CHALMERS
UNIVERSITY OF TECHNOLOGY

High Performance Autonomous Racing with RC Cars

Designing models and controllers to implement a small scale system of multiple autonomous cars

Bachelor's thesis EENX15-18-23

Patrick Engström
Daniel Fredriksson
Oscar Olesen
David Olsson
Sanna Sandberg
Ahmed Hamdy Faramawy Mahmoud Shams El Din

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018-05-14

Bachelor thesis 2018:05:14

High Performance Autonomous Racing with RC Cars

Designing models and controllers to implement a small scale system of multiple autonomous cars

Patrick Engström
Daniel Fredriksson
Oscar Olesen
David Olsson
Sanna Sandberg
Ahmed Hamdy Faramawy Mahmoud Shams El Din

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2018-05-14

High Performance Autonomous Racing with RC Cars
Designing models and controllers to implement a small scale system of multiple autonomous cars

Supervisor: Baláz Kulcár - Department of Electrical Engineering
Examiner: Torsten Wik - Department of Electrical Engineering
Co-examiner: Péter Kovács - Department of Electrical Engineering

Cover: Picture of remote controlled car used in the project.

Abstract

The field of autonomous cars has been developing rapidly in recent years. One of the primary arguments in favour of autonomous cars is the potentially reduced number of traffic accidents. There are several critical areas that must be thoroughly researched and investigated. The most fundamental aspect of an autonomous car is being able to follow the road while maintaining speed. This is the primary focus of this thesis.

One of the problems with autonomous driving is to make the system react in adequate time, so that control over the vehicle is preserved at high speed. Since racing inherently demands the car to travel as fast as possible while staying on course and avoiding obstacles, developing a system that allows this, although in a small scale, is the purpose of this thesis. The methodology is to first mathematically model the car using a single-track model. From this model it is possible to design controllers for the steering angle and speed of the car. This is then simulated, and afterwards realised indoors with small scale remote controlled cars, using a camera mounted in ceiling to determine the position and direction of the car. Obstacle avoidance can then be added, to let multiple cars avoid, and possibly race, each other.

This gives new insight into the difficulties and necessities of driving autonomous vehicles at high speed. The data collecting and processing cause delays that simple controllers without prediction cannot handle to satisfaction, and high accuracy in the vehicle model is crucial to the performance of the controllers. For future work, more focus on predictive control can therefore make even more contribution to this area of research.

Keywords: Control, PID controller, Autonomous vehicle, RC Racing

Sammanfattning

Forskningsarbetet kring autonoma bilar har utvecklats mycket snabbt under senaste år. Ett av de främsta argumenten för autonoma bilar är möjligheten att minska olyckor i trafiken. Det finns flera kritiska områden som måste noggrannt forskas och undersökas. De mest grundläggande aspekterna av en autonom bil är förmågan att hålla sig på vägen och samtidigt hålla hastighet. Detta är rapportens fokus.

Ett problem med autonom transport är att utveckla system med tillräckligt korta reaktionstider, så att fordonet kan kontrolleras även i höga hastigheter. Eftersom motorsport oundvikligen kräver att bilarna kör så fort som möjligt, samtidigt som de undviker hinder och håller sig på banan är den här rapportens syfte att skapa ett småskaligt system som kan hantera detta. Metoden är att först utveckla en matematisk modell, och utifrån denna model sedan designa regulatorer för styrvinkel och hastighet. Avslutningsvis simuleras detta, för att sedan förverkligas inomhus med småskaliga radiostyrda bilar genom att använda en kamera som monteras i taket för att bestämma position och riktning på bilarna. Undanmanövrering kan då utvecklas, så att flera bilar kan köra samtidigt utan att krocka.

Arbetet ger nya insikter om svårigheterna med självkörande fordon i höga hastigheter. Att samla in data och sedan behandla den skapar fördröjningar som icke-prediktiva regulatorer inte kan hantera tillräckligt väl, samt kräver fordonsmodellerna hög noggrannhet för regulatorernas prestanda. Senare arbeten bör därför fokusera på mer avancerade regleringsstrategier, särskilt för hastigheten.

Keywords: Reglering, Självkörande fordon, radiostyrning, racing.

Contents

Nomenclature

- X - Global X position (m)
- Y - Global Y position (m)
- ψ - Car yaw angle. (rad)
- v_x - Car longitudinal velocity (m/s)
- v_y - Car lateral velocity (m/s)
- ω - Car angular velocity (rad/s)
- F_{rx} - Car rear longitudinal force (N)
- F_{fx} - Car frontal longitudinal force (N)
- F_{ry} - Car rear lateral force (N)
- F_{fx} - Car frontal lateral force (N)
- m - Car mass (kg)
- I_z - Car rotational inertia ($kg \cdot m^2$)
- δ - Car steering angle (rad)
- D - Car control signal
- l_r - Distance from c.g to rear wheel (m)
- l_f - Distance from c.g to front wheel (m)
- α_r - Rear slip angle (rad)
- α_f - Frontal slip angle (rad)

- C_d - Drag coefficient
- K_d - Derivative gain
- K_i - Integral gain
- K_p - Proportional gain
- T_d - Derivative time constant
- T_i - Integral time constant
- τ - Time constant
- φ_m - Phase margin
- ω_c - Cross-over frequency
- M_S - The sensitivity functions maximal gain
- M_T - The complementary sensitivity functions maximal gain
- $r(t)$ - Reference signal
- $u(t)$ - Control signal
- $e(t)$ - Error value
- $e_m(t)$ - Measured error value
- $y(t)$ - System output
- $y_m(t)$ - Measured system output

1

Introduction

The area of autonomous cars has long been of interest to researchers[1]. Companies such as Google, Volvo and Tesla are working to develop a truly autonomous car. Progress towards a fully autonomous vehicle has been made, with some companies test driving autonomous cars on public roads. Volvo is one example, where the company has been testing its autonomous system in Gothenburg with ordinary families [2]. If the automotive industry succeeds with developing fully autonomous vehicles, traffic accidents could potentially be reduced, as human error is the cause of most incidents. Traffic efficiency also could be improved, reducing travel time [3].

Despite great progress in autonomous car technology over the past few years, a fully autonomous vehicle is yet to be introduced to the general public. This is due to the limitations in the current technology. An example is Tesla's most advanced Autopilot, which has caused crashes. In Tesla's defence however, the Autopilot was never advertised as fully autonomous [4].

The main challenges to an autonomous car would be lane keeping, speed control and obstacle avoidance. If a vehicle could handle these challenges at high speed, it is believed that the same system would be able to operate at lower speeds with higher stability. It is also reasonable to think that autonomous cars very seldom will drive as fast as physically possible, but if it is safe at those speeds, it most certainly will be safe in lower speeds. Therefore, it is relevant to investigate the behaviour of an autonomous system at high speed.

Today, tests are performed with autonomous racing cars competing against each other. Several new automobile features like improved aerodynamics and active suspension, were introduced through motor sport [5]. It is therefore reasonable to suggest that developments made in high speed autonomous racing could be applied to the commercials car market, which would provide even safer autonomous cars for consumers. Then crashes caused by human error could be reduced, making the roads safer, and the relevance of high performance racing important.

Therefore, this bachelor thesis project is aiming towards creating a autonomous small scale system, similar to a racing situation, where the cars try to drive as fast as possible. Thus, developing an environment to learn and investigate the difficulties regarding high speed autonomous racing.

1.1 Background

It is only recently that the connections between autonomous racing and autonomous vehicles in regular traffic have started to be investigated. As already established, this kind of research aims to increase safety routines in autonomous vehicles by learning from race car drivers and autonomous race cars.

Standford University has conducted tests with their autonomous vehicle in racing conditions. The Lab Director of Stanford University's dynamic design lab claims that if they can push the autonomous car to its limits and still and maintain control, that could help provide insight on how to improve safety in commercial autonomous cars [6]. The Stanford University's dynamic design lab has researched high-speed performance of cars and has published works on different topics regarding problems occurring when racing with both normal and autonomous cars. For example a Ph.D. dissertation by M.Kritayakirana at Stanford Dynamic design lab, regarding high-speed performance vehicle control, notes that

> "Many road accidents are caused by the inability of drivers to control a vehicle at its friction limits, yet racecars drivers routinely operate a vehicle at the limits of handling without losing control. If autonomous vehicles or driver assistance systems had capabilities similar to those of racecar drivers, many fatal accidents could be avoided" [7].

At the University of California, Berkeley, their Model Predictive Control (MPC) lab does real-time implementations of constrained predictive model-based control on vehicles. Among many MPC projects, they have also investigated problems occurring when predictive control methods are applied to autonomous racing cars. For instance, they simulate a nonlinear MPC, to improve lap time and increase speed [8]. Drift control, occurring at high speeds, was also investigated[9]. Some results from their real-sized autonomous cars doing extreme manoeuvres in icy conditions are shown in videos on their website [10].

This project is inspired by the Optimal RC Autonomous Racing (ORCA) project carried out at the Swiss Federal Institute of Technology in Zürich. That project is aimed to demonstrate predictive control solutions working in real life using autonomous Remote Controlled (RC) cars where the goal was to improve lap-times while avoiding obstacles [11]. The problem presents many challenges regarding a multitude of aspects in autonomous control of high speed objects and obstacle avoidance capabilities. The results from these kinds of project are valuable for real life applications such as autonomous cars.

Previous success with obstacle avoidance for RC cars has been made. For example at Uppsala and Chalmers[12][13]. ORCA is yet the only project pushing the small scale test to its maximum velocities using advanced control systems such as MPC.

1.1.1 Purpose

The purpose of this project is to develop a small scale system that allows high speed autonomous driving and obstacle avoidance. Resembling the system architecture of the ORCA project it will be implemented at an indoor track with two small RC cars and an overhead camera tracking the cars. **Therefore, the purpose is to create a system architecture which can handle all necessary calculations to autonomously drive the small RC cars as fast as possible.** This system will involve control theory, indoor positioning and car modifications.

This project will give an understanding of the problems and limitations that such a system could have and to what extent it could be applied in modern day automotive applications. This subject is prevalent in today's society where the automotive industry is racing to develop a truly autonomous car.

1.1.2 Problem description and aims

The main challenge with the project would be developing a system able to handle high speed performance. To achieve this, a similar system architecture to the one used in the ORCA project was chosen, as shown in figure 1. The system consists of two miniature RC cars fitted with IR-LEDs. A camera, overlooking the virtual track, provides data, such as velocity and orientation, and sends it to the controller. Using a predetermined virtual path, the controller utilises the provided data to control the paths and velocities of the car. Developing this system involves the points below:

- Mathematical modelling of the small RC cars.
- Creating a virtual path.
- Controller design for path keeping and velocity maximisation.
- Indoor positioning with image analysis tools.
- Modifying the cars and their remote controllers to communicate with a computer.

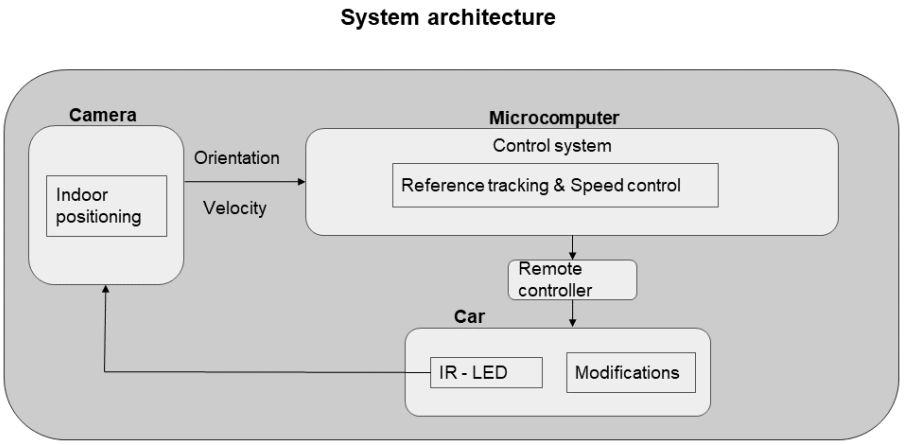


Figure 1: An overall description of a system architecture inspired by the ORCA project.

Aims

The aim, as partially described in the purpose, is to develop the system described as above, and drive the small RC cars as fast as possible around a virtual track, while avoiding obstacles. This involves linearising a nonlinear vehicle model, and constructing a linear control system, inspired by established areas of research within controller design. Moreover an accurate tracking system needs to be developed to provide the controller with required data. Finally a remote connection needs to be established, to send controller outputs to the cars. Thus the goal of creating a small scale learning environment for autonomous racing with cars is achieved.

1.2 Scope and delimitations

Since the small cars will be tested to their limits, high demands will be put on making an accurate mathematical model of the car, and a precise indoor tracking system. This will be put into priority. Since several vehicle models exists, a simple bicycle vehicle model will be used, for its simplicity and ability to simulate the dynamics with satisfying accuracy.

There are enough aspects regarding just the controller to make it a project on its own. Due to time constraints the controller type will be limited to a Proportional, Integral and Derivative gain (PID) controller, as opposed to model predictive control types. The robustness of the controller will also be considered of lesser importance. The reason for this is that high-speed performance is a strong primary focus, which means the system will operate close to stability failure, particularly within the testing environment.

Complexity and versatility in obstacle avoidance will be limited. The project aims to deliver a system that can avoid static obstacles at high speed. Since previous project has been conducted when RC cars avoid obstacles, it will not be of highest priority. If there is time, the system can be further developed to avoid moving obstacles or overtaking of other cars.

2

Theory

Creating the system architecture in figure 1 involves mathematical modelling and controller design. This chapter presents the basic structure of the mathematical model used and how an optimum controller is adapted to the model and purpose.

The mathematical model of the car will be a nonlinear system of differential equations. To enable the use of more simple control methods, the model is linearised about certain work points. The linear approximation of the car will be a Multiple Input Multiple Output (MIMO) system. To use linear control methods, the MIMO system will be converted to a Single Input Single Output (SISO) system. This SISO system is then used to develop linear PID controllers.

The methods concerning indoor positioning are described in Chapter 3, since it is not developed from any special mathematical or control theory.

2.1 Mathematical model of the car

To create a working control system, a sufficiently precise model of the car is needed so that the controllers can be designed and simulated. The vehicle model is based on a single-track type of model referred to as a bicycle model.

2.1.1 The bicycle model

To describe the dynamics of the car, a bicycle-vehicle model is used. This is a simple vehicle model that describes the motion of a symmetric vehicle by simplifying it as a a single track vehicle, such as a bicycle [14]. A bicycle model is sufficiently accurate under the assumptions that load transfer is negligible and that vehicle width is by comparison smaller than turning radius. Negligible load transfer can be assumed because the RC car has very low centre of gravity and only some minor suspension. The assumption that turning radius is larger than the car width is made because the RC car has a width of 6.9 cm, and testing shows that the RC car has a no-slip minimal turning radius of roughly 23 cm. By examining the different forces acting

upon the vehicle, the equations for the dynamics of the car can be obtained. Figure 2 illustrates the dynamics of the bicycle model.

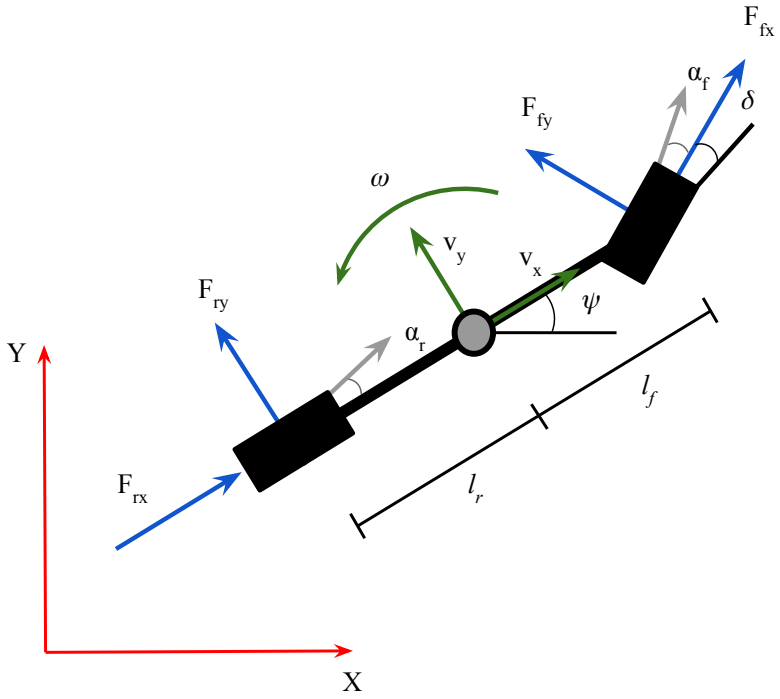


Figure 2: A visual representation of the bicycle model used in the project.

Equations of motion

From the physical quantities illustrated in figure 2, equations describing the motion of the vehicle are defined as the following.

$$\begin{aligned} \dot{v}_x &= \tfrac{1}{m}(F_{rx} - F_{fy}sin(\delta) + mv_y\omega) \\ \dot{v}_y &= \tfrac{1}{m}(F_{ry} - F_{fy}cos(\delta) - mv_y\omega) \\ \dot{\omega} &= \tfrac{1}{I_z}(F_{fy}l_f cos(\delta) - F_{ry}l_r) \end{aligned} \tag{1}$$

In the equations of motion (1), the x and y subscripts denote local coordinates. To be able to determine the position of the car, a conversion to global X and Y coordinates is needed. These are particularly important for simulation. This conversion achieved by expressing the global velocities $\dot{X}$, $\dot{Y}$ and yaw rate $\dot{\psi}$ in terms of the local velocities v_x, v_y and w as shown in 2.

$$\begin{aligned} \dot{X} &= v_x cos(\psi) - v_y sin(\psi) \\ \dot{Y} &= v_x sin(\psi) + v_y sin(\psi) \\ \dot{\psi} &= \omega \end{aligned} \tag{2}$$

Nonlinear model

By combining the equations in (1) and (2) a state-space model of the car is acquired.

$$\begin{cases} \dot{X} = v_x cos(\psi) - v_y sin(\psi) \\ \dot{Y} = v_x sin(\psi) + v_y sin(\psi) \\ \dot{\psi} = \omega \\ \dot{v}_x = \frac{1}{m}(F_{rx} - F_{fy} sin(\delta) + m v_y \omega) \\ \dot{v}_y = \frac{1}{m}(F_{ry} - F_{fy} cos(\delta) - m v_y \omega) \\ \dot{\omega} = \frac{1}{I_z}(F_{fy} l_f cos(\delta) - F_{ry} l_r) \end{cases} \tag{3}$$

The state-space model describes the nonlinear behaviour of the vehicle. The states Y and X are the global coordinate positions, v_x and v_y are the longitudinal and lateral velocities, ψ and ω are the yaw angle and yaw rate. The variable δ is a control signal that represents the steering angle of the wheels. The second control signal which the forces are dependant on is the PWM duty cycle percentage D for the motor, which ranges from 0-1 and is a part of the motor force F_{rx} as shown in section 2.1.2. The state space will be linearised around specific operating points and used to design the controllers. Since the goal of the project is to maximise track performance, it is essential that the vehicle model is accurate to achieve the desired result.

Since some parameters are specific to the car in question, they will be determined by test driving the car in different driving scenarios. With an accurate model, the dynamics of the car can be simulated and measured, which is essential to the design of the controller.

Acting forces

The bicycle model describes all tyres connected to one axle as one single, meaning that the four tyres of the car could be simplified to one frontal and one rear. The forces acting on the tyres can then be described as the following four:

- F_{fx} - Longitudinal forces acting upon the front wheel, only rolling resistance due to rear-wheel drive.
- F_{fy} - Lateral forces acting upon the front wheel caused by steering, cornering, slipping, and other lateral movement that depends on the tyres.
- F_{rx} - Longitudinal force acting upon the rear wheel, net force from the drive line.

- F_{ry} - Lateral force acting upon the rear wheel caused by cornering, slipping, and other lateral movement that depends on the tyres.

Because the car has rear-wheel-drive the only longitudinal force contribution to the front wheel is rolling resistance caused by friction. The rolling resistance is hard to estimate due to small scale. Therefore F_{fx} is regarded as zero, but its effect will indirectly be included in the motor equation. Another force necessary to include is drag. Drag force is the resistive force that acts upon a body moving through a fluid such as air, in other words air resistance. The drag force is dependant on the fluid density, surface area, velocity, and a coefficient determined by shape factors [15]. The drag force affects the whole body but will be included in the rear longitudinal force equation for the sake of compactness.

2.1.2 Motor model

As previously stated, the driving force that makes the car move is caused by the motor F_{rx} acting upon the rear wheels. This force can be calculated using a model of an electric motor.

Usually when modelling an electric motor, measurements and different motor parameters are needed. In this case however, since sufficient data about the motor used was not available, a simpler parameterised model was used to describe the driving force F_{rx}. [16]

$$F_{rx} = C_{m0}D - C_0 - C_1 v_x - C_D A\frac{\rho v_x^2}{2} \tag{4}$$

Here, C_{m0} is the force parameter of the electric motor. C_0 is the constant resistive parameter for the driveline, which contains contributions from rolling resistance and motor inertia. C_1 is the resistive force in the drive line depending on vehicle speed. The final term $C_D A\frac{\rho v_x^2}{2}$ is a contribution from the drag force included for the sake of keeping the equations compact.

Because a separate force for friction is included but instead indirectly included in the term C_0 the motor model's accuracy will be limited to the specific surface used for the project.

2.1.3 Tyre model

The tyres of the car play an important role in its behaviour. All lateral movement, slipping, and rotation will depend upon them. There exists plenty of in-depth work and studies on the subject, but some of it is excessive for the simple design of the tyres for an RC car. The chosen model is a simplified variant of the Pacejka magic tyre formula [17] [18]. Figure 3 shows how the formula works. The lateral force is about linear for small slip angles and for higher angles reaches a peak and later

saturates.

$$R(k) = d \cdot sin\Big\{c \cdot arctan\Big[b(1-e)k + e \cdot arctan(bk)\Big]\Big\} \tag{5}$$

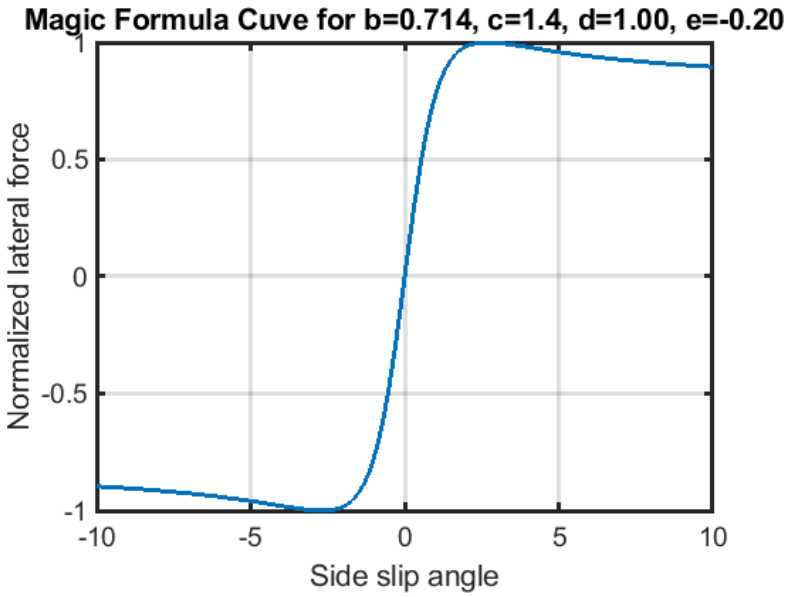


Figure 3: Illustration of the magic tyre formula and how the lateral force depends on slip angle

The Pacejka magic tyre formula is a model based off semi-empirical experiments. It is commonly used because of its accuracy compared to its relative simplicity. The simplified formula of (5) sets the term e to zero.

$$\begin{aligned} F_{fy} &= d_f sin\Big(c_f(b_f\alpha_f)\Big) \\ F_{ry} &= d_r sin\Big(c_r(b_r\alpha_r)\Big) \\ \alpha_f &= \delta - arctan\Big(\tfrac{\omega l_f + v_y}{v_x}\Big) \\ \alpha_r &= arctan\Big(\tfrac{\omega l_r - v_y}{v_x}\Big) \end{aligned} \tag{6}$$

The parameters d, c, and b with corresponding subscript are the peak factor, shape factor, and stiffness factors in respective order. α_f and α_r are the front and rear slip angles. It is assumed that the parameters for both front and rear tyres are equal, due to the simplicity of the tyres, thus the subscripts are from now on omitted. The model possess a weakness at very low or zero velocity due to the division by zero. This is an accepted inaccuracy because the model only intends to describe the car during motion, starting from standing still is not of interest.

2.2 Linearising the model

To linearise a nonlinear model is the process of formulating a linear approximation of a nonlinear system around a specific point. In situations where a nonlinear model is hard to work with and it is assumed to operate at or close to certain points, in other words variable values, a linear approximation at those points can be used instead. [19]

$$f(z,u) \approx f(z_0,u_0) + \frac{\partial f(z_0,u_0)}{\partial z}\bigg|_{z_0,u_0}(z-z_0) + \frac{\partial f(z_0,u_0)}{\partial u}\bigg|_{z_0,u_0}(u-u_0) \quad (7)$$

To linearise a nonlinear model a suitable equilibrium work point has to be determined.

$$z_0 = [X_0, Y_0, \psi_0, v_{x0}, v_{y0}, \omega_0]$$

Equilibrium implies that the car is driving in steady-state conditions, with no changes to acceleration in any direction. Since a linear model is specific to one work point, deviation from said work point will decrease the accuracy. Therefore a variety of work points for different driving conditions are chosen because no absolute state of equilibrium will exist.The driving conditions that are of interest are steady-state cornering at different speeds and driving straight.

By choosing multiple work points z_0 and linearising the model accordingly, the overall accuracy of the system can be improved. Multiple linear models require multiple controllers, created according to each model. Thus the system, though linear, will be able to handle many different scenarios. A linear state space model can then be constructed for analysis and controller design.

$$\begin{aligned} A &= \frac{\partial f(z_0,u_0)}{\partial z}\bigg|_{z_0,u_0} & B &= \frac{\partial f(z_0,u_0)}{\partial u}\bigg|_{z_0,u_0} \\ C &= \frac{\partial g(z_0,u_0)}{\partial z}\bigg|_{z_0,u_0} & D &= \frac{\partial g(z_0,u_0)}{\partial z}\bigg|_{z_0,u_0} \end{aligned} \quad (8)$$

$$\begin{aligned} \Delta\dot{z} &= A\Delta z(t) + B\Delta u(t) \\ \Delta u &= C\Delta z(t) + D\Delta u(t) \end{aligned} \quad (9)$$

Equation 8 show how the state matrices are constructed, and equation 9 how the linearised state space is represented [20]. The function f is a function of the state variables z and control signals u that returns the derivative of a state $\dot{z}$. The function g is a function that simply returns whichever signal being measured. The matrix D is a feedforward matrix and unused because the control signals do not directly affect the states.

2.3 Controller design

The previous section has stated that multiple linear models will be used to describe the car in different situations such as driving straight forward or turning, thus simulating a nonlinear behaviour with linear models. This demands multiple controllers for each linear model. Furthermore, the linear models are MIMO state space models with two inputs and six states as outputs, from equation 1. The two, inputs δ represents the "steering wheel" of the car, and D represents the "gas pedal", hence δ affects the direction of the car and D affects speed. These signals will be the control signals to follow a virtual track as fast as possible.

Established research about controlling RC cars, to resemble a small scale autonomous racing situation, uses nonlinear control methods as stated in section 1.1. This project is as mentioned in the delimitations 1.2 constrained to linear and basic control methods, hence a PID controller is the best suited for the intended goal [21].

Using PID controller means no methods to handle a MIMO systems is possible [21]. Instead one controller for each input signal is the most reasonable solution. Yet, linear model provides six different outputs. The desired PID controller is only compatible with a SISO model. Meaning that the six outputs signals from the model needs to be modified or combined in a way to only supply the controller with one error value per input and PID without losing to much information. Again using multiple simpler control methods to mimic a more advanced nonlinear control system. Intuitively choosing yaw angle, ψ, as the only output from the model to the PID controlling steering should be sufficient, as further elaborated in section 3.3.1 about steering control.

The multiple PID parameters will be gathered in a look-up table. Using look-up tables is commonly used to model nonlinearities over a wide range of work points [22]. This project applies it to linear and basic control methods instead, creating a look-up table to enable functionally for the cars over a great amount of work points.

An overall control design satisfying all constraints from the complex structures of velocity dynamics fitted in a simpler system, is described in figure 4. A detailed course of action to implement that design is described in Chapter 3, however the two decoupled PID controllers do not need to resemble each other. The controller steering the cars along the virtual track will contain the look-up table and requires great accuracy to stay on the line. Speed control could simply be an algorithm pushing the cars to drive as fast as possible without deviating to much from the track.

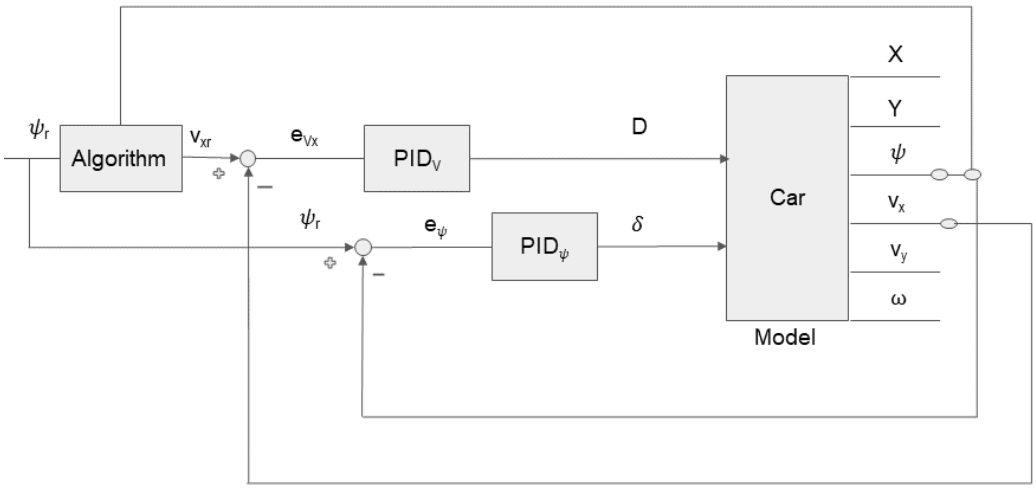


Figure 4: Scheme describing an overall control system, with two decoupled control loops. Both control loops will be further elaborated in section 3.3. The structure of the algorithm is further elaborated on in section 3.3.1.

The PID controller

A PID controller is a control loop feedback controller, used in a variety of applications. The controller calculates an output signal using error values, which is the deviation of the systems current state from the desired operating point. As the name implies, a PID controller consists of three parts. Proportional gain, integral gain and derivative gain. These values are obtained by analysing the system's transfer functions. This requires a linear SISO model.

Expressed in the time plane the PID controller is as in equation 10. Where the control signal u depends of the measured error value: $e_m(t) = r(t) - y_m$. See figure 5.

$$u(t) = K_p\left(e_m(t) + \frac{1}{T_i}\int_0^t e_m(\tau)d\tau + T_d\frac{de_m(t)}{dt}\right) \tag{10}$$

Laplace transformation of the relations in the time plane means: $U(s) = F_{PID}(s)E_m(s)$ which leads to the transfer function of the PID controller as in equation 11. Introducing the relation of both $K_i = K_p/T_i$ and $K_d = K_pT_d$ the PID is commonly described as in equation 12.

$$F_{PID} = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \tag{11}$$

$$F_{PID} = K_p + \frac{K_i}{s} + K_d s \tag{12}$$

The proportional gain K_p is proportional towards the control signal: $u(t) = K_p e_m(t)$. Increased K_p means increased response speed in the loop transfer function since the output y is the natural frequency of the loop transfer function (see figure 5). Increased value of K_p does decrease stability margin and raises the integral- and derivative gain, which could lead to bigger overshoots in y.

The integral gain K_i/s raises the gain at lower frequencies, hence increases the control signals accuracy in the static responses and enhances the compensation of low frequency upsets from the load disturbances. However, too great value of K_i causes bad stability margins.

The derivative gain, $K_d \cdot s$ contributes to stability margins and compensation for overshoot and inferior stability margins due to the integral gain. Mathematically it gives an increasing contribution to the loop transfer function, unfortunately which also increases the high frequency gain. The high frequency gain can increase sensitivity for measurement noise. When the car is following a line a too great value of K_d could cause the car to oscillate around the line in a wobbly way. This section (2.3) is based on theory from B.Lennartsson's book: *Reglerteknikens grunder* [21].

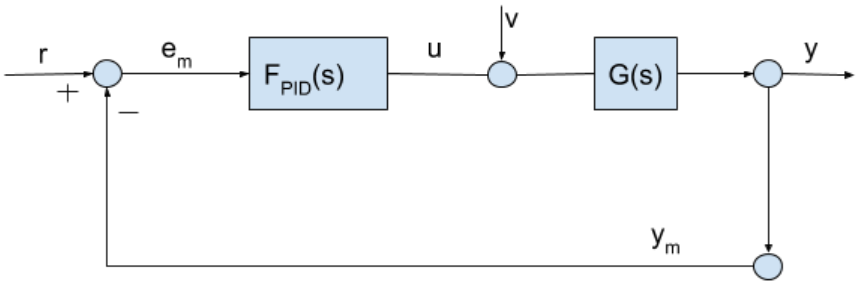


Figure 5: Simple feedback control-loop with PID controller. G(s) represents the model, and F_{PID} the PID controller, u is the control signal and e_m the measured error value, y is the output and r is the reference signal [21].

Look-up table

To address the issue of the RC car and model behaving differently in different regions of it, a look-up table is applied to the control loop for steering. A look-up table is an array of values stored before a program is run. During operation, the program will fetch these values according to a predetermined algorithm. This replaces run time computing of different parameters. In this project, a look up table is used to store different PID parameters for the steering control. During racing, the program will use known data about the car, such as speed, angle and angular velocity, to determine suitable parameters for controlling the car.

2.3.1 PID tuning with heuristic stability margins

Functional and stable PID controllers will be created using heuristic reasoning about the sensitivity function equation (13) and the complementary sensitivity function equation (14). The theory behind this section is according to [23] and [21].

$$S(s) = \frac{1}{1 + L(s)} \tag{13}$$

$$T(s) = 1 - S(s) = \frac{L(s)}{1 + L(s)} \tag{14}$$

The goal is to find a PID controller that corresponds to the maximum values of the sensitivity and complementary sensitivity function, named: M_s in equation (15) and M_t in equation (16). Then according to [23] this could create an optimum controller. Optimum controller defined as having good mid frequency robustness and good balance between control activity and the accuracy of the output signal.

$$M_S = max_\omega |S(j\omega)| = \frac{1}{min_\omega |1 + L(j\omega)|} \tag{15}$$

$$M_T = max_\omega |T(j\omega)| = max_\omega \left| \frac{L(j\omega)}{1 + L(j\omega)} \right| \tag{16}$$

For designing a PID controller there are several possibilities, here the parameters will be based upon deciding phase margin (φ_m) and cross over frequency (ω_c). Phase margin being the maximum amount of phase shift possible before the loop transfer function turns unstable. Cross over frequency means the frequency where the magnitude of the model is 0 dB and the phase corresponding to that frequency. When designing a PID controller it is possible to start by designing the less complex Proportional and Integral gain (PI) controller to obtain initial values for ω_c and φ_m.

The basic idea is to create an amount of PI controllers by using an assumed interval of crossover frequencies and phase margins. Then choosing the most optimal of those PI controllers by calculation M_s and M_t, which provides great initial values to calculate the desired PID controller.

A simple principle of controller design is to specify the loop transfer function $L(s) = G(s)F(s)$ for a frequency ω_0, if the models transfer function is known for that particular frequency: $G(j\omega_0) = |G(j\omega_0)|\angle G(j\omega_0)$. This makes is possible to calculate the controllers magnitude: $|F(j\omega_0)| = |L(j\omega_0)|/|G(j\omega_0)|$ and phase: $\angle F(j\omega_0) = \angle L(j\omega_0) - \angle G(j\omega_0)$. Since this method is based upon ω_0 being know, an application of this method is to put ω_0 as the cross over frequency ω_c, which corresponds well to the desired method of tuning presented above. Then the magnitude and phase of the loop transfer function equals $|L(j\omega_c)| = 1$ $\angle L(j\omega_c) = -180^\circ + \varphi_m - \angle G(j\omega_c)$. Then a PI controller can be design using equation 17, where φ_m is the desired phase margin for the controller, hence both ω_c and φ_m are known [21].

$$\begin{aligned} |F(j\omega_c)| &= \tfrac{1}{|G(j\omega_c)|} \\ \angle F(j\omega_c) &= -180^\circ + \varphi_m - \angle G(j\omega_c) \end{aligned} \tag{17}$$

The above presented method of designing a controller is applicable to the system in question, since the phase margin and amplitude of the model is known and can be used to find the phase margin and magnitude of the controller. To find a suitable cross over frequency for the system to start with, the following observation was made [23]:

$$\omega_c = 0.6 \cdot \omega_{G180} \tag{18}$$

To tune this PI controller an interval of the phase margin (φ_m) and the cross over frequency (ω_c) for example a PI controller: $45^\circ < \varphi_m < 60^\circ$ and $0.3\omega_{G180} < \omega_c < 0.6\omega_{G180}$ where $\omega_{G180} = \angle G(j\omega_{G180} = -180^\circ)$. Then calculating several PI controllers and their several M_s and M_t. The PI with the highest Ki that also fulfils:

$$\begin{aligned} M_s &\leq 1.7 \\ M_t &\leq 1.3 \end{aligned} \tag{19}$$

is the most optimal PI controller within the interval [23]. Normally φ_m is 50° for both PI and PID controllers [23]. The limits in (19) are based of empirical observations [23], to guarantee a good combination of robustness and speed for a stable model. It is relevant to note that increasing the value of ω_c could increase the speed of the loop transfer function. Choosing the correct value of ω_c is crucial to achieve the correct speed that correspond to that of the cars.

3

Methods

For the development of this system, miniature RC cars were chosen instead of full-sized real ones. This choice is due to economic limitations and has the advantage of being much safer for a student project, where the goal is to achieve maximum speed. It also reduces the complexity of the system since RC cars are easier to handle compared to full-sized ones. The small RC cars will serve as sufficient models for real cars since the vehicle model and control algorithms should be rather similar for small and large vehicles. Using the vehicle model described in state space model (3), the RC car's dynamics could be simulated which is then used to design the controllers.

The system architecture consists of positioning and controlling of the cars, and the control system. The system also has to be modelled to make it possible to simulate it and get accurate control parameters. A nonlinear model is created and linearised around multiple work points. The linear models needs to be precise, since a high performance system demands precise calculations. The linear model is further used to design a special type of PID controller which utilises all of the different linear models. Each linear model represents a situation, for example a sharp turn or straight forward. This information is merged to one PID controller for each linear model combined in a Look-Up table and is used to control the steering angle of the car.

The speed control will be achieved by an algorithm based on the slowest of two sub-algorithms. The first of which slows the car down if its current direction deviates from the target direction, otherwise it can increase it's velocity. The second tracks if there is an upcoming angle change in the track ahead, if there is then the car must slow down, otherwise it can increase it's velocity. This is then used as the reference value to a PID which controls the velocity of the car.

The indoor positioning system is based on a camera hanging down from the ceiling, detecting the movements of the cars. The information obtained through the camera can then be used to control the cars in the direction of choice.

3.1 Autonomous system structure

The autonomous system used in this project consists of a miniature RC cars that are controlled by a computer. All calculations needed for controlling the cars are done on this computer, which is sends control signals to the cars. The computer gets all its information from a camera hanging from the ceiling.

3.1.1 Remote controlled cars

The RC cars in the system are two RC model cars of type Kyosho Mini Z. Scaled at 1:27 in relation to their real counterparts, with a length of around 160mm[24]. They are fitted with LEDs, emitting IR light, to facilitate the positioning by imaging as shown in figure 6. It is desirable to detect both direction and position of the car, as such the LEDs are positioned as a non-equilateral triangle with one in the front and two in the back; essentially forming an arrow. One of the cars also have an additional LED to allow separation of two cars by counting LEDs. To power the LED the cars are supplied with an additional battery mounted on the inside of the chassis.

Figure 6: This picture shows one of the cars used in the project, including the addition of the LEDs. The RC car measures 16.5x6.9 [cm].

3.1.2 Remote controller

The controllers are disassembled and modified by disconnecting the existing potentiometers for speed and steering, which are replaced by a Digital to Analogue Converter (DAC) circuit (see figure 7). Using a dual channel DAC connected to an

Arduino over Serial Peripheral Interface (SPI) in the DAC circuit enables digital control of the cars.

The Arduino is connected to the central computer controlling the system with a Robot Operating System (ROS) node running on the Arduino. The Arduino then receives the control signals to change speed and steering angle, configuring the DACs accordingly.

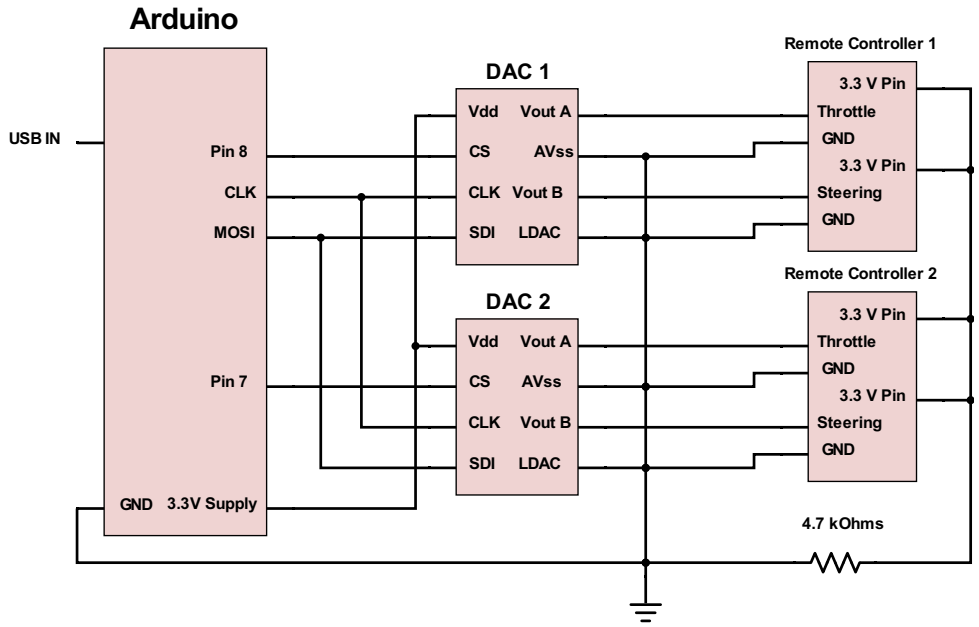


Figure 7: The simplified circuit diagram over how the cars are controller. In this setup, pin 7 and 8 on the Arduino are used for chip select.

3.1.3 Camera

The positioning system uses a Point Grey Flea3 camera[25]. The camera captures 150 frames per second (FPS) and uses a monochrome sensor and a global shutter. The high frame rate is necessary for maintaining precision while tracking since it minimises movement between data points, even at higher speeds. The monochrome sensor limits the data to a single channel instead of the multiple channel colour sensor. Since only infrared light is of interest, more channels would only result in more data to gain the same result. A global shutter is preferred to the alternative rolling shutter. This since a rolling shutter captures the image row by row, meaning that each row of pixels will not all be from the same time instance[26]. Therefore, if there are fast moving objects in the image it will be distorted, making the positions of the cars at a given time uncertain. A global shutter, on the other hand, captures all the pixels in the same instant, which results in more reliable data.

An additional physical filter that removes light outside the IR spectrum is fitted to the camera. By limiting input to light of the IR spectrum little noise is permitted to disturb the light of the LEDs on the cars, compared to allowing the full spectrum. This results in a high contrast image of the cars LEDs.

This is an easier way of acquiring precision in a confined space in comparison to radar or triangulation solutions.

Software architecture

The system built utilises the ROS as a development platform. The system is developed purely in C++. It has, through OpenCV, necessary tools for image processing. The ROS architecture of using nodes for tasks combined with the possibility to record data between nodes is useful since it enables offline testing and isolated development and debugging of tasks.

3.2 Modelling

For the computer to be able to control the car, an accurate mathematical model of the car is needed. This model will be the foundation of almost all the design and theoretical work regarding the car and therefore has been given a lot of attention. Since PID controllers are used in this project the model needs to be linearised, when this is done the different parameters needs to be identified to complete the model of the car.

3.2.1 Linearisation

From the vehicle model (3), a linearised system of equations can be derived, from which PID controllers can be designed. Classic PID controller design strategy dictates that the controller is designed from the transfer functions of a system that is acquired by linearising the system around an equilibrium. Because the system describes a car in motion with global coordinates, a state of equilibrium cannot be found for $\dot{X}$, $\dot{Y}$ or $\dot{\psi}$. Thus the chosen strategy is to linearise the system around multiple states of motion for the car, meaning a work point will be defined only by v_{x0}, v_{y0} and ω_0.

Linearising the equations in (3) poses a few problems. The latter three states can be linearised around where they equate to zero because it only implies a state of no accelerations, which it is fair to design the controller around. The first three states bring some issues. The global velocities $\dot{X}$ and $\dot{Y}$ do not posses any values where a linearisation is reasonable because the car can move at various speed in the whole

coordinate system. The yaw state $\dot{\psi}$ should not be zero either. Because the yaw derivative is equal to the angular velocity ω, there is no point in linearising it. It is already equal to ω_0.

Because of these factors, as well as the global velocity equations being the only two that depend on ψ, the chosen strategy is to focus only on the latter three equations of (3). Doing so, a system of three equations with five variables, the three states v_x v_y ω and the two control signals δ and D, is obtained. This system is overdetermined. As previously mentioned one of the states which are of particular interest in choosing work points is ω_0. If a value for ω_0 is already chosen, the number of variables is reduced to four. If v_{x0} is also predetermined as per choice of work points, the system becomes solvable and the corresponding control signals and resulting lateral velocity v_{y0} can be calculated. To accomplish a full linearisation of the whole system of equations, work points for ψ_0 would also necessary. Because the linearisations of $\dot{X}$ and $\dot{Y}$ are not of any particular use, ψ_0 can be set to zero or otherwise ignored.

Thus with the three states ψ_0, v_{x0} and ω_0 defining the work points for the system, and the global velocities $\dot{X}$ and $\dot{Y}$ being allowed to take whatever values they may as a result of the other equations, the linearisation can be accomplished. Due to the design strategy of using multiple work points, the calculations are done using Matlab.

By producing several linear systems around different combinations of the aforementioned states, transfer functions that can describe most of the physical states the car could be in are collected. From these, it is possible to design a space of PID parameters to form a look-up table from which the controller will estimate and choose its parameters depending on the input signals from the camera and tracking software.

Simulation

Simulating the model is a vital step. It enables the verification of model validity as well as comparison between nonlinear and linear models by studying their signals and state values. A simulation also helps with tuning physical parameters to suit the real car and testing control structure. The most suitable software for this is Simulink. In the software the equations of motion (3) are written into function blocks that simply returns the state derivatives for corresponding inputs, integrating these returns the state values.

3.2.2 Parameter identification

Many of the parameters in the equations that model the RC car are unknown at the point in time when the equations are derived. Much of the theoretical framework can be laid out with mostly symbolic parameters or placeholder values. Before applicable results can be obtained from the model, all of the parameters have to

be determined. These include mass, inertia, dimensions, centre of gravity, drag coefficient, tyre parameters, and motor parameters. Most of these properties are the same or close for both the RC cars. For the sake of simplicity and time the parameters are only calculated for one of the cars, and used for both.

Mass is measured using a scale with error margin of 2 grams. Length and width is measured using a ruler. Centre of gravity from the front is estimated with sufficient accuracy through repeated tests trying to balance the car on a thin surface. The moment of inertia around the car's vertical axis is calculated by separating the it into its three main components. Chassis, body, and batteries. The respective moments of inertia for the components is calculated by simplifying each into a standardised shapes with available formulas. They can then be summed by moving them to the assembled car's centre of gravity using Steiner's theorem [27]. The drag force $C_D A \frac{\rho v_x^2}{2}$ is estimated by measuring the car's frontal area A and using standard table values for the air density ρ [15]. The drag coefficient C_D is assumed to be comparable to that of a real car with a similar body, because the coefficient is primarily dependant on shape factors. The Ferrari model is made to resemble a Ferrari 360 GTC and therefore it is the model the coefficient should be chosen after. Ferrari does not publish this specific data for their models, so a less credible source had to be used and for a slightly different model, Ferrari 360 Modena [28]. The provided coefficient is within expected values for a sports car however and can be expected to suffice [29].

The tyre and motor parameters are difficult to determine exactly. Thus they are estimated by fitting the model behaviour to the real car with minimal deviation. The basic method for the both the tyres and motor are the same. Tests are performed with the RC car in situations where the specific parameters produce data that is easy to quantify and compare with the model. An exact solution is not very probable so instead a set of parameters is produced. The identification process is then repeated but with the identified sets, and the deviation from the test data is computed using mean squared errors.

$$E_i = \sum_{n=1}^{N} \Big(Y_n - f(x_n, y_n, P_i) \Big)^2 \tag{20}$$

By summarising the mean of the errors squared according to (20) for N amount of tests for i amount of parameter sets, where f is a function of input signals x and output states y that depends on the parameters P_i in question resulting in a value that can be compared to tested property Y_n. By selecting the parameter set with the smallest E_i it is possible to determine a good estimation that takes sign into account and emphasises larger errors.

Motor parameters

With the motor modelled as in (4) there are three parameters that have to be identified. To find suitable parameters, the model is tuned to match the behaviour of the RC car when driving in a straight line, to achieve similar acceleration and velocity. The car is run at different input signals and its velocity is measured with the tracking software. To determine the parameters, the steady state speed and momentary accelerations a_x with corresponding momentary velocity are used to construct linear equation systems. These equations are then solved which returns a set of parameters. The parameters with the least mean square error is to be used.

By setting up a system of linear equations, an exact set of parameters for the specific test sample can be determined.

$$\begin{bmatrix} D_0 & -1 & 0 \\ D & -1 & -v_x \\ D & -1 & -v_x \end{bmatrix} \begin{bmatrix} C_{m0} \\ C_0 \\ C_1 \end{bmatrix} = \begin{bmatrix} 0 \\ C_D A \frac{\rho v_x^2}{2} \\ a_x \cdot m \cdot C_D A \frac{\rho v_x^2}{2} \end{bmatrix} \tag{21}$$

The variable D_0 is the input signal for which the RC car just starts rolling, therefore the velocity and acceleration is zero. The second row equation has velocity but no acceleration, it represents the signal and velocity for which the RC car has stopped accelerating. By calculating the momentary acceleration and velocity for several points during the acceleration phase the third row can be calculated. Solving the equation system 21 for all tested signals D with corresponding measurement data v_x and a_x.

Tyre Parameters

The tyres are modelled using a simplified Pacejka's tyre model (6). The equipment normally used to determine the parameters in the tyre formula is expensive and unavailable, so empirical tuning methods are used instead. The simplest behaviour where the tyre model will come into effect is circular motion as it affects lateral slipping and rotation.

Tests are performed with the RC car being given a constant signal to motor and steering angle. Its speed and positional data is measured and processed using the camera and tracking software. The linear equation method used for the motor is harder to apply in this case because the behaviour is dependant on more than one equation, all which are ordinary differential equations. Brute force iteration is used in Matlab to find parameters that produce a reasonably similar behaviour by using the ODE45 function for a large combination of parameters. The ODE45 function is applied on the system of equations formed by the equations of motion in (1). It returns the state values over a specified time span. With the formula $v_x/\omega = r$ a steady state circle radius can be expressed. The brute force loop runs the script for a set of input signals D and δ that were used during tests and saves the parameters

if they produce a radius and velocity that are close enough to the test data. For the accepted sets of parameters the set with the least mean square error is used.

Ideally the parameters would be fitted using measurements of lateral force and slip angles, but since measuring these is not possible due to equipment limitations, the above described method is instead chosen.

3.3 Controller design and implementation

With the mathematical model in place it was possible to design a control system for the car. According to the theory chapter the control structure will consist of two separate loops, one to control steering to keep the car one the line and one to control the velocity. The steering controller is based on the linearisation cases and the reference value is the desired direction. The reference speed is calculated using an algorithm that is based on the deviation between the current global angel ψ and the target angle, and how sharp the curve of upcoming segments of the virtual track is.

3.3.1 Steering control

By splitting the control structure in two separate control loops (see figure 4) it provides a natural solution to the problem of using a MIMO state space model with a PID controller (SISO). The loop controlling steering is built around the yaw angle ψ, thus the C matrix in the state space model can be adjusted accordingly so that the only output will be ψ with respect to δ. One advantage with choosing ψ as a control signal is that it will allow for drift control if the back wheels lose traction. In the third equation of the nonlinear state space model (3), ψ is only dependant on the integration of ω, and ω in turn is independent of D. Thus no information is lost if the steering control is based entirely upon the state of ψ, since it is only dependent of the input δ, see $\dot{\omega} = \frac{1}{I_z}(F_{fy} l_f cos(\delta) - F_{ry} l_r)$. The independence of ψ from D is further shown in table 1.

Table 1: State space model, the D-matrix contains only zeros

A =

	x1	x2	x3	x4	x5	x6
x1	0	0	-0.1084	0.995	-0.09983	0
x2	0	0	0.5921	0.09983	0.995	0
x3	0	0	0	0	0	1
x4	0	0	0	-2.437	6.757	0.5998
x5	0	0	0	4.357	-56.57	-1.018
x6	0	0	0	1373	-1041	-966

B =

	u1	u2
x1	0	0
x2	0	0
x3	0	0
x4	-3.935	9.052
x5	17.07	0
x6	3935	0

C =

	x1	x2	x3	x4	x5	x6
y1	1	0	0	0	0	0
y2	0	1	0	0	0	0
y3	0	0	1	0	0	0
y4	0	0	0	1	0	0
y5	0	0	0	0	1	0
y6	0	0	0	0	0	1

Since the goal is to be able to generate a great amount of PID parameters, a method for looping through the controller designing steps is necessary. This will perhaps take away the possibility of tuning every PID to optimum, but the hypothesis is that using a look-up table will compensate for small tuning effects. Matlab was used to create this PID generating loop.

To create a PID controller from the state space model in table 1 a transfer function from δ to ψ is computed in Matlab using the command:

```
[NUM_Delta,DEN_Delta]=ss2tf(s.sys2.A,s.sys2.B,s.sys2.C,s.
    sys2.D,1)
```

which creates one transfer function per state in the A matrix regarding the input δ. Then by saving the transfer function for ψ, a SISO model is created to be used when designing PID controllers.

```
%SISO plant to be used in steering control
psi_siso=tf(NUM_Delta(3,:),DEN_Delta);
```

To create several PID controllers in a loop the Matlab function:

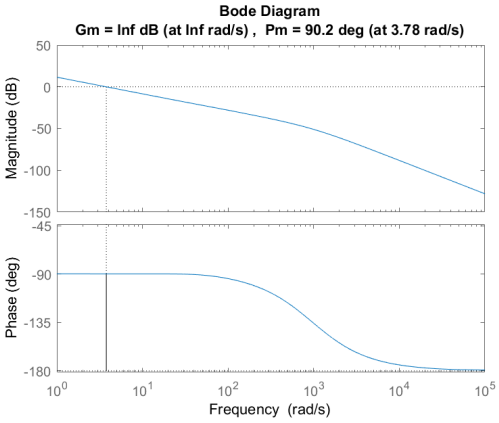
```
[C info] = pidtune(psi_siso,'PID',wc_opt, opts);
```

is utilised. As arguments it takes a desired crossover frequency, phase margin, a SISO model and a string saying which type is desired. To be able to, in a loop, enter appropriate values off both cross over frequency and phase margin some approximation needs to be done before using the function.

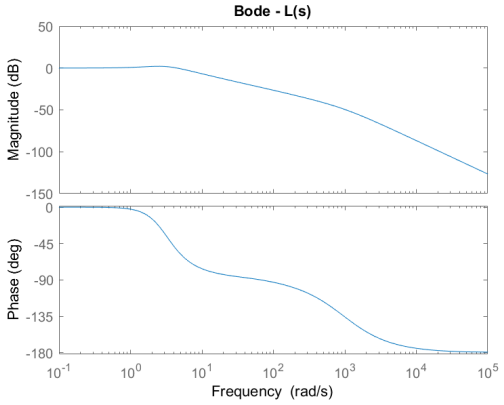
Following the theory detailed in section 2.3.1 it is possible to design and create a number of PI regulators and choose the most optimal controller by using the limits in equation (19). An initial value of both the crossover frequency and phase margin can be chosen, and later used as input to the Matlab function "pidtune();". Code to generate multiple PI controllers and choosing the optimum is available in Appendix 1. The code is from the book *Reglerteknikens grunder* [21]. Those optimisation functions also require an initial interval of frequencies. In section 2.3.1 the reasoning stated that $\omega_c = 0.6 \cdot \omega_{G180}$, but after studying how the RC cars behave another approach is possible.

For the RC car to receive a control signal and for the camera to observe any change approximately 0.8 seconds goes by. Although the car has a real reaction time closer to 0.2 seconds, it appears as many uncertainties collaborate, for example the servos taking some time stabilise. Therefore, the delay time was approximated to be higher.

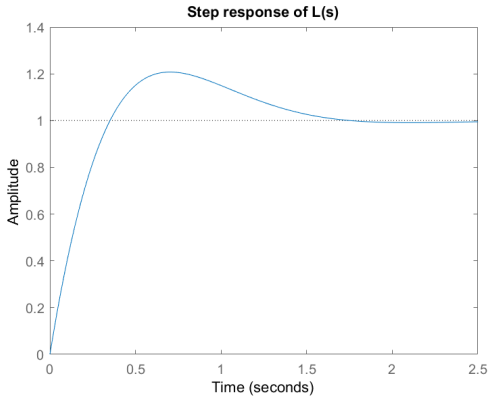
Accordingly the response time of the loop transfer function does need to be any faster then that of the reaction time. When running the code in Appendix 1 to create optimal PI controllers by using the suggested interval of frequencies, the loop transfer function's time to reach steady state is too fast. Meaning that the car will not be able to react to those signals before a new one is received. Instead of creating PI controllers with the tuning strategy from section 2.3.1, instead using the crossover frequency of the model does create more reasonable response times. See figure 8.



(a) Crossover frequency of the one model.



(b) Bode plot of L(s) after using the optimisation algorithms from section 2.3.1 and code in appendix 1.



(c) Step response of L(s) from figure 8b

Figure 8: Different plots of one SISO model with δ as input and ψ as output. This model is linearised around the work point: [X=0 Y=0 ψ=0.3 v_x=0.3 v_y=0 ω=0.8]

When an interval of crossover frequency is decided, an interval for the phase margins is approximated according to [23], the optimisation code will generate values for phase margin and cross over frequencies to input in the pidtune(); command in Matlab. The Matlab code which is available in appendix 1, uses both the optimisation code from [21] and Matlab " pidtune();" it generates PID controllers for each linear state space model. This example generates five PID controllers for five different work points.

Because the controller uses integral gain there is a need for anti-windup [30]. In systems where the actuator has a physical limit, such as a car with a maximum steering angle, a problem arises when the actuator is saturated. If the error is nonzero when the actuator is saturated the system cannot physically increase the control signal, even though the controller keeps integrating the error. This means that when the output signal reaches its reference value there is still built-up signal from the integral gain that the controller has to overcome until the output signal

decreases, which results in overshoots. To handle this a simple clamping algorithm is implemented that constricts the integral gain to $-0.3 < I < 0.3$ when the steering output δ is saturated.

Neighbouring Look-up table

A look-up table typically uses interpolation to generate control parameters. The table used here does not employ interpolation due to computation limits. Instead the closest values for any work point is used. This variation will still be referred to as a "look-up table" even though it's more a neighbouring look-up table. The look-up table is computed using different values for v_x and ω. These values represent work points for the car. During operation, the car might be operating at work points not included in the look-up table. In these cases an approximation is made to the nearest work point. It is therefore important to have linearised the model about many work points. This should improve steering control and overall performance. The overall view of the steering control is shown in figure 9.

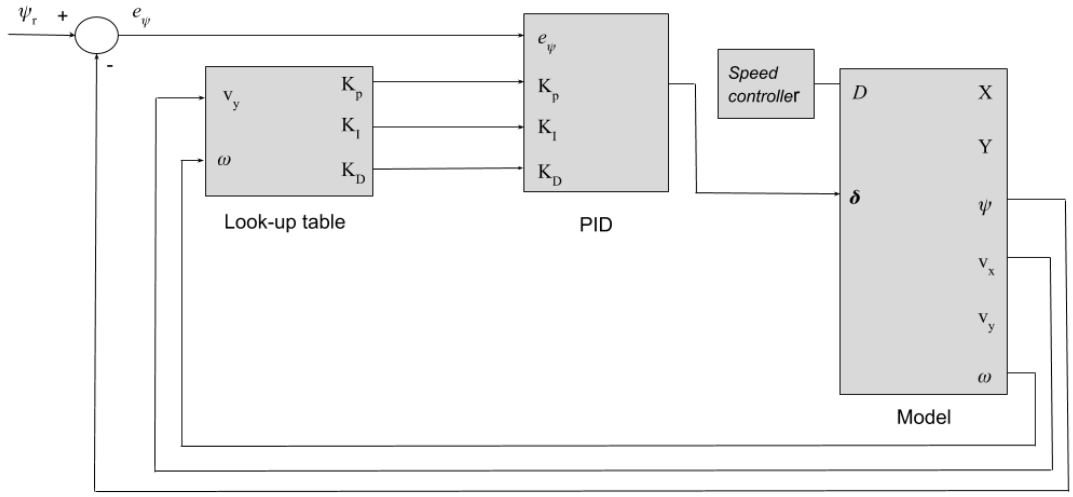


Figure 9: Scheme describing the overall steering control loop

3.3.2 Speed control

The goal of simply driving as fast as possible provides necessary constrains for the speed controller. To drive a lap as fast as possible includes staying on the line and while maintaining high speed. Controlling the speed is then simply an algorithm saying that the speed can be increased if the present global angle does not departs to much from the virtual track or decrease speed if there is an upcoming curve.

The velocity of the car is also controlled using a PID controller, see figure 4. The reference value for the PID controller is calculated using the lowest value of two separate algorithms.

One algorithm adjusts the speed based on the error in angle between where the car is actually going and where it should be going. Alas if the car is pointing against the target point, the speed will be maximum.

$$Sm_{\psi} = 1 - \frac{(\psi_r - \psi)}{\pi} \cdot W_{\psi} \tag{22}$$

Where W_{ψ} is a constant weight.

The other algorithm adjusts the speed based on the upcoming curvature of the track. It looks at a distance in front of the car, determine the same way as the target point, two equally long consecutive track segments as straight lines and determines their angular difference. If they do not differ the speed will be maximum. See figure 10 for a visualisation.

$$Sm_{Curve} = 1 - \frac{|\angle_1 - \angle_2|}{\pi} \cdot W_c \tag{23}$$

Where W_c is a constant weight. $\angle_1$ and $\angle_2$ are the angles of the two consecutive track segments starting from a distance before the car.

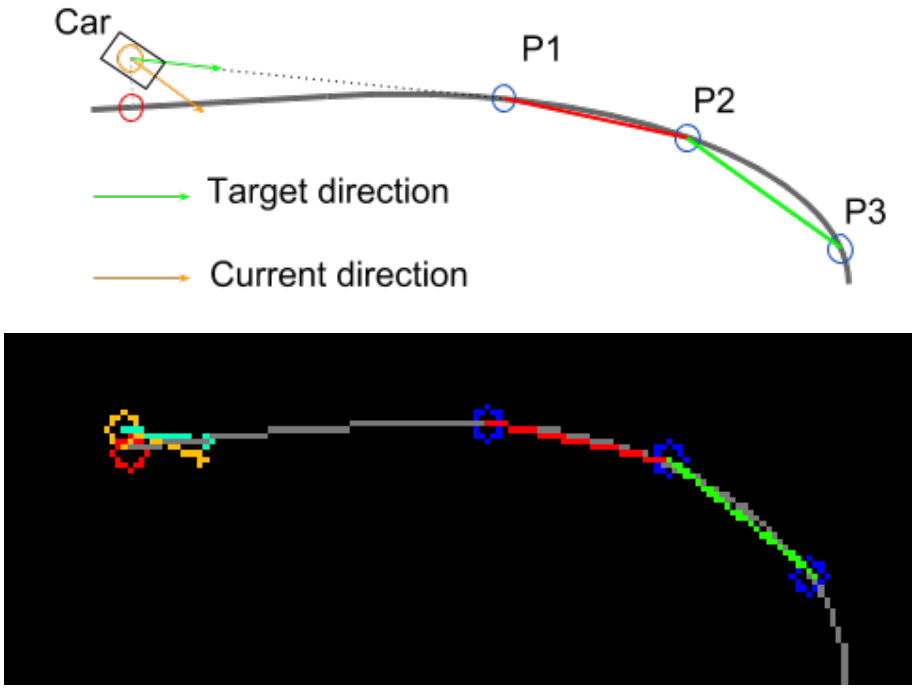


Figure 10: The two track segments are marked red and green. Their global angle corresponds to $\angle_1$ and $\angle_2$

The results of these algorithms are then chosen by which is the smallest, with the minimum value being chosen is 0.25, and then multiplied with the wanted speed.

$$S_r = S \cdot max(min(Sm_{Curve}, Sm_{\psi}), 0.25) \tag{24}$$

3.4 Indoor positioning

An important part in controlling the car is being able to get the position, direction and velocity of the car. To do this the information from the camera are fed into the computer and with the help of the LEDs mounted to the car it is possible to determine the position, direction and velocity

3.4.1 Position, direction, and identification

To establish the position, direction and identity of the cars, multiple operations has to be done. Initially the incoming camera frame is analysed. The frame is searched at every other row and column for a 3 by 3 squares of pixels where the middle pixel has a brightness over the set threshold and the surrounding pixels have a brightness of more than half the threshold. This method speeds up the analysis immensely since only half the data is analysed. The only downside is that the LEDs are required to be larger than 3 by 3 pixels on the camera frame.

All found squares are then iterated to merge closely located squares into larger circles where the middle of the circle is the average middle position of the squares. At this stage, if the parameters are properly set, circles originating from the centre of each LED should have appeared. This is then analysed in the same matter, but without merging and using a larger circle, to see which points belong to which cars. See figure 11 for visualisation. If a fourth identification dot is found within the car circle the car is identified as number two otherwise it is identified as number one.

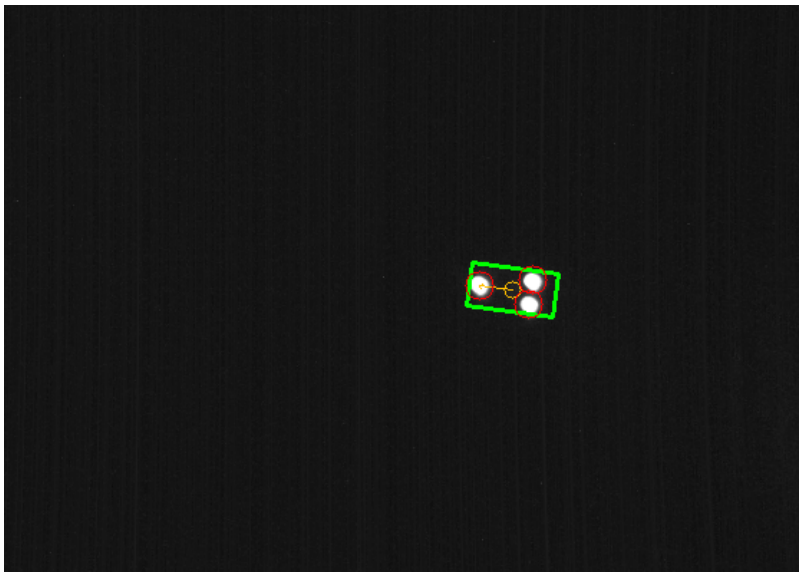


Figure 11: A car found by the image processing, with red circles around LEDs, orange circle and arrow for centre and direction of the car, and a green rectangle around it all, symbolising the car.

3.4.2 Velocity

The velocity is calculated by measuring time between two points at a distance of 5 centimetres or if the time passed between measurements is over 100 milliseconds. Only measuring between two well distanced points allows for greater precision since small noise in movement is mitigated by the larger movement. The fallback of a maximum time between measurements ensures that some velocity always gets reported, even at standstill.

3.4.3 Calibration

To utilise the images from the camera system in a relative matter (difference in distance). The system needs to be calibrated to ensure a difference in distance is the same physical distance at different parts of the image.

Each image is first rectified so that it is not bent at the edges. This ensure that distances can be calculated correctly. This is done using built-in ROS calibration tools.

To integrate the measured data with the model the physical distance a pixel represents is calculated. It is done through a calibration process with two IR LEDs at a known distance in meters. The calibration is done two times, one for the x-axis and one for the y-axis. The conversion factor is the distance in meters divided by the distance in pixels, for each axis respectively.

3.5 Creating the track

For the system to be able to know where the car should drive a Matlab script was created for generating a path. As one of the goals for this project was to be able to avoid obstacles an obstacle avoidance algorithm was also developed.

To generate a track for the car to follow a Matlab script was created. The user inputs a few points where they want car to drive, the script then calculates piece wise polynomials between these points forming a spline. The order of the points are taken into account so that the user can form almost any shape of track as they want. See figure 12.

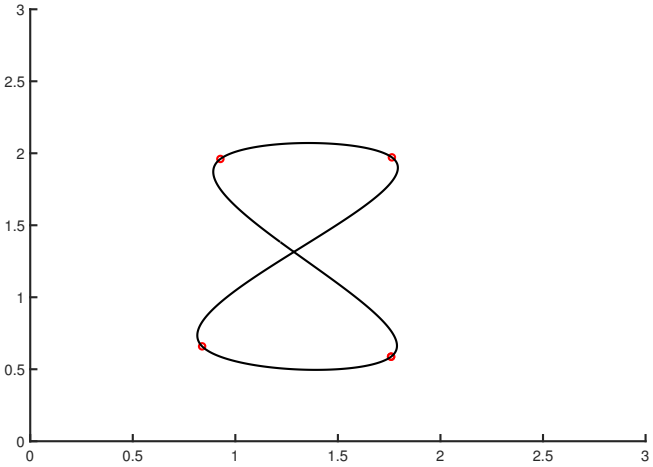


Figure 12: The program used to make a track for the car to follow

3.5.1 Obstacle avoidance

The obstacles being used in this project is other cars, therefore the positioning algorithms are used for detecting obstacles. When the position of an obstacle is known the edges of the obstacles are marked, then half the width of a car is added to the obstacle in every direction, the reason for this will be explained later in this section.

After the obstacle is identified and located the system checks if the planned path intersects the obstacle, if this is the case three point are added on the side closest to the planned path. Two point are added at the edges of the obstacle and a third in the middle of the first two points, see figure 13.

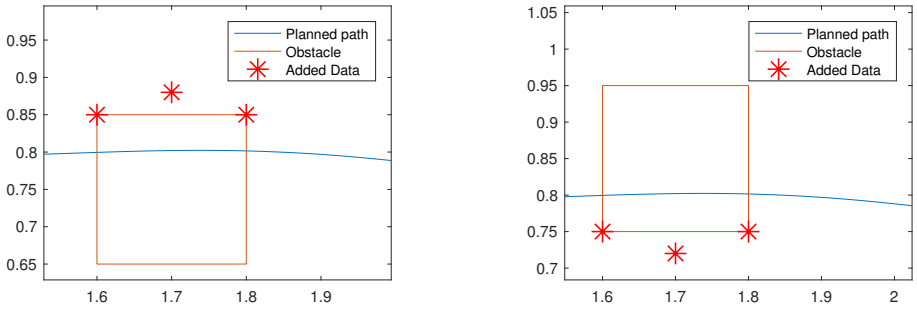


Figure 13: An example of how the system adds points for obstacle avoidance.

When these three points are marked, the two outermost point are connected to the planned path and to the middle point with straight lines, essentially forming a triangle around the obstacle, see figure 14. Even though the path is jagged and

might seem to make the car drive in a jagged way it actually turns out that the car drives in a rather smooth line around the obstacle. This is because the car cannot react as fast as it would require to drive in a triangular way, but also because the target point will have changed before the car arrives at the last target point resulting in a smooth path.

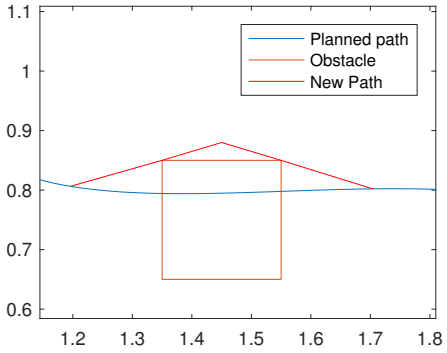


Figure 14: The planned path around an obstacle shown in red.

A problem with this method for obstacle avoidance is that it does not take the size of the car into consideration when planning a new path. So, even if the car drives along the new path it would still crash into the obstacle. This is solved by adding half the width of the car in every direction to an obstacle when feeding the position and size to the system. Then when planning the new path it is not necessary to take the size of the car into consideration, simplifying the programming.

Special case

There is a special case where only one of the corners of the obstacle is in the planned path, see figure 15. If the obstacle avoidance would work the same for these kinds of cases as it normally does it would result in an odd path.

The way the obstacle avoidance works in this case is checking where the planned path enters and exist the obstructed area, if the entrance and exit is on two sides that border each other, as it does in figure 15 then only one point on the obstacle is used for the new path resulting in a path like the one in figure 15.

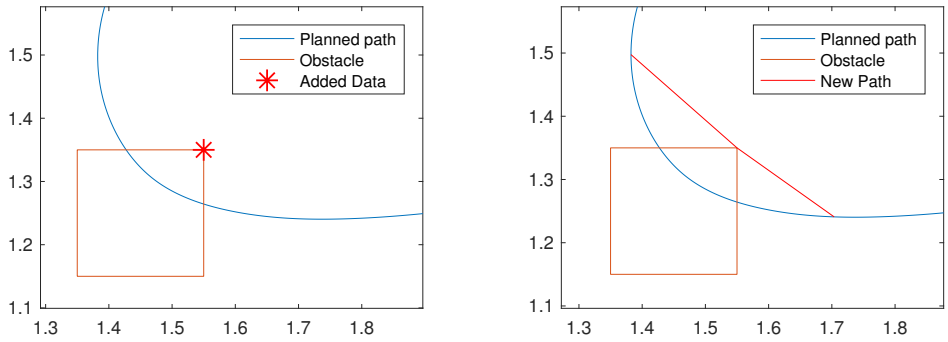


Figure 15: A situation that the obstacle avoidance needs to handle differently than most cases.

4

Results

Following sections contains both results and discussion, as well a reasoning about improvements and areas for further development. The results concern both real tested results and simulations, as well as comparisons between real and simulated.

4.1 Model

Numerous results from the model are obtained. Various parameters and how well they perform affect the final result. The controllers are designed from the linearised model, so it is important to check its accuracy. It is also of high interest to see how the simulation performs and how reliable it is.

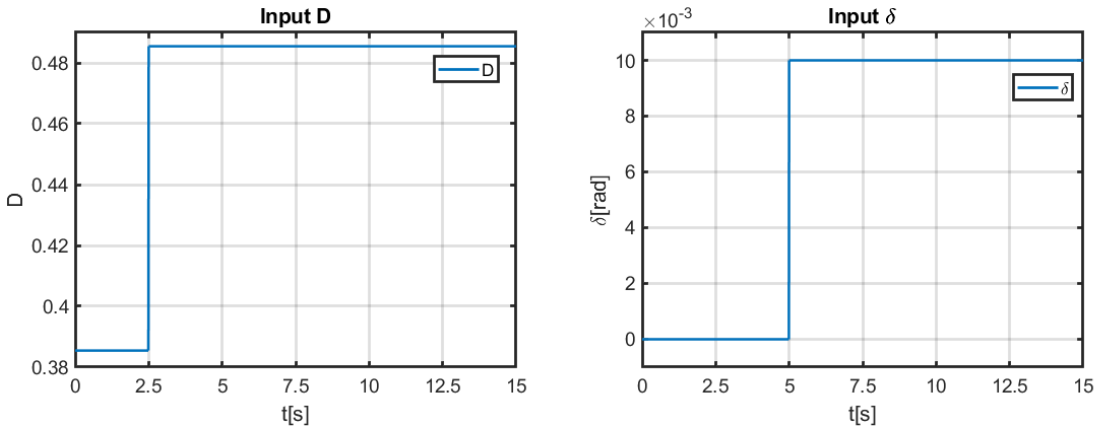
4.1.1 Simulation based validation of the linear model

Before using a linearised model to develop a controller, the behaviour of the linear model is compared to the nonlinear one, as to verify the reliability of the obtained values. To test this, output data from a linearised model is compared to output data from the nonlinear model, operating at the same work point. To further prove the validity of the data, a different input value is introduced after a certain time in the simulation. This shows how the model behaves close to set work point. This is an open loop test, meaning there is no feed back from the system. The several of the parameters used are placeholder values. The test is therefore not indicative of anything with concerns to the nonlinear model or its behaviour. As long as the parameters are the same for linear and nonlinear, the comparison is still valid.

The figures bellow are the results from the comparison. For this test, the model is linearised about the following work point:

v_x	2
v_y	0
ω	0

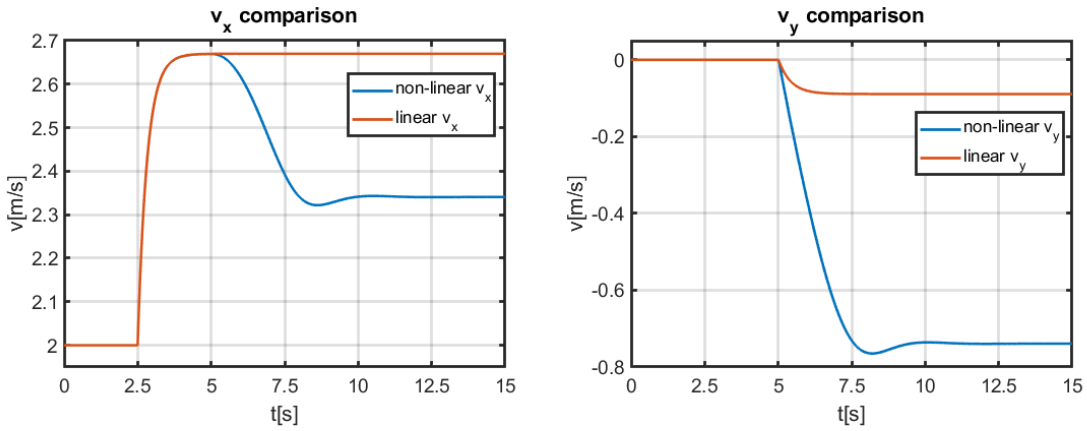
In practical terms, this work point describes the situation where the car is driving at a constant speed, in a straight line. The input values to the model are shown in figure 16.



(a) $InputsignalD$ **(b)** $Inputsignal\delta$

Figure 16: Input values for D and δ

Initially, the input values are $D = 0.39$ and $\delta = 0$, which means the car is driving straight at a constant speed. At $t = 2.5s$, D is incremented by 0.1, which results in an increase in v_x. At $t = 5s$, the input signal δ in incremented by 0.01, which results in a change in direction.



(a) State v_x **(b)** State v_y

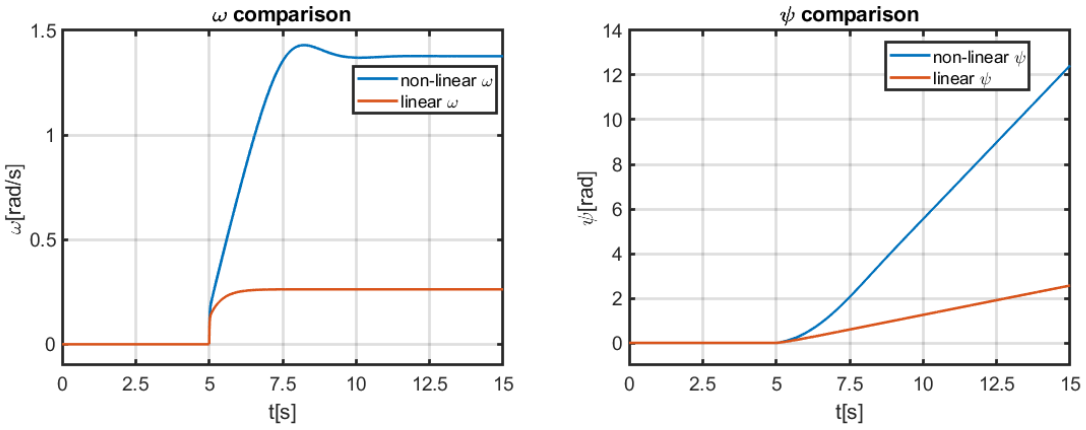
Figure 17: v_x and v_y comparison

Initially, as shown in figure 17a the longitudinal speed v_x is constant in both models. At $t = 2.5s$, when D is incremented, v_x increases in both models at the same rate, which is on par with the expected behaviour. Figure 17b shows that $v_y = 0$, which means the car has no lateral speed, which is the case when the car is driving straight. At $t = 5s$, when δ is incremented, the car starts cornering. Until this point in time, both the linear and nonlinear models behave identically.

When the car starts cornering, there is a decrease in nonlinear v_x. There is also a change in both nonlinear v_y and linear v_y. The nonlinear behaviour is accurate to how a car drives in reality, where longitudinal speed v_x decreases, and lateral speed v_y becomes non-zero. A negative v_y means that the car is performing a left turn.

The linear behaviour is no longer identical during this manoeuvre. Linear v_x is unchanged during cornering, while linear v_y is changed to become non zero. This is due to the linear model being linearized about a certain work point, where in this case $v_x = 2m/s$, which means the model will behave similarly, but not identically, to the nonlinear model when deviating from said work point. A decreased accuracy is therefore expected from the linear model.

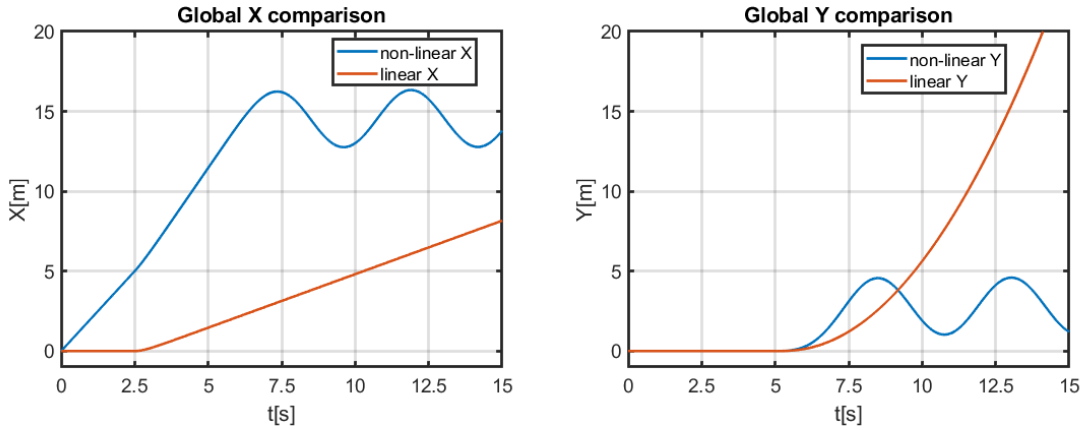
This is further proved when comparing the cars angular velocity ω and global angle ψ positions, as is shown in figure 18.



(a) State ω **(b)** Global state ψ

Figure 18: ω and ψ comparison

Here, the same pattern as in figure 17 reappears, where both models produce identical results until the car starts cornering at $t = 5s$.



(a) Global state X **(b)** Global state Y

Figure 19: Global X and Y comparison

Here, the linear X and Y are not accurate any time during this test. Even though Y seems to be accurate during the first 5 seconds, it is only so because the simulation was initiated with $\psi = 0$, meaning the car is driving straight in the X direction. It was believed that the linear global positions would be accurate while the linear model is operating about its work point. As shown in figure 19, however, points towards the contrary.

The reason for this large deviation is uncertain, but a possible cause could be a problem in the simulation algorithm. Due to the restricted time frame of this project, this issue was not further investigated. It has, however, been deemed unimportant for the intended goal, since the global positions are not used in the PID controller.

To conclude, the four states v_x, v_y, ω and ψ, behave similarly, but not identically, in both the linear and nonlinear models. This proves the need for multiple linear model to describe the nonlinear one with sufficient accuracy.

4.1.2 Parameters

Most of the physical parameters in the model are simply measured or calculated as described in the methods chapter 3.2.2.

Mass and geometry can be assumed to have about as much accuracy by-hand measurement with ordinary tools can achieve. The drag coefficient has some degree of unreliability. As described in section 3.2.2 the drag coefficient was assumed to be about the same as that of a full-scale Ferrari GTC 360, same model as the RC car is made to look like. This assumption has somewhat weak support because the shape is not guaranteed to be the exact same, and the material is vastly different. Ferrari also does not provide drag coefficient in the data specifications for this model, so the reference used is not the most reliable. Neither was a full aerodynamic model used,

m	Mass	183 g
L	Length	16.5 cm
W	Width	6.9 cm
c_g	Centre of gravity	9.25 cm
l_f	Distance from front to c_g	9.25 cm
l_r	Distance from read to c_g	7.25 cm
I_z	Moment of inertia	$7.3526 \cdot 10^{-5} kg \cdot m^2$
A	Frontal cross sectional area	0.2135 m^2
C_d	Drag coefficient	0.335

Table 2: RC car physical parameters

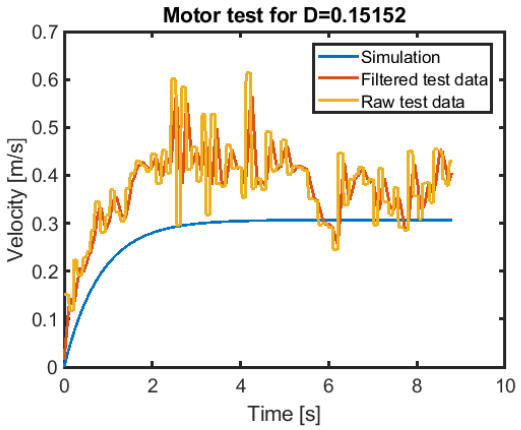
no lift forces or skin friction were taken into account. These weakness is accepted because the motor parameters can compensate to some degree.

Proceeding as detailed in section 3.2.2 the parameters for the motor are found to be the following:

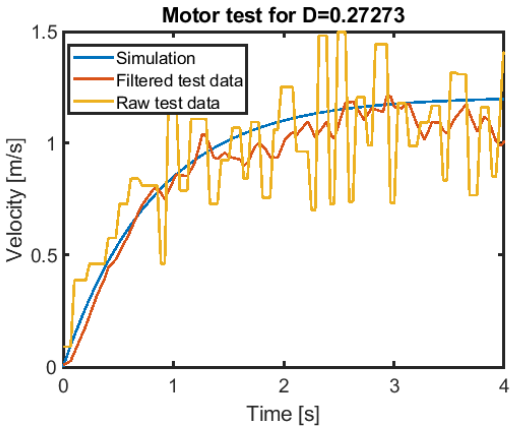
C_{m0}	1.6584
C_0	0.2226
C_1	0.1829

Table 3: Motor parameter values

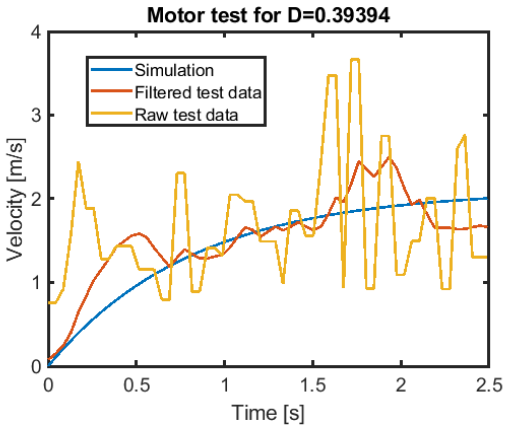
By running the simulation for the duty cycle percentage D corresponding to the different voltage inputs in the tests, the motor's behaviour can be compared in the following graphs



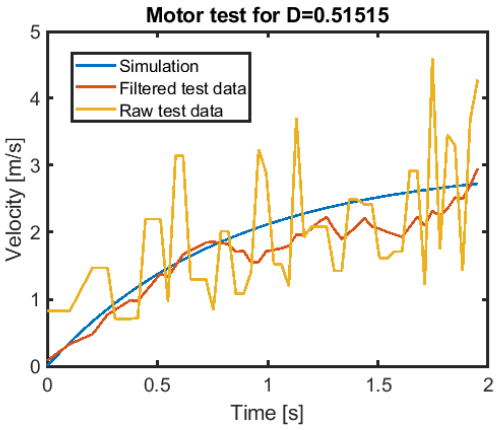
(a) Test with motor signal D=0.1515



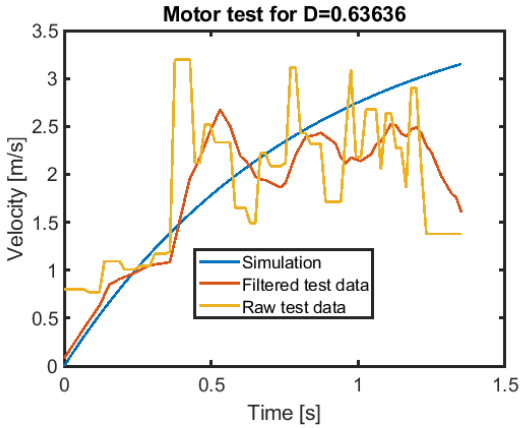
(b) Test for motor signal D=0.2727



(c) Test with motor signal D=0.3939



(d) Test for motor signal D=0.5152



(e) Test for motor signal D=0.6364

Figure 20: Motor tests

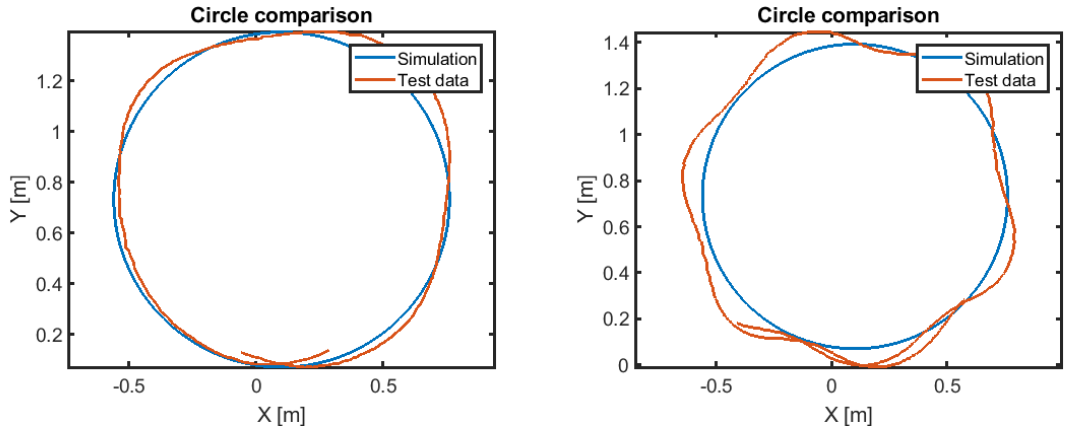
A number of observations can be made. The result for the slowest test as seen in figure 20a is poor. The middle tests presented in figures 20b, 20c, and 20d appear to be quite accurate. The fifth test as shown in 20e is decently accurate for the first second, the simulation continues to accelerate whereas the test data dips because it reached the end of the testing course.

The results for the motor parameters are sufficient, but not completely accurate. There are a number of errors that could be the cause for this. The most obvious cause is the erratic behaviour of the raw test data. Attempting to tune the motor parameters after the raw data would be unreasonable. Therefore the filtered data has to be used. The laboratory in which the tests where performed had space limitations, as can be noticed in how the tests time span decreases as the duty cycle increases. This meant that the higher speeds could not be tested for. The method for identifying the parameters also lacked any sort of time dependency. Motor temperature, and battery wear are also possible factors that could affect results but were not taken in to account.

The tyre parameters were determined as described in section 3.2.2 and found to be the following values:

d	1.16
c	1.96
b	1.44

Table 4: Tyre parameter values



(a) Test 1 **(b)** Test 2

Figure 21: Comparison between circular trajectory for simulation and RC car at different speed and radius

Figures 21a and 21b shows the trajectory for the simulation and RC car during a test. The RC car is set to follow a circular path with an empirical P-controller. The simulation runs in a circle with the mean control signals. For the first test the results are very accurate with some deviation from slipping. In the second test the results are decently accurate, but the trajectory of the car is wavy. The wavy path is a result of poor control, but the circle has about the same radius. The results indicate that the simulation's behaviour is accurate at least for circular trajectories. No tests for other slip and lateral movement were conducted because they are very hard to compare and repeat with the available tools.

4.2 Positioning and computer performance

The positioning and computer performance are critical for testing the systems in reality. If either one performs badly the data will be skewed and controllers will fail. In this project the systems performed adequately but not error-free and not all cases could be tested.

Positioning

The positioning of a car worked well, both the position and the direction could be established at 150Hz, although some calibration to match the light of the diodes had to be done. The calibration had to be redone as the battery of the LEDs discharged since the light intensity depend on the battery voltage.

When having two cars on the track, the positioning was not as satisfying. As the intensity of light on both cars have to be similar to be able to calibrate properly. The second car have one diode more and therefore the battery discharge faster and thus changing intensity faster than the other car. Thus the intensity is rarely the same, and it is therefore hard to detect both of the cars at the same time. In addition, the extra diode on the second was placed close to the others, putting even higher demands on the calibration for proper separation of dots. These problems made driving with two (or more) cars unfeasible since the second car would falsely identify as the first and thus skew data. As the focus of the project was elsewhere, these issues were never resolved.

Computations

As the computations for each frame delivered from the camera, including calculating new control signals, was done in sufficient pace, the system could in general keep up with the frame rate of 150 fps. As the behaviour of the car depended on the computer used, it seems reasonable that the delay between a frame arriving to new control signals sent out depends on the computational speed of the computer. This is reasonable as the system fully depends on reacting to the inputs it gets and a longer reaction time will impact the performance. Although this can be compensated for, it can not be removed unless some predictive control is used.

4.3 Track performance

The goal of the project was to develop stable controllers, for steering and speed, able to drive the car around the track at relatively high speed. The mathematical model of the car was used to make multiple PID controllers for steering. Speed control is achieved using an improvised algorithm that uses the curvature of the oncoming segments of the track to determine the controller output.

4.3.1 Steering control

With several linear models, multiple PID controllers were developed using the methods described in section 3.3.1. These controllers are only for steering. The obtained parameters were saved in a look-up table, and tested in the simulation. Figure 22 shows the simulation results of the car driving around a track with a constant duty cycle of $D = 0.4$.

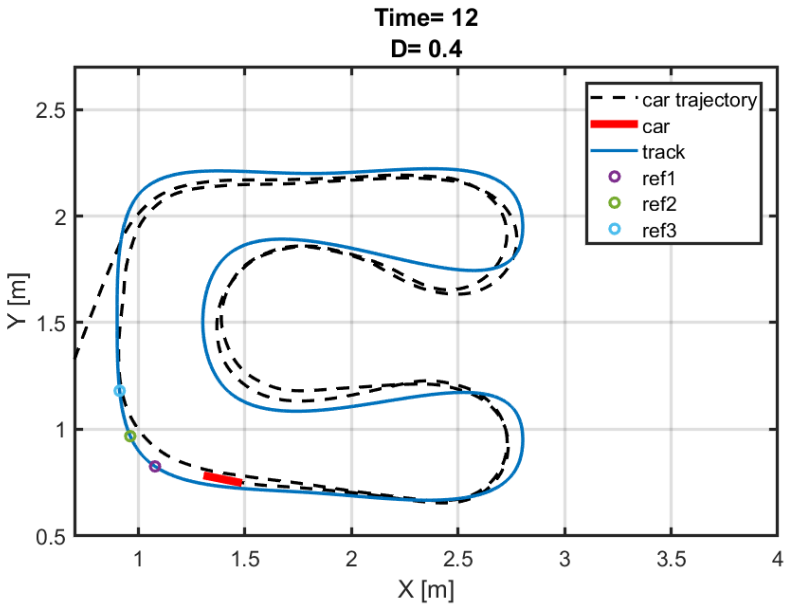


Figure 22: A simulation of the car driving with the acquired PID parameters

From this simulation, it is apparent that the car can steer around the track with a constant D. This result proves that the constructed PID controller, using the look-up table, is in fact capable of steering the car to follow the track with good stability.

To test the effect of the look-up table, an other test was conducted with one single PID controller. The simulation results are shown figure 23

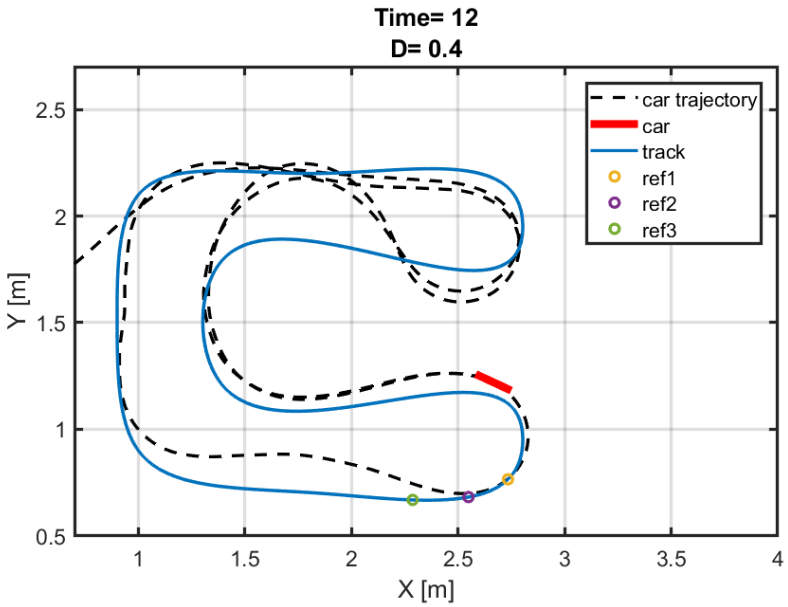


Figure 23: A simulation of the car driving with one single PID controller

As shown in figure 23, the single PID controller was able to steer the car around the track. However, the trajectory is not as smooth or close to the track as in Figure 22. These tests, as expected, prove the need for a look-up table to improve track performance.

4.3.2 Speed control

The simulation results showed that the steering controller works in theory. To improve overall track performance, however, the need for speed control is essential. This would allow the car to accelerate on straights and brake before corners, which improves lap times. This was implemented according to the methods in section 3.3.2.

As the project mainly focused on steering control, the PID controller for speed was tuned manually. The most effective laptimes measure occurred using $K_p = 0.52$, $K_i = 0.37$, $K_d = 0$. The distance from the cars position to the curve algorithms measuring points was manually set at 0.3m with 0.225m long segments. The weights were set at $W_\psi = 1.0$ and $W_{curve} = 3.2$.

Notably, the K_d parameter for the PID was set to zero. This was a result of noisy speed measurement data, causing false actions with an added derivative gain.

An observation here was that the curve algorithm's, S_{curve}, distance needed to be manually adjusted during actual racing to be fully tuned, as it was speed dependent. This could probably have improved speed handling significantly if tuned and

adjusted to the actual velocity of the car.

The S_ψ algorithm, which based speed on the angular error of the car, was mostly used to resume control over the car when slipping, and did so effectively.

The figure 24 clearly show the need to maintain high speeds when arriving at straight segments, and slowing down before entering curves.

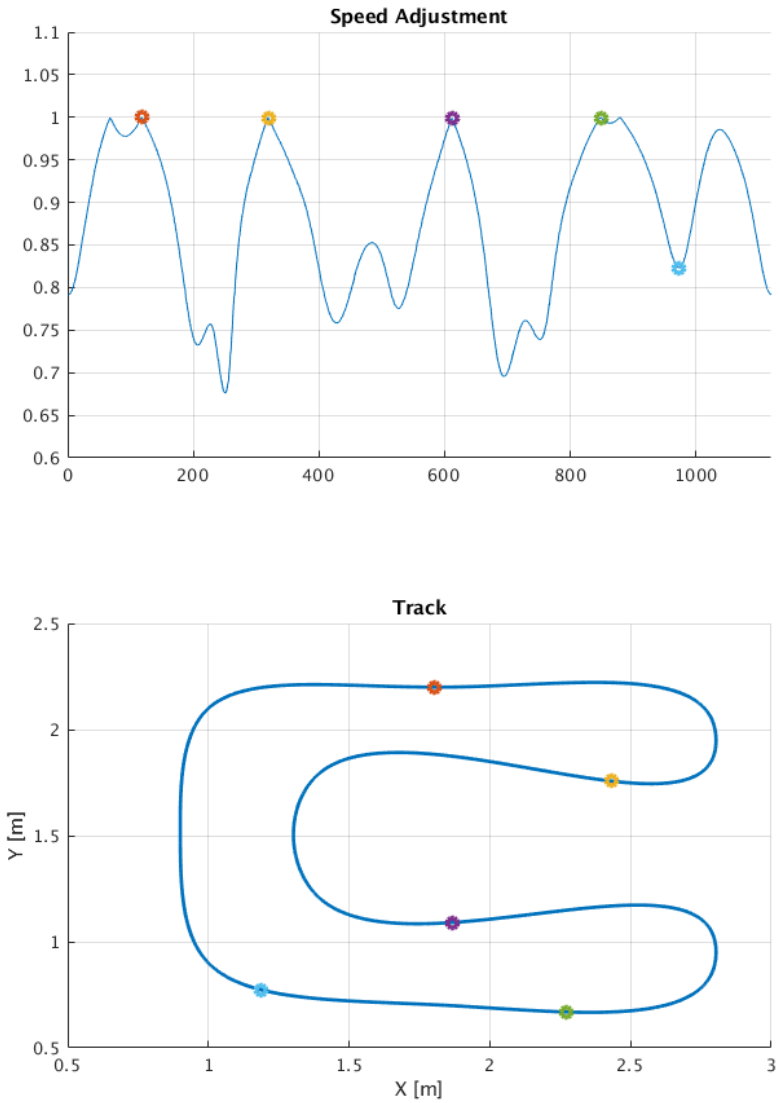


Figure 24: A comparison of the curve segment dependent adjustment of speed along the map. The speed adjustment value is to be read as a percentage of target speed in that instance.

4.3.3 Real-time implementation

Simulating the car driving around the track by using a look-up table based control method for steering control was a success. However the results of the real-time implementation for the same look-up table is somewhat inconclusive and contents several sources of error. These results are shown in figure 25 and figure 26.

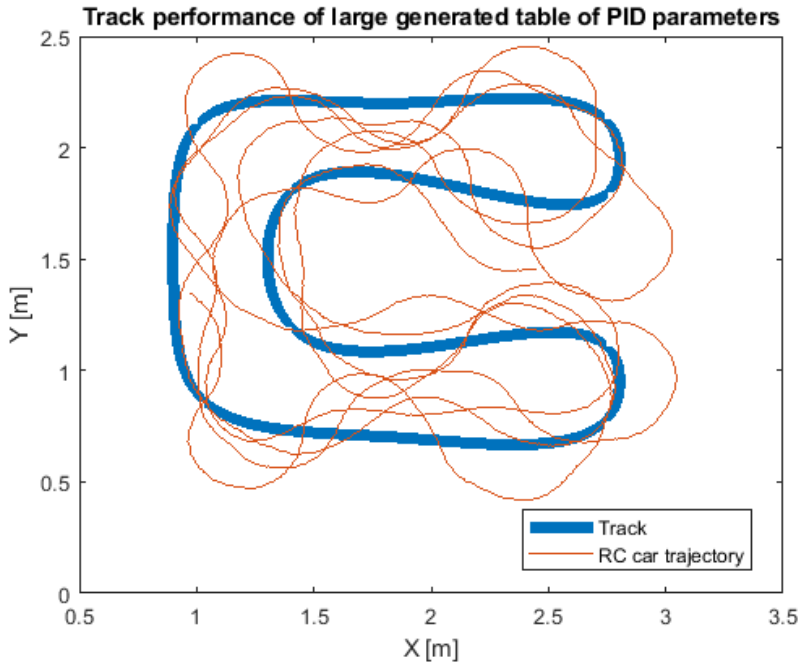
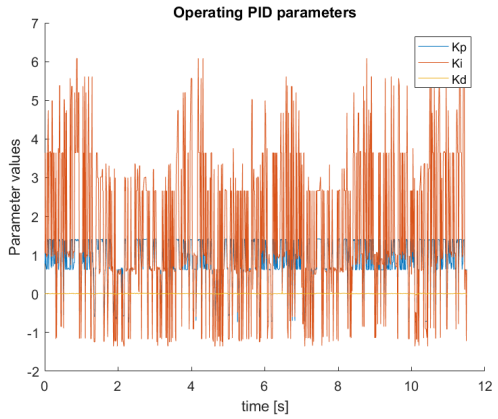


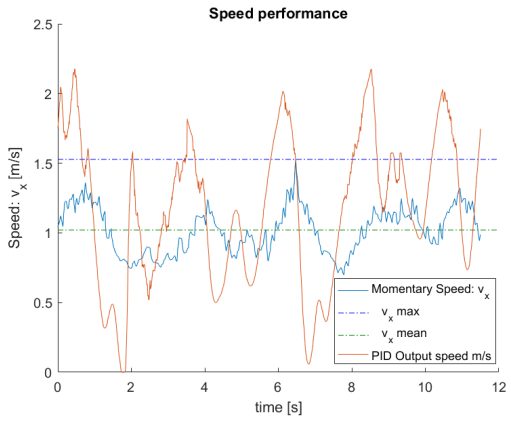
Figure 25: Recorded results of one car following a virtual line with the look-up table based PID controller. Observe that the car does manage to complete several laps.

Table 5: Results of speed performance for the big look-up table

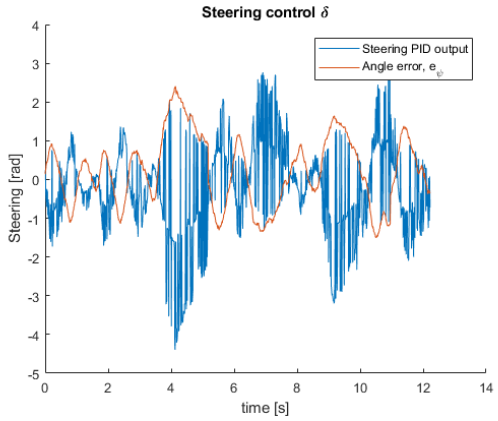
Mean velocity	1.021 m/s
Maximum velocity	1.527 m/s



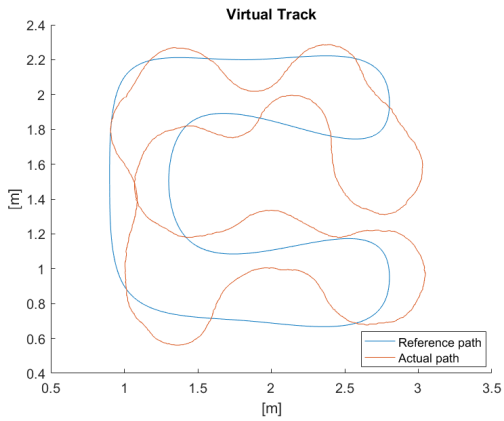
(a) PID parameters during one lap.



(b) v_x and control signal in [m/s].



(c) Control signal δ and error value of e_ψ



(d) The observed lap in (a)-(c)

Figure 26: The different control signals and system responses during one lap when controlled by a big look-up table. PID output speed is the signal sent from the PID controller to the car measured in m/s.

The controllers used in figure 25 and figure 26 is the speed controller from results in section 4.3.2, and the steering controller is the same as in figure 22, which is a look-up table consisted of 744 sets of PID parameters. When using a great amount of PID-parameters in this real-time implementation the result is not as successful. They differ greatly in the ability to follow a line. The real-time implementation of the large look-up table does manage to complete several laps, but the car oscillates significantly around the reference line. Due to overshoots in almost every attempt to turn. Since the speed controller only accelerates the car if the error value for direction (e_ψ) is moderate, it never succeeds to drive the RC car at any high velocity. Hence several sources of error are present in the real-time implementation causing the simulation and implementation results to differ.

Constructing another look-up table enabled satisfying results and the possibility to further investigate the potential sources of error. It was made by identifying error

sources when observing the systems behaviour with different PID controllers, thus providing unique constrains to create a new look-up table. Following are the observations. The system had a high sensibility towards the derivative gain, which approximately should not exceed 0.03. To minimise overshoot when steering, the ratio between the proportional- and integral gain could not be greater than approximately 1:2.5. The system was slower than earlier anticipated and the big look-up table containing small incremented steps of v_x and ω was unmanageable for the system. In figure 26a it is clear that the PID parameters change very many times during one lap. Thus, creating a delay affecting the reaction time and making the system unable to control the momentary scenario. When the car starts to react towards the new PID parameters it has already moved to a complete different scenario. Combined with to many PID parameters for similar v_x and ω the steering control is unable to keep the car on the path.

Given this, the work points was chosen to make sure it corresponded to the observed constrains for integral- and derivative gain. Resulting in 12 different sets of work points representing 12 linear models with great stability. Also representing less scenarios to prohibit the system from switching PID parameters before leaving the corresponding work points. To represent a slower system 0.3 radians is subtracted from the crossover frequency as input to the PID generating code in appendix 1, decreasing the raise time to reach a steady state output. As well, the speed controller was set to decelerate more before every turn. After these adjustments the results was a great success and the validity of the assumed error sources was their by strengthen. The smaller look-up table and the work points can be seen in appendix 2 and the result in figure 27 and figure 28 .

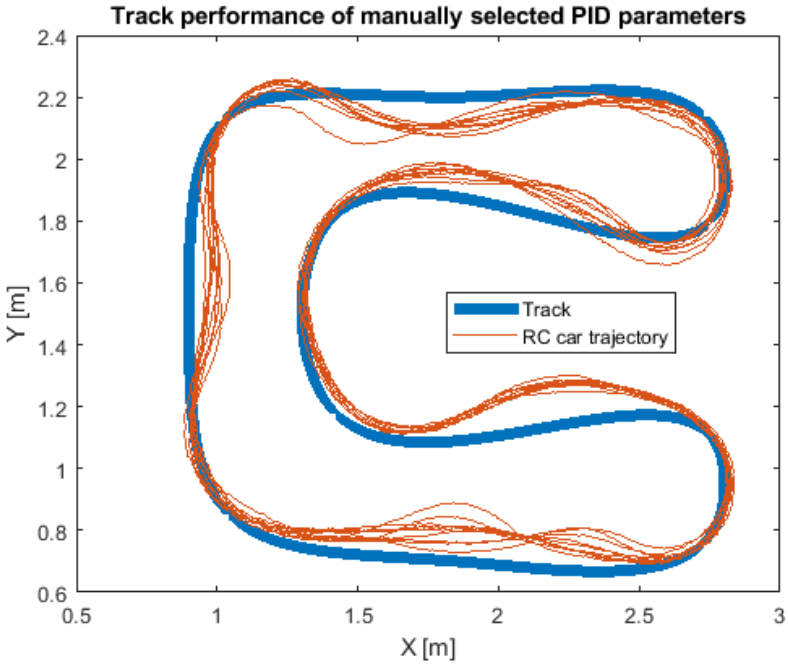
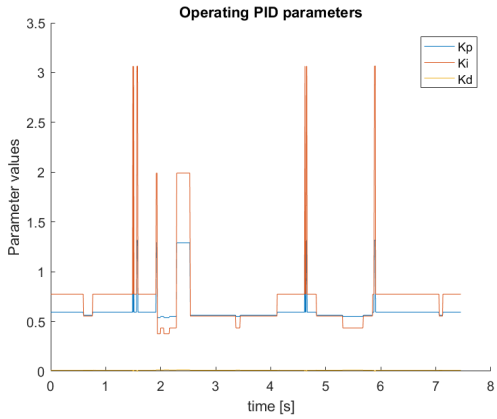


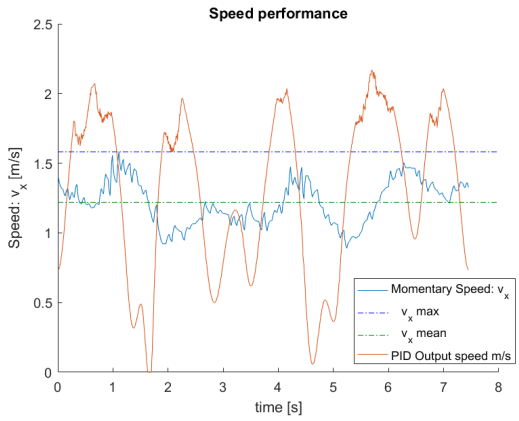
Figure 27: Recorded results of one car following a virtual line, with a smaller look-up table of only 12 PID controllers, by fine tuning which particular linear models are suitable for driving the RC car.

Table 6: Results of speed performance for the smaller look-up table.

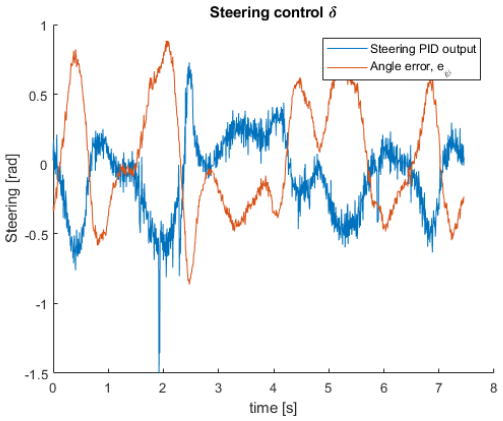
Mean velocity	1.219 m/s
Maximum velocity	1.581 m/s



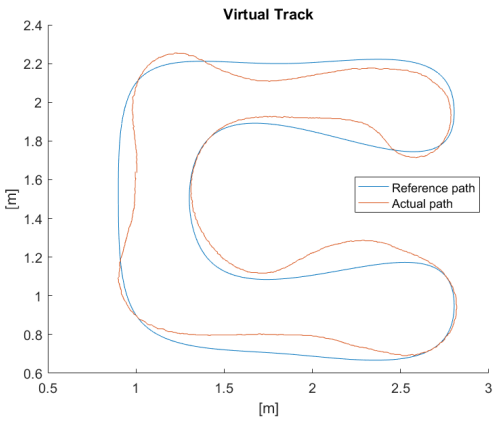
(a) PID parameters during one lap.



(b) v_x and control signal in m/s.



(c) Control signal δ and error value e_ψ



(d) The observed lap in (a)-(c)

Figure 28: The different control signals and system responses during one lap when the system is controlled by a small Look-up table of 12 PID controllers.

The most obvious reason for this great improvement in steering control is best visualised when comparing figure 28a to figure 26a and figure 28b to figure 26b. Now the system is given the chance to operate according to the momentary situation, causing less oscillation. If switching PID parameters causes one of the most significant sources of error, it is possible to assume that using only one PID controller could improve the performance even more. However, this is not correct. Similar to the results when simulating in Simulink, one single PID controller could not operate the system better than a look-up table. See results from using a single set of PID parameters in figure 29. It is able to steer the RC car around the virtual track but the trajectory is much worse compared to figure 27.

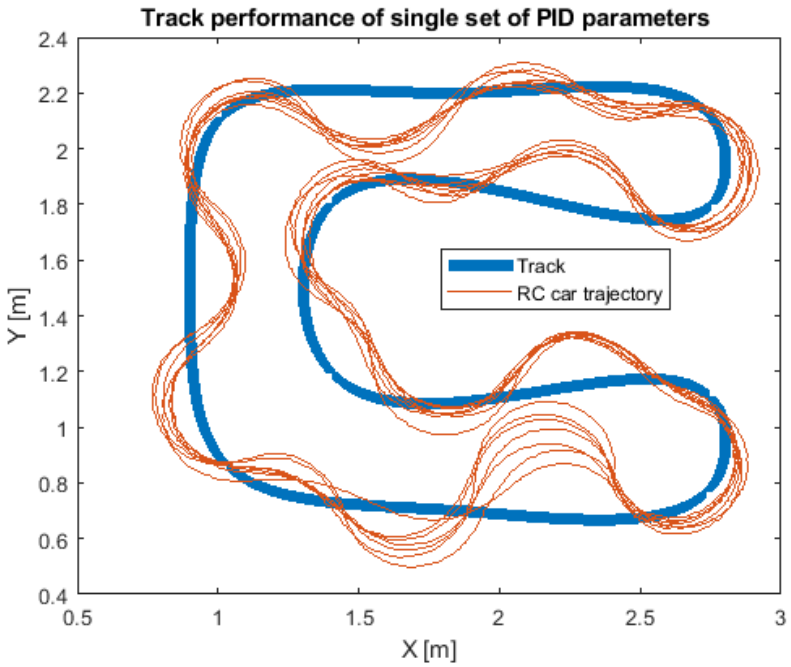
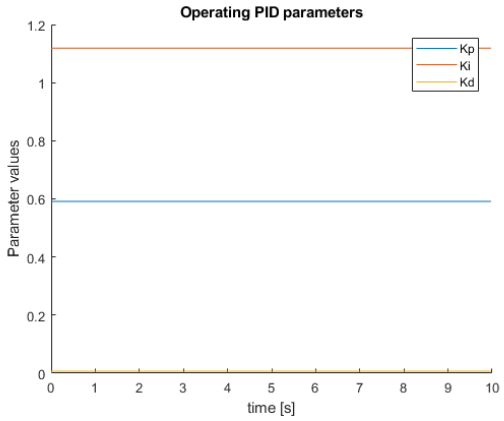
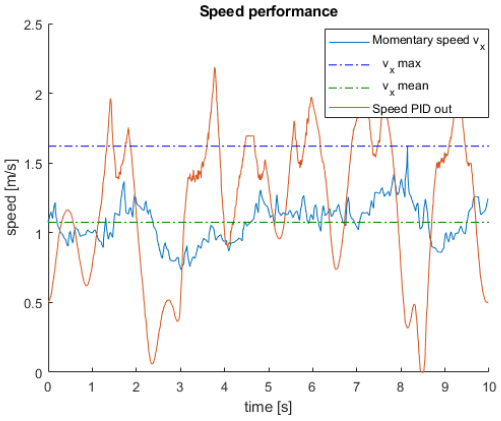


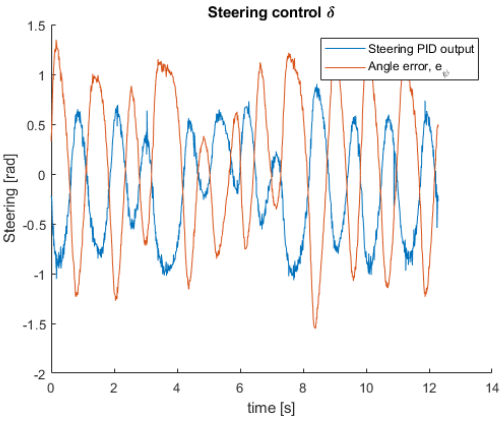
Figure 29: Real-time test result using only one single PID controller with parameters Kp = 0.5912 Ki = 1.119 Kd = 0.00713



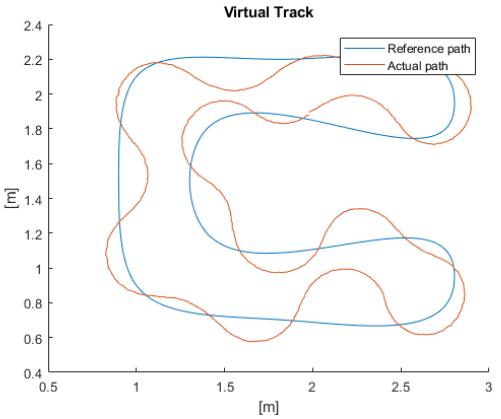
(a) PID parameters during one lap.



(b) v_x and control signal in m/s.



(c) Unsaturated control signal δ and error value e_ψ



(d) The observed lap in (a)-(c)

Figure 30: The different control signals and system responses during one lap when the system is controlled by one PID controller with the same parameters as in figure 29.

It is very clear from 7 that the small look-up table had much better performance than the big look-up table and the single PID controller. The lap times are roughly 2-3 seconds faster consistently. The single PID controller does reach a higher maximum velocity than the small look-up table, but a lower mean. The higher maximum does not mean much in terms of race performance since the overshoots it causes results in slower lap times.

	Big look-up table	Small look-up table	Single PID
Lap 1 [s]:	10.7961	7.4290	9.6511
Lap 2 [s]:	10.8713	7.5631	9.7738
Lap 3 [s]:	10.2807	7.4396	9.0610
Max velocity [m/s]:	1.527	1.581	1.62
Mean velocity [m/s]:	1.021	1.219	1.075

Table 7: Performance data for the tests

In the introduction it was stated that this project could contribute to understanding difficulties in ongoing research about autonomous racing cars. Since the RC cars do not reach any high speed, the amount of correlations between this project and real scale autonomous racing are not many. However one of this system's sources of error could be counteracted with help of a more predictive control method. The controller seem to improve their accuracy when given the chance to look ahead. Which could be the case even in real scale. When an autonomous racing car drives progressively faster it would need to look increasingly further ahead. which could be problematic since the further is hard to predict. This is most likely an area in need of research to develop secure traffic flows in the further autonomous traffic.

4.3.4 Obstacle avoidance

Unfortunately the obstacle avoidance was not tested due to the tracking system having problems separating the cars when they got close to each other. The algorithm was however tested in simulations were made using Simulink, giving an indication of how well it would work in reality. Results from the simulations are shown in figure 31. The car comes from the bottom of the picture and drives in the positive Y-direction.

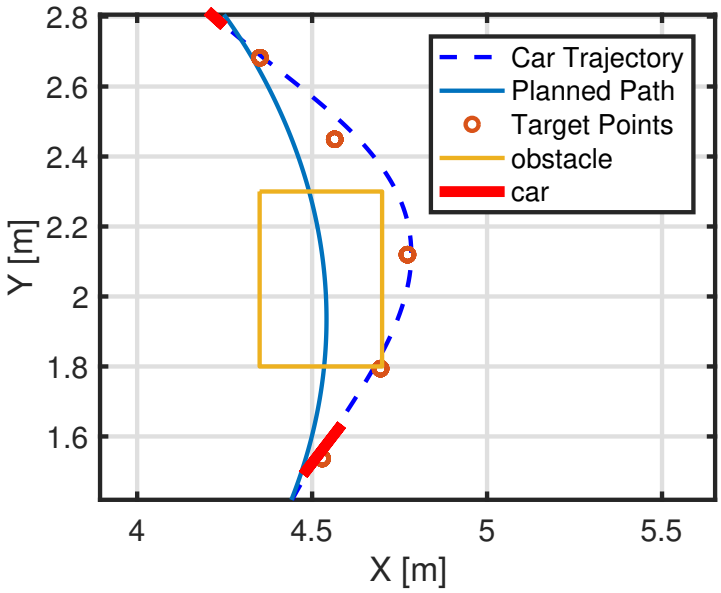


Figure 31: A simulation of the obstacle avoidance. The red box marked as the car is not to scale.

When looking at figure 31, it does look like the car crashes right into the obstacle. But this is not necessarily the case since half of the width of the car is added in every direction of an obstacle, as explained in section 3.5.1. Even though trajectory of the car is a bit close to the obstacle in the first corner, it probably wont crash but even if it doesn't, the margin will be very small. This can be solved by adding more safety margin to the area that is marked as obstructed.

As shown in figure 31 the track around the obstacle seems to be shifted a bit behind the obstacle. Possible improvements to this could be to shift the obstacle avoidance track back, before the obstacle, resulting in a better obstacle avoidance. The method used could also be improved from the ground up since this method is a reworked version of an earlier idea, partly to save computing power.

4.3.5 Areas of improvements and error sources

Identified error sources:

- Incorrect ratios between K_i and K_p.
- The big look-up table contained PID parameters constructed from unrealistic work points.
- The slow reaction time of the system causing the controller to not use the right PID parameters for the right situation.
- No anti-windup in the PID implementation
- Not using a MIMO supported controller method is most likely causing loss of information that could have tuned the controllers further.

Whose effects could be decreased by following improvements:

The linear state space model is not a good enough representation of the RC car. Since the linear controllers are created after the insufficient model, they could not handle the real situation accurately. As seen in the simulation results in figure 22 and results of the real time implementation in figure 25, the controllers are well adapted to the model but not to the RC cars. Nevertheless, it was possible to create a good look-up table for the RC car by observing the error sources of the system. Several of them, for example sometime unrealistic ratio between PID parameters are a result of an insufficient linear model. It is also important to note that using work points representing only stable models and realistic scenarios complicates the routine of generating more than 700 PID parameters. Overlooking the work points is necessary with this method.

Reaction time of the system As mentioned, the system switches PID parameters to quick compared to the reaction time of the car.

The image sent out from the camera is a matrix of pixels and the position of each LED is calculated simply by using the brightest pixel that is showing the LED. The calculated position of each LED is therefore rarely matching the actual exact position. This impacts the readings of the velocity, as the calculated position jumps between discrete values. The speed reading may therefore oscillate, as the system detects movements that are either smaller or larger than they actually are. Since this happens for each LED, and position of the car is calculated by using at least three, this is made even worse. Therefore, the velocity data was highly unreliable.

Anti-windup design, or rather lack of, is a weakness in the implementation that is important to discuss. Anti-windup is necessary in any system with integral gain and a limited actuator, to avoid signal build-up when the actuator is saturated. A simple clamping algorithm was implemented to constrict the integral gain when the steering was saturated, and other options were not considered.

Predictive control methods could improve how the system chooses PID parameters, if the systems speed could not be increased. Obviously many of those problems could be solved by using more advanced control methods, like MPC which also can handle MIMO systems. To manage it with linear methods, a more advanced predictive method could be develop to predict curves and

Material, different RC cars and track surface could improve the stability of the system. Using bigger RC cars with better motor parameters could simplify the linear model. The track surface of this project was a rubber carpet but with a rather slippery surface whenever dust accumulated. Using fabric or a more roughened rubber carpet could help the RC cars drive faster, since it often slipped when accelerating.

Obstacle Avoidance, there is some problems here, the first and most obvious, it was never implemented and therefore never properly tested. To implement this the positioning algorithm needs improvement so that it does not lose track of the cars. The margin to the area marked as obstructed is a bit small, this can be fixed by adjusting the method or just by adding more safety margin to the obstruced area.

Summary of track performance

After considering the result from simulation in Matlab and the real-time implementation, it is obvious that a look-up table is needed to control a RC car when using linear control methods. Since the car is never operating in a single steady state when following a path, several linear models is necessary to mimic this nonlinear scenario. However, the look-up table failed to make the RC car follow the line well and since only one PID controller also failed to do so, the best compromise in this project was to create a smaller look-up table by specifically choosing linear models corresponding to observed sources of error.

Because of problems implementing obstacle avoidance, no tests were conducted, but it was simulated and the RC car did avoid an obstacle which shows that the concept works. However, it needs more work.

There are room for improvements of the system since the mean velocity is only 24.3% of the cars modelled maximum on the track used in this project. The velocity is track dependant, and since this track is rather difficult with many sharp turns a velocity of this sort is not unexpected without methods of for drifting or better grip. However, creating a learning environment for autonomous RC racing is definitely achieved. From this platform a few hypotheses can be made about the future need of predictability in traffic.

5

Conclusion

Developing a system to autonomously drive two small RC cars around a virtual track, thus creating a platform for learning about autonomous racing, was the main goal. This was partially achieved.

It is possible to drive one car around the virtual track with satisfying accuracy when following a line but with somewhat inconclusive results regarding speed. The steering control was an attempt to simulate a nonlinear system with multiple linear ones and the simulation results was a great success. The performance of a large look-up table over a wide span of work points was very poor compared to the significantly better results when using a smaller table designed from choice work points, shown in the lap times and speed table 7. Since the linear model does not represent the real RC car well enough, the implementation of the controllers needed extra constraints to fulfil the goals.

The system is not able to avoid obstacles or drive two cars simultaneous in the real-time implementation, but the basis for development is done in simulation with promising results.

This project was a great learning platform for autonomous cars in general, and autonomous racing specifically. It has provided many insights on the limitations and the possibilities on racing with autonomous cars. To further develop this system another linear model is necessary, as well a more advanced predictive algorithm is necessary to help the linear controllers predict the next scenario on the track, thus increasing reaction time of the system.

Suggestions for future research

Due to difficulties grasping all the challenges when planning the project, the speed controllers were not optimised. It also became apparent that the speed control affected the car performance more than first estimated. This resulted in a control system that while working, could be improved by implementing MIMO control. Problems with implementing the look-up table and the accuracy of the linear model also affected lap times negatively.

Since a nonlinear system is being controlled with linear control strategies, a lot more emphasis has to be put on developing methods for state estimation, in other words looking ahead to see what is most likely to happen next. How the next curve will look and what will happen after that curve, this would likely improve the speed control to a greater degree than the steering control.

Bibliography

[1] K. Bimbraw. Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology. In *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 01, pages 191–198, July 2015.

[2] Volvo cars. *Drive me trial in Gothenburg*. [Online] Available: `https://www.volvocars.com/intl/buy/explore/intellisafe/autonomous-driving`, [Accessed: 2018-04-16].

[3] P. Fernandes and U. Nunes. Platooning with ivc-enabled autonomous vehicles: Strategies to mitigate communication delays, improve safety and traffic flow. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):91–106, March 2012.

[4] David Shepardson. Tesla says crashed vehicle had been on autopilot prior to accident. [Online] Available: `https://www.reuters.com/article/us-tesla-crash/tesla-says-crashed-vehicle-had-been-on-autopilot-prior-to-accident-idUSKBN1H7023`, [Accessed: 2018-04-20].

[5] Hyundai. *How racing car technology influenced the future of commercial vehicles*. [Online] Available: `https://www.hyundai.news/eu/technology/how-racing-car-technology-influenced-the-future-of-commercial-vehicles/`, [Accessed: 2018-04-16].

[6] Stanford University dynamic design lab. Shelley, stanford's autonomous car, at thunderhill raceway. [Video: Online] Available: `https://news.stanford.edu/news/2012/august/videos/1117.html`, [Accessed: 2018-04-26].

[7] M.Kritayakirana. *Autonomous vehicle control at the limits of handling*. Ph.d dissertation, Stanford University, June, 2012.

[8] U. Rosolia, A. Carvalho, and F. Borrelli. Autonomous racing using learning model predictive control. In *Proceedings 2017 IFAC World Congress*, 2017.

[9] F. Zhang, J. Gonzales, K. Li, and F. Borrelli. Autonomous drift cornering with mixed open-loop and closed-loop control. In *Proceedings IFAC World Congress*, 2017.

[10] MPC Lab @ UC Berkely. Automotive driving. [Online] Available: `http://www.mpc.berkeley.edu/research/advanced-vehicle-dynamic-control`, [Accessed: 2018-04-26].

[11] Swiss Federal Institute of Technology Zürich (ETH). *Autonomous RC Racing*. [Online] Available: `http://control.ee.ethz.ch/~racing/`, [Accessed: 2018-01-05].

[12] Uppsala University. *Camera-based autonomous racing system*. [Online] Available: `http://cars.it.uu.se/`, [Accessed: 2018-04-20].

[13] J. Ek et a. Advanced vehicle control systems. Chalmers University of Technology, 2015.

[14] E. Velenis, E. Frazzoli, and P. Tsiotras. On steady-state cornering equilibria for wheeled vehicles with drift. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 3545–3550, Dec 2009.

[15] F. White. *Fluid Mechanics*, chapter 7. McGraw Hill, 7 edition, 2011.

[16] J. Liljestrand. State estimation of RC cars for the purpose of drift control. Master's thesis, Linköping University: Faculty of Science and Engineering, Linköping, 2011.

[17] H. Pacejka. *Tire and Vehicle Dynamics*, chapter 3&4. Elsevier Science, 2012.

[18] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Applications and Methods*, 36(5):628–647, 9 2015.

[19] Massachusetts Institute of Technology Open Courseware. Feedback control systems. [Online] Available: `https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-30-feedback-control-systems-fall-2010/lecture-notes/MIT16_30F10_lec05.pdf`, Accessed : 2018-05-11.

[20] K. J. Åström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008. Available online at `http://press.princeton.edu/titles/8701.html`.

[21] B.Lennartsson. *Reglerteknikens grunder*, chapter 7&8. Studentlitteratur, 4:9 edition, 2002.

[22] Magnus Nilsson Björn Fridholm, Torsten Wik. Kalman filter for adaptive learning of look-up tables with application to automotive battery resistance estimation. *Control Engineering Practice*, 48:78–86, 2016.

[23] B.Lennartson B.Kristiansson. Robust tuning of pi and pid controllers: using derivative action despite sensor noise. *IEEE Control Systems*, 26(1):55–69, 2 2006.

[24] Kyosho. *Kyosho Mini Z MR03 Sports 2 Ferrari 360 GTC*. [Online] Avail-

able: https://goteborg.reservdelsrc.com/product/kyosho-miniz-mr03-sports-2-ferrari-360-gtc, [Accessed: 2018-02-02].

[25] FLIR (EU). Flea3 1.3 mp mono usb3 vision (vita 1300). [Online] Available: https://eu.ptgrey.com/flea3-13-mp-mono-usb3-vision-vita-1300-2-eu, [Accessed: 2018-02-02].

[26] M. Sakakibara et al. A back-illuminated global-shutter cmos image sensor with pixel-parallel 14b subthreshold adc. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pages 80–82, Feb 2018.

[27] P. Jansson R. Grahn. *Mekanik*, pages 505–509,517–521. Studentlitteratur, 3 edition, 2013.

[28] Heriot Watt MSc Mechanical Engineer Educator Dimitrios Christoloukas. Racing Cars Technology - Aerodynamics. [Online] Available: http://tech-racingcars.wikidot.com/aerodynamics, [Accessed: 2018-05-14].

[29] J. Katz. *Race Car Aerodynamic - Designing for Speed*, chapter Appendix. Bentley Publishers, 2 edition, 1995.

[30] Youbin Peng, D. Vrancic, and R. Hanus. Anti-windup, bumpless, and conditioned transfer techniques for pid controllers. *IEEE Control Systems*, 16(4), Aug 1996.

A

Appendix 1

A.1 Matlab code

A.1.1

These two following functions are from the book *Reglerteknikens grunder* by Bengt Lennartsson. They contain minor changes to be compatible with present linear model.

PI_{opt}

```
function [F,wc,phm,Ki,MS,MT] = pi_opt(G,wc_min,wc_max,
    phm_min,phm_max);
%Searching for optimal PI regulator

Nwc=20; Nphm=10;
wc=[wc_min:(wc_max-wc_min)/(Nwc-1):wc_max];
phm=[phm_min:(phm_max-phm_min)/(Nphm-1):phm_max];
[F,Ki,Ti] = pi_wc_phm(G,wc,phm);
MS=norm(feedback(1,G*F),inf,1e-3);
MT=norm(feedback(G*F,1),inf,1e-3);
stab=norm(feedback(G*F,1));
[Ki,iopt]=max(Ki'.*(MS<=1.7).*(MT<=1.3).*(stab<inf)); %MT
    =1.3
F=F(:,:,iopt); MS=MS(iopt); MT=MT(iopt);
[Am,phm,wpi,wc]=margin(G*F);
end
```

PI_{wc-phm}

```
function [ F,Ki,Ti ] = pi_wc_phm(G,wc,phm);
%Creating multiple pid regulators
```

```
3
4  Nwc=length(wc); Nphm=length(phm);
5  for k=1:Nphm
6      for i=1:Nwc
7          j=i+(k-1)*Nwc;
8          [gain_G,phase_G]=bode(G,wc(i));
9          %[gain_G,phase_G]=bode(NUM,DEN,wc(i));
10         %gain_G = [1 1 0 0 0 0]*gain_G.';
11         %phase_G = [1 1 0 0 0 0]*phase_G.';
12         %[~,indx] = min(phase_G-wc(i));
13         phase_F=-180+phm(k)-phase_G;
14         Ti(j)=cot(-phase_F*pi/180)/wc(i);
15         Ki(j)=wc(i)/(gain_G*sqrt(1+(Ti(j)*wc(i))^2));
16         F(:,:,j)=tf([Ki(j)*Ti(j) Ki(j)],[1 0]);
17         %create for PIDparameters...
18         %.
19         %.
20     end
21 end
```

[21]

A.1.2

A.1.3 PID Generator loop

psi_siso is the name of the SISO-linear state space model used in this report.

```
for i=1:5
    nr=strcat('sys',num2str(i)); %Psi_siso

    [NUM_Delta,DEN_Delta]=ss2tf(s.(nr).A,s.(nr).B,s.(nr).C,s
        .(nr).D,1); %Input Delta
     psi_siso=tf(NUM_Delta(3,:),DEN_Delta);

     [Gpsi,Pm_psi, Wcg_psi,Wcp_psi]= margin(psi_siso);
     PmF= [30 70];

    [F,wc,phm,Ki,MS,MT]=pi_opt(psi_siso,0.7*Wcp_psi,1.3*
        Wcp_psi,PmF(1),PmF(2));
    % Thinner intervall
    [F_opt,wc_opt,phm_opt,Ki_opt,MS_opt,MT_opt]=pi_opt(
        psi_siso,0.9*wc,1.1*wc,phm-2,phm+2);

    [Gm,Pm_feedback,Wcg,Wcp] = margin(psi_siso*F_opt);
    if(Ki==0)
        Pm_feedback=50;
    end

    opts = pidtuneOptions('PhaseMargin',Pm_feedback); %'
        DesignFocus','reference-tracking');
    [C info] = pidtune(psi_siso, 'PID',wc_opt, opts);
    [Kp Ki_pid Kd] = piddata(C);


    PID.(nr)= [Kp Ki_pid Kd wp(i,4) wp(i,6)];
end
```

B

Appendix 2

Here are the work points and PID parameters presented which are used to produce the results in figure 27 and figure 28. They produce a very successful result for the RC cars when following a virtual track.

Work points

Work points						
States:	X	Y	ψ	v_x	v_y	ω
wp1	0	0	0.1	0.1	0	0.3
wp2	0	0	0.1	0.6	0	0.8
wp3	0	0	0.1	0.9	0	0.3
wp4	0	0	0.1	0.9	0	0.8
wp5	0	0	0.1	1	0	0.8
wp6	0	0	0.1	1.2	0	0.8
wp7	0	0	0.1	1.3	0	3
wp8	0	0	0.1	1.3	0	0.8
wp9	0	0	0.1	1.5	0	0.8
wp10	0	0	0.8	1.5	0	0.3
wp11	0	0	0.8	1.7	0	0.8
wp12	0	0	0.1	1.7	0	3

Look-up table

PID parameters		
K_p	K_i	K_d
0.00013009	2.6601e-09	0
1.2266	1.212	0.020638
1.2892	1.9909	0.0013805
0.53992	0.37601	0.013611
0.54906	0.43472	0.012175
0.56309	0.55411	0.0099844
0.59208	0.77443	0.010641
1.3148	3.0652	0
0.57026	0.66539	0
0.570356	0.64082	0
0.57444	0.73399	0
0.59122	1.1186	0.0071305