DEGREE PROJECT, IN AUTOMATIC CONTROL , SECOND LEVEL
STOCKHOLM, *SWEDEN 2015*

Autonomous driving system for reversing an articulated vehicle

AMRO ELHASSAN

KTH ROYAL INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL ENGINEERING

Autonomous driving system for reversing an articulated vehicle

Amro Elhassan

Supervisor and Examiner:
Jonas Mårtensson

Co-Supervisor:
Rui Oliveira

Master of Science Thesis
School of Electrical Engineering
Department of Automatic Control
The Royal Institute of Technology

2015

"The weak point of the modern car is the squidgy organic bit behind the wheel."

Jeremy Clarkson

Abstract

Articulated vehicles are widely used in the economically vital cargo industry as they provide a greater maneuverability than their rigid counterparts. Hence, autonomous driving systems for articulated vehicles have become the subject of intense research in the robotic community. This thesis analyzes the reverse motion of an articulated vehicle, namely a tractor-trailer with one on-axle hitched semitrailer, and develops a full autonomous driving system that enables reverse parking in the presence of static obstacles. The motion controller used in the autonomous driving system is based on a two-level feedback control system, with a path stabilization controller in the first level and a hitch angle controller in the second level. The path planner used is a modified RRT planner where the Dubins path has been incorporated in order to enable the planning towards a goal pose rather than merely a goal region. The modifications made have resulted in several improvements, such as more accurate planning and higher computational efficiency. Using a 1:32 scale remote controlled tractor-trailer, and a Qualisys motion capture system for pose estimation, the autonomous driving system was successfully implemented and validated.

Contents

Chapter 1

Introduction

The vision of having autonomous vehicles fill our roads has been here for many decades, but only in recent years have substantial advances been made towards its realization. A lot of this advancement is owed to the remarkable progress in the field of computer technology, which provides the necessary tools for many of the difficult tasks in autonomous driving.

The motivations behind the development of autonomous vehicles are not few in number, comprising of everything from efficiency and quality of life improvements to environmental and monetary benefits. Additionally, with 93 % of all road accidents being caused by the human factor, a dramatic improvement in safety is expected to be seen with the application of these vehicles [1]. Consequently, this field has been a hot topic amongst researchers and developers throughout the years.

A special share of the attention has been directed towards articulated vehicles. Much of this is due to the role they play in the cargo industry, which is regarded as being a vital component of the economy [2]. The reasons for using articulated vehicles over rigid vehicles are mainly two: 1) articulated vehicles allow for much sharper turns than their rigid counterparts of the same size and 2) they make it possible for the wheels to keep contact with the ground when its surface is uneven [3]. However, articulated vehicles do have the crucial drawback of the jackknife effect, i.e. the folding of the vehicle into a position that it cannot come out of. This position is easier to fall into during the reverse motion as compared to the forward motion, due to the former being unstable as opposed to the latter [4].

Hence, as a contribution to the field of autonomous vehicles, this thesis will develop and evaluate a complete autonomous driving system for reversing an articulated vehicle.

1.1 Problem statement

The preferred choice of motion for vehicles is typically not the reverse motion if the task at hand can also be completed with the forward motion (even if it might require some additional driving distance). In other words, the reverse motion is typically used only to complete a task that cannot be performed with the forward motion. A good example where this is true when it comes to articulated vehicles is the task of reversing a tractor-trailer to a loading dock, which is also why the task of reversing an articulated vehicle to a particular pose is commonly referred to as the docking task. This thesis will therefore focus on the docking task which evidently is a far more complex task than that of reversing an articulated vehicle into a specified region.

An autonomous driving system can be divided into two parts, namely, a path planning and a motion control part. In the planning part, the path that the vehicle will take to reach the goal pose is determined. For this to be possible, the vehicle has to have some knowledge of the environment (its location relative the goal, if there are any obstacles, etc.), which can be given beforehand or obtained with the use of sensors. We will assume here that the knowledge is given, i.e. the environment in our workspace is known. Also, in order to be able to work with the task in the two-dimensional space, we will assume that the workspace is a plain surface. Furthermore, the environment in the workspace should resemble the environment in which the docking task is usually performed in, such as a docking station or construction site. The environment we will deal with will hence be unstructured with the presence of static obstacles that the articulated vehicle should be able to avoid.

As for the control part, it is much dependent on the type of articulated vehicle in question. This is because the dynamics of the articulated vehicle that we are trying to control depends on several factors, which can vary significantly from one articulated vehicle to another. An important factor is, for example, whether the hitching of the trailer is on-axle, meaning the hitching point is on-top of the tractor's rear wheel axle, or off-axle, the hitching point being placed a distance away from the rear wheel-axle. Another factor is the number of trailers in the system. Obviously, the more trailers there are, the more complex the dynamics become. The type of articulated vehicle that will be modeled here might be the most commonly seen type: a tractor-trailer where the tractor has only one on-axle hitched semitrailer.

It should also be mentioned that the docking task should be completed with the reverse motion only - that is, completing the task by way of switching between the reverse and forward motion is considered to fall outside the scope of this work.

In summary, the aim of this thesis will be to implement and validate an autonomous driving system for a tractor-trailer in order for it to complete the docking task in a known and unstructured environment that includes static obstacles.

1.2 Objectives

The specific objectives of this thesis are to:

- Analyze and model the reverse motion of a tractor-trailer.
- Investigate the motion control task and implement a suitable controller for the autonomous driving system.
- Investigate the path planning task and implement a suitable planner.
- Implement an autonomous driving system by combining the motion controller and path planner.
- Validate the implementation of the autonomous driving system with experiments.

1.3 Outline of the report

This thesis is organized in the following chapters:

Chapter 2: Background, gives a literature review on path planners and motion controllers and presents related work.

Chapter 3: Mathematical model of tractor-trailer, presents a mathematical model of the tractor-trailer and discusses the avoidance of the jackknife effect.

Chapter 4: Testbed, introduces the testbed used for the implementation and validation of the autonomous driving system.

Chapter 5: Motion control, gives a discourse of the applied motion controller.

Chapter 6: Path planning, gives a discourse of the applied path planner.

Chapter 7: Implementation, presents the results from the validation experiments on the implementation.

Chapter 8: Discussion and conclusions, discusses and draws conclusions from the results and suggests future work.

Chapter 2

Background

The evolution of autonomous driving systems is something that has been on-going since the early 20th century, and the aspiration has always been the same: having a vehicle independently navigate itself to a selected destination. This means that a proficient autonomous driving system requires a robust motion controller and a reliable path planner.

2.1 Motion controllers

The control of the reverse motion of autonomous articulated vehicles has become somewhat of a benchmark nonlinear control problem, which has given rise to many approaches. Due to the existence of nonholonomic constraints on the motion, along with holonomic constraints on the joints of the articulated vehicles, this control problem has been recognised as a truly challenging and complex one [5].

Model predictive control (MPC) provides the ability to anticipate future events, work close to the constraints and is excellent at taking the actuator limitations into consideration [6]. The authors of [7] apply a mixed logical dynamics model to describe the open-loop kinematics of the tractor-trailer vehicle and propose a cost function for it in quadratic form that considers both the steering angle and the orientation error of the trailer simultaneously. Subsequently, by using the multi-parametric mixed-integer quadratic programming technique, an MPC-controller that achieves accurate path tracking and asymptotic stability is designed and evaluated with simulations.

In an attempt to ease computational load, neural network approaches have been employed. They seek to provide mappings between the tractor-trailer's current state, the required input that leads to the desired state and the desired state itself. Backpropagation with a two-layer neural network comprising of 26 adaptive neural elements has been

used in [8]. The learning was done on a computer simulated tractor-trailer with successful results. However, it required thousands of back-ups to train the network, which is the main drawback of this approach. It is not practical to perform the learning with a real tractor-trailer due to the unpredictable behavior during the learning phase, but limiting the learning to simulations means any errors in the model will cause problems in the controller. Article [9] builds upon the previous one with a new set of dynamic equations and uses a three-layer feedforward neural network. Their most accurate network used 8 inputs, resulting in nodes of (8-17-1), as they followed the Kolmogorov model. It is not easy to compare the results of these two approaches as the latter deals with docking tasks where the goal state is much further from the initial state as compared to the former.

Fuzzy controllers enable the construction of nonlinear control by the use of heuristics. The design of a fuzzy controller for the reverse motion of an articulated vehicle is not an easy one due to the existence of multiple errors. Nonetheless, an approach proposed by [10] manages to design a fuzzy controller by incorporating the line-of-sight method. The effectiveness of this controller is demonstrated with numeric simulations, showing a small cross-track error on the trailer.

A two-level feedback control system with a path stabilization controller in the first level and a hitch angle controller in the second level is proposed in [11]. The strength of this control scheme lies in its simplicity and clarity (with tuning parameters having clear physical meanings). Experiments were conducted with a real tractor-trailer on different paths and the performance of the controller was compared to a range of human drivers of different skill levels. The controller proved to give similar results as a "professional", i.e. a human driver of the highest skill level, albeit with a lower velocity. The controller used in the autonomous driving system in this thesis is inspired by this controller.

2.2 Path planners

The subject of path planning is one that has been studied extensively and there are a wide variety of different approaches that can be found in the literature. Generally, one can divide these approaches into local and global planners.

Local planners operate with short-sighted tasks, such as avoiding a nearby obstacle, and are for this reason also known as short-term planners. The insect inspired bug algorithm family is a well-known example of local planners. The way they work is simply by drawing the path straight towards the goal point from the start point, following the borders of obstacles whenever they are encountered until the goal point can be

pursued again. In [12], an improved bug algorithm is presented and it is shown to have outstanding obstacle avoidance efficiency as well as generating shorter paths when compared to two classic bug algorithms, Bug2 [13] and Rev[14].

Another well-known local planner is the one based on potential fields [15]. The idea behind it is to represent the workspace with potential fields, the goal with attractive potential and the obstacles with repulsive potential. The resulting potential-contours are then used to derive a feasible path. One should however be careful when using this approach in a heavily cluttered workspace as the potential fields might cancel out one another, yielding unexpected and undesired results.

A path planner that uses the concept of Voronoi diagram is a unique type of local planner that aims to maximize the margins between the planned path and the obstacles. Although being used predominantly as a local planner, Voronoi diagrams have been shown to also work as a global planner. Article [16] proposes a global planner based on Voronoi diagrams in combination with the fast marching method and demonstrates its safe path generating ability.

The inability of planning in the global scope can be a crucial limitation, especially in environments with higher complexity and when dealing with nonholonomic constraints, which is why many turn to global planners when developing autonomous driving systems. Unlike local planners, global planners possess a more holistic planning. A popular approach is the grid-based search [17] that represents the workspace with nodes and edges, which can be thought of as states and state transitions. Finding a path from one state to another corresponds to the aggregation of the edges that connect the states that the robot need to go through to reach the goal. A conventional grid-based search algorithm is the regular A^* [18], which uses a heuristic cost function to determine how the search procedure should be executed. Although being a *complete* algorithm, meaning it will always find a solution if there is one, the use of A^* is hampered by its inefficient nature as it re-plans from scratch at every session despite the fact that most of the edge costs remain unchanged when, e.g., detecting a new obstacle.

An algorithm that handles this problem well is the D^* algorithm [19]. In the case of a new obstacle being detected, it only updates the nodes that are affected. The focussed D^* is an extension of the aforementioned algorithm and gives better performance as it redistributes the computational load of the planning process more evenly [20]. Both the original and focussed D^* algorithms have been widely used in the world of autonomous vehicles but there has been a steady decrease ever since the development of the D^* Lite by Koenig and Likhachev. Despite having many strategies in common with the focussed D^*, e.g. conducting the search from the goal point to the current point and minimizing the reordering of the priority queue in a similar way, the D^* Lite is not based on it or

the original D^*. Rather, it is based on the lifelong planning A^*, a previous algorithm by the developers. The advantages of the D^* Lite over its two mentioned family members are proven and demonstrated experimentally in [21].

For problems that involve static obstacles and nonholonomic constraints, the rapidly exploring random tree (RRT) has been shown to be a suitable choice of algorithm [22]. Its strength lies in, as the name implies, the fast exploration of substantially large areas. This is achieved by iteratively generating a random state in the state space and extending the closest branch in the tree towards this state. Another nice property it possesses is the simplicity with which it allows the application of nonholonomic constraints. The main disadvantage of this planner comes from its stochastic nature. That is, given a difficult problem, there is no guarantee it will find a solution even if it has previously managed to do so. Nonetheless, the RRT planner is in fact probabilistically complete, meaning the probability of it finding a solution (if there is one) increases towards one as time goes on.

The Dubins path planner has been widely used in the field of robotics ever since its introduction by the renowned mathematician Lester Dubins in 1957 [23, 24]. Unlike the aforementioned planners, the Dubins path planner does not merely find a path to a goal point, but rather to a goal pose. Meaning the orientation at the goal is considered, which gives it the ability of handling parking tasks. This, along with the fact that it generates the shortest path to the goal pose, constitutes the main strength of the Dubins planner. Nevertheless, it does have the drawback of not being able to deal with obstacles.

In this thesis, the Dubins path planner together with the RRT planner are used to develop a path planner that is able to complete the docking task in environments with static obstacles.

Chapter 3

Mathematical model of tractor-trailer

3.1 Kinematic model

To model the dynamics of the tractor-trailer in a two-dimensional space, using the standard Cartesian coordinate system (x, y), at least four states will be needed. Two states to obtain the orientation of the articulated vehicle and two states to obtain the position. The orientation states will be given by the orientation angles θ_1 and θ_2 of the tractor and trailer relative to the x-axis, respectively. The position states can be the coordinates of really any point on the tractor-trailer. For the sake of simplicity, however, the chosen point here will be the center of the trailer's wheel axle, (x_C, y_C). Figure 3.1 shows a model of the tractor-trailer system along with these states.

Note that the hitch angle, the angle between the tractor and trailer, is $\delta = \theta_1 - \theta_2$. We can now set up an expression for the kinematic model with the input signals being the tractor's velocity v (defined positive in the forward direction) and its steering angle ϕ.

The kinematic model:

$$\begin{aligned} \dot{x}_C &= v\cos(\theta_1 - \theta_2)\cos(\theta_2) \\ \dot{y}_C &= v\cos(\theta_1 - \theta_2)\sin(\theta_2) \\ \dot{\theta}_1 &= \frac{v\tan(\phi)}{L_1} \\ \dot{\theta}_2 &= \frac{v\sin(\theta_1 - \theta_2)}{L_2} \end{aligned} \tag{3.1}$$

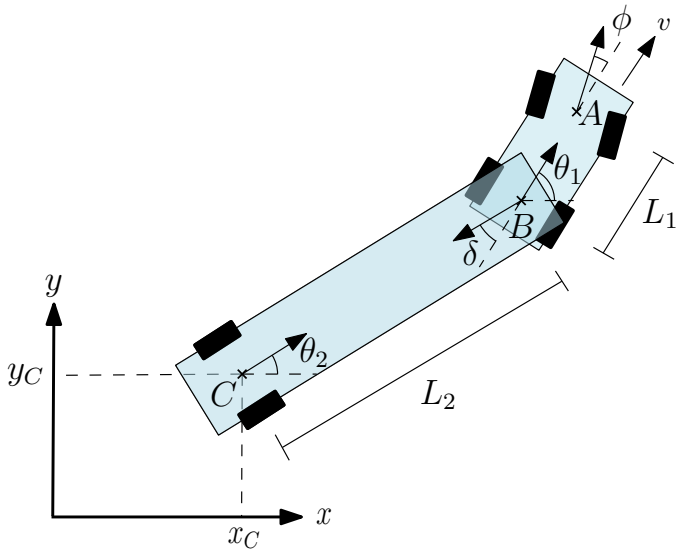


FIGURE 3.1: Model of a tractor-trailer where the tractor has one on-axle hitched semitrailer.

3.2 Circulating state

From (3.1) a relationship can be established between the angular velocity of the hitch angle and the steering angle as

$$\dot{\delta} = \dot{\theta}_1 - \dot{\theta}_2 = \frac{v\tan(\phi)}{L_1} - \frac{v\sin(\delta)}{L_2} \tag{3.2}$$

The system will thus have two equilibrium points with regards to δ, one stable and the other unstable. The stable equilibrium point is given when the trailer spins halfway around the hitch point, $\delta = \pi$. However, this point is not possible to reach due to the tractor being in the way for the trailer completing the half spin. The unstable equilibrium point, on the other hand, is given when the tractor-trailer is circulating around a fixed point. We will refer to this state as the *circulating state* (it should be noted that the straight reverse motion of the tractor-trailer, $\phi = \delta = 0$, is just a special case of this state where the circulating point is at infinity). The steering angle ϕ_{circ} in the circulating state (where $\dot{\delta} = 0$) is obtained from (3.2) as

$$\phi_{\text{circ}} = \tan^{-1}\left(\frac{L_1\sin(\delta)}{L_2}\right) \tag{3.3}$$

Increasing the steering angle in this state ($\phi > \phi_{\text{circ}}$) results in a decreasing hitch angle ($\dot{\delta} < 0$), and decreasing it results in the opposite, as can be deduced from (3.2) (recall

that v is negative in reverse motion). Thus, a direct relationship for control is established as the steering angle together with the velocity constitutes our two control variables.

Note that the number of control variables is less than the number of state variables, which results in our system being a nonholonomic system [25]. A nonholonomic system is simply a system in which one does not have full control. Hence, it is no surprise our system is nonholonomic as we do not have full control over the movement of the tractor-trailer, e.g. we cannot make it slide sideways to get to a certain coordinate point. This can be compared to the movement of a holonomic agent, such as a human, which would be completely unrestricted as it would be possible for it to go straight to any coordinate point.

3.3 Jackknife avoidance

The circulating point P around which the tractor-trailer is circulating can be obtained by having the perpendicular trajectories of all wheel pairs in the tractor-trailer intersect [26], as shown in Figure 3.2. The fact that the steering angle has a maximum point brings the question: what if the circulating state is reached with the maximum steering angle and one would like to straighten out the tractor-trailer? Unfortunately, this cannot be done without a forward motion since a decrement in the hitch angle will require an increment in the steering angle. In other words, the tractor-trailer is stuck in this position. Or even worse if the hitch angle increases slightly, causing trajectory R_A in Figure 3.2 to intersect R_C before R_B. If this occurs with the maximum steering angle, the jackknife effect will be inevitable since the system will start to converge to the stable equilibrium, where the two bodies occupy the same place. That is, any way the tractor-trailer is reversed will only increase the hitch angle and eventually cause the two bodies to collide. This is what is known as the jackknife effect.

To avoid the jackknife effect and always be able to increase or decrease the hitch angle, the circulating state should never be reached with the maximum steering angle. Another way to put it: the hitch angle should never be allowed to reach the point where it together with the maximum steering angle forms the circulating state. In order to calculate this hitch angle, which from now on will be called the critical hitch angle δ_{crit}, assume that the steering angle in Figure 3.2 is the maximum, $\phi_{\max}$. Simple trigonometry yields

$$\delta_{\text{crit}} = \sin^{-1}\left(\frac{L_2}{R_B}\right) = \sin^{-1}\left(\frac{L_2 \tan(\phi_{\max})}{L_1}\right) \tag{3.4}$$

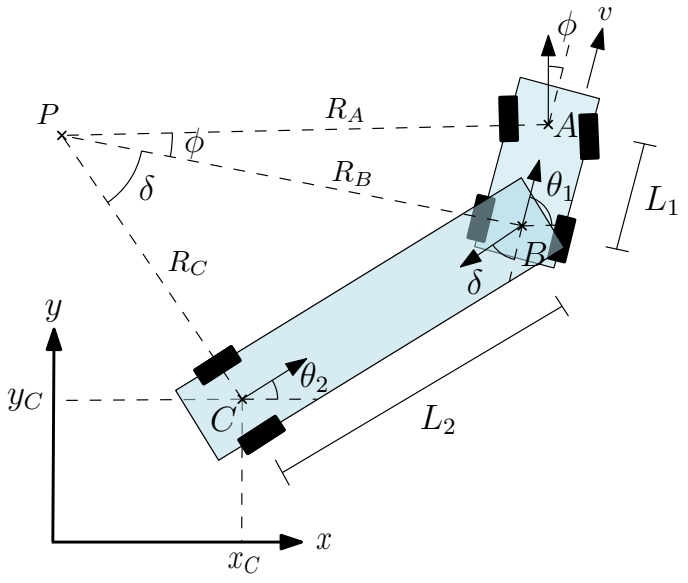


FIGURE 3.2: Tractor-trailer in circulating state with the circulation center point in P.

Thus, the critical angle is dependent on the length ratio between the tractor and trailer and the maximum steering angle. Once the critical hitch angle is known, the jackknife effect can be avoided by simply placing a restriction on the system so that the maximum hitch angle allowed will not reach this angle, $\delta_{\max} < \delta_{\text{crit}}$.

It is interesting to note that there are times where one does not have to worry about the jackknife effect as there will be no critical hitch angle. This is easy to see from (3.4) with the fact that the function $\sin^{-1}(x)$ is defined for $|x| \leq 1$. In other words, the critical hitch angle will not exist if the lengths of the tractor and trailer together with the maximum steering angle satisfy the condition

$$\frac{L_2 \tan(\phi_{\max})}{L_1} > 1 \tag{3.5}$$

In this case, the maximum hitch angle should be set to 90 °. The reason for this is that the vehicle would be able to achieve a sharper turn with a larger maximum hitch angle until it reaches a perpendicular angle, at which point it is able to turn on the spot, .i.e. with the point (x_C, y_C) fixed. Hence, there is no need to ever go beyond this angle, even though it is possible in the absence of the critical hitch angle. However, it should be mentioned that it is not common to find a tractor-trailer that does not have a critical hitch angle and the avoidance of the jackknife effect is therefore something that one should keep in mind when working with this system.

Chapter 4

Testbed

The testbed used for the implementation and validation of the autonomous driving system is setup in the Smart Mobility Lab at KTH and will be presented in this chapter. It is important to analyze the equipment that will be used before starting the actual development of the system. This is because having the knowledge of the limitations and the real-life impediments that exist from the start makes it easier to develop a system that will be able to handle them. The testbed consists of a remote controlled miniature tractor-trailer whose controller has been dismantled and connected to a National Instruments acquisition board. The board in turn is connected to a PC running MATLAB. Via these connections, MATLAB commands are used to control the tractor-trailer. The pose of the tractor-trailer is estimated with a motion capture system which is also connected to the PC. An illustration of the network that makes up the testbed is given in Figure 4.1. The workspace in which the tractor-trailer is controlled is a plain 4 x 4 m surface and there are five obstacles available, four of which have surfaces of 0.6 x 0.4 m and one has a surface of 0.4 x 0.3 m.

4.1 Pose estimation

4.1.1 Qualisys motion capture system

The pose estimation of the tractor-trailer is achieved with a Qualisys motion capture system [27], which uses infra red cameras for tracking. The tracking of a body is accomplished by attaching special markers to it that reflect infra red light. With twelve cameras strategically mounted around the workspace in the testbed, a good coverage is obtained.

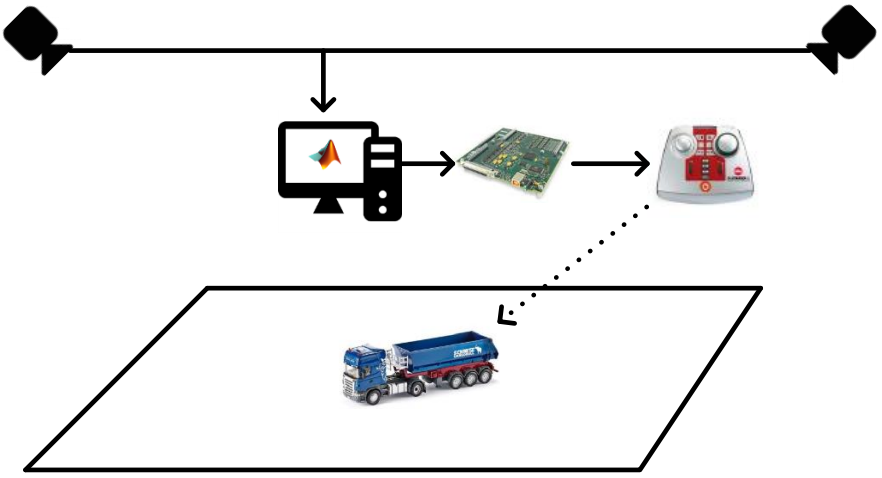


FIGURE 4.1: The network that makes up the testbed. The pose of the tractor-trailer is estimated with a motion capture system that is connected to a computer, from which the tractor-trailer can be controlled. This control is done via an acquisition board which is connected to a remote controller.

The motion capture system can be integrated with MATLAB which enables the use of real-time data with simple commands. Hence, by attaching markers to the tractor and trailer to form two separate bodies for the motion capture system to track, their poses can be obtained in real-time from MATLAB.

4.1.2 Measurement reliability

A small experiment was performed to evaluate the reliability of the measurements given by the motion capture system. The tractor-trailer was kept static in the workspace and measured with a frequency of 10 Hz. This was done for a time duration of ten minutes, resulting in 6000 measurements. Modeling the pose measurements with Gaussian distributions gives the standard deviations shown in Table 4.1. These low values indicate a high measurement precision.

4.2 The tractor-trailer

The remote controlled tractor-trailer is composed of a 1:32 scale model of the Scania R620 towing vehicle and a halfpipe tipper trailer from the Schmitz-Cargobull brand.

Pose measurement	Standard deviation σ
x	$1.0 \cdot 10^{-4}$ m
y	$2.2 \cdot 10^{-4}$ m
θ	$4.0 \cdot 10^{-2}$ °

TABLE 4.1: Standard deviations of pose measurements.

The wireless transmission of the tractor-trailer is based on a 2.4 GHz radio technology and enables control of the steering and driving functions of the tractor as well as coupling and tipping the trailer [28].

4.2.1 Dimensions

The dimensions of a vehicle are essential in the development of an autonomous driving system. One reason is their use in the collision prevention, and another is that many controllers require them in their computations. They are, however, of special importance in our case due to their use in the jackknife avoidance. The measurements of the full length of the tractor and trailer, L_{tractor} and L_{trailer}, together with the dimensions shown in Figure 4.2 are given in Table 4.2. Note that both the tractor and trailer have the same width L_w and that the full length of the tractor-trailer L_L is less than the sum of the lengths of the separate bodies due the overlapping way of which they are coupled. Also, the tractor-trailer does not have a perfect on-axle hitching as the hitching point is slightly in front of the tractor's wheel axle. This deviance from the mathematical model in place could be a source of error in the future but will probably be overridden by other ones since it is so small.

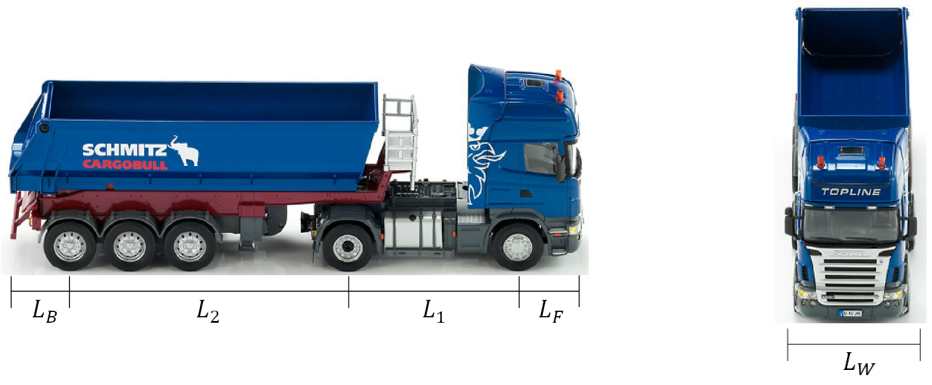


FIGURE 4.2: Dimensions of the tractor-trailer.

Dimension	Measurement (mm)
L_1	118
L_2	192
L_B	40
L_F	44
L_W	88
L_L	394
L_{tractor}	187
L_{trailer}	280

TABLE 4.2: Measurements of the tractor-trailer's dimensions.

4.2.2 Velocity and steering estimation

Estimation

The velocity and steering of the tractor are not given and have to be estimated. This can be done by using the tractor pose (x, y, θ) given from the motion capture system and the sample time T_s. The velocity is estimated as

$$v(t) = \frac{\sqrt{\big(x(t) - x(t - T_s)\big)^2 + \big(y(t) - y(t - T_s)\big)^2}}{T_s} \tag{4.1}$$

and the steering as

$$\phi(t) = \tan^{-1}\left(L_1 \frac{\omega(t)}{v(t)}\right) \tag{4.2}$$

where the angular velocity $\omega(t)$ is obtained by

$$\omega(t) = \frac{\theta(t) - \theta(t - T_s)}{T_s} \tag{4.3}$$

The lower the sampling time, the better the results are in theory. However, in reality, the pose measurement will be afflicted with some degree of measurement noise, which will then be amplified with a smaller sample time. Thus, one has to keep this trade-off in mind when setting the sample time.

Filtering

Although setting the sample time to a larger value will decrease the effect of the measurement noise, it will usually not bring it down to a satisfactory level. Some sort of filtering

should therefore be performed on the measurements. The median filter is a nonlinear filtering technique that is well-known for its outstanding ability of removing outliers. Using the median filter with k previous measurements, the filtered measurement $v^*(n)$ of the velocity measurement $v(n)$ becomes

$$v^*(n) = \text{median}\big(v(n), v(n-1), ..., v(n-k)\big) \tag{4.4}$$

Using a big value for k will cause a filtered signal to have a longer settling time compared to the original signal, which results in a bad filtering. Although this effect is much less than in the mean filter, it does exist and presents a trade-off for the choice of k. The best result in our case was found to be given by $k = 2$. The result brought by the median filter can be seen in Figure 4.3.

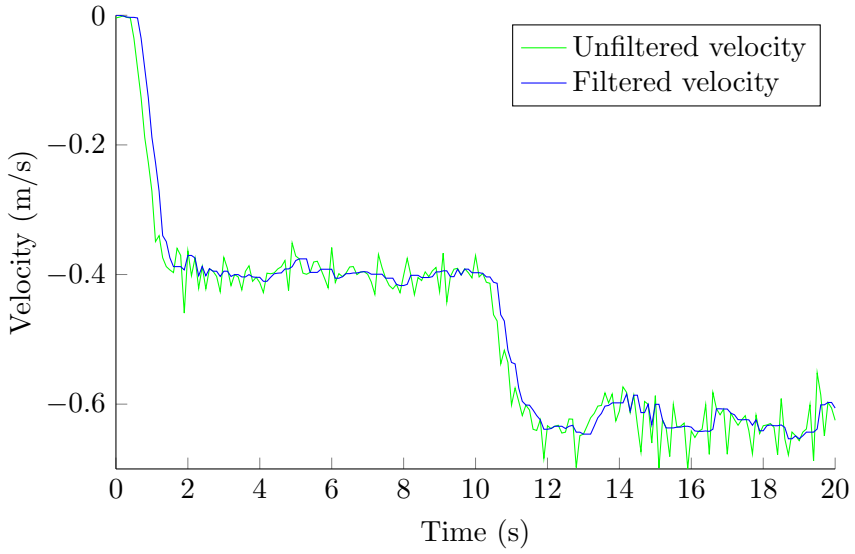


FIGURE 4.3: Filtering the velocity with the median filter using $k = 2$.

Since the steering angle is computed using the estimates of the velocity and the angular velocity, one can obtain a filtered steering angle by either using the raw estimates in the computations and then filter or by using the filtered estimates. Using the former approach has shown itself to be better, see Figure 4.4.

4.2.3 Actuator input-output relations

Both the velocity and steering actuators in the tractor have a working region of 0–3.2 V. For the velocity actuator, the lower half of the working region corresponds to reversing

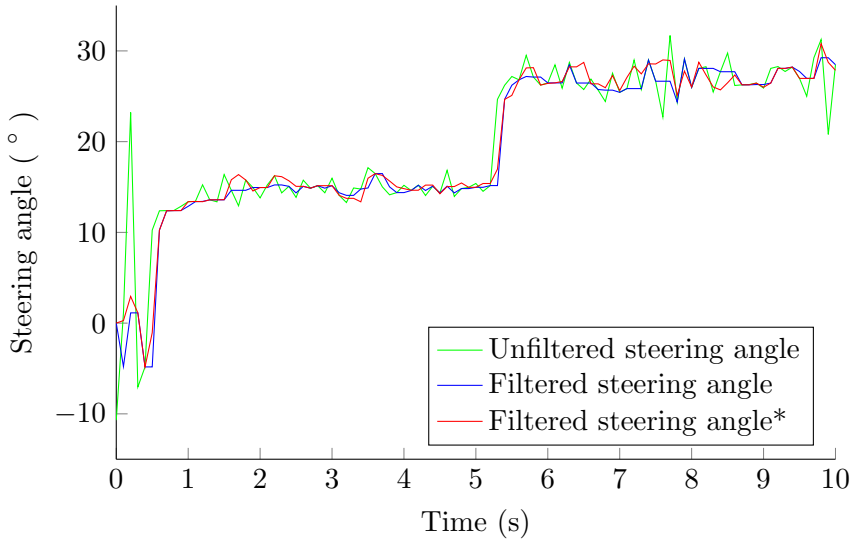


FIGURE 4.4: Filtering the steering angle with the median filter using $k = 2$. The first filtering is done using the raw estimates of the velocity and angular velocity and applying the median filter. The second filtering * is achieved by computing the steering angle from the median filtered estimates.

the tractor, with 0 V giving the maximum reversing velocity and 1.6 V giving stoppage. The upper half corresponds to forward driving, with 3.2 V giving the maximum velocity. For the steering actuator, the lower half corresponds to turning the wheels to the left, with 0 V giving the maximum left turn and 1.6 V giving straight steering. The upper half corresponds to turning the wheels to the right, with 3.2 V giving the maximum right turn.

In order to get an idea of the input-output relation of the actuators, experiments were conducted were the voltages were gradually increased and decreased over the desired working region, which for the velocity is only the lower half of the full working region (since we will only be dealing with the reverse motion). The experiments were performed with a time duration of 120 s. The reason for using such a long time duration is to diminish the effect of inertia.

Figure 4.5 and 4.6 shows the input-output relation for the velocity and steering actuators. As can be noted, the reverse velocity stops increasing when further decreasing the voltage below 0.5 V. This is due to the actuator saturation and the same phenomenon can be seen in the steering actuator. Also, the effect of inertia is visible at the stoppage velocity 1.6 V, with the increasing-voltage-curve having a velocity at a higher voltage. There is obviously a deadband/neutral-zone after 1.6 V before the velocity starts becoming positive (which can be revealed to be at 1.98 V). The higher velocities seems rather

noisy. However, our working region will be in the lower voltages where it is smooth and, hence, increasing the filtering will not be necessary. Another phenomenon that is possible to realize from the steering actuator's input-output relation is the one of hysteresis, i.e. the gap between the curves which is not likely to be caused by the inertia due to the changing rate of the voltages being so low.

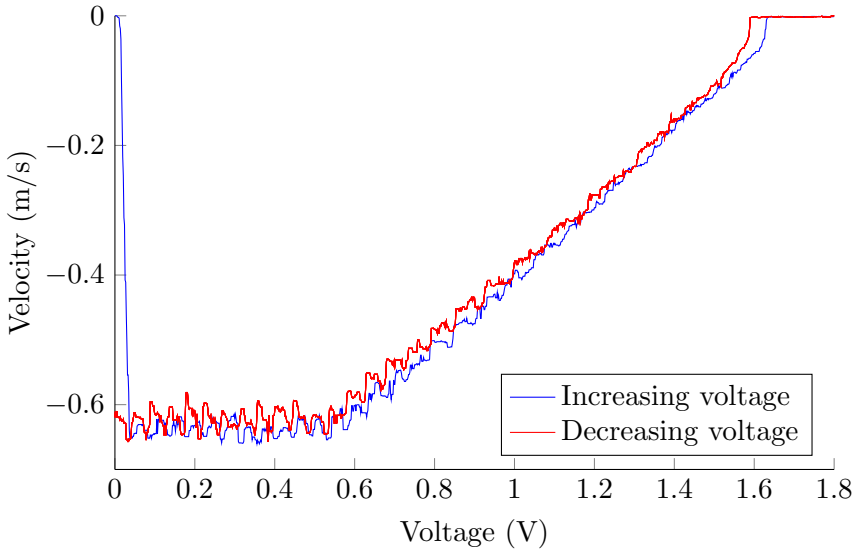


FIGURE 4.5: Input-output relation of the velocity actuator with increasing and decreasing input voltages.

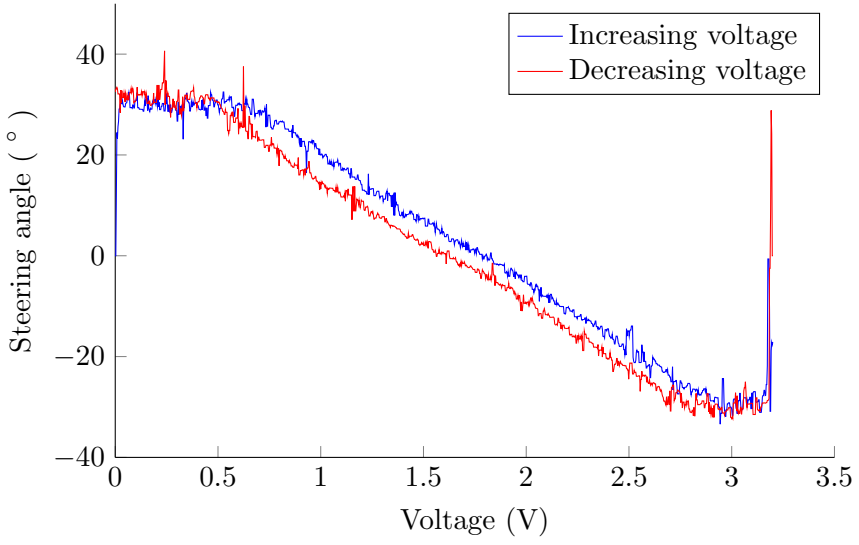


FIGURE 4.6: Input-output relation of the steering actuator with increasing and decreasing input voltages.

Furthermore, the minimum velocity of the tractor can be seen from the figure to be around -0.08 m/s. This is obtained with the minimum voltage that can overcome the friction and inertia to get the tractor moving. The steering angle on the other hand is seen to be able to attain a maximum value of 30 °. However, the measurement of the steering angle becomes quite noisy over 20 °. Due to this, and the general rule of keeping a margin to the maximum values in order to avoid problems, the maximum steering angle will be set to 20 °.

4.2.4 Actuator delays

The actuator delays, i.e. the delay from when an input command is given until the actuator starts reacting to it, for both the velocity and steering actuators in the tractor are measured with step response experiments. Figure 4.7 and 4.8 show the delays of the velocity and steering actuators to be approximately 0.4 s and 0.2 s, respectively. Luckily, the steering actuator has the shorter time delay of the two since the steering angle is more important than the velocity from a control perspective. This is because the former will typically be changing constantly during the control process while the latter will be held constant.

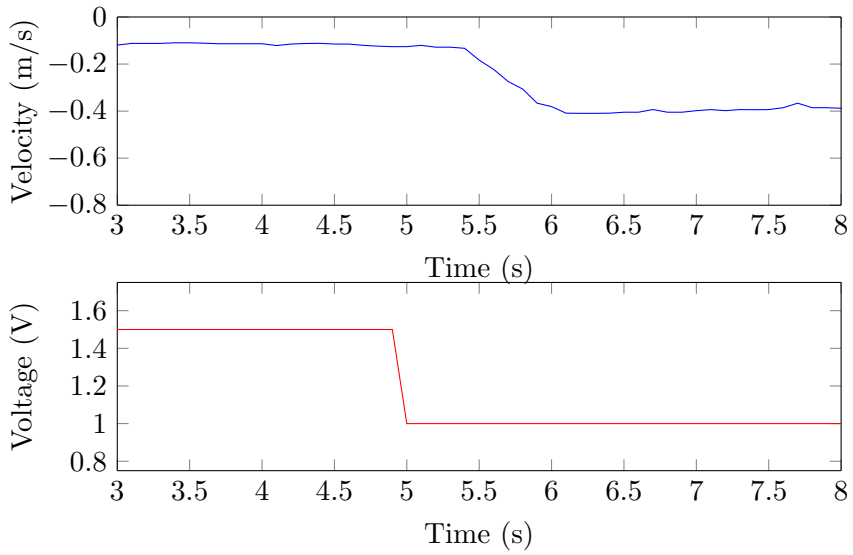


FIGURE 4.7: A step response of the velocity actuator to measure the actuator delay, which is seen here to be approximately 0.4 s.

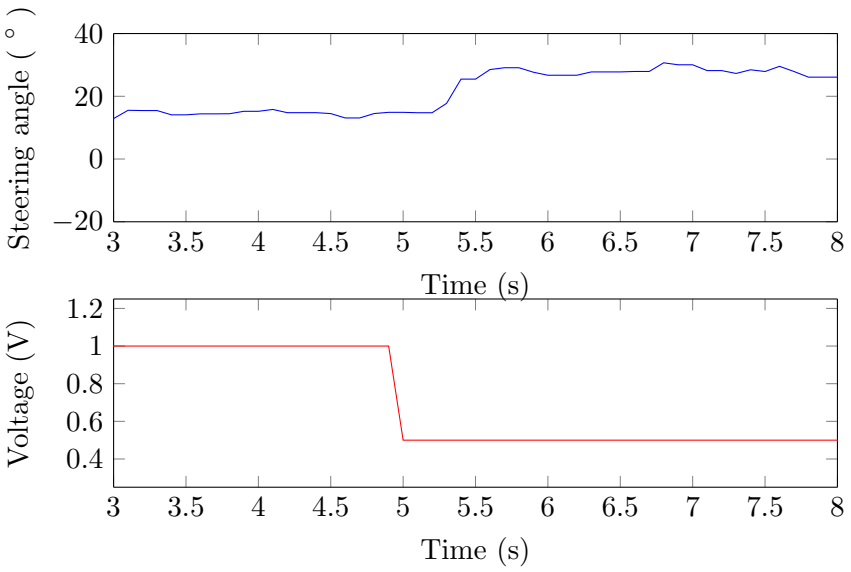


FIGURE 4.8: A step response of the steering actuator to measure the actuator delay, which is seen here to be approximately 0.2 s.

4.3 System constraints and limitations

With a maximum steering angle of 20 ° and the dimensions given in Table 4.2, we can conclude that a critical hitch angle exists in our tractor-trailer, see section 3.3. Meaning, it will be possible for the tractor-trailer to jackknife, and the jackknife avoidance discussed in the aforementioned section will thus be relevant in this case. The critical angle is determined to be 36.3 ° and, to keep a good safety margin, the maximum hitch angle is set to 30 °. This results in a minimum turning radius of 0.33 m, although we will set it to 0.35 m.

The minimum achievable velocity is of interest due to the fact that a lower velocity improves the controllability of the system. While the minimum velocity of the tractor is about 0.08 m/s, attaching the trailer raises the minimum velocity a little due to the added friction as it results in more power being used. The friction makes itself more noticeable when the tractor-trailer is reversing with a large hitch angle because of the slip angle on the trailer's front wheels [29]. Therefore, the front wheels of the trailer are removed, leaving it with only one wheel axle. After doing so, the tractor-trailer's minimum velocity in reverse is dropped to the same as the tractor's. It should be noted that this velocity is relatively high, corresponding to 9.22 km/h (2.56 m/s) for the full scaled tractor-trailer. This is more than twice as high as the typical velocity, 4.03 km/h, with which a professional truck driver reverses a tractor-trailer [11].

Since the steering angle will be the main control signal in the motion controller, the delay of the steering actuator becomes more relevant than that of the velocity actuator. For all intents and purposes, the system can be modeled with a pure delay of 0.2 s.

The maximum angular velocity of the steering angle can be determined by acquiring a big step response from the steering actuator and computing the changing rate of the steering angle. Such a step response can be seen in Figure 4.9, from which the maximum angular velocity is determined to be approximately 120 °/s. However, we will be setting it to 90 °/s in our model.

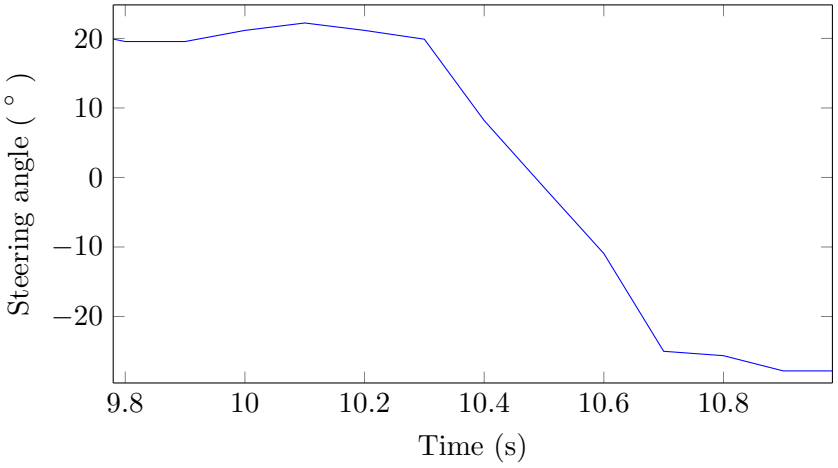


FIGURE 4.9: A step response of the steering actuator, in which the angular velocity is approximately 120 °/s.

Moreover, the trade-off in control performance and noise amplification has resulted in our sample time choice of 0.1 s. That is, the control rate will be set to 10 Hz.

4.4 Simulation

Aside from being used to run the experiments in the testbed, MATLAB will also be used for the development of the autonomous driving system. To save time, before any experiments are made in the testbed, the system is first made to work in a simulation environment created in MATLAB. The simulation is made by using the differential equations in the mathematical model of the tractor-trailer along with MATLAB's `ode45` integrator, which is based on an explicit Runge-Kutta method, the Dormand-Prince pair. By applying the system constraints and inducing the measurement noise and actuator delay into the simulation, one can get an idea of how well the system will do in practice.

Chapter 5

Motion control

An autonomous driving system needs a robust motion controller to be able to follow a determined path. The forward motion of a tractor-trailer vehicle forms a stable system, which means that one can ignore the presence of the trailer and develop a controller with only the tractor in mind. This is not the case for the reverse motion since the system it forms is inherently unstable, as previously discussed, and it becomes necessary to take the trailer into consideration.

Also, in the reverse motion of a tractor-trailer, the steering angle influences the hitch angle, and the hitch angle in turn is what directly influences the way the vehicle turns. That is, as opposed to the case of forward motion, the influence of the steering angle on the way the vehicle turns is indirect, and the hitch angle can in this case be seen as the actual control variable for steering the vehicle.

The control approach that will be employed here is inspired by [11], with many of the differences occurring due to the fundamental difference in the type of tractor-trailer used, ours being on-axle hitched while theirs being off-axle hitched. The motion controller is based on a two-layer control loop; the outer control loop stabilizes the system to the path and the inner control loop stabilizes the hitch angle. We will refer to the outer and inner control loops as the path stabilization controller and the hitch angle controller, respectively.

5.1 Path stabilization controller

The path to be followed here is made up of a set of closely placed points. A good way to estimate if these points are close enough is by making the smallest circular path possible, which depends on the vehicle's minimum turning radius, and connecting its points with

lines. If the path still looks like a circle and not a polygon, it should be good enough for the motion controller to follow. After this point, increasing the density of the points will be redundant and should therefore be avoided. Another requirement that can be placed on the path to facilitate the computations is that each point should in addition to the position have information about its orientation and the curvature of the path segment in which it lies. That is, instead of computing this information on the fly in every iteration, it is computed once and for all in the beginning. Therefore, in a way, each point can be viewed as a tractor-trailer pose.

The stabilization of the tractor-trailer to the path relies on three error measurements: the lateral error $e_{\tilde{y}}$, the orientation error e_θ, and the curvature error e_c. Together, these so-called path following errors can be used to decide which hitch angle the tractor-trailer should take on, i.e. the reference hitch angle δ_{ref}, in order to follow the path. This is achieved with the following linear control law:

$$\delta_{\text{ref}} = K_{\tilde{y}} e_{\tilde{y}} + K_\theta e_\theta + K_c e_c \tag{5.1}$$

with $K_{\tilde{y}}$, K_θ, and K_c being tuning parameters. These parameters are in essence speed dependent. Nevertheless, since the velocity in backward motion is typically in the lower region, these tuning parameters can be regarded as speed independent [30].

5.1.1 Path reference point

The path following operation is carried out in a way that resembles greyhound track racing. The role of the artificial rabbit in the latter is played by a moving reference point P_r on the path in the former. The reference point is chosen by placing a point, which we will refer to as the *search point* P_s, a distance d_s behind the trailer and determining which point in the path it is closest to. The reason for using the search point instead of simply determining which path point is closest to the tractor-trailer, e.g. the midpoint P_t of the trailer's wheel axle, can be seen in Figure 5.1. If the chosen reference point would have been set as the one closest to P_t, i.e. P_0 in the figure, it would have been too close to the tractor-trailer which inevitably would have lead to a more aggressive steering towards the path. Choosing the closest point to P_s instead would result in a reference point a bit further down the path, i.e. P_1 in the figure, and consequently a smoother steering towards it.

The choice of the distance d_s is a delicate one. Choosing a too short distance, and much of the desired effect from the search point, i.e. the smooth steering, will be lost. On the other hand, if the chosen distance is too long, the vehicle will react to path segments

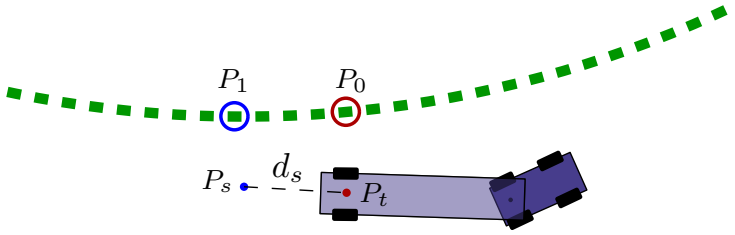


FIGURE 5.1: Choosing the reference point P_r as the closest point in the path to the midpoint P_t of the trailer's wheel axle yields a point P_0 that is too close to the vehicle and will result in an aggressive steering towards the path. Using the search point P_s instead to find the reference point gives a point P_1 further down the path, and a smoother steering towards it.

that lie far ahead rather than the one it is currently on. It will also obtain a violent and unpredictable behavior since a small change in the trailer's orientation will result in a substantial change in the location of the search point, leading to big jumps in the change of reference point.

A path might sometimes intersect itself as in Figure 5.2, which could be problematic when updating the reference point since the new reference point might all of sudden lie in the crossing segment of the path and not the current one. Due to this, it is wise to only use the current segment of the path when determining the closest point to the reference point. The length of this segment should of course be shorter than the shortest possible segment needed to make an intersection. In other words, it has to be shorter than the circumference of the smallest circle the path can possibly form, which is a circle with a radius corresponding to the minimum turning radius r_{min} of the tractor-trailer. Also, since the reference point should only be moving in the forward direction, the path segment used to determine the next reference point should start from the current one. This means that the path segment used in the search will initially have a constant length but towards the end of the path, when the remaining part of the path is just as long as itself, it will start to decrease until the end point of the path is reached.

5.1.2 Path following errors

Longitudinal and lateral errors

The lateral error $e_{\tilde{y}}$ is used to make the tractor-trailer converge to the reference point in the path. To determine this error one has to consider the tractor-trailer's own coordinate system; the Cartesian coordinate system in which the origin is defined as the midpoint of the trailer's wheel axle and that has the same orientation as the trailer. Figure 5.3

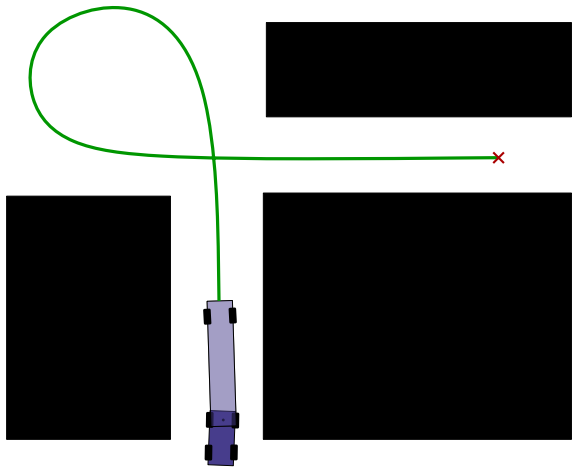


FIGURE 5.2: An example of a case where the path might intersect itself.

illustrates this coordinate system along with the position errors to a reference point (x_r, y_r, θ_r) in the path. Although not used in the path following procedure per se, the longitudinal error $e_{\tilde{x}}$ can be of great benefit and should therefore be determined in this step as well. From the figure, the longitudinal and lateral errors can be determined as

$$\begin{aligned} e_{\tilde{x}} &= e_x \cos(\theta_2) + e_y \sin(\theta_2) \\ e_{\tilde{y}} &= e_y \cos(\theta_2) - e_x \sin(\theta_2) \end{aligned} \tag{5.2}$$

with e_x and e_y being

$$\begin{aligned} e_x &= x_r - x_C \\ e_y &= y_r - y_C \end{aligned} \tag{5.3}$$

It is easy to see whether the lateral error is positive or negative by checking in which side of the trailer the reference point lies. If it lies on the right side (as in Figure 5.3), the lateral error is negative, and positive if the reference point lies on the left side.

Orientation error

The orientation error e_θ is used to make the trailer have the same orientation as the reference point. The simple answer for how to determine the orientation error is $e_\theta = \theta_r - \theta_2$. However, one should keep in mind that the orientation error should be computed in the direction in which it is smaller, since we evidently want the tractor-trailer to take the shortest way to the path. So a rectification becomes necessary in the case where the

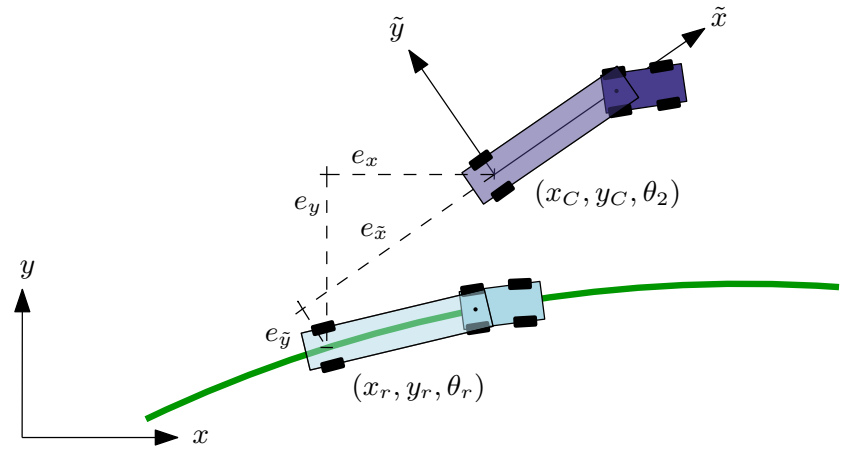


FIGURE 5.3: The longitudinal and lateral errors, $e_{\tilde{x}}$ and $e_{\tilde{y}}$, along with the position errors, e_x and e_y, to a reference point (x_r, y_r, θ_r) in the path.

magnitude of the orientation error is larger than π, which is the largest orientation error possible when going in the shortest direction. A more complete answer for determining the orientation error is therefore

$$e_\theta = \begin{cases} \theta_r - \theta_2, & \text{if} \quad |\theta_r - \theta_2| \leq \pi \\ \theta_r - \theta_2 - 2\pi \cdot \text{sign}(\theta_r - \theta_2), & \text{otherwise} \end{cases} \tag{5.4}$$

To illustrate the need for the rectification procedure, consider the case where the current trailer orientation is $\theta_2 = 2\pi/3$ and the reference orientation is $\theta_r = -\pi/2$. Without the rectification procedure in (5.4), the result would have been $e_\theta = -7\pi/6$, while its application yields $e_\theta = 5\pi/6$.

Note that the problem above arises from the fact that the orientation is bounded within a region (and rightly so), which results in a discontinuity at the region's start and end points. The problem occurs when the discontinuity point is found between the current orientation and the reference orientation point in the shorter direction. Had the region in the example above been $[0, 2\pi]$ instead of $[-\pi, \pi]$, i.e. with $\theta_r = 3\pi/2$, the discontinuity point would have lied between the two orientation points in the longer direction instead and no rectification procedure would have been needed. A problem in this case would have occurred if the current orientation was $\theta_2 = \pi/3$. In other words, the rectification procedure will always be a necessary step as long as the orientation is bounded within a region, which it should to be for practicality.

Curvature error

The curvature error e_c is used to make the tractor-trailer form the same curvature as the path segment in which the reference point lies, i.e. the reference curvature. The reference curvature is determined by the circle formed by the reference point together with its two neighboring points. There are different ways of determining a circle from three points; a simple one is given in [31]. The curvature reference is simply the inverse of this circle's radius r_p. Depending on the direction of the circle, the radius is either positive or negative, in order to differentiate between right and left turns. The direction of this circle will always be the same as the direction of the curve in which the reference point lies, i.e. clockwise when the curve is turning right and counterclockwise when turning left. To know which way the curve is turning, one can simply look at how the orientation changes in the curve. An increasing orientation means the curve is turning left and a decreasing orientation means it is turning right. We will define the radius to be positive for right turns and negative for left turns. Hence, the range of the reference curvature $1/r_p$ in the path is $[-1/r_{\min}, 1/r_{\min}]$. A straight line has an infinitely large radius and thus zero curvature, as expected.

Each hitch angle in the tractor-trailer has a corresponding circular state that it can form with a certain steering angle. The curvature of the tractor-trailer is given by the inverse of the radius r_t of the circle which is formed by this circular state. The way the sign of the radius is defined here must be the same as for the path segment. In other words, the radius should be positive for clockwise circles, i.e. when the hitch angle is positive.

Figure 5.4 shows the radiuses used to determine the curvatures of the reference point and the tractor-trailer. Once both the reference curvature and the tractor-trailer's curvature are known, the curvature error can be determined as

$$e_c = 1/r_p - 1/r_t \tag{5.5}$$

5.1.3 Tuning the path stabilization controller

Before starting the tuning procedure, the sign of the tuning parameters $K_{\tilde{y}}$, K_θ, and K_c should be determined. The way to go about this is to set up a scenario with all path following errors non-zero and isolate each and every term in the controller (5.1) with the aim of determining whether the term should generate a positive or negative change to the hitch angle. If the sign of the error in the isolated term and the change in the hitch angle it should generate are the same, the corresponding tuning parameter will have a positive sign. If they don't, the parameter will have a negative sign.

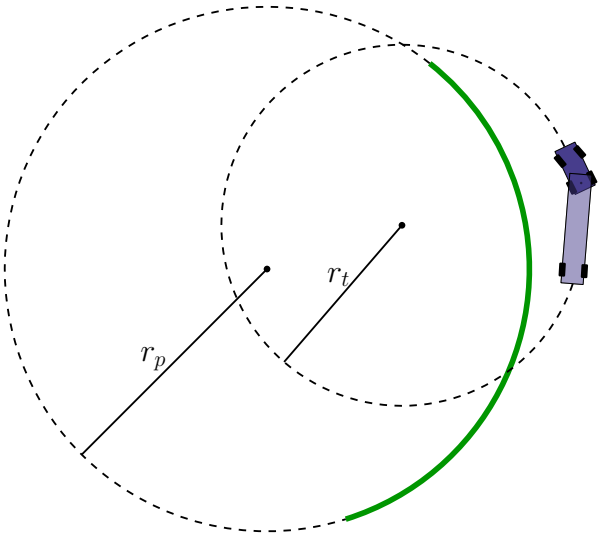


FIGURE 5.4: The turning radius r_t of the tractor-trailer and r_p of the path are used to determine the curvature error.

Figure 5.5 shows a scenario where the path following errors are non-zero. Starting with the lateral error, which is clearly positive since the reference point P_r lies in the left side of the trailer, the desired behavior to diminish this error while reversing is to increase the hitch angle. Hence, the tuning parameter $K_{\tilde{y}}$ must be positive. The next term is the one of the orientation error, which is positive as the reference point lies on the perfectly straight path that has a larger orientation value than the trailer. In order to diminish the orientation error the hitch angle should start to decrease, meaning the parameter K_θ is negative. The tractor-trailer in the figure has a positive hitch angle, resulting in a positive curvature while the path, being a straight line, has a curvature value of zero. This gives a negative curvature error which needs the hitch angle to decrease for it to diminish, hence, yielding a positive parameter K_c.

One way to tune the parameters $K_{\tilde{y}}$, K_θ, and K_c is to start the tractor-trailer parallel to a straight path with a small lateral error. The parameters $K_{\tilde{y}}$ and K_θ can initially be set so as to give the maximum reference hitch angle once certain errors have been reached, e.g. once $|e_\theta| \geq 30\,^\circ$. The parameter K_c, on the other hand, should initially be set to zero. The controller should be able to stabilize to the straight path with only these two parameters, which at this point should be tuned to give as smooth of a performance as possible. The role of the curvature term comes into play when the controller is operating on a curvature. As soon as both the lateral and orientation errors have been eliminated, the tractor-trailer will start straightening out (since the new reference hitch angle will now be zero). Once it has diverted a bit from the path, producing some new errors, it

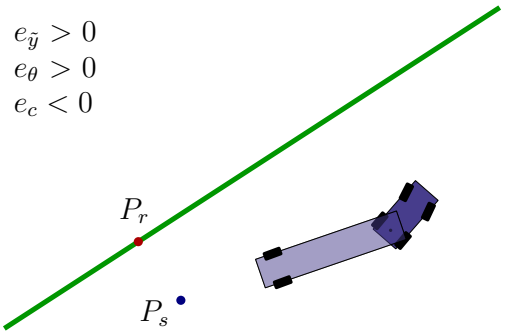


FIGURE 5.5: A scenario with all path following errors non-zero to aid the sign determination of the tuning parameters.

will start converging back to the path. This wobbly behavior is what is treated with the curvature error. To tune the parameter K_c, the tractor-trailer should be made to follow the smallest circular path possible while gradually increasing this parameter until the wobbly behavior has been removed.

5.2 Hitch angle controller

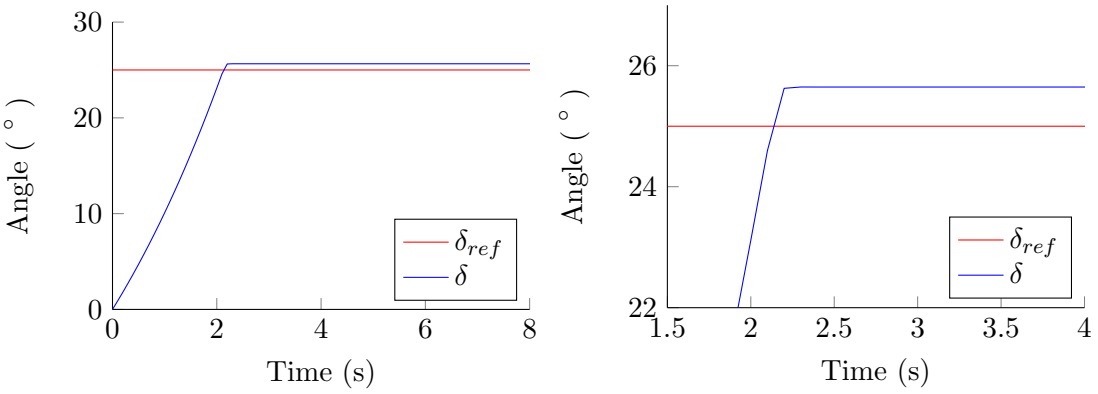
Once the reference hitch angle δ_{ref} is obtained from the path stabilization controller, a hitch angle controller is used to compute the required steering angle ϕ. Two different controllers will be presented here, one linear and the other nonlinear.

5.2.1 PI controller

The hitch angle control can be accomplished fairly well with a simple PI controller:

$$\phi = K_P(\delta_{\text{ref}} - \delta) + K_I \int_0^t (\delta_{\text{ref}} - \delta)dt \tag{5.6}$$

where K_P and K_I are tuning parameters (and the time dependency of δ and δ_{ref} is omitted for simplicity). However, due to non-linearities, a steady state error will always present itself for non-zero operating points in the P controller, i.e. with only the proportional part of the PI controller. This can be seen in the step response of the P controller in Figure 5.6. The integral part of the PI controller will of course handle this error, but some time will go to waste in doing this as it is done gradually, which results in a somewhat slower controller. This problem is dealt with in [11] by a pre-modification of



(A) Full (B) Zoomed in

FIGURE 5.6: A steady state error appears in the P controller due to non-linearities.

the reference hitch angle. The corresponding modification expression in our case (since the tractor-trailer model here is different from theirs) becomes

$$\bar{\delta}_{\text{ref}} = \left(1 - \frac{L_1}{K_P L_2}\right) \delta_{\text{ref}} \tag{5.7}$$

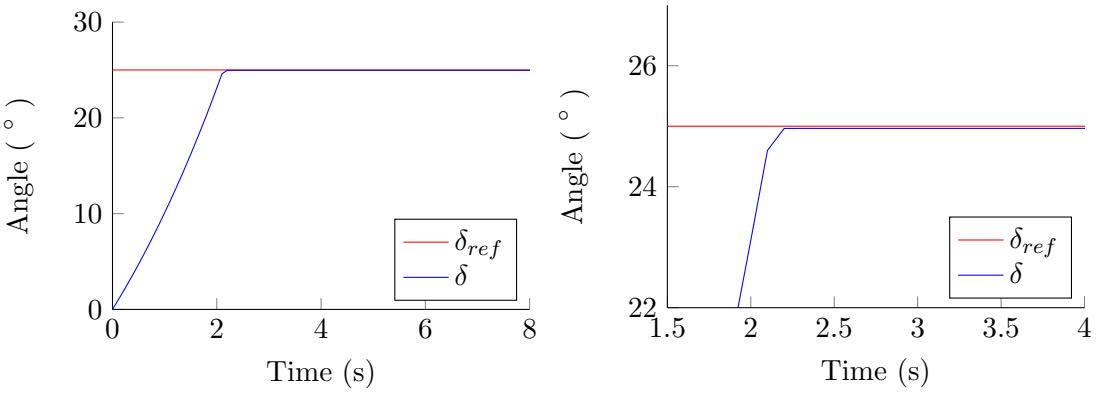
This modified reference angle is only used in the proportional term in the PI controller. Thus the modified PI controller becomes

$$\phi = K_P(\bar{\delta}_{\text{ref}} - \delta) + K_I \int_0^t (\delta_{\text{ref}} - \delta)dt \tag{5.8}$$

The improvement brought by this modification can be seen in Figure 5.7. It almost eliminates the steady state error in the P controller completely.

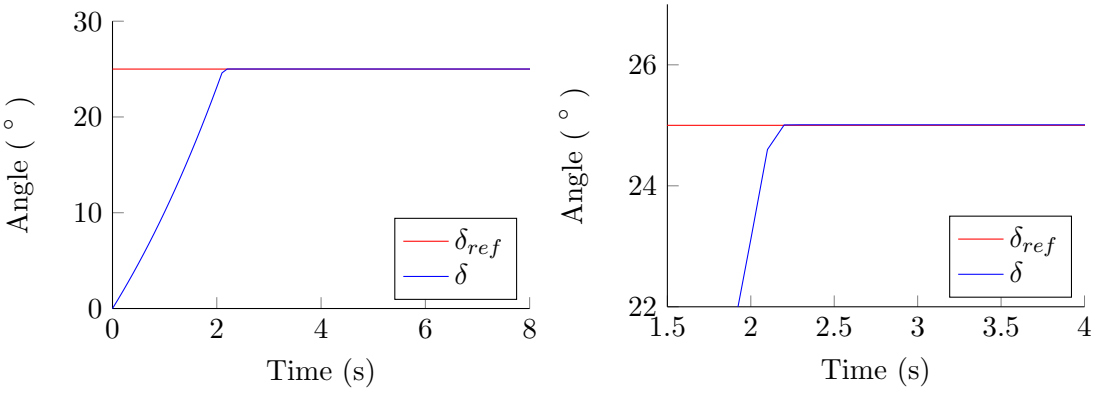
Incorporating the integral at this point will eliminate the steady state error completely but this at the cost of an overshoot. This overshoot comes from the integration windup, the fact that the integral term accumulates during the rise, which can be remedied with a variety of integral anti-windup solutions. One simple solution is to put a constraint on the maximum value of the integral term. The performance of the PI controller with the modified proportional part and the integral anti-windup can be seen in Figure 5.8.

As mentioned in the tuning of the path stabilization, one should determine the sign of the tuning parameters before starting with the actual tuning work. Having a positive hitch angle error while starting with a zero hitch angle means the hitch angle should be increased which requires the steering to be done to the right. According to our definition, a right steering angle is negative. Hence, the tuning parameters in (5.8)



(A) Full (B) Zoomed in

FIGURE 5.7: The steady state error is almost eliminated from the P controller after applying the modification to the reference angle.



(A) Full (B) Zoomed in

FIGURE 5.8: The steady state error is completely eliminated with the integral part in the PI controller.

should be negative. Nevertheless, since both of them are negative, one can place the negative sign outside the entire term and work with positive parameters.

There are many ways of tuning the PI controller and quite a few heuristics have been developed for this purpose [32]. A standard method is to set $K_P = 1$ and $K_I = 0$, and stabilize the system to a zero hitch angle. The proportional gain K_P should be increased until oscillation starts to appear. After optimizing K_P, the hitch angle should instead be stabilized to a non-zero point in order to reveal the steady state error. The integral gain K_I should then be increased with the aim of removing this error. One can expect to see an overshoot in this step, which can be counterbalanced by decreasing K_P.

5.2.2 Lyapunov controller

Lyapunov theory is one of the most important tools in the analysis of nonlinear dynamic systems. Lyapunov's direct method, in particular, is frequently used in the stability analysis and control design of these systems. In essence, it states that if a system's total energy, which is based on its physical properties, is decreasing, the system will eventually reach an equilibrium point, resulting in the system being stable. The system is asymptotically stable if the energy is strictly decreasing, meaning the energy can be brought down to zero from every state in the domain (feedback stabilizable). An energy function that fulfills Theorem 5.1, i.e. the theorem of Lyapunov's direct method for which the proof can be found in [33], is called a Lyapunov function. Determining this function is considered to be the main challenge in the design of a Lyapunov controller [34].

Theorem 5.1. (**Lyapunov's direct method)** *If there exists a continuously differentiable positive definite function $V(x)$ whose derivative $\dot{V}(x)$ is negative semidefinite, the origin is a stable equilibrium point, and asymptotically stable if $\dot{V}(x)$ is negative definite:*

- $V(x) \geq 0$, *with equality only for* $x = 0$ *(positive definite).*
- $\dot{V}(x) \leq 0$, *(negative definite for asymptotic stability).*

then $x = 0$ *is stable.*

The aim of a controller is to bring a certain error to zero, hence, the Lyapunov function should be based on this error. In our case the error is that of the hitch angle in comparison to its reference. Thus, we can set the Lyapunov function as

$$V(x) = \frac{x^2}{2} = \frac{1}{2}(\delta_{\text{ref}} - \delta)^2 \tag{5.9}$$

This gives the derivative

$$\dot{V}(x) = \dot{x}x \approx \dot{\delta}(\delta - \delta_{\text{ref}}) \tag{5.10}$$

The approximation $\dot{\delta}_{\text{ref}} \approx 0$ made here is based on the assumption that the rate of variation in the reference hitch angle δ_{ref} is low in comparison to the one of the actual

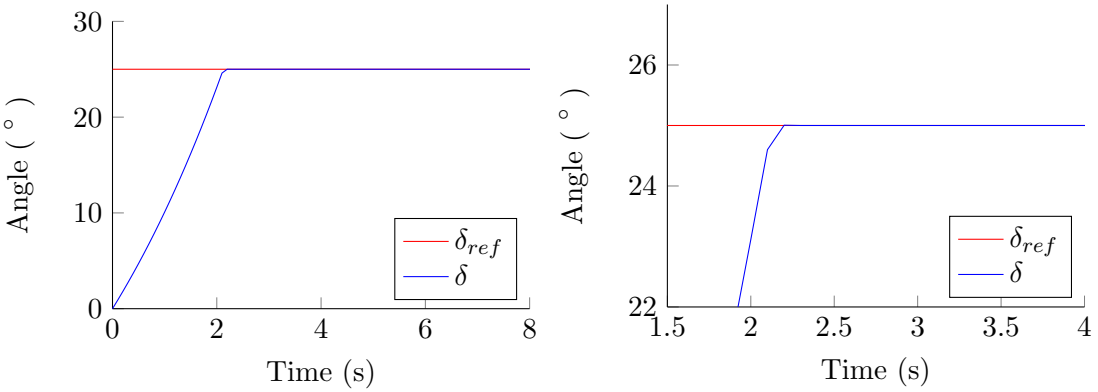
hitch angle δ. In order to fulfill the condition of the $\dot{V}(x)$ being negative definite, $\dot{\delta}$ is set to

$$\dot{\delta} = -K(\delta - \delta_{\text{ref}}) \tag{5.11}$$

where K is a positive tuning parameter. Using the expression given for $\dot{\delta}$ in (3.2), one can extract the steering angle ϕ as

$$\phi = \tan^{-1}\left(\frac{L_1}{v}\left(\frac{v}{L_2}\sin(\delta) + K(\delta_{\text{ref}} - \delta)\right)\right) \tag{5.12}$$

Thus, the Lyapunov controller is established. With only one tuning parameter, the tuning work here is as easy as it gets. Notice also how the system converges to the circulating state as the hitch angle error decreases; the steering angle becoming ϕ_{circ} as the hitch angle error goes to zero. Furthermore, the step response of the Lyapunov controller is shown in Figure 5.9.



(A) Full (B) Zoomed in

FIGURE 5.9: Step response of the Lyapunov controller

5.3 Angular velocity of the hitch angle

The angular velocity of the hitch angle is not constant, which can be compared to the fairly constant angular velocity of the steering angle for forward motion, as both of these serve the same purpose. From the expression of the angular velocity $\dot{\delta}$, see (3.2), it can be seen how it depends on the current hitch angle δ. The effect of the steering angle ϕ, and consequently the hitch angle control, is therefore dependent on the current state of

the hitch angle. Figure 5.10 shows the effect of the (positive) maximum steering angle on the angular velocity with respect to the current state of the hitch angle. The desired effect is to bring the hitch angle down to zero but the changing rate, i.e. the magnitude of $\dot{\delta}$, with which this is done keeps declining as the current state of the hitch angle is increased. At some point ($\delta \approx 36\,^\circ$) the hitch angle will start to increase instead, indicating the jackknife point (the critical hitch angle δ_{crit}).

Simply put, the control effect on the hitch angle will be slower the larger the angle is. This phenomenon can be seen in Figure 5.11. At the points where the hitch angle reference starts reversing, i.e. starts to go towards zero, the controller quickly applies the maximum steering angle to follow it. Although keeping the same steering angle, one can clearly see how the changing rate of the hitch angle increases with time. This is until it reaches the next peak, where the reverse action occurs one more time and the phenomenon is repeated. Hence, it is wise to keep the maximum hitch angle to a low value, this being a trade-off between controllability and maneuverability since a too low maximum hitch angle will make it difficult to steer the tractor-trailer.

Another thing to note about the angular velocity of the hitch angle from Figure 5.11 is how low it is, reaching the maximum magnitude at approximately $10\,^\circ/\text{s}$, see Figure 5.10. This can be compared to the angular velocity of the steering angle, which was found to be around $120\,^\circ/\text{s}$ in our tractor-trailer (see section 4.3). Unfortunately, there is no practical way of increasing the angular velocity of the hitch angle, which is realized from its expression. The possible options are to either change the dimensions of the tractor-trailer (decreasing the length of the wheelbase of the tractor and increasing the one of the trailer), increase the maximum steering angle, or increase the velocity. The former two are obviously not possible in most cases. This leaves the increment of the velocity which in our case is not really an option since it is already relatively high at 0.08 m/s. Moreover, it would have an impairing effect on the controllability and should therefore be avoided.

5.4 Adding look-ahead

The transitions between curves and straight lines in a path can pose a great difficulty on the tractor-trailer due to the low angular velocity of its hitch angle. There are two cases where these transitions occur: 1) going from a straight line to a curve and 2) going from a curve to a straight line. We will assume the curve is the maximum curvature that the tractor-trailer can follow. In the first case, the tractor-trailer starts out with a zero hitch angle since it is following a straight line. Although the angular velocity of the hitch angle will be at its maximum at this point, once it realizes the path has turned (by

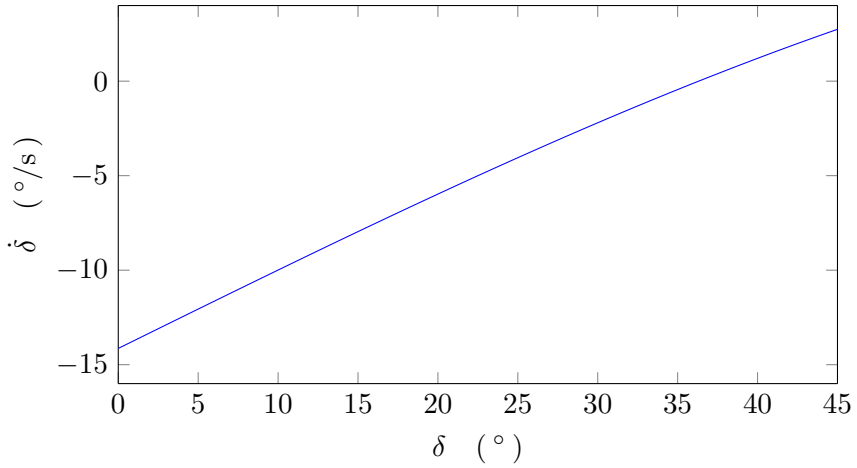


FIGURE 5.10: The effect of a positive maximum steering angle on the angular velocity $\dot{\delta}$ of the hitch angle with respect to the current hitch angle δ.

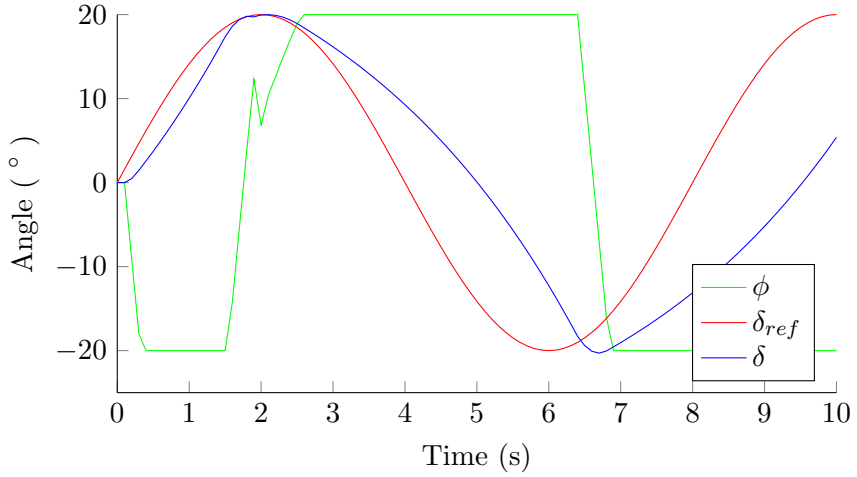


FIGURE 5.11: The rate with which the hitch angle is reversed from 20 ° to −20 ° increases as the hitch angle decreases. This is despite using the same steering angle ϕ throughout this action, which illustrates the control effect's dependency on the hitch angle.

the increase in the path following errors), it will still require some time before getting back on the path. The primary reason for this is that the path is turning away from the tractor-trailer, at an increasing rate since it is a curve, and the angular velocity is just too low to keep up with this turn, despite being at its maximum. Once the tractor-trailer has been given some time to reach the same curvature as the path it will increase this curvature some more and start converging back to the path. In the second case, the tractor-trailer starts with a large hitch angle as it is following a curvature. This

time, the path will not start to turn away from the tractor-trailer after the transition point, quite the opposite, it will stop turning away. However, the angular velocity of the hitch angle will be at its minimum at this point and it will therefore take time for it to straighten out, causing it to lose the path for a moment before converging back. The behaviors of these two cases are shown in Figure 5.12.

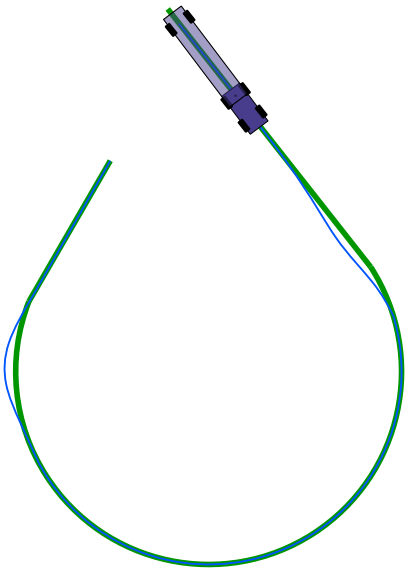


FIGURE 5.12: The behavior of the tractor-trailer caused by its low angular velocity in the two transition cases: 1) going from a straight line to a curve and 2) going from a curve to a straight line. This behavior is remedied by adding look-ahead to the system.

The solution for this problem is to increase the forth-sight of the tractor-trailer, so it has more time to complete the maneuvers that are required for following the path. In other words, a look-ahead should be incorporated to the system. A simple, yet effective, way of doing this is to let the curvature error be computed from a point further down the path, look-ahead point P_l. In our first case, this would mean that the curvature error would tell the tractor-trailer to start folding while it is still on the straight line, and since the two other errors are zero at this time, the effect of the curvature error will be prominent and, thus, have a more immediate influence on the tractor-trailer. This concept is illustrated in Figure 5.13.

The number of points n_p the look-ahead point P_l is ahead of the reference point P_r, should be determined with the time t_l that the tractor-trailer requires to complete the required maneuver before the transition point. Knowing the distance d_p between the points in the path, n_p can be computed as

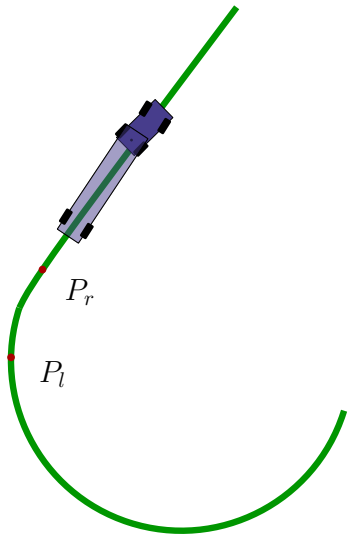


FIGURE 5.13: The look-ahead point P_l reaches the transition point sooner than the reference point P_r and will hence notify the tractor-trailer about the upcoming curve earlier, giving it more time to complete the required maneuver for staying on the path.

$$n_p = \text{round}\left(\frac{|v|t_l}{d_p}\right) \tag{5.13}$$

This gives us the following expression for the look-ahead point:

$$P_l = \min\{P_r + n_p, P_f\} \tag{5.14}$$

where P_f is the final point in the path. The time t_l can be seen as a tuning parameter in this procedure, and one should start with a small value and increase it gradually until the tractor-trailer follows the path without losing it at the transition points.

5.5 Braking and stopping at goal

Although the reverse motion of a tractor-trailer is typically done with a low velocity, slamming the brakes the second it reaches the goal point will still be an uncomfortable stoppage. For a smooth and comfortable stoppage, the velocity should be decreased gradually from a distance before reaching the goal point. This distance, which we will refer to as the *braking distance* d_b, is determined in the longitudinal direction. The braking should be initiated once two conditions are met. The first condition is that the reference point should be the final point P_f in the path to make sure the tractor-trailer

is at the end of the path and not merely passing by. The second condition is that the final point should be within the breaking distance. Once these two conditions are met and the braking has been initiated, the tractor-trailer should be reversing with a velocity v_b, see Figure 5.14. This velocity is directly proportional to the longitudinal error $\tilde{e}_x$, i.e. the remaining distance to the goal point:

$$v_b = v\frac{|\tilde{e}_x|}{d_b} \tag{5.15}$$

where v is the initial velocity of the tractor-trailer. This will result in the desired braking behavior; the velocity will decrease gradually as the tractor-trailer approaches the goal point and it will come to a stop once the longitudinal error has perished.

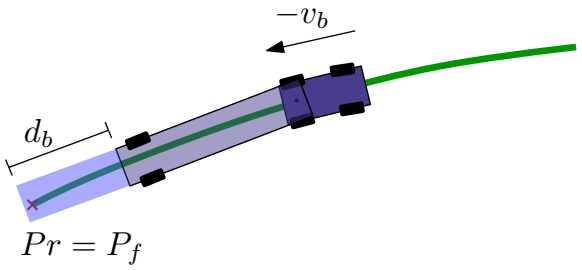


FIGURE 5.14: The braking procedure is initiated once the reference point P_r becomes the final point P_f in the path and lies within a distance d_b from the tractor-trailer. The velocity will at this point be set to v_b (here depicted with a minus sign since the velocity is defined positive in the forward direction), which will gradually decrease as the tractor-trailer approaches P_f.

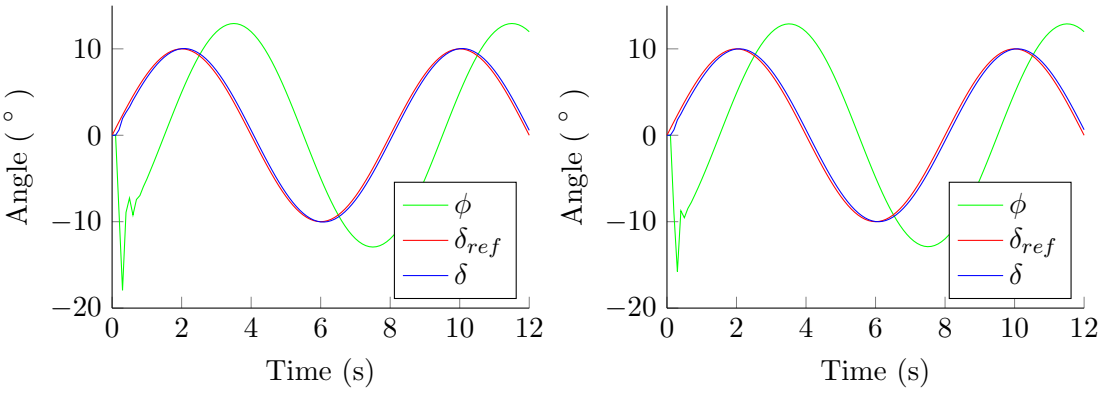
5.6 Simulation results

The system parameters and constraints found in section 4.3 will be used in the simulation of the motion controller. To recap, the velocity is set to $v = -0.08$ m/s and the control rate to 10 Hz. The constraints of the steering angle, the angular velocity of the steering angle, and the hitch angle are given by $\phi_{\max} = 20\,^\circ$, $\dot{\phi}_{\max} = 90\,^\circ$/s, and $\delta_{\max} = 30\,^\circ$, respectively. We will first compare the two hitch angle controllers before viewing the path following results of the motion controller.

5.6.1 Comparing the PI and Lyapunov controllers

The step response of both the PI and Lyapunov controllers are very similar, see Figure 5.8 and 5.9. To compare the controllers' performances with a more realistic task, the

reference hitch angle will be a sine signal with an amplitude of 10 ° and a time period of 8 s.



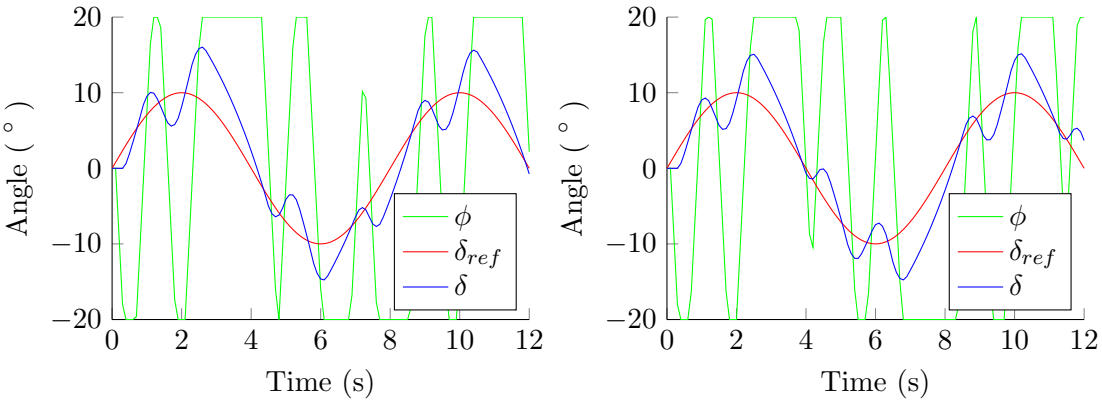
(A) PI controller (B) Lyapunov controller

FIGURE 5.15: Comparing the control performance of the PI and Lyapunov controllers with a sine signal as the hitch angle reference.

As seen from Figure 5.15 the results are once more almost identical and whatever small differences exist seem to be originating from the choice of tuning parameters. To increase the difficulty level, a pure time delay of 0.2 s is introduced to the system. Since the performances of both controllers are virtually the same even in this case, see Figure 5.16, we conclude that none of them is better than the other, at least not in simulation. Given the choice between the two, one might intuitively want to go with the Lyapunov controller, considering how much easier it is to tune. However, the Lyapunov controller uses the velocity in its computations which in reality (in our case) will be an estimate, computed by differentiation, and therefore have some degree of error. Taking this into account, we will go with the PI controller to keep the estimation down to that of the hitch angle.

5.6.2 Path following results

Recall that the maximum hitch angle $\phi_{\max}$ results in a minimum turning radius of 0.35 m. To evaluate the motion controller's ability of handling a curvature with such a small turning radius, a clockwise circular path with this radius was created. The path starts at the right most point and the tractor-trailer was placed right on top of this point with a straight pose facing up (with an orientation of 90 °). The path following results on this path can be seen in Figure 5.17 where the path taken by the tractor-trailer corresponds to that of the midpoint of the trailer's wheel axle. To allow a fair visual evaluation of the performance, the tractor-trailer is also depicted in the figure, since the significance



(A) PI controller (B) Lyapunov controller

FIGURE 5.16: Comparing the control performance of the PI and Lyapunov controllers with a sine signal as the hitch angle reference and a pure time delay of 0.2 s.

of the path following error is relative to the size of the tractor-trailer. In the figure, the initial deviation from the path is caused by the low angular velocity of the hitch angle, i.e. the fact that tractor-trailer requires some time to change its pose to the required one, as previously discussed. Increasing the radius/size of the circular path, and hence decreasing the required change of the hitch angle, reduces the difficulty level of the task and better results can be achieved with the motion controller, see Figure 5.18.

Next, to evaluate the controller's ability of handling transitions between straight lines and curves, a figure-eight path with curvature radiuses of 0.5 m is created. This path has four transitions, testing all possible combinations of transitions: straight line to left curve, left curve to straight line, straight line to right curve, and right curve to straight line. It is good to evaluate the path following on all four but it is enough to evaluate it on two transitions, one going from a straight line to a curve and one going from a curve to a straight line. We will place the tractor-trailer with a straight pose on the lower left transition facing the center, and the path will be set to start from this point with a curve and end with a straight line. This gives us three transitions but also makes it possible to still see how the motion controller handles the initial change requirement of the tractor's pose from the last step. As seen from Figure 5.19, the motion controller's path following performance on this path is excellent.

However, it is just as important to evaluate the actual control behavior of the motion controller throughout the process. This is because a good path following performance can in fact be achieved with a poor control behavior. That is, it is possible for a motion controller to follow a path well while displaying a jerky or even oscillatory steering control action. Figure 5.20 shows the steering control action along with the hitch angle

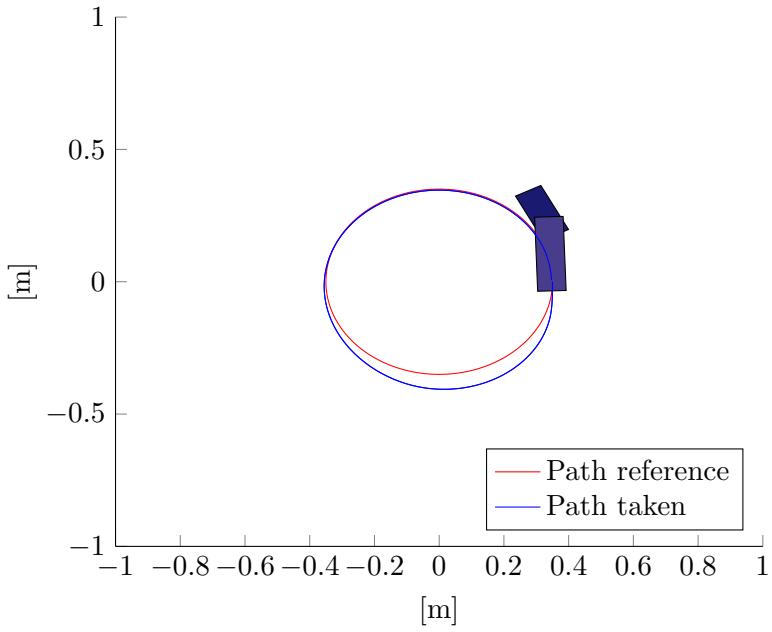


FIGURE 5.17: The motion controller's path following performance on a clockwise circular path with radius 0.35 m. The initial deviation is caused by the large initial change requirement on the hitch angle and the fact that its angular velocity is not able to make this change fast enough.

control performance during the path following of the figure-eight path. A tiny oscillation can be observed in the steering control when settling at a certain angle. There are many ways of dealing with this. For instance, one can use a filter on the controller's steering output or increase the control rate. However, these remedies do come with drawbacks. Using a filter on the control output gives a slower controller and increasing the control rate will amplify noise in the system. Hence, it is important to look at the size and the effect of the oscillation to see if it is worth applying any of these remedies. In our case, the amplitude does not even measure to half a degree, so it is not even detectable with the naked eye when looking at the actual procedure. Also, the hitch angle is not affected by this oscillation, being smooth and steady as seen in the figure. With this in mind, the control behavior of the motion controller is deemed to be a perfectly good one.

To get an idea of what type of results the motion controller would generate in practice with the presence of measurement noise and actuator delays, Gaussian noise with the standard deviations presented in Table 4.1 is induced to the system and a pure delay of 0.2 s is added. The performance of the motion controller gets visibly worse, see Figure 5.21. A noteworthy fact is that the ability to handle the transition from straight line to curve is affected to a lesser degree than the transition from curve to straight line. This is most probably due to the angular velocity of the hitch angle being lower

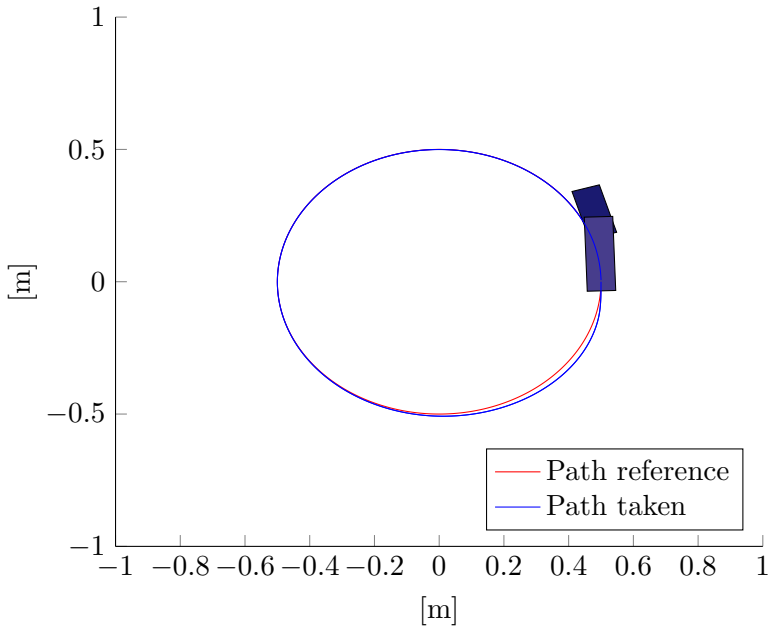


FIGURE 5.18: The motion controller's path following performance on a clockwise circular path with radius 0.5 m. The required initial change in the hitch angle is small enough to be made quickly and the initial deviation is therefore small in this case.

in the transition from curve to straight line as the hitch angle will be larger at that point. The effect of noise and delays will hence be more noticeable in this transition. Nonetheless, although having a poorer path following performance, the real decline of the motion controller is seen in the control behavior, see Figure 5.22. This time the control output has an oscillation with an amplitude greater than 10 ° and a visible effect on the hitch angle. The oscillation in the hitch angle means the tractor is moving from side to side during the path following operation. However, in this case, the movement is quite modest due to the amplitude of the oscillation being relatively small (about 3 °). With a larger amplitude, the result would have been in the best case an uncomfortable driving experience, and in the worst case a collision due to the trajectory of the tractor-trailer being wider than expected. Thus, going by these results, one can predict that the controller will in practice not perform well on a path with a minimum turning radius that is less than 0.5 m.

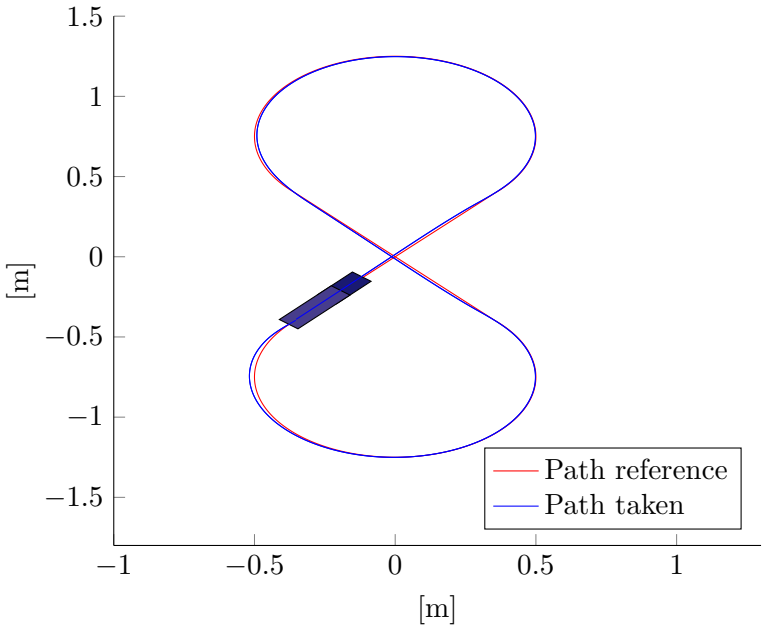


FIGURE 5.19: A figure-eight path with curvature radiuses of 0.5 m is used to evaluate the motion controller's path following performance on transitions.

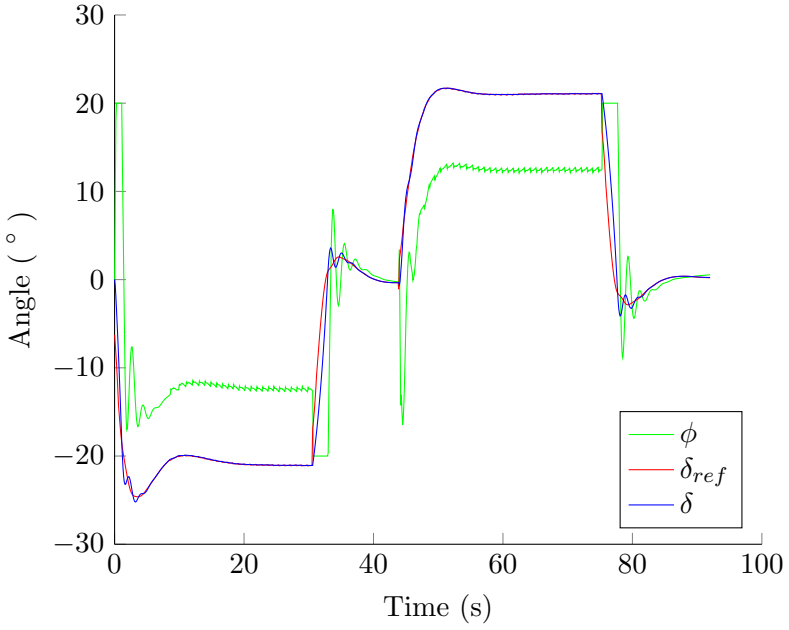


FIGURE 5.20: The steering control action and the hitch angle control performance during the path following operation of the figure-eight path.

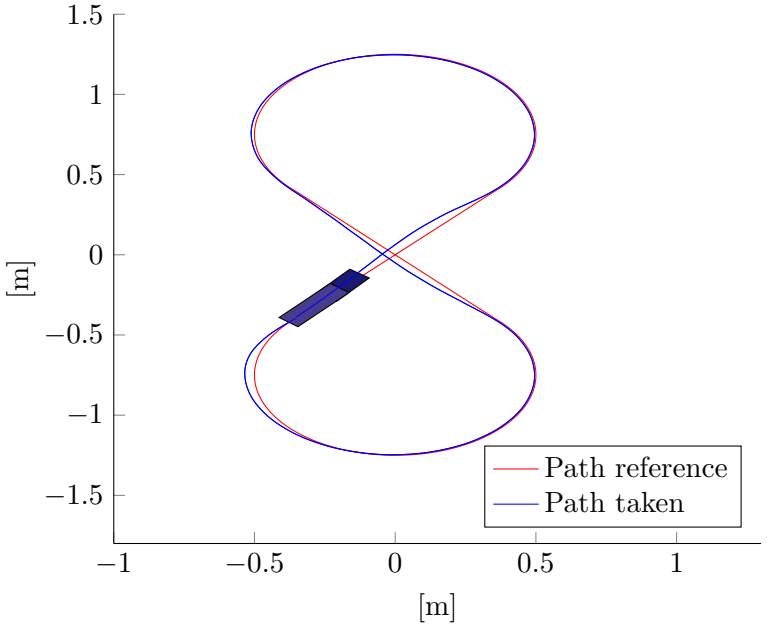


FIGURE 5.21: The path following performance of the motion controller with added Gaussian noise and a pure delay of 0.2 s.

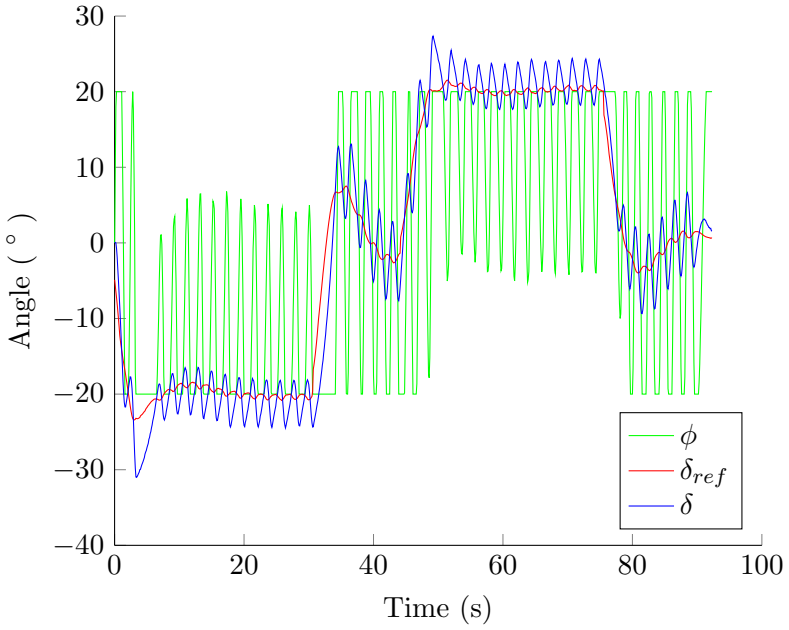


FIGURE 5.22: The steering control action and the hitch angle control performance during the path following operation of the figure-eight path with added Gaussian noise and a pure delay of 0.2 s.

Chapter 6

Path planning

Planning a path for a nonholonomic agent in the presence of obstacles can be a complex task, especially if the task is not merely to reach a region but rather a specific pose. For a car vehicle, the most optimal planner for the parking task in an obstacle-free environment is the Dubins path planner. This planner only requires the car's minimum turning radius and it will always generate the shortest path to the parking space, the so-called Dubins path. Hence, this planner can be used for the docking task of the tractor-trailer by simply giving it the minimum turning radius of the tractor-trailer's reverse motion. This then leaves us with the issue of obstacles. The straight forward idea that will be applied here is to drive to a pose where there are no obstacles in the way of completing the Dubins path. For this approach to be feasible, the workspace should be explored as fast and thoroughly as possible, which is achieved well with the use of the rapidly exploring random tree (RRT). Thus, this chapter will first cover the theory of the Dubins path and the RRT planner, and then present a modified RRT planner that incorporates the Dubins path to achieve a suitable path planner for the docking task of the tractor-trailer.

6.1 Dubins path

6.1.1 Control sequences

The car for which the Dubins path planner was developed is known as the Dubins car. This car drives with a constant velocity and, hence, has only one control variable, namely the steering. The steering control has only three discrete states: turn right at maximum, turn left at maximum, and turn straight. These control actions will from now on be denoted as R, L, and S, respectively. What the developer proved was that

the optimal path from one pose to another for the Dubins car can always be achieved with six types of control sequences: RSR, RSL, RLR, LSL, LSR, and LRL. To simplify matters further, R and L, who both result in curvatures, can be defined as C. In other words, the six control sequences can be described by CSC and CCC. Note that not all the control actions in a sequence have to be applied. Meaning, a path that only requires a right turn will be possible to attain with any of the sequences RSR, RSL, and RLR by suppressing the last two control actions.

6.1.2 Construction of possible paths

The paths given by the six control sequences are determined using merely circles with radius $r_{\min}$ and straight lines. These paths can be divided into two parts, the ones given by CSC and the ones given by CCC. The construction of these paths will be shown here, starting with those given by CSC. First, tangent circles should be drawn to the right and left of the start and goal poses of the Dubins car. The right circles will have a clockwise direction and the left circles will have a counterclockwise direction, which shows the direction the Dubins car would go while following the circles. From these circles, four circle pairs are formed, with one of the circles being from the start pose and the other from the goal pose. Each circle pair has four tangent lines. However, only one of these four lines will be a valid one considering the directions of the circles, see Figure 6.1. Drawing all the valid tangents between each possible circle pair combination from the start and goal poses yields all four possible paths from the control sequences in CSC, see Figure 6.2.

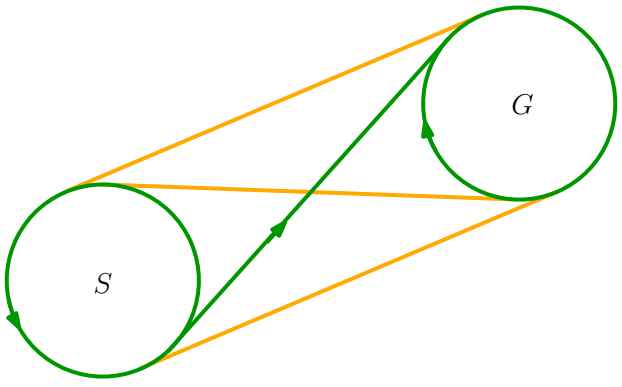


FIGURE 6.1: A circle pair (with circle S being from the start pose and circle G from the goal pose) has four tangent lines, out of which only one is valid considering the direction of the circles.

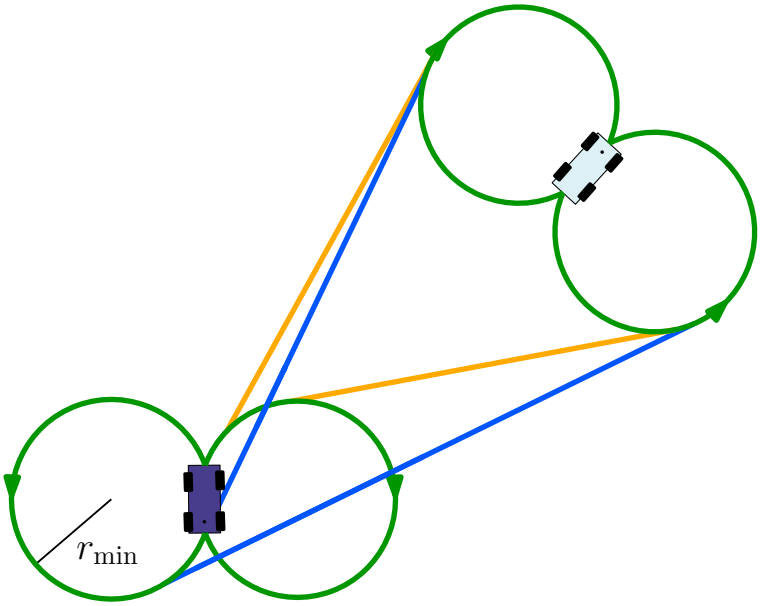


FIGURE 6.2: Drawing all the valid tangent lines between the circle pair of the start and goal poses gives all four possible paths from the control sequences in CSC. The orange tangent lines belong to the right tangent circle of the starting pose (purple car), i.e. starting with control action R, and the blue ones correspond to the left tangent circle, i.e. starting with control action L.

A path from CCC is constructed by drawing a tangent circle to a circle pair with such a direction that the Dubins car can go from the circle of the start pose into it and out to the circle of the goal pose. Note how close the poses have to be for this to be possible. The path generated from sequence RLR in CSC is shown in Figure 6.3. To get the path generated from the other control sequence in CSC, i.e. LRL, one can simply switch the locations of the start and goal poses. The paths generated by CCC are used when they become shorter than the ones generated by CSC, as shown in the aforementioned figure. That is, the parking task could have also been completed with the control sequence RSR in CSC. Nevertheless, that would have generated a longer path, and hence a suboptimal path in comparison to the one from CCC. Note that a path from CCC will not always be shorter than a path from CSC every time the start and goal poses are close enough to each other for it to be formed. This is illustrated in Figure 6.4.

There are numerous methods for the computation of the paths discussed above. These are generally categorized into geometric and analytic methods. A geometric method is presented in [35] and an analytical one is given in [24]. Nonetheless, it is not hard to find ready-made functions for the Dubins path, and we will be using the one found in [36].

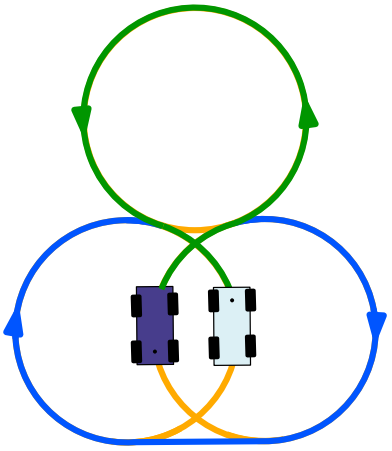


FIGURE 6.3: A case where the path given by CCC is shorter than the path given by CSC. That is, the control sequence RLR (green path) generates a shorter path than that of RSR (blue path).

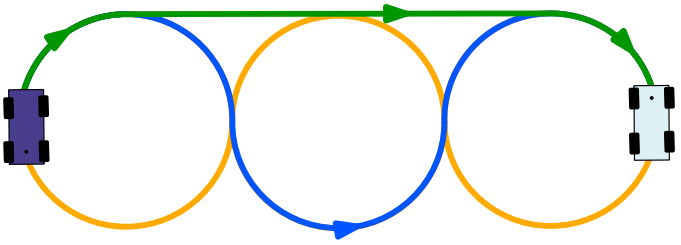


FIGURE 6.4: A case where the path given by CSC is shorter than the path given by CCC. That is, the control sequence RSR (green path) generates a shorter path than that of RLR (blue path).

6.2 RRT

6.2.1 RRT algorithm

A step-by-step explanation of the RRT planner algorithm, see algorithm 1, will be given here. A more detailed description of this planner and its properties can be found in [22]. In the first step, a tree $\mathcal{T}$ is initiated with a node x_{start}, which corresponds to the vehicle's initial pose and serves as the root node in the tree. A node x_{rand} is then generated with a random pose somewhere in the workspace. Next, the nearest node x_{near} in the tree to node x_{rand} is determined. The steering input u that would take the vehicle closer from x_{near} to x_{rand} is then computed. Applying this steering input to node x_{near} and letting the corresponding vehicle pose drive for a time period Δt equates

the node x_{near} branching out towards x_{rand}. A new node x_{new} is formed with the pose where the vehicle stops in this branching and is then added to the tree. This procedure is repeated until the goal region is reached or the number of nodes n in the tree has grown to the max allowed number N.

```
T.init(x_start);
while goal not reached & n ≤ N do
    x_rand ← randomPose();
    x_near ← nearestNeighbour();
    u ← steeringInput(x_rand, x_near);
    x_new ← newNode(x_near, u, Δt);
    T.addNode(x_new);
end
```

Algorithm 1: RRT algorithm

To help the tree arrive at the goal pose, the growth of the tree should be biased towards the region in which it lies rather than growing equally towards all directions. To achieve this, the random pose for x_{rand} is chosen as the goal pose with a probability of P_{greed}, which we will refer to as the *greediness factor*. The choice of the greediness factor of the tree can be crucial for the performance of the RRT planner. Unfortunately, this choice is not easy and depends on a number of factors; the most important of these being the obstacle density. In the case where the area between the start and goal poses is free, having a greedy tree will be a good choice as it will waste less time growing towards other directions than that which leads to the goal pose. On the other hand, the more obstacles between the start and goal poses, the greater exploration of the workspace will be required of the tree. A greedy tree is a bad choice in this case as it is by nature a bad explorer. Figure 6.5 shows a case where a greedy tree will have a poor performance. The tree will have a hard time finding the opening at the upper left corner since it will be too greedy towards the goal pose. Therefore, there is a trade-off here between the exploration of the workspace and the growth of the tree towards the goal.

The Euclidean distance is typically used as the distance metric when determining the nearest node x_{near} to node x_{rand}. However, this distance metric is problematic since it merely reflects the distance between the locations of the nodes and not the travel distance between them, which is more interesting from the perspective of the tree's growth. In other words, the tree might grow in an awkward manner since a node might be selected as the nearest one to x_{rand}, despite having a longer way to reach it than another node in the tree, as shown in Figure 6.6. Hence, for a more just determination of the nearest node, it is important to have a distance metric that takes into consideration the nonholonomic

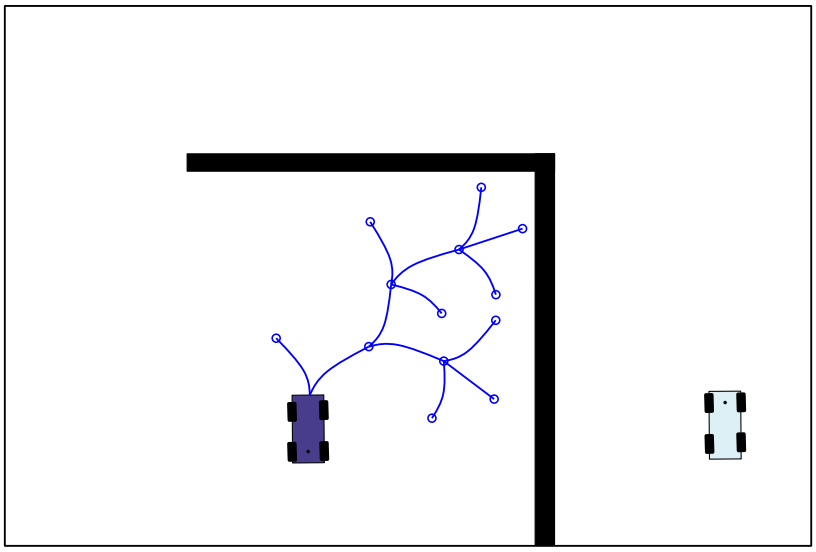


FIGURE 6.5: A case where a tree with a high greediness factor performs poorly. The tree will have a hard time finding the opening at the upper left corner due to its greedy growth towards the goal pose.

constraints of the vehicle. The sensitivity of the RRT to distance metrics is discussed more in [37].

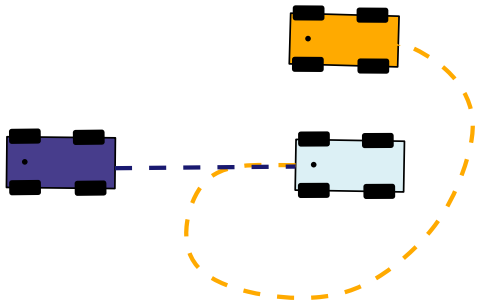


FIGURE 6.6: The choice of the Euclidean distance as the distance metric in the determination of the nearest node is problematic as it does not reflect the distance which the vehicle has to travel. The orange vehicle above will be determined as being the nearest one to the blue vehicle with the Euclidean distance, despite having a longer way to go.

The steering input u is determined from a discrete set of steering angles, usually $[-\phi_{\max}, 0, \phi_{\max}]$. A set of steering angles with a finer discretization would obviously provide a more accurate steering towards x_{rand}, but this would increase the computational load of the algorithm immensely. The reason for this is that the algorithm simulates the vehicle's motion for each possible steering input in the set to determine which one would get the pose of the current node closer to that of the random node.

The process of obtaining the new node x_{new} must include some sort of collision check. Otherwise, it would be possible for the tree to grow into obstacles or workspace boundaries (these will be thought of as walls). A common way of doing this check is by simulating the vehicle's movement with the given steering input for the time period Δt while checking in each iteration step in the simulation if it collides with an obstacle or a workspace boundary. The new node is a valid one if it passes this check, and can subsequently be added to the tree. If it does not, the process is repeated from the top by generating a new random node. Notice that this indicates the presence of another while-loop inside the one in algorithm 1, which is not depicted for simplicity.

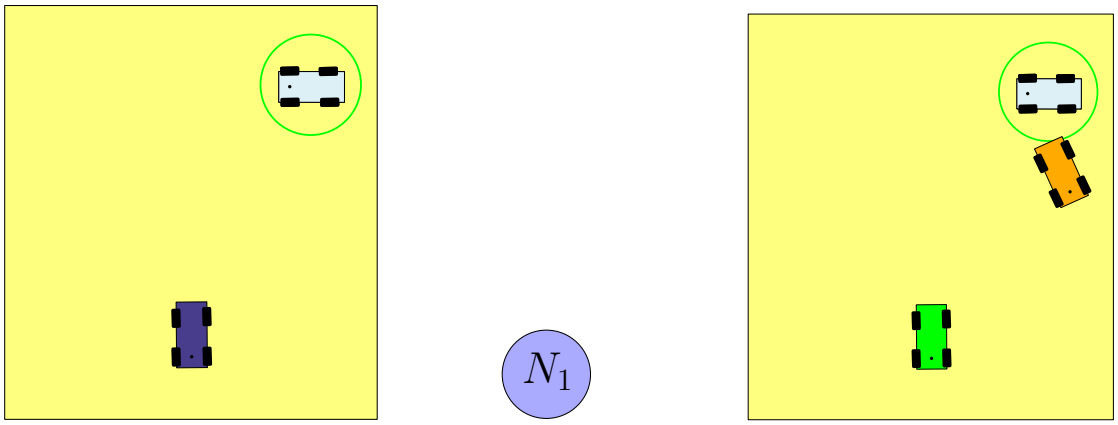
Since the RRT planner will typically not be able to reach the exact goal pose, one will just have to suffice with reaching the goal region. The condition for this is simply done by checking whether the new node that is added to the tree lies within a circle that has its midpoint at the goal pose. The smaller the radius of this circle, the harder the task is for the RRT planner. On the other hand, the larger the radius, the poorer the solution might be since the planned path may be to a point that lies further away from the goal pose. Thus, one should be careful when choosing the radius of this circle.

6.2.2 RRT execution example

A simple execution example of the RRT algorithm is given here in 8 steps. Each two steps are given in one of the figures from Figure 6.7 to 6.10. In the first step, given the start and goal poses, the RRT planner determines the goal region that it should find a path to. Also, the tree is initialized with the starting pose, which becomes the root node N_1. Next, a random node N_{rand} is generated and since there are no obstacles in this case, the algorithm only has to check that the vehicle that corresponds to this random pose is within the borders of the workspace. This process is repeated until it finds a random node that is valid (shown as an orange vehicle in the figure). Once it does, the nearest node to the random node is determined (highlighted in green), which obviously is node N_1 since it is currently the only node in the tree.

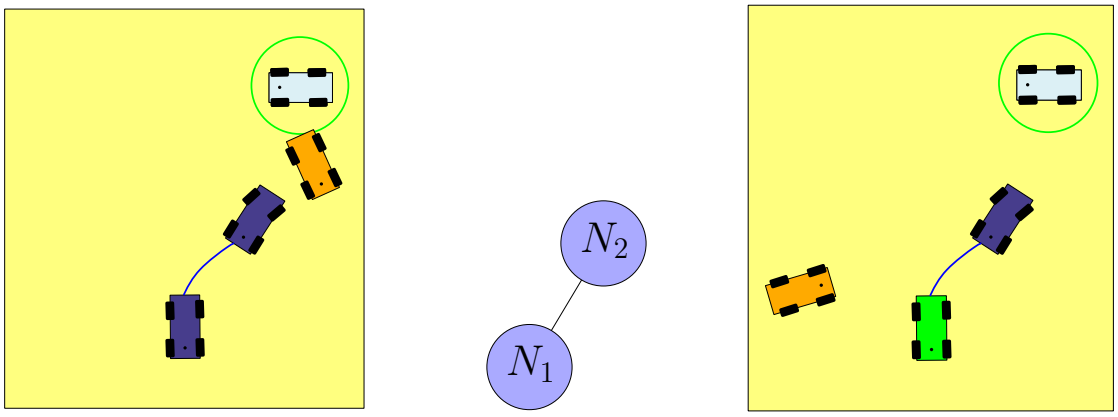
In step three, node N_1 is simulated with steering angles from a predetermined set. The one that brings it closer to N_{rand} is chosen and the pose this steering angle leads to after driving for a time period of Δt is used to form a new node, N_2, which is then added to the tree. The last two steps, step two and three, are repeated in step four and five, resulting in an additional node in the tree, node N_3.

Recall, that the pose of the random node was to be chosen as the goal pose with a probability of P_{greed}. This occurs in step six, and this time the nearest node is N_2. The goal region is reached in the following step when the tree branches out from this



(A) Step 1 (B) Step 2

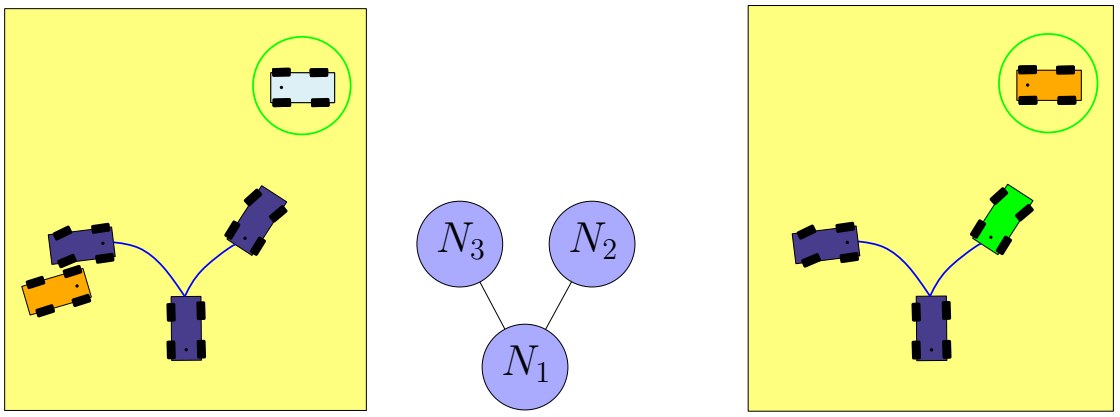
FIGURE 6.7: The first two steps in the execution example of the RRT algorithm.



(A) Step 3 (B) Step 4

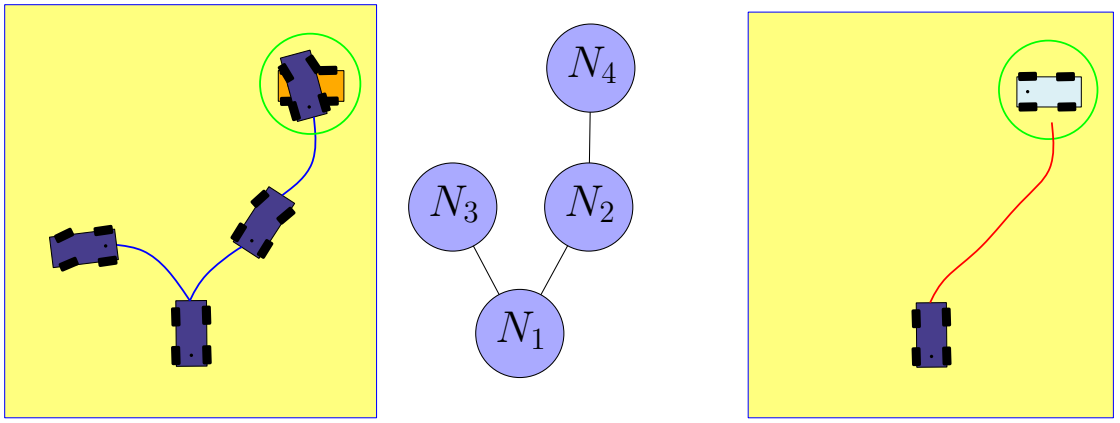
FIGURE 6.8: Step three and four in the execution example of the RRT algorithm.

node and the tree is then completed by adding the new node, N_4, as a child of N_2. By tracking back from the last node added in the tree to the root node, the path from the start pose to the goal region can be extracted from the tree, which is shown in the last step. Notice that for this to be possible each node needs to know its parent node.



(A) Step 5 (B) Step 6

FIGURE 6.9: Step five and six in the execution example of the RRT algorithm.



(A) Step 7 (B) Step 8

FIGURE 6.10: The last two steps in the execution example of the RRT algorithm.

6.3 Modified RRT

6.3.1 Modifications

The modifications made to the RRT planner will be covered in this section. The incorporation of the Dubins path is responsible for most of these and results in a myriad of improvements.

Distance metric

The determination of the nearest node in the modified algorithm is made by measuring the length of the Dubins path of each node in the tree to the random node and selecting the one with the shortest distance. This gives a significantly better evaluation than the one in the standard RRT planner since the tree in the modified RRT planner will in fact grow using the Dubins path, which means the distance metric reflects the travel distance exactly.

Checking growing ability of nodes

As previously stated, the process of obtaining the new node must include some sort of collision check. To keep a node that will not be able to grow from being selected as the nearest node, this check should be made before registering the measured path lengths of the nodes. A node will typically not grow the entire way to the random node since it only grows a distance d_g on the computed Dubins path, which corresponds to the distance the vehicle travels in Δt. Thus, the check is only made on this part of the Dubins path, which will be referred to as the *growth part*. If it turns out that a node will not be able to grow, its path length to the random node is set to infinity. This way, it will not be selected when the nearest node is being determined.

The collision check is made by simulating the movement trajectory of the tractor-trailer while following the growth part of the Dubins path and checking for collisions in the process. Performing the collision check by ways of simulating the tractor-trailer's movement for every node is computationally expensive, which becomes very apparent as the number of nodes increases. Yet, this collision check results in a lower computational load than the one in the standard RRT planner as it only simulates one movement per node, while the latter simulates several movements with different steering inputs. Nonetheless, it is still computationally expensive and it is therefore a good idea to use some heuristics to decrease the frequency with which it is applied. One way of doing this is by checking that the growth part does not intersect or even come close to the borders of the workspace or any obstacles. Surely, there would be no need to simulate the movement of the tractor-trailer if we already know that the growth part would lead to a collision.

Not allowing the growth part to even come close to anything it might collide with is of special importance in the case of reversing a tractor-trailer. The reason being that, although the trailer's wheel axle might always be on the path (with an ideal motion controller), one cannot expect this from the rest of the tractor-trailer, especially in sharp

turns where it has a tendency to flare out. This is shown in Figure 6.11 and it is obvious that the tractor-trailer would have collided with the wall if the path was allowed to get closer to it. A simple way to prevent this from happening is by placing a safety margin around the workspace borders and obstacles. It can be noted that setting the safety margin large enough, the collision check on tractor-trailer's movement trajectory could be skipped since it would not even be possible for the path to get close enough to an obstacle or border to cause a collision. However, this would require a fairly large safety margin, which will generally increase the difficulty of the planning task. For example, it can make a passage between two obstacles extremely narrow or close it all together and force the tree to find another way. Hence, it is better to perform the collision check for a node by combining the check on the growth part, using a safety margin of a reasonable size, and the check on the movement trajectory of the tractor-trailer.

It is wise to use an additional safety margin when checking the collision in the movement trajectory. This is because the simulated movement trajectory will most likely not be identical to the real one. Dodging an obstacle with no margin in the simulation could very well lead to a collision in reality. The size of this margin depends on how good the simulation is but it will generally be smaller than the one used in the collision check for the growth part.

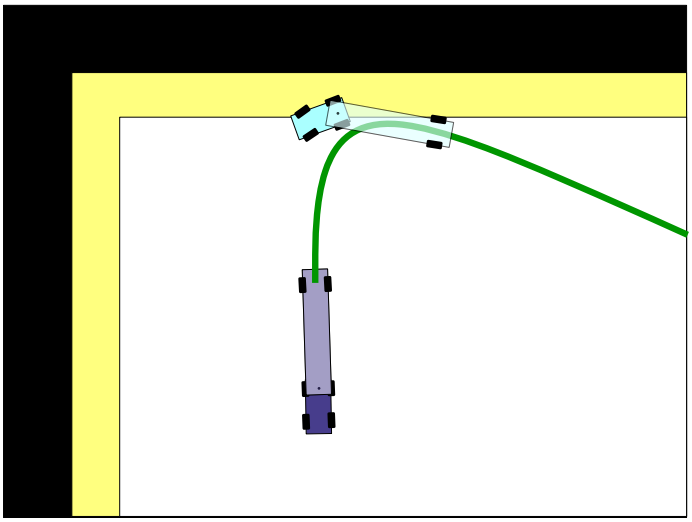


FIGURE 6.11: The tractor-trailer cannot be expected to be on the path at all times, even if follows it perfectly. At sharp curves in particular, a big part of the tractor-trailer will be outside of the path while following it. Hence, it is important to prevent the path from getting too close to obstacles and workspace boundaries, which can be achieved by placing a safety margin around them.

Growing nearest node

The next modification is found at the time of growing the nearest node. As mentioned above, the nearest node will grow a distance d_g on the computed Dubins path. This gives a more accurate movement towards the random node than the previous approach, where a set of discrete steering angles were used to figure out how the vehicle should move to get closer. An important thing to notice here is that the growth direction of the nearest node towards the random node is no longer discrete but continuous. Hence, a parent node that gets selected as the nearest node can grow and form a child node that is very close to a previous child node. This defeats the purpose of using the RRT since the aim is to explore the workspace as efficiently as possible. It can also cause a major problem in a situation where the parent node gets repeatedly selected as the nearest node, because the children are unable to grow, and the random node has a high probability of lying in the same direction. The tree will in a case like this keep growing nodes that lie very close to each other, see Figure 6.12. A condition should therefore be placed on the nodes to prevent them from growing child nodes within a distance from pre-existing ones.

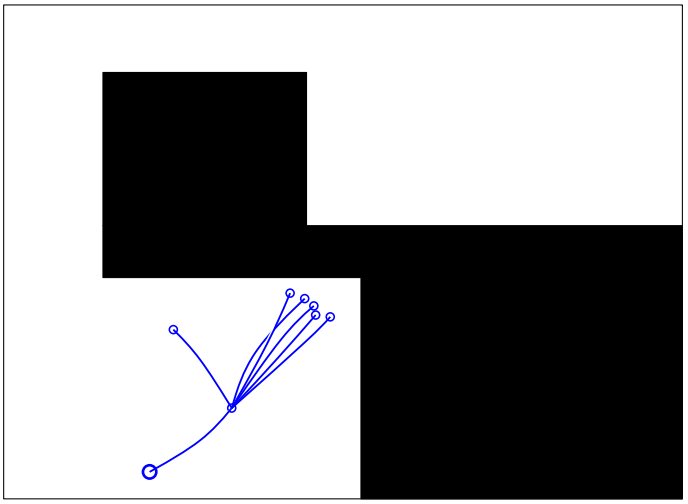


FIGURE 6.12: Since the direction of growth for a node in the modified RRT is continuous, it is possible for a parent node to produce child nodes that lie very close to each other. This can become a major problem if the parent node gets repeatedly selected as the nearest node and it is therefore necessary to place a condition that will not allow the nodes to grow child nodes within a distance from pre-existing ones.

Greediness factor

The utilization of the greediness factor in the modified RRT planner was shown to be ineffective. It is simply not needed because the main task of the tree here is to explore and the growing towards the goal is left to the Dubins path. Also, since many of the tasks solved by this planner require the tree to initially grow towards other directions before being able to head towards the goal pose, the use of the greediness factor hurts the planning more than it helps.

Planning time

It makes more sense to limit the planning by time rather than by number of nodes in the tree. The tree in the modified RRT algorithm will therefore grow with no restrictions in the number of nodes, and be limited by a maximum time $t_{\max}$ instead.

Parking at goal pose

Before the tree starts growing, a check is made to see whether the tractor-trailer can connect to the goal pose using the Dubins path. If not, it will grow a child node and do the same check for it. This process is repeated until the tree grows a node that has the ability to connect to the goal pose or the time runs out. Using the Dubins path here results in two important advantages over the RRT. Firstly, since the tractor-trailer now has the ability to reach a goal pose, and not merely a goal region, it can now complete parking tasks. Secondly, the planning can be completed much faster since the Dubins path takes almost no time to complete and the RRT, which does take some time, is only used to clear the way for the completion of the Dubins path.

The desired goal pose for the tractor-trailer in the parking problem is typically that of a straight one. However, the Dubins path will often times end in a curve, which is no problem for a vehicle with a rigid body like the Dubins car but a big one for an articulated vehicle like the tractor-trailer since it will end up in a curved pose. To make sure the tractor-trailer always ends with a straight pose, the Dubins path has to be modified a little so it will end with a straight line to give the tractor-trailer a chance for straightening. An easy way to achieve this is by drawing a straight path to the goal pose, the former with the same orientation as the latter. The Dubins path is then simply connected to this straight path. Figure 6.13 illustrates this idea. The length of the straight path depends on the minimum turning radius used in the Dubins path. A Dubins path that ends with a sharper curve means the tractor-trailer will have a larger curve at the end of it and, thus, require a longer path for straightening. Nonetheless,

this path will be about the same length of the tractor-trailer itself and will therefore not pose any problems or difficulties in the planning. Quite the contrary, it will facilitate the connection of the Dubins path if the goal pose is in a parking space that is surrounded by obstacles. That is, the straight path makes the goal pose more visible and thereby easier for the tree to connect to.

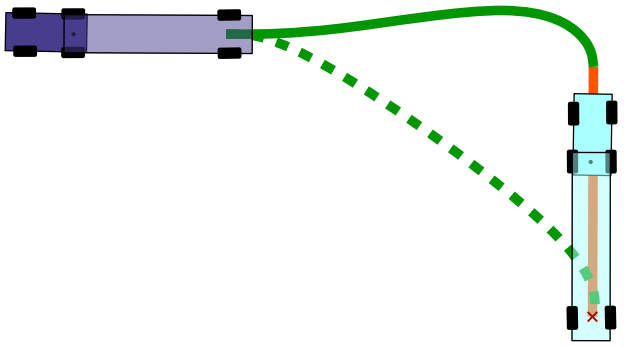


FIGURE 6.13: The Dubins path will often times end in a curve which will cause the tractor-trailer to end with a curved pose. Since the desired goal pose is typically that of a straight tractor-trailer, a straight path (shown in orange) has to be added to give the tractor-trailer a chance to straighten out.

6.3.2 Path optimization

The RRT planner by itself yields a highly suboptimal path. The more the nodes involved in the path, the higher the suboptimality usually is. This is because many of the nodes in the path do not contribute with anything but unnecessary turns, and they typically increase with the number of nodes in the path. Having many unnecessary turns not only makes the driving distance longer, it makes the task of the motion controller unnecessarily difficult and results in an awkward motion of the vehicle. There is a number of ways to optimize the path, although many of them tend to have high computational costs.

The use of the Dubins path proves to be useful yet again as it is used to achieve a path optimization that is both of good quality and computationally cheap. The algorithm of this path optimization is conceptually simple, see algorithm 2. Forming a stack with the nodes that make up the suboptimal path, the algorithm will try to connect one node n_{current} (initially set as the root node) to another node n_{target} (initially set as the second to last node) using the Dubins path. If this path is feasible (which is checked in the same way as the growth part), it will eliminate all the intermediate nodes in the stack, since they are obviously not needed, and form a new stack. If it cannot, it will

switch the target node n_{target} one step down in the stack and repeat the same procedure. This continues until it finds a target node that it can connect to or arrives to the node after n_{current}, as there will be no more intermediate nodes to eliminate. The algorithm then switches node n_{current} one step up in the new stack (or the same stack if no nodes have been removed) and repeats the procedure. Once the algorithm has switched node n_{current} all the way up to the second to last node, the optimized path will be formed by connecting the nodes that have survived in the stack using the Dubins path. The concept of this optimization is depicted in Figure 6.14.

The reason the algorithm tries to connect each node in the stack to the second to last node rather than the last node is because it knows that it is not possible, since this is what the modified RRT planner already tried to do in every step. That is, the second to last node cannot be unnecessary and, thus, there is no point in trying to check if it can be eliminated.

```
stack := (n_1, n_2, ..., n_end);
n_current = n_1;
n_target = n_end−1;
while n_current ≠ n_end−1 do
    while index(n_current) + 1 < index(n_target) do
        connectionPath ← connectNodes(n_current, n_target);
        if connectionPath is feasible then
            stack ← removeIntermediateNodes(stack, n_current, n_target);
            break;
        else
            n_target = n_target−1;
        end
    end
    n_current = n_current+1;
    n_target = n_end−1;
end
optimizedPath ← connectNodes(stack);
```

Algorithm 2: Optimization algorithm

6.3.3 Implementation results

The implementation results of the modified RRT planner on the docking task are shown here on a 4x4 m workspace. The minimum turning radius was set to 0.5 m since this was the smallest turning radius with which the motion controller performed well according to the simulation results. While the simulation of the tractor-trailer's movement trajectory

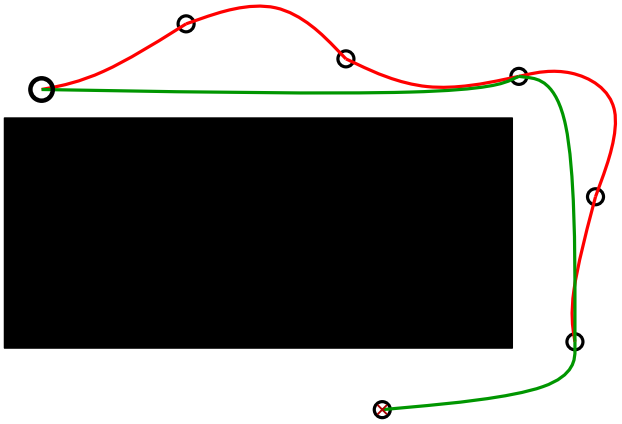


FIGURE 6.14: The path optimization performed on a suboptimal path (red). All the unnecessary nodes are eliminated, which results in an optimized path (green) that is both shorter and smoother.

in the collision check was done with a control rate of 10 Hz, the actual collision check was made with a rate of 2 Hz, which gives an adequate coverage of the trajectory as seen in Figure 6.15. The small gaps between the tractor-trailers in the trajectory do not really pose a problem since a safety margin of 0.02 m was used in the collision check of the trajectory. The figure also shows that at least 0.14 m would be required as a safety margin for the collision check on the growth part to make sure the path keeps enough distance from the workspace borders and the obstacles to render the collision check made on the movement trajectory unnecessary. However, it would be highly unpractical to use such a large safety margin. Hence, a more reasonable size for the safety margin was used in the implementation, namely 0.06 m which corresponds to approximately two meters for the full scaled tractor-trailer.

Moreover, the size of the safety margin affects the feasible parking space width since a large margin could cover the entire parking space and prevent the path from reaching the goal pose. With the safety margin in use, a parking space width as small as 0.15 m could be used (leaving 0.03 m for the path to enter), see Figure 6.16. However, this sort of defeats the purpose of the safety margin and places great stress on the motion controller. Hence, a parking space width of 0.20 m is more reasonable, which was used in the docking tasks shown in Figure 6.17 and 6.18.

Notice the significant improvements obtained with the path optimization in the planning results. Furthermore, although the existence of obstacles in an environment is often linked to an increased difficulty in the planning, if they are not in the way of finding the parking space, they will actually facilitate the planning. The reason for this is that

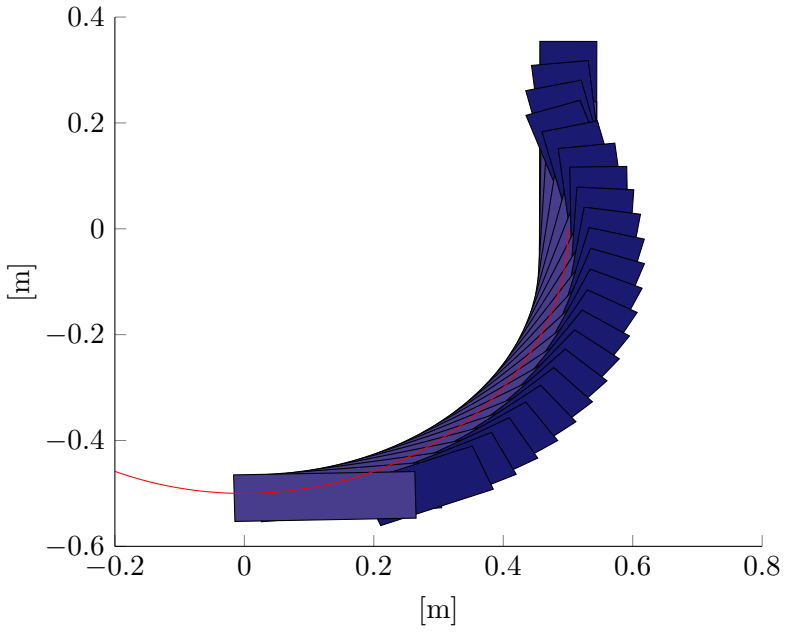


FIGURE 6.15: Trajectory of the tractor-trailer sampled with 2 Hz while following a curve with a 0.5 m turning radius. The maximum distance the trajectory reaches from the path is 0.14 m which occurs at the beginning as the tractor-trailer flares out since it starts with a straight pose.

with less space to explore, the tree will have a greater likelihood of growing in the right direction. This can be observed when comparing Figure 6.17 and 6.18.

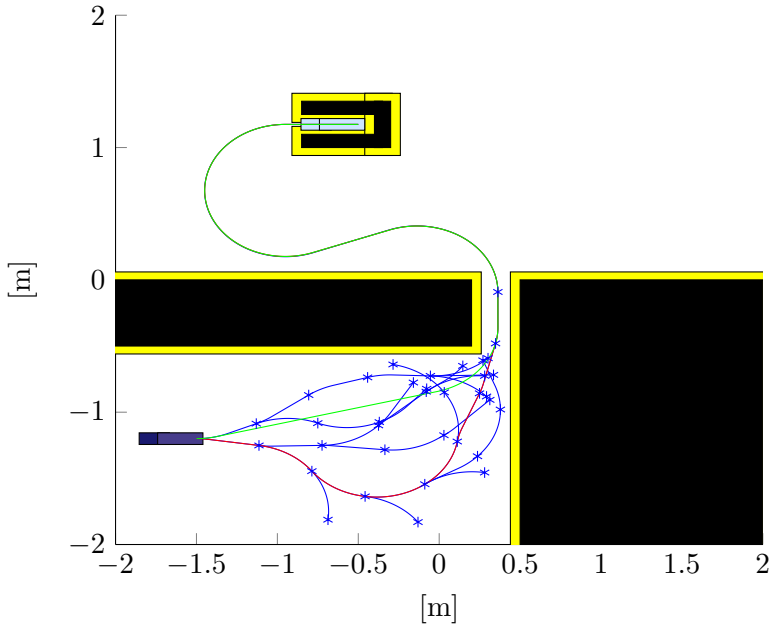


FIGURE 6.16: Docking task 1 - planning results of the modified RRT planner. The path found by the tree is shown in red and the one given by the path optimization is shown in green.

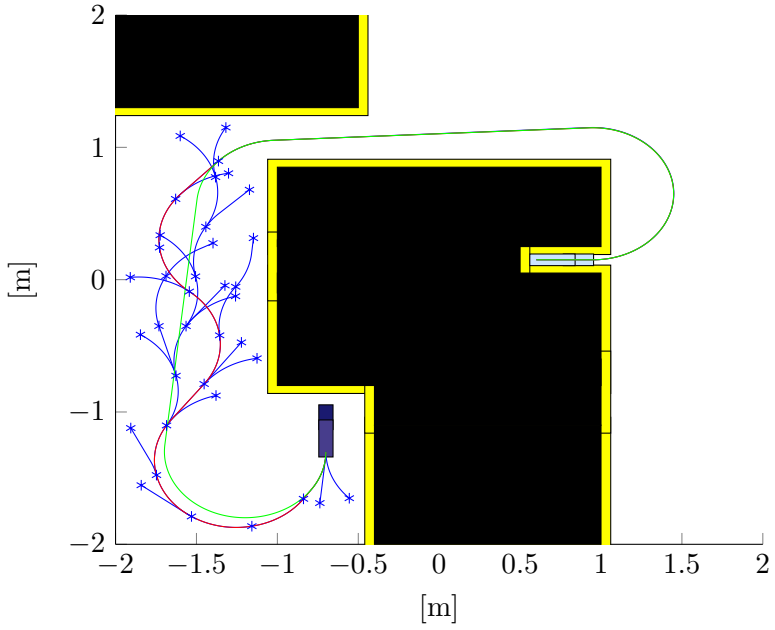


FIGURE 6.17: Docking task 2 - planning results of the modified RRT planner. The path found by the tree is shown in red and the one given by the path optimization is shown in green.

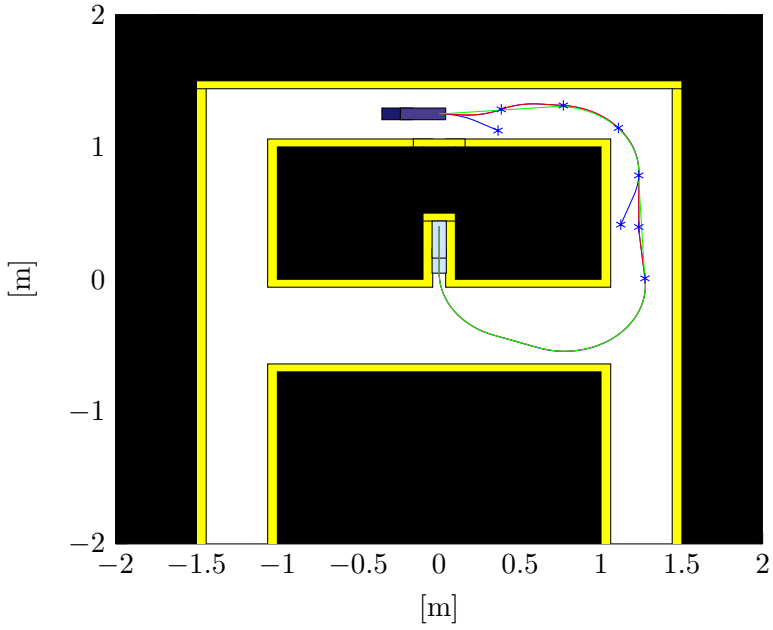


FIGURE 6.18: Docking task 3 - planning results of the modified RRT planner. The path found by the tree is shown in red and the one given by the path optimization is shown in green.

Chapter 7

Implementation

The difference between the autonomous driving system's performance in the simulation and in reality boils down to the implementation of the motion controller. This is because, given the knowledge of the workspace, the obstacles therein, and the start and goal poses, the path planner will in principle operate with no interaction with the workspace environment. The issues pertaining to the implementation of the motion controller will be discussed before presenting the results of the implemented autonomous driving system.

7.1 Implementing the motion controller

Although the velocity and steering actuators' input-output relations are linear to a certain degree, they are not perfectly linear and are affected by a number of factors such as the inertia. Hence, one cannot make a perfect mapping between the outputs and inputs of the actuators and in such a way be able to set up a function that gives the desired output. The motion controller therefore skips from consisting of two controllers, the path stabilization controller and the hitch angle controller, to four controllers with the additional velocity and steering controllers. Before testing the motion controller, all internal controllers should be tuned to a satisfactory level. The internal controllers are those of the velocity, steering, and hitch angle, which are all in essence PI controllers. Since the hitch angle controller will depend on the steering controller which in turn will depend on the velocity controller, these controllers must be developed in the aforementioned order.

7.1.1 Velocity controller

Recalling the input-output relation of the velocity actuator, see Figure 4.5, it is clear that the PI controller that makes up the velocity control must have a bias term since the stoppage velocity V_{stop} is nonzero. Knowing that we are operating on negative velocities, the PI controller takes the following form:

$$V_{\text{velocity}} = V_{\text{stop}} + K_P(v_{\text{ref}} - v) + K_I \int_0^t (v_{\text{ref}} - v)dt \tag{7.1}$$

where V_{velocity} is the velocity voltage and v_{ref} is the reference velocity. To evaluate the velocity controller, a reference velocity in the shape of a sine signal, with an amplitude of 10 m/s and a period of 8 s, is used. The results can be seen in Figure 7.1. The actuator delay is visible, as expected, but other than that, the performance of the controller is very good. Nevertheless, since we will be driving with a constant velocity one can actually run a few experiments to see what voltage will give the desired velocity, which in our case is as low of a velocity as possible without having a jerky movement due to friction. Going with this approach instead is found to give better results.

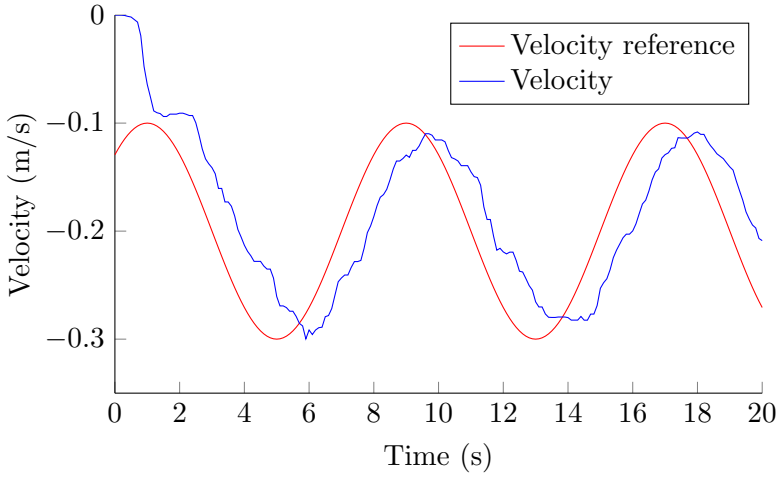


FIGURE 7.1: Evaluating the velocity controller with a sine reference.

7.1.2 Steering controller

Just like in the velocity controller and for the same reason, see Figure 4.6, the PI controller that makes up the steering controller has a bias term, namely V_{straight}. Notice that the relationship between the steering voltage and the steering angle is negative,

meaning an increase in the steering angle requires a decrease in the steering voltage. The PI controller therefore takes the following form:

$$V_{\text{steering}} = V_{\text{straight}} - \left(K_P(\phi_{\text{ref}} - \phi) + K_I \int_0^t (\phi_{\text{ref}} - \phi)dt\right) \tag{7.2}$$

where V_{steering} is the steering voltage and ϕ_{ref} is the reference steering angle. Using a similar sine signal as in the evaluation of the velocity controller gives the results shown in Figure 7.2. The oscillation in the beginning is caused by the fact that the velocity is used in the estimation of the steering angle (recall from (4.2)) and since the velocity is near zero in the beginning, it will give a bad estimation which will result in a bad control performance. Nonetheless, the steering controller as a whole does perform well, with the delay in the steering actuator being visibly smaller than the one found in the velocity controller.

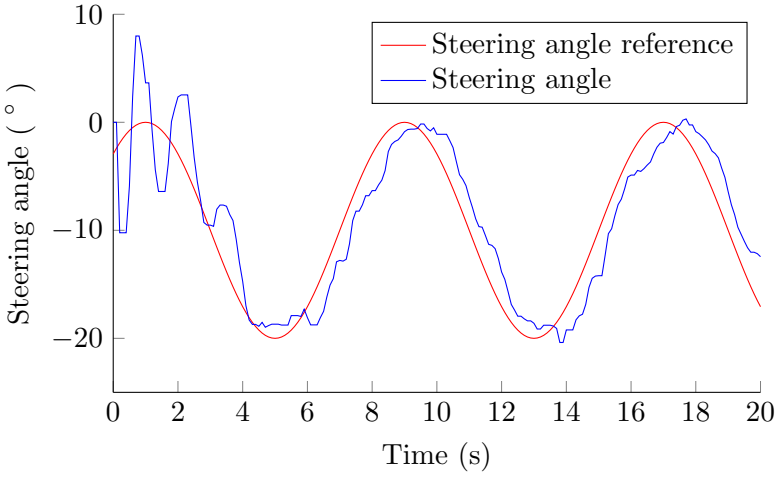


FIGURE 7.2: Evaluating the steering controller with a sine reference.

7.1.3 Hitch angle controller

The hitch angle controller implemented here is the one discussed in subsection 5.2.1. However, there is an important difference, namely that the output is not applied immediately, rather, it is passed to the steering controller which is responsible for its application. This means that any imperfection in the steering controller becomes amplified and visible in the hitch angle controller's performance. Thus, it can be a good idea to make sure the steering controller is working as smoothly as possible. Increasing the filtering of the steering estimation can be helpful in doing this, although it comes

with the price of an increased delay in the controller. Figure 7.3 shows the performance of the hitch angle controller, which is deemed to be sufficient for our purpose of path following.

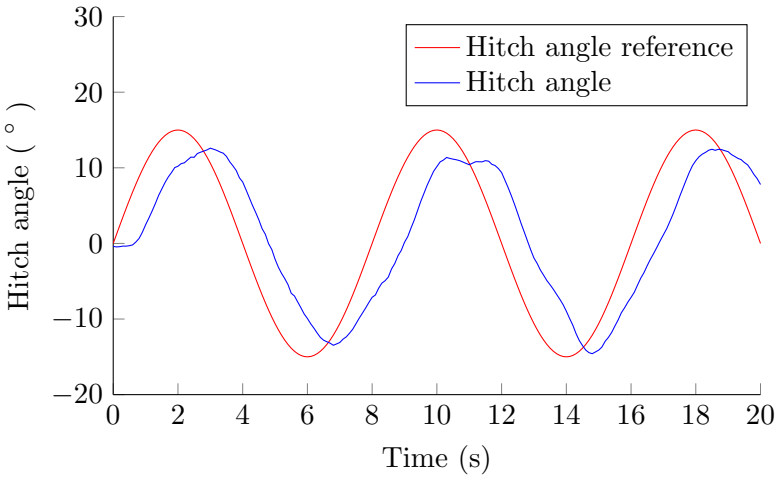


FIGURE 7.3: Evaluating the hitch angle controller with a sine reference.

7.2 Implementation results

7.2.1 Path following results

A few path following experiments should be conducted with the motion controller in order to obtain the optimal minimum turning radius for the path planner. The goal is to find the lowest minimum turning radius possible that the motion controller is able to handle well. The simulation results indicated that the motion controller would not perform well on the minimum turning radius 0.35 m. To validate this, a clockwise circle with this radius was created. Since it is not easy to place the tractor-trailer at the exact initial point of the path, like in the simulations, the order is flipped. That is, the current pose of the tractor-trailer is first obtained and using it, the path is created starting from the midpoint of the trailer's wheel axle. The path following performance on this path can be seen in Figure 7.4, which is, as expected, not a good performance.

Increasing the radius of the circle to 0.5 m gives an outstanding performance, see Figure 7.5, just as predicted in the simulation work. A figure-eight path with the aforementioned radius was created to evaluate the ability of handling transitions. Also this path was handled with a great path following performance, as seen in Figure 7.6, proving the simulation to be pessimistic in this case. The performances of the steering and hitch

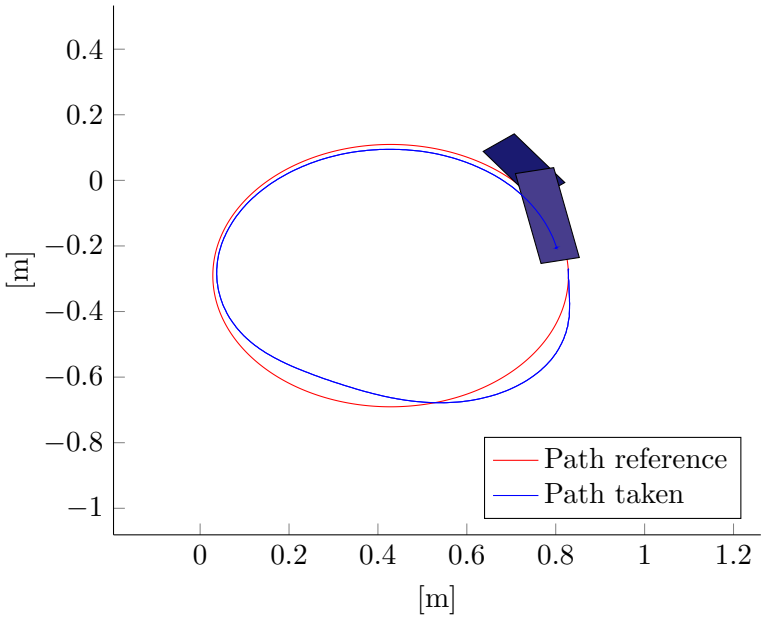


FIGURE 7.4: Path following performance on a clockwise circular path with a 0.35 m radius.

angle controllers throughout this path following task are shown in Figure 7.7 and 7.8, which in reality give a better evaluation than the ones obtained with the sine signal references since these reference signals cover a whole spectrum of frequencies. The steering controller exhibits an oscillatory behavior, especially during the first curvature of the path (the oscillation seen at the end is merely due to the braking of the tractor-trailer, i.e. the fact that the velocity that is needed to estimate the steering angle is brought down to zero). The main cause of this oscillation is most likely the delay in the steering actuator. However, its amplitude is fairly small and the impact on the hitch angle is barely noticeable, which is what really matters from a movement trajectory perspective. Hence, we conclude that the motion controller handles this path well and that the minimum turning radius in the path planner should be set to 0.5 m.

7.2.2 Docking results

In order to test the complete autonomous driving system, docking tasks were created using the five obstacles available in the testbed. Three of these were used to make up a parking space and the remaining two to increase the difficulty of the planner by placing them on the way of what would seem to be the most plausible path from the start pose to the goal pose.

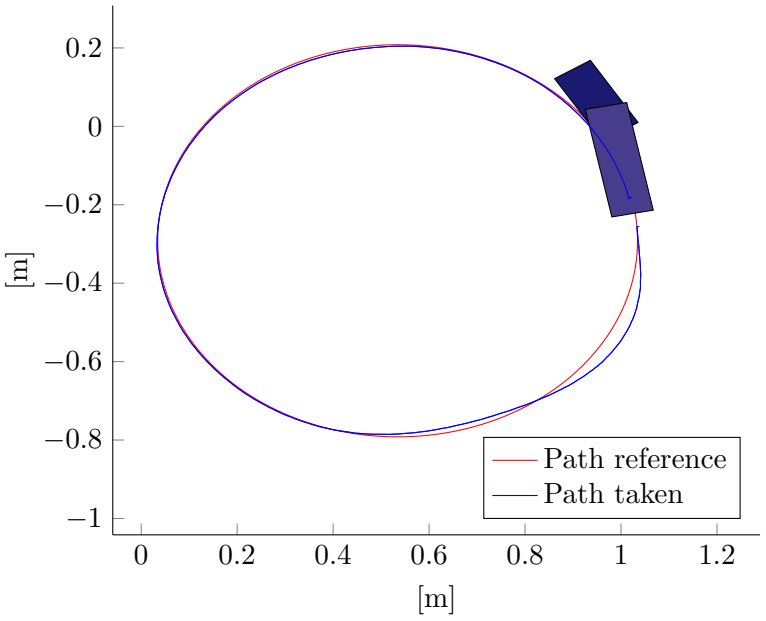


FIGURE 7.5: Path following performance on a clockwise circular path with a 0.5 m radius.

One of these docking tasks is shown in Figure 7.9 together with the planned tree and the optimized path from the modified RRT planner. Notice that the minimum turning radius of 0.5 m is too large for the path planner to immediately connect to the goal pose from the narrow passage. Therefore, the planner has to make some sort of loop after it drives through the passage.

The path following performance in this task can be seen in Figure 7.10 and the steering control action and hitch angle control can be seen in Figure 7.11 and 7.12, respectively. With such a great performance, one might get tempted to further decrease the minimum turning radius. However, although rarely occurring, there are instances where the motion controller will not give as great path following with a minimum turning radius of 0.5 m. This happens primarily when the path has a sharp curve with one direction (e.g. clockwise direction) which is immediately followed by another sharp curve with an opposite direction. The cause of this is the notorious slow angular velocity of the hitch angle, i.e. the fact that the process of changing the tractor-trailer pose, and consequently the steering direction of the tractor-trailer, is a slow one. Figure 7.13 shows the path following performance on such a path from the previously discussed docking task. From the figure, it is clear how the tractor-trailer loses the path temporarily at the transition. Therefore, the minimum turning radius should be kept at 0.5 m since any decrease would have in this case resulted in a worse performance or even instability. Nonetheless, it should be mentioned that this deviation from the path does not pose

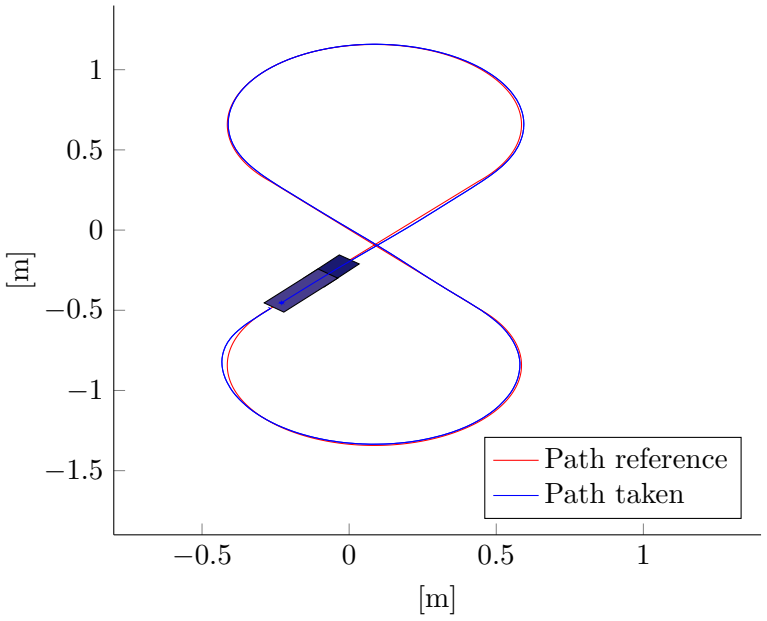


FIGURE 7.6: Path following performance on a figure-eight path with a 0.5 m radius in the curves.

any collision danger since the modified RRT planner checks in advance if the path will cause the tractor-trailer to collide. That is, by simulating the movement trajectory of the tractor-trailer, the planner predicts this deviation and checks that it does not cause a collision. How efficient this is depends on how well the simulation reflects the actual motion of the tractor-trailer. Nevertheless, using an additional safety margin in this collision check gives some leeway in terms of the simulation's accuracy.

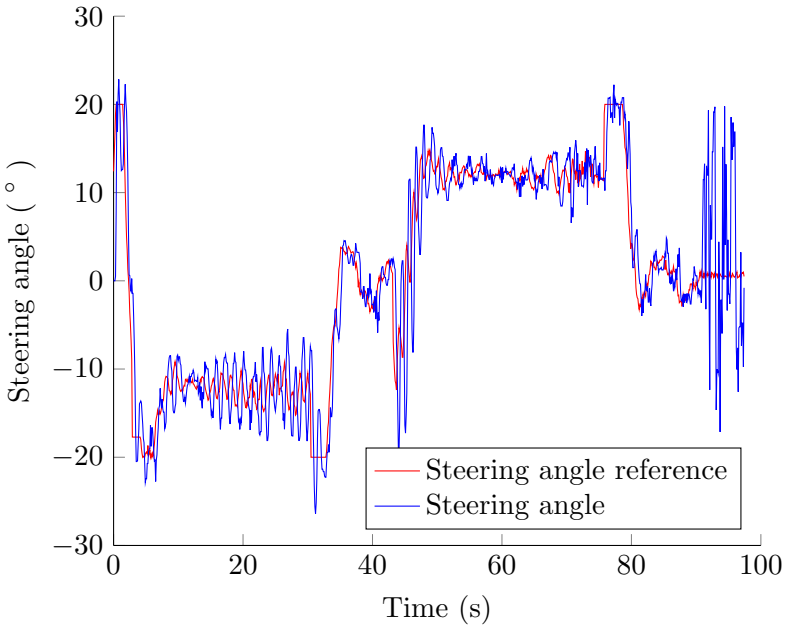


FIGURE 7.7: Performance of the steering controller during the path following of the figure-eight path.

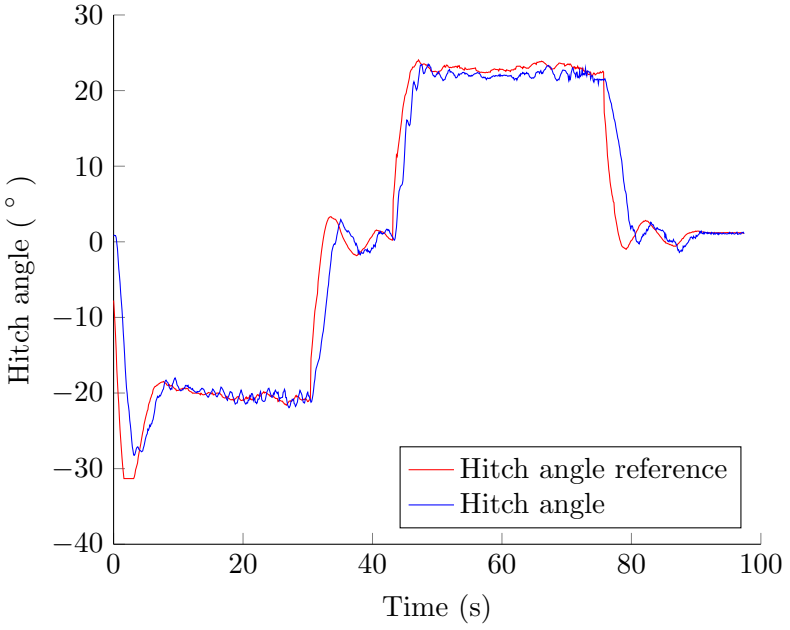


FIGURE 7.8: Performance of the hitch angle controller during the path following of the figure-eight path.

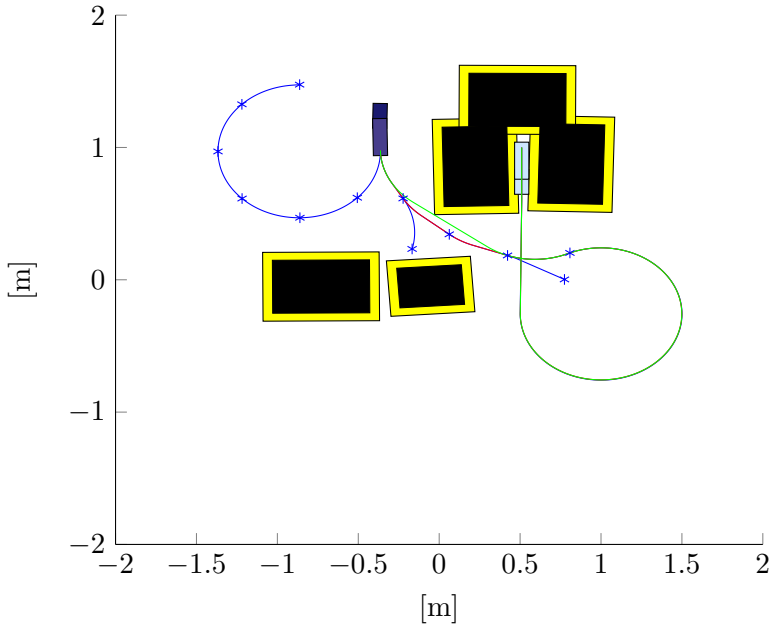


FIGURE 7.9: Docking task with tree and optimized path from the modified RRT planner.

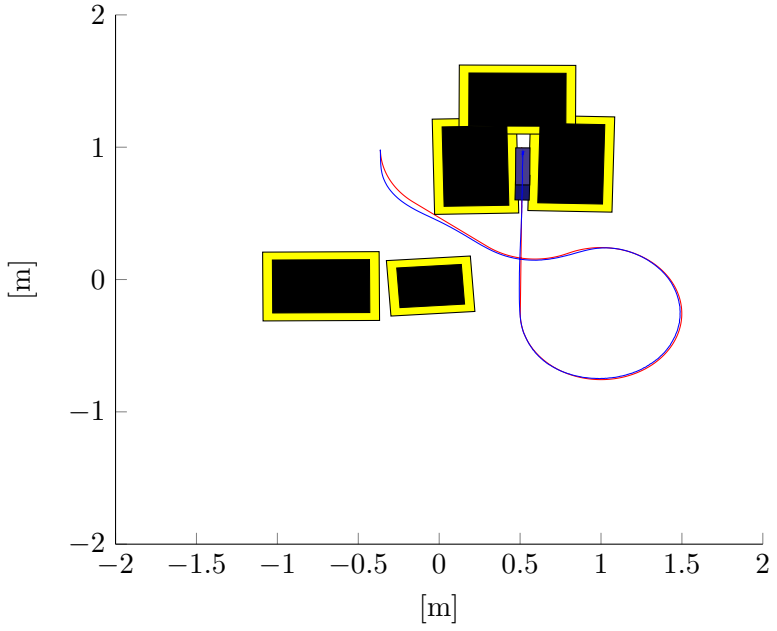


FIGURE 7.10: Path following performance on a planned path in the docking task.

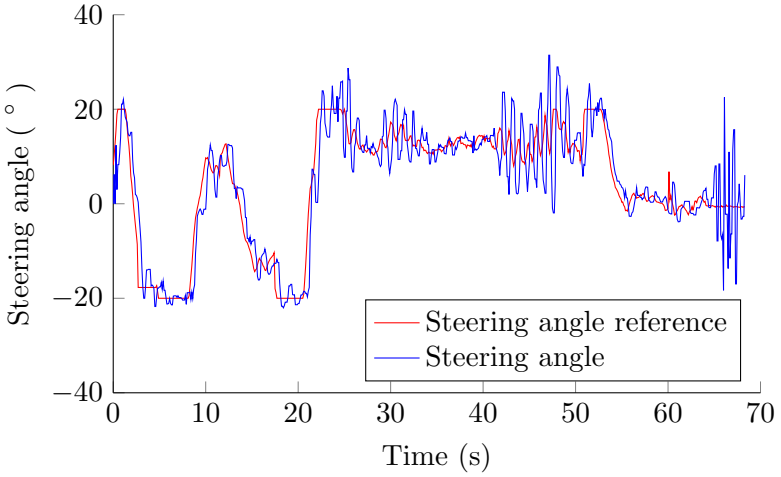


FIGURE 7.11: Performance of the steering controller during the docking task.

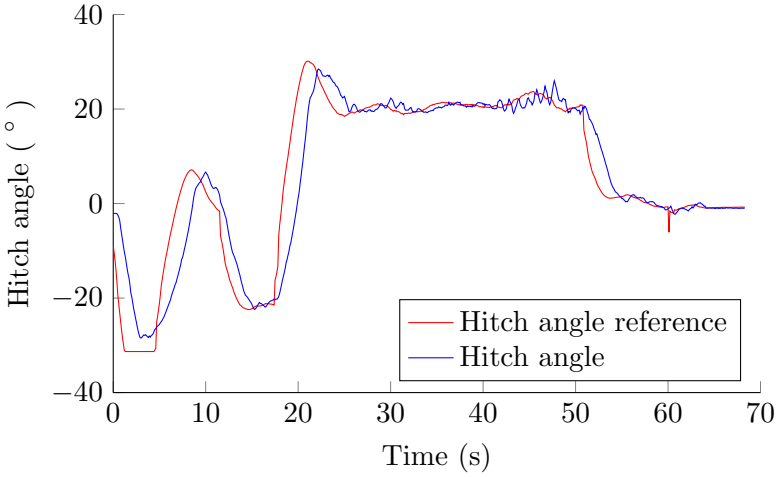


FIGURE 7.12: Performance of the hitch angle controller during the docking task

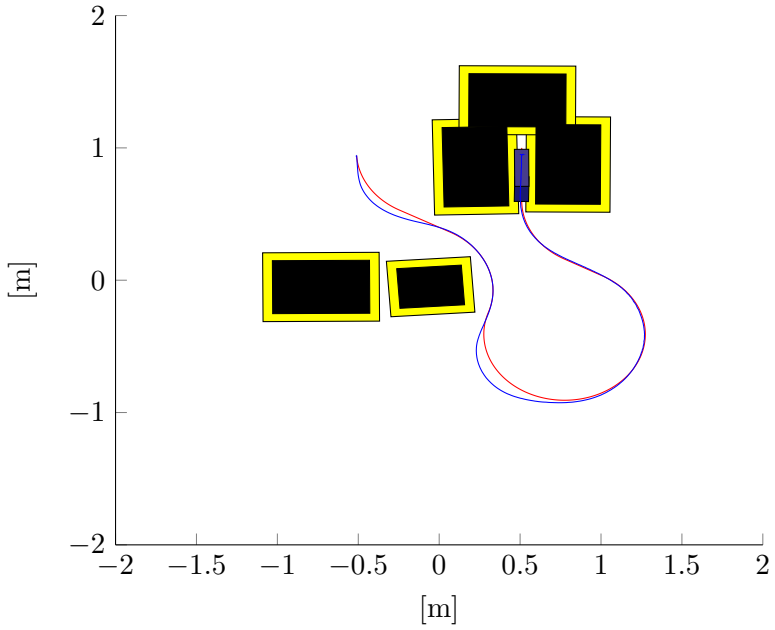


FIGURE 7.13: Path following performance on a planned path with two consecutive sharp curves in opposite directions.

Chapter 8

Discussion and conclusions

The implemented autonomous driving system proved to complete docking tasks successfully in an unstructured environment with static obstacles. The motion controller was shown to have the ability of following paths with minimum turning radiuses of 0.5 m. The performance of the motion controller is excellent, having difficulties only in the rare situation were a path has two consecutive sharp curves with opposite directions. The main cause of the diminished performance in this case is the fact that the angular velocity of the tractor-trailer's hitch angle is inherently low. That is, the tractor-trailer changes its pose quite slowly, which is crucial considering the pose is what determines the direction of the movement when reversing the tractor-trailer. Unfortunately, there is no practical way of increasing this angular velocity. The possible approaches either deal with modifying the tractor-trailer, e.g. increasing the steering angle, which is not easy to do, or increasing the velocity, which makes the control task much more difficult. Also, the velocity used was already relatively high, corresponding to a velocity of a full scaled tractor-trailer that is more than twice as high as the typical velocity with which a professional truck driver reverses a tractor-trailer.

Given the opportunity to construct the tractor-trailer, there is a number of things one can do to facilitate the motion control considerably. For example, one can set the dimensions of the tractor-trailer in a way that will remove the jackknife effect. However, many of these improvements have trade-offs, usually between maneuverability and controllability. The most important thing one can do construction-wise to the tractor-trailer seems to be giving it a large steering angle. This does not really have a trade-off and will improve the controllability immensely.

The effect of the steering actuator's delay also played a role in the performance of the motion controller. In a closed-loop feedback system, an actuator delay has the same effect as a low control frequency. Hence, the magnitude of this delay is of great interest,

especially since the steering has a direct influence on the tractor-trailer's pose. Again, this is something that one does not usually have any direct control over but is worth mentioning for the one who does.

Although the measurements obtained from the motion capture system had high precision, the fact that the steering angle was estimated, using estimations of the velocity and the angular velocity, did make the task of the steering controller harder, which directly affects the motion controller. Using a steering angle sensor, such as an encoder, to measure the steering angle would probably result in a remarkable difference in the steering controller.

It should be noted that the performance of the motion controller is dependent on several factors, such as the articulated vehicle type and the actuators in use. Thus, it is not easy to compare the performances of different motion controllers unless they have been implemented in the same platform.

As for the developed path planner, the modified RRT planner resulted in numerous improvements compared to the standard RRT planner. The most important of these is that the modified RRT planner does not merely find a path to a goal region but rather to a goal pose which enables parking tasks. This is due to the incorporation of the Dubins path, which is responsible for most of the modifications and improvements. Yet another major improvement its use brings about is a modified approach for the tree's growth that not only results in a better growth but also reduces the computational effort. Also, to deal with the suboptimality of the paths generated by the RRT planner, the modified planner utilizes a simple yet effective optimization algorithm. Nonetheless, just like the standard RRT planner, the modified RRT planner has the weakness of stochasticity, i.e. the element of randomness with which the paths are generated.

One way to work around the problem of the low angular velocity of the hitch angle is to use clothoids, i.e. curves with linear curvature changes, in the path planning. This provides the path with soft transitions from one path segment to another, which results in a lower angular velocity requirement on the tractor-trailer's hitch angle. However, this also means that one will lose the benefit of using the Dubins path, which although has a discontinuous curvature profile by nature, does give the shortest path between two poses. It would therefore be wise to combine the use of clothoids together with the Dubins path as done in [38].

The developed autonomous driving system would probably give even better results if implemented on a real full scale tractor-trailer. This is because many of the factors that have caused a reduction in the performance would diminish. For example, the aforementioned delay in the steering actuator would most likely be shortened since the

quality of actuators in a real tractor-trailer would be far better than the ones that were used. Also, the steering angle would not be estimated, rather it would be measured with a very high accuracy. This aside, while the collision check in the modified RRT planner would probably be sufficient for the use in a real tractor-trailer, one would have to treat this subject with a considerably greater care.

Potential future work for this autonomous driving system would primarily be to incorporate the forward motion into the system. This would be a huge step as it would enable treatment of more complex tasks, such as parallel parking. Developing a motion controller for the forward motion of an articulated vehicle is not a difficult task, especially in comparison to the backward motion since the system will turn into a stable one. The big challenge lies instead in the planning part. Reeds-Shepp curves [39] would probably be of great use for this purpose. Although it should be mentioned that the difficulty level of doing this with an articulated vehicle is much greater than with a rigid vehicle. The optimization of the modified RRT planner can still be improved further. Use of genetic algorithms has been shown to give good optimization results [40], they are however computationally expensive in general and one might have to come up with some heuristics to make their use more practical. This could however be worked around by using a very fast programming language, such as `C`. Another feature that can be added to increase the complexity of the tasks handled by the autonomous driving system is avoidance of dynamic obstacles, which would require a highly responsive planning algorithm. An approach that adapts the RRT planner for this sort of planning in dynamic environments is found in [41].

Furthermore, the field of autonomous driving systems is one that is in rapid motion with many researchers involved in it. Hence, one can expect to see a lot of new and interesting approaches within the near future.

Acknowledgements

I would like to thank my supervisor Jonas Mårtensson for granting me the opportunity of working on this project and for his sage advice during the course of its execution. My friend and co-supervisor Rui Oliveira for all his help and guidance, and for never failing to find the bugs in my code. You really went the extra mile with me. The guys at the Smart Mobility Lab for all the laughter and conversations, specially the ones that turned into debates. My family for being patient with my frequent absence. Your support has been invaluable.

Bibliography

[1] Gillian Yeomans. Autonomous vehicles, handing over control: Opportunities and risks for insurance. Technical report, Loyd's Exposure Management, 2014.

[2] In RandolphW. Hall, editor, *Handbook of Transportation Science*, volume 56 of *International Series in Operations Research & Management Science*. 2003. ISBN 978-1-4020-7246-8.

[3] A. González-Cantos and A. Ollero. Backing-up maneuvers of autonomous tractor-trailer vehicles using the qualitative theory of nonlinear dynamical systems. 28(1): 49–65, 2009.

[4] Jae Il Roh and Woojin Chung. Reversing control of a car with a trailer using the driver assistance system. *International Journal of Advanced Robotic Systems*, 2011.

[5] J. Morales, A. Mandow, J.L. Martinez, and A.J. Garcia-Cerezo. Driver assistance system for backward maneuvers in passive multi-trailer vehicles. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4853–4858, Oct 2012.

[6] Thaker Nayl. *Modeling, control and path planning for an articulated vehicle.* PhD thesis, 2013.

[7] Yang Bin and Taehyun Shim. Constrained model predictive control for backing-up tractor-trailer system. In *Intelligent Control and Automation (WCICA), 2012 10th World Congress on*, pages 2165–2170, July 2012.

[8] D. Nguyen and B. Widrow. The truck backer-upper: an example of self-learning in neural networks. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 357–363 vol.2, 1989.

[9] M. Schoenauer and E. Ronald. Neuro-genetic truck backer-upper controller. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 720–723 vol.2, Jun 1994.

[10] Jin Cheng, Yong Zhang, and Zhonghua Wang. Curve path tracking control for tractor-trailer mobile robot. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, volume 1, pages 502–506, July 2011.

[11] Cédric Pradalier and Kane Usher. Robust trajectory tracking for a reversing tractor trailer. *Journal of Field Robotics*, 25(6-7). ISSN 1556-4967.

[12] Jiliang Jiang, Dawei Tu, Shuo Xu, and Qijie Zhao. Cognitive response navigation algorithm for mobile robots using biological antennas. *Robotica*, 32:743–756, 8 2014. ISSN 1469-8668.

[13] Vladimir J. Lumelsky and A.A. Stepanov. Dynamic path planning for a mobile automaton with limited information on the environment. *Automatic Control, IEEE Transactions on*, 31(11):1058–1063, Nov 1986.

[14] H. Noborio, K. Fujimura, and Y. Horiuchi. A comparative study of sensor-based path-planning algorithms in an unknown maze. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 2, pages 909–916 vol.2, 2000.

[15] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[16] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin. Path planning for mobile robot navigation using voronoi diagram and fast marching. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2376–2381, Oct 2006.

[17] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. ISBN 9780511546877. Cambridge Books Online.

[18] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, July 1968.

[19] Anthony Stentz. Optimal and efficient path planning for unknown and dynamic environments. Technical report, DTIC Document, 1993.

[20] Anthony Stentz. The focussed d* algorithm for real-time replanning. In *In Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652–1659, 1995.

[21] Sven Koenig and Maxim Likhachev. D* lite. In *AAAI/IAAI*, pages 476–483, 2002.

[22] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[23] Jean-Daniel Boissonnat, André Cérézo, and Juliette Leblond. Shortest paths of bounded curvature in the plane. *Journal of Intelligent and Robotic Systems*, 11 (1-2):5–20, 1994. ISSN 0921-0296.

[24] Xuan-Nam Bui, Jean-Daniel Boissonnat, P. Soueres, and J.-P. Laumond. Shortest path synthesis for dubins non-holonomic robot. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 2–7 vol.1, May 1994.

[25] John Stergiopoulos and Stamatis Manesis. Anti-jackknife state feedback control law for nonholonomic vehicles with trailer sliding mechanism. *International Journal of Systems, Control and Communications*, 1:297–311, 2009.

[26] Johan Nilsson and Shant Abraham. Trailer parking assist. Master's thesis, Chalmers University of Technology, Division of Automatic Control, Automation and Mechatronics, 2013.

[27] Qualisys Motion Capture System. Robust High-performing Real-time. [Online], Accessed: 8 June 2015. Available at: http://www.qualisys.com/products.

[28] Siku Control32. Scania with tipping trailer remote control and charger. [Online], Accessed: 8 June 2015. Available at: http://www.siku.de/en/sortiment/sikucontrol32/lkw-traktoren/scania-zugmaschine-mit-kippsattelauflieger.html.

[29] Hans Pacejka. *Tire and vehicle dynamics*. Elsevier, 2005.

[30] Peter Ridley and Peter Corke. Load haul dump vehicle kinematics and control. *Journal of dynamic systems, measurement, and control*, 125(1):54–59, 2003.

[31] Paul Bourke. Equation of a circle from 3 points (2 dimensions). [Online], Accessed: 8 June 2015. Available at: http://paulbourke.net/geometry/circlesphere.

[32] Z Shafiei and AT Shenton. Tuning of pid-type controllers for stable and unstable systems with time delay. *Automatica*, 30(10):1609–1615, 1994.

[33] Hassan K Khalil. Nonlinear systems, 3rd. *New Jewsey, Prentice Hall*, 9, 2002.

[34] Marcio S. de Queiroz, Darren M. Dawson, Siddharth P. Nagarkatti, and Fumin Zhang. *Lyapunov-based control of mechanical systems*. Control engineering. Birkhäuser, Boston, 2000. ISBN 0-8176-4086-X.

[35] David A Anisi. Optimal motion control of a ground vehicle. *Swedish Defense Research Agency, Tech. Rep*, 2003.

[36] Scott Teuscher. Dubins curve mex. [Online], Accessed: 8 June 2015. Available at: http://www.mathworks.com/matlabcentral/fileexchange/40655-dubins-curve-mex.

[37] Peng Cheng and S.M. LaValle. Reducing metric sensitivity in randomized trajectory design. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 43–48 vol.1, 2001.

[38] A. Scheuer and Th. Fraichard. Continuous-curvature path planning for car-like vehicles. In *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, volume 2, pages 997–1003 vol.2, Sep 1997.

[39] James Reeds and Lawrence Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific journal of mathematics*, 145(2):367–393, 1990.

[40] Rui Oliveira. Planning and motion control in autonomous heavy-duty vehicles. Master's thesis, KTH, Automatic Control, 2014.

[41] Matt Zucker, James Kuffner, and Michael Branicky. Multipartite rrts for rapid replanning in dynamic environments. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1603–1609. IEEE, 2007.

TRITA XR-EE-RT 2015:012
ISSN 1653-5146