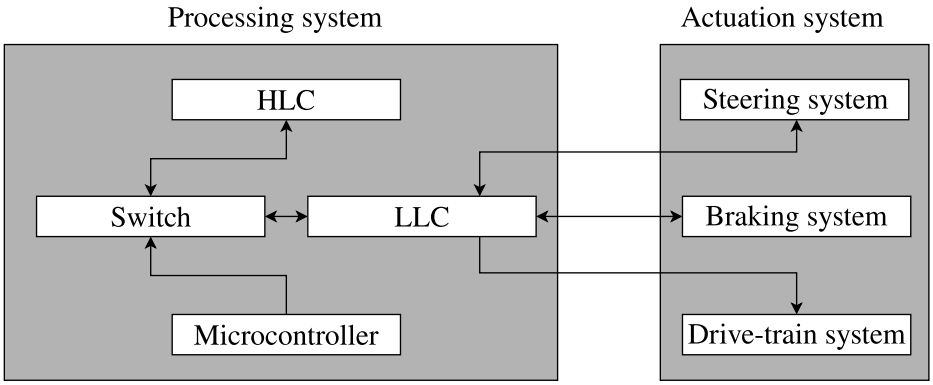


Figure 2.3: Diagram showing the interconnectivity between the components of the processing system and actuator system.

Steering system

The steering system is able to control the position of the steering wheel. It consists of an actuator with an encoder. A proportional controller with negative feedback from the encoder is implemented in the LLC. To actuate this subsystem, the LLC receives setpoint references from the HLC and the proportional controller is able to control the steering wheel position using this reference.

Braking system

The braking system is similar to that of the steering system consisting also of an actuator and an encoder. The purpose of the braking system is to control the position of the vehicle's braking pedal. The braking actuator is also controlled by a proportional controller implemented on the LLC. Similarly, the HLC sends setpoint references to the LLC to control the position of the pedal. An important feature of it is that the driver can manually override the brake in a potential emergency situation.

Drive-train system

The drive-train system is actuated by overriding the throttle pedal with an analogue voltage signal as an angular velocity reference to the vehicle's proprietary motor controller. To drive autonomously, the LLC shall receive an analogue reference from the HLC and send this reference to the vehicle's motor controller.

2.3 Sensor system

The sensor system is comprised of all the relevant sensors on the vehicle. Each sensor provides information, some relating to the movement and position of the vehicle, while others provide information regarding the vehicle's surroundings. In this section, the functionality and purpose of each of the sensors is described.

In Figure 2.4, the platform diagram of the full vehicle is shown. It also shows how each of the sensors connects to the processing system.

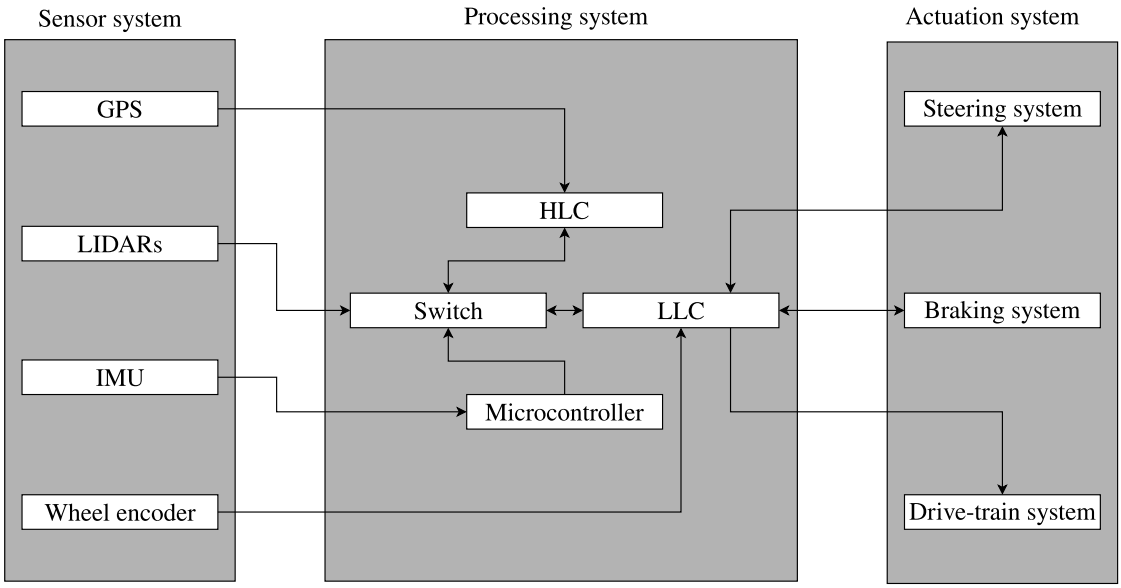


Figure 2.4: Diagram showing the full platform available and the interconnectivity between the sensor system and processing system.

Front facing LIDARs

There are two front facing 2D LIDARs present on the system. One of them is mounted at the front of the vehicle in place of its bumper horizontally and the other is mounted on the roof of the vehicle in a push-broom configuration.

Both the LIDARs have Ethernet connection and communicate directly with the HLC. The purpose of the LIDARs is to be able to sense the environment around the vehicle.

GPS

The GPS installed on the vehicle is a single receiver. The main purpose of including GPS in the vehicle is to have absolute position of the vehicle. It interfaces directly to the HLC to receive positioning information.

Due to the size of the vehicle, centimetre precise GPS is not needed. Hence, a single receiver GPS is chosen opposed to more costly options such as Differential GPS or Real-Time Kinematic GPS, which may provide more precise and accurate measurements. Due to using a single receiver GPS, its the position in the vehicle is not particularly important and it is mounted towards the rear of the vehicle.

IMU

The IMU on the vehicle, connecting to the microcontroller, provides 3-axes linear accelerations and 3-axes angular velocities. To fulfil the inexpensive nature of the platform, it is chosen to be low-cost. In addition, better IMUs may need to be calibrated before use and would not be possible with the IMU fixed to the vehicle. Its position is the centre of the rear axle.

Wheel encoder

On the vehicle, two encoders are installed on each of the rear wheels, connecting to the LLC. The reading of the encoders provides the angular position of the wheel. Using this, the angular velocity and angular acceleration of the wheels can be calculated.

3 System overview

The aim of this thesis is to bring a vehicle from its location to a desired one. This general problem can be scoped to concern only the vehicle platform described in Chapter 2 and the solution should aim to build the basis to satisfy the greater problem of creating an AMoD service for the AAU campus. Using the previously stated facts, the development of a solution has been focused to solve the problem by incorporating these considerations.

One of the considerations for AAUs AMoD service, stated in Chapter 1, is that the vehicle's cost should be kept low. This has also been seen through the choice of sensors in Chapter 2. With all of these initial design considerations taken into account, the problem is better defined and can be further decomposed and analysed.

When analysing the objective *to bring a vehicle from its location to a desired one*, two well-defined problems are intrinsic to the full statement. The first concept is *to bring a vehicle*. At this moment, it is important to consider this statement in the context of the AMoD problem. Thus, to reformulate this statement in an engineering sense, it can be said to driving the vehicle autonomously. The second statement in the problem is *from its location to a desired one*. Thus, in a more formal context, this can be said as being able to navigate from point A to B. Considering the vehicle being autonomous, this subproblem can be stated as navigation, given a map.
Now, it can be seen that the general problem has been analysed into these two subproblems.

3.1 Autonomous driving

In this section, the problem of autonomous driving is discussed. The idea behind it is the vehicle driving by itself without human intervention. In Section 2.2 the actuation system of the vehicle is described, where the LLC is able to set references to motor, steering wheel and brake pedal. From this, it should be understood that the platform can be moved without human intervention, yet there are still no algorithms in the HLC to set these references. Thus, algorithms to create these references for the vehicle shall be developed in this report.

The chosen method to develop this algorithm is to use model-based control design. As in the name, the designed control algorithm is based on a model of the vehicle. Thus, the first task to be done is to develop a model of the Yamaha golf cart for the basis of the control algorithm. This model is explained in Chapter 4.

Once the vehicle is modelled, the control algorithms for the vehicle can be developed. The control design shall be able to autonomously navigate the car on the prescribed route from its location to the desired location. This problem is handled in Chapter 6.

At this stage it is possible to simulate the controller with the model but it is not yet possible to implement the controller on the platform. One of the critical components of control theory is feedback. While in simulation, the mathematical model is able to give information about the motion and pose of the vehicle, this cannot be provided by the vehicle itself. Due to this, the sensors described in Section 2.3 are installed on the platform. Since each sensor provides diverse information, the idea to fuse it together and provide an accurate estimate of the motion and

pose of the vehicle is explored in Chapter 7.

With the model, controller and sensor fusion developed, the vehicle is able to autonomously navigate itself from its location to another one in a predefined route. An overview of how the autonomous driving system is interconnected can be seen Figure 3.1.

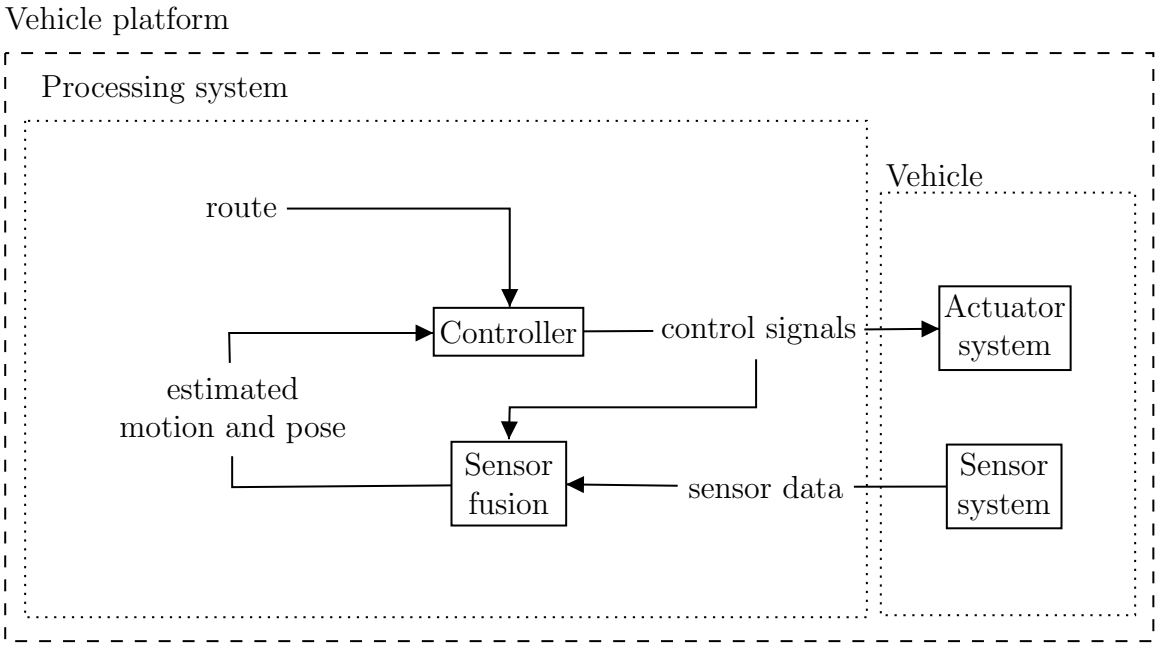


Figure 3.1: Overview of solution for autonomous driving.

Figure 3.1 shows how the algorithms of the proposed solution are implemented in the platform described in Chapter 2. It also gives meaning to the information type being transferred between the algorithms. It is important to note that the modelling to be done is not shown but is inherent in the controller and is also used to simulate the vehicle.

3.2 Navigation

In this section, the possibility of navigating from point A to B is discussed. This problem is analysed further in the context of the AAU AMoD problem. The AAU campus is an urban environment comprised of static features such as buildings, roads and road signs but also has dynamic features including additional vehicles, bicycles and people. Furthermore, there may be many possibilities in which the vehicle can reach the desired location. It can already be seen that the scope of this problem is great but as mentioned in Chapter 1, the aim of this thesis is to provide a satisfactory basis to realise the goal of creating the AMoD service at AAU.

For these reasons, the scope for navigation is reduced to consider points A and B are nearby to each other. Thus, the focus taken in Chapter 8 explores planning the geometric path between the two points, considering the behaviour of the vehicle. This should also ensure that the vehicle is able to obey traffic rules and avoid static or static obstacle.

3.3 Solution overview

With the proposed solutions of the subproblems in Section 3.1 and Section 3.2, the combined solution for the problem formulated in Chapter 1 is shown in Figure 3.2.

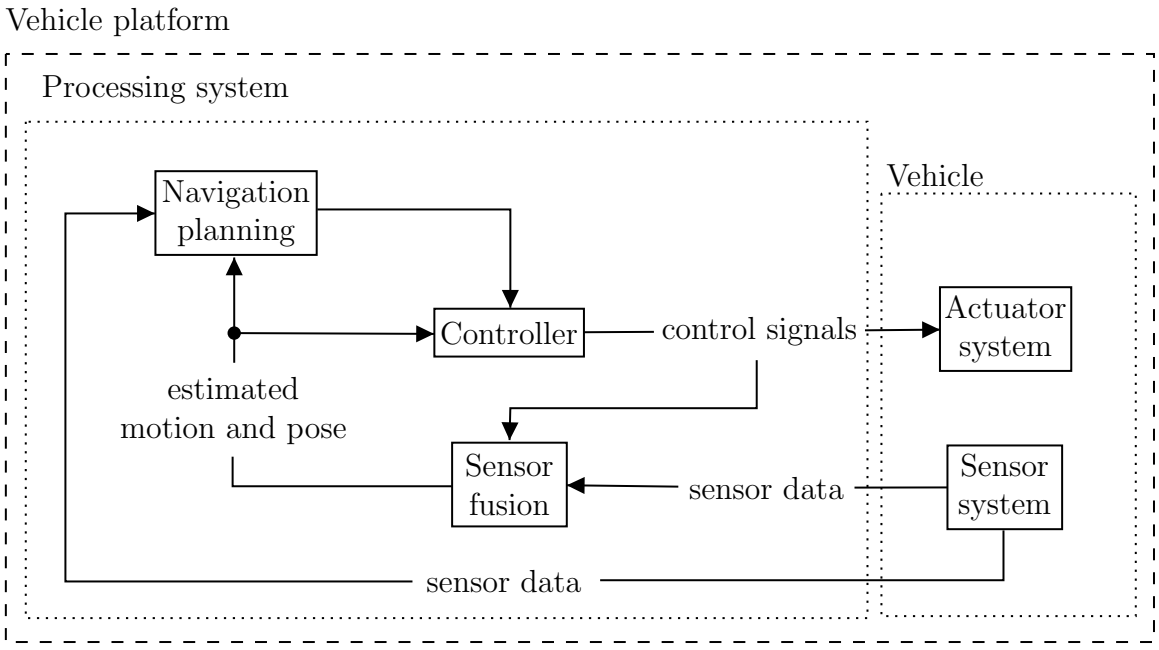


Figure 3.2: Overview of the solution to navigate an autonomous vehicle from its location to the desired location.

Figure 3.2 builds on to the developed solution shown in Figure 3.1. It adds the navigation layer to the foundation layer of autonomous driving. Notice, that the sensor data for the navigation planner differs to the sensor data for the sensor fusion. As previously stated, this solution provides a foundation for further projects to build upon in order to reach the goal for AAUs AMoD service.

Part II

Autonomous driving

4 Dynamic model

In this chapter, a first-principles model is derived using Newton-Euler methods based on the approach presented in [9] for a vehicle running at low speeds, not exceeding 15 $\frac{\text{km}}{\text{h}}$. The purposes of having a model are to design a controller and to simulate the system. For the ease of the controller design, some model simplifications may be required. However, for simulation purposes a detailed model is preferred.

In the following, two dynamic models are presented, one that satisfies the controller design requirements and another one that satisfies the simulation requirements. In Section 4.1 a framework is described such that it can be used to derive the dynamic models. The model equations of the tyre forces are described in Section 4.2. The first-principles model is then derived in Section 4.3. The complexity of the model derived in Section 4.3 makes it suitable for simulation purposes but is too complex for the controller design. Due to this, a second model is derived in Section 4.4 based on the bicycle model. This consists of a simplification of the first-principles one, where only two wheels are considered instead of the vehicle having four wheels. This simplification can be done due to the steering system used in the vehicle being of an Ackermann type, which properties are explained in detail within Section 4.4.1.
Closing the chapter, Section 4.5 goes through the parameter estimation process for the motion models, yielding the complete vehicle model.

4.1 Model framework

In the following section, the framework used to describe the motion of the vehicle is analysed.

In order to ease the modelling of the vehicle, some assumptions are made. The first assumption is to consider the vehicle's movement to be planar such that its position can be described with two coordinates. This assumption is deemed reasonable as changes in a third coordinate are expected to be very small compared with the other two, and in case these changes in the third coordinate appear to be significant they can be included in the model as disturbances. The second and last assumption is to consider the orientation of the vehicle to be described only by its heading. Because the vehicle movement has been assumed to be planar, pitch and roll are considered to remain null due to no inclination of the terrain considered. Changes in pitch and roll can occur due to accelerations, decelerations or due to cornering. However, as mentioned in the introduction of this chapter, the speed of the vehicle is limited to 15 $\frac{\text{km}}{\text{h}}$ and at such speeds it seems reasonable to consider changes in roll and pitch due to accelerations, decelerations and cornering to be negligible.

With these assumptions, the vehicle's pose, i.e. its position and orientation, is described by three degrees of freedom (DOF) as shown in Figure 4.1. There are two DOF for the vehicle's position and a third DOF for its orientation. It becomes evident then, that any frame in which the pose of the vehicle is described has to belong to $\mathbb{E}^3$ at least.

Two frames are used to describe the motion of the vehicle: an inertial frame $\mathcal{R} \in \mathbb{E}^3$ where Newtonian mechanics apply and a non-inertial frame $\mathcal{B} \in \mathbb{E}^3$ which is fixed to the body to participate in its motion. The origin of $\mathcal{B}$ is conveniently coincident with the center of gravity (G) of the vehicle and its axes are conveniently coincident with the vehicle's principal axes of inertia.

$\mathcal{B}$ is also aligned with the vehicle such that the x-axis is pointing on the heading direction of the vehicle as it can be seen in Figure 4.1.

To distinguish between the two frames, let the inertial frame $\mathcal{R}$ be defined by the X, Y and Z axes and the non-inertial frame $\mathcal{B}$ be defined by the x, y and z axes as it is shown in Figure 4.1. Moreover, a vector v in inertial frame coordinates will be denoted as $\{v\}_{\mathcal{R}}$ and a vector v in non-inertial frame coordinates will be denoted as $\{v\}_{\mathcal{B}}$.

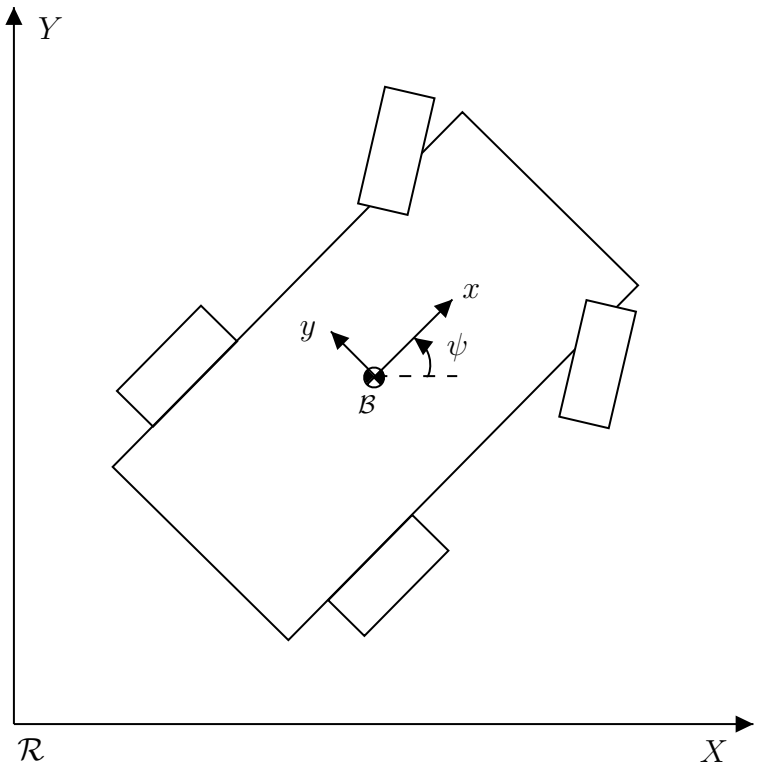


Figure 4.1: Diagram showing both the inertial frame $\mathcal{R}$ and the non-inertial frame $\mathcal{B}$.

The pose vector of the vehicle is described in the $\mathcal{R}$-frame, denoted by $\{\mathbf{x}\}_{\mathcal{R}} = \begin{bmatrix} X & Y & \psi \end{bmatrix}^T$. It cannot be described in the $\mathcal{B}$-frame but changes in the pose vector are directly affected by the vehicle's motion which is described in the $\mathcal{B}$-frame. Mapping from $\mathcal{B}$ to $\mathcal{R}$ can easily be done by multiplying by the rotation matrix in (4.1).

$$ {}^{\mathcal{R}}_{\mathcal{B}}R = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1} $$

Where ψ is the orientation of the non-inertial frame $\mathcal{B}$ with respect to the inertial frame $\mathcal{R}$ as it is shown in Figure 4.1.

4.2 Tyre model

The principle of the vehicle's motion is based on the friction between its tyres and the ground. This friction can be expressed as forces applied at the tyres, which are modelled in the following.

In Figure 4.2, the force diagram of the vehicle is shown, from which the model equations are derived in Section 4.3. From Figure 4.2, the forces $F_{xi} \in \mathbb{R}^2$ are the propulsion forces supplied by the motor. The angle δ_i is the steering angle of each wheel, being 0 for δ_3 and δ_4, as the rear wheels cannot be steered. The forces $F_{yi} \in \mathbb{R}^2$ represent the sideways friction forces between the tyres and the ground, which make the vehicle capable of turning when the front wheels are being steered, i.e. when $\delta_i \neq 0$.

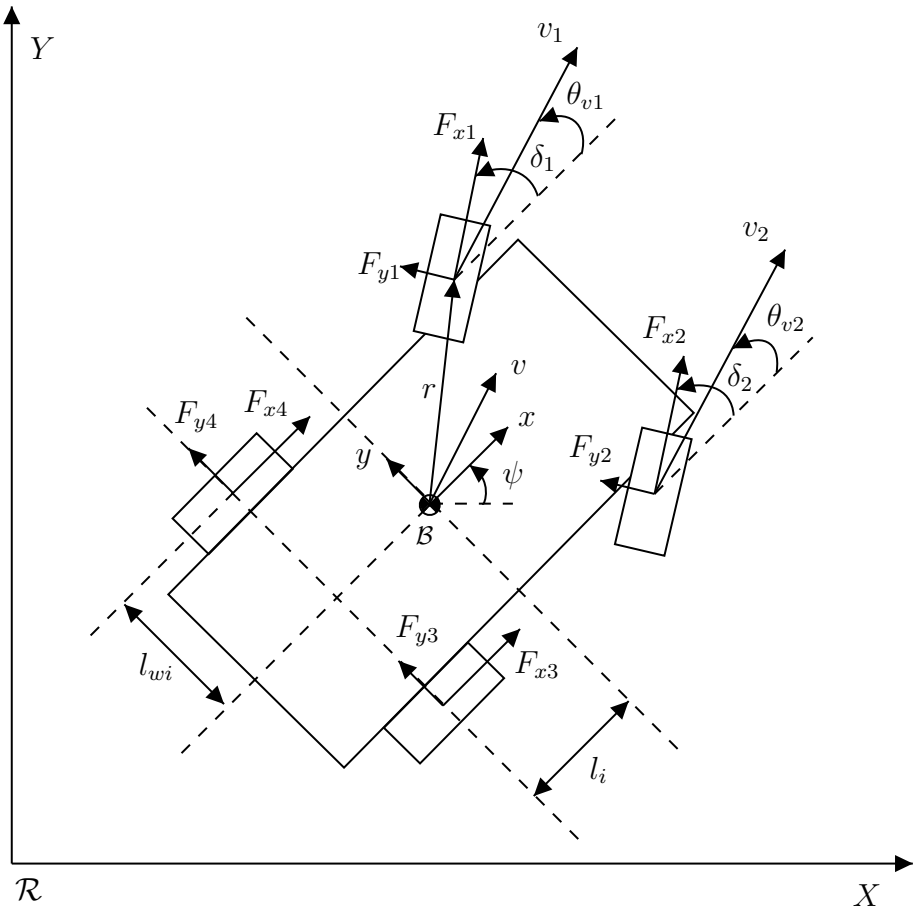


Figure 4.2: Force diagram of a four wheel vehicle with front steering and rear propelled wheels.

From [9], it is known that each lateral force, F_{yi}, depends on the slip angle of each wheel respectively, which is denoted as α_i. In [9], it is further specified that for slow speeds it is found heuristically that F_{yi} is in fact proportional to α_i, as expressed in (4.2).

$$F_{yi} = C_{\alpha i}\, \alpha_i \tag{4.2}$$

Moreover, it is further specified in [9] that for slow speeds the slip angle of each wheel is equivalent to (4.3).

$$\alpha_i = \delta_i - \theta_{vi} \tag{4.3}$$

An equivalent expression is found by plugging (4.3) into (4.2), yielding (4.4).

$$F_{yi} = C_{\alpha i}\, (\delta_i - \theta_{vi}) \tag{4.4}$$

θ_{vi} is the angle between the longitudinal axis of the vehicle, i.e. x-axis, and the actual velocity of the wheel $\{\boldsymbol{v_i}\}_{\mathcal{B}}$, see Figure 4.2.

$\{\boldsymbol{v_i}\}_{\mathcal{B}}$ is the sum of two contributions. As the vehicle is considered a rigid body, the first contribution is the velocity of G, $\{\boldsymbol{v}\}_{\mathcal{B}}$, while the second contribution comes from the rotation of the vehicle around G, which is denoted as $\{\boldsymbol{v_{ti}}\}_{\mathcal{B}}$, yielding (4.5).

$$\{\boldsymbol{v_i}\}_{\mathcal{B}} = \{\boldsymbol{v}\}_{\mathcal{B}} + \{\boldsymbol{v_{ti}}\}_{\mathcal{B}} \tag{4.5}$$

$\{\boldsymbol{v}\}_{\mathcal{B}}$ is defined as $\{\boldsymbol{v}\}_{\mathcal{B}} = \begin{bmatrix} \dot{x} & \dot{y} & 0 \end{bmatrix}_{\mathcal{B}}$. On the other hand, $\{\boldsymbol{v_{ti}}\}_{\mathcal{B}}$ can be found by differentiating a vector $\boldsymbol{r_i}$ with respect to time in the moving frame $\mathcal{B}$, where $\boldsymbol{r_i}$ is the vector going from G to the center of the i^{th} wheel as shown in Figure 4.2. The expression for the time derivative of a vector in a moving frame is defined in the literature, e.g. [10], to be as shown in (4.6).

$$\{\boldsymbol{v_{ti}}\}_{\mathcal{B}} = \left\{\dot{\boldsymbol{r}}_{\boldsymbol{i}}]_{\mathcal{R}}\right\}_{\mathcal{B}} = \frac{d}{dt}\{\boldsymbol{r_i}\}_{\mathcal{B}} + \left\{\boldsymbol{\Omega}^{\boldsymbol{\mathcal{B}}}_{\boldsymbol{\mathcal{R}}} \times \boldsymbol{r_i}\right\}_{\mathcal{B}} \tag{4.6}$$

$\left\{\boldsymbol{\Omega}^{\boldsymbol{\mathcal{B}}}_{\boldsymbol{\mathcal{R}}}\right\}_{\mathcal{B}}$ is the vector expressing the rotation of the moving frame $\mathcal{B}$ with respect to the fixed frame $\mathcal{R}$ and thus $\left\{\boldsymbol{\Omega}^{\boldsymbol{\mathcal{B}}}_{\boldsymbol{\mathcal{R}}}\right\}_{\mathcal{B}} = \begin{bmatrix} 0 & 0 & \dot{\psi} \end{bmatrix}^T$, while $\{\boldsymbol{r_i}\}_{\mathcal{B}} = \begin{bmatrix} l_i & l_{wi} & 0 \end{bmatrix}^T$. To be able to keep the equations in a generic form, l_i and l_{wi} are defined according to the $\mathcal{B}$ frame, meaning that each l_i and l_{wi} have the same sign as the axis where they lay on. Hence, as it can be seen from Figure 4.2, $l_1, l_2, l_{w1}, l_{w4} > 0$ and $l_3, l_4, l_{w2}, l_{w3} < 0$. Notice that the notation $\left\{\dot{\boldsymbol{r}}_{\boldsymbol{i}}]_{\mathcal{R}}\right\}_{\mathcal{B}}$ stands for the time derivative of the vector $\boldsymbol{r_i}$ in the moving frame $\mathcal{B}$ with respect to the fixed frame $\mathcal{R}$. Evaluating (4.6) with the given values it yields:

$$\{\boldsymbol{v_{ti}}\}_{\mathcal{B}} = \left\{\dot{\boldsymbol{r}}_{\boldsymbol{i}}]_{\mathcal{R}}\right\}_{\mathcal{B}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}_{\mathcal{B}} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}_{\mathcal{B}} \times \begin{bmatrix} l_i \\ l_{wi} \\ 0 \end{bmatrix}_{\mathcal{B}} = \begin{bmatrix} -l_{wi}\dot{\psi} \\ l_i\dot{\psi} \\ 0 \end{bmatrix}_{\mathcal{B}} \tag{4.7}$$

Once $\{\boldsymbol{v}\}_{\mathcal{B}}$ and $\{\boldsymbol{v_{ti}}\}_{\mathcal{B}}$ are known, (4.5) becomes (4.8).

$$\{\boldsymbol{v_i}\}_{\mathcal{B}} = \{\boldsymbol{v}\}_{\mathcal{B}} + \{\boldsymbol{v_{ti}}\}_{\mathcal{B}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix}_{\mathcal{B}} + \begin{bmatrix} -l_{wi}\dot{\psi} \\ l_i\dot{\psi} \\ 0 \end{bmatrix}_{\mathcal{B}} = \begin{bmatrix} \dot{x} - l_{wi}\dot{\psi} \\ \dot{y} + l_i\dot{\psi} \\ 0 \end{bmatrix}_{\mathcal{B}} \tag{4.8}$$

By using trigonometry, since $\{\boldsymbol{v_i}\}_{\mathcal{B}}$ is expressed in $\mathcal{B}$-coordinates, $\tan(\theta_{vi})$ can be found as the relation between the x-coordinate and the y-coordinate of $\{\boldsymbol{v_i}\}_{\mathcal{B}}$ such that θ_{vi} can be expressed as (4.9).

$$\theta_{vi} = \arctan\left(\frac{\dot{y} + l_i\dot{\psi}}{\dot{x} - l_{wi}\dot{\psi}}\right) \tag{4.9}$$

Plugging (4.9) into (4.4) yields (4.10), which is referred further on as the model for the sideways friction force F_{yi} without simplifications.

$$F_{yi} = C_{\alpha i}\left(\delta_i - \arctan\left(\frac{\dot{y} + l_i\dot{\psi}}{\dot{x} - l_{wi}\dot{\psi}}\right)\right) \tag{4.10}$$

The model of the propulsion force F_{xi} is obtained by applying Newton's second law for rotational bodies, to the wheel axis which is found from Figure 4.3 to be (4.11).

$$J_{\omega i}\,\dot{\omega}_i = T_{mi} - F_{xi}\,R_{\text{eff}i} \tag{4.11}$$

ω_i is the angular velocity of the i^{th} wheel, $J_{\omega i}$ is the inertia of the wheel in the axial direction of the i^{th} wheel, $R_{\text{eff}i}$ is the effective radius of the i^{th} wheel and T_{mi} is the torque supplied by the

motor when applied at the axis of the i^{th} wheel, being $T_1 = T_2 = 0\,\mathrm{N\,m}$.

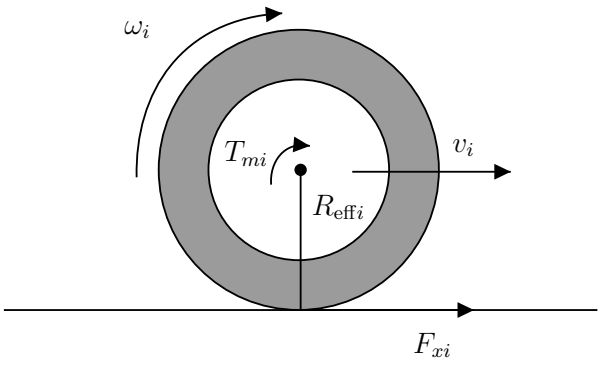


Figure 4.3: Force diagram of one rear wheel with forward speed v_i.

Due to the limited speed of the vehicle, it is reasonable to assume null tyre longitudinal slip and thus (4.12) stands.

$$\dot{x_i} = \omega_i \, R_{\mathrm{eff}i} \tag{4.12}$$

Plugging (4.12) into (4.11) and solving for F_{xi} yields (4.13), which is referred further on as the model for the propulsion force F_{xi} without simplifications.

$$F_{xi} = \frac{T_{mi}}{R_{\mathrm{eff}i}} - \frac{J_{\omega i}}{{R_{\mathrm{eff}i}}^2}\ddot{x}_i. \tag{4.13}$$

To summarise, in this section the model equations for the tyre forces have been derived. The sideways friction forces, F_{yi}, model equations are described by (4.10), whereas the propulsion forces, F_{xi}, are described by (4.13).

4.3 Motion equations

In this section, a first-principles model for the vehicle motion is derived using Newtonian mechanics. It is considered that the motion of the vehicle can be described by the accelerations in the $\mathcal{B}$-frame, $\{\ddot{\mathbf{x}}\} = \begin{bmatrix}\ddot{x} & \ddot{y} & \ddot{\psi}\end{bmatrix}^T$. The force F is the sum of the propulsion forces F_{xi}, the sideways friction forces F_{yi} and the centripetal forces acting on the vehicle due to rotational motion.

The translational movement is analysed using Newton's second law in G, where the acceleration of the vehicle is related to the applied forces as (4.14).

$$\sum \{F\} = m \, \{a\,(\mathrm{G})\}_{\mathcal{B}} \tag{4.14}$$

$\{a\,(\mathrm{G})\}_{\mathcal{B}}$ is the translational acceleration of G and it is defined as $\{a\,(\mathrm{G})\}_{\mathcal{B}} = \begin{bmatrix}a_x & a_y & 0\end{bmatrix}^T$. Extending (4.14) based on the force diagram presented in Figure 4.2 yields (4.15).

$$m \begin{bmatrix} a_x \\ a_y \\ 0 \end{bmatrix} = \sum_{i=1}^{4} \begin{bmatrix} F_{xi}\cos\delta_i - F_{yi}\cdot\sin\delta_i \\ F_{xi}\cdot\sin\delta_i + F_{yi}\cdot\cos\delta_i \\ 0 \end{bmatrix} \tag{4.15}$$

To find the components of $\{a\}_{\mathcal{B}}$, the vector $\{v\}_{\mathcal{B}}$, which has been defined in Section 4.2, is differentiated with respect to time in the moving frame $\mathcal{B}$, as shown in (4.16).

$$\{a\}_{\mathcal{B}} = \left\{ \dot{v}]_{\mathcal{R}} \right\} = \frac{d}{dt}\{v\}_{\mathcal{B}} + \left\{ \Omega^{\mathcal{B}}_{\mathcal{R}} \times v \right\}_{\mathcal{B}} \tag{4.16}$$

$\{v\}_{\mathcal{B}} = \begin{bmatrix} \dot{x} & \dot{y} & 0 \end{bmatrix}^T$ from Section 4.2 and $\Omega^{\mathcal{B}}_{\mathcal{R}}$, which represents the rotation of the $\mathcal{B}$-frame with respect to the $\mathcal{R}$-frame, is $\Omega^{\mathcal{B}}_{\mathcal{R}} = \begin{bmatrix} 0 & 0 & \dot{\psi} \end{bmatrix}$ from Section 4.2. The effect of this rotation is known as Coriolis effect. By evaluating (4.16), (4.17) is obtained.

$$\{a\}_{\mathcal{B}} = \left\{ \dot{v}]_{\mathcal{R}} \right\} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ 0 \end{bmatrix}_{\mathcal{B}} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}_{\mathcal{B}} \times \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix}_{\mathcal{B}} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ 0 \end{bmatrix}_{\mathcal{B}} + \begin{bmatrix} -\dot{y}\dot{\psi} \\ \dot{x}\dot{\psi} \\ 0 \end{bmatrix}_{\mathcal{B}} = \begin{bmatrix} \ddot{x} - \dot{y}\dot{\psi} \\ \ddot{y} + \dot{x}\dot{\psi} \\ 0 \end{bmatrix}_{\mathcal{B}} \tag{4.17}$$

Plugging the expression for the translational acceleration of G, $\{a\}_{\mathcal{B}}$, obtained in (4.17) into the translational motion equations expressed in (4.15), it yields (4.18).

$$m \cdot \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ 0 \end{bmatrix} = \sum_{i=1}^{4} \begin{bmatrix} F_{xi}\cos\delta_i - F_{yi} \cdot \sin\delta_i \\ F_{xi} \cdot \sin\delta_i + F_{yi} \cdot \cos\delta_i \\ 0 \end{bmatrix} + m \cdot \begin{bmatrix} \dot{y}\dot{\psi} \\ -\dot{x}\dot{\psi} \\ 0 \end{bmatrix} \tag{4.18}$$

(4.18) describes the translational accelerations of the vehicle. To analyse the rotational movement of the vehicle, Newton's second law applied to rotational movement is used in G, where the angular acceleration of the vehicle is related to the sum of momentum applied to G as (4.19).

$$\sum \{\tau\} = \mathrm{I} \cdot \{\alpha\}_{\mathcal{B}} \tag{4.19}$$

$\{\alpha\}_{\mathcal{B}}$ is the angular acceleration around the three axes of $\mathcal{B}$ from G. As introduced in Section 4.1, $\mathcal{B}$ is conveniently defined such that its origin is coincident with G, and its three axes are coincident with the principle axis of the vehicle. This framework definition enables the inertia tensor (I) to become diagonal such as (4.20).

$$\mathrm{I} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{4.20}$$

Therefore, the expanded version of (4.19) becomes (4.21).

$$\sum \{\tau\} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \cdot \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix}_{\mathcal{B}} \tag{4.21}$$

Where ϕ, θ and ψ are respectively the roll, pitch and yaw angles of the frame $\mathcal{B}$ with respect to the frame $\mathcal{R}$. If one recalls the assumptions taken at Section 4.1, the vehicle movement is considered to be planar, i.e. inclination of the terrain is neglected, which together with the fact of the vehicle speed being limited to 15 $\frac{\mathrm{km}}{\mathrm{h}}$, changes in roll and pitch angles are assumed null and thus $\ddot{\phi} = \ddot{\theta} = 0$. Developing (4.21) according to the force diagram represented in Figure 4.2 yields (4.22), which is the third required equation do describe the vehicle motion.

$$\begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ \ddot{\psi} \end{bmatrix}_{\mathcal{B}} = \sum_{i=1}^{4} \begin{bmatrix} 0 \\ 0 \\ l_i F_{xi}\sin\delta_i + l_i F_{yi}\cos\delta_i + l_{wi} F_{xi}\cos\delta_i - l_{wi} F_{yi}\sin\delta_i \end{bmatrix} \tag{4.22}$$

To summarise, the equations describing the motion of the vehicle according to $\ddot{\mathbf{x}} = \begin{bmatrix} \ddot{x} & \ddot{y} & \ddot{\psi} \end{bmatrix}^T$ have been derived and can be found in (4.15) and (4.22) which can be expressed together in compact form as (4.23).

$$\ddot{\mathbf{x}} = \sum_{i=1}^{4} \begin{bmatrix} \frac{1}{m}\left(F_{xi}\cos\delta_i - F_{yi}\sin\delta_i\right) \\ \frac{1}{m}\left(F_{xi}\sin\delta_i + F_{yi}\cos\delta_i\right) \\ \frac{1}{I_z}\left(l_i F_{xi}\sin\delta_i + l_i F_{yi}\cos\delta_i + l_{wi}F_{xi}\cos\delta_i - l_{wi}F_{yi}\sin\delta_i\right) \end{bmatrix} + \begin{bmatrix} \dot{y}\dot{\psi} \\ -\dot{x}\dot{\psi} \\ 0 \end{bmatrix} \tag{4.23}$$

The forces F_{xi} and F_{yi} are given by its respectively tyre model (4.13) and (4.10).

The complexity of this model makes it suitable for simulation purposes but it could potentially make the controller design troublesome. It has previously been introduced that the vehicle is equipped with an Ackermann steering and that vehicles with such steering systems are commonly modelled following what is called the bicycle model, a simpler version of the model presented in Section 4.4.3. In the following sections, the Ackermann steering and its characteristics are presented, which are used in the derivation of the bicycle model of the vehicle.

4.4 Model simplifications

Some simplifications to the motion equations presented in Section 4.3 can be done. These simplifications are meant to ease upcoming work such as the parameter estimation and controller design. The first simplification applies to the steering model, which follows an Ackerman geometry. In Section 4.4.1 the properties of this geometry are explored, and it is seen that the steering model can be reduced from two steered wheels to only one virtual wheel. The second simplification applies to the tyre model, in Section 4.4.2, the small angles approximation is explored to reduce the nonlinearities in the tyre model.

4.4.1 Steering model simplification

For a front wheel steered vehicle as the one shown in Figure 4.4 to be able to make a turn, the inner and outer wheels have to follow a trace with different radius each. The vehicle is a rigid body and it is known that the rotational motion of a rigid body is defined around an Instantaneous Center of Rotation (ICR), also referred as Instantaneous Center of Zero Velocity in [10]. The existence of an ICR is proven in [10] in the following way. Assume that the vehicle is performing an absolute circular motion, and consider any two points of the vehicle A and B with non-parallel velocities. If there is a point around which A has absolute circular motion, it is true that this point lies on the normal to v_a and thus the same reasoning can be applied to B. There must exist a point where all normal directions intersect, which is the ICR. In the case study, it is proven that both front wheels have to be steered with different angles such that the normals to v_1 and v_2 intersect at the ICR.

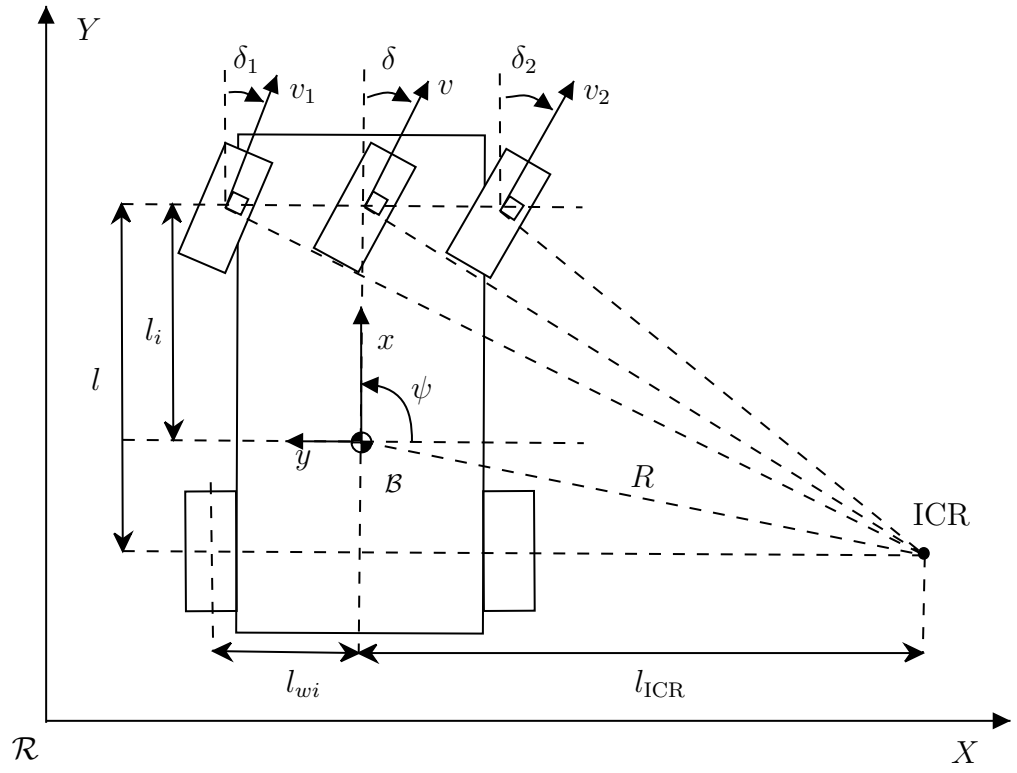


Figure 4.4: Illustration of the Ackermann steering geometry for a four wheel vehicle with front steered wheels where the concept of ICR is represented. The virtual wheel with steering angle δ, result of the Ackermann geometry, is also shown.

Following the same reasoning, it is possible to find the steering angle of a virtual wheel which lies in the midpoint of the front axle as it is shown in Figure 4.4. Note, it is also assumed that there is null slip [11], hence, the velocity of the wheel v_i is coincident with the corresponding steering angle δ_i. It can be proven that the steering model can then be described only by this virtual wheel. To do so, an expression that relates the steering angle of the wheels to the virtual wheel is found in the following. From Figure 4.4, the trigonometric relations

$$\tan\delta = \frac{l}{l_{ICR}} \tag{4.24}$$

$$\tan\delta_i = \frac{l}{l_{ICR} + l_{wi}\,\text{sgn}\,(\delta_i)} \tag{4.25}$$

result evident, where the sign function is defined as (4.26) and it is used to distinguish if the i^{th} wheel is inner or outer to the turn, and as well as it is considered in Section 4.3, the signs of l_{wi} are chosen according to their position with respect to the $\mathcal{B}$-frame, i.e. $l_{w3} < 0$ and $l_4 > 0$.

$$\text{sgn}\,(\delta_i) = \begin{cases} -1 & , \quad \delta_i < 0 \\ 1 & , \quad \delta_i > 0 \\ 0 & , \quad \delta_i = 0 \end{cases} \tag{4.26}$$

Solving then (4.25) for l_{ICR}, it is obtained that

$$l_{\text{ICR}} = \frac{l}{\tan\delta_i} + l_{wi}\,\text{sgn}\,(\delta_i)\,,$$

which can be plugged into (4.24), and solving for δ yields (4.27), the expression that relates δ with δ_i.

$$\delta = \arctan\left(\frac{l}{\frac{l}{\tan\delta_i} + l_{wi}\,\mathrm{sgn}\,(\delta_i)}\right) \tag{4.27}$$

The sign function of an angle can alternatively be expressed as $\mathrm{sgn}\,(\delta_i) = \frac{\tan|\delta_i|}{\tan\delta_i}$. Therefore, expression found in (4.27) can equivalently be written as (4.28).

$$\delta = \arctan\left(\frac{l\ \tan\delta_i}{l + l_{wi}\ \tan|\delta_i|}\right) \tag{4.28}$$

The use of one unique δ is common to simplify modelling of vehicles with front steering wheels, where the vehicle is modelled with only one front wheel with steering angle δ and only one rear wheel. This simplified model is commonly known as bicycle model, seen for example in [12], [13] and [14]. The motion equations of the bicycle model are derived in Section 4.4 and are used later for the parameter estimation and controller design.

4.4.2 Tyre model simplification

In Section 4.2, a tyre model inspired in [9] is presented. The model of one tyre is based on two components, sideways friction force noted as F_{yi} and modelled in (4.10), and propulsion force noted as F_{xi} and modelled in (4.13). This subsection explores the simplification of the expression for F_{yi} in (4.10) by linearising it around $\theta_{vi} = 0$ or in other words, by applying the small angles approximation. Before, however, it is essential to notice the difference between front sideways forces, $\left\{F_{y1}, F_{y2}\right\}$, and rear sideways forces, $\left\{F_{y3}, F_{y4}\right\}$. From (4.4), front sideways forces F_{y1} and F_{y2} can be expressed as (4.29).

$$\begin{aligned} F_{y1} &= C_{\alpha 1}\ (\delta_1 - \theta_{v1}) \\ F_{y2} &= C_{\alpha 2}\ (\delta_2 - \theta_{v2}) \end{aligned} \tag{4.29}$$

The delta term in (4.4) for the rear sideways forces is zero, given that δ_3 and δ_4 are null. Thus, (4.4) becomes (4.30).

$$\begin{aligned} F_{y3} &= C_{\alpha 3}\ (-\theta_{v3}) \\ F_{y4} &= C_{\alpha 4}\ (-\theta_{v4}) \end{aligned} \tag{4.30}$$

Because rear sideways forces do not depend on any steering angle, they only depend on pure sideways slip of the full vehicle, which at slow speeds is expected to be close to zero. On the other hand, front sideways forces depend on the difference between the steering angles δ_3, δ_4 and the velocity direction of the respective wheel. As it can be seen, the nature of the rear and front sideways forces is different and for that reason the linearisation of the respective expressions has to be treated separately.

Front sideways forces

First, the case of the front sideways forces is analysed. To do so, (4.29) is expanded such as (4.10), yielding the expressions in (4.31).

$$\begin{aligned} F_{y1} &= C_{\alpha 1}\left(\delta_1 - \underbrace{\arctan\left(\frac{\dot{y}+l_1\dot{\psi}}{\dot{x}-l_{w1}\dot{\psi}}\right)}_{\theta_{v1}}\right) \\ F_{y2} &= C_{\alpha 2}\left(\delta_2 - \underbrace{\arctan\left(\frac{\dot{y}+l_2\dot{\psi}}{\dot{x}-l_{w2}\dot{\psi}}\right)}_{\theta_{v2}}\right) \end{aligned} \tag{4.31}$$

These can be simplified if the terms θ_{v1} and θ_{v2} are linearised around zero. To explore this possibility, the interpretation of θ_{v1} has to be analysed. Consider a scenario where the vehicle is performing a turn without slipping. The velocity of the i^{th} front wheel will be aligned with δ_i, and thus $\theta_{vi} = \delta_i$. If the same scenario is taken with the vehicle performing its maximum sharpen turn, i.e $\delta_{i,\max}$, then it is the scenario where θ_{vi} is maximum. In the following, this scenario is investigated. To do so, the value for $\theta_{vi,\max}$ is found according to $\delta_{i,\max}$ and then it is seen whether it falls inside an acceptable range defined by the linearization of the arctangent function around zero. To draw this acceptable range, a representation of the arctangent function and its linearised version with operating point zero is shown in Figure 4.5. From Figure 4.5, it is reasonable then to consider the acceptable range to be $\arctan\theta_{v1} \approx \theta_{v1}$ for $|\theta_{v1}| \leq 0.25$.

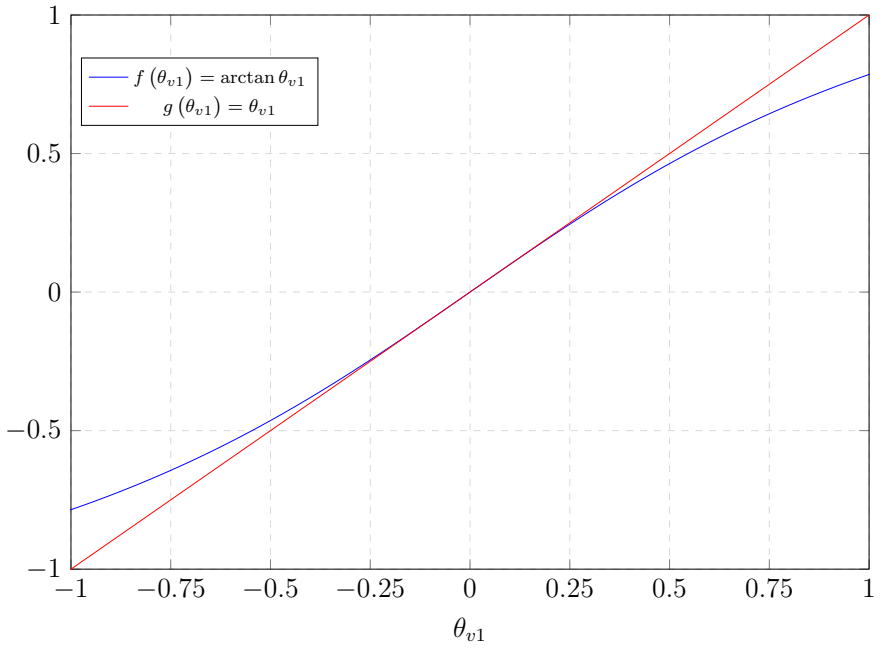


Figure 4.5: Plot showing the comparison between the function $f(\theta_{vi}) = \arctan\theta_{vi}$ in blue and the function $g(\theta_{vi}) = \theta_{vi}$ in red which can be used to determine whether the function $f(\theta_{vi})$ can be linearised around zero.

The maximum steering angle, $\delta_{i,\max}$, is found with trigonometric relations given the minimum instantaneous cornering radius, $R_{\min}$, found in [15]. From Figure 4.4, the relation (4.32) results evident.

$$R = \sqrt{l_r^2 + l_{\text{ICR}}^2} \tag{4.32}$$

In (4.32), $l_r = l_3 = l_4$. Moreover, solving (4.24) for l_{ICR} and then plugging it to (4.32) yields (4.33).

$$R = \sqrt{l_r^2 + \left(\frac{l}{\tan \delta_1}\right)^2} \tag{4.33}$$

To calculate the steering angle δ_i given the turning radius, (4.33) has to be solved for δ_i such as (4.34).

$$\delta_i = \arctan\left(\sqrt{\frac{l^2}{R^2 - l_r^2}}\right) \tag{4.34}$$

If (4.34) is evaluated with $R = R_{\min} = 2.8$ m, $l_r = 0.72$ m and $l = 2.37$ m, which can be found in [15], it yields a maximum steering angle of $\delta_{i,\max} = 0.7193$ rad. As defined before, the scenario taken into account is when the vehicle performs the sharpest turn possible and the wheels do not slip, thus $\theta_{vi,\max} = \delta_{i,\max} = 0.7193$ rad which lays outside of the acceptable range that was defined to be $|\theta_{vi}| \leq 0.25$.

However, it is decided to use small angles approximation with operating point at zero and compensate for deviations in the controller design. If deviations appear to be significantly different, operating points should be taken and different controllers could be designed according to these. The expression (4.31) with small angles approximation becomes (4.35).

$$\begin{aligned} F_{y1} &= C_{\alpha 1} \cdot \left(\delta_1 - \arctan\left(\tfrac{\dot{y}+l_1\dot{\psi}}{\dot{x}-l_{w1}\dot{\psi}}\right)\right) \approx C_{\alpha 1} \cdot \left(\delta_1 - \tfrac{\dot{y}+l_1\dot{\psi}}{\dot{x}-l_{w1}\dot{\psi}}\right) \\ F_{y2} &= C_{\alpha 2} \cdot \left(\delta_2 - \arctan\left(\tfrac{\dot{y}+l_2\dot{\psi}}{\dot{x}-l_{w2}\dot{\psi}}\right)\right) \approx C_{\alpha 2} \cdot \left(\delta_2 - \tfrac{\dot{y}+l_2\dot{\psi}}{\dot{x}-l_{w2}\dot{\psi}}\right) \end{aligned} \tag{4.35}$$

Rear sideways forces

A similar procedure can be followed to simplify the sideways force expression for the rear wheels. For this case (4.30) is expanded such as (4.10), yielding (4.36) which can be simplified if the terms θ_{v3} and θ_{v4} are linearised around zero.

$$\begin{aligned} F_{y3} &= C_{\alpha 3} \left(\underbrace{-\arctan\left(\frac{\dot{y} + l_3\dot{\psi}}{\dot{x} - l_{w3}\dot{\psi}}\right)}_{\theta_{v3}} \right) \\ F_{y4} &= C_{\alpha 4} \left(\underbrace{-\arctan\left(\frac{\dot{y} + l_4\dot{\psi}}{\dot{x} - l_{w4}\dot{\psi}}\right)}_{\theta_{v4}} \right) \end{aligned} \tag{4.36}$$

Differently than the front sideways forces, rear sideways forces are only dependent on pure slip of the full vehicle, which is expected to be small given the limited speed of the vehicle. Therefore, it is deemed reasonable to consider θ_{v3} and θ_{v4} to be small angles and apply the linearisation around zero such that (4.36) becomes (4.37).

$$\begin{aligned} F_{y3} &= C_{\alpha 3}\left(-\arctan\left(\frac{\dot{y}+l_3\dot{\psi}}{\dot{x}-l_{w3}\dot{\psi}}\right)\right) \approx C_{\alpha 3}\left(\delta_3 - \frac{\dot{y}+l_3\dot{\psi}}{\dot{x}-l_{w3}\dot{\psi}}\right) \\ F_{y4} &= C_{\alpha 4}\left(-\arctan\left(\frac{\dot{y}+l_4\dot{\psi}}{\dot{x}-l_{w4}\dot{\psi}}\right)\right) \approx C_{\alpha 4}\left(\delta_4 - \frac{\dot{y}+l_4\dot{\psi}}{\dot{x}-l_{w4}\dot{\psi}}\right) \end{aligned} \tag{4.37}$$

4.4.3 Bicycle model

For upcoming work, i.e. parameter estimation and controller design, it might be beneficial to use a simplified version of the model equations presented in Section 4.3. As discussed previously, a commonly used model for vehicles with an Ackermann steering is the bicycle model. The bicycle model is based on the principles presented in Section 4.4.1, which enable to consider a unique front steering wheel and a unique rear propulsion wheel as shown in Figure 4.6. Throughout this section, the equations for the bicycle model are derived. The derivation is based on the force diagram presented in Figure 4.6 and it follows the same principles used to derive the model equations in Section 4.3. Notice that the framework defined in Section 4.1 is used here as well.

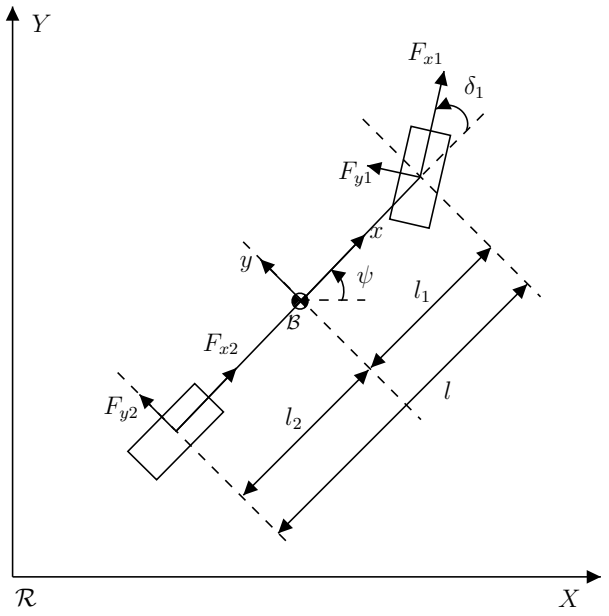


Figure 4.6: Force diagram of the bicycle model with front steered wheel which is equivalent to the virtual wheel found in Section 4.4.1.

From Figure 4.6, it can be seen that, due to the front wheel not being propelled, $F_{x1} = 0$ and that since only the front wheel is steered, $\delta_2 = 0$, while δ_1 is equivalent to the virtual δ found in Section 4.4.1. Moreover, as well as it is considered in Section 4.3, signs of l_i are chosen according to their position with respect to the $\mathcal{B}$-frame, i.e. $l_2 < 0$ and $l_1 > 0$.

The translational movement is also analysed using Newton's second law as in Section 4.3 and the equations describing it end up being the same with only two wheels considered. Hence, (4.15) becomes (4.38).

$$m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ 0\end{bmatrix} = \sum_{i=1}^{2}\begin{bmatrix}F_{xi}\cos\delta_i - F_{yi}\sin\delta_i\\ F_{xi}\sin\delta_i + F_{yi}\cos\delta_i\\ 0\end{bmatrix} + m\begin{bmatrix}\dot{y}\dot{\psi}\\ -\dot{x}\dot{\psi}\\ 0\end{bmatrix} \tag{4.38}$$

This time, since only two wheels are considered, it is easy to develop this equation, which can be expressed as two different equations: one for the translational movement in x, (4.39), and another one for the translational movement in y, (4.40).

$$m\,\ddot{x} = F_{x2} - F_{y1}\,\sin\delta_1 + m\,\dot{y}\dot{\psi} \tag{4.39}$$

$$m\,\ddot{y} = F_{y1}\,\cos\delta_1 + F_{y2} - m\,\dot{x}\dot{\psi} \tag{4.40}$$

The same procedure can be performed with the rotational movement. As it is done in Section 4.3, rotational movement is analysed using Newton's second law applied to rotational movements around G. The equations describing it can be derived from (4.22) applied to two wheels, with the main difference in this case being that the terms multiplied by l_{wi} cancel out. Therefore, (4.22) becomes (4.41).

$$\begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ \ddot{\psi} \end{bmatrix}_{\mathcal{B}} = \sum_{i=1}^{2} \begin{bmatrix} 0 \\ 0 \\ l_i F_{xi} \sin\delta_i + l_i F_{yi} \cos\delta_i \end{bmatrix} \tag{4.41}$$

If (4.41) is developed as it has been done with the translational movement equations, it yields (4.42).

$$I_z\,\ddot{\psi} = l_1 F_{y1} \cos\delta_1 + l_2 F_{y2} \tag{4.42}$$

It has to be noticed that the equation has to agree to the sign convention taken, $l_2 < 0$.

With (4.39), (4.40) and (4.42), the bicycle model is concluded. In the following, it is used such that it is easier to estimate the parameters of the model and further on, in Chapter 6 a version of the bicycle model with further simplifications is used to design a controller to fulfil the objective.

4.5 Parameter estimation

The motion model equations found describing the system depend on diverse parameters related to the vehicle. To be able to implement a simulation of the vehicle, as well as future controllers, all of the parameters describing the model need to be found.
It is clear that some of the parameters of the model will not be fixed to a specific value and instead, they can change under different circumstances or over time. The mass, inertias, distances to G and radius of the wheels are known to vary in each situation the vehicle can be in. Therefore, variances on these values need to be included in the controller design. But, nevertheless, the model needs to be validated, and nominal values for the parameters need to be found.

Since some of the parameters are known from the specifications of the vehicle, such as the radius of the wheels or the distances to G, and others such as the mass of the vehicle can be considered as known for specific experiments; it is decided to fix the value of these to ease the estimation of the rest of the parameters. Furthermore, to make the estimation process easier, the bicycle model of the vehicle described in Section 4.4.3 is used, leaving then a reduced amount of parameters to be estimated. However, no other simplifications of the model are used, maintaining the non-simplified version of the tyre model.
To find the values for the parameters, it is decided to use the Parameter Estimation Toolbox within Simulink, which given the inputs and outputs of the system fits the simulation model implemented by changing the specified parameters.

After analysing the model equations of the bicycle model; the parameters needed to be estimated are found to be the inertia of the vehicle, I_z, the inertia of the wheels, J_w, and the longitudinal cornering coefficients in the front and back wheels, C_{lf} and C_{lr} respectively.

However, it is known from Section 2.2 that the actuation on the cart does not consist on a torque input as considered in the model derived. Instead, through the drive-train system, the rotational speed of the wheels could be considered to be the input of it. From the wheel encoders, it is simple to extract the rotational speed of each back wheel. Therefore, a model from the accessible rotational speed in the wheels to torque needs to be included in the derived model.
To design this mapping, the dynamics of the motor shown in (4.43) are considered.

$$J_m \dot{\omega}_m = T - b\,\omega_m \tag{4.43}$$

Where:

J_m	Is the inertia of the motor	$\left[\mathrm{kg\,m^2}\right]$
ω_m	Is the rotational speed of the motor	$\left[\frac{\mathrm{rad}}{\mathrm{s}}\right]$
T	Is the torque applied in the motor	$[\mathrm{N\,m}]$
b	Is a friction coefficient	$[\mathrm{N\,m\,s}]$

To map from the torque applied in the motor to the one applied at the wheels, the gearbox ratio is considered, which is known from the vehicle specifications to be $1 : 11.34$.
However, including this map in the model adds two more parameters to be estimated, since their value is unknown. This hardens the estimation based on the full model derived when all parameters are forced to be estimated at once.

Hence, the parameter estimation is divided into two different steps. Since it is seen from the equations describing the model that when the longitudinal velocity and acceleration are known the torque does not affect the other two variables of the system, a first estimation of the set of parameters without the ones coming from the motor is performed. This is accomplished taking the steering, the longitudinal velocity and the acceleration as inputs, and fitting the yaw rate and lateral acceleration to the measured values from the IMU.
Once these parameters are found to fit the experiment data, an estimation is performed with the full model fixing their values to the estimated values. Taking now the rotational speed of the wheels and the steering as inputs to the system and fitting the simulated longitudinal velocity to the one measured by the GPS, the values for the parameters describing the motor dynamics are estimated.

The final estimated values for the parameters are shown in Table 4.1.

Parameter	Value	Unit
C_{lf}	41498	
C_{lr}	30009	
I_z	350.72	$\mathrm{kg\,m^2}$
J_w	12.046	$\mathrm{kg\,m^2}$
J_m	42.717	$\mathrm{kg\,m^2}$
b	0.13993	$\mathrm{N\,m\,s}$

Table 4.1: Estimated parameters of the bicycle model.

To analyse the fitting of the variables, the plot of their evolution during the simulation is compared with the data extracted from the sensors in the vehicle.
From the IMU, the heading rate of the vehicle can be extracted from the gyroscope, allowing a direct comparison between the simulated and measurement values of $\dot{\psi}$. The heading rate measured during the experiment and the one given by the simulation are compared in Figure 4.7.

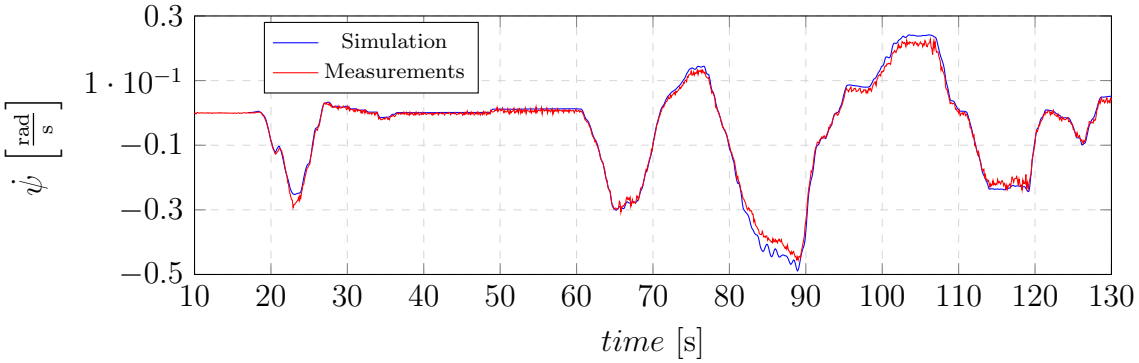


Figure 4.7: Simulated yaw rate compared to the measured data from the gyroscope in the IMU.

Furthermore, the accelerations in the IMU position can be extracted from the accelerometer in the IMU. However, these are not the ones being simulated in the model, since the IMU is not considered to be placed in G. Hence, the relation between the acceleration in the IMU location and G needs to be found.
First, the velocity measured in the IMU position is derived in the body frame. This translation is performed in Section 4.2 and shown in (4.8) for the four wheel model of the vehicle. If the derivation is extrapolated to the bicycle model, the velocity in the IMU is described as in (4.44).

$$\{v_{IMU}\}_{\mathcal{B}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -l_r\,\dot{\psi} \\ 0 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} - l_r\,\dot{\psi} \\ 0 \end{bmatrix} \tag{4.44}$$

Now, in order to calculate the measured acceleration in the IMU, the same procedure used for the velocity is used, shown in (4.45).

$$\{a_{IMU}\}_{\mathcal{B}} = \left\{ \dot{v}_{IMU}]_{\mathcal{R}} \right\}_{\mathcal{B}} = \left\{ \dot{v}_G]_{\mathcal{R}} \right\}_{\mathcal{B}} + \left\{ \dot{v}_{t2}]_{\mathcal{R}} \right\}_{\mathcal{B}} \tag{4.45}$$

The acceleration in G is derived in (4.17), in Section 4.3. On the other hand, the second vector defining the acceleration in the IMU is defined as the derivative of the second component of the velocity in (4.44). Therefore, this vector can be derived as shown in (4.46), which plugged into (4.45) yields the final expression for the acceleration in the IMU, as in (4.47).

$$
\begin{aligned}
\left\{\dot{v}_{t2}]_{\mathcal{R}}\right\}_{\mathcal{B}} &= \frac{d}{dt}\{v_{t2}\}_{\mathcal{B}} + \left\{\Omega_{\mathcal{R}}^{\mathcal{B}} \times v_{t2}\right\}_{\mathcal{B}} \\
&= \frac{d}{dt}\begin{bmatrix} 0 \\ -l_r\,\dot{\psi} \\ 0 \end{bmatrix}_{\mathcal{B}} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}_{\mathcal{B}} \times \begin{bmatrix} 0 \\ -l_r\,\dot{\psi} \\ 0 \end{bmatrix}_{\mathcal{B}} \\
&= \begin{bmatrix} 0 \\ -l_r\,\ddot{\psi} \\ 0 \end{bmatrix}_{\mathcal{B}} + \begin{bmatrix} l_r\,\dot{\psi}^2 \\ 0 \\ 0 \end{bmatrix}_{\mathcal{B}} \\
&= \begin{bmatrix} l_r\,\dot{\psi}^2 \\ -l_r\,\ddot{\psi} \\ 0 \end{bmatrix}_{\mathcal{B}}
\end{aligned} \tag{4.46}
$$

$$
\{a_{IMU}\}_{\mathcal{B}} = \begin{bmatrix} \ddot{x} - \dot{\psi}\dot{y} \\ \ddot{y} + \dot{\psi}\dot{x} \\ 0 \end{bmatrix}_{\mathcal{B}} + \begin{bmatrix} l_r\,\dot{\psi}^2 \\ -l_r\,\ddot{\psi} \\ 0 \end{bmatrix}_{\mathcal{B}} = \begin{bmatrix} \ddot{x} - \dot{\psi}\dot{y} + l_r\,\dot{\psi}^2 \\ \ddot{y} + \dot{\psi}\dot{x} - l_r\,\ddot{\psi} \\ 0 \end{bmatrix}_{\mathcal{B}} \tag{4.47}
$$

Once this relation is derived, the simulation results can be translated to the IMU position and then compare the values with those extracted from the accelerometer. Although, during the experiment it is observed that the roll angle around the forward pointing direction of the vehicle is of a considerable value when the vehicle turns, due to its suspension. Since the IMU can provide an approximation of the rotations around its principal axes, an estimate of these values during the whole experiment can be extracted. The values found are used to transform the measured acceleration in the IMU during the experiment from a rotating frame to the planar one being considered in the model. The comparison between the simulated acceleration in the IMU location and the rotated one coming from the accelerometer is presented in Figure 4.8.

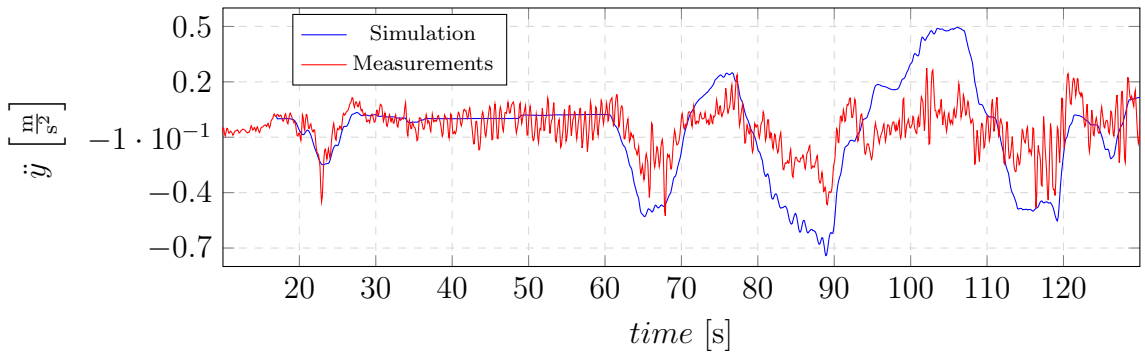


Figure 4.8: Simulated acceleration in the IMU location compared to the measurements extracted from the accelerometer in the IMU, once these have been rotated according to measurements.

As for the longitudinal velocity of the vehicle, it is deemed necessary to have an accurate model of it, since it is of big interest for control purposes. Therefore, instead of fitting the parameters for the longitudinal acceleration that could be extracted from the IMU, it is decided to fit the parameters to the measured velocity from the GPS. It is possible to get an approximation of the longitudinal velocity of the vehicle from the GPS data, considering that the vehicle always moves forward between each GPS measurement. Moreover, another approximation of the longitudinal velocity is extracted from the encoders in the rear wheels. The rotational speed of the wheels is

extracted from the encoders, and the longitudinal velocity is calculated multiplying it times the radius of the wheels.
Both approximations are compared and show a big resemblance, as seen in Figure 4.9. However, it is decided to use the GPS data for the estimation process since the conversion from the encoders to rotational speed is not considered accurate enough.

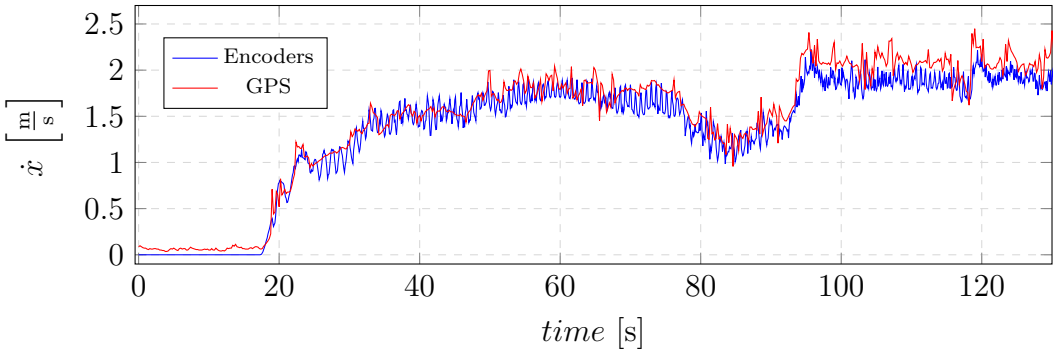


Figure 4.9: Longitudinal velocity extracted from the GPS compared to the encoders conversion from the back wheels.

The fitting results for the longitudinal velocity are shown in Figure 4.10.

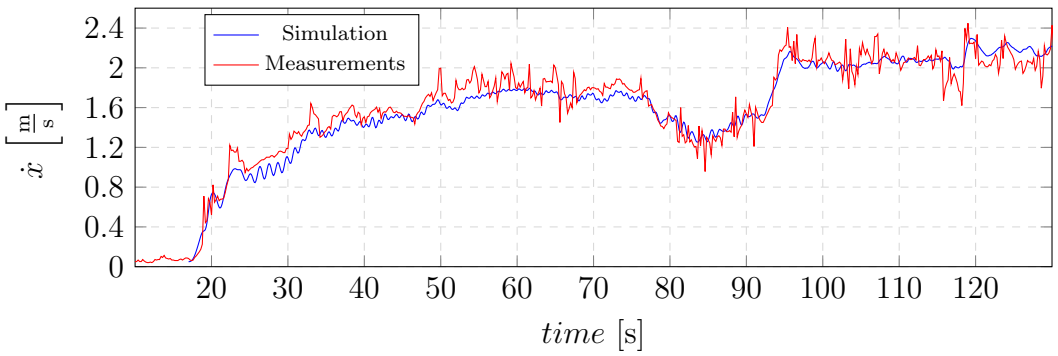


Figure 4.10: Simulated longitudinal velocity compared to the measurements extracted from the GPS.

During the estimation of the parameters, the data used for the fitting of the simulation is extracted from an experiment with the vehicle in which the driving shown in Figure 4.11 is performed. The data shown in the 2D plot of the global coordinates is extracted from the GPS, and is compared to the simulation, starting the experiment at the same point. It can be seen from the simulation results that both plots do not match exactly, but the performance of the vehicle is analogous. This is due to the result of integrating non-zero bias signals over extended periods of time. Moreover, the initial orientation of the simulation is inaccurate, which might increase the difference between the two plots. From the GPS data plot on top of the map, it can also be seen how this data is not perfectly accurate, since it shows a path inside a building at some points.

Figure 4.11: 2D plot of the global coordinates X and Y. The plot shows the comparison between the GPS data from the experiment and the simulated results on top of the map location.

4.5.1 Model validation

Once these values have been found for the parameters, it is necessary to confirm their validity. The parameters have been fitted with one specific experiment, and thus, it has to be shown how they fit with a different experiment.
Therefore, a simulation is run with the inputs extracted from a different experiment performed with the vehicle. Afterwards, the results shown by the simulation are compared with the measurements extracted from the sensors. Figure 4.12, Figure 4.13 and Figure 4.14 show the comparison between the measured data and the simulation results, in the same way it is performed during the estimation.

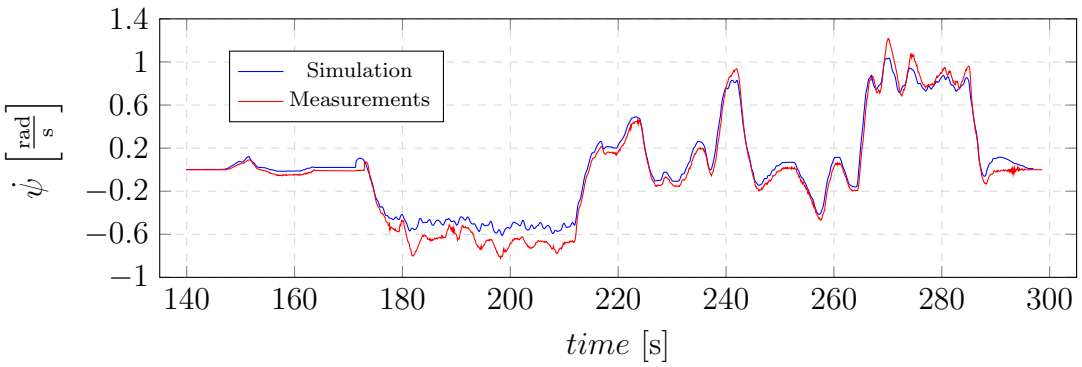


Figure 4.12: Simulated yaw rate compared to the measured data from the gyroscope in the IMU.

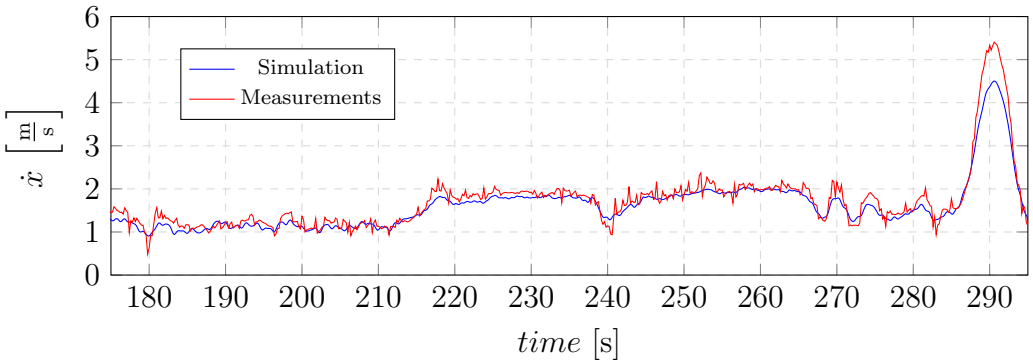


Figure 4.13: Simulated longitudinal velocity compared to the measurements extracted from the GPS.

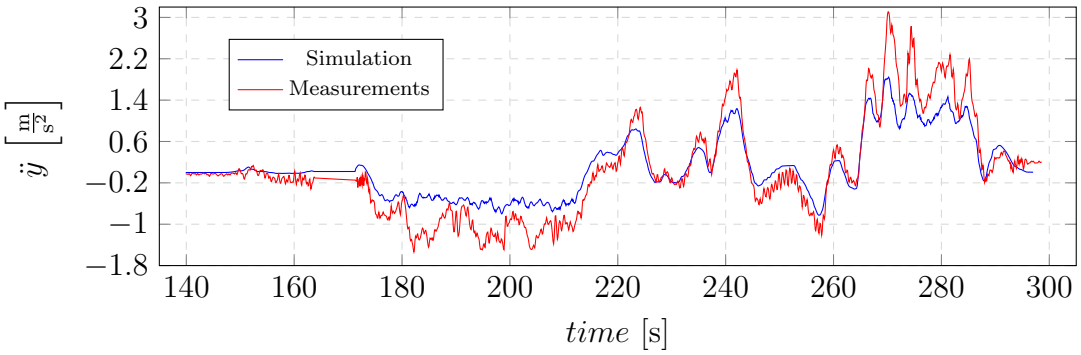


Figure 4.14: Simulated acceleration in the IMU location compared to the measurements extracted from the accelerometer in the IMU, once these have been rotated according to measurements.

In can be noted specifically in Figure 4.12 and Figure 4.14 that in the higher measured values the model does not fit with great precision. These differences in acceleration and velocity values will result in bigger errors when integrated to position values. However, the simulated plots show a comparable behaviour to the measurements. Therefore, an estimator based on the model derived and the measurements provided is expected to provide a good prediction of the variables' values.

4.6 Dynamic model conclusion

In this chapter, a model of the platform is derived from first-principles. For simulation purposes, a complete dynamic model is first presented. Afterwards, using model simplifications, a simpler version of it is found in order to ease the controller design. Parameter estimation is performed based on the latter model. The values found are used in the simulation model and its validation is deemed satisfactory when compared to sensor measurements.

5 Actuator models

In this chapter, a model for each actuator of the platform characterised in Section 2.2 is derived. It is known from Section 2.2 that actuation is needed to operate the vehicle autonomously. The actuation system is formed by the original drive-train mounted on the vehicle and two additional actuators, one of the steering wheel and a another one of the brake. Additionally, controllers have been implemented previously within the LLC for each actuator. In this section, models for each of the systems are found.

First, in Section 5.1 a model for the steering system is derived based on known properties of the actuator. Afterwards, a model for the drive-train system is derived in Section 5.2 based on the actuator properties and the golf cart structure. In every section, an estimation of the parameters present in the derived models is shown. Last, a model of the braking system is derived in Section 5.3 based on the capabilities of the cart actuators.

5.1 Steering system model

Recalling from Section 2.2, the steering system is based on an actuator which is coupled to the steering column using gears and timing belts which enables to set steering wheel positions. However, it is known from Section 4.3 that the model of the vehicle accepts steering angle δ_i inputs or equivalently Ackermann steering angle δ inputs. In the following, a model for the steering system is derived. The model consists of two parts, the relation from steering wheel, β, to Ackermann steering angle, δ, known as steering ratio and the dynamics of the actuator. A block diagram shows the structure of the steering model in Figure 5.1.

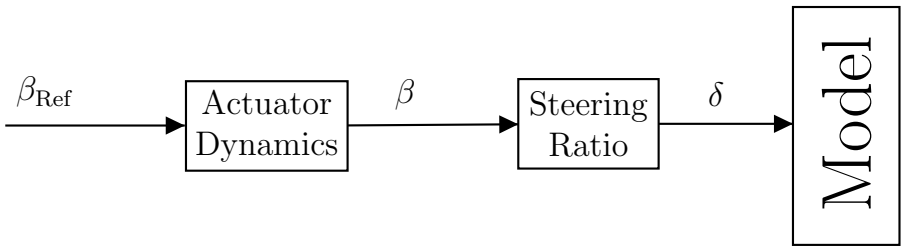


Figure 5.1: Block diagram of the steering actuation model. It includes actuator dynamics from β_{Ref} to β and the relation between β and δ known as steering ratio.

The steering gear-set of the vehicle is rack-and-pinion, from [16] it is known that the steering ratio for these gear-sets are nonlinear. However, the nonlinearities are expected to be rather small, thus it is considered linear. The steering ratio is found empirically, and to find it $\beta_{\max}$ and $\beta_{\min}$ are measured while $\delta_{\max}$ and $\delta_{\min}$ are known given the minimum turning radius of the vehicle in [15]. The steering ratio is then found straightforward as (5.1) considering a straight line with slope $S = \frac{\delta_{\max} - \delta_{\min}}{\beta_{\max} - \beta_{\min}}$.

$$\delta = \delta_{\min} + S\ (\beta - \beta_{\min}) \tag{5.1}$$

Using the encoder of the actuator, it is possible to measure β such that $\beta_{\max}$ and $\beta_{\min}$ can be found by manually steering in both directions. $\beta_{\max}$ and $\beta_{\min}$ are measured three times each and

the three measurements are averaged. The values that have been found are $\beta_{\max} = 2.4756 \cdot 10^4$ and $\beta_{\min} = -2.3557 \cdot 10^4$ while it is known from Section 4.4.2 that $|\delta|_{\max} = 0.7193$ rad. Therefore $S = 2.9777 \cdot 10^{-5}$.

The second part of the modelling consists of the actuator dynamics. For the actuator to be able to reach specified β_{Ref}, a controller was previously implemented in the LLC. This low level controller has been implemented in-house and thus it is known that it only includes proportional action. The block diagram of the steering actuator is the one shown in Figure 5.2 where K_p represents the proportional controller gain and $G(s)$ represents the transfer function (TF) of the actuator itself.

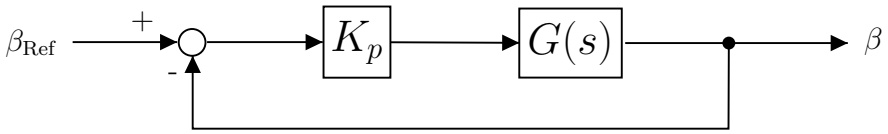


Figure 5.2: Closed loop block diagram of the steering system. It represents a DC Motor, $G(s)$, with negative unit feedback and a proportional controller K_p given β_{Ref}.

The TF of the DC motor can be found in literature, e.g. [17], to be (5.2).

$$G(S) = \frac{\beta}{v_a} = \frac{k_t}{J_m R_a s^2 + (bR_a + k_t k_e)\, s} \tag{5.2}$$

Where:

β	is the angular position of the DC motor	[rad]
J_m	is the inertia of the DC motor	$\left[\text{kg m}^2\right]$
b	is the friction coefficient of the DC motor	[N m s]
v_a	is the armature voltage of the DC motor	[V]
k_t	is the torque gain of the DC motor	$\left[\frac{\text{N m}}{\text{A}}\right]$
k_e	is the electric gain of the DC motor	[V s]
R_a	is the Resistance of the armature	[Ω]
s	is the Laplace operator.	

Once $G(s)$ is known, it is possible to find the transfer function from the angular position reference to the angular position of the DC motor. The closed loop system TF $T_s(s)$ of the system shown in Figure 5.2 is found straightforward as (5.3).

$$T_s(s) = \frac{\beta}{\beta_{\text{Ref}}} = \frac{K_p k_t}{J_a R_a s^2 + (bR_a + k_e k_t)\, s + K_p k_t} \tag{5.3}$$

The implementation of the steering model is shown in Figure 5.3.
The values of the different parameters are not known, except for the proportional gain which is implemented in the LLC and it is known to be $K_p = 0.01$. The rest of the parameters need to be estimated.

5.1.1 Parameter estimation

In Section 5.1, a TF that captures the dynamics of the steering system actuation has been found. The TF for this system is expressed in (5.3) and it includes DC motor actuator dy-

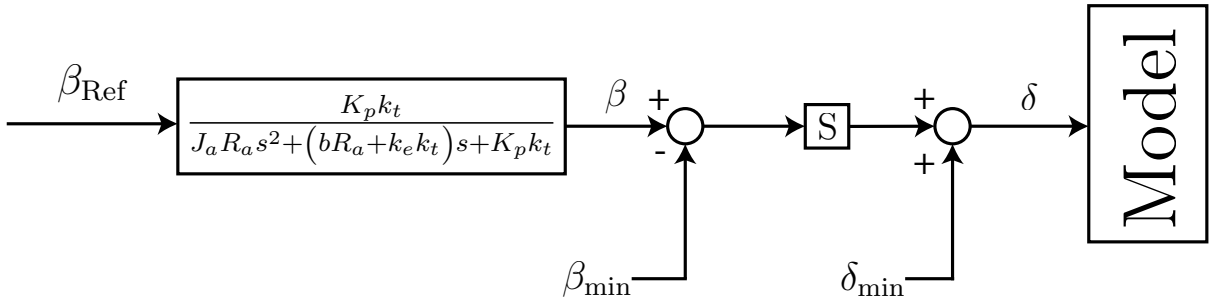


Figure 5.3: Block diagram showing the steering actuation model. The implementation consists of two steps, one mapping from β_{Ref} to β and another using the linear steering ratio to maps from steering rod angle β to to Ackermann steering angle δ.

namics found within a closed loop with a proportional controller. This TF is dependent on some parameters of the DC motor, e.g. motor inertia J_m and the armature resistance R_a among others, and only the proportional gain K_p is known, the rest of the parameters have to be estimated. Recalling (5.3), one can realize that the parameters are coupled and thus cannot be estimated separately. Therefore, the TF is fitted considering it have two poles and no zeros.

To find the parameters, different input references β_{Ref} are given and the response of the actuator is recorded. Given both input and response signals, the TF is fitted using the System Identification Toolbox in MATLAB® which yields the TF shown in (5.4).

$$T_s(s) = \frac{5.609}{s^2 + 3.814s + 5.609} \tag{5.4}$$

The comparison between the measured and the simulated response is shown in Figure 5.4.

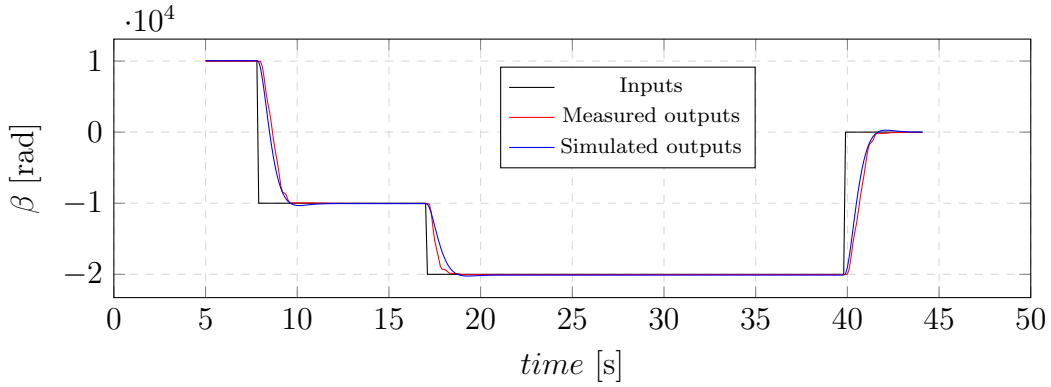


Figure 5.4: Comparison between the measured steering position β and the simulated steer position with the estimated TF given different inputs in an indoor test.

The test used to fit a model of the actuator system has been performed indoors. Its behaviour when used outdoors may change due to frictional differences. Figure 5.5 shows a comparison between the model and the measured values of an outdoor test. This test is performed with the vehicle in motion, and it displays a good fitting. At time $t = 24$s, it can be seen how the model differs to the actual measurements, due to the velocity of the vehicle being too low.
However, when the same test is performed statically, the actuator does not reach the references as Figure 5.6 illustrates. This behaviour needs to taken into consideration in the controller design.

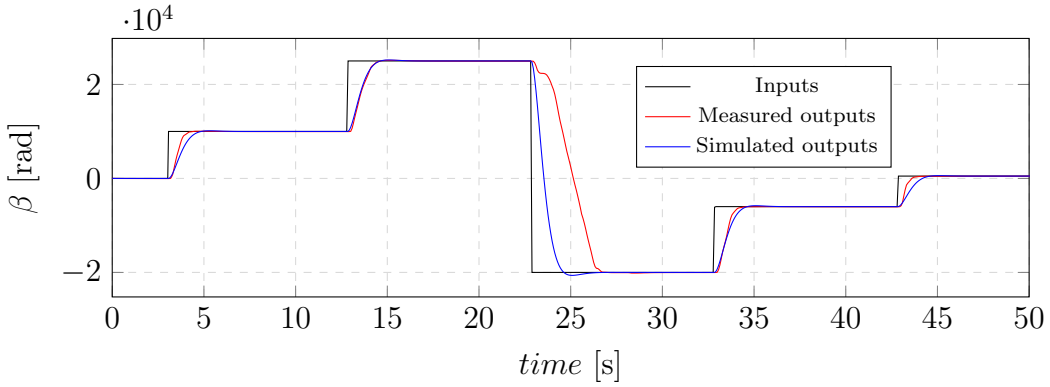


Figure 5.5: Comparison between the measured steering position β and the simulated steer position with the estimated TF given different inputs in a dynamic outdoor test.

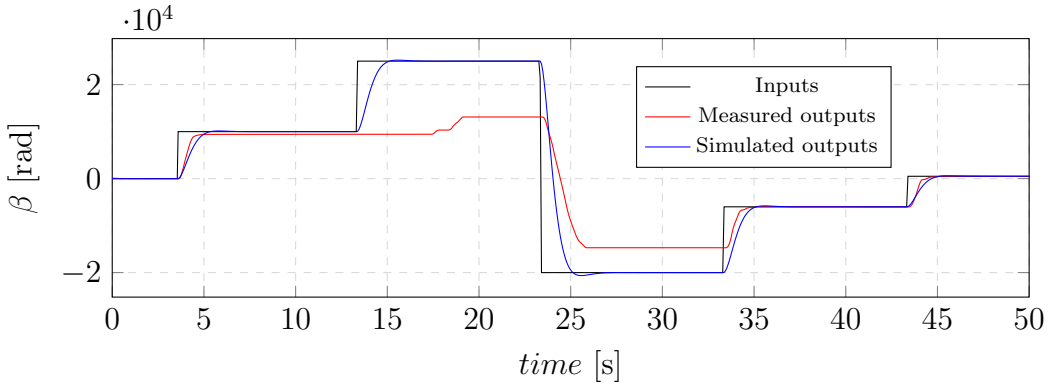


Figure 5.6: Comparison between the measured steering position β and the simulated steer position with the estimated TF given different inputs in a static outdoor test.

5.2 Drive-train system model

In Figure 2.2, it is known that the vehicle is propelled with a DC Motor. When the vehicle is driven manually it gets an input signal from the throttle pedal actuated by the driver. The input signal, generates motor angular velocity references $\omega_{m,\text{Ref}}$ and an embedded motor controller controls the motor such that it follows the references. A block diagram representation of this subsystem is shown in Figure 5.7.

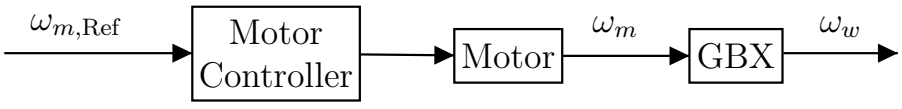


Figure 5.7: Block diagram representing the acceleration system. It includes the motor controller, the DC motor itself and the Gearbox. Only $\omega_{m,\text{Ref}}$ and ω_w can be measured.

The components of the drive-train system are all part of the original set-up and the only available information available about it is the gearbox ratio. Also, only measurements of $\omega_{m,\text{Ref}}$ and ω_w are available so it is not possible to decouple the system in different models. To be able to model the system from $\omega_{m,\text{Ref}}$ to ω_w it is needed to use a black-box model. However, because

the controller structure is not known, it is infeasible to know the order of the system. To find a black-box model for the system different TFs with different combinations of poles and zeros are tried and the one with better fit is chosen. Details of this process are given in Section 5.2.1.

Moreover, to be able to drive the vehicle autonomously the LLC can send $\omega_{m,\text{Ref}}$ references to the embedded controller. Recalling from Section 4.3, motion equations, which are summarized in (4.23), accept steering angle inputs and torque inputs but not $\omega_{m,\text{Ref}}$. For the parameter estimation in Section 5.2.1 mapping from ω_w to T_m is done, while for the controller design in Chapter 6, since the controller is based on the motion equations it relies on T_m inputs, and thus, mapping from T_m to $\omega_{m,\text{Ref}}$ must be considered.

5.2.1 Parameter estimation

In Section 5.2 it has been decided to model the dynamics of the drive-train actuation as a black-box due to uncertainty of its components. It has been mentioned as well that very little information is given and thus it is infeasible to guess the order of the system. In this case the parameter estimation has been done slightly different than before, some inputs $\omega_{m,\text{Ref}}$ has been given to the system and ω_w outputs has been measured and given this data, different TFs have been fitted. The best fitting TF is shown in (5.5), and its response is shown in Figure 5.8.

$$T_{dt} = \frac{2.972}{s^3 + 1.166s^2 + 1.255s + 0.5978} \tag{5.5}$$

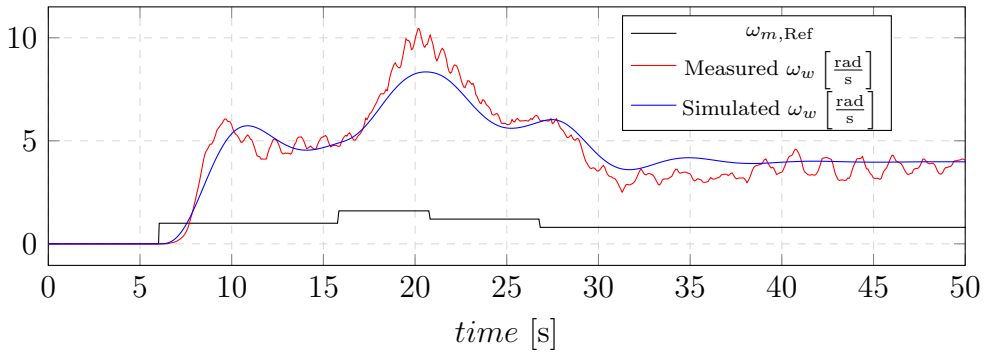


Figure 5.8: Comparison between the measured ω_w and the simulated with the estimated TF $T_{dt}(s)$ given different inputs.

5.3 Braking system model

The vehicle platform is known to be equipped with a braking actuator. A model from the brake's position to torque applied on the wheels is known not to be straightforward. The brake's position could be considered to effect the friction coefficient in the motor dynamics, yet it is not known how the brake position affects the coefficient itself. Nonetheless, from (4.43), it can be seen that it renders a nonlinear model.
Furthermore, the actuator is known to be slow. Two tests are performed to show the differences between the natural braking of the vehicle compared to using the braking actuator. In both tests, A reference of $\omega_{m,\text{Ref}} = 2$ is given to the drive-train system from $t = 4\,\text{s}$ until $t = 22\,\text{s}$. At this time $\omega_{m,\text{Ref}}$ is set to 0 in both tests. In the braking actuator test, its reference is set to 0, i.e.

the maximum value of the brake.
In Figure 5.9, the behaviour of the platform is inspected. In red, longitudinal velocity of the cart is shown when the braking actuator is used. It can be seen how even with the maximum value, the braking time is around 3 s. This behaviour is compared to the one shown by the platform when the brake actuator is not used, shown in blue.

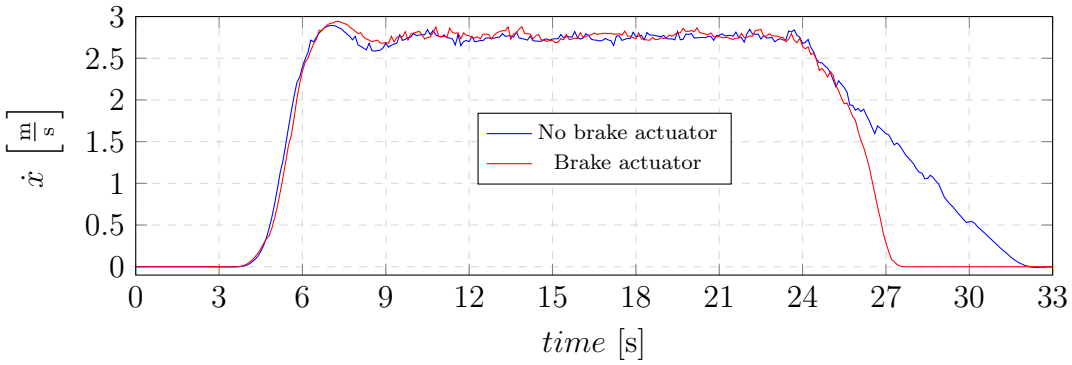


Figure 5.9: Comparison between the measured ω_w during two tests. Measurements in blue show the cart's behaviour when given a zero reference in the motor at $t = 22\,\text{s}$. Measurements in red show the cart's behaviour when given a maximum brake value reference at the same time.

Figure 5.9 shows that there is not a significant time-advantage to use the braking actuator over the natural braking of the cart. For this, it is decided not to derive such a complex braking model and rather to include it in the drive-train model.

5.4 Actuator models conclusion

In this chapter, a model for each actuator system is derived. For the steering actuation, a satisfactory model is found even though it is not applicable when driving on asphalt roads at velocities close to zero. For the accelerator and braking actuation, a black-box model is found based on the drive-train system. Furthermore, it is decided to use the drive-train system for braking.

6 Control structure

This chapter focuses on the structure of the controller to be implemented in the cart. After studying the existing solutions to the objective, a decision on how to address the control solution is taken and the controller is consequently derived and implemented.

First, the control objective is discussed and defined in Section 6.1. Two main decisions to be taken are discussed. In Section 6.1.1, different methods to define the objective are presented and a decision based on the state-of-the-art implementations is taken. In Section 6.1.2, state-of-the-art controllers for the method defined in Section 6.1.1 are exposed, and a decision on the controller design is taken. Afterwards, an exhaustive explanation of the objective definition taken is shown in Section 6.2. In Section 6.3, the controller is designed based on the model derived in Section 4.4.3. Last, the controller is adapted to the existing platform in Section 6.4, based on the actuator models found in Chapter 5. Real-life implementation results of the full control structure are shown in Section 6.5.

6.1 Control objective

As it has been stated in Section 1.2, the goal of the thesis is to be able to move the golf cart from its actual position to a desired location. The objective of the controller is to bring the cart to the end location. However, it is not only desired to reach the desired location but it is of great interest to control how the location is reached. The cart must be able to follow the roads accurately and to adapt its speed under conditions such as crossroads or intersections. Therefore, the specific objective to be controlled needs to be defined. The most common methods for motion control of vehicles are trajectory-tracking and path-following. The two methods and their differences are explained in Section 6.1.1, and a decision based upon them is taken.

Once the approach taken to control the cart's motion has been decided, the design of the controller can be performed. Numerous approaches to control the motion of a cart have been tested. In Section 6.1.2, a few state-of-the-art controllers are mentioned, and the final decision to be implemented is taken based on the existing solutions.

6.1.1 Trajectory-tracking or path-following?

First of all, the definitions of both trajectory and path need to be stated.
A trajectory to be tracked for a vehicle is defined as a time parametrised reference, i.e. a geometric path with an associated timing law. When designing a trajectory-following controller for an underactuated vehicle, it may not be possible to render the zero-dynamics of the system stable [18].
A trajectory-tracking problem can be formulated to bring the reference-tracking error to zero, as defined in (6.1).

$$e_t(t) = y(t) - r(t) \tag{6.1}$$

Where the error, e_t, the output, y, and the trajectory reference, r, are time dependent.

A path, on the other hand, is a reference defined without a temporal law. It can be stated then, that the fundamental difference between trajectory tracking and path following is that the first one focuses on specifying a reference to be tracked at a specific moment in time, whereas the second one focuses on the tracking of the reference with no time-dependency.
Furthermore, as shown in [19], the path-following approach does not have the zero-dynamics instability problem appearing on reference-tracking.
A path-following problem can be formulated the same way as the trajectory-tracking, with a difference on the reference desired to be tracked, as shown in (6.2).

$$e_p(t) = y(t) - \rho(\theta(t)) \tag{6.2}$$

Where the error, e_p, depends on a desired geometric path, ρ, specified with a parameter θ.

In conclusion, the fundamental idea behind both path-following and trajectory-tracking is to achieve a desired output of the system. Trajectory-tracking relies on providing the desired output values at specific times, whereas path-planning relies on providing the desired output values at certain values of a chosen parameter.
Due to the zero-dynamics behaviour, the path-following approach for controlling the motion of the vehicle is chosen to be the one implemented. Therefore, the references provided to the system defined in Section 6.2 should not be given depending on time, but on a parameter.

6.1.2 State-of-the-art controllers

The motion control of a vehicle is a widely studied control problem, and therefore, numerous solutions to it have already been found. When focusing specifically in the path-following problem, a wide range of controller designs are found to exist.

In [13] and [20], path-following controllers based on the Model Predictive Control (MPC) method are shown to follow the desired path correctly under different sets of road circumstances. Nevertheless, the results are presented in a simulation environment.
From other work as [12] and [21], nonlinear controllers have been designed to fulfil the path-following motion. It is seen that these work adequately in simulation but in most cases real implementation results are not shown.

Considering the vast existing solutions to the path-following problem, it is decided for academic purposes to study the behaviour of nonlinear controllers. Furthermore, it is desired to study its behaviour out of a simulation environment, and examine their performance when implemented in the real platform.

6.2 Path-following motion

This section is intended to find the system equations to be controlled. Based on the decision in Section 6.1.1, a path-following approach is taken. The equations defining the path to be followed by the golf cart need to be found.

Some considerations have to be taken before describing these equations.
In order to ease the controller design, the system is found to be divisible into two separate motion objectives that can be described as minimising the lateral error of the vehicle with respect to the desired path, and forcing the vehicle to a desired longitudinal velocity. These two systems are coupled as shown in Section 4.3 both by the Coriolis effect and the forces acting on the front

wheels when these are turned.
The lateral slip ratio becomes small when the vehicle is running at low speeds. The bicycle model simplifications explained in Section 4.4.3 and the tyre model simplifications from Section 4.4.2 are used for easing the controller derivation. It is decided then to consider two different control objectives, one following a desired longitudinal velocity of the vehicle and another one that takes care of the path-following by the means of controlling the lateral dynamics of the vehicle.
Consequently, it is necessary to define the systems deemed to be controlled.

6.2.1 Longitudinal velocity error

The longitudinal velocity control is desired to generate the necessary control input to track a velocity reference profile. The controller to be designed is then desired to bring the error between the velocity profile reference and the longitudinal velocity of the vehicle towards zero. The error is defined in (6.3).

$$e_{\dot{x}} = \dot{x} - \dot{x}_{\text{Ref}} \tag{6.3}$$

$\dot{x}$ is defined as the longitudinal velocity of the cart. However, in order to make the overall controller easier to derive, the longitudinal dynamics of the car need to be simplified. To do so, in addition to the already stated simplifications, the turning of the front wheels is not included in the force balance, and thus, the dynamics become as shown in (6.4).

$$m\ddot{x} = F_{xr} + m\dot{\psi}\dot{y} \tag{6.4}$$

Where, with no slip-ratio, F_{xr} can be found as shown in (4.13), leaving then the dynamic equation as (6.5).

$$\ddot{x}\left(m + \frac{J_\omega}{R_{\text{eff}}^2}\right) = m\dot{\psi}\dot{y} + \frac{T_m}{R_{\text{eff}}} \tag{6.5}$$

When combined with the longitudinal velocity error, the system to be controlled becomes the one shown in (6.6).

$$\begin{cases} e_{\dot{x}} & = \dot{x} - \dot{x}_{\text{Ref}} \\ \ddot{x} & = \frac{m}{m_e}\dot{\psi}\dot{y} + \frac{1}{R_{\text{eff}} m_e} T_m \end{cases} \tag{6.6}$$

Where m_e is equal to $m + \frac{J_\omega}{R_{\text{eff}}^2}$.

Recalling from the path-following approach taken, it is clear that the reference to the system, $\dot{x}_{\text{Ref}}$, must not be time-dependent, but it should be provided based on a parameter instead. This parametrisation is further explained in Section 6.2.3.

6.2.2 Lateral dynamics error

In order to force the cart to be following a specified path, the lateral dynamics error between the cart position and the path needs to be defined. Two different errors are considered to do so; one being the orientation error and the other being the lateral error.

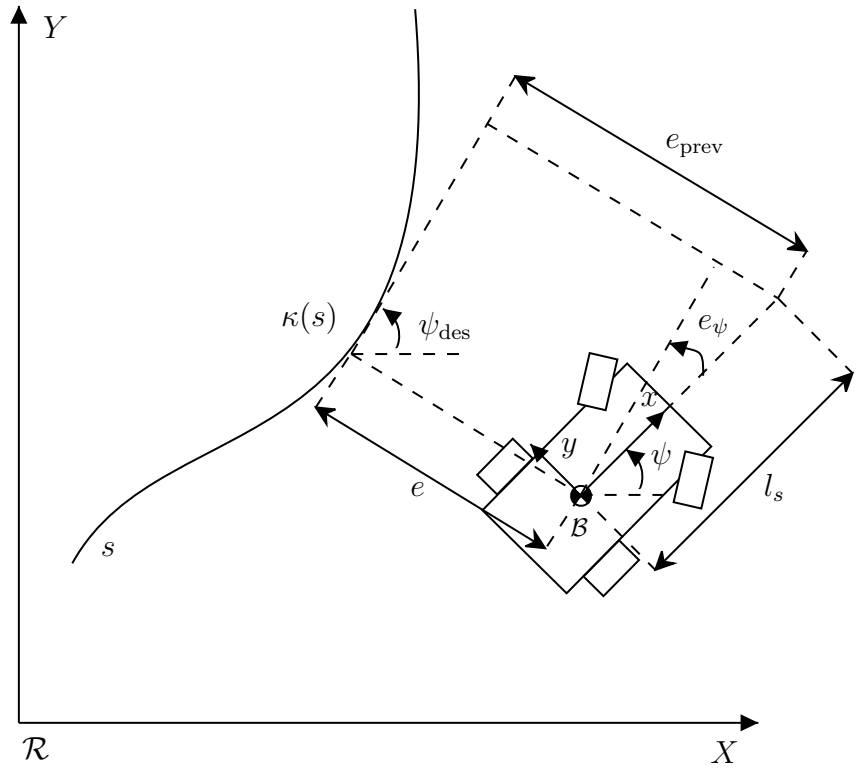


Figure 6.1: Path-following errors description.

Following the Serret-Frenet equations as in [22] and [23], the problem of defining the distance and orientation error of a body from a desired path can be stated as in (6.7). These equations are derived assuming that the curvature is not too curved (bounded from above) and that the vehicle does not deviate too far from the path (local parametrisation) [24]. Although it is stated in [22] that defining the orientation error, e_ψ, as such may cause trouble, it is considered to be the angle difference between the orientation of the cart and the tangent line of the path. The lateral error, e, is considered to be the closest lateral distance difference between the path and the cart. These errors are shown in Figure 6.1.

$$\begin{cases} \dot{e} & = \dot{x}\sin e_\psi + \dot{y}\cos e_\psi \\ \dot{e}_\psi & = \dot{\psi} - \kappa\dot{x} \end{cases} \tag{6.7}$$

$\dot{x}$ and $\dot{y}$ denote the longitudinal and lateral velocities of the cart respectively and κ denotes the curvature of the path.
As stated in [22], the preview control is a widely used practice, which consists of adding the preview lateral error to the previously defined lateral error. This is shown in (6.8).

$$\begin{cases} e_{\text{prev}} & = l_s \sin e_\psi \\ e_y & = e + e_{\text{prev}} \end{cases} \tag{6.8}$$

Assuming that the vehicle is close to being on the path, and consequently, that the difference orientation error, e_ψ, is small; taking the small angle approximation such that $\sin\beta \approx \beta$ and $\cos\beta \approx 1$ as $\beta \to 0$, the error dynamics of the path-following approach become the ones shown in (6.9).

$$\begin{cases} \dot{e}_y &= \dot{x}e_\psi + \dot{y} + l_s e_\psi \\ \dot{e}_\psi &= \dot{\psi} - \kappa\dot{x} \end{cases} \tag{6.9}$$

If the lateral dynamic equations for $\dot{y}$ and $\dot{\psi}$ are included into the error dynamics, taking the small angle approximation of δ, the system to be controlled becomes the one shown in (6.10).

$$\begin{cases} \dot{e}_y &= \dot{x}e_\psi + \dot{y} + l_s e_\psi \\ \dot{e}_\psi &= \dot{\psi} - \kappa\dot{x} \\ \ddot{y} &= -\frac{C_{\alpha r}+C_{\alpha f}}{m\dot{x}}\dot{y} + \frac{l_r C_{\alpha r} - l_f C_{\alpha f}}{m\dot{x}}\dot{\psi} - \dot{x}\dot{\psi} + \frac{C_{\alpha f}}{m}\delta \\ \ddot{\psi} &= \frac{l_r C_{\alpha r} - l_f C_{\alpha f}}{I_z\dot{x}}\dot{y} - \frac{l_r^2 C_{\alpha r} + l_f^2 C_{\alpha f}}{I_z\dot{x}}\dot{\psi} + \frac{l_f C_{\alpha f}}{I_z}\delta \end{cases} \tag{6.10}$$

Again, considering the approach taken to control the motion of the vehicle, the reference to the lateral dynamics, κ, should be provided by the value of a parameter and not based on a time-law.

6.2.3 Path parametrisation

To provide parameter-dependent references for the path-following objective, it is needed to define the parameter to be used.
Based on the work presented in [22], the parameter used to define the references to the system is decided to be the distance along the path from the initial position. First, the curvilinear coordinate of a point in the path needs to be introduced. Following [22], the coordinate of a point along the path is given by (6.11).

$$\dot{s} = \frac{1}{1 - e\,\kappa}\left(\dot{x}\cos e_\psi - \dot{y}\sin e_\psi\right) \tag{6.11}$$

Therefore, the distance along the path can be found by integrating this value, where the curvature at the point, κ, depends on the distance s.
An example of references given based on the distance along the path is shown in Figure 6.3, where the path being described is shown in Figure 6.2. Although the velocity references are not shown in the path description, they must be specified according to the path-following method, and thus, depending on s, as shown in Figure 6.4.

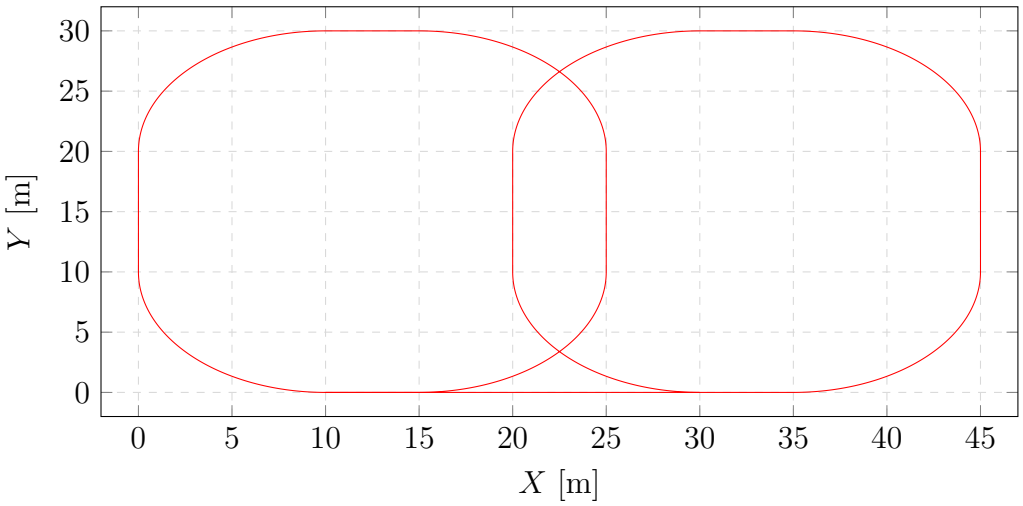


Figure 6.2: 2D path representation given parameter-dependent curvature references.

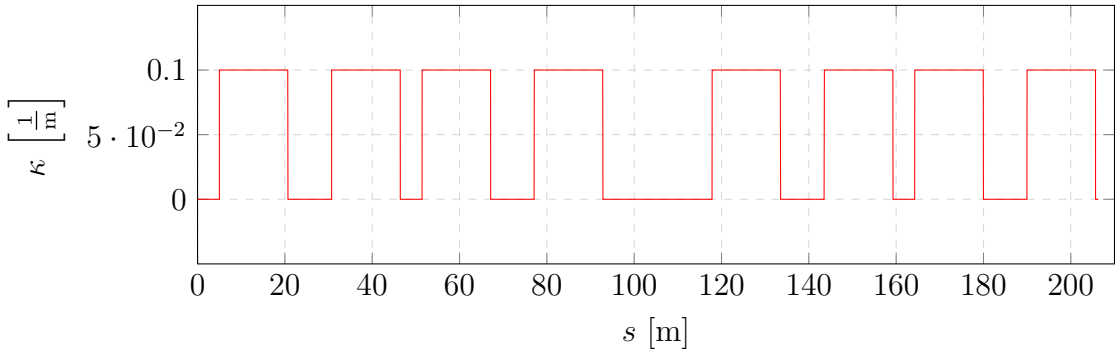


Figure 6.3: Curvature definition based on the distance along the path.

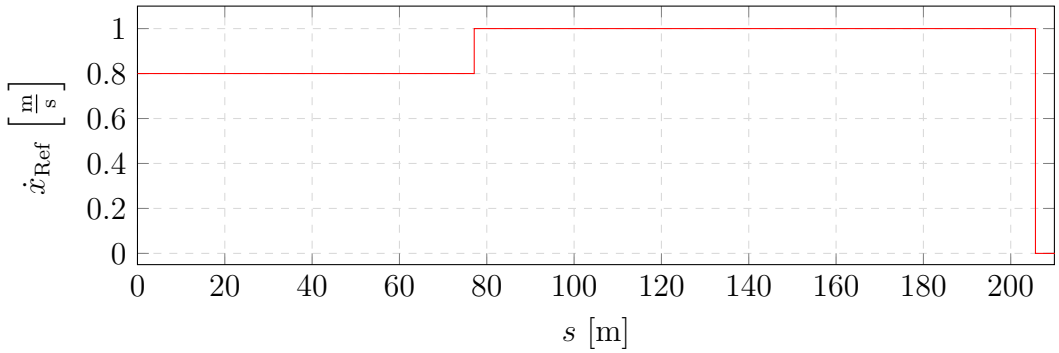


Figure 6.4: Velocity references based on the distance along the path.

6.3 Controller design

Since two separate systems in the error dynamics derivation have been assumed, the controllers for the lateral errors and the longitudinal velocity error are designed separately.

6.3.1 Longitudinal velocity controller

Examining the dynamic equations desired to control exposed in (6.6), and based on the decision from Section 6.1.2, it is decided to implement a nonlinear control law for the input of the system T_m to bring the longitudinal velocity error asymptotically towards 0.

It is known, however, that there is uncertainty in some of the parameters in the model since these will change in different situations. To give an example, the mass of the cart and consequently its inertia and distances from the front and rear axes to G, depend on the number of passengers being carried. Therefore, the designed controller needs to take the deviation of the parameter from the nominal values found in the parameter estimation into account.

It is decided to design the longitudinal velocity controller using the Lyapunov redesign method as explained in [25].
The derivation of the controller is explained in detail in Appendix B.1. In the following, the

outcome of the derivation is shown, yielding the control law to be used.

A control law for a nominal system (6.12) needs to be derived, where the denotation $\hat{\bullet}$ describes the nominal value of the parameters. This derivation is further described in Appendix B.1.

$$\begin{cases} e_{\dot{x}} &= \dot{x} - \dot{x}_{\text{Ref}} \\ \ddot{x} &= \frac{\hat{m}}{\hat{m}_e}\dot{\psi}\dot{y} + \frac{1}{\hat{R}_{\text{eff}}\hat{m}_e}T_m = \hat{a}\dot{\psi}\dot{y} + \hat{b}T_m \end{cases} \tag{6.12}$$

Taking the nominal parameters found in Section 4.5, the nominal control law is designed as (6.13).

$$\begin{cases} \frac{1}{\hat{R}_{\text{eff}}\hat{m}_e}T_m = & -ke_{\dot{x}} - \frac{\hat{m}}{\hat{m}_e}\dot{\psi}\dot{y} + \ddot{x}_{\text{Ref}} \\ 0.0056\,T_m = & -ke_{\dot{x}} - 0.6676\,\dot{\psi}\dot{y} + \ddot{x}_{\text{Ref}} \end{cases} \tag{6.13}$$

The controller gain k is chosen to be $k = 0.8$.

The control law found for the nominal system is then redesigned to correct for the parameter changes. In Appendix B.1, the uncertainty in the parameters is found to be bounded by (6.14).

$$\begin{aligned} |\delta| &\leq \left|\frac{1}{\hat{b}}\right|\left[\left|\frac{a\hat{b} - b\hat{a}}{\hat{b}}\right| \|\dot{y}\|_2 \left\|\dot{\psi}\right\|_2 + \left|\frac{\hat{b}-b}{\hat{b}}\right| k\,\|e_{\dot{x}}\|_2 + \left|\frac{b-\hat{b}}{\hat{b}}\right| \|\ddot{x}_{\text{Ref}}\|_2\right] + \left|\frac{b-\hat{b}}{\hat{b}}\right| \|v\|_2 \\ |\delta| &\leq \rho(x) + \kappa_0 \|v\|_2 \end{aligned} \tag{6.14}$$

Considering reasonable intervals for the values that the parameters can take it is possible to find a function $\rho(x)$ and a constant κ_0.
In Table 6.1, minimum and maximum values for the parameters in the system are shown. The values have been taken accordingly to what can be expected from the real system. The mass of the golf cart is considered to vary from carrying no passengers to carrying two passengers, and to vary depending on the water level of the battery. The radius of the wheels is considered to decrease from its standard value. The inertia of the wheels is approximated from the other intervals as well as m_e.

Parameter	Minimum value	Nominal value	Maximum value	Unit
m	392	586	652	kg
R_{eff}	0.17	0.2032	0.2032	m
J_w	8	12.046	14	$\text{kg}\,\text{m}^2$
m_e	585.75	877.73	1136.4	kg

Table 6.1: Minimum, nominal and maximum values for the model parameters.

From these values, $\rho(x)$ and κ_0 are defined as in (6.15).

$$\begin{aligned} \rho(x) &= 178.3567\left[0.5\,\|\dot{y}\|_2 \left\|\dot{\psi}\right\|_2 + 0.6\,k\,\|e_{\dot{x}}\|_2 + 0.6\,\|\ddot{x}_{\text{Ref}}\|_2\right] \\ \kappa_0 &= 0.6 \end{aligned} \tag{6.15}$$

Now the continuous control law, v, is designed according to the bound found as shown in (6.16).

$$v = \begin{cases} -\eta(x)\frac{w}{\|w\|_2}, & if \quad \eta(x)\,\|w\|_2 \geq \varepsilon \\ -\eta^2(x)\frac{w}{\varepsilon}, & if \quad \eta(x)\,\|w\|_2 < \varepsilon \end{cases} \tag{6.16}$$

Where the non-negative function $\eta(x)$ is chosen to be $\eta(x) = \frac{\rho(x)}{1-\kappa_0}$ and the variable w is defined following Appendix B.1 as $w = e_{\dot{x}}\hat{b} = 0.0056\, e_{\dot{x}}$.

If an upper bound, r, for the error in the velocity , $e_{\dot{x}}$, is taken to be $4\frac{\mathrm{m}}{\mathrm{s}}$, then $\varepsilon = 0.01 < 5$ is taken as the value to be used for the control strategy v, following the equations found in Appendix B.1. This value for ε gives a boundary for the output of $b(\varepsilon) < 0.18$.

The performance of the longitudinal controller is shown in the following.
A model simulation with torque and steer inputs is used. The parameters used by the model equations are randomised such that the full performance of the controller is tested. While the longitudinal controller is being used, the cart is forced to use the lateral controller derived in Section 6.3.2 such that the longitudinal controller is not tested with the cart driving in straight lines.
In Figure 6.5, it is shown how the longitudinal velocity of the cart is driven towards the reference provided by the algorithm shown in Figure 6.4.

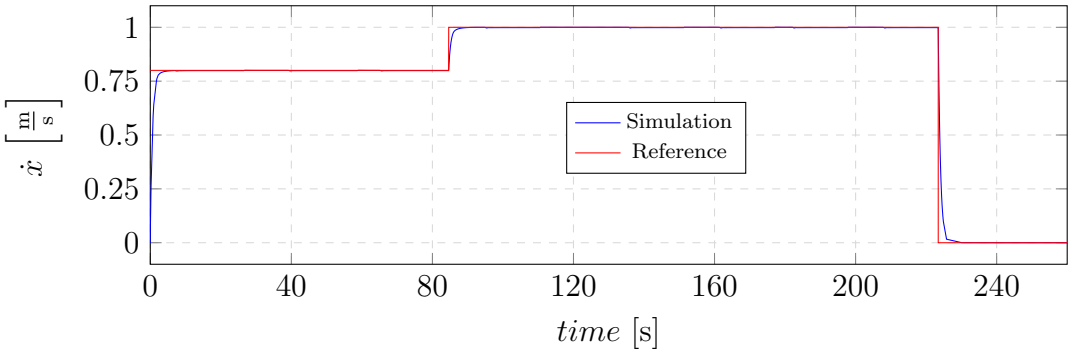


Figure 6.5: Longitudinal controller performance in simulation.

In Figure 6.6, the response of the longitudinal controller with the nominal parameters is shown. A series of Monte Carlo simulations are run such that the response of the controller to randomised model parameters can be examined. Figure 6.6 also shows the area comprising all of the simulated responses. During these simulations the cart is forced to perform turns such that the full capabilities of the controller are tested. It can be seen how, with different settling times, all of the run simulations reach the desired velocity reference.

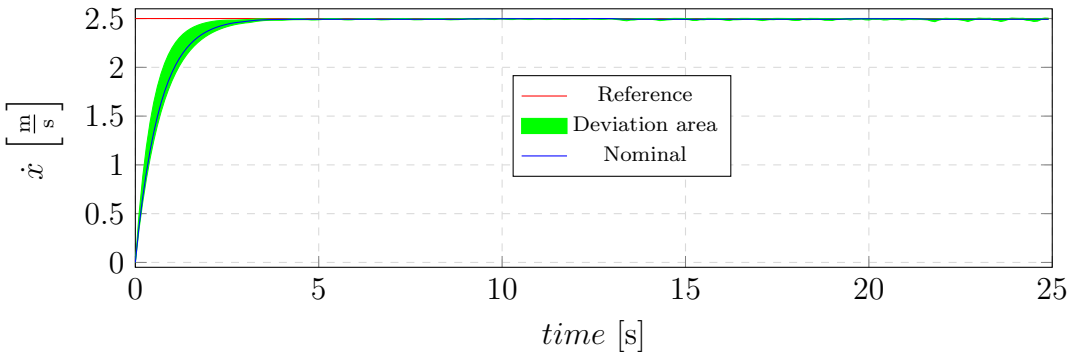


Figure 6.6: Longitudinal controller performance in simulation. Performance of the controller with the nominal parameters compared to a series of Monte Carlo simulations with varying parameters.

These results can be unrealistic, since a torque input to the system can not be provided directly to the drive-train system. Being able to replicate the presented controller results depends then on the dynamics of the drive-train system. However, if the actuation on the cart was achieved to be fast enough, the results presented in Figure 6.5 could be achieved.

6.3.2 Lateral dynamics controller

From the awareness of the nonlinearities in the cart model and following the decision taken in Section 6.1.2, it is chosen to use a nonlinear controller for the lateral dynamics of the cart. Examining the lateral dynamics derived in Section 6.2.2, the objective is to drive the lateral errors from (6.10) towards zero by manoeuvring the input, δ. It can be seen from the equations how δ does not affect directly the errors described in (6.10). Therefore, out of the nonlinear control possibilities, it is deemed interesting to use the backstepping approach.

A full derivation of the backstepping controller used is shown in Appendix B.2, based on the approach used in [25]. In Appendix B.2 it is concluded, however, that it is not possible to drive the two errors describing the lateral dynamics to zero. The approach taken, then, uses the equations shown in (6.17), (6.18) and (6.19), and consists in driving the lateral error e_y towards zero by controlling it through the heading error e_ψ. Accordingly, the heading error is brought to its desired value by controlling the heading rate, $\dot{\psi}$, which is in turn brought to its desired value with the true input to the system, δ.

$$\dot{e}_y = \dot{x}e_\psi + \dot{y} + l_s e_\psi \tag{6.17}$$

$$\dot{e}_\psi = \dot{\psi} - \kappa\dot{x} \tag{6.18}$$

$$\ddot{\psi} = \frac{l_r C_{\alpha r} - l_f C_{\alpha f}}{I_z \dot{x}}\dot{y} - \frac{l_r^2 C_{\alpha r} + l_f^2 C_{\alpha f}}{I_z \dot{x}}\dot{\psi} + \frac{l_f C_{\alpha f}}{I_z}\delta \tag{6.19}$$

In the following, the outcome of the derivation is shown, yielding the control law desired for the lateral dynamics of the golf cart.

Taking first (6.17), the heading error e_ψ is considered as an input to stabilize e_y. It is shown in Appendix B.2 how taking the control law (6.20) guarantees the system to be asymptotically stable at the origin.

$$e_\psi \to \phi_1 = \frac{1}{\dot{x} + l_s}\left(-k_1 e_y - \dot{y}\right) \tag{6.20}$$

Now, a new variable, $z_1 = \phi_1 - e_\psi$, is introduced such that it expresses the difference between the desired value of the heading error, ϕ_1, and its true value, e_ψ. The first subsystem used, (6.17), is then left to be (6.21).

$$\dot{e}_y = -k_1 e_y - (\dot{x} + l_s) z_1 \tag{6.21}$$

The second system to be stabilised is now the one described by z_1, which is deemed to be brought to zero. It is shown again in Appendix B.2 that the subsystem (6.22) is asymptotically stable at the origin if the control law for $\dot{\psi}$ is taken as shown in (6.23).

$$\dot{z}_1 = \frac{\partial \phi_1}{\partial t} - \dot{e}_\psi = \frac{\partial \phi_1}{\partial t} + \kappa \dot{x} - \dot{\psi} \tag{6.22}$$

$$\dot{\psi} \to \phi_2 = \frac{\partial \phi_1}{\partial t} + \kappa \dot{x} + k_2 z_1 - (\dot{x} + l_s) e_y \tag{6.23}$$

Where the expression $\frac{\partial \phi_1}{\partial t}$ denotes the time derivative of ϕ_1.

Finally, a new variable, $z_2 = \phi_2 - \dot{\psi}$, is introduced such that it expresses the difference between the desired value for the heading rate, ϕ_2, and its true value, $\dot{\psi}$. The system defined in (6.22) changes then to be as shown in (6.24).

$$\dot{z}_1 = -k_2 z_1 + z_2 + (\dot{x} + l_s) e_y \tag{6.24}$$

The newly introduced variable z_2 is deemed to be brought to zero such that the heading rate takes the necessary values to stabilise the lateral error. Considering then the dynamics of it shown in (6.25), the system is found to be asymptotically stable at the origin, i.e. $e_y = 0$, $z_1 = 0$ and $z_2 = 0$, when applying the control law shown in (6.26) to the true input to the system, the steering angle δ.

$$\begin{aligned} \dot{z}_2 &= \frac{\partial \phi_2}{\partial t} - \ddot{\psi} \\ &= \frac{\partial \phi_2}{\partial t} - \left(\frac{l_r C_{\alpha r} - l_f C_{\alpha f}}{I_z \dot{x}} \dot{y} - \frac{l_r^2 C_{\alpha r} + l_f^2 C_{\alpha f}}{I_z \dot{x}} \dot{\psi} + \frac{l_f C_{\alpha f}}{I_z} \delta \right) \\ &= \frac{\partial \phi_2}{\partial t} - a_1 \dot{y} - a_2 \dot{\psi} - b\delta \end{aligned} \tag{6.25}$$

$$\delta = \frac{1}{b}\left(\frac{\partial \phi_2}{\partial t} - a_1 \dot{y} - a_2 \dot{\psi} + k_3 z_2 + z_1 \right) \tag{6.26}$$

In order to define the values to be taken for the controller gains k_1, k_2 and k_3, the closed loop system shown in (6.27) has to be found to be Hurwitz.

$$\begin{bmatrix} \dot{e}_y \\ \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} -k1 & -(\dot{x} + l_s) & 0 \\ (\dot{x} + l_s) & -k_2 & 1 \\ 0 & -1 & -k_3 \end{bmatrix} \begin{bmatrix} e_y \\ z_1 \\ z_2 \end{bmatrix} \tag{6.27}$$

Since the closed loop system depends on the longitudinal velocity of the cart, $\dot{x}$, the operating range of it needs to be taken into account in the design. As stated in Chapter 4, the cart is not

considered to exceed 15 $\frac{\text{km}}{\text{h}}$.
Satisfactory results for the lateral controller are found using $k_1 = 8$, $k_2 = 10$ and $k_3 = 10$. These are shown in Figure 6.7, where in a simulation environment, the lateral controller derived is tested.
The curvature input to the lateral controller is parameter-based, as shown in Figure 6.3. The curvature reference is dependent on the path distance. This curvature references define the path reference expected shown in Figure 6.2.

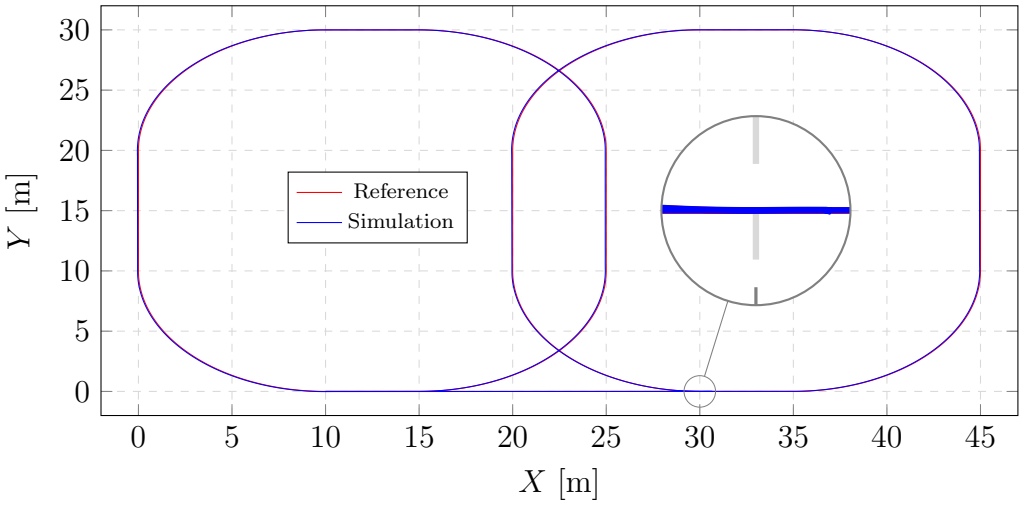


Figure 6.7: Lateral controller performance in simulation.

6.4 Cascaded structure for cart actuators

Since, as described in Section 2.2, the cart has actuators implemented with their own dynamics, it is not possible to implement the controllers derived in Section 6.3 such that the model is provided the desired values for torque and steer inputs straightforwardly.
The longitudinal subsystem acts on the cart through an angular velocity reference that provides the wheels with an angular velocity, which relates to the torque as described in Section 4.5 for the parameter estimation. The lateral subsystem acts on the cart through a steering wheel reference that is followed, which relates to the steer value as described in Section 5.1.
Therefore, a new approach is taken to guarantee that the desired inputs given by the longitudinal and lateral controllers are actuating on the cart.
This is performed in a different way for the longitudinal subsystem than for the lateral one.

6.4.1 Longitudinal subsystem

The modelled longitudinal subsystem of the cart takes as input the torque provided to its wheels. The actuator model for this subsystem is described in Section 5.2, which shows the relation between the DC motor input and the angular velocity of the wheels.
In Section 4.5, the conversion from angular velocity to torque has already been defined.

The desired torque value coming from the longitudinal controller has been designed taking into account the uncertainty in the model parameters, and thus, adapting it to give rotational speed references would require to reverse some important well-functioning work.

Instead of adding the actuator model in the nonlinear control design, it is decided to design a cascaded controller that, using the actuator model found for the drive-train system, forces it to provide the cart with the torque desired from the longitudinal controller.
This structure is shown in Figure 6.8.

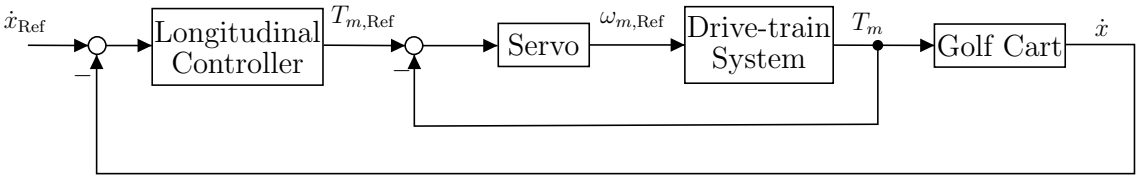


Figure 6.8: Cascaded structure of the longitudinal subsystem.

It is known that for the system to be stable, the cascaded controller needs to be faster than the longitudinal controller, such that the desired torque is being applied to the system.
When the DC-motor dynamics are combined with the drive-train system found in Section 5.2, a transfer function from motor reference to applied torque can be found to be (6.28).

$$G(s) = \frac{127s + 0.4159}{s^3 + 1.166s^2 + 1.255s + 0.5978} \tag{6.28}$$

The servo controller is decided to be designed as a PID for the simplicity presented when tuning it. Its objective is to bring the error between the desired torque and the one applied towards zero in steady state.

When examining the system response using a PID controller, it is found how a fast tracking response of the torque requires high gains in the controller. This is due to the response presented by the full system (6.28), which is shown in Figure 6.9.

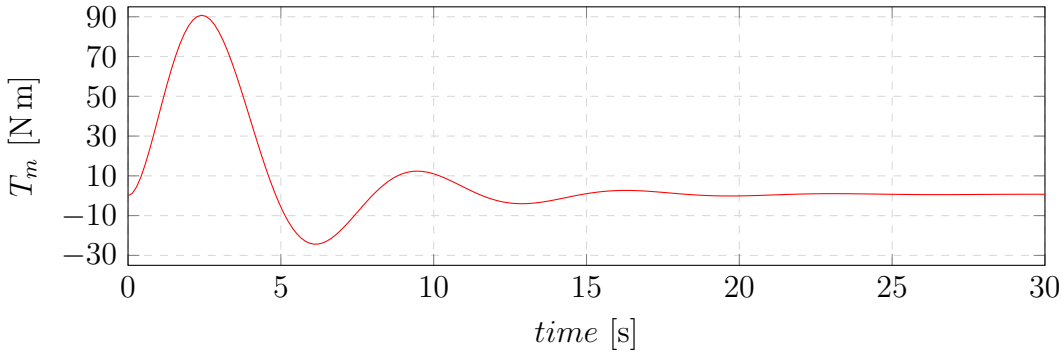


Figure 6.9: Step response of the drive-train system, from input $\omega_{m,\text{Ref}}$ to torque.

This is not desired, since it makes the real input to the cart, $\omega_{m,\text{Ref}}$, to change abruptly. Furthermore, it requires $\omega_{m,\text{Ref}}$ to take large values as well as negative ones. High values of it are not desired for safety reasons, and negative ones activate the backward-driving mode, which is not feasible in the set up. Therefore, this controller needs to be designed considering an input saturation.
Consequently, it is decided to design the controller with a trade-off between its gains and velocity until the system is stable when the saturation of the input is implemented.
The gains found for the PID controller are shown in Table 6.2.

P	I	D
0.03	0.0235	0.0105

Table 6.2: PID gains for the servo controller.

The results of the cascaded structure for the longitudinal subsystem are shown in Figure 6.10, where the simulation of the golf cart uses the actuator models instead of torque and steer inputs. It can be appreciated in Figure 6.10, how the desired values for the velocity are not reached with high precision. As mentioned, the servo controller is designed to be slow, causing the torque applied to follow the desired one slowly, yielding this behaviour.

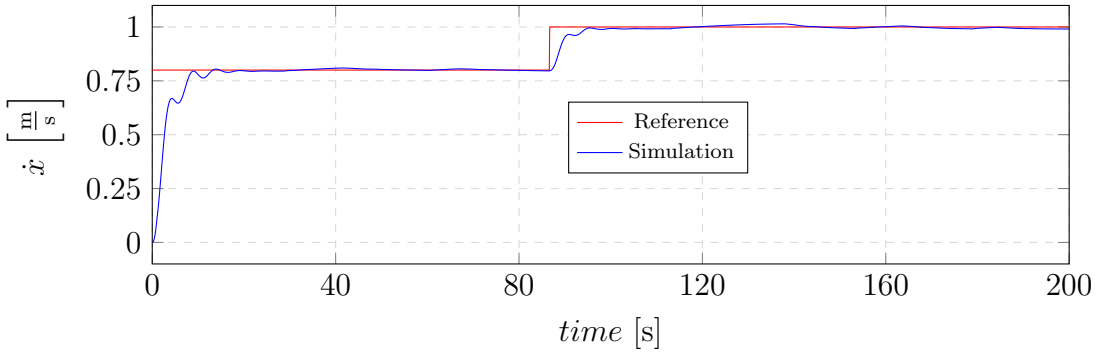


Figure 6.10: Longitudinal velocity performance in simulation.

In Figure 6.11, the applied torque to the model is compared to the desired one, coming from the lateral controller.

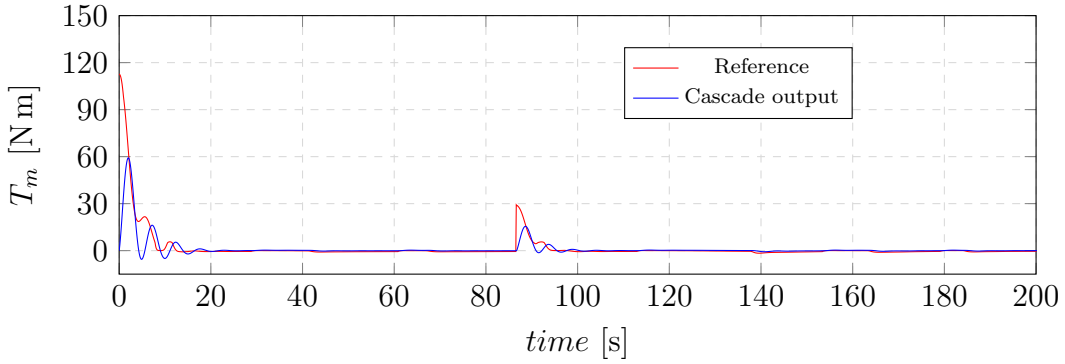


Figure 6.11: Comparison between desired torque and implemented torque.

As stated in Section 4.5, there are no measurements of torque available, and neither are there measurements for the angular acceleration, $\dot{\omega}_m$. Therefore, an estimate of their values based on measurements of the angular velocity of the wheels has been implemented.
Following the solution in Section 4.5 for the torque input, the angular velocity of the wheels can be extracted from the wheel encoders. Knowing this value and the DC motor dynamics from (4.43), the values of the rotational acceleration of the wheels are needed to find the applied torque.
It is decided to calculate the torque by differentiating the rotational speed and using the known

relation between them. Due to the noise introduced by the differentiation, a simple low-pass filter is implemented such that the torque estimate is smoother.
The filter used is shown in (6.29).

$$T_{filtered} = \frac{1}{10s+1}T \tag{6.29}$$

In the real platform, the filter is implemented with its discrete version, running at the same sample rate that the controllers use, i.e. 200Hz.

6.4.2 Lateral subsystem

The dynamics of the lateral subsystem are fast, which when designing a cascaded structure as described for the longitudinal subsystem in Section 6.4.1 forces a need for particularly high gains, leading to simulated instability. Furthermore, the backstepping approach is similar to what a cascaded control structure represents, and can be easily adapted to include dynamics in the input such as the actuator model shows.
A different approach to the one presented for the longitudinal subsystem is taken. In the backstepping design used for the lateral controller it is decided to include the steering actuator dynamics. In this way, the controller would provide the system with steering wheel references, β_{Ref} instead of steering angle values, δ.

This solution is indeed a cascaded controller without integral action, but it allows to design both the previous controller gains of the system and the new ones at once.
Therefore, the structure of the system is left as shown in Figure 6.12.

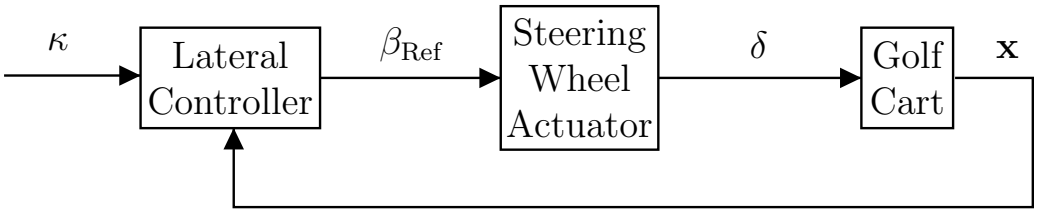


Figure 6.12: Structure of the full lateral subsystem.

Retaking the lateral backstepping controller at the latest point, a desired steer value for the cart, δ, is found.
The first step is then to find the conversion from steer value to steering wheel position. This conversion has been found in Section 5.1, and more specifically, the conversion from steering wheel to steer value is expressed in (5.1).

However, it is convenient to ensure that this conversion maps $\beta = 0$ to $\delta = 0$. Hence, before every experiment, a procedure is performed to center the steering wheel encoder at the corresponding zero steer. The conversion from one to the other is then (6.30).

$$\delta = \frac{\delta_{\max} - \delta_{\min}}{\beta_{\max} - \beta_{\min}}\beta = S\,\beta \tag{6.30}$$

Since the steer actuator model presents a second order system, two more equations are added in the system to be controlled that map from steering wheel reference to steer value. These are shown in (6.31).

$$\begin{bmatrix} \ddot{\beta} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -3.814 & -5.609 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\beta} \\ \beta \end{bmatrix} + \begin{bmatrix} 5.609 \\ 0 \end{bmatrix} \beta_{\text{Ref}} \tag{6.31}$$

The fact that the input reference only affects one of the states allows the use of the backstepping procedure, avoiding the problem described in Appendix B.2.

To proceed with the controller design, it is retaken from the definition of z_2 in (6.25).
The expression found for δ in (6.23) is now considered a desired value for it, as performed before with the heading error, e_ψ, and the heading rate, $\dot{\psi}$.
Therefore, the desired value of δ, is considered as a new function ϕ_3, although it is decided to include the gain to convert it to a desired steering wheel value. This is shown in (6.32).

$$\beta \to \phi_3 = \frac{1}{S\,b}\left(\frac{\partial \phi_2}{\partial t} - a_1\dot{y} - a_2\dot{\psi} + k_3 z_2 + z_1\right) \tag{6.32}$$

Introducing then a new variable defining the error between the desired value for β and its true value, $z_3 = \phi_3 - \beta$, leaves the subsystem (6.25) as shown in (6.33).

$$\begin{aligned} \dot{z}_2 &= \frac{\partial \phi_2}{\partial t} - a_1\dot{y} - a_2\dot{\psi} - b\delta \\ &= \frac{\partial \phi_2}{\partial t} - a_1\dot{y} - a_2\dot{\psi} - b\left[S\left(\phi_3 - z_3\right)\right] \\ &= -k_3 z_2 - z_1 + S z_3 \end{aligned} \tag{6.33}$$

Now, the system to be stabilized in zero is described by z_3. Shown in (6.34), it is stabilised in zero when the virtual input $\dot{\beta}$, is taken as shown in (6.35).

$$\dot{z}_3 = \frac{\partial \phi_3}{\partial t} - \dot{\beta} \tag{6.34}$$

$$\dot{\beta} \to \phi_4 = \frac{\partial \phi_3}{\partial t} + k_4 z_3 + S\,b\,z_2 \tag{6.35}$$

Again, a new variable $z_4 = \phi_4 - \dot{\beta}$ is introduced describing the difference between the desired value for $\dot{\beta}$ and its true value. The subsystem (6.34) is left as (6.36).

$$\dot{z}_3 = \frac{\partial \phi_3}{\partial t} - \dot{\beta} = \frac{\partial \phi_3}{\partial t} - \phi_4 + z_4 \tag{6.36}$$

The system to be stabilised again at zero is defined by z_4. The system (6.37), extracted from (6.31), is deemed to be stable when the control input is taken as (6.38).

$$\dot{z}_4 = \frac{\partial \phi_4}{\partial t} - \ddot{\beta} = \frac{\partial \phi_4}{\partial t} - \left(-3.814\dot{\beta} - 5.609\beta + 5.609\beta_{\text{Ref}}\right) \tag{6.37}$$

$$\beta_{\text{Ref}} = \frac{1}{5.609}\left(\frac{\partial \phi_4}{\partial t} + 3.814\dot{\beta} + 5.609\beta + k_5 z_4 + z_3\right) \tag{6.38}$$

Finally, the controller gains defined along the derivation need to be designed. This is done by inspecting the closed loop system in (6.39).

$$\begin{bmatrix} \dot{e}_y \\ \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \dot{z}_4 \end{bmatrix} = \begin{bmatrix} -k_1 & -(\dot{x} + l_s) & 0 & 0 & 0 \\ (\dot{x} + l_s) & -k_2 & 1 & 0 & 0 \\ 0 & -1 & -k_3 & b\,S & 0 \\ 0 & 0 & -b\,S & -k_4 & 1 \\ 0 & 0 & 0 & -1 & -k_5 \end{bmatrix} \begin{bmatrix} e_y \\ z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} \tag{6.39}$$

The gains need to be designed such that the closed-loop is Hurwitz, but moreover, it is necessary that the subsystem defining the steering wheel is faster. This is due to the cascaded-like structure of the backstepping.
The choice of gains shown in Table 6.3 shows good simulation results.

k_1	k_2	k_3	k_4	k_5
8	8	10	50	80

Table 6.3: Controller gains for the lateral subsystem.

With the stated gains, the results shown in simulation are shown in Figure 6.13. The desired path to be followed is the same shown in Figure 6.3. The model uses the actuator models described.

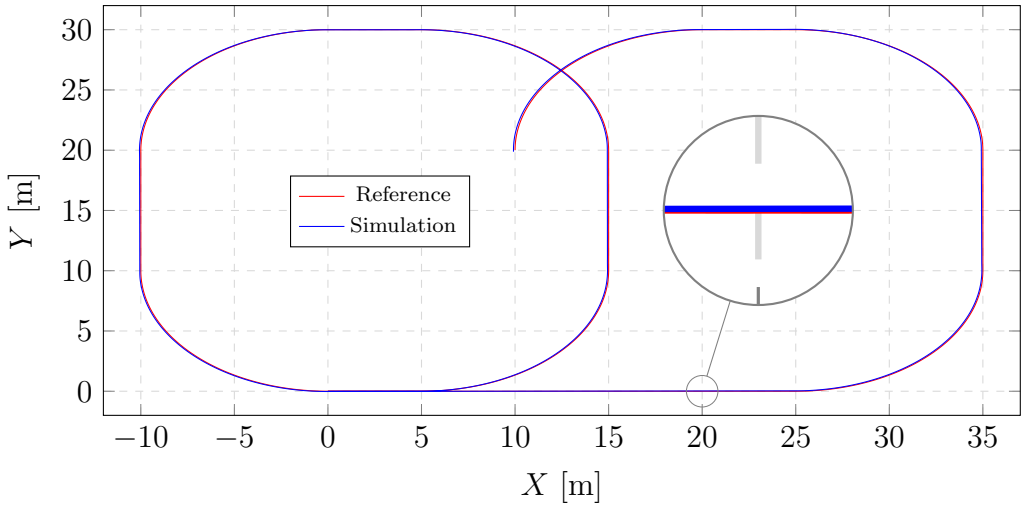


Figure 6.13: Comparison between desired path and simulation.

In Figure 6.14 the desired delta inside the backstepping controller is shown to be tracked by using the steer actuator.
However, to implement the backstepping controller it is needed to have estimates of the steering wheel position, β and its time derivative, $\dot{\beta}$. The steering wheel position value is known since it is given by the platform. In order to get estimates for these two variables, an observer is designed based on the actuator model. This is done by implementing a Kalman filter based on the state-space representation of the transfer function in (5.4), shown in (6.31). However, since the true implementation of the controller is in discrete time, a discretised version of it with the proper sampling time is found.

The observer gains found are shown in (6.40), based on the process and sensor noise covariances from (6.41).

$$L = \begin{bmatrix} -0.0127 & 0.0946 \end{bmatrix} \tag{6.40}$$

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 100 \tag{6.41}$$

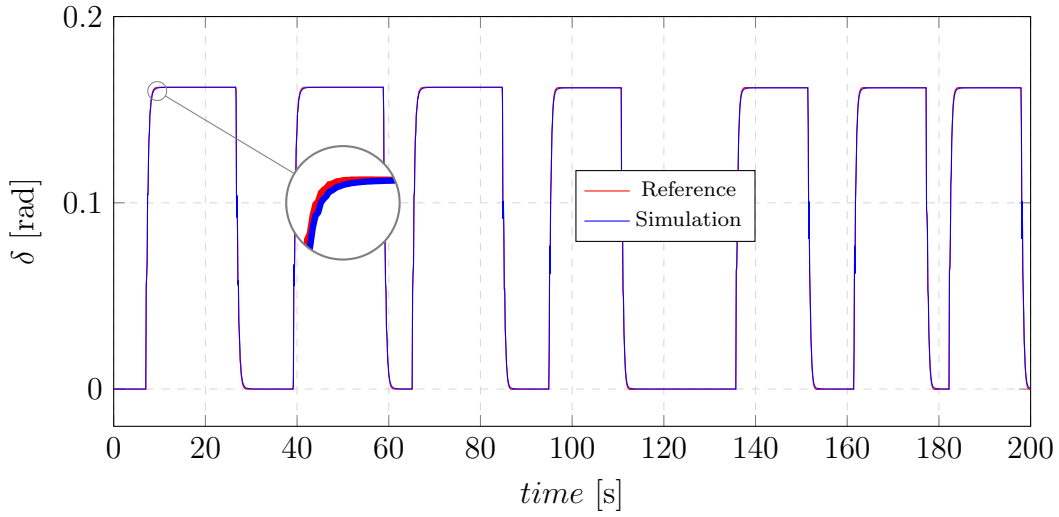


Figure 6.14: Comparison between desired δ and simulated δ.

Finally, it can be appreciated from (6.25), how, since the system is nonholonomic, the controller can not be used at longitudinal velocities close to zero. Moreover, as stated in Section 5.1.1, when running at low speeds, the friction between the wheels and the cart does not allow the actuator to excite the system and the TF used in the derivation does not apply. Therefore, an algorithm is designed such that when running at low velocities the desired steer is set to zero.

6.5 Results

This section presents the results of the controllers when implemented in the real platform.

First, the longitudinal controller results are presented. Figure 6.15 shows the evolution of the cart's longitudinal velocity. A step in the reference is given such that the cart runs at a constant velocity. It can be seen how the presented results are not the same as shown in simulation, since the velocity does not stabilise at the desired value, but instead it seems to oscillate around it.

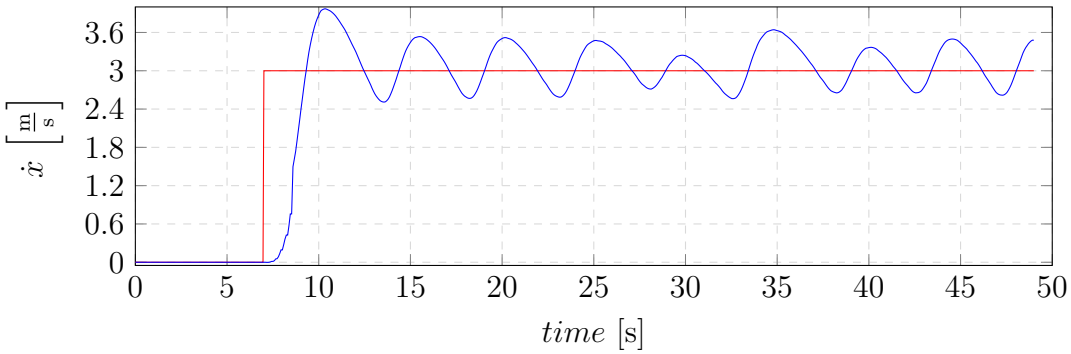


Figure 6.15: Comparison between velocity reference and estimated velocity.

When examining the desired torque coming from the lateral controller as compared to the real torque applied, these oscillations can be explained. In Figure 6.16, it can be seen how the servo

controller is not able to provide the cart with the desired value of torque. The provided torque is applied too slowly, leading to oscillations in the desired torque to correct for the velocity error.

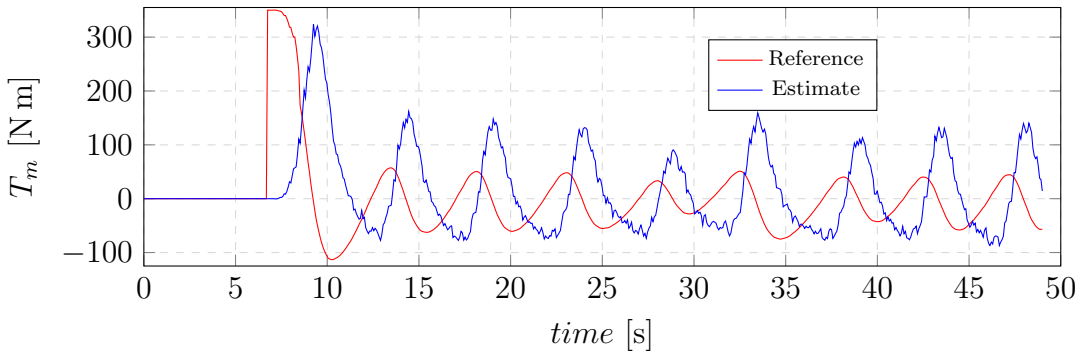


Figure 6.16: Comparison between desired and estimated torques.

Furthermore, examining the transfer function of the drive-train system shown in (5.5), it can be seen in Figure 6.17 that it is not analogue to the actual values measured. This suggests a nonlinear behaviour of the actuator system that the servo controller can not cope with.

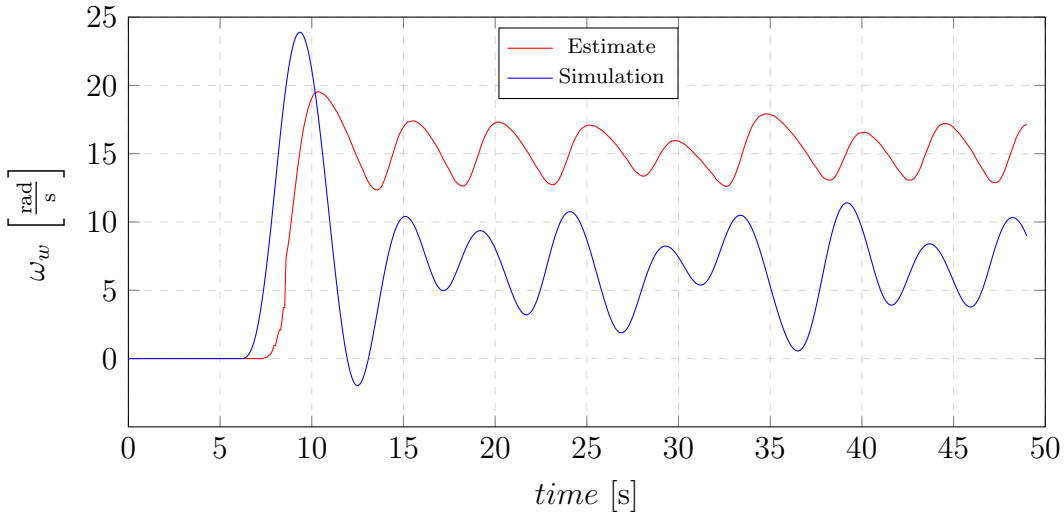


Figure 6.17: Comparison between rotational speed of the wheels and the simulated response.

Last, the results of the lateral controller implementation are shown. A test is performed such that the cart follows a straight line. Figure 6.18 shows the global position estimates. In this particular test, the cart does not seem to deviate completely of the desired shape of the path. However, as experienced in the cart, the system is not considered to fulfil its objective and it is considered to be unstable. This behaviour can be observed in Figure 6.19, where the desired steering value is plotted along its estimated value. It can be appreciated how from time $t = 9\,\text{s}$, when the controller is initialised, the system presents growing oscillations. Therefore, the controller is deemed not to be robust enough to overcome model changes.

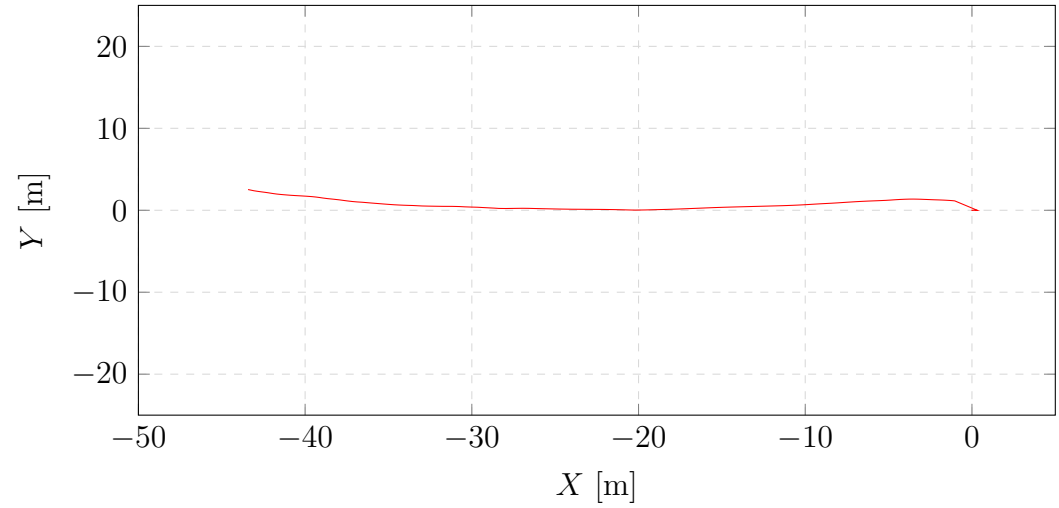


Figure 6.18: 2D plot of the global position estimates.

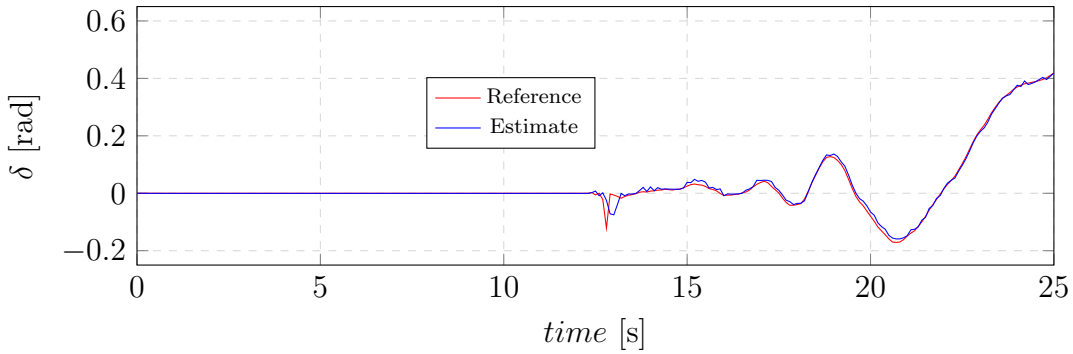


Figure 6.19: 2D plot of the global position estimates.

6.6 Control structure conclusion

In this chapter, the control structure to fulfil the objective of bringing a vehicle from one point to another, specified in Section 1.2, is derived. First, the approach to define the control objective is chosen to be the path-following due to its better properties compared to trajectory-tracking. Two nonlinear controllers are designed to address the path-following formulation. A longitudinal dynamics controller is derived based on the Lyapunov redesign method and a lateral dynamics controller is derived based on the backstepping method. The controllers are shown to fulfil the path-following objective in a simulation environment, using the complete model derived in Chapter 4. Furthermore, the longitudinal controller is derived based on uncertainty in the model parameters and proven to work effectively when these values are randomised. To be able to implement the controllers in the platform, the actuator models found in Chapter 5 are included in the control structure in a cascaded-like approach. The results of it are deemed again satisfactory in a simulation environment. Results of the implementation on the platform show a different behaviour than the simulation. On the one hand, the longitudinal structure shows an oscillatory behaviour around the desired reference, not present in the simulation results. A comparison between the drive-train actuator model and the measured values suggests that the model is

not sufficient. On the other hand, the lateral structure shows an unstable behaviour. When using the data from the implementation in the simulation environment, the controller shows the same behaviour, which suggests that it is not sufficiently robust. This is further discussed in Chapter 10.

7 Sensor fusion

In this chapter, the sensor fusion implemented in the vehicle is explained. Sensor fusion is necessary to provide accurate estimates of the vehicle's pose and motion. A detailed explanation of the objective for having sensor fusion is discussed is Section 7.1. Following the discussion of the choice of the Extended Kalman Filter (EFK), in Section 7.2, the system and measurement models for the EKF design are explained in Section 7.3 and Section 7.4 respectively. Once the models have been discussed, the EKF algorithm is presented in Section 7.5. Finally, results from both simulation and implementation are discussed in Section 7.6.

7.1 Sensor fusion objective

As explained in Section 2.3, the vehicle is equipped with multiple sensors each providing various forms of information about its movement. While each sensor may only give information about some of the states of the vehicle, fusing their data together allows for a more accurate estimate of the vehicle's pose and motion. Some sensor information may be related to the states due to an integration or differentiation of them, while in other cases they can provide similar information about a state and by fusing them, a better estimate of it can be found.

The need for accurate estimates is motivated by the control structures designed in Chapter 6, which require feedback of the vehicle's pose and motion in order to calculate the required control inputs such that the vehicle can follow desired references. The sensors chosen to be fused together to estimate the vehicle's states are the GPS, IMU and encoders on the rear wheels.

There exist various estimators available that can be used to perform sensor fusion. In particular, for use with an IMU, the commonly used techniques include the Madgwick filter [26] and the Mahony filter [27]. More commonly, a well-known estimation technique used is the Kalman Filter (KF), a linear recursive estimator [28]. The KF technique is widely used for estimating the state of dynamic systems [29]. The wide acceptance of the KF technique is due to it taking the system's dynamics into consideration, allowing precise estimates of the state over time [29]. Furthermore, the foundation of the KF has been built on such that it can be applied into nonlinear systems. Two of other EKF flavours used in nonlinear systems are Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF).

7.2 Extended Kalman Filter

The chosen estimator to provide estimates of the vehicle's states is the EKF. The motivation to go with a variation of the KF is that it allows consideration for the vehicle model and does not rely purely on the measurements of the sensors. The model that shall be used for the EKF is based on the dynamic model derived in Section 4.3. The choice of the EKF is motivated as it is able to consider nonlinearities in the model unlike the simpler KF. This is the fundamental difference between the EKF and KF.

The EKF algorithm can be described as performing two steps in each iteration of it, the measurement update and the time update. In the measurement update, the sensors are sampled

and the EKF corrects for the difference between the actual measurement and the predicted measurement of the system, and uses this to provide an update of the state estimate. In the time update the EKF propagates the states forward one sample using the nonlinear dynamic model, thus providing the forward prediction of the states using the current estimate of the states.

It is important to note that the EKF is to be implemented in ROS and the sensors attached to the vehicle publish data at a fixed sampling rate. Due to this, the type of EKF designed is chosen to be of the discrete-discrete type where both the system model and the measurement model are discretised.

7.3 System model

As mentioned in Section 7.2 the discrete dynamic model derived from Chapter 4 shall be used as the system model for the EKF. The chosen method to discretise the system is the Forward Euler method. The Forward Euler method is an explicit numerical method which means it is less computationally heavy in comparison to other discretisation methods, such as the Backward Euler. This is desirable due to the EKF being implemented as an online estimator running at a high sampling rate. A drawback of the Forward Euler is that it can be less accurate in comparison to other methods as well as that it can become unstable if the sampling rate is not sufficiently high [30].

In this section, the discrete model for the EKF is derived and it is verified afterwards by comparing it to the continuous-time model. Closing this section, the addition of bias estimates of the measurements is discussed.

7.3.1 Discrete model

It is important to define the states needed to describe the pose and motion of the vehicle. These are the states the EKF will estimate based on the measurements. The system states defining them are given as $\mathbf{x}_{\text{sys}} = \begin{bmatrix} X & Y & \psi & \dot{x} & \dot{y} & \dot{\psi} & \ddot{x} & \ddot{y} \end{bmatrix}^T$.

The first three states of $\mathbf{x}_{\text{sys}}$ describe the pose of the vehicle in the inertial frame, the next three states describe the motion of the vehicle and the last two states are necessary to relate the measured accelerations from the IMU to the discrete dynamic model.

The full continuous-time model is stated in (7.1), where the equations for the pose in continuous-time are found by applying the rotation matrix, given by (4.1).

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\psi} \\ \ddot{x} \\ \ddot{y} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{x}\cos\psi - \dot{y}\sin\psi \\ \dot{x}\sin\psi + \dot{y}\cos\psi \\ \dot{\psi} \\ \sum_{i=1}^{4}\left(\frac{1}{m}\left(F_{xi}\cos\delta_i - F_{yi}\sin\delta_i\right)\right) + \dot{y}\dot{\psi} \\ \sum_{i=1}^{4}\left(\frac{1}{m}\left(F_{xi}\sin\delta_i + F_{yi}\cos\delta_i\right)\right) - \dot{x}\dot{\psi} \\ \sum_{i=1}^{4}\left(\frac{1}{I_z}\left(l_i F_{xi}\sin\delta_i + l_i F_{yi}\cos\delta_i + l_{wi}F_{xi}\cos\delta_i - l_{wi}F_{yi}\sin\delta_i\right)\right) \end{bmatrix} \tag{7.1}$$

As mentioned before, the Forward Euler method for discretisation has been used. Consider the continuous-time model, from (7.1), in the form

$$\dot{\mathbf{x}}_{\text{sys}} = f\Big(\mathbf{x}_{\text{sys}}, \mathbf{u}\Big),$$

where, $\mathbf{u}$ is the inputs, defined by the motor torque T_m and the steering angle to each of the front wheels, δ_1 and δ_2, since the rear wheels steering angles are $\delta_3 = \delta_4 = 0$.

The Forward Euler method gives the approximation in discrete-time as (7.2), where h is the chosen sampling time. It calculates the state at $k+1$ by adding the discrete integral of $f\Big(\mathbf{x}_{\text{sys}}, \mathbf{u}\Big)$ to the state at k.

$$\mathbf{x}_{\text{sys}}\left(k+1\right) = \mathbf{x}_{\text{sys}}\left(k\right) + hf\Big(\mathbf{x}_{\text{sys}}\left(k\right), \mathbf{u}\left(k\right)\Big) \tag{7.2}$$

7.3.2 Discrete model verification

The discrete-time model is compared to the continuous-time model to verify its performance. It is known that the performance of the Forward Euler discretisation method is conditional on the chosen sampling time, h, hence, it is necessary to ensure that the discrete-time model remains stable. Furthermore, a smaller sampling time reduces the error between the continuous-time and the discrete-time models but it also increases the number of computational cycles needed [30].

The chosen sampling time for the model is, $h = 0.005$ s. This shows to be sufficient to ensure stability as well as fast enough for the development of the EKF and for the use of the states by the controller. For this reason, the implementation in the platform is decided to be sampled at 200 Hz.

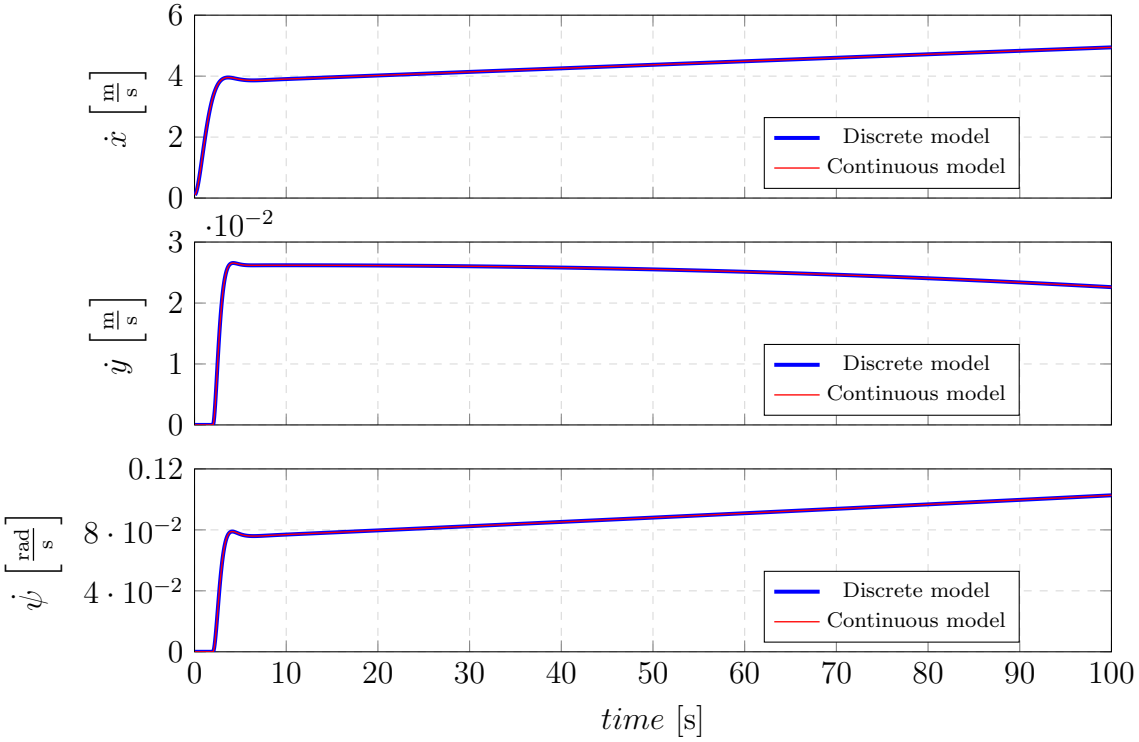


Figure 7.1: Comparison of $\dot{x}$, $\dot{y}$ and $\dot{\psi}$ from the discrete-time model and continuous-time model.

Both the discrete-time and continuous-time models are given the same inputs with $\omega_w = 4$ $\frac{\text{rad}}{\text{s}}$ stepped at $t = 0$ s followed by $\delta = 0.03$ rad at $t = 2$ s. The model mapping ω_w to T_m derived

in Section 5.2 is used such that the input to the models a reference for the motor controller. Figure 7.1 shows the comparison of $\dot{x}$, $\dot{y}$ and $\dot{\psi}$ between the two models. It also shows that the states of the discrete model are equivalent to the continuous model. It should be noted that the differences between these models are minimal and thus are unable to be seen.

For further verification, the results in the inertial frame are compared in Figure 7.2. As the inertial frame states are regarded as integrated and rotated states from the body frame, it is expected that the behaviour of both models should be identical and is seen Figure 7.2.

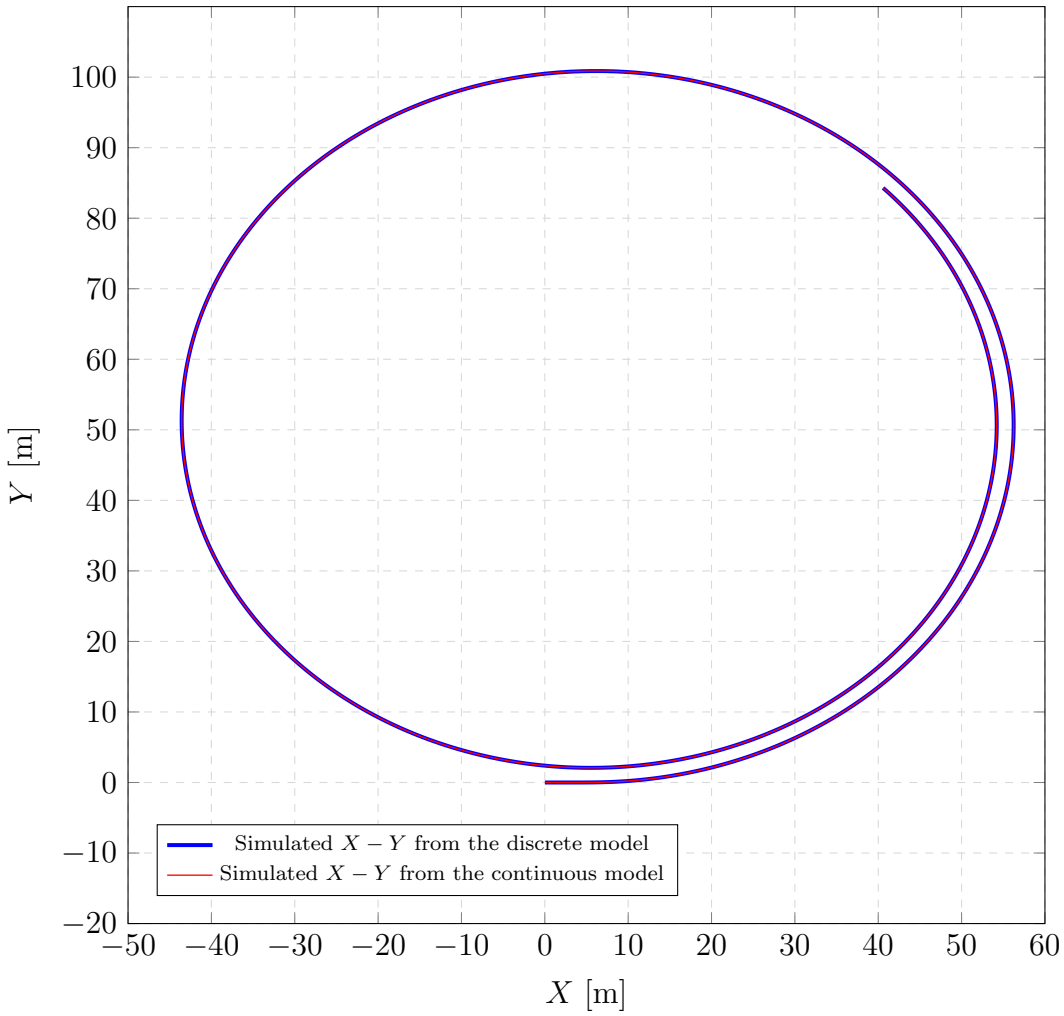


Figure 7.2: Comparison of the discrete-time model and continuous-time model in the inertial frame.

From the results shown, the discrete-time model is deemed satisfactory due to it being stable and that its propagation of the states is that of the continuous model.

7.3.3 Linear acceleration states

The discretised model in (7.2) has no explicit equations that govern the linear accelerations of the vehicle, yet the propagation of the vehicle's linear accelerations is inherent in the equations

describing the linear velocities of the vehicle. Therefore, to include equations for the linear accelerations, the linear velocity states of the system are approximated using a discrete differentiation with respect to time to yield (7.3).

$$\mathbf{x}_{\text{acc}}(k+1) = \begin{bmatrix} \ddot{x}(k+1) \\ \ddot{y}(k+1) \end{bmatrix} = \frac{1}{h} \begin{bmatrix} \dot{x}(k+1) - \dot{x}(k) \\ \dot{y}(k+1) - \dot{y}(k) \end{bmatrix} = \frac{1}{h} \begin{bmatrix} f(\dot{x}(k), u(k)) - \dot{x}(k) \\ f(\dot{y}(k), u(k)) - \dot{y}(k) \end{bmatrix} \tag{7.3}$$

In (7.3) the differentiation is done by calculating the linear velocity at $(k+1)$ using the discrete equation from (7.2).

7.3.4 Bias states

It is widely known that measurements from the IMU drift over time [31]. It is also possible that the IMU attached to vehicle is not correctly levelled. These effects can be seen in Appendix D.1, where the linear accelerations in the x and y axes have bias offsets and that angular velocity around z tends to drift over time. To overcome these issues, the original state vector is extended with three bias states, for each measurement of the IMU. The purpose of these extended states is to estimate the biases present in the IMU such that the estimated model states are unbiased.

The bias states estimated by the EKF are modelled as (7.4), to be constant.

$$\mathbf{x}_{\text{bias}}(k+1) = \begin{bmatrix} \dot{\psi}_b(k+1) \\ \ddot{x}_b(k+1) \\ \ddot{y}_b(k+1) \end{bmatrix} = \begin{bmatrix} \dot{\psi}_b(k) \\ \ddot{x}_b(k) \\ \ddot{y}_b(k) \end{bmatrix} \tag{7.4}$$

To generate the full state model for the EKF, the discrete model from Section 7.3.1 is extended with the linear acceleration model from Section 7.3.3 and the bias states. The EKF states are given as in (7.5).

$$\mathbf{x}_{\text{ekf}} = \begin{bmatrix} \mathbf{x}_{\text{sys}} \\ \mathbf{x}_{\text{acc}} \\ \mathbf{x}_{\text{bias}} \end{bmatrix} \tag{7.5}$$

Thus, the EKF system model can be written using the compact notation, in (7.6).

$$\mathbf{x}_{\text{ekf}}(k+1) = f_{\text{ekf}}(\mathbf{x}_{\text{ekf}}(k), \mathbf{u}(k)) \tag{7.6}$$

7.4 Measurement model

In Section 7.3 the discrete-time system model has been derived with all the necessary states to be estimated by the EKF. In this section, the measurement model is derived. The equations of the measurement model show how the vehicle's states relate to the information being measured by the various sensors.

The measurements from the sensors consist of X and Y values, in metres, which are processed from the GPS's latitude and longitude using the algorithm given in Appendix C.2. The GPS also provides information about the track angle which relates to heading, ψ, as well as the speed over ground which relates to the velocity, $\dot{x}$.

Using the wheel encoders on the rear wheels of the vehicle, the rotational speed ω_w is measured and this is translated into a linear speed using the radius of the wheel R_{eff}. This also provides a

measurement of $\dot{x}$, where $\dot{x} = R_{\text{eff}}\,\omega_w$.

The IMU measurements provide the angular velocity around the z-axis relating to $\dot{\psi}$ and the linear accelerations in the x-axis and y-axis. It is important to note that the IMU is located in the middle of the rear axle and not in G of the vehicle. Therefore, the IMU measurements are translated from its position to G. Leading to the measurement model being (7.7).

$$\begin{bmatrix} X_{GPS} \\ Y_{GPS} \\ \psi_{GPS} \\ \dot{x}_{GPS} \\ \dot{x}_{\text{enc}} \\ \dot{\psi}_{IMU} \\ \ddot{x}_{IMU} \\ \ddot{y}_{IMU} \end{bmatrix} = \begin{bmatrix} X \\ Y \\ \psi \\ \dot{x} \\ \dot{x} \\ \dot{\psi} + \dot{\psi}_b \\ \ddot{x} - \dot{\psi}\dot{y} + l_r\dot{\psi}^2 + \ddot{x}_b \\ \ddot{y} + \dot{\psi}\dot{x} - l_r\dot{\psi}^2 + \ddot{y}_b \end{bmatrix} \tag{7.7}$$

The measurement model can be rewritten using the notation in (7.8).

$$\mathbf{y}_{\text{ekf}}\left(k+1\right) = h_{\text{ekf}}\left(\mathbf{x}_{\text{ekf}}\left(k\right), \mathbf{u}\left(k\right)\right) \tag{7.8}$$

7.5 Procedure

The discrete-time model for the vehicle given by (7.6) and (7.8) is considered to be ideal as no noise is included in the model. This is not the case for the real system in which noise is inherent in the measurements. Due to this, process noise and measurement noise are added to the state model and measurement model, respectively. The noise is assumed to be white Gaussian noise [28], where, $Q(k)$ and $R(k)$ represent the covariance matrices of the respective noise. This yields a stochastic model defined as

$$\mathbf{x}_{\text{ekf}}\left(k+1\right) = f_k\left(\mathbf{x}_{\text{ekf}}\left(k\right), \mathbf{u}\left(k\right)\right) + \mathbf{w}\left(k\right), \quad \mathbf{w}\left(k\right) \sim \mathcal{N}\left(0, Q\left(k\right)\right), \tag{7.9}$$

$$\mathbf{y}_{\text{ekf}}\left(k+1\right) = h_k\left(\mathbf{x}_{\text{ekf}}\left(k\right), \mathbf{u}\left(k\right)\right) + \mathbf{v}\left(k\right), \quad \mathbf{v}\left(k\right) \sim \mathcal{N}\left(0, R\left(k\right)\right), \tag{7.10}$$

where the process noise and measurement noise are uncorrelated, thus,

$$E(\mathbf{w}(k)\mathbf{v}(l)^T) = 0.$$

The differentiating factor of the EKF process in comparison to the KF is the use of a nonlinear model. Since certain steps of the EKF algorithm are the same as those of the KF algorithm, the model equations need to be linearised to be used by these. When necessary, the model is linearised by deriving the Jacobian matrices of the system model, $F_k(\mathbf{x})$, and measurement model, $H_k(\mathbf{x})$. These are found using

$$F_k(\mathbf{x}) = \frac{\partial f_k\left(\mathbf{x}\right)}{\partial \mathbf{x}^T}, \tag{7.11}$$

$$H_k(\mathbf{x}) = \frac{\partial h_k\left(\mathbf{x}\right)}{\partial \mathbf{x}^T}, \tag{7.12}$$

where $\mathbf{x}$ is the latest estimated value of $\mathbf{x}_{\text{ekf}}$; i.e., $\hat{\mathbf{x}}_{\text{ekf}}(k|k-1)$, $\hat{\mathbf{x}}_{\text{ekf}}(k|k)$ or $\hat{\mathbf{x}}_{\text{ekf}}(k+1|k)$.
The EKF process also needs initial estimates for both the states, $\mathbf{x}_{\text{ekf}}(0|-1)$, and the error covariance matrix, $P(0|-1)$, where initial state estimate is initialised with the starting pose and motion values.

7.5.1 Measurement update

The measurement update estimates the value of $\mathbf{x}_{\text{ekf}}(k|k)$. The steps of the estimation are shown from (7.13) to (7.17).

$$\hat{\mathbf{y}}_{\text{ekf}}(k|k-1) = h_k(\hat{\mathbf{x}}_{\text{ekf}}(k|k-1), \mathbf{u}(k)) \tag{7.13}$$

$$\tilde{\mathbf{y}}_{\text{ekf}}(k|k-1) = \mathbf{y}_{\text{ekf}}(k) - \hat{\mathbf{y}}_{\text{ekf}}(k|k-1) \tag{7.14}$$

$$K(k) = P(k|k-1)H_k^T(\mathbf{x})\left(H_k(\mathbf{x})P(k|k-1)H_k^T(\mathbf{x}) + R(k)\right)^{-1} \tag{7.15}$$

$$\hat{\mathbf{x}}_{\text{ekf}}(k|k) = \hat{\mathbf{x}}_{\text{ekf}}(k|k-1) + K(k)\tilde{\mathbf{y}}_{\text{ekf}}(k|k-1) \tag{7.16}$$

$$P(k|k) = \left(I - K(k)H_k(\mathbf{x})\right)P(k|k-1)\left(I - K(k)H_k(\mathbf{x})\right)^T + K(k)R(k)K^T(k) \tag{7.17}$$

In (7.13) the estimated measurements are calculated using (7.8) with the predicted state, $\mathbf{x}_{\text{ekf}}(k|k-1)$. In (7.15) the Kalman gain $K(k)$ is calculated with the Jacobian, $H_k(\mathbf{x})$, using $\mathbf{x} = \hat{\mathbf{x}}_{\text{ekf}}(k|k-1)$. This and the difference between in the actual measurement and estimated measurement, (7.14), are used to estimate the states, $\mathbf{x}_{\text{ekf}}(k|k)$, as shown in (7.16). The final step in the measurement update is to calculate $P(k|k)$, the process covariance noise matrix. In this case, the Jacobian, $H_k(\mathbf{x})$, uses $\mathbf{x} = \hat{\mathbf{x}}_{\text{ekf}}(k|k)$.

7.5.2 Time updates

In the time update, the predicted state, $\hat{\mathbf{x}}_{\text{ekf}}(k+1|k)$, is updated using the nonlinear state model, in (7.18) and then the process covariance noise matrix, $P(k+1|k)$, is recalculated using (7.19). In (7.19), $F_k(\mathbf{x})$ is the linearised state model, where $\mathbf{x} = \hat{\mathbf{x}}_{\text{ekf}}(k+1|k)$.

$$\hat{\mathbf{x}}_{\text{ekf}}(k+1|k) = f_k(\hat{\mathbf{x}}_{\text{ekf}}(k|k), \mathbf{u}(k)) \tag{7.18}$$

$$P(k+1|k) = F_k(\mathbf{x})P(k|k)F_k^T(\mathbf{x}) \tag{7.19}$$

7.5.3 Covariance matrices

The process noise covariance matrix is used to tune the EKF's performance. By manipulating $Q(k)$ as the performance of the EKF can be tuned to give estimates of a more correct pose and motion of the vehicle. Furthermore, giving the controller more accurate estimates of the vehicle's states on which it can act on.

The $Q(k)$ tuning matrix is chosen as shown in (7.20).

$$Q(k) = diag(5\cdot10^{-3},\ 5\cdot10^{-3},\ 1\cdot10^{-2},\ 1,\ 1,\ 1,\ 1,\ 1,\ 1\cdot10^{-5},\ 1\cdot10^{-5},\ 1\cdot10^{-5}) \quad \forall,\ k \tag{7.20}$$

The $R(k)$ covariance matrix is modelled according to the variances found in the signals in Appendix C.3 and Appendix D.1. Both matrices are assumed to be diagonal matrices as each state and measurement signal is independent of each other.
The error covariance matrix $P(k)$ describes the covariance of the errors between the real (unknown) state and estimated state of the EKF.

7.6 Results

In this section, the simulated results and implemented results of the EKF are presented and discussed. The simulation is conducted in Simulink and once satisfactory is implemented in ROS to be used on the vehicle platform.

7.6.1 Simulation results

The nonlinear model is simulated with $\delta = 0.0625$ rad and $\omega_w = 2\ \frac{\text{rad}}{\text{s}}$. The initial states for the system are chosen as $\mathbf{x}_{\text{ekf}}(0|-1) = [0\ 0\ 0\ 0.1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$ as the model is simulated with $\dot{x} = 0.1\ \frac{\text{m}}{\text{s}}$ as an initial condition. The noise added to the measurements is based on the values found in Appendix C.3 and Appendix D.1. The EKF is tuned to provide a satisfactory response.

With the specified inputs, the results presented in this section compare the estimated states of the EKF with the simulated states of the continuous-time model and the simulated measurements expected in the vehicle. The measurements for the EKF are simulated with noise and biases added where applicable.

The inertial frame states are presented in Figure 7.3. In this, it is shown that the EKF is able to provide accurate estimates of the states while rejecting noise in the measurements.

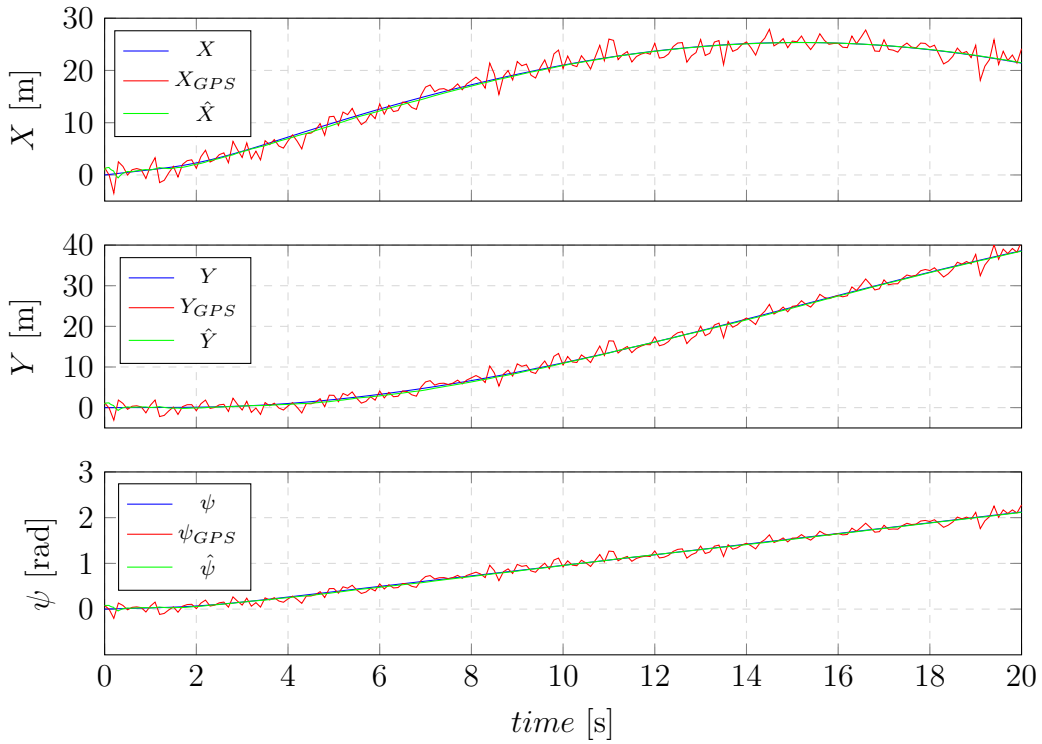


Figure 7.3: Simulation results of X, Y and ψ from EKF.

In Figure 7.4, the velocity states in the body frame are shown. It can be seen that the EKF is able to provide an accurate estimate of the system's $\dot{x}$ with two different measurements of $\dot{x}$. Between 1.5s and 7s it can be seen that the estimated $\dot{y}$ differs from the model's one. This relates to the fact that there are no measurements of the state, thus a more accurate estimate cannot be achieved. As for the estimates of $\dot{\psi}$ and $\dot{\psi}_b$, it shows that the EKF is able to reject the noise of the measurement and estimate the bias state, such that the estimated $\dot{\psi}$ is similar to that of the model.

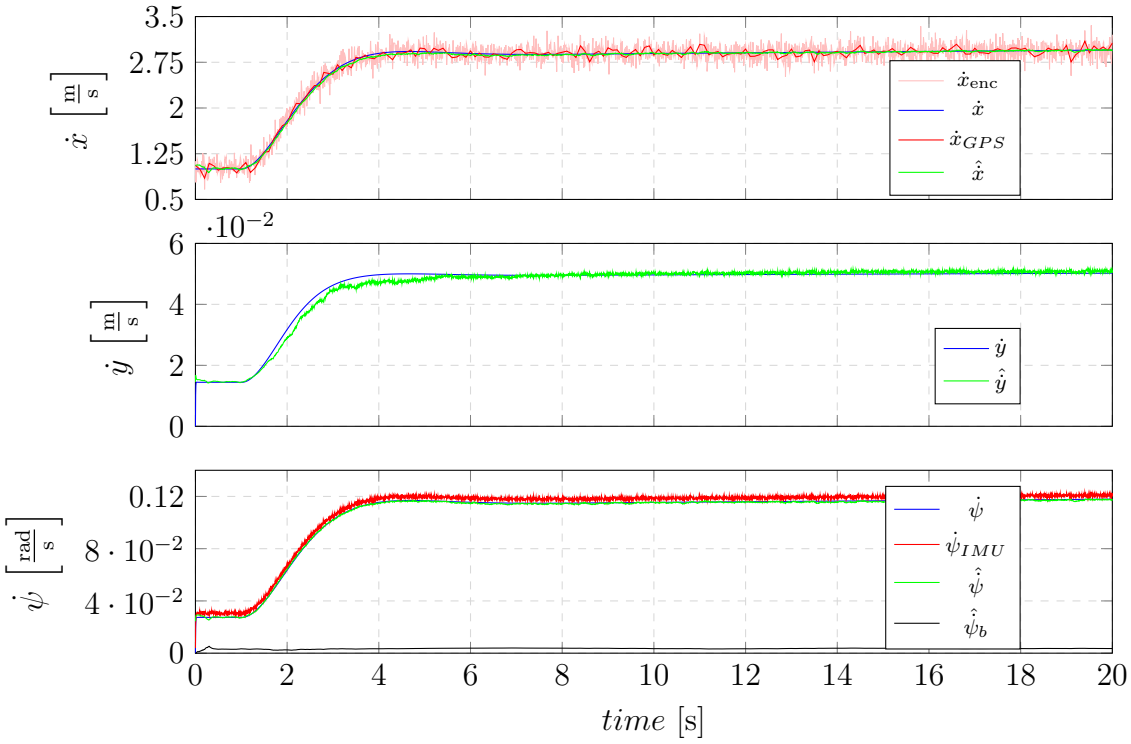


Figure 7.4: Simulation results of $\dot{x}$, $\dot{y}$ and $\dot{\psi}$ from EKF.

The last figure of the simulation results is Figure 7.5, showing the acceleration states of the system. The $\ddot{x}$ estimate in Figure 7.5 shows that it is extremely similar to that of the model, thus the differences are not visibly noticeable. It is also noted that the biases $\ddot{x}_b$ and $\ddot{y}_b$ are estimated well and both $\hat{\ddot{x}}$ and $\hat{\ddot{y}}$ are accurate to that of the model's states in G.

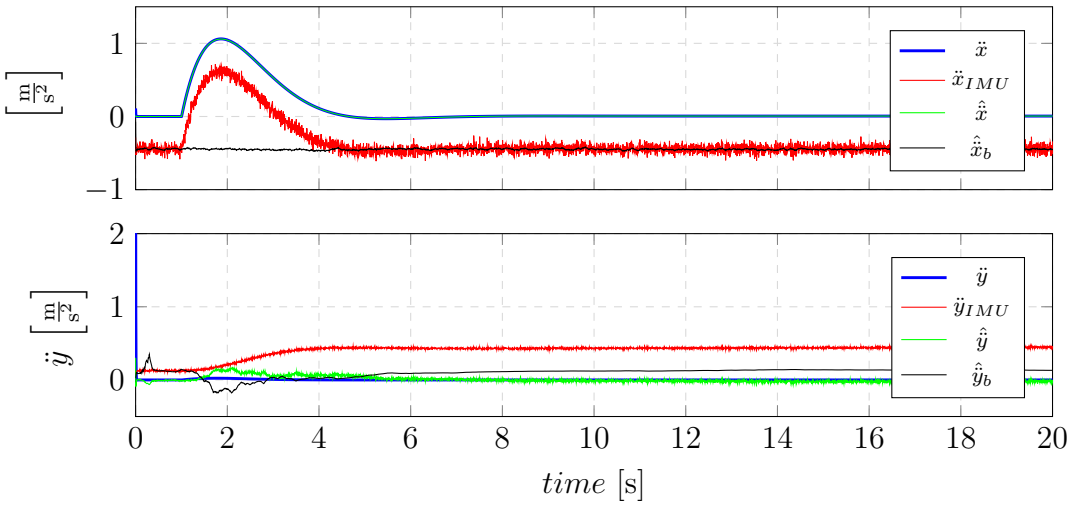


Figure 7.5: Simulation results of $\ddot{x}$ and $\ddot{y}$ from EKF.

7.6.2 Implementation results

The test to verify the EKF in the vehicle platform is performed by driving it manually. The inputs to the EKF are T_m, calculated as done in Section 4.5, and δ_1 and δ_2, calculated as in Section 5.1.

Due to the nonholonomic nature of the vehicle, the dynamic model used to design the EKF would give errors at low velocities. Therefore, a simpler estimator from ROS using the IMU and wheel encoders are used when $\hat{\dot{x}} < 0.5\frac{\text{m}}{\text{s}}$. Thus, when this threshold is surpassed, the initial states for the system are taken directly from the previous measurement of the simpler estimator. The results presented in this section compare the estimated states of the EKF with the measurements from the sensors.
Figure 7.6 shows the inertial frame estimates of the EKF with measurements from the GPS. The greater differences between the estimates of X and Y occur between 100s and 120s. These are possibly due to the GPS receiver reacting later to the change in direction of the vehicle. As a single receiver GPS is typically not extremely accurate, more trust has been given to the model to estimate these states. Another possible reason for this could be that the transformation from Appendix C.2 may not be an accurate. As for ψ, the EKF puts more trust in the measurement since it is deemed to be fairly accurate. It is still able to reject anomalous peaks, such as the one around 77 s.

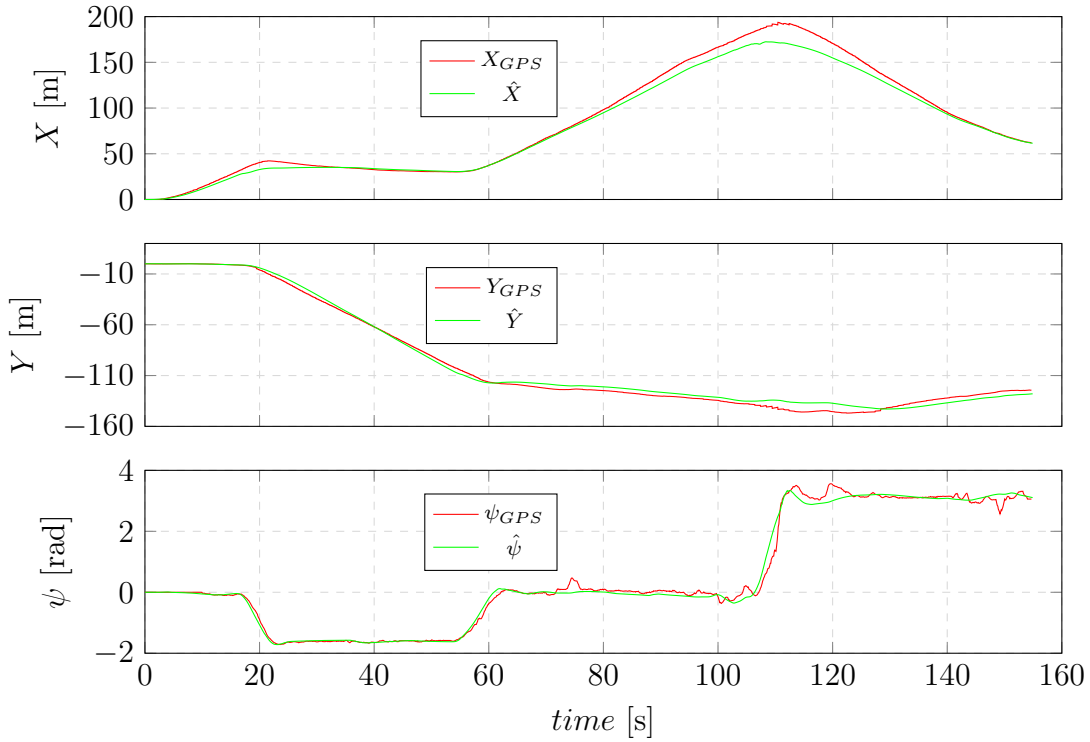


Figure 7.6: Implementation results of X, Y and ψ from EKF.

The velocity states in body frame are shown in Figure 7.7. It can be seen that the noise in the measurements is significantly reduced. Furthermore, spikes from the $\dot{x}_{GPS}$ are also rejected as can be seen in the one at 110 s. The estimated $\dot{\psi}$ also has a reduced amount of noise compared to the measurement. Testament to the $\dot{y}$ estimate in is that its features are similar to that of $\hat{\dot{\psi}}$ which is as expected from the model.

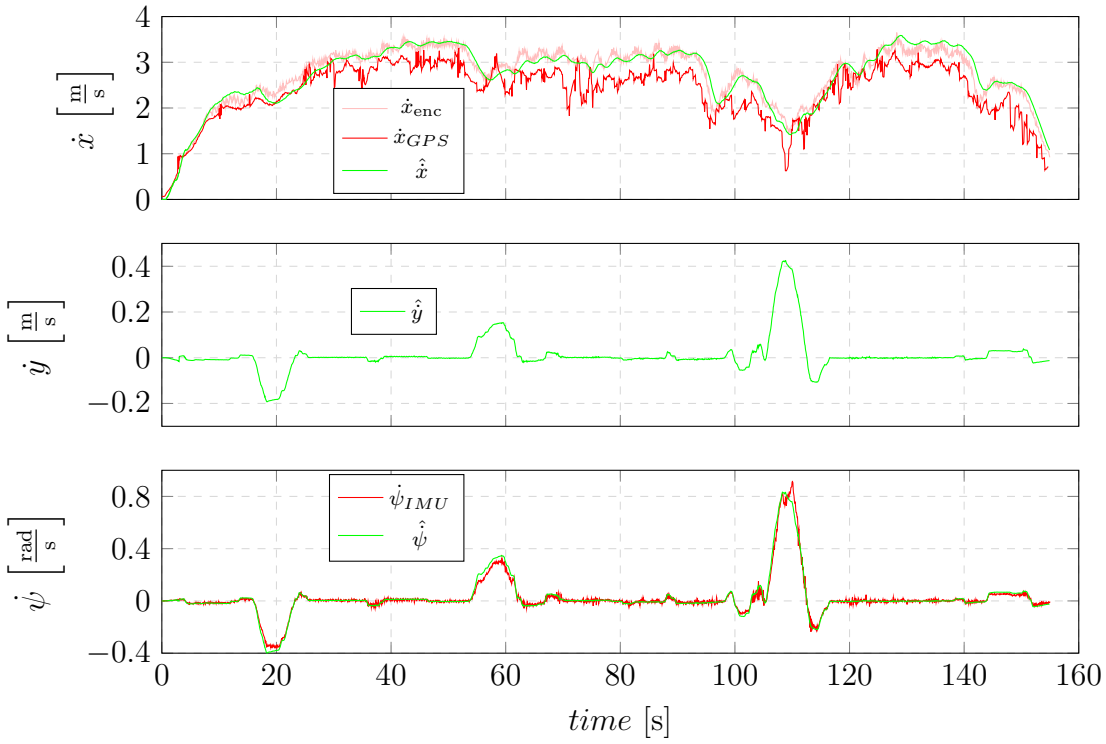


Figure 7.7: Implementation results of $\dot{x}$, $\dot{y}$ and $\dot{\psi}$ from EKF.

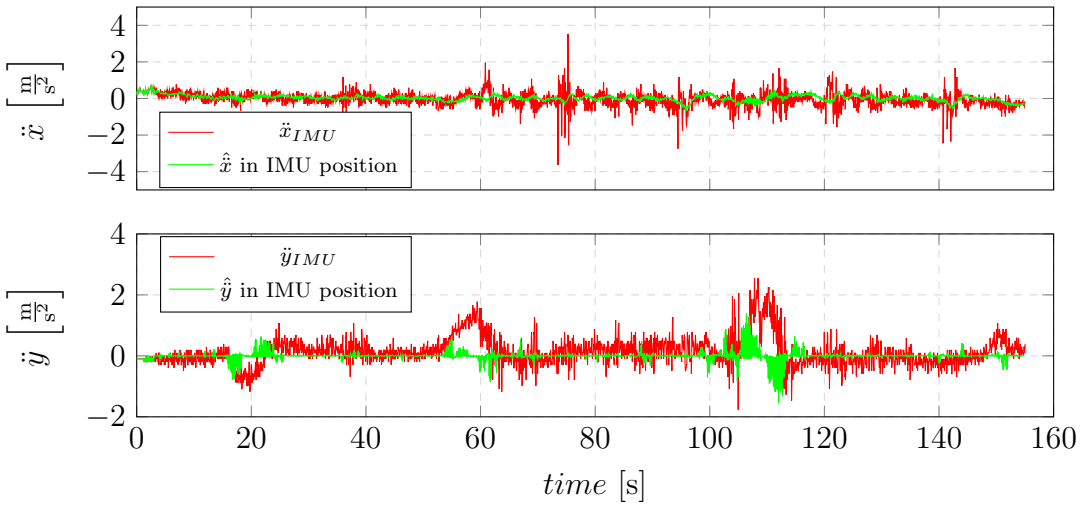


Figure 7.8: Implementation results of $\ddot{x}$ and $\ddot{y}$ from EKF.

The acceleration states in the position of the IMU are shown in Figure 7.8. It is shown that both $\hat{\ddot{x}}$ and $\hat{\ddot{y}}$ from the EKF are centred around 0 $\frac{\text{m}}{\text{s}^2}$, thus showing that any bias in the measurements is corrected for. There is a significant noise reduction, but the substandard performance of estimating $\ddot{y}$ can be seen. While, this estimate may not be correct, this state is not used for the controller feedback, thus the performance of the controller is not be compromised by this. A possible reason for wrongly estimating $\ddot{y}$ is that the IMU changed during the project and better performance of the EKF using new IMU is not achieved. Furthermore, the distance to the

IMU from G may not be correct as it is kept fixed at its nominal value yet it may vary between different situations.

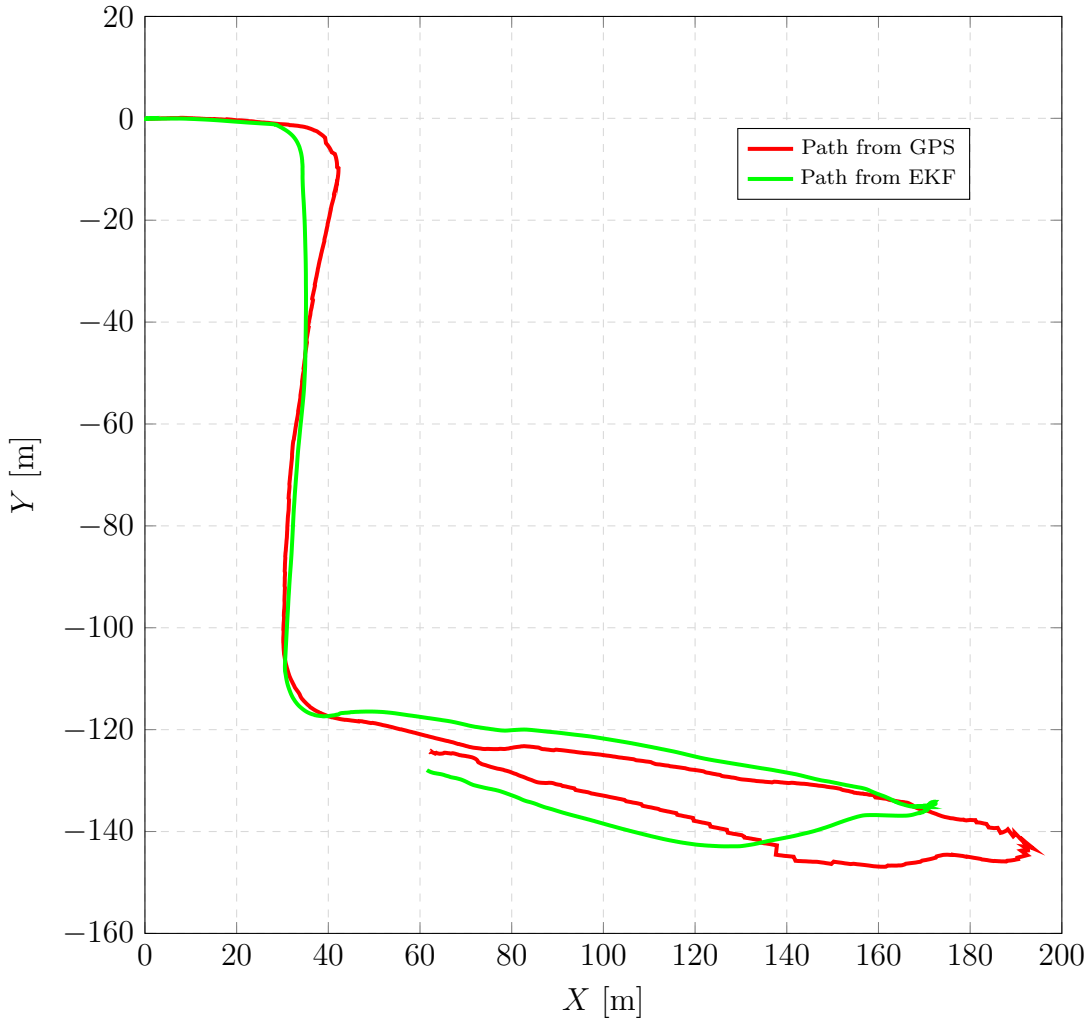


Figure 7.9: $X - Y$ plot from EKF vs. raw GPS measurements.

Figure 7.9 shows the path taken by the vehicle by comparing the GPSs measurements to the EKF estimates. It can be seen that the EKF is able to estimate the motion of the vehicle better than the GPS alone, due to the EKF estimated pose is closer to what was driven.

7.7 Sensor fusion conclusion

It is now concluded that the sensor fusion designed in this chapter is satisfactory to be used by the controller. The EKF is chosen as the technique to be used due to its characteristic of including the nonlinear dynamics of the vehicle into consideration. In this chapter, the models for the EKF are designed, the algorithm for the estimator is explained and finally the performance of the EKF is evaluated.

Part III

Navigation

8 Navigation Planning

The navigation planning can be defined as the problem of how to get from an origin position to a goal position. Although this is a recurring problem in mobile robotics, the navigation planning considered in this thesis is carried out in urban environments, making it very specific. Two of the particularities of such an environment are mobile obstacles such as other vehicles or pedestrians and its semi-structured nature, i.e. the vehicle has to obey circulation rules.

The navigation planning can be divided in two subproblems. The first one, referred to as route-planning, is to find the fastest or shortest route from the origin position to the goal position, similarly as a web mapping service such as Google Maps™ would do. This route is then defined as a sequence of waypoints which the vehicle has to attend. An example taken from OpenStreetMap™ is shown in Figure 8.1 where the planned route is shown in blue and the sequence of waypoints are represented in red. The second one, referred to as path-planning, is to convert the sequence of waypoints into a path.

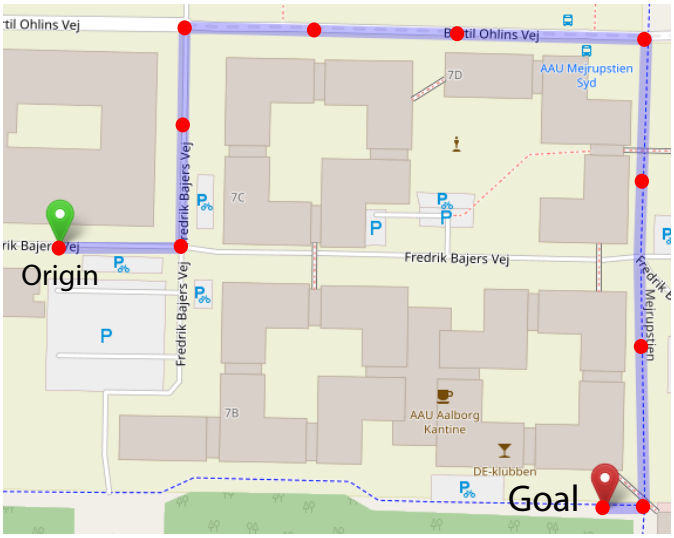


Figure 8.1: Extract taken from OpenSourceMaps™ showing, in blue, the planned route and, in red, the waypoints derived from it.

In the following, the path-planning problem is investigated while the route planning problem is left out of the scope. Nonetheless, the intuition about solving the route planning is to use an existing mapping service such as OpenSourceMaps™, which given an origin and a goal location it would yield a sequence of latitude/longitude coordinates that could be used to choose the needed waypoints used by the path-planner later on. Alternatively, there already exist Libraries in ROS that allow to do route planning given a map from OpenSourceMaps™.

Because the authors of this thesis are not previously introduced to the path-planning problem, it is assumed that neither is the reader. In this direction, this chapter first analyses the path-planning problem in Section 8.2 and then proceeds with a state-of-the-art review of different path-planning approaches in Section 8.3 which highlights some of the challenges in path-planning and leads to the decision to use the Optimal Rapidly Random exploring Tree (RRT*) algorithm.

8.1 Problem Formulation

To proceed with the chapter, first it is deemed necessary to formulate the path-planning problem. This is done in this section which, without loss of generality introduces the nomenclature that is used throughout the chapter.

Recalling from Section 4.1, the vehicle motion is considered to be restricted to $\mathbb{E}^3$. Keeping this in mind, let $\mathcal{X} \subset \mathbb{R}^3$ be a compact set and $\sigma(s) \in \mathcal{X}\ \forall s$ describe a unique path, where $s \in [0, S]$ and $S \in \mathbb{R}_{\geq 0}$. s describes the length of a path and therefore S represents its total length.

Furthermore, let $\mathcal{X}_{\text{obs}}$ and $\mathcal{X}_{\text{goal}}$ be open subsets of $\mathcal{X}$. These subsets are referred to as the obstacle space and the goal region. $\mathcal{X}_{\text{free}}$ is also a subset of $\mathcal{X}$, defined as $\mathcal{X} \setminus \mathcal{X}_{\text{obs}}$ and referred to as the free space. Furthermore, $\mathcal{X}_{\text{goal}} \subset \mathcal{X}_{\text{free}}$.

Informally speaking, the path-planning problem is to find a path $\sigma(s)$ from an initial state $z_{\text{init}} \in \mathcal{X}_{\text{free}}$ that reaches the goal region and that avoids the obstacle space.

8.2 Problem Analysis

The path-planning problem has been defined in Section 8.1. To better understand the problem and most importantly to understand it given the conditions in this thesis, the problem is broken down in three subproblems which are consequently addressed in the following subsections.

8.2.1 Continuous path

Based upon the description of the vehicle motion presented in Chapter 4, it is easy to realise that the golf cart belongs to the group of mobile robots with constrained motion. These constraints are referred to as holonomic constraints and the group of mobile robots affected by them are usually referred to as nonholonomic robots. The holonomic constraints affecting the golf cart are evident since its motion is dominated by two propulsion wheels which are not steerable and two steering wheels with limited steering actuation. These constraints prevent the vehicle to freely visit $\mathbb{E}^3$. E.g. the vehicle cannot change its orientation without changing its X and Y position.

Due to these holonomic constraints, the vehicle is not be able to follow certain paths, in fact it is only capable of following a path without stopping if it has continuous curvature. This can be easily illustrated when such vehicle is following a path made of the concatenation of two circular arcs as shown in Figure 8.2. When the vehicle reaches the junction, it has to steer from positive to negative steering and unless it stops and steers, in a sequential manner, it would require steering actuation with infinite acceleration.

Tracking such a path, because steering acceleration is bounded, would result on tracking errors. To overcome such problem, some have tried to develop path-planners with continuous curvature such as the Continuous-Curvature Path Planner (CCPP) presented in [32] which considers a piecewise path representation with clothoid arcs and continuous curvature at the joints between segments. Others just assume tracking errors and leave the controller to account for them, a common choice being Dubins' curves [33] which consist on concatenations of circular arcs and straight lines. The ease of finding the shortest path between two poses with Dubins' curves makes it a preferable choice.

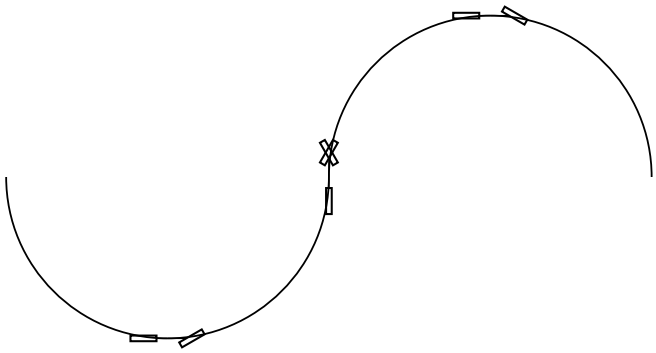


Figure 8.2: Example of a path with noncontinuous curvature. When the vehicle reaches the intersection point it has to change from left curvature to right curvature immediately, i.e. a discontinuity on the path curvature.

8.2.2 Configuration space

Path-planning can generally be viewed as a search in a metric space, $\mathcal{X}$, for a path $\sigma(s)$ from an initial state z_{init} to a goal region $\mathcal{X}_{\text{goal}}$. It is assumed that a fixed obstacle region, $\mathcal{X}_{\text{obs}}$ must be avoided. Then the problem can be reformulated to only consider paths which lie entirely in the subspace $\mathcal{X}_{\text{free}}$.

States belonging to $\mathcal{X}_{\text{obs}}$ can correspond not only to fixed obstacles but also includes other interpretations which may depend on the particular problem. An example of such a configuration space is represented in Figure 8.3 where $\mathcal{X}_{\text{obs}}$ is represented as the areas in red and blue and the rest is $\mathcal{X}_{\text{free}}$. The red areas represent fixed obstacles while the areas in blue represent other interpretations, e.g. dynamic velocity bounds, which grow with increasing velocity to account for braking distance.

How this representation is done usually depends on the path-planning approach. However, two main representation methods can be differentiated. Some planners need an explicit representation of the configuration space to define the path, while others need an implicit representation of it. The latter usually uses an occupancy map which is a discretisation of the configuration space in the form of a grid where each cell is defined either as belonging to $\mathcal{X}_{\text{obs}}$ or $\mathcal{X}_{\text{free}}$. On the other hand, explicit representations need of mathematical models of the environment.

At this stage, a representation of the configuration space, i.e. the university campus, is not available. Hence, before any path-planning can be performed it is important to obtain such a representation of the configuration space. Because each planning relies on a specific kind of representation, the decision to either perform an explicit or an implicit representation of the configuration space is left for later with the planner choice. Nevertheless, the intuition of the authors is that obtaining an implicit representation would result much easier to perform rather than an explicit one.

8.2.3 Perception

It is seen in Section 8.2.2 that the representation of the configuration space is really important for path-planning. Also in Section 8.2.2, two kinds of representation are introduced, explicit maps and occupancy maps where either representation is not available and thus it has to be built. To do so, indistinctly of an explicit or an implicit representation, the golf cart is equipped with LIDARs.

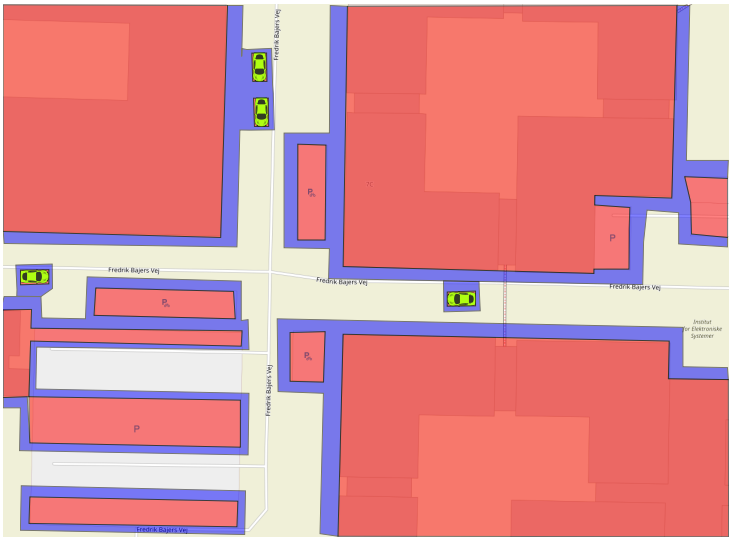


Figure 8.3: Extract of map obtained from OpenSourceMaps^{TM}. It shows a representation of the configuration space for this particular piece of map, where the addition of blue and red areas form the $\mathcal{X}_{\text{obs}}$ and the rest belongs to $\mathcal{X}_{\text{free}}$. The areas in red represent the area taken by fixed obstacles while the blue areas represent other interpretations.

It is true that a fair first approach could be to consider a reduced environment and carry out a representation by just manually measuring it. However, this solution is not scalable for future work that might come and so it is discarded. Instead, the approach is to transform the data coming out of the LIDARs into such a representation.

This being said, it is important to understand the configuration of the LIDARs mounted on the golf cart. In fact, previous work in autonomous navigation claim the importance of perception, particularly in the DARPA Urban Challenge (DUC) [8], where no car without actuated LIDARs managed to finish the challenge. However, this thesis pursues to reach inexpensive autonomous navigation. In this regard, as it is mentioned in the introduction (Chapter 1) this project mirrors the approach taken by the research venture SMART and adopts the same configuration for its LIDARs.

Recalling from Chapter 2, one LIDAR is mounted on a support with variable height intended to detect objects at a "knee" level and another is fixed to the roof of the vehicle with a push-broom configuration. The intention behind this configuration is to detect objects around the vehicle and their height.

8.3 State-of-the-art review

This section is intended to review the state-of-the-art of path-planning methods used for urban navigation. In 2004 the DARPA organized a competition where fully autonomous ground vehicles had to complete a substantial off-road course within limited time, named Grand Challenge (DGC). A second event took place in 2005 and later in 2007 DARPA organized the Urban Challenge which extended the competition to autonomously operate in an urban environment.

The work done by the different teams in these competitions has become the ground milestone in autonomous driving for car-like vehicles. Within this work, novel solutions for path-planning are presented. In fact, most of the recent research in path-planning is based in such solutions presented after the DGC and the DUC.

Hereunder, three path-planner alternatives are presented. The first path-planner is based on artificial potential fields and was presented back in 1986. Besides its popularity as a planner itself it has received a large amount of attention as extension of new planners. The second planner is based on an algorithm named A* and the third one on another algorithm named RRT. Both algorithms are meant for path-planning and both have evolved into many other planners up until today. For that reason, such algorithms are presented along with different modifications.

8.3.1 Artificial potential field

The artificial potential field method is introduced in [34]. The principle behind it is that the vehicle moves in a field of forces where the goal position is represented as an attractive pole while obstacles are represented as repulsive forces.

This technique was mainly developed for obstacle avoidance for manipulators even though it is used in mobile robotics. If properly implemented, it demonstrates remarkable performance on accomplishing various tasks in rather complex environments. A shortcoming of this path-planning technique may be the existence of a local minima in which the vehicle would remain trapped. A more specific shortcoming for the path-planning problem presented here is that it relies on an explicit representation of the obstacles in the configuration space. This may result in an expensive computational load, but more important, such an explicit representation is not available at the moment and thus it should be made, which intuitively does not seem like an easy task.

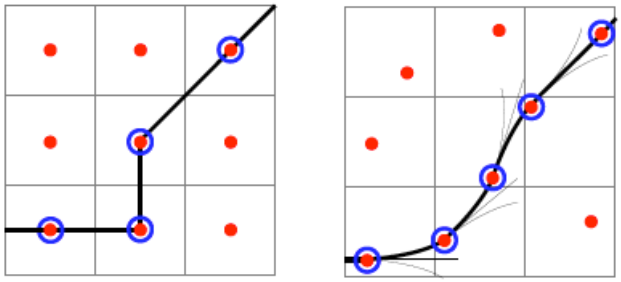
8.3.2 Hybrid-state A*

A* is a type of graph search algorithm used for path finding. Peter Hart, Nils Nilsson and Bertram Raphael from Stanford Research Institute (SRI) first described the algorithm in 1968 in [35] as an extension of the Dijkstra algorithm [36]. The A* algorithm is based on a discretisation of the configuration space on a grid or cell decomposition form and uses heuristics to find the shortest path.

-In 2007 for the DUC, the Junior team from Standford University used a modified version of this approach to design their path-planning. The work is presented in [37] and names the path-planning approach as Hybrid-state A* search.

The Hybrid-state A* is a modification of the traditional A*, which only allows visiting centres of cells. Instead, the hybrid-state A* associates each cell on the grid with a continuous 3D state of the vehicle. An illustration of it is shown in Figure 8.4. As the authors claim, this approach is not guaranteed to find the minimal-cost solution but it does guarantee drivability. However, from the same work, a solution to find such an optimal path is proposed, where conjugate gradient descent produces a path that is at least locally optimal, which usually matches with the global optimal as well.

As the A* algorithm, the Hybrid-state A* is based on a discretisation of the configuration space, i.e. an implicit representation of the configuration space, e.g an occupancy map. This kind



(a) (b)

Figure 8.4: Graphical comparison between the traditional A* on the left and the hybrid-state A* on the right found in [37]. It can be seen how the traditional A* associates costs with the center of the cell while the hybrid-state A* associates a continuous state with each cell.

of map stores locations of particular landmarks that are detected with the vehicle sensors, an example is shown in Figure 8.5. These maps are usually generated offline and have a bigger resolution than the grid used by the A* algorithm. Grid maps or occupancy maps are usually used by other mobile robotics applications such as SLAM.

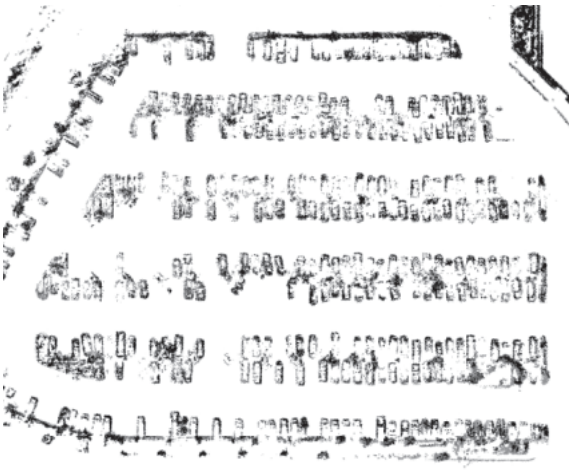


Figure 8.5: A grid-based obstacle map from [38] with size 160 m $\times$ 160 m with a resolution 0.0015 $\frac{\text{m}}{\text{pixel}}$.

The authors of the hybrid-state A* refer to this kind of planning as free-space planning because it just finds a path within the free-space left from the obstacles around. Further improvements to this approach are presented in [38] and [39] where this free-space planning is extended. On this latter work, the authors consider that the free-space path-planner is well-suited for unstructured environments. However, in structured environments such as parking lots, it is important to realise advanced driving behaviours. For example the free-space path-planner might plan paths cutting across through empty parking spots which is not a desired behaviour. To overcome this problem previous work focused on capturing these complex driving behaviours in a lane-network graph and then perform path-planning based on this graph, known as graph-based planning. Conversely to the free-space planning, graph-based might be too restrictive for the vehicle. The solution proposed in [38] and [39] is to seamlessly integrate graph-based and free-space planning. These graphs can be obtained either manually or automatically estimated from sensor data. To

do so some techniques are available, an example is the one presented by the same authors in [38].

8.3.3 Rapidly-exploring Random Trees (RRT)

RRT is one kind of sampling-based path-planning, which was first introduced by S.M. LaValle in [40]. It is a heuristic and randomised approach that quickly explores the configuration space. The basic idea behind this method is to generate a tree $G = (V, E)$ that grows by randomly placing sample state (z_{rand}) within $\mathcal{X}_{\text{free}}$ and extend it to its nearest node. If the path between the new sample and the nearest node does not collide with any obstacle, i.e. if $\sigma(s) \notin \mathcal{X}_{\text{obs}}\ \forall s$, then the new sample is added to the tree, else a new random state sample is drawn and the process is repeated sequentially. The path is initialised using the state z_{init} and the algorithm stops when the region $\mathcal{X}_{\text{goal}}$ is reached. In this particular approach, the path connecting the nodes is straight segments.

Similarly to the A* algorithm, RRT has received a large amount of attention. For this reason, it is possible to find several versions of it. Two of these are considered here for the state-of-the-art review.

Closed-Loop RRT (CL-RRT)

The first version considered is the one the MIT team implemented on their vehicle named Talos at the DUC. This RRT version is presented in [41] and it mainly differs from the traditional RRT by sampling reference paths instead of random poses. These references, referred as $r(s)$, are fed to a stable closed-loop system which consists of a controller and a vehicle model, see the structure in Figure 8.6. These reference paths are considered in [41] to be just simple straight lines.

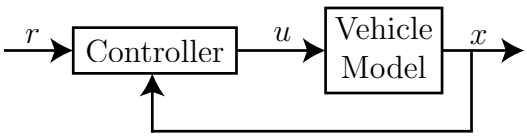


Figure 8.6: Block diagram of the closed-loop system used for the CL-RRT algorithm. The system is formed by a controller and a model of the vehicle, where $r(s)$ is the reference path fed to the controller and $x(s)$ is the output of running the forward simulation.

The algorithm flow is similar to the traditional RRT, as it also builds a tree incrementally. In CL-RRT, the tree is built by sampling random reference paths and running a forward simulation through the closed-loop system. The output of the simulation, referred to as $x(s)$, is the resultant path from the closed-loop system tracking $r(s)$. Also, as in the traditional RRT, the output is added to the tree if $x(s) \notin \mathcal{X}_{\text{obs}} \quad \forall s$. An illustration of this procedure is shown in Figure 8.7.

The main advantage of this approach is that it generates a path that accounts for the dynamic behaviour of the vehicle. Furthermore, because the dynamics of the vehicle might be unstable, including a controller within the forward simulation guarantees that it is stable to follow the generated path.

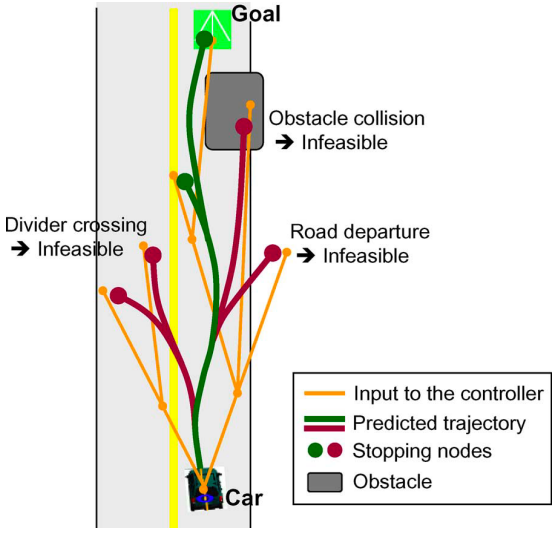


Figure 8.7: Illustration of how the tree is built incrementally in CL-RRT [41]. The reference paths, $r(s)$, are shown in orange and the output paths, $x(s)$, are shown in green if the path does not collide with any obstacle or on the contrary in red.

Optimal RRT (RRT*)

The second version considered is found within a more recent work presented by Sertac Karaman et al. in [42]. Here the authors analyse the flagships of sampling-based algorithms, i.e. Probabilistic Road Maps (PRM) and RRT, to later present new and improved versions of these referred as Optimal PRM (PRM*) and Optimal RRT (RRT*) respectively. The two approaches are analysed in terms of probabilistic completeness, asymptotic optimality and complexity of the algorithm. These properties are defined in [43]. In [44] the authors present an extension to RRT* which is able to cope with kinodynamic constraints and in [45] an any-time motion planning using RRT* is presented.

Regardless of the RRT version, to evaluate whether a path collides with an obstacle or not, an implicit representation of the configuration space is used. Similarly to the A* algorithm, the RRT algorithms use an occupancy map for that purpose. For the case of CL-RRT, such map is implemented as a look-up table and is referred as driveability map. It includes static and moving objects, lane boundaries and other hazards from the road. These are captured in the drivability map as infeasible regions which are no-go areas, high-cost regions which have to be avoided if possible and restricted regions which may only be entered under certain circumstances. Each region is given a cost that allows decision making. To illustrate this behaviour one can think of using the oncoming traffic lane to overtake. An example of such map is given in Figure 8.8.

8.4 Choice of algorithm

The aim on this project for path-planning is not to present a novel approach nor improvements of any existing one. Instead, the aim is to choose the existing approach that better suits the characteristics of this project. The intention of this section is to discuss the different state-of-the-art solutions presented in Section 8.3 based on the analysis of the path-planning problem introduced in Section 8.2 and lead to a choice of algorithm.

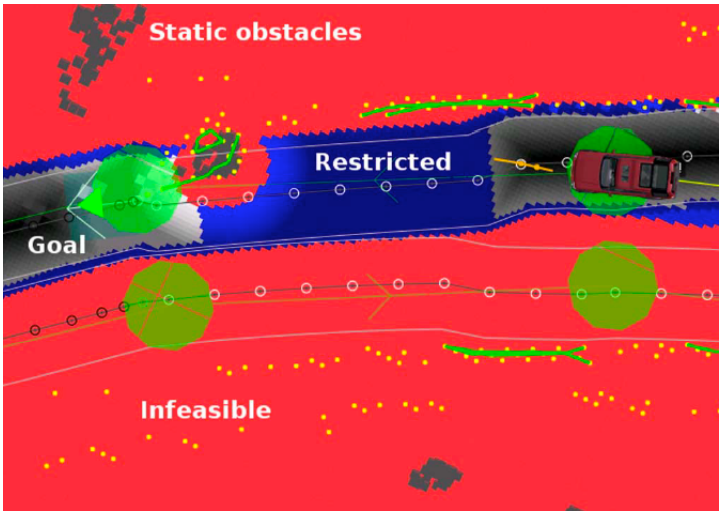


Figure 8.8: Example of drivability map from [41] where the different regions can be identified, i.e. the restricted area in blue, in this case representing the follow distance, and the infeasible area in red. In particular, the map shows a single lane road. The green circles represent other vehicles. Without oncoming vehicles in the opposite lane the cost would change from infeasible to restricted, allowing overtaking if needed.

The first factor to take into account is the continuous curvature of the generated path. As seen in Section 8.2.1, continuous curvature is an important factor when path-planning for vehicles with holonomic constraints. However, due to complexity of urban environments it might be expensive to compute a path with continuous curvature. Instead, some path-planners work with noncontinuous curvature paths which allows them to reduce computational power, assuming deviations from the path reference when driving. An alternative that accounts for computational efficiency and still guarantees continuous curvature is the RRT* with kinodynamic constraints. The approach presented in [44] uses Dubins' curves to find the shortest path between two nodes in a tree and it is known that such curves do not have continuous curvature. However, when deriving the final path, the algorithm propagates the Dubins' curve through a simple vehicle model obtaining a continuous curvature path.

Another important factor for the choice of algorithm is the representation of the configuration space. In Section 8.2.2 two differentiated ways of representing the configuration space have been presented: explicit and implicit. As it has already been exposed, such a representation for the environment where the golf cart drives is not available and thus has to be made from scratch. From the three alternatives presented in Section 8.3, only the artificial potential field approach makes use of the explicit representation. This approach is rather popular in path-planning for end-effectors. From the literature one can notice that most of the planners requiring an explicit representation of the configuration space are meant for small and static environments. For this reason this kind of representation is dismissed.

Thus, the final choice is left between A*, RRT and their respective variations. These remaining options have been broadly used for path-planning in urban environments, e.g. the Hybrid-state A* and the CL-RRT used in the DUC. However, these implementations have been tested on vehicles with a high level of perception, i.e. using multiple LIDARs, 360° LIDARs, cameras, stereo cameras and radars, which is not the case of the platform available. A similar platform is the one presented by the SMART team that has previously been introduced in Chapter 1. In an effort to replicate the results obtained by this research group the choice of algorithm is to

similarly use RRT* and in particular the RRT* with kinodynamics constraints presented in [44].

8.5 Optimal RRT (RRT*)

The choice of path-planner has been to use optimal RRT, also referred as RRT*. Recalling from the previous sections and in particular from Section 8.3.3, this is a sampling-based motion planning presented by S. Karaman et al. for first time in [42]. Further work in RRT* is done by the authors, here it is considered the RRT* with kinodynamic constraints which was presented in [44]. To ease the reading, RRT* with kinodynamic constraints is simply referred as RRT*. This section is meant to give an insight into the algorithm.

The aim is to replicate the work done in [44] such that the problem formulated in Section 8.1 is fulfilled. In this regard, the optimality of RRT* is taken as an additional feature and there is no intention to prove so.
An implementation of the algorithm is developed. The results obtained from this implementation are presented in Section 8.5.4.

8.5.1 Problem Formulation

The problem formulation of RRT* slightly differs from the one presented in Section 8.1. The one presented here is not contradictory to the one in Section 8.1 but needs further formulation. One of the reasons is that optimally is discussed within this approach. For this reason, the problem formulation in Section 8.1 is extended to include the optimality of RRT*.

Let $\mathcal{U} \subset \mathbb{R}^m$ and consider the compact set $\mathcal{X}$ and the dynamic system

$$\dot{x}(t) = f(x(t), u(t)), \qquad x(0) = x_0 \tag{8.1}$$

where $x(t) \in \mathcal{X}$, $u(t) \in \mathcal{U}$ $\forall t$, $x_0 \in \mathcal{X}$, and f is a continuously differentiable function. Furthermore, the set of bounded measurable functions from $[0, T]$ to $\mathcal{X}$ for any $T \in \mathbb{R}_{\geq 0}$ is defined as $\mathbf{x}_{\mathrm{RRT}^*}$ and the same can be done to define the set $\mathbf{u}$. The functions belonging to $\mathbf{x}_{\mathrm{RRT}^*}$ are referred as trajectories and the functions belonging to $\mathcal{U}$ are referred as controls.

Instead of looking for a path $\sigma(s)$, RRT* aims to find the optimal control law $u : [0, T] \longrightarrow \mathcal{U}$ that minimizes a given cost functional $J(x)$ such that the unique corresponding trajectory $x(t)$ considering (8.1) avoids the obstacles in the configuration space, i.e. $x(t) \in \mathcal{X}_{\mathrm{free}}$ for all t.

Notice that RRT* produces trajectories $x(t)$ and not paths $\sigma(s)$. This is not considered a problem as the trajectory can be transformed into a path.

8.5.2 Algorithm

Before proceeding to the algorithm it is necessary to introduce the following primitive procedures upon which the algorithm relies.

Sampling: The sampling procedure `Sample` returns an independent and identically distributed (i.i.d.) sample from the obstacle-free space, i.e. $z_{\mathrm{rand}} = \{X_{\mathrm{rand}}, Y_{\mathrm{rand}}, \psi_{\mathrm{rand}}\}_{\mathcal{R}}$. The sampling is done with a uniform distribution following [44]. Nonetheless, it is claimed in [44] itself that the results hold for a large class of sampling strategies.

Distance: The distance function `dist` returns the cost of the optimal trajectory between two states if there are no obstacles. More specifically, $\texttt{dist}\,(z_1, z_2)$ can be defined as the cost minimising the optimisation problem:

$$\begin{aligned} \min_{T \in \mathbb{R}_{\geq 0},\, u:[0,T] \longrightarrow \mathbf{u}} \quad & J(x) \\ \text{s.t.} \quad & \dot{x}(t) = f\left(x(t), u(t)\right), \forall t \in [0,T], \\ & x(0) = z_1, \\ & x(T) = z_2 \end{aligned}$$

Nearest neighbour: Given a graph $G = (V, E)$ and a state $z \in \mathcal{X}$, the procedure `Nearest` returns the closest vertex $v \in V$ to z according to the function `dist`, i.e. $\texttt{Nearest}\,(G, z) = \arg\min_{v \in V} \texttt{dist}\,(v, z)$.

Near-by vertices: Given a graph $G = (V, E)$, a state $z \in \mathcal{X}$ and a number $n \in \mathbb{N}$, the procedure `NearVertices` returns Z_{nearby}, i.e. the set of all the vertices in V that are near to z. For any $z \in \mathcal{X}$, let the function $\text{Reach}\,(z, l) : \left\{ z' \in \mathcal{X} : \texttt{dist}\left(z, z'\right) \leq l \vee \texttt{dist}\left(z', z\right) \leq l \right\}$. The distance threshold $l(n)$ is chosen in such a way that the set $\text{Reach}\,(z, l(n))$ gets smaller as n grows. In [44] this decreasing behaviour is captured with $l(n) = \gamma \log(n)/n$, where γ is a tuning parameter. To sum up, $\texttt{NearVertices}\,(G, z, n) = V \cap \text{Reach}\,(z, l(n))$.

Local steering: Given an initial state $z_1 \in \mathcal{X}$ and a goal state $z_2 \in \mathcal{X}$, the procedure `Steer` returns the optimal trajectory from z_1 until z_2 in some local neighbourhood, i.e. there exists a $\bar{\epsilon} > 0$ such that `Steer` returns a trajectory $x : [0,T] \longrightarrow \mathcal{X}$, with $x(0) = z_1$ and $x(T) = z_2$, the input $u : [0,T] \longrightarrow \mathcal{U}$ that drives the system through the trajectory x and the time T such that $J\left(\texttt{Steer}\,(z_1, z_2)\right) = \texttt{dist}\,(z_1, z_2)\ \forall\ \|z_1 - z_2\| \leq \bar{\epsilon}$.

Collision Check: Given a trajectory $x : [0,T] \longrightarrow \mathcal{X}$ the procedure `ObstacleFree` returns true if the trajectory x lies entirely in $\mathcal{X}_{\text{free}}$, i.e. $x(t) \in \mathcal{X}_{\text{free}}$ holds $\forall\ t \in [0,T]$.

Now that the primitive procedures have been explained, it is possible to proceed with the introduction of the RRT* algorithm. The algorithm is divided in two: Algorithm 1, which represents its essence, and Algorithm 2, which represents the `Extend` procedure. In Algorithm 1, the tree is initialized (line 1) as a graph with z_1 as the only vertex and and empty set of edges. The algorithm iteratively builds a tree that grows by including collision-free trajectories. This is done by first sampling a random pose z_{rand} (line 4) and then extending the tree towards this sample (line 5).

The `Extend` procedure is given in Algorithm 2. The first step is to compute the trajectory x_{new} that steers from the nearest node in the tree, i.e. z_{nearest} obtained with the `Nearest` procedure, until z_{rand} (lines 1 to 4). If this trajectory is collision-free, z_{new} is added to the tree (line 5) and the algorithm proceeds to find the parent of z_{new}. Given a tree $G = (V, E)$, the function $\texttt{Parent} : V \longrightarrow V$ maps a vertex $v \in V$ to the unique vertex $u \in V$ such that $(u, v) \in E$.
To begin with this *parenting* procedure, the unique cost of the new vertex is calculated (line 7). For simplicity, this cost is considered to be an additive function, so that $\texttt{Cost}\,(v) = \texttt{Cost}\left(\texttt{Parent}\,(v)\right) + J\left(\texttt{Steer}\left(\texttt{Parent}\,(v), v\right)\right)$. Then, the near-by vertices are found using the function `NearVertices` (line 8), where the value of n is chosen to be the number of vertices in G at this iteration, i.e. $n = |V|$. With this information, the algorithm initiates a search among the near vertices in Z_{nearby} and computes the cost of steering from each of the near vertices to z_{new}. If there is a

vertex z_{near} that minimizes $\texttt{Cost}\,(z_{\text{new}})$ and satisfies $\texttt{ObstacleFree}\,(x_{\text{near}})$ and $x_{\text{near}}\,(T) = z_{\text{new}}$, then z_{near} is chosen as the new parent of z_{new} (lines 9 to 13). Once the parent of z_{new} is found, the edge $(z_{\text{min}}, z_{\text{new}})$ is added to the tree (line 14).
What is left of Algorithm 2 (lines 15 to 20) is a procedure referred as rewiring. It first steers from z_{new} to the different vertices in the set Z_{nearby}. Then, if the trajectory obtained from z_{new} to z_{near} is collision-free and the cost incurred is lower than the current $\texttt{Cost}\,(z_{\text{near}})$, z_{new} becomes the new parent of z_{near}, i.e. the edge $\left(z_{\text{parent}}, z_{\text{near}}\right)$ is removed from E' (line 19) and the edge $(z_{\text{new}}, z_{\text{near}})$ is added to E' (line 20).

Algorithm 1: The RRT* Algorithm

```
1  V ⟵ {z_init}; E ⟵ ∅, i ⟵ 0;
2  while i < N do
3      G ⟵ (V, E);
4      z_rand ⟵ Sample (i); i ⟵ i + 1;
5      (V, E) ⟵ Extend (G, z_rand);
```

Algorithm 2: The `Extend` Procedure

```
1  V' ⟵ V; E' ⟵ E;
2  z_nearest ⟵ Nearest (G, z);
3  (x_new, u_new, T_new) ⟵ Steer (z_nearest, z);
4  z_new ⟵ x_new (T_new);
5  if ObstacleFree (x_new) then
6      V' := V' ∪ {z_new};
7      z_min ⟵ z_nearest; c_min ⟵ Cost (z_new);
8      Z_nearby ⟵ NearVertices (G, z_new, |V|);
9      for all z_near ∈ Z_nearby do
10         (x_near, u_near, T_near) ⟵ Steer (z_near, z_new);
11         if ObstacleFree (x_near) and x_near (T_near) = z_new and
             Cost (z_near) + J (x_near) < c_min then
12             c_min ⟵ Cost (z_near) + J (x_near);
13             z_min ⟵ z_near;
14     E' ⟵ E' ∪ {(z_min, z_new)};
15     for all z_near ∈ Z_nearby \ {z_min} do
16         (x_near, u_near, T_near) ⟵ Steer (z_new, z_near);
17         if x_near (T_near) = z_near and ObstacleFree (x_near) and
             Cost (z_near) > Cost (z_new) + J (x_near) then
18             z_parent ⟵ Parent (z_near);
19             E' ⟵ E' \ {(z_parent, z_near)};
20             E' ⟵ E' ∪ {(z_new, z_near)};
21 return G' = (V', E')
```

8.5.3 Dynamic system

The explained algorithm shows the necessity of choosing a dynamic system. In [44], three systems are given as example: the Dubins' vehicle, a 2D Double Integrator model and a simple 3D aeroplane model. Another option could be to consider the system described in Chapter 4 or one of the simplified versions of it, e.g. the bicycle model used for controller design. However, this option is discarded because it involves complex nonlinearities that could make the optimization problem very difficult to solve. For simplicity, the Dubins' vehicle is chosen. This subsection introduces it and explains how it is used in the algorithm, i.e. how optimal trajectories are calculated, e.g. to find the nearest neighbour. How the `Steer` procedure is performed is also shown.

1) *System Dynamics:* The system dynamics of the Dubins' vehicle are based on the differential equations of a unicycle model:

$$\begin{aligned} \dot{X} &= v\cos(\psi) \\ \dot{Y} &= v\sin(\psi) \\ \dot{\psi} &= u, \qquad |u| \leq \tfrac{v}{\rho} \end{aligned} \tag{8.2}$$

where X and Y are the Cartesian coordinates of the vehicle in the inertial frame $\mathcal{R}$ described in Section 4.1, ψ is the heading angle, u is the steering input, v is the longitudinal speed and ρ is the minimum turning radius. The velocity and the minimum turning radius are considered constants. The system states are constituted by $z = \begin{bmatrix} X & Y & \psi \end{bmatrix}^T \in \mathbf{x}_{\mathrm{RRT}^*}$, while the input is constituted by $w = \begin{bmatrix} u \end{bmatrix} \in \mathcal{U}$, where $\mathbf{x}_{\mathrm{RRT}^*}$ and $\mathcal{U}$ represent the state-space and the input-space respectively as introduced in Section 8.5.1.

2) *Dubins' curves:* These kind of curves were introduced by L.E. Dubins in [33] and are concatenations of circular arcs and straight lines. More specifically, six families of paths are considered, which are denoted as RSR, LSL, RSL, LSR, RLR and LRL, where each letter L denotes left, R denotes right and S denotes straight. L.E. Dubins demonstrates that the optimal trajectory between two states $z_1, z_2 \in \mathcal{X}$ considering the system described in (8.2), is in fact one of the six aforementioned.

   Due to its simplicity, Dubins' curves are broadly used. For further simplification, the search is restricted to only four families of paths, excluding RLR and LRL, as these are not intuitive paths for a vehicle as the golf cart. In fact, this simplification is adopted following [44], but is known to be a commonly decision taken.

   The length of each of the three segments is denoted as t, p, q for the first, middle, and last segments respectively. For instance, if the RSR path is considered, to express the length of each of its segments, the following notation can be used: $\mathrm{R}_t\mathrm{S}_p\mathrm{R}_q$. Within the algorithm, the addition of these lengths $\mathcal{L} = t + p + q$ is the one measured by the *distance* procedure. In [46], Shkel and Lumelsky present a method for computing these lengths without computing the path itself. The optimization problem then boils down to compare the lengths of each of the four families and pick the shortest one. This approach consists of turning the problem into a canonical form before computation. The canonical form places the initial state z_1 in the origin and rotates to place the final state z_2 on the x-axis.

For the sake of completeness, `Steer` is further detailed now that the dynamic system has been described. In short, the procedure `Steer` (z_1, z_2) can be summarized in two steps. The first step

is to find the optimal Dubins' path between z_1 and z_2, i.e. the path that minimises $\mathcal{L}$. Recall that this procedure is carried out by computing $\mathcal{L}$ for each of Dubins' path using the method proposed by Shkel and Lumelsky in [46]. Then, the optimal path is selected by comparison. This is possible because only four different paths are considered. The second step is to actually compute the Dubins' path. This is done by propagating the model described in (8.2) given the lengths calculated in the previous step.

8.5.4 RRT* Results

An implementation of the presented RRT* algorithm is done based on the work available in [47]. This subsection is devoted to present its results. To prove the implementation satisfactory, a test is performed with two different objectives. The first and most basic is to prove that the algorithm is capable of finding a path. Furthermore, although it is not considered a requirement according to Section 8.1, it is interesting to analyse the asymptotic optimality of the RRT* claimed in [44]. To do so, a maze is designed where given an initial pose z_{init} the RRT* tries to reach a specified goal region $\mathcal{X}_{\text{goal}}$. The goal region is specified with a pose z_{goal}, a distance tolerance of 0.5 m and an orientation tolerance of 1°. Notice that the minimum turning radius is set to $\rho = 1$ and the parameter γ from the `NearVertices` procedure is set to $\gamma = 60$.

The test is run for 2000 iterations. Throughout the test, the algorithm reaches $\mathcal{X}_{\text{obs}}$ multiple times. Figure 8.9 illustrates that behaviour, showing from Figures 8.9(a) to 8.9(c) the path solution after 650, 1000 and 2000 iterations respectively. These representations prove that the algorithm is capable of reaching the goal region with the specified tolerances.

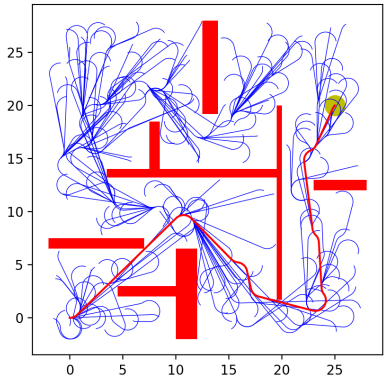
Notice that the algorithm keeps producing solution paths but only returns the path that minimises the `Cost` after the i^{th} iteration. To illustrate such behaviour, the evolution of the `Cost` over iterations is shown in Figure 8.10.

The results presented prove that the algorithm is capable of finding a solution path and hence the implementation is deemed satisfactory. Furthermore, from Figure 8.10 it is possible to perceive how the cost decreases with increasing iterations. This shows an asymptotic optimality which exemplifies that the implementation is correct. In any case, it has been mentioned at the beginning of Section 8.5 that there is no interest in proving the asymptotic optimality of RRT* but it is taken as an additional feature.

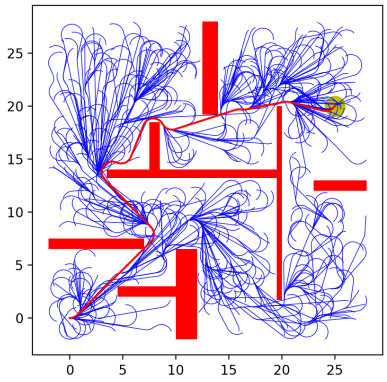
Now that the implementation is proven satisfactory, it is deemed interesting to analyse its performance. To do so, the previous test is ran 4 times. The result is shown in Figure 8.11 where it is possible to appreciate the random behaviour of the algorithm. The most valuable conclusion that can be drawn is that the `Cost` does not decrease significantly after 600 s and that in the four runs the largest `Cost` decrease seems to happen before 300 s.

To show the influence of γ, the test shown in Figure 8.10 is repeated with $\gamma = 10$, $\gamma = 30$, $\gamma = 60$, $\gamma = 70$ and $\gamma = 100$. The results of such test are shown in Figure 8.12 from where it can be seen that the similar asymptotic behaviour is expected if γ is chosen between 60 and 100 but for γ values under 30 it is expected to obtain final path solutions with a higher cost.

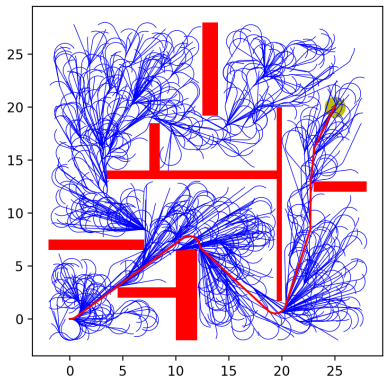
To further exemplify the effect of γ, the resultant path after 2000 iterations when $\gamma = 30$ is shown in Figure 8.13, where it is easy to appreciate the tendency of the path to generate loops.



(a)



(b)



(c)

Figure 8.9: The RRT* is provided with z_{init} and $\mathcal{X}_{\text{goal}}$ within a maze. With this information, the algorithm is run for 2000 iterations. Figures (a) to (c) show the resultant optimal path after 650, 1000 and 2000 iterations respectively. This test has been performed considering $\rho = 1$, and $\gamma = 60$ and shows that the implementation done is able to reach $\mathcal{X}_{\text{goal}}$.

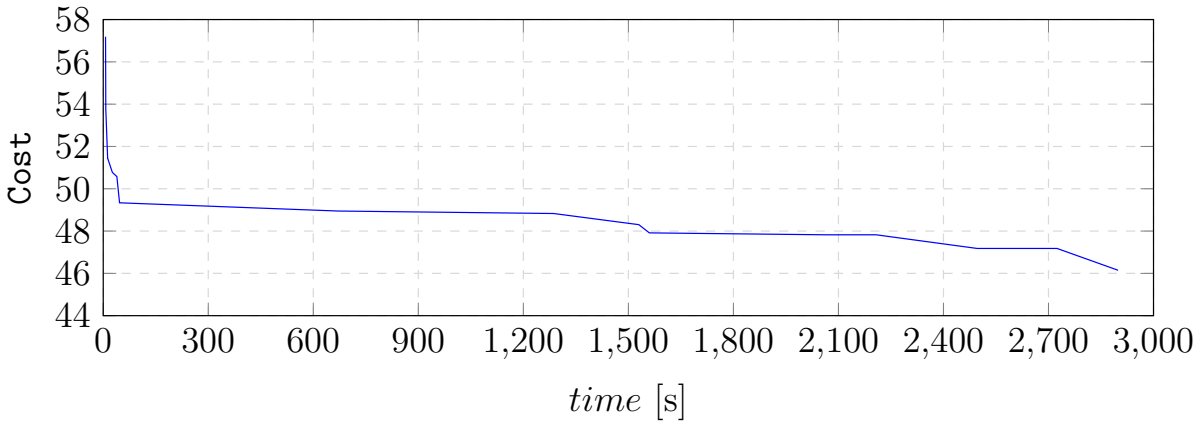


Figure 8.10: The implementation is run for 2000 iterations and it is proven to provide a path. This plot shows the evolution of the `Cost` of the resultant optimal paths over time. It can be seen that the first solutions appear early and that the `Cost` quickly decreases. However, it remains almost stationary for most of the time.

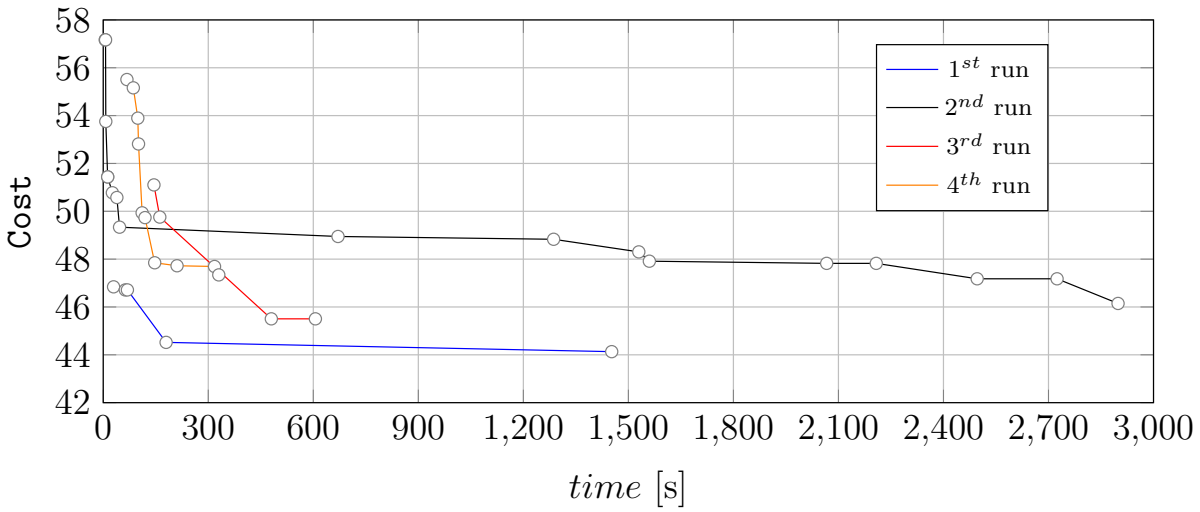


Figure 8.11: The implementation is run four times, each for 2000 iterations to analyse the random behaviour of the algorithm. Regardless of the apparent randomness of the different runs, in all four cases the `Cost` seems to almost reach a stationary cost before 300s.

8.6 Implementation

The path-planner is defined to be RRT* which has been introduced in the previous section. Furthermore, an implementation of it is analysed in Section 8.5.4. This section is devoted to introduce how RRT* is implemented in the given platform. Such implementation is broken down in two steps which are introduced in the following two subsections. The first addresses the representation of the configuration space and how it is obtained. The second details the junction between the path-planning and the controller, i.e. how the reference path generated by the RRT* is fed to the controller.

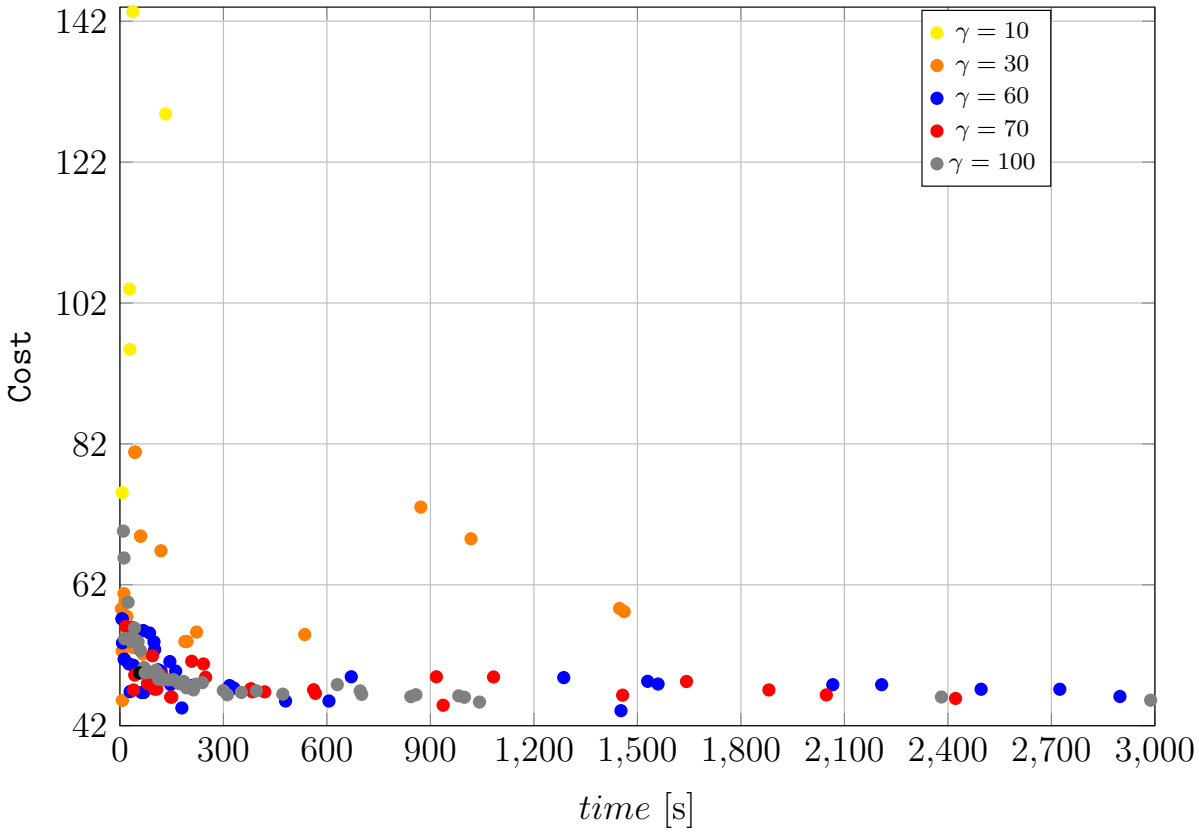


Figure 8.12: Point cloud plot showing the `Cost` evolution through iterations. To take into account the random effect of the algorithm, the implementation is run four times with each $\gamma = 10$, $\gamma = 30$, $\gamma = 60$, $\gamma = 70$.

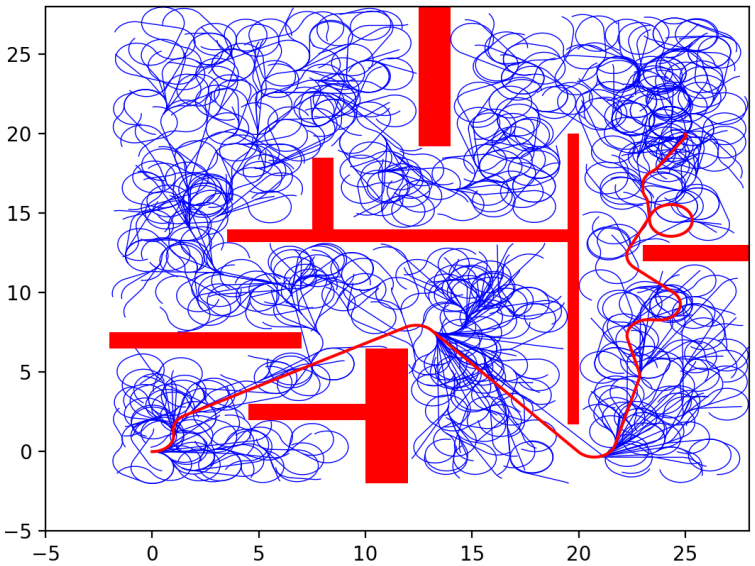


Figure 8.13: Generated path after 2000 iterations with $\gamma = 30$

8.6.1 Representation of the configuration space

Recalling from the problem analysis presented in Section 8.2, a key factor in the path-planning problem is the representation of the configuration space $\mathcal{X}$. In fact, one of the reasons to choose the RRT* approach is that it relies on an implicit representation of $\mathcal{X}$. Now that the RRT* has been presented in detail, the necessity of an occupancy map becomes apparent with the *Collision check* procedure. Furthermore, in Section 8.2.2 it is claimed that such representation is not available for the environment considered, i.e. the AAU campus in Aalborg. Hence, one needs to be constructed.

As discussed in Section 8.2.3, an efficient and scalable solution to construct such a map is using the two LIDARs mounted on the golf cart. Their configuration is also introduced, i.e. one placed horizontally at a "knee" height and the other tilted at the roof of the golf cart.

It is known that there exists another application in mobile robotics that constructs such an occupancy-based map. This application is the Simultaneous Localization and Mapping (SLAM) which addresses the chicken-or-egg problem of constructing and updating a map of an unknown environment while trying to localize the robot within it.
SLAM is a highly recurrent problem in mobile robotics, which makes it a widely studied problem. Therefore, many SLAM approaches are proposed, most of them differring on the perception considered, i.e. LIDARs, monocular cameras or stereo cameras among other strategies. For instance, the SMART research venture which uses a similar platform presents in [48] novel work on mapping 3D environments using SLAM and a 2D tilted LIDAR.

Besides the fact that SLAM could perform the mapping task, it could also provide a more reliable localization than the GPS does. However, SLAM does not have a straightforward implementation and implementing a novel solution is not considered in this work. Alternatively, off-the-shelf SLAM implementations in ROS are investigated. The most popular SLAM implementations in ROS that allow LIDAR perception are hector_slam$^{\text{TM}}$ [49] and gmapping$^{\text{TM}}$ [50]. However, it is chosen to use Cartographer$^{\text{TM}}$ [51], a less popular yet emerging SLAM Library available in ROS.

The main advantage of Cartographer$^{\text{TM}}$ with respect to the other two is that it accepts multiple LIDARs, which allows both the tilted and the horizontal LIDARs to be used. Cartographer$^{\text{TM}}$ is a tool addressed to a wide range of applications and for this reason offers a broad range of configuration options. Part of these configuration options consists of tuning different parameters. The parameters used are given in Appendix E. Furthermore, Cartographer$^{\text{TM}}$ offers the option to include IMU and odometry measurements. From Chapter 2, it is known that IMU measurements are available and thus it is straightforward to use them in Cartographer$^{\text{TM}}$.

The first test is performed indoors in the AAU control laboratory (C2-104), using only IMU measurements and the horizontal LIDAR. The results of this mapping are shown in Figure 8.14, which are accepted as satisfactory as it is able to capture the bounds of the lab. Notice that details such as table legs and other objects present are not well mapped.

Using the same configuration sensors, similar results cannot be obtained outdoors. If only IMU measurements are used, the trajectory produced by Cartographer$^{\text{TM}}$ presents drift over time and the SLAM loop-closure is not able to correct for it. For this reason, it is decided to feed odometry measurements to Cartographer$^{\text{TM}}$.
Odometry measurements are not available directly, and for this reason a kinematic model derived in Appendix F is used. In the model, $V_W = \omega_W R_{\texttt{eff}}$, ω_W is the rotational speed of the wheel obtained from the wheel encoders and $R_{\texttt{eff}}$ is the effective radius of the wheel. To improve the results of this kinematic model, the equation describing $\dot{\psi}$ is removed and the IMU measurement

Figure 8.14: Occupancy map obtained by driving the golf cart through the control laboratory (C2-104) using the Cartographer$^{\text{TM}}$ library from ROS. IMU measurements and the horizontal LIDAR are used to produce this result.

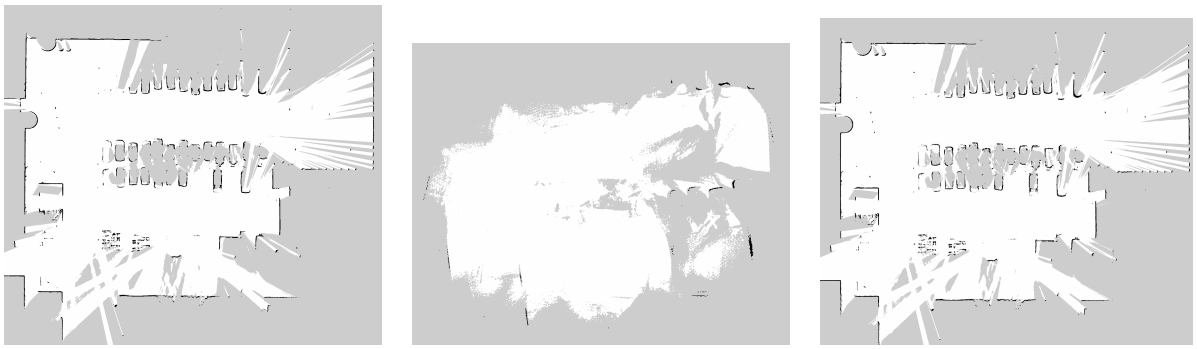
$\dot{\psi}_{\text{IMU}}$ is used instead. The resultant kinematic model used for odometry is represented by (8.3).

$$\begin{aligned} \dot{X} &= \frac{V_W}{\cos(\beta)} \cos\left(\psi_{\text{IMU}} + \beta\right) \\ \dot{Y} &= \frac{V_W}{\cos(\beta)} \sin\left(\psi_{\text{IMU}} + \beta\right) \end{aligned} \tag{8.3}$$

By including odometry measurements, the trajectory drift over time is reduced and the loop-closure of SLAM is able to correct for the remaining drift. Results of mapping outdoors are shown in Figure 8.15 where different LIDAR combinations are used to map a parking lot. In Figure 8.15(a) only the horizontal LIDAR is used, in Figure 8.15(b) only the tilted LIDAR is used and in Figure 8.15(c) both LIDARs are used at the same time. The test in particular has been performed by driving two U-shaped loops, s.t. Cartographer$^{\text{TM}}$ can perform loop-closure.

The results shown in Figure 8.15 are revealing. Figure 8.15(b) shows that the tilted LIDAR alone is not adequate enough to obtain an outdoor map. This is associated to a low resolution of the LIDAR used in the tilted position. The LIDAR used for this purpose is the SICK TIM551-205001, from [52] it is known that it has an angular resolution of 1° and the maximum range is 10 m reduced down to 8 m with 10 % reflectivity. The resolution of the LIDAR in the tilted position is considered critical because most of the laser scans will hit the floor where cartographer is not able to find features. For instance, it will only find features on both sides of the vehicle and if the resolution is low, very few features can be found.
On the other hand, the maps obtained with the horizontal LIDAR, shown in Figure 8.15(a), and using both LIDARs, shown in Figure 8.15(c) might be considered satisfactory at first glance. However, as it is expected, the horizontal LIDAR is not able to perceive certain obstacles. In particular, it is not able to perceive curbs which results critical. It is expected that using both LIDARs can overcome this problem, but again the low resolution of the tilted LIDAR compared to the horizontal one results in almost no additional information on the map, yielding two almost identical maps as it can be seen comparing Figure 8.15(a) and Figure 8.15(c).

To further analyse the quality of these maps, a binary image of Figure 8.15(c) is considered, which can be seen in Figure 8.16. From this map, three problems can be identified. The first



(a) (b) (c)

Figure 8.15: Three occupancy maps from the same parking lot. From left to right, these maps have been performed with Cartographer^{TM} using the horizontal, the tilted and both LIDARs respectively. To obtain these maps outdoors, Cartographer^{TM} is provided with IMU measurements as well as odometry measurements. It can be seen in (b) that the tilted LIDAR cannot be used alone and the similarity of (a) and (c) suggests that the tilted LIDAR does not provide additional information of the environment.

problem is that the current perception is not able to detect curbs and other objects lower than the horizontal LIDARs height. The second problem is that the height of the horizontal LIDAR is critical. This can be seen from the cars in the map, where some are detected as obstacles while others are seen only by their bound, suggesting that some laser scans were able to go under the vehicle. This height is considered fixed and the study of its optimal value is left out of the scope. The third issue and probably the most difficult to appreciate from the map, is that the laser scans are capable of passing glass material, which even though it is not critical in this particular scenario it might be in another environment.

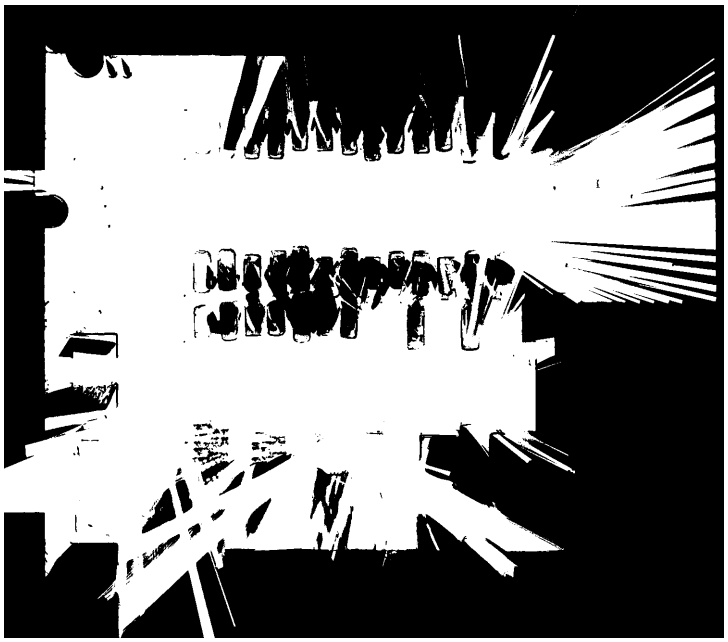


Figure 8.16: Binary representation of the map Figure 8.15(c). It shows how objects lower than the tilted LIDARs height are not perceived. Furthermore, the parked cars are not detected completely detected as objects. In the top left corner of the map, it can be seen how the LIDARs do not detect a continuous wall. This is due to it being a glass door.

This binary image is the visual representation of $\mathcal{X}$, where each pixel represents a cell in the occupancy-based map. If a pixel has value 0, i.e. a black pixel, it belongs to $\mathcal{X}_{\text{obs}}$. This map in particular has been performed with a resolution of $0.05\frac{\text{m}}{\text{pixel}}$.

In conclusion, the results presented show that the perception of $\mathcal{X}$ from Cartographer$^{\text{TM}}$ is not sufficient to provide a satisfactory RRT* representation. This in turn means that it cannot be performed without colliding with a curb or any other obstacle that has not been included in the representation. Alternative solutions for future work are given in the discussion carried out in Chapter 10.

8.6.2 Reference path

To conclude the implementation, the trajectory found by the RRT* must be transformed to be fed to the controller as reference path for it to follow. From Section 6.2, it is known that the controller expects the reference path as two separate references: a curvature reference, referred to as $\kappa(s)$, and a velocity reference, referred to as $\dot{x}_{\text{ref}}(s)$.

The dynamic system considered in RRT* assumes a constant velocity throughout the path. Thus, $\dot{x}_{\text{ref}}(s)$ is simply the same constant velocity used to derive the path. On the other hand, to obtain $\kappa(s)$, RRT* provides the path, referred to as $\sigma(s)$, as a sequence of poses $z = \{X, Y, \psi\}$. Furthermore, RRT* can provide the distances from the Dubins' curves, i.e. the distance travelled between poses, referred to as λ.

The curvature of $\sigma(s)$ considering the unicycle model presented in (8.2) is defined as (8.4).

$$\kappa = \frac{\dot{\psi}}{v} \tag{8.4}$$

To be able to compute (8.4), let $z(k)$ be the k^{th} pose in $\sigma(s)$ and define $\lambda(k)$ as the travelled distance between $z(k)$ and $z(k+1)$. Furthermore, consider the discrete interpretation of $\kappa(s)$ to be the curvature of the segment between $z(k)$ and $z(k+1)$ and define it as $\kappa(k)$. Then (8.4) can be reinterpreted in a discretised manner yielding an expression for $\kappa(k)$ which is shown in (8.5).

$$\kappa(k) = \frac{\psi(k+1) - \psi(k)}{\lambda(k+1)} \tag{8.5}$$

$\kappa(k)$ is supplied to the controller as the curvature reference in a look-up table form together with the travelled distance up until the k^{th} pose. $s(k)$ is calculated as $s(k) = \sum_{i=1}^{k} \lambda(i-1)$ where $\lambda(0) = 0$.

Then the controller uses the algorithm shown in Algorithm 3 to read the look-up table.

Algorithm 3: The RRT* Algorithm

```
if s(k) ≤ s < s(k+1) then
    κ(s) = κ(k)
```

8.7 Navigation planning conclusion

Following the line of thought from the introduction (Chapter 1), it is desired to provide a scalable solution to the navigation planning for future projects that contribute to develop an AMoD service at the AAU campus in Aalborg.
The problem scope is narrowed down to only consider the path-planning problem which is introduced from scratch. First it is formulated, which enables a fair and thorough analysis of it. From this analysis two necessities arise: the need of a path-planner and the need of a representation of the environment where the path-planning is carried out. The scope is further narrowed to use an existing planner, which leads to a state-of-the-art review and the decision to use RRT* with kinodynamic constraints is constituted. The implementation of such planner is carried out. Once tested, it shows satisfactory results, i.e. the implementation is capable of finding a feasible path that does not collide with fixed obstacles. Furthermore, the results show that the cost of the paths that are generated decrease with increasing iterations.
Finally, following the line of thought of developing autonomous navigation with an inexpensive platform, it is decided to construct a representation of the configuration space as an occupancy-based map using the two LIDARs mounted on the golf cart. The construction of such a map is carried out using SLAM. Narrowing down the scope a third and last time, the ROS library named Cartographer^{TM} is chosen to perform SLAM. Although the library produces adequate maps of the environment, some of the essential features for the car's circulation do not appear in the produced maps which is attributed to a lack of resolution of the push-broom LIDAR compared to the horizontal one.

To sum up, this chapter concludes with a validated implementation of RRT* with high improvement potential and a solution to map the environment that does not produce satisfactory maps, but it is thought to be improvable with slight changes in the perception without compromising the cost of the platform. This and further development is later on discussed in Chapter 10.

Part IV

Conclusion & Discussion

9 Conclusion

As stated in Section 1.2, the intention of this thesis is to derive a solution to the problem: *to bring a vehicle from its location to a desired one*. Furthermore, this must be done without compromising the equipment cost.

In Chapter 3, the problem presented is divided into two big subproblems. The first, labelled autonomous driving, addresses the problem of driving a vehicle without human intervention. The other one, labelled navigation, considers the objective of finding a way from the vehicle's location to the desired position.

To fulfil the first subproblem's objective, a first-principles model of the cart is derived. In Chapter 4, a full dynamic model of the golf cart is found for simulation purposes. By the means of model simplifications, a less complex model is derived for the controller design. In order to have a full model of the platform, the actuators of the cart are modelled in Chapter 5. The control objective is decided to be the path-following approach in Section 6.2. Two nonlinear controllers are designed in Section 6.3 to approach the path-following problem. A controller for the longitudinal dynamics is derived with the Lyapunov redesign method. This controller is shown to work in simulation under an uncertainty in the model parameters. Another one for the lateral dynamics is derived based on the backstepping method, and is shown to work in a simulation environment. To implement the derived controllers in the real platform, they are adapted with the actuator models in Section 6.4, and shown to be succesful in simulation. To provide the controllers with feedback, an EKF based on the sensors present in the platform and the model derived is implemented. When tested in the platform, the overall solution does not fulfil the objective of autonomous driving.

The second subproblem is narrowed down to only consider the path-planning problem which is introduced from scratch. After a thorough analysis of the problem, two necessities arise: the need of a path-planner and the need of a representation of the environment where the path-planning is carried out. A state-of-the-art review leads to the decision of using RRT* with kinodynamic constraints. The implementation of such planner is carried out and tests show satisfactory results, i.e. the implementation is capable of finding a feasible path that does not collide with obstacles. The urban environment and its features are captured in an occupancy-based map that is build using the two LIDARs mounted on the golf cart. The construction of such a map is carried out using the ROS library named Cartographer$^{\text{TM}}$. Although the library produces adequate maps of the environment, some of the essential features for the cart's circulation are not captured. This is attributed to a lack of resolution of the push-broom LIDAR compared to the horizontal one.

The two subsystems are not tested together. However, the solutions presented in both cases are meant to work in a coupled manner. Judging their decoupled behaviour of, their blending is expected to work.

10 Discussion

In this chapter, the outcome of this thesis is discussed. It is done in a manner that addresses the overall goal of it as well as how it sets a basis for future work to realise the goal of AAUs AMoD service.
For this reason, the discussion is broken down into the two subproblems that have been the common theme throughout the report. The problem of autonomous driving is addressed in Section 10.1, where the vehicle model, controller design and sensor fusion technique are discussed. Afterwards, the navigation problem is reviewed in Section 10.2 where the chosen path-planner algorithm and mapping approach are discussed.

10.1 Autonomous driving

Recalling from Chapter 3, the subproblem of autonomous driving comes from the statement *to bring a vehicle*. This problem has been approached with a vehicle model, a model-based control structure and sensor fusion for state feedback.

10.1.1 Modelling

In Chapter 4, an first-principles model of the system has been presented. One of the main reasons for it is that the expected velocities of the vehicle are near the limit where modelled effects, such as lateral slip, become relevant. Another reasoning for this model is to provide a strong simulation model for future work on the platform.
While the intentions behind this model are well reasoned, throughout the project there have been revelations exemplifying that a dynamic model may not have been the ideal choice. Evidence of this is the fact that one is not able to set T_m and δ references to the platform. This can be further embodied as knowing the platform better. While the vehicle model is detailed, the actuator models developed in Chapter 5 are not of the same standard. Unfortunately, this means that the overall platform model developed is restricted by the performance of these models.
Nevertheless, if future work includes more accurate models of the actuators or direct access to manipulate the inputs, T_m and δ, this model can prove indispensable towards providing AAU with an AMoD service.

10.1.2 Control structure

The chosen control structure is presented in Chapter 6, where the control objective is chosen as path-following. Furthermore, it is decided to implement nonlinear controllers in the platform, for academic purposes. One of the key statements from the state-of-the-art is that the nonlinear controllers for similar vehicles are only implemented in simulation. This gives further motivation to design nonlinear controllers as this work offers the opportunity to evaluate the performance of nonlinear controllers in a vehicle.
The designed control structures are designed with the simplified dynamic model. This also provides further reasoning to the dynamic model being more complex than necessary for the purpose of achieving autonomous driving.
In Chapter 6, the control objective is decoupled into one for the longitudinal velocity and another for the lateral dynamics. Similarly the discussion is decoupled.

Longitudinal velocity

The chosen nonlinear control method for the longitudinal velocity is Lyapunov redesign. This controller proves to be satisfactory in simulation as well as being robust to variations in the model parameters. In order to implement this controller in the platform, a cascade structure is used. In the inner loop of the cascade structure, a PID controller is implemented. This is done to ease tuning during testing as the drive-train model is not considered accurate enough. Unfortunately, with an inadequate performance of the servo controller the capabilities of the nonlinear one are restricted.

Lateral dynamics

To control the lateral dynamics of the vehicle, the backstepping control method is chosen. The designed controller is proven to successfully control these dynamics in simulation yet similar issues present themselves when implementing the algorithms on the platform. Again, problems regarding the actuator model restrict the performance of the lateral dynamics controller. It is also concluded that the implemented controller is unstable. An explanation for this is that the controller is not robust enough to overcome model changes. This could be overcome by redesigning the already existing controller to be more robust or focusing directly on a robust control structure.

Overall, the difficulty of implementing nonlinear controllers in a platform is shown. While showing the nonlinear controllers to work in a simulated environment, their behaviour when implemented on the platform is not proven to work. In future work, the types of controllers investigated can be chosen if better knowledge of the platform is available. These future control structure could be designed to provide references directly to the system, or the actuator models could be further detailed for the cascaded structure to be designed.

10.1.3 Sensor fusion

The development of the sensor fusion is presented in Chapter 7, with an EKF chosen as the algorithm for it. This decision is deemed appropriate to estimate the states of the platform. This is further substantiated by the results shown within the section. While the high fidelity model is used for EKF, the model fitted with the parameter estimation, in Section 4.5, is the simpler bicycle model. This illustrates that the model does not have to be extremely accurate for it to be used in an EKF. It is interesting for future work to investigate how accurate EKF results can be achieved with a model that deviates further from the platform. This can reduce computational power, which may become more relevant at a later stage in the AAU AMoD project, when efficiency of the algorithms on the platform are probed.
It can also be interesting to include further sensors in the fusion process, to provide better estimates of the vehicle state. Another suggestion is to investigate different conversions of the GPS coordinates to Cartesian coordinates. This may provide better estimates of the inertial states of the system.

Overall, this work is dedicated in most part to investigate the performance of nonlinear controllers applied in a real platform. The uncertainty around the actuator models inhibits the performance of the controllers. It is considered that in an ideal platform where the inputs are the same of the high fidelity model, they would present satisfactory performance as the one shown in simulation.

10.2 Navigation

The purpose of this section is to provide a solution to the second subproblem from Chapter 3, which is formulated as *from its location to a desired one*. The problem is scoped down due to the novelty of this field to the authors.

10.2.1 Navigation planning

The knowledge shared on the topic of navigation is explained in Chapter 8, where it is further analysed and broken down into route-planning and path-planning. A thorough state-of-the-art on path-planners is presented. This is primarily done to the advantage of the authors but it also lays the foundation for future projects requiring knowledge about path-planning. This foundation has been cemented in this work, thus future work can focus on improving the chosen algorithms as well as integrating them into the vehicle.
The primary work done is to develop a path-planner using the RRT* with kinodynamic constraints method. Furthermore, to implement this solution, an occupancy map is built using the LIDARs of the platform. The discussion concerning Chapter 8 shall also be split into these these subtopics, below.

RRT*

The knowledge gained regarding path-planners is built from the ground up, with the focus of this work being to replicate the work done in [44]. It is shown in Section 8.5 that the RRT* path-planner is successfully implemented. That being said, to be suitable for AAUs AMoD service, improvements to the algorithm should be made. A suggested improvement for RRT* is to investigate other propagation models, e.g. incorporate using variable velocities into the planner as opposed to the current implementation where the velocities are kept constant.

Occupancy map

The occupancy map in this project is obtained using SLAM and the LIDARs. The ROS package Cartographer^{TM} is used for SLAM. It is seen that generating a map outdoors from SLAM is possible but not to the level of accuracy sufficient to perform the RRT* algorithm. With this being said, Cartographer^{TM} may have been a great implementation to demonstrate the capabilities that LIDARs have in an outdoors environment, which has been a controversial topic of the field. Unfortunately, the impact of the tilted LIDAR using Cartographer^{TM} is negligible. To further investigate whether this LIDAR is suitable to create the occupancy map, the work done by SMART using the tilted LIDAR may be replicated. In such a project, it can also be interesting to test LIDARs with better resolution and range than the current LIDAR. However, the results shown suggest that the perception solution can be kept inexpensive.

Bibliography

Bibliography

[1] S. Pendleton, T. Uthaicharoenpong, Z. J. Chong, Guo Ming James Fu, B. Qin, Wei Liu, Xiaotong Shen, Zhiyong Weng, C. Kamin, M. A. Ang, L. T. Kuwae, K. A. Marczuk, H. Andersen, Mengdan Feng, G. Butron, Z. Z. Chong, M. H. Ang, E. Frazzoli, and D. Rus. Autonomous golf cars for public trial of mobility-on-demand service. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1164–1171, Sept 2015. doi: 10.1109/IROS.2015.7353517.

[2] G. Liu, M. Pan, Y. Li, Z. L. Zhang, and J. Luo. Modeling urban trip demands in cloud-commuting system: A holistic approach. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 857–862, May 2017. doi: 10.1109/INFCOMW.2017.8116488.

[3] Marco Pavone. *Autonomous Mobility-on-Demand Systems for Future Urban Mobility*, pages 399–416. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. ISBN 978-3-662-45854-9. doi: 10.1007/978-3-662-45854-9_19. URL `https://doi.org/10.1007/978-3-662-45854-9_19`.

[4] World class traffic jam 2. URL `https://www.flickr.com/photos/joiseyshowaa/7454479488/`.

[5] MIT. Future urban mobility. URL `https://ares.lids.mit.edu/fm/`.

[6] SMART. Fact sheet: Smart autonomous vehicle. URL `https://smart.mit.edu/news-events/news/fact-sheet-smart-autonomous-vehicle`.

[7] Yamaha g29-e dc owner's manual, . URL `https://www.yamahagolfcar.com/ownersmanualdownload/`.

[8] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA Urban Challenge*. 2009.

[9] Rajesh Rajamani. *Vehicle Dynamics and Control*. 2012.

[10] J.L. Meriam and L.G. Kraige. *Engineering Mechanics volume 2 (Dynamics)*. Seventh edition edition, 2012.

[11] J. Franch and J. M. Rodriguez-Fortun. Control and trajectory generation of an ackerman vehicle by dynamic linearization. In *2009 European Control Conference (ECC)*, pages 4937–4942, Aug 2009. doi: 10.23919/ECC.2009.7075182.

[12] L. Nehaoua and L. Nouvelière. Backstepping based approach for the combined longitudinal-lateral vehicle control. In *2012 IEEE Intelligent Vehicles Symposium*, pages 395–400, June 2012. doi: 10.1109/IVS.2012.6232257.

[13] H. Zhou, L. Güvenç, and Z. Liu. Design and evaluation of path following controller based on mpc for autonomous vehicle. In *2017 36th Chinese Control Conference (CCC)*, pages 9934–9939, July 2017. doi: 10.23919/ChiCC.2017.8028942.

[14] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099, June 2015. doi: 10.1109/IVS.2015.7225830.

[15] Yamaha g29-e dc datasheet, . URL `https://www.yamaha-motor.eu/dk/produkter/golfbiler/golf-fleet/g29e-48-volt.aspx?view=featurestechspecs`.

[16] William F. Milliken and Douglas L. Milliken. *Race Car Vehicle Dynamics*. 1995.

[17] J. David Powell Gene F. Franklin and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems*. Seventh edition edition, 2015.

[18] A. P. Aguiar and J. P. Hespanha. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *IEEE Transactions on Automatic Control*, 52(8):1362–1379, Aug 2007. ISSN 0018-9286. doi: 10.1109/TAC.2007.902731.

[19] A. Pedro Aguiar, Dragan B. Dačić, João P. Hespanha, and Petar Kokotović. Path-following or reference tracking?: An answer relaxing the limits to performance. *IFAC Proceedings Volumes*, 37(8): 167 – 172, 2004. ISSN 1474-6670. doi: https://doi.org/10.1016/S1474-6670(17)31970-5. URL `http://www.sciencedirect.com/science/article/pii/S1474667017319705`. IFAC/EURON Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal, 5-7 July 2004.

[20] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat. A model predictive control approach for combined braking and steering in autonomous vehicles. In *2007 Mediterranean Conference on Control Automation*, pages 1–6, June 2007. doi: 10.1109/MED.2007.4433694.

[21] R. Mohajerpoor, S. Salavati Dezfuli, and B. Bahadori. Teleoperation of an unmanned car via robust

adaptive backstepping control approach. In *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 1540–1545, July 2013. doi: 10.1109/AIM.2013.6584314.

[22] Rongrong Wang, Chuan Hu, Fengjun Yan, M. Chadli, and Nan Chen. Should the desired vehicle heading in path following of autonomous vehicles be the tangent direction of the desired path? In *2015 American Control Conference (ACC)*, pages 489–494, July 2015. doi: 10.1109/ACC.2015.7170783.

[23] R. Skjetne and T. I. Fossen. Nonlinear maneuvering and control of ships. In *MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No.01CH37295)*, volume 3, pages 1808–1815 vol.3, 2001. doi: 10.1109/OCEANS.2001.968121.

[24] Joanna Płaskonka. Different kinematic path following controllers for a wheeled mobile robot of (2,0) type. *Journal of Intelligent & Robotic Systems*, 77(3):481–498, Mar 2015. ISSN 1573-0409. doi: 10.1007/s10846-013-9879-6. URL `https://doi.org/10.1007/s10846-013-9879-6`.

[25] Hassan K. Khalil. *Nonlinear Systems*. Third edition edition, 2000.

[26] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *2011 IEEE International Conference on Rehabilitation Robotics*, pages 1–7, June 2011. doi: 10.1109/ICORR.2011.5975346.

[27] R. Mahony, T. Hamel, and J. M. Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218, June 2008. ISSN 0018-9286. doi: 10.1109/TAC.2008.923738.

[28] Simon Haykin. *Kalman Filtering and Neural Networks*. 2001.

[29] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering: Theory and Practice with MATLAB*. Wiley-IEEE Press, 4th edition, 2014. ISBN 1118851218, 9781118851210.

[30] Timothy Sauer. *Numerical Analysis*. Second edition edition, 2012.

[31] G. G. Scandaroli and P. Morin. Nonlinear filter design for pose and imu bias estimation. pages 4524–4530, 2011.

[32] A. Scheuer and T. Fraichard. Planning continuous-curvature paths for car-like robots. In *Intelligent Robots and Systems '96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 3, pages 1304–1311 vol.3, Nov 1996. doi: 10.1109/IROS.1996.568985.

[33] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3): 497–516, 1957.

[34] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505, Mar 1985. doi: 10.1109/ROBOT.1985.1087247.

[35] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968. ISSN 0536-1567. doi: 10.1109/TSSC.1968.300136.

[36] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1 (1):269–271, 1959.

[37] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Practical search techniques in path planning for autonomous driving. *Ann Arbor*, 1001(48105):18–80, 2008.

[38] Dmitri Dolgov and Sebastian Thrun. Autonomous driving in semi-structured environments: Mapping and planning. pages 3407–3414, 2009.

[39] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 29(5):485–501, 2010.

[40] Steven M. LaValle and Jr. James J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001. doi: 10.1177/02783640122067453. URL `https://doi.org/10.1177/02783640122067453`.

[41] John Leonard, Jonathan How, Seth Teller, Mitch Berger, Stefan Campbell, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Sertac Karaman, et al. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008.

[42] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104:2, 2010.

[43] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[44] Sertac Karaman and Emilio Frazzoli. Optimal kinodynamic motion planning using incremental

sampling-based methods. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 7681–7687. IEEE, 2010.

[45] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1478–1483. IEEE, 2011.

[46] Andrei M Shkel and Vladimir Lumelsky. Classification of the dubins set. *Robotics and Autonomous Systems*, 34(4):179–202, 2001.

[47] Atsushi Sakai. Rrtstardubins. URL `https://github.com/AtsushiSakai/PythonRobotics/tree/master/PathPlanning/RRTStarDubins`.

[48] ZJ Chong, Baoxing Qin, Tirthankar Bandyopadhyay, Marcelo H Ang, Emilio Frazzoli, and Daniela Rus. Mapping with synthetic 2d lidar in 3d urban environment. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4715–4720. IEEE, 2013.

[49] Stefan Kohlbrecher and Johannes Meyer. hector_slam. URL `http://wiki.ros.org/hector_slam`.

[50] Brian Gerkey. gmapping. URL `http://wiki.ros.org/gmapping`.

[51] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1271–1278. IEEE, 2016.

[52] Tim551-2050001 product page. URL `https://www.sick.com/dk/en/detection-and-ranging-solutions/2d-lidar-sensors/tim5xx/tim551-2050001/p/p343045`.

[53] Ni compactrio crio-9037. URL `http://www.ni.com/da-dk/support/model.crio-9037.html`.

[54] Robot operating system lunar loggerhead. URL `http://wiki.ros.org/lunar`.

[55] Ubuntu 16.04 lts. URL `https://www.ubuntu.com/download/desktop`.

[56] Arduino uno. URL `https://store.arduino.cc/arduino-uno-rev3`.

[57] Arduino ethernet shield. URL `https://store.arduino.cc/arduino-ethernet-shield-2`.

[58] Netgear prosafe datasheet, . URL `http://www.downloads.netgear.com/files/GDC/datasheet/en/GS105v5-GS108v4.pdf`.

[59] Netgear prosafe image, . URL `https://www.amazon.com/NETGEAR-Ethernet-Lifetime-Replacement-Unmanaged/dp/B00MPVR50A`.

[60] Maxon escon 70/10 servo controller datasheet, . URL `https://www.maxonmotor.com/maxon/view/product/control/4-Q-Servokontroller/422969`.

[61] Maxon actuator. URL `https://www.maxonmotor.com/maxon/view/service_search?query=496553`.

[62] Maxon motor image, . URL `https://www.maxonmotor.com/maxon/view/product/motor/dcmotor/re/re50/370356`.

[63] Cahb-21 linear actuator, . URL `http://www.skf.com/my/products/actuation-systems/linear-actuators/cahb-series/cahb-21/index.html`.

[64] Cahb-21 linear actuator image, . URL `https://uk.rs-online.com/web/p/electric-linear-actuators/8855316/`.

[65] Lms151-10100 product page. URL `https://www.sick.com/dk/en/detection-and-ranging-solutions/2d-lidar-sensors/lms1xx/lms151-10100/p/p141840`.

[66] Garmin gps 19x hvs datasheet, . URL `http://static.garmin.com/pumac/GPS19x_HVS_INST_ML.pdf`.

[67] Garmin gps 19x hvs image, . URL `https://marinestore.co.uk/Garmin_19x_Antenna_HVS_NMEA_0183.html`.

[68] Sparkfun mpu-6050 imu breakout. URL `https://www.sparkfun.com/products/11028`.

[69] Trojan t-875 battery specifications. URL `http://www.trojanbattery.com/product/t-875/`.

[70] Movable type scripts. URL `https://www.movable-type.co.uk/scripts/latlong.html`.

Appendices

A Component description

In this chapter, the technical information about the components on the vehicle platform are given. The information here is relevant to ease the use of the platform for further projects. All components are are referenced such that further information about them can be obtained through these.

A.1 Processing system

Low Level Computer

The Low Level Computer (LLC) is a cRIO-9037 CompactRIO Controller from National Instruments [53] shown in Figure A.1. It runs NI Linux Real-Time OS.

Figure A.1: cRIO-9037 CompactRIO [53]

High Level Computer

The High Level Computer (HLC) is a computer running Robot Operating System (ROS) Lunar Loggerhead [54] on Ubuntu 16.04 [55]. To interface with the LLC a ROS node written by Karl Damkjær Hansen of Aalborg University is used. It is known as the *tpod_driver*, this node is responsible for communicating with the LLC through a User Datagram Protocol (UDP) connection.

Microcontroller

The microcontroller on the vehicle is an Arduino Uno [56] including Ethernet shield [57] can be seen, respectively, in Figure A.2 and Figure A.3. The Ethernet shield is included to allow communication between attached sensors and the HLC.

Figure A.2: Arduino Uno [56]

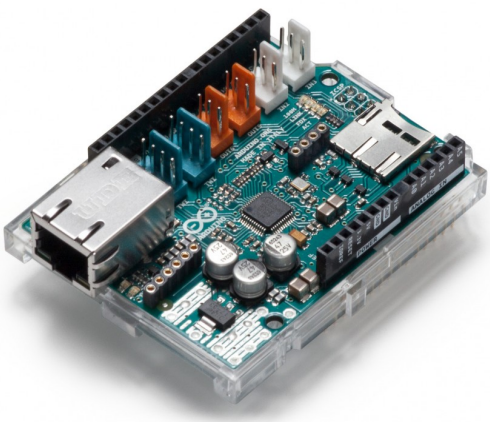


Figure A.3: Arduino Ethernet Shield [57]

Switch

The switch used is the NETGEAR ProSAFE GS108E [58] and can be seen in Figure A.4. It is an unmanaged switch that handles the communications between the devices and sensors that require Ethernet connections.

Figure A.4: NETGEAR ProSAFE GS108E [59]

A.2 Actuation system

The Yamaha vehicle set-up is equipped with a 48 V DC motor such that the vehicle can be manually driven. For the vehicle to be driven autonomously, some modifications to the existing set-up have been done. The vehicle's drive-train has a 48 V battery, a proprietary Yamaha motor controller, a 48 V DC motor and a gearbox. In the original set-up, the drive-train is controlled by the driver through the throttle pedal. When driving autonomously, the drive-train is then controlled by inputs set through the LLC mimicking the accelerator pedal and they can be set by a self-driving controller. Additionally, two additional actuators were installed to control the steering and the braking.

Steering system

The steering system consists of a MAXON ESCON 70/10 Servo Controller [60] and MAXON Actuator 496553 [61], shown in Figure A.5 and Figure A.6 respectively. This Actuator is coupled to the steering column of the vehicle using 1:1 gears and a timing belt. The use of the timing belt is such that the actuator is not rigidly connected to the steering column. This is helpful if

any accidents occur, the timing belt will help to dampen the vibrations created; thus reducing possible damage to the actuator. The actuator also includes a rotary encoder.

Figure A.5: MAXON ESCON 70/10 Servo Controller [60]

Figure A.6: MAXON Actuator 496553 [62]

Braking system

The braking system consists of a MAXON ESCON 70/10 Servo Controller [60], shown in Figure A.5, and the SKF linear actuator CAHB-21-B3N-102431-AAALP0-000 [63], shown in Figure A.7.

Figure A.7: SKF linear actuator CAHB-21-B3N-102431-AAALP0-000 [64]

The linear actuator is tied to the braking pedal with a cable using pulleys to set the braking pedal to a desired position. This also allows the driver to manually override the brakes in an emergency situation. This actuator also includes an encoder on it, similarly to the steering system.

Drive-train system

The vehicle drive-train consists of a 48 V battery, a proprietary Yamaha motor controller, a 48 V DC motor and a gearbox with a 1 : 11.34 gearing ratio. In the original set-up, the drive-train is controlled by the driver through the throttle pedal. When driving autonomously, the drive-train is then controlled by inputs set through the LLC mimicking the accelerator pedal sending an analogue voltage signal to the motor controller. From [7] it is known that the motor controller in turn sends a Pulse-Width Modulation (PWM) signal to the DC motor.

A.3 Sensor system

Front facing LIDARs

There are two front facing LIDARs present on the system. The two LIDARs are the SICK LMS151-1010 [65] and SICK TIM551-205001 [52], shown in Figure A.8 and Figure A.9 respectively. The LMS151-1010 is mounted at the front of the vehicle in place of its bumper horizontally while the TIM551-205001 is mounted on the roof of the vehicle ain a push-broom configuration.

Figure A.8: SICK LMS151-1010 [65]

Figure A.9: SICK TIM551-205001 [52]

GPS

The GPS installed on the vehicle is the GARMIN GPS 19x HVS [66]. The GPS uses the RS232 communication standard to parse the GPS messages. A USB-RS232 cable is used to connect the GPS directly to the HLC. A ROS node is then able to read and interpret the relevant NMEA strings. The GPS can be seen in Figure A.10.

Figure A.10: GARMIN GPS 19x HVS [67]

IMU

The IMU used is the MPU-6050 Breakout [68], shown in Figure A.11, developed by SparkFun. It is a low cost 6-axis IMU. The IMU interfaces to the Arduino Uno via I2C. The Arduino Uno in turn broadcasts a UDP message which can then be interpreted by a ROS node on the HLC.

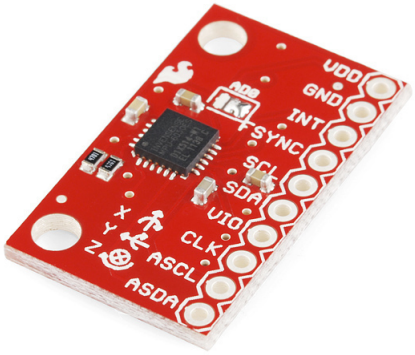


Figure A.11: MPU-6050 Breakout [68]

Wheel encoder

On the vehicle, there are two encoders. One on each of the rear wheels. The encoders are connected to the LLC and through ROS the HLC is able to attain the number of ticks on each wheel. These specific encoders have been designed to give 400 ticks per revolution. As one can not ensure the wheels are correctly balanced, the number of ticks per revolution may vary.

A.4 Power supply unit

The vehicle is powered by six 8V Trojan-T875 [69], shown in Figure A.12, in series. This makes up the 48V power supply needed to power the vehicle. There is also a DC-DC converter that converts the 48V power supply to 12V for all the additional sensors and components.

Figure A.12: Trojan-T875 Battery [69]

B Control methods

This appendix provides the derivation of the longitudinal controller in Section 6.3.1 and the lateral controller in Section 6.3.2. In each section of the appendix is shown the procedure used during each derivation. First, a generic explanation of the Lyapunov redesign and the backstepping methods is provided. Finally, the particular solution used for the golf cart controller is derived based on the stated generic method.

B.1 Lyapunov redesign

Knowing that the golf cart model is sensible to changes in the parameters describing it, a control algorithm capable of stabilizing the system even with uncertainty in the parameters should be designed. The method chosen for designing such an algorithm is the Lyapunov redesign.
Briefly stated, the Lyapunov redesign method consists in adapting or, as the name states, redesigning a control law designed for a nominal version of the system such that the possible uncertainties present in the real system are taken into consideration.

The following description of the Lyapunov redesign method is based on the one explained in [25]. Consider a real system to be described as shown in (B.1).

$$\dot{x} = f(x) + G(x)u \tag{B.1}$$

If all of the uncertainties in the model are gathered in an input term, $\delta(x, u)$, the real system (B.1) can be described by (B.2).

$$\dot{x} = f(x) + G(x)u = \hat{f}(x) + \hat{G}(x)\left[u + \delta(x, u)\right] \tag{B.2}$$

The denotation $\hat{\bullet}$ describes the nominal value of the parameters, specified for a nominal description of the system (B.3).

$$\dot{x} = \hat{f}(x) + \hat{G}(x)u \tag{B.3}$$

First of all, a stabilizing feedback controller is designed for the nominal system.
Using the Lyapunov stability theorem, if there exists a Lyapunov candidate, V, such that $V > 0$ and $V = 0$ in the origin, the system is stable if the time derivative of the function, $\dot{V}$, fulfils $\dot{V} < 0$, and asymptomatically stable if $\dot{V} = 0$ only at the origin.

Following [25], suppose the proposed Lyapunov function satisfies the inequalities (B.4) and (B.5), where α_i are class $\mathcal{K}$ functions and $\psi(x)$ is the feedback law for the nominal system.

$$\alpha_1\left(\|x\|\right) \leq V(x) \leq \alpha_2\left(\|x\|\right) \tag{B.4}$$

$$\frac{\partial V}{\partial x}\left[f(x) + G(x)\psi(x)\right] \leq -\alpha_3\left(\|x\|\right) \tag{B.5}$$

Using the control input $u = \psi(x) + v$, if the uncertainty term δ satisfies the inequality (B.6), the additional input v can be designed such that the real system (B.1) is stabilised.

$$\left\|\delta(x, \psi(x) + v)\right\| \leq \rho(x) + \kappa_0\left\|v\right\|, \quad 0 \leq \kappa_0 < 1 \tag{B.6}$$

It can be seen that when taking derivative of the Lyapunov candidate along the true trajectories of the system its expression is left as (B.7).

$$\dot{V} = \frac{\partial V}{\partial x}(f + G\psi) + \frac{\partial V}{\partial x}G(v + \delta) \leq -\alpha_3(\|x\|) + \frac{\partial V}{\partial x}G(v + \delta) \tag{B.7}$$

If a new variable $w^T = \frac{\partial V}{\partial x}G$ is defined, (B.7) is left as (B.8), which shows how a proper choice of v will allow the controller to cancel the destabilizing effect of δ on $\dot{V}$.

$$\dot{V} \leq -\alpha_3(\|x\|) + w^T v + w^T \delta \tag{B.8}$$

The last term of (B.8) is known to be bounded according to (B.6) and as a consequence v can be designed according to this bound in order to guarantee that $\dot{V} \leq 0$.
Using the inequality (B.6) taking the $\|\bullet\|_2$, v can be taken as (B.9), with η being a non-negative function such that $\eta(x) \geq \frac{\rho(x)}{1-\kappa_0}$.

$$v = -\eta(x)\frac{w}{\|w\|_2} \tag{B.9}$$

This definition of $\eta(x)$ and therefore of v, yields the derivative of the Lyapunov function V along the trajectories of the true system to be negative definite, from what is shown in (B.10).

$$\begin{aligned} w^T v + w^T \delta &\leq w^T v + \|w\|_2 \|\delta\|_2 \\ &\leq w^T v + \|w\|_2 \left[\rho(x) + \kappa_0 \|v\|_2\right] \\ &\leq -\eta(x)\|w\|_2 + \rho(x)\|w\|_2 + \kappa_0\eta(x)\|w\|_2 \\ &\leq -\rho\|w\|_2 + \rho\|w\|_2 = 0 \end{aligned} \tag{B.10}$$

However, the control law described by v is a discontinuous function when $\|w\|_2 = 0$, which may lead to theoretical problems. Therefore, the control law is changed to be the continuous function shown in (B.11).

$$\begin{cases} -\eta(x)\frac{w}{\|w\|_2}, & if \quad \eta(x)\|w\|_2 \geq \varepsilon \\ -\eta^2(x)\frac{w}{\varepsilon}, & if \quad \eta(x)\|w\|_2 < \varepsilon \end{cases} \tag{B.11}$$

The new continuous control law, however, only guarantees $\dot{V}$ to be negative definite when $\eta(x)\|w\|_2 \geq \varepsilon$, satisfying the inequality (B.12) at all times.

$$\dot{V} \leq -\alpha_3\left(\|x\|_2\right) + \frac{\varepsilon(1-\kappa_0)}{4} \tag{B.12}$$

This means that the feedback controller only stabilises the closed-loop system towards an area bounded by ε. In fact, the following is proven in [25]: choose ε as (B.13) and let the control v be given by (B.11).

$$\varepsilon < \frac{2\alpha_3\left(\alpha_2^{-1}(\alpha_1(r))\right)}{1-\kappa_0} \tag{B.13}$$

Then, for any starting point $\|x(t_0)\|_2 < \alpha_2^{-1}(\alpha_1(r))$, where $r > 0$ belongs to the domain, there exists a finite time t_1 such that the closed-loop system satisfies (B.14) and (B.15).

$$\|x(t)\|_2 \leq \beta\left(\|x(t_0)\|_2, t - t_0\right), \quad \forall t_0 \leq t < t_1 \tag{B.14}$$

$$\|x(t)\|_2 \leq b(\varepsilon), \quad \forall t \geq t_1 \tag{B.15}$$

Where β is a class $\mathcal{KL}$ function and b a class $\mathcal{K}$ function. The last one, which bounds the solution, is defined by (B.16).

$$b(\varepsilon) = \alpha_1^{-1}\left(\alpha_2\left(\alpha_3^{-1}\left(\frac{\varepsilon(1-\kappa_0)}{2}\right)\right)\right) \tag{B.16}$$

In conclusion, the redesign given by (B.11) does not stabilise the origin as the discontinuous one given by (B.9) does, but it guarantees ultimate boundedness of the solutions.

Focusing now in the golf cart model, all of the uncertainties are considered to be in its parameters, meaning that the real system described by (B.17)

$$\begin{cases} e_{\dot{x}} &= \dot{x} - \dot{x}_{\text{Ref}} \\ \ddot{x} &= \frac{m}{m_e}\dot{\psi}\dot{y} + \frac{1}{R_{\text{eff}} m_e} T_m = a\dot{\psi}\dot{y} + bT_m \end{cases} \tag{B.17}$$

has a corresponding nominal system described by (B.18), where each parameter takes its nominal value.

$$\begin{cases} e_{\dot{x}} &= \dot{x} - \dot{x}_{\text{Ref}} \\ \ddot{x} &= \frac{\hat{m}}{\hat{m}_e}\dot{\psi}\dot{y} + \frac{1}{\hat{R}_{\text{eff}} \hat{m}_e} T_m = \hat{a}\dot{\psi}\dot{y} + \hat{b}T_m \end{cases} \tag{B.18}$$

Following the generic procedure, the positive Lyapunov function candidate is chosen as $V_1 = \frac{1}{2}e_{\dot{x}}^2$, which has the time derivative described as (B.19).

$$\dot{V}_1 = e_{\dot{x}}\dot{e}_{\dot{x}} = e_{\dot{x}}\left[\ddot{x} - \ddot{x}_{\text{Ref}}\right] \tag{B.19}$$

It can be seen that taking the input, T_m, as shown in (B.20), yields the nominal system to be globally asymptotically stable at $e_{\dot{x}} = 0$, , as shown in (B.21).

$$\frac{1}{\hat{R}_{\text{eff}}\hat{m}_e} T_m = -ke_{\dot{x}} - \frac{\hat{m}}{\hat{m}_e}\dot{\psi}\dot{y} + \ddot{x}_{\text{Ref}} \tag{B.20}$$

$$\ddot{x} = -k\left(\dot{x} - \dot{x}_{\text{Ref}}\right) + \ddot{x}_{\text{Ref}} \Leftrightarrow \dot{e}_{\dot{x}} = \left[\ddot{x} - \ddot{x}_{\text{Ref}}\right] = -ke_{\dot{x}} \tag{B.21}$$

The time derivative of the Lyapunov candidate is then shown in (B.22) to be always negative when $k > 0$, yielding the nominal system to be then asymptotically stable.

$$\dot{V}_1 = e_{\dot{x}}\dot{e}_{\dot{x}} = -ke_{\dot{x}}^2 \tag{B.22}$$

From the nominal controller design it can be seen how the inequalities in (B.4) are fulfilled with (B.23).

$$\alpha_1 = \frac{1}{4}e_{\dot{x}}^2 \leq V = \frac{1}{2}e_{\dot{x}}^2 \leq \alpha_2 = e_{\dot{x}}^2 \tag{B.23}$$

Choosing the controller gain $k = 0.8$, it can be seen how the inequality (B.5) is fulfilled with (B.24).

$$\dot{V} = -ke_{\dot{x}}^2 \leq -\alpha_3 = -\frac{1}{4}e_{\dot{x}}^2 \tag{B.24}$$

From the description of the input uncertainties in (B.2), δ can be isolated as in (B.25).

$$\delta = \hat{G}^{-1}\left[f(x) - \hat{f}(x) + \left(G(x) - \hat{G}(x)\right)\psi(x) + \left(G(x) - \hat{G}(x)\right)v\right] \tag{B.25}$$

Considering the real system (6.6) and the nominal system (B.18), the expression for the uncertainty δ in the studied case can be extracted as shown in (B.26).

$$\delta = \frac{1}{\hat{b}}\left[\left(\frac{a\hat{b} - b\hat{a}}{\hat{b}}\right)\dot{y}\dot{\psi} + \left(\frac{\hat{b} - b}{\hat{b}}\right)ke_{\dot{x}} + \left(\frac{b - \hat{b}}{\hat{b}}\right)\ddot{x}_{\text{Ref}}\right] + \left(\frac{b - \hat{b}}{\hat{b}}\right)v \tag{B.26}$$

The inequality (B.6) needs to be fulfilled. In (B.27) it is shown how a non-negative continuous function $\rho(x)$ and a constant κ_0 can be defined such that the inequality holds.

$$\begin{aligned}
|\delta| &= \left|\frac{1}{\hat{b}}\left[\left(\frac{a\hat{b} - b\hat{a}}{\hat{b}}\right)\dot{y}\dot{\psi} + \left(\frac{\hat{b} - b}{\hat{b}}\right)ke_{\dot{x}} + \left(\frac{b - \hat{b}}{\hat{b}}\right)\ddot{x}_{\text{Ref}}\right] + \left(\frac{b - \hat{b}}{\hat{b}}\right)v\right| \\
&\leq \left|\frac{1}{\hat{b}}\right|\left[\left|\left(\frac{a\hat{b} - b\hat{a}}{\hat{b}}\right)\dot{y}\dot{\psi}\right| + \left|\left(\frac{\hat{b} - b}{\hat{b}}\right)ke_{\dot{x}}\right| + \left|\left(\frac{b - \hat{b}}{\hat{b}}\right)\ddot{x}_{\text{Ref}}\right|\right] + \left|\left(\frac{b - \hat{b}}{\hat{b}}\right)v\right| \\
&\leq \left|\frac{1}{\hat{b}}\right|\left[\left|\frac{a\hat{b} - b\hat{a}}{\hat{b}}\right|\left|\dot{y}\dot{\psi}\right| + \left|\frac{\hat{b} - b}{\hat{b}}\right|k\left\|e_{\dot{x}}\right\|_2 + \left|\frac{b - \hat{b}}{\hat{b}}\right|\left\|\ddot{x}_{\text{Ref}}\right\|_2\right] + \left|\frac{b - \hat{b}}{\hat{b}}\right|\left\|v\right\|_2 \\
&\leq \left|\frac{1}{\hat{b}}\right|\left[\left|\frac{a\hat{b} - b\hat{a}}{\hat{b}}\right|\left\|\dot{y}\right\|_2\left\|\dot{\psi}\right\|_2 + \left|\frac{\hat{b} - b}{\hat{b}}\right|k\left\|e_{\dot{x}}\right\|_2 + \left|\frac{b - \hat{b}}{\hat{b}}\right|\left\|\ddot{x}_{\text{Ref}}\right\|_2\right] + \left|\frac{b - \hat{b}}{\hat{b}}\right|\left\|v\right\|_2 \\
&= \rho(x) + \kappa_0\left\|v\right\|_2
\end{aligned} \tag{B.27}$$

Now, the control input v defined in (B.11) can be designed. The non-negative function $\eta(x)$ is chosen to be $\eta(x) = \frac{\rho(x)}{1-\kappa_0}$ and the variable w is defined as $w^T = \frac{\partial V}{\partial x}G = e_{\dot{x}}\hat{b}$.

When reasonable intervals of values for the varying parameters are taken into account, it is concluded that $\kappa_0 = 0.6 > \left|\frac{b-\hat{b}}{\hat{b}}\right|$. If r is taken to be $4\frac{\text{m}}{\text{s}}$, which is an upper bound for the error in the velocity, $e_{\dot{x}}$, then $\varepsilon = 0.01 < 5$ is taken as the value to be used for the control strategy v, following (B.13). This value for ε gives the boundary $b(\varepsilon) < 0.18$.

B.2 Backstepping

Being aware of the nonlinearities present in the lateral dynamics of the golf cart, a control algorithm capable of stabilizing the system in Section 6.2.2 should be designed. Inspecting the expressions describing the errors to be minimised, the method chosen for designing such an algorithm is backstepping. This is because the real input to the system, δ, is not present in the equations defining the errors deemed to be brought towards zero.
Briefly stated, the backstepping method consists in stabilizing a subsystem where the input is not directly affecting by designing a control law for a so called pseudo-input. This pseudo-input is another state of the full system with its own dynamics affected by the real input, which needs to be forced to follow the found control law. This is achieved by "backstepping" or designing a control law for each subsystem until the real input to the system appears to affect directly the

subsystem, and therefore, a control law for it can be found.

The following description of the backstepping controller is based on the one described in [25]. Consider the system to be divisible in two subsystems as (B.28) and (B.29).

$$\dot{\eta} = f_1(\eta) + g_1(\eta)\xi + h_1(\eta)\nu_1 \tag{B.28}$$
$$\dot{\xi} = f_2(\xi) + g_2(\xi)u \tag{B.29}$$

Where η and ξ define the states of the full system, u is the control input of the system and ν_1 is an external input.

The first step is to design a feedback to stabilize the subsystem $\dot{\eta}$ at the origin $\eta = 0$. Taking ξ as the virtual control input, the feedback law is designed as $\xi = \phi(\eta)$.
Taking a Lyapunov function candidate for (B.38) as $V_1 = \frac{1}{2}\eta^T\eta$, then its derivative becomes (B.30).

$$\dot{V}_1 = \eta^T\left(f_1 + g_1\xi + h_1\kappa\right) \tag{B.30}$$

If the virtual controller ξ is taken as shown in (B.31), then the error, η, becomes asymptotically stable at the origin, and the derivative of the Lyapunov function is negative when the matrix $K_1 > 0$, as shown in (B.32).

$$g_1\xi = -f_1 - h_1\kappa - K_1\eta \tag{B.31}$$

$$\begin{aligned}\dot{\eta} &= -K_1\eta \\ \dot{V}_1 &= -\eta^T K_1\eta\end{aligned} \tag{B.32}$$

As it can be seen from (B.31), it is required that the function g_1 is invertible to find the exact value of ξ.

Now, a change of variables is applied, such that $z = \xi - \phi(\eta)$, and a control function for the control input u is designed such that it brings z to 0 in steady state. If z is driven towards zero, the states ξ are driven towards the control law $\phi(\eta)$, which stabilizes the states η at zero.
Using this new variable with the same Lyapunov function candidate yields the new derivative $\dot{V}_1 = -\eta^T K_1\eta + \eta^T g_1 z$. Taking another Lyapunov function candidate as $V_2 = V_1 + \frac{1}{2}z^T z$, its derivative is obtained as shown in (B.33).

$$\dot{V}_2 = \dot{V}_1 + z^T\left(f_2 + g_2u - \frac{\partial\phi}{\partial\eta}(\eta)\dot{\eta}\right) \tag{B.33}$$

If the input of the system, u, is then taken to be as shown in (B.34), with the matrix $K_2 > 0$, it can be seen how the system with the change of variables applied, becomes stable at $z = 0$; as shown in (B.43).

$$g_2u = -K_2z - f_2 + \frac{\partial\phi}{\partial\eta}(\eta)\dot{\eta} - g_1^T\eta \tag{B.34}$$

It can be seen again, how it is required for g_2 to be invertible in order to find the needed value for the input u.

$$\dot{\xi} = -K_2z + \frac{\partial\phi}{\partial\eta}(\eta)\dot{\eta} - g_1^T\eta \Leftrightarrow \dot{z} = -K_2z - g_1^T\eta \tag{B.35}$$

Furthermore, as seen in (B.36), the derivative of V_2 is always negative, meaning that the system is deemed to be asymptotically stable at the origin, $e = 0$ and $z = 0$.

$$\dot{V}_2 = -\eta^T K_1\eta - z^T K_2 z \tag{B.36}$$

Examining now the particular case being studied (B.37), the system described by (B.28) and (B.29) is found to follow the structure shown in (B.38) and (B.39).

$$\begin{cases} \dot{e}_y &= \dot{x}e_\psi + \dot{y} + l_s e_\psi \\ \dot{e}_\psi &= \dot{\psi} - \kappa\dot{x} \\ \ddot{y} &= -\frac{C_{\alpha r}+C_{\alpha f}}{m\dot{x}}\dot{y} + \frac{l_r C_{\alpha r} - l_f C_{\alpha f}}{m\dot{x}}\dot{\psi} - \dot{x}\dot{\psi} + \frac{C_{\alpha f}}{m}\delta \\ \ddot{\psi} &= \frac{l_r C_{\alpha r} - l_f C_{\alpha f}}{I_z\dot{x}}\dot{y} - \frac{l_r^2 C_{\alpha r} + l_f^2 C_{\alpha f}}{I_z\dot{x}}\dot{\psi} + \frac{l_f C_{\alpha f}}{I_z}\delta \end{cases} \tag{B.37}$$

This is when the curvature term $-\kappa\dot{x}$ is added to the system as $h_1(\eta)\nu_1$, and because the second subsystem is linear when the longitudinal velocity is considered to be constant, which is necessary in order to split the system into longitudinal and lateral dynamics.

$$\dot{\eta} = f_1(\eta) + g_1(\eta)\xi + h_1(\eta)\kappa \tag{B.38}$$

$$\dot{\xi} = A\xi + Bu \tag{B.39}$$

The different functions from (B.38) are described in (B.40).

$$\begin{aligned} \eta &= \begin{bmatrix} e_y & e_\psi \end{bmatrix}^T \\ f_1 &= \begin{bmatrix} (\dot{x} + l_{\text{prev}})e_\psi \\ 0 \end{bmatrix} \\ g_1 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \xi &= \begin{bmatrix} \dot{y} & \dot{\psi} \end{bmatrix}^T \\ h_1 &= \begin{bmatrix} 0 \\ -\dot{x} \end{bmatrix} \end{aligned} \tag{B.40}$$

Taking the equations simplified shown in (6.10), the functions for (B.39) are shown in (B.41).

$$\begin{aligned} A &= \begin{bmatrix} -\frac{C_{\alpha r}+C_{\alpha f}}{m\dot{x}} & \frac{l_r C_{\alpha r} - l_f C_{\alpha f}}{m\dot{x}} - \dot{x} \\ \frac{l_r C_{\alpha r} - l_f C_{\alpha f}}{I_z\dot{x}} & -\frac{l_r^2 C_{\alpha r} + l_f^2 C_{\alpha f}}{I_z\dot{x}} \end{bmatrix} \\ B &= \begin{bmatrix} \frac{C_{\alpha f}}{m} \\ \frac{l_f C_{\alpha f}}{I_z} \end{bmatrix} \\ u &= \delta \end{aligned} \tag{B.41}$$

Now, following the procedure described previously, the input δ, is found as (B.42) with the matrix $K_2 > 0$, it can be seen how the system with the change of variables applied, becomes stable at $z = 0$; as shown in (B.43). This is due to the time derivative Lyapunov function being (B.44).

$$Bu = -K_2 z - A\xi + \frac{\partial\phi}{\partial\eta}(\eta)\dot{\eta} - g_1^T\eta \tag{B.42}$$

$$\dot{\xi} = -K_2 z + \frac{\partial\phi}{\partial\eta}(\eta)\dot{\eta} - g_1^T\eta \Leftrightarrow \dot{z} = -K_2 z - g_1^T\eta \tag{B.43}$$

$$\dot{V}_2 = \dot{V}_1 + z^T\left(A\xi + Bu - \frac{\partial\phi}{\partial\eta}(\eta)\dot{\eta}\right) = -\eta^T K_1\eta - z^T K_2 z \tag{B.44}$$

Unfortunately, as stated before, B should be invertible for this approach to be implementable. This is due to it trying to bring two outputs to a desired value with a single input to control them. It can be seen from Figure B.1 how a solution to the problem can only be found if the desired values for the outputs to be controlled lay in the space defined by the vector B.

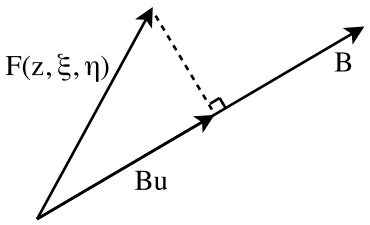


Figure B.1: Vector representation of the control problem. The outputs of the system can only be brought to their desired value when this is in the space defined by B.

Therefore, the objective of bringing the errors to zero needs to be addressed differently. It is not possible to bring both errors describing the lateral dynamics to zero in this manner.

It is decided then to approach the control problem in a different manner. It is established that only one of the errors describing the lateral error will be brought to zero. The backstepping technique is used considering the system equations (B.45), (B.46) and (B.47). The error to be stabilised in zero is decided to be the lateral distance to the path. This is to force the cart to be close to the actual desired position and not in a parallel path. The controller is then designed backstepping more than once. First, a control law for the heading error is found in order to bring the lateral distance to the path towards zero. Afterwards, a control law for the yaw rate, $p\dot{s}i$, is found to bring the heading error to its desired control law. Finally, a control law for the true input to the system, the steering angle δ, is found to bring the yaw rate to its desired control law.

$$\dot{e}_y = \dot{x}e_\psi + \dot{y} + l_s e_\psi \tag{B.45}$$

$$\dot{e}_\psi = \dot{\psi} - \kappa\dot{x} \tag{B.46}$$

$$\ddot{\psi} = \frac{l_r C_{\alpha r} - l_f C_{\alpha f}}{I_z \dot{x}}\dot{y} - \frac{l_r^2 C_{\alpha r} + l_f^2 C_{\alpha f}}{I_z \dot{x}}\dot{\psi} + \frac{l_f C_{\alpha f}}{I_z}\delta \tag{B.47}$$

Taking first (B.45), the heading error e_ψ is considered as an input to stabilize e_y. Considering the positive Lyapunov candidate $V_1 = \frac{1}{2}e_y^2$, it is shown how taking the control law (B.48) guarantees the time derivative of the Lyapunov function, (B.49), to be negative and thus, the system to be asymptotically stable at the origin.

$$e_\psi \to \phi_1 = \frac{1}{\dot{x} + l_s}\left(-k_1 e_y - \dot{y}\right) \tag{B.48}$$

$$\dot{V}_1 = e_y \dot{e}_y = e_y\left((\dot{x} + l_s)e_\psi + \dot{y}\right) = -k_1 e_y^2 < 0 \tag{B.49}$$

Now, a new variable, $z_1 = \phi_1 - e_\psi$, is introduced such that it expresses the difference between the desired value of the heading error, ϕ_1, and its true value, e_ψ. The first subsystem used, (B.45), is then left to be (B.50), which leaves the time derivative of V_1 as shown in (B.51).

$$\dot{e}_y = -k_1 e_y - (\dot{x} + l_s) z_1 \tag{B.50}$$

$$\dot{V}_1 = -k_1 e_y^2 + (\dot{x} + l_s) e_y z_1 \tag{B.51}$$

The second system to be stabilised is now the one described by z_1, which is deemed to be brought to zero. Considering a new positive Lyapunov function candidate, $V_2 = V_1 + \frac{1}{2} z_1^2$, it is shown that the subsystem (B.52) is asymptotically stable at the origin if the control law for $\dot{\psi}$ is taken as shown in (B.53).

$$\dot{z}_1 = \frac{\partial \phi_1}{\partial t} - \dot{e}_\psi = \frac{\partial \phi_1}{\partial t} + \kappa \dot{x} - \dot{\psi} \tag{B.52}$$

$$\dot{\psi} \rightarrow \phi_2 = \frac{\partial \phi_1}{\partial t} + \kappa \dot{x} + k_2 z_1 + (\dot{x} + l_s) e_y \tag{B.53}$$

Where the expression $\frac{\partial \phi_1}{\partial t}$ denotes the time derivative of ϕ_1, and yields then the time derivative of the Lyapunov candidate, $\dot{V}_2$, to be negative as shown in (B.54).

$$\dot{V}_2 = \dot{V}_1 + z_1 \dot{z}_1 = -k_1 e_y^2 - (\dot{x} + l_s) e_y z_1 + z_1 \left(\frac{\partial \phi_1}{\partial t} - \dot{e}_\psi \right) = -k_1 e_y^2 - k_2 z_1^2 < 0 \tag{B.54}$$

Finally, a new variable, $\lambda = \phi_2 - \dot{\psi}$, is introduced such that it expresses the difference between the desired value for the heading rate, ϕ_2, and its true value, $\dot{\psi}$. The system defined in (B.52) changes then to be as shown in (B.55), leaving the time derivative of V_2 to be as shown in (B.56).

$$\dot{z}_1 = -k_2 z_1 + z_2 + (\dot{x} + l_s) e_y \tag{B.55}$$

$$\dot{V}_2 = -k_1 e_y^2 - k_2 z_1^2 + z_1 z_2 \tag{B.56}$$

The newly introduced variable z_2 is deemed to be brought to zero such that the heading rate takes de necessary values to stabilize the lateral error. Considering then the dynamics of it shown in (B.57) and the Lyapunov candidate $V_3 = V_2 + \frac{1}{2} z_2^2$, the system is found to be asymptotically stable at the origin when applying the control law shown in (B.59) to the true input to the system, the steering angle δ.

$$\begin{aligned} \dot{z}_2 &= \frac{\partial \phi_2}{\partial t} - \ddot{\psi} \\ &= \frac{\partial \phi_2}{\partial t} - \left(\frac{l_r C_{\alpha r} - l_f C_{\alpha f}}{I_z \dot{x}} \dot{y} - \frac{l_r^2 C_{\alpha r} + l_f^2 C_{\alpha f}}{I_z \dot{x}} \dot{\psi} + \frac{l_f C_{\alpha f}}{I_z} \delta \right) \\ &= \frac{\partial \phi_2}{\partial t} - a_1 \dot{y} - a_2 \dot{\psi} - b\delta \end{aligned} \tag{B.57}$$

(B.58)

$$\delta = \frac{1}{b} \left(\frac{\partial \phi_2}{\partial t} - a_1 \dot{y} - a_2 \dot{\psi} - k_3 z_2 + z_1 \right) \tag{B.59}$$

The derivative of the Lyapunov function V_3 is then showed to be negative in (B.60).

$$\dot{V}_3 = \dot{V}_2 + z_2 \dot{z}_2 = -k_1 e_y^2 - k_2 z_1^2 + z_1 z_2 + z_2 \left(\frac{\partial \phi_2}{\partial t} - a_1 \dot{y} - a_2 \dot{\psi} - b\delta \right) = -k_1 e_y^2 - k_2 z_1^2 - k_3 z_2^2 < 0 \tag{B.60}$$

The full system is then found to be stable in $e_y = 0$, $z = 0$ and $\lambda = 0$. In order to define the values to be taken for the controller gains k_1, k_2 and k_3, the closed loop system shown in (B.61) has to be found to be Hurwitz.

$$\begin{bmatrix} \dot{e}_y \\ \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} -k1 & -(\dot{x} + l_s) & 0 \\ (\dot{x} + l_s) & -k_2 & 1 \\ 0 & -1 & -k_3 \end{bmatrix} \begin{bmatrix} e_y \\ z_1 \\ z_2 \end{bmatrix} \tag{B.61}$$

Since the closed loop system depends on the longitudinal velocity of the cart, $\dot{x}$, the operating range of it needs to be taken into account in the design.

B.2.1 Actuator dynamics inclusion

As explained in Section 6.4.2, in order to include the inputs actuating in the real cart, it is decided to expand the backstepping design such that the actuator dynamics are included. This would yield a steering wheel reference, β_{Ref} as an output of the controller instead of the steering value δ.

The new equations to be included are explained in Section 6.4.2, which specify the relation between the steer value, δ, and the steering wheel position, β, and also the actuator dynamics relating β to the reference given to it, β_{Ref}. The first relation is a gain, meaning that it does not have to be included as dynamics, and it is shown in (B.62). The dynamics of the actuator are shown in (B.63).

$$\delta = S\,\beta \tag{B.62}$$

$$\begin{bmatrix} \ddot{\beta} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -3.814 & -5.609 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\beta} \\ \beta \end{bmatrix} + \begin{bmatrix} 5.609 \\ 0 \end{bmatrix} \beta_{\text{Ref}} \tag{B.63}$$

To proceed with the controller design, it is retaken from the definition of z_2, V_3 and its derivative ,$\dot{V}_3$.
The expression found for δ is now considered a desired value for it, as performed before with the heading error, e_ψ, and the heading rate, $\dot{\psi}$.
Therefore, the desired value of δ, is considered as a new function ϕ_3, although it is decided to include the gain to convert it to a desired steering wheel value. This is shown in (B.64).

$$\beta \to \phi_3 = \frac{1}{S\,b}\left(\frac{\partial \phi_2}{\partial t} - a_1\dot{y} - a_2\dot{\psi} + k_3 z_2 + z_1\right) \tag{B.64}$$

Introducing then a new variable defining the error between the desired value for β and its true value, $z_3 = \phi_3 - \beta$, leaves the subsystem (B.57) and the time derivative of the Lyapunov function (B.60) as shown in (B.65) and (B.66) respectively.

$$\begin{aligned} \dot{z}_2 &= \frac{\partial \phi_2}{\partial t} - a_1\dot{y} - a_2\dot{\psi} - b\delta \\ &= \frac{\partial \phi_2}{\partial t} - a_1\dot{y} - a_2\dot{\psi} - b\left[S\left(\phi_3 - z_3\right)\right] \\ &= -k_3 z_2 - z_1 + S z_3 \end{aligned} \tag{B.65}$$

$$\dot{V}_3 = -k_1 e_y^2 - k_2 z_1^2 + z_1 z_2 + z_2 \dot{z}_2 = -k_1 e_y^2 - k_2 z_1^2 - k_3 z_2^2 + S\,b\,z_3 z_2 \tag{B.66}$$

Now, the system to be stabilized in zero is described by z_3. Shown in (B.67), it is stabilised in zero when the virtual input $\dot{\beta}$, is taken as shown in (B.68). This is when the new Lyapunov function is taken as $V_4 = V_3 + \frac{1}{2}z_3^2$.

$$\dot{z}_3 = \frac{\partial \phi_3}{\partial t} - \dot{\beta} \tag{B.67}$$

$$\dot{\beta} \rightarrow \phi_4 = \frac{\partial \phi_3}{\partial t} + k_4 z_3 + S\, b\, z_2 \tag{B.68}$$

As shown in (B.69), the derivative of the Lyapunov function is negative.

$$\begin{aligned} \dot{V}_4 = \dot{V}_3 + z_3 \dot{z}_3 &= -k_1 e_y^2 - k_2 z_1^2 - k_3 z_2^2 + S\, b\, z_3 z_2 + z_3 \left(\frac{\partial \phi_3}{\partial t} - \dot{\beta} \right) \\ &= -k_1 e_y^2 - k_2 z_1^2 - k_3 z_2^2 - k_4 z_3^2 \end{aligned} \tag{B.69}$$

Again, a new variable $z_4 = \phi_4 - \dot{\beta}$ is introduced describing the difference between the desired value for $\dot{\beta}$ and its true value. The subsystem (B.67) is left as (B.70) and the derivative of the Lyapunov function (B.69) is left as (B.71).

$$\dot{z}_3 = \frac{\partial \phi_3}{\partial t} - \dot{\beta} = \frac{\partial \phi_3}{\partial t} - \phi_4 + z_4 \tag{B.70}$$

$$\begin{aligned} \dot{V}_4 = \dot{V}_3 + z_3 \dot{z}_3 &= -k_1 e_y^2 - k_2 z_1^2 - k_3 z_2^2 + S\, b\, z_3 z_2 + z_3 \dot{z}_3 \\ &= -k_1 e_y^2 - k_2 z_1^2 - k_3 z_2^2 - k_4 z_3^2 + z_3 z_4 \end{aligned} \tag{B.71}$$

The system to be stabilised again at zero is defined by z_4. Taking a new Lyapunov candidate $V_5 = V_4 + \frac{1}{2} z_4^2$, the system (B.72), extracted from (B.63), is deemed to be stable when the control input is taken as (B.73).

$$\dot{z}_4 = \frac{\partial \phi_4}{\partial t} - \ddot{\beta} = \frac{\partial \phi_4}{\partial t} - \left(-3.814 \dot{\beta} - 5.609 \beta + 5.609 \beta_{\text{Ref}} \right) \tag{B.72}$$

$$\beta_{Ref} = \frac{1}{5.609} \left(\frac{\partial \phi_4}{\partial t} + 3.814 \dot{\beta} + 5.609 \beta + k_5 z_4 + z_3 \right) \tag{B.73}$$

The derivative of the Lyapunov function is shown to be negative in (B.74), yielding a stable system.

$$\begin{aligned} \dot{V}_5 = \dot{V}_4 + z_4 \dot{z}_4 &= \dot{V}_4 + z_4 \left[\frac{\partial \phi_4}{\partial t} - \left(-3.814 \dot{\beta} - 5.609 \beta + 5.609 \beta_{\text{Ref}} \right) \right] \\ &= -k_1 e_y^2 - k_2 z_1^2 - k_3 z_2^2 - k_4 z_3^2 - k_5 z_4^2 \end{aligned} \tag{B.74}$$

Finally, the controller gains defined along the derivation need to be designed. This is done by inspecting the closed loop system in (B.75).

$$\begin{bmatrix} \dot{e}_y \\ \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \dot{z}_4 \end{bmatrix} = \begin{bmatrix} -k_1 & -(\dot{x} + l_s) & 0 & 0 & 0 \\ (\dot{x} + l_s) & -k_2 & 1 & 0 & 0 \\ 0 & -1 & -k_3 & b\, S & 0 \\ 0 & 0 & -b\, S & -k_4 & 1 \\ 0 & 0 & 0 & -1 & -k_5 \end{bmatrix} \begin{bmatrix} e_y \\ z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} \tag{B.75}$$

The gains need to be designed such that the closed-loop is Hurwitz, but moreover, it is necessary that the subsystem defining the steering wheel is faster. This is due to the cascaded-like structure of the backstepping.

C GPS

In this appendix details of the GPS is provided. This includes understanding the type of messages the GPS sends as well as how this data is transformed into ROS.

C.1 GPS NMEA sentences

As described in Figure 2.3 the GPS uses NMEA 0183 standard and the two message types used are the GPRMC and GPGGA messages. The GPGGA is a message that provides 3D location (latitude, longitude and altitude) and accuracy data. Table C.1 gives an example of a GPGGA message and explains each part of the message.

	$GPGGA,130001.8,5700.86299,N,00959.19505,E,1,09,0.9,18.8,M,40.7,M,,*66
130001.8	Gives the time in the format 13:00:01.8 UTC
5700.86299,N	Latitude 57° 00.86299' N
00959.19505,E	Longitude 9° 59.19505' E
1	Fix Quality (1 = valid, 0 = invalid)
09	Number of satellites connected to
0.9	Horizontal dilution of position (accuracy of data)
18.8,M	Altitude 18.8 m above mean sea level
40.7,M	Height of mean sea level above WGS84 ellipsoid
*66	Checksum to ensure integrity of message

Table C.1: Explanation of GPGGA NMEA string

The GPRMC message is described as the recommended minimum data for GPS. In Table C.2 an example of a GPRMC message is given as well as an explanation about its contents.

	$GPRMC,130001.9,A,5700.86302,N,00959.19508,E,000.24,172.1,200318,002.9,E,A*32
130001.9	Gives the time in the format 13:00:01.9 UTC
A	Status (A = active, V = void)
5700.86302,N	Latitude 57° 00.86302' N
00959.19508,E	Longitude 9° 59.19508' E
000.24	Speed over ground (knots)
172.1	Track angle in degrees (0 is True North)
200318	The date 20^{th} March 2018
002.9,E	Magnetic variation 2.9° East
40.7,M	Height of mean sea level above WGS84 ellipsoid
A*32	Checksum to ensure integrity of message

Table C.2: Explanation of GPRMC NMEA string

C.2 Transformation from GPS to XY

In this section the transformation used between latitude and longitude of the GPS messages to an XY plane such as that described in Section 4.1 is given. To transform the GPS coordinates given in by the GPS to an XY plane, the origin of the XY plane needs to be set. For this, a reference latitude, φ_r and reference longitude λ_r are used as the origin. When GPS coordinates are recieved the distance and bearing to the origin are calculated [70]. The haversine formula is used to calculate the distance between the GPS coordinates. Given the coordinates φ, λ, the distance to the reference is calculated. Let,

$$a = \sin^2 \frac{\Delta\varphi}{2} + \cos\varphi_r \cos\varphi \sin^2 \frac{\Delta\lambda}{2}, \tag{C.1}$$

where $\Delta\varphi = \varphi - \varphi_r$ and $\Delta\lambda = \lambda - \lambda_r$. Thus, the distance is calculated by,

$$d = 2R \arctan2\left(\sqrt{a}, \sqrt{1-a}\right). \tag{C.2}$$

The bearing, where North is 0°, between these two points is calculated using,

$$\theta = \arctan2\left(\sin\Delta\lambda \cos\varphi, \cos\varphi_r \sin\varphi - \sin\varphi_r \cos\varphi \cos\Delta\lambda\right). \tag{C.3}$$

Now that the distance, in meters, and bearing, in radians, between the coordinates are calculated, the X and Y distances can be calculated using,

$$X = d\sin\theta \tag{C.4}$$
$$Y = d\cos\theta. \tag{C.5}$$

Now that the GPS measurements are transformed to the local inertial frame it is available to be used by the EKF in Chapter 7.

C.3 Results of GPS test

In this section, the means and variances of the noise of the GPS signals are determined. To do this the vehicle's GPS data was recorded for approximately 45 minutes while stationary. The reasoning for determining the means and variances were in order to design the EKF.

The signals used from the GPS are the latitude and longitude, converted to XY using the transformation described in Section C.2, the ψ which is taken from the magnetic variation parameter of the GPRMC message and the $\dot{x}$ coming from the speed over ground parameter of the GPRMC message. It is important to note that the first three signals describe the vehicles pose in the inertial frame while the last signal describes the vehicles motion in the body frame.

C.3.1 Inertial frame measurements

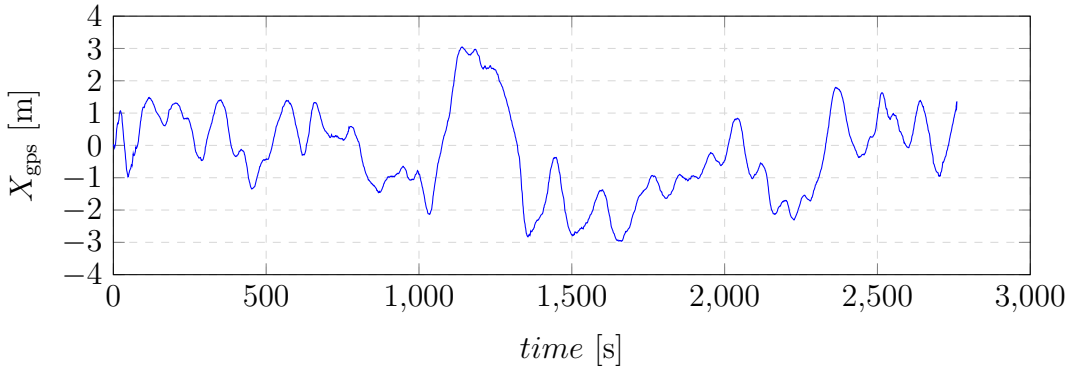


Figure C.1: Plot of X_{gps} from stationary GPS test.

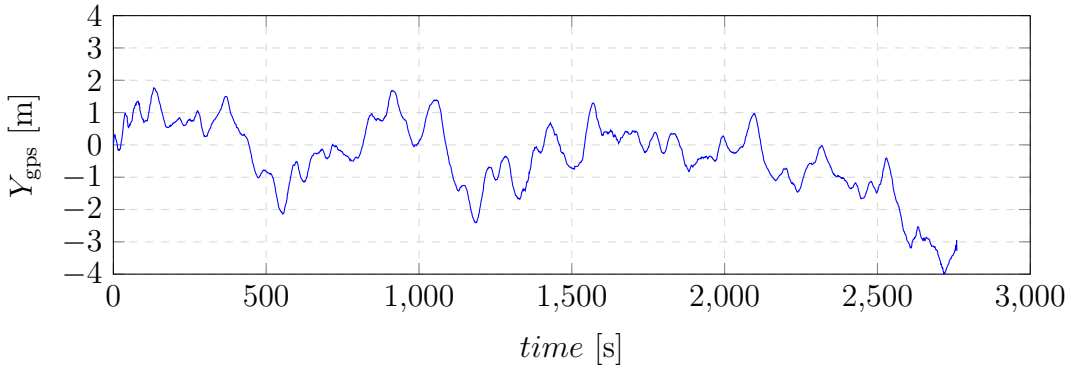


Figure C.2: Plot of Y_{gps} from stationary GPS test.

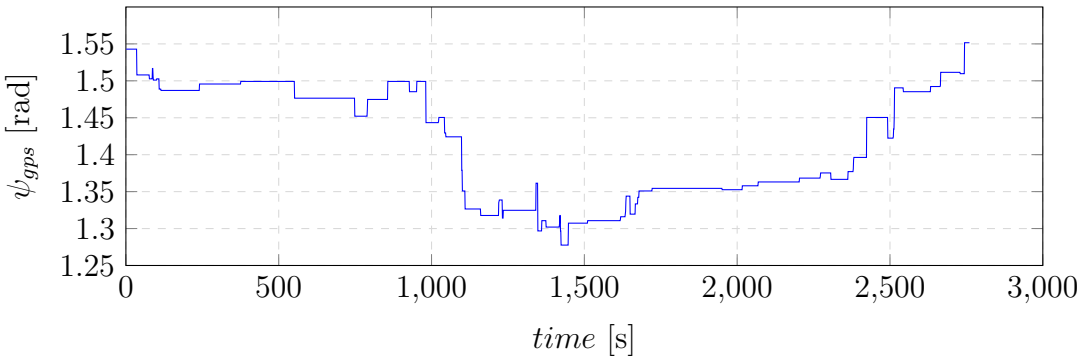


Figure C.3: Plot of ψ_{gps} from stationary GPS test.

$$\mu_{X,\text{gps}} = -0.2476 \text{ m}, \tag{C.6}$$
$$\mu_{Y,\text{gps}} = -0.3690 \text{ m}, \tag{C.7}$$
$$\mu_{\psi,\text{gps}} = 1.4162 \text{ rad}, \tag{C.8}$$
$$\sigma_{X,\text{gps}} = 1.8146 \text{ m}^2, \tag{C.9}$$
$$\sigma_{Y,\text{gps}} = 1.2713 \text{ m}^2, \tag{C.10}$$
$$\sigma_{\psi,\text{gps}} = 0.0057 \text{ rad}^2, \tag{C.11}$$

C.3.2 Body frame measurement

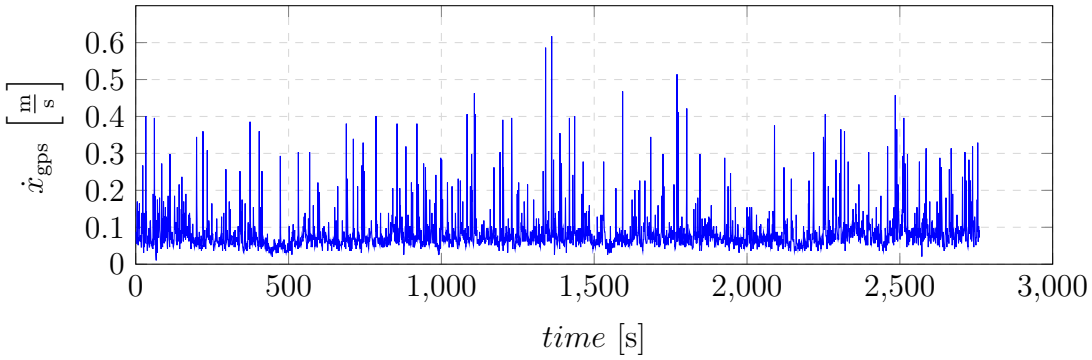


Figure C.4: Plot of $\dot{x}_{\text{gps}}$ from stationary GPS test.

$$\mu_{\dot{x},\text{gps}} = 0.08513 \ \frac{\text{m}}{\text{s}} \qquad \text{(C.12)}$$

$$\sigma_{\dot{x},\text{gps}} = 0.0031 \ \frac{\text{m}^2}{\text{s}^2} \qquad \text{(C.13)}$$

D IMU

The IMU described in Figure 2.3 provides accelerometer and gyrometer data, both being 3-axis sensors. The accelerometer provides measurements of the linear accelerations acting on the sensors in x, y and z directions. On the other hand, the gyrometer provides measurements of the angular velocities experienced by the sensor about the x, y and z direction.

As these measurements are used in the design for the EKF in Chapter 7, the measurement noise needs to be analysed. To do so, the vehicle was left stationary, for approximately 11 hours while the IMU data was recorded.

The data was treated to have a normal distribution such that the mean and variance of the data could be calculated. The results are shown in Section D.1.

D.1 Results of IMU test

In this section, the data recorded from the IMU stationary test are shown and the means and variances of the sensor are stated. As the vehicles is only considered to be in 2D, only the linear accelerations in x and y, and the angular velocity about the z axis are considered.

D.1.1 Linear accelerations

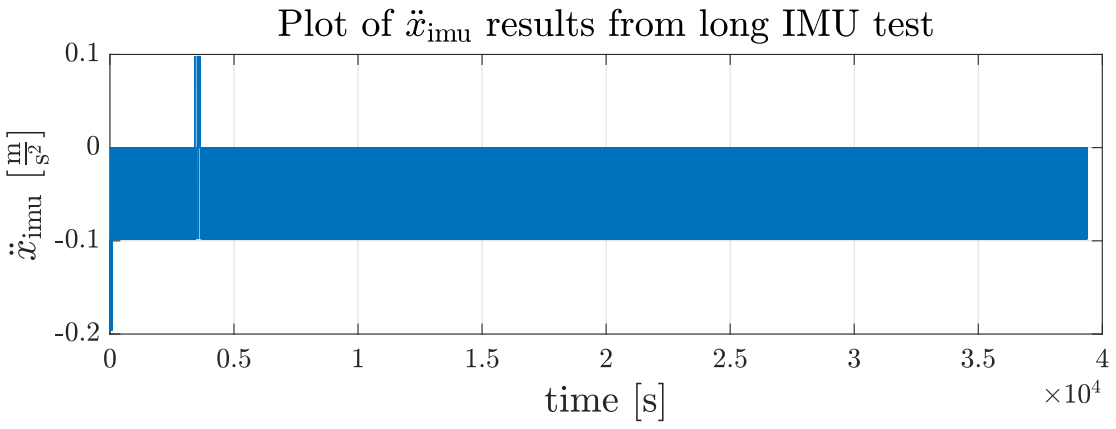


Figure D.1: Plot of $\ddot{x}_{\text{imu}}$ from long IMU test.

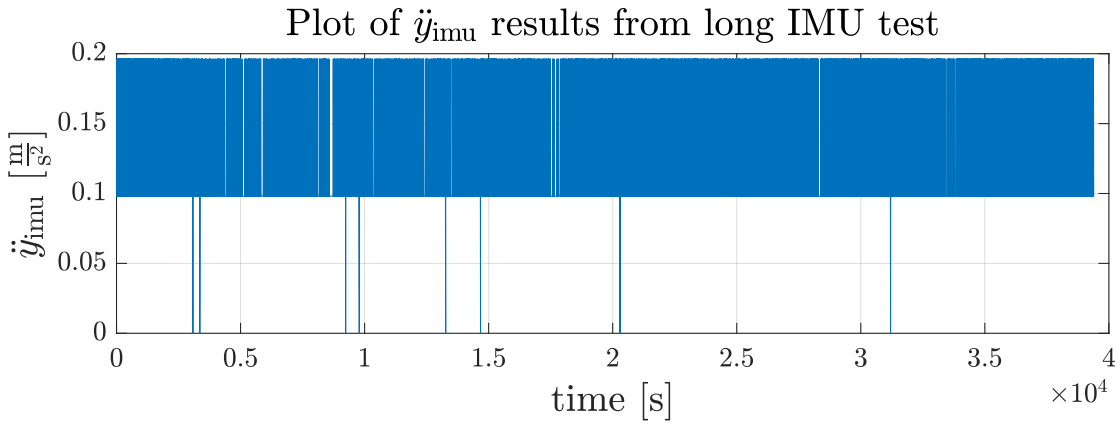


Figure D.2: Plot of $\ddot{y}_{\text{imu}}$ from long IMU test.

$$\mu_{\ddot{x},\text{imu}} = -0.0448 \ \frac{\text{m}}{\text{s}^2}, \tag{D.1}$$

$$\mu_{\ddot{y},\text{imu}} = 0.0986 \ \frac{\text{m}}{\text{s}^2}, \tag{D.2}$$

$$\sigma_{\ddot{x},\text{imu}} = 0.0024 \ \frac{\text{m}^2}{\text{s}^4}, \tag{D.3}$$

$$\sigma_{\ddot{y},\text{imu}} = 4.8747 \times 10^{-5} \ \frac{\text{m}^2}{\text{s}^4} \tag{D.4}$$

D.1.2 Angular velocity

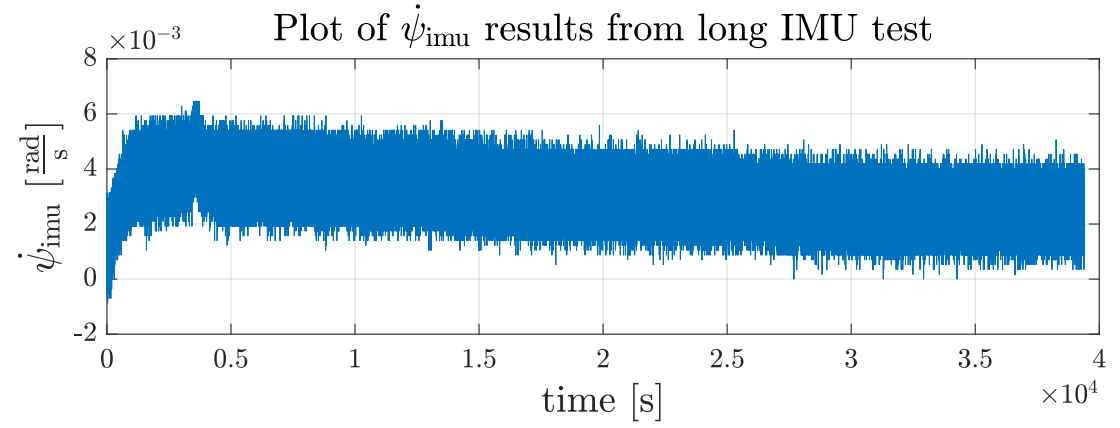


Figure D.3: Plot of $\dot{\psi}_{\text{imu}}$ from long IMU test.

$$\mu_{\dot{\psi},\text{imu}} = 0.0031 \ \frac{\text{rad}}{\text{s}}, \tag{D.5}$$

$$\sigma_{\dot{\psi},\text{imu}} = 6.096 \times 10^{-7} \ \frac{\text{rad}}{\text{s}^2} \tag{D.6}$$

E Cartographer configuration

The configuration file used to perform the results shown in Figure 8.15 can be found in the following:

```
include "map_builder.lua"
include "trajectory_builder.lua"

options = {
  map_builder = MAP_BUILDER,
  trajectory_builder = TRAJECTORY_BUILDER,
  map_frame = "map",
  tracking_frame = "imu_link",
  published_frame = "odom",
  odom_frame = "odom",
  provide_odom_frame = false,
  publish_frame_projected_to_2d = false,
  use_odometry = true,
  use_nav_sat = false,
  use_landmarks = false,
  num_laser_scans = 2,
  num_multi_echo_laser_scans = 0,
  num_subdivisions_per_laser_scan = 1,
  num_point_clouds = 0,
  lookup_transform_timeout_sec = 0.2,
  submap_publish_period_sec = 0.3,
  pose_publish_period_sec = 0.1,
  trajectory_publish_period_sec = 30e-3,
  rangefinder_sampling_ratio = 1.,
  odometry_sampling_ratio = 1.,
  fixed_frame_pose_sampling_ratio = 1.,
  imu_sampling_ratio = 1.,
  landmarks_sampling_ratio = 1.,
}

MAP_BUILDER.use_trajectory_builder_2d = true

-- Local SLAM

TRAJECTORY_BUILDER_2D.use_imu_data = true
TRAJECTORY_BUILDER_2D.num_accumulated_range_data = 5
TRAJECTORY_BUILDER_2D.min_range = 0.5
TRAJECTORY_BUILDER_2D.max_range = 40

TRAJECTORY_BUILDER_2D.ceres_scan_matcher.translation_weight = 1
TRAJECTORY_BUILDER_2D.ceres_scan_matcher.rotation_weight = 20

```

```
43 TRAJECTORY_BUILDER_2D.submaps.num_range_data = 20
44 TRAJECTORY_BUILDER_2D.submaps.resolution = 0.08
45
46
47 --GLobal SLAM
48
49 -- POSE_GRAPH.optimize_every_n_nodes = 10
50
51 POSE_GRAPH.constraint_builder.ceres_scan_matcher.translation_weight = 1
52 POSE_GRAPH.constraint_builder.ceres_scan_matcher.rotation_weight = 50
53
54 POSE_GRAPH.optimization_problem.local_slam_pose_translation_weight = 1e2
55 POSE_GRAPH.optimization_problem.local_slam_pose_rotation_weight = 50
56 POSE_GRAPH.optimization_problem.odometry_translation_weight = 2e3
57 POSE_GRAPH.optimization_problem.odometry_rotation_weight = 3e3
58
59
60 return options
```

Code snippet E.1: Commands to set-up a ROS workspace

F Odometry kinematic model

To generate odometry measurements a kinematic model is derived in the following. The framework considered is described in Section 4.1 and the model is derived from the geometry presented in Figure F.1.

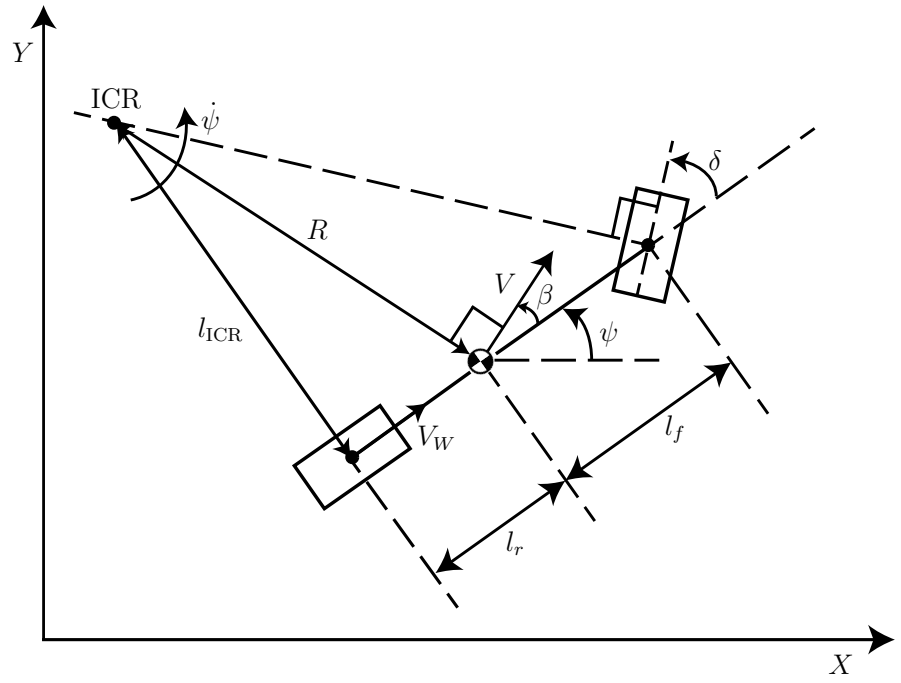


Figure F.1

The model is fully derived using trigonometry. First, the ICR is defined as shown ni Figure F.1. This allows to find the equation (F.1) which describes the the yaw rate $\dot{psi}$.

$$\dot{\psi} = \frac{V_W}{l_{\text{ICR}}} \tag{F.1}$$

In equation (F.1), l_{ICR} is determined by trigonometry to be $l_{\text{ICR}} = \left(l_f + l_r\right) / \tan(\delta)$. By substitution, (F.1) becomes:

$$\dot{\psi} = V_W \frac{\tan(\delta)}{l_f + l_r} \tag{F.2}$$

which is considered the first equation of the kinematic model. The two remaining equations correspond to $\dot{X}$ and $\dot{Y}$. Notice that these are the components of V, i.e. the velocity at G. Hence, an expression is deemed as first step. Once again, the equation (F.3) for V is found by trigonometry.

$$V = \dot{\psi} R = V_W \frac{\tan(\delta)}{l_f + l_r} \tag{F.3}$$

From (F.3) it is possible to obtain then the equations (F.4) and (F.7) for $\dot{X}$ and $\dot{Y}$.

$$\dot{X} = V\cos\left(\psi + \beta\right) \tag{F.4}$$

$$\dot{Y} = V\sin\left(\psi + \beta\right) \tag{F.5}$$

At this point the kinematic model is defined by equations (F.2), (F.4) and (F.7):

$$\begin{array}{l} \dot{X} = V\cos\left(\psi + \beta\right) \\ \dot{Y} = V\sin\left(\psi + \beta\right) \\ \dot{\psi} = \frac{V_W}{l_f + l_r}\left(\tan\left(\delta\right)\right) \end{array} \tag{F.6}$$

where β is defined as $\beta = \tan^{-1}\left(\frac{l_r}{l_{\text{ICR}}}\right)$ and developing such expression with substitution of l_{ICR} by $l_{\text{ICR}} = \left(l_f + l_r\right)/\tan\left(\delta\right)$ the equation for β is obtained.

$$\beta = \tan^{-1}\left(\frac{l_r\tan\left(\delta\right)}{l_f + l_r}\right) \tag{F.7}$$