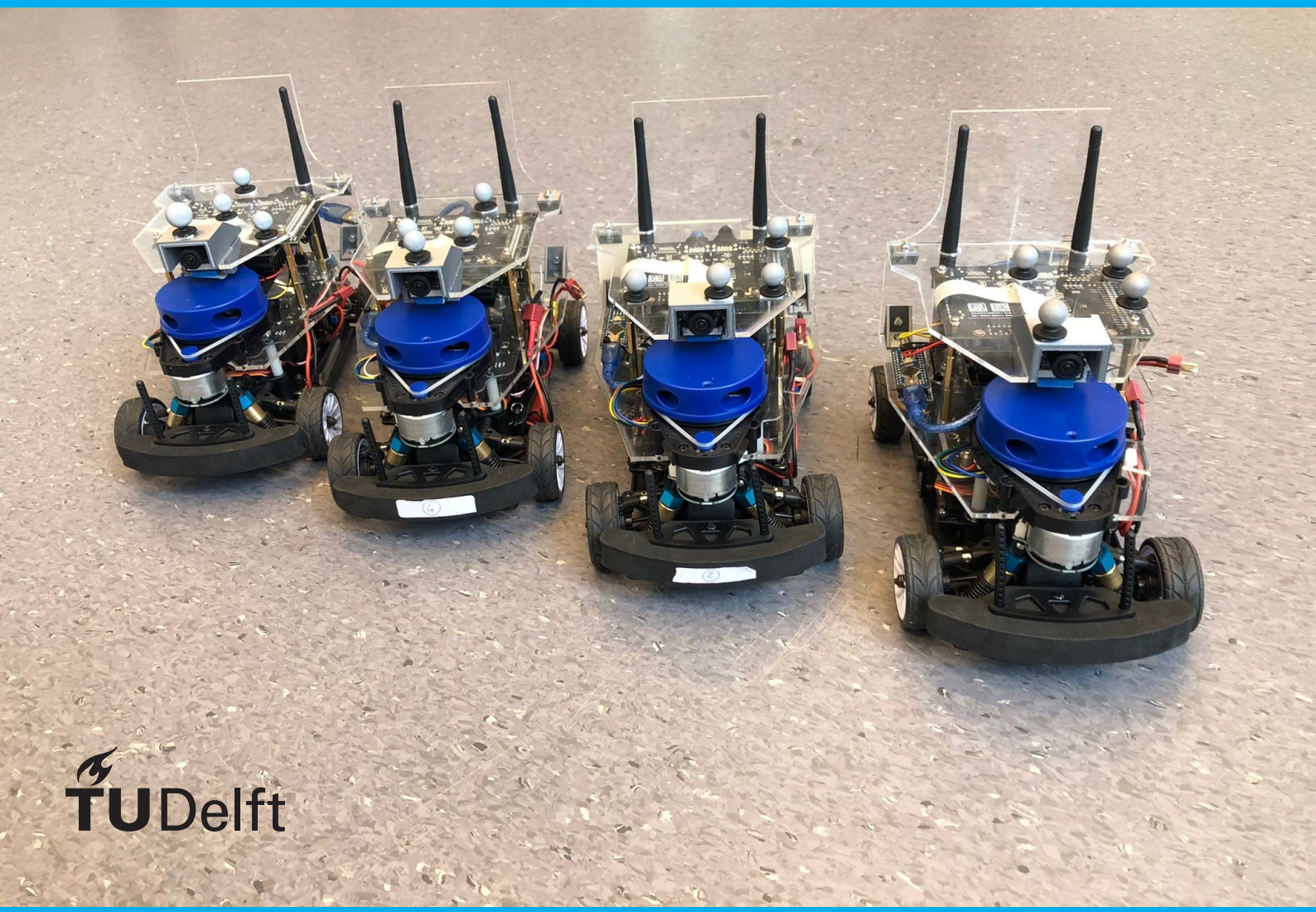
Master of Science Thesis

Design of Automated R2C Cars for Cooperative Driving Experiments

T. Niesten



TU Delft

Master of Science Thesis

Design of Automated R2C Cars for Cooperative Driving Experiments

by

T. Niesten

to obtain the degree of Master of Science
in Vehicle Engineering
at the Delft University of Technology.

Student number:	4157516	
Thesis committee:	Dr. L. Ferranti,	TU Delft, supervisor
	L. Lyons,	TU Delft, daily supervisor
	Dr. B. Shyrokau	TU Delft

TU Delft

Abstract

Cooperative driving controllers are becoming interesting subjects for future research in automated driving with the increase in connectivity. Using true-scale autonomous vehicles to properly test new control algorithms can be a challenge due to three main factors: costs, safety, and space requirements. To overcome these problems, scaled-down vehicle platforms or so-called Remote Control (RC) car platforms can be used to test algorithms in a safe environment without the large costs involved with true-scale platforms. At present, existing RC platforms are unnecessarily expensive or not able to drive autonomously.

This thesis presents the design of a low-cost 1/12 scale RC platform, consisting of modified JetRacer Pro AI's called **R2C Car's**. The sensor suite consists of a 160°FoV 8MP camera and an added 2D LiDAR. These are used in a sensor fusion and filtering framework to detect and locate other R2C Cars. Furthermore, wheel encoders are added to retrieve a velocity estimate of the R2C Car. To showcase the possibilities of the **R2C platform**, a longitudinal cooperative controller is implemented. This controller is based upon a Distributed Model Predictive Control algorithm, whereby 5Ghz WiFi is used for the communication between R2C Cars. Evaluation of the R2C platform together with the cooperative controller demonstrates that the platform is suitable for cooperative driving experiments. Furthermore, information is presented on the performance of the distributed algorithm.

Acknowledgements

The work presented is the final thesis for obtaining the degree of Master of Science. The goal of this thesis is to design and implement a new RC car platform for cooperative driving experiments. The hardware equipment for this platform was supplied by the **TU Delft Safety and Security Institute**. So I would like to thank them for this joyful experience.

I would like to sincerely thank my supervisors Dr. L. Ferranti and L. Lyons from the Delft University of Technology (TU Delft). Both are amazing people to work with and are always ready to assist whenever needed. L. Ferranti for introducing me to the RC cars, giving me full freedom for the design, and making it a privilege to work with them. L. Lyons for the assistance in the experiments and the assembly of the RC cars, reducing the time needed, making it more fun, and having a discussion where needed.

I would also like to thank my fellow master's students that were present in the bi-weekly meetings. The feedback and comments were much appreciated and it was a good way to keep progressing without getting stuck. Not only on the content of my thesis but also their experience on their thesis journey, especially in the beginning when you can get lost pretty quick on what to do next.

Finally, I would like to thank my family and girlfriend. They kept supporting me in difficult times, providing distractions when needed but also pushing me in the direction of the finish line.

T. Niesten
Delft, University of Technology
May 4, 2022

Contents

1 Introduction 1
1.1 Problem Formulation . . . 1
1.2 Research Objectives . . . 2
1.3 Thesis Outline . . . 2

2 Related Works 5
2.1 Existing RC Car Platforms . . . 5
2.2 Waveshare JetRacer Pro AI . . . 8
2.3 Comparison . . . 9

3 Hardware Design 11
3.1 Waveshare JetRacer Pro AI: Hardware Review . . . 11
3.2 Design Criteria . . . 12
3.3 Proposed Upgrades . . . 13
3.3.1 Localization . . . 13
3.3.2 Feedback from the control inputs . . . 15
3.3.3 Weight distribution and stiffer shock absorbers . . . 18
3.3.4 Release the full computational power of the Jetson Nano . . . 19
3.4 Proposed design . . . 20

4 Software Design 23
4.1 Detection Framework . . . 23
4.2 Velocity Measurement framework . . . 26

5 Experimental Cooperative Controller Design 29
5.1 Model Predictive Control . . . 29
5.1.1 Discrete Dynamic Model . . . 30
5.1.2 Cost function . . . 30
5.1.3 Constraints . . . 31
5.1.4 Optimization Problem . . . 32
5.2 Distributed Optimisation . . . 32
5.2.1 Lagrange Formulation . . . 32
5.2.2 Primal-Dual Method of Multipliers . . . 32
5.3 Proposed Distributed Model Predictive Control Framework . . . 33

6 Experiments 35
6.1 Communication . . . 35
6.2 Cooperative Driving . . . 36

7 Discussion and Future Work 41
7.1 R2C Car . . . 41
7.2 Cooperative controller . . . 43

8 Conclusion 45

Bibliography 47

1 Introduction

Autonomous vehicles captured the interest of many researchers and companies like Tesla [36], and Waymo [41]. The main promise of autonomous vehicles is to reduce traffic accidents, travel time, and traffic and contributes to a more comfortable driving experience. One of the most promising use cases for autonomous vehicles is the so-called Robotaxi, which is an autonomous taxi operating on public roads in a pre-specified area. Two examples are China's largest fully driverless Robotaxi AutoX [3]and Waymo's Robotaxi Waymo One deployed in East Valley of Phoenix, Arizona, USA [42].

Interesting applications of autonomous vehicles are not only limited by single-vehicle automation, groups of vehicles could communicate with each other to jointly agree upon maneuvers and navigation strategies. This *cooperative driving* can further reduce traffic accidents and can create faster more efficient transportation. For example, *Cooperative Adaptive Cruise Control (CACC)*, extends the already available Adaptive Cruise Control (ACC) for commercial vehicles. ACC is an advanced driver assistance system that maintains a constant longitudinal velocity but adjusts this velocity to maintain a safe distance from vehicles ahead. Both systems are considered a level 1 automation as defined by the SAE International [17], when combined with lane-centering control the vehicle is considered a level 2 autonomous vehicle. At these levels, the systems are support features while the driver maintains constant supervision.

ACC systems rely on onboard sensors to derive the distance to the vehicle in front and adjust the vehicle velocity accordingly. Due to the delay between measurement and action, ACC may not exhibit string stability [13], whereby oscillations are amplified in the traffic flow by acceleration and deceleration in vehicles. This can lead to collisions or so-called phantom traffic jams. CACC in contrast receives information about vehicles in front using communication systems, reducing the delay in responding to the vehicles in front. This improves or addresses string instability [23] [43] [21]. But communication problems like network imperfections and maintaining sensitive data private can become a problem for CACC controllers.

Cooperative driving controllers like the CACC are thus interesting subjects for future research. A challenge in this research is to properly test these controllers. Most of the research is done in simulation [13] [23] [43] [21], without the verification of physical implementation.

Figure 1.1: MCity Test Facility [24]

1.1. Problem Formulation

Using true-scale autonomous vehicles is very expensive and requires large testing areas. One such example is the MCity Test Facility from the University of Michigan, whose development costs were €8.8 million and it covers $0.13 km^2$ (Figure 1.1). Maintaining the platform will introduce more expenses (maintaining the vehicles or repairing them when damaged). Another concern is the safety of the researchers working with the platform, therefore only a few certified

researchers can work with the platform limiting the possible usage of the platform itself. However, these platforms are useful for validating final design control algorithms but might be unnecessary for the design process.

To overcome these problems, scaled-down vehicle platforms or so-called Remote Control car platforms can be used to test algorithms in a safe environment without the large costs involved with a true-scale platform. With the latest state-of-the-art sensors and mini-computers that can process all sensor data and algorithms onboard, RC cars can perform similarly to true-scaled vehicles making it a great opportunity for research purposes. Note that the name Remote Control car insists that the cars are remotely controlled, this is not always the case anymore due to automation. Still, the same naming convention is used in literature to describe scaled-down vehicles.

These RC cars could communicate with each other to also be suitable for cooperative experiments. These cars should be small enough to perform multi-robot experiments indoors. Since the use of multiple RC cars increases the cost of the platform, each of them should be as cheap as possible but perform well enough for the experiments. Finally, these cars should drive autonomously, and thus have enough sensors and resources to do so.

Available RC car platforms used for research are still quite expensive for autonomous cooperative driving experiments (prices are in the range of 1000 euros and more for each car) or are not suitable for autonomous cooperative driving. Thus reducing these costs will result in a platform that is cheaper to purchase and maintain , opening up more possibilities for research purposes. With a new autonomous RC car available, the Jetracer Pro AI from Waveshare, a new reduced-cost platform could be made for autonomous cooperative driving experiments. The JetRacer Pro AI is easy to set up and less expensive than similar platforms. One problem is that it still only relies on one camera, which can be used for autonomous driving but its capabilities are very limited. Therefore this project aims to enhance the Nvidea Jetracer Pro AI while still keeping the platform low cost and easy to set up.

1.2. Research Objectives

The main objective of this project is:

To design an affordable autonomous RC car, that can be used for non-cooperative and cooperative driving experiments and implement a longitudinal cooperative controller to showcase its possibilities.

The research objective is formalized in the following steps:

- Review the main RC platforms to understand their potential and limitations
- Redesign and construct a low-cost multi-modal sensor suite for the JetRacer Pro AI to allow it to drive autonomously in cooperative driving experiments.
- Design and implement a framework that can process the sensor data.
- Design a longitudinal cooperative controller for multiple RC cars that proposes an input at a frequency of 10 Hz.
- Create a test setup for testing the performance of the modified JetRacer Pro AI.
- Evaluate the performance of the modified JetRacer Pro AI by implementing a cooperative controller.

1.3. Thesis Outline

The outline for this work is as follows:

- Chapter 2 mentions related RC car platforms. These RC car platforms will be compared with a newly available RC car platform, the JetRacer Pro AI.
- Chapter 3 proposes hardware modifications for the JetRacer Pro AI to extend its capabilities in autonomous cooperative driving experiments.
- Chapter 4 will discuss software implementations for the sensor suite of the modified JetRacer Pro AI.

- Chapter 5 proposes an experimental cooperative controller to test the capabilities of the modified JetRacer Pro AI in cooperative driving experiments.
- Chapter 6 evaluates the implemented controller and the modified JetRacer Pro AI.
- Chapter 7 will provide a conclusion and reflection on our research objectives.
- Chapter 8 contains a discussion on the conducted research, and recommendations for future research are provided.

2

Related Works

Many mobile robot platforms have been developed in the last couple of years, that are used in the field of automated driving. This report mainly focuses on platforms that use realistic vehicle dynamics like Ackermann steering, the ability to drive at high speeds, and consists of 4 wheels whereby two wheels are steerable. Ackermann's steering geometry is intended to reduce slip in the wheels when a vehicle is turning by having the axles of the wheels pointing to the center of rotation, as the Figure 2.1 depicts. Platforms like the two fixed wheeled robot TurtleBot2[27] or the four fixed wheeled robot Jackal UGV[26] are therefore excluded in this review. Driving multiple robots indoors can also be a problem when the size of the robots is too large. Hence, we do not consider vehicles like the 1/5 scaled vehicle-like GATech AutoRally [12] which is specifically designed for outdoor experimentation.

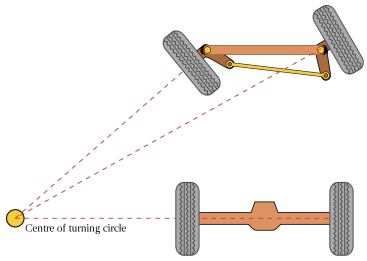


Figure 2.1: Ackermann steering geometry (Wikipedia, 2022).

In this chapter, we first review the main related RC car platforms. Then, we introduce the new Waveshare JetRacer Pro AI which is the backbone of our design. Finally, we compare existing platforms to the JetRacer Pro AI and discuss why this new robot could lead to a better Platform for Autonomous driving in cooperative driving scenarios.

2.1. Existing RC Car Platforms

Listed below are some commonly known related platforms ordered in scale, from 1/10 to 1/43 scale. Other less known platforms are comparable to the listed ones and are therefore omitted. A summary can be found in Table 2.1.

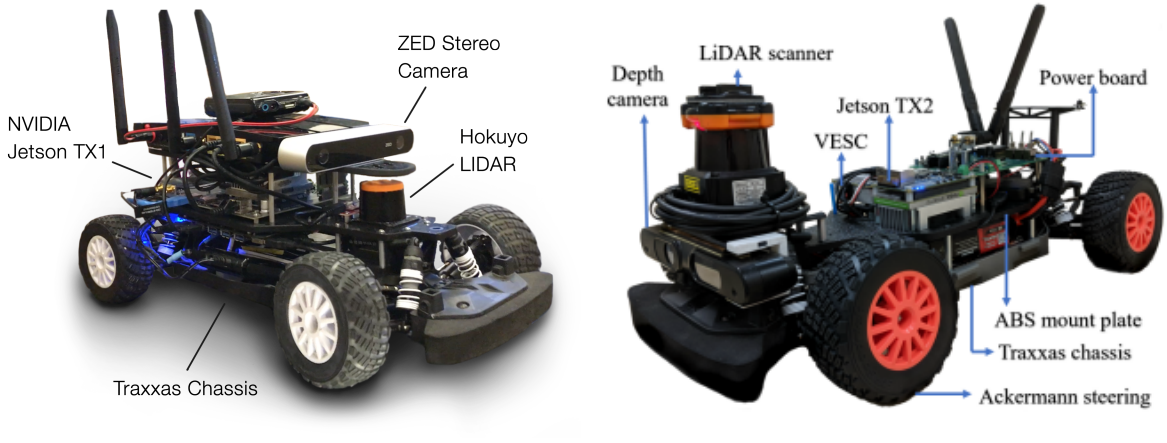
- **MIT RACECAR [2]**
  The Rapid Autonomous Complex-Environment Competing Ackermann-steering Robot, or RACECAR, is a 1/10 scale vehicle-like robot platform developed by MIT. The RACECAR uses a Traxxis Rally Car platform with a powerful Nvidia Jetson TX1 computer to process all the data from the attached state-of-the-art sensors, consisting off:

  - Hokuyo UST-10LX laser range finder
  - Stereolabs ZED stereo camera
  - Structure.io depth camera
  - Sparkfun IMU

  Having all these high-end sensors makes it easy to measure its surroundings and drive autonomously, but it increases the cost of the platform. The total cost of this platform is around $3663, with a sensorless base of around $888. Communication is done through 2.4GHz/5GHz Dual-Band High-Speed WiFi supported by the Nvidia Jetson TX1. Furthermore, the system is equipped with a self-made Vedder

ESC. This VESC unit controls the brushless motor and has the advantage of acting as a wheel odometer. Building the VESC can be problematic for inexperienced people since all components need to be soldered together.

A similar setup can be found in the F1/10 platform [25]. Only this setup uses an off-the-shelf ESC unit. Since the Platforms are almost identical, only one is mentioned in detail. Both setups can be found in Figure 2.2.



(a) (b)

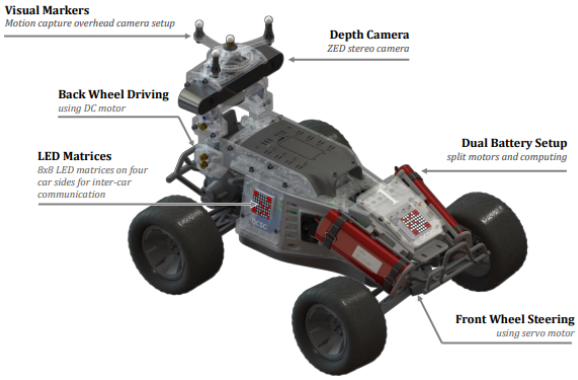
Figure 2.2: (a) MIT RACECAR, (b) F1/10

- **Design by S.J. van der Marel [37]**
  The 1/10 scale vehicle-like robot designed by S.J. van der Marel for his master's degree at TU Delft is developed for platooning purposes, with a specific focus on vision-based communication. The robot uses a powerful Nvidia Jetson TX2 computer, an upgraded TX1 with twice the performance and twice the efficiency. This board also uses 2.4GHz/5GHz Dual-Band High-Speed WiFi for communication. The original Erle Robotics Erle Rover base, depicted in Figure 2.3a, is no longer available, and therefore the LRP S10 Twister 2 2WD monster truck in combination with a Pixhawk PX4 is mentioned as a suitable replacement. The only sensor attached to the robot is a Stereolabs ZED stereo camera for robot vision and depth estimation. Visual markers are placed on the robot for location estimation using a motion capture system. Such systems use multiple cameras to track the positions of markers, with a cost of around $666 to $6660 for each camera. The total cost of this robot platform without the motion capture system is around $1720, with a sensor-less base of around $1332.

- **BARC [22]**
  The Berkeley Autonomous Race Car is a 1/10 scale vehicle-like robot platform developed for autonomous driving tasks such as drifting, lane changes, and obstacle avoidance, see Figure 2.3b. The latest version of the BARC, V4.0, is built with a Tamiya TT02 chassis and a Nvidea Jetson Nano mounted on top. The Jetson Nano board is not as powerful as the TX series, but with a fraction of the cost and smaller size can still be very interesting. The robot uses an Intel RealSense D435i camera, with the option of adding an additional Intel RealSense T265 camera for better localization. The cost for a robot with one Intel Realsense D435i camera will be approximately $871, with a sensor-less base of around $500.

- **MuSHR [32]**
  The Multi-agent System for non-Holonomic Racing, or MuSHR, is a 1/10 scale vehicle-like robotics platform developed at the University of Washington's Paul G. Allen School of Computer Science Engineering, see Figure 2.4a. Inspired by the MIT RACECAR project, they set out to create a more affordable robotic system. MuSHR is built using the Redcat Racing Blackout SC 1/10 chassis with a Nvidea Jetson Nano computer. The attached sensors are,

  – Intel RealSense D435i camera
  – YDLIDAR X4 laser scanner



(a)

(b)

Figure 2.3: (a) Design by S.J. van der Marel, (b) BARC V4.0

– VEX Bumper switch for collision detections

The MuSHR platform can be built for around $888, without any sensors the robot costs $588.

- **Donkey Car [4]**
  Donkey Car is a 1/16 scale vehicle-like robot platform aimed at self-driving experiments. There are 4 fully supported chassis available from the brand Exceed, some need adapters which can be easily 3D printed. The robot uses a Raspberry Pi with onboard 2.4 GHz WiFi or Bluetooth for communication. The only sensor attached is a Wide Angle Raspberry Pi Camera, the computational power of the Raspberry Pi falls short when more sensors are added. Therefore the author proposes the use of a Nvidea Jetson Nano instead of the Raspberry Pi, to increase the computational power significantly and add 5 GHz WiFi [5]. The system with the Jetson Nano can be built for around $333, without the camera for around $305.

(a)

(b)

Figure 2.4: (a) MuSHR, (b) Donkey Car

- **Cambridge minicar [15]**
  The Cambridge Minicar is a 1/24 scale vehicle-like robot built by the University of Cambridge, depicted in Figure 2.5a. Motivated to design a testbed that scales to a large number of robots for cooperative as well as non-cooperative experiments. The Cambridge Minicar consists of a 1/24 Range Rover Sport chassis and motor with a small-sized Raspberry Pi Zero W computer. The processing power is only a fraction of that of the Nvidia Jetson boards, but it can receive data by 2.4 GHz WiFi or Bluetooth, control

the car and process various small sensors. Although the design allows for the integration of an IMU or camera, the position tracking is done with a motion capture system, and communication is done using a Broadband Radio unit added to the robot. All intelligence is kept off-board since the minicomputer is not capable of computationally heavy algorithms. The sensor-less robot costs a mere $74, making it the cheapest robot on our list but not the cheapest platform since an expensive motion capture system and a powerful computer is needed for computationally heavy algorithms.

(a) (b)

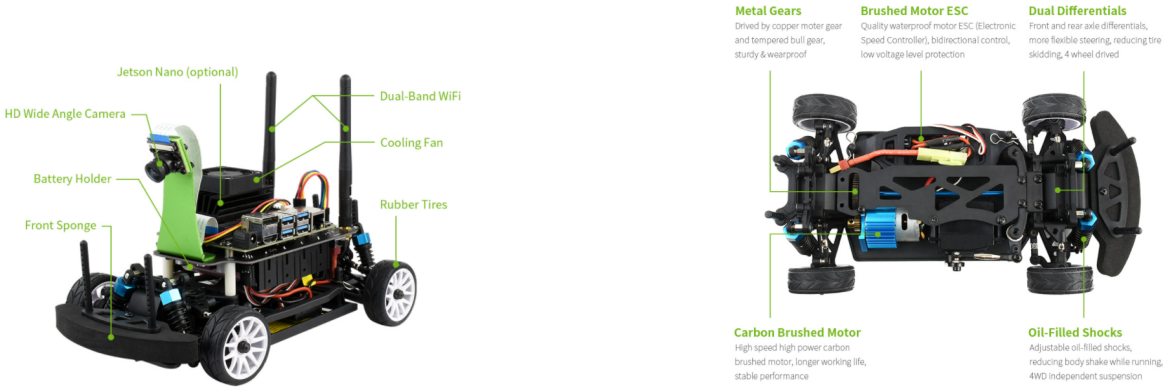
Figure 2.5: (a) Fleet of Cambridge Minicars, (b) ETHZ ORCA Racer

- **ETHZ ORCA Racer [18]**
  The Optimal RC Racing project , or ORCA, is a 1/43 scale RC cars platform made by ETH Zürich as an experimental testbed designed for testing autonomous racing controllers, as Figure 2.5b depicts. The cars are built with a Kyosho nano RC race car chassis and a custom-designed PCB with an ARM Cortex M4 Microcontroller and Bluetooth communication. Since this PCB is custom-made, the authors note that other off-the-shelf microcomputers, like the Raspberry Pi, can be used instead. The platform uses a motion capture system for localization of the cars and an external computer for running the computationally heavy algorithms. The cars do have an IMU sensor onboard, but in the current setup is only used to track forces in corners. The authors note that wheel speed sensors can be added using external connectors. The overall cost of one car is $458, by replacing the custom PCB with an off-the-shelf unit the cost can be reduced to approximately $222. Since the PCB has an integrated IMU, the sensor-less price will be the same as choosing an off-the-shelf PCB. An IMU could then be added for around $10

2.2. Waveshare JetRacer Pro AI

The JetRacer Pro AI kit from Waveshare [1] is an almost ready-to-go vehicle-like racing robot that can be assembled with a few simple steps. On top, an Nvidia Jetson Nano board needs to be placed which can be easily connected to the Jetracer Pro AI power board. From the Waveshare website a prebuild image can be downloaded which is made for the JetRacer Pro AI. This image contains the operating system Ubuntu 18.04 with software to operate the JetRacer Pro AI. Furthermore, The Waveshare website gives a few simple steps to download the Robotic Operating System (ROS) environment [33] and some example files to quickly start driving. ROS is an open-source development platform for robotic applications used by many researchers due to the easy implementation of processes. All related platforms mentioned in the previous Section 2.1, except for the Cambridge Minicar and ETHZ ORCA Racer, also utilize ROS for the software implementation.

The cost for a fully functional JetRacer AI Pro is around €460 for non-educational purposes, for educational purposes, a discount of 50% can be applied on both the kit and Jetson Nano board.
The system relies on one sensor for autonomous driving, which is an 8MP 160 deg FOV Camera. Communication is done using 2.4GHz/5GHz Dual-Band High-Speed WiFi with low latency. For better driving capabilities, all four wheels are connected to the brushed motor using a front and rear differential. The JetRacer Pro AI with all its components can be viewed in Figure 2.6.



(a) (b)

Figure 2.6: Waveshare Jetracer Pro AI (a) and chassis structure (b) [1]

2.3. Comparison

For our platform, an RC car is needed that can drive autonomously with computationally heavy algorithms (such as nonconvex solvers) running on board. Therefore the Cambridge Minicar and ETHZ ORCA racer are platforms not able to comply with these criteria. 3 out of the 4 leftover platforms are for most cooperative driving experiments unnecessary expensive since these are built for state-of-the-art detection algorithms that utilize expensive sensors. Our platform is built for cooperative driving algorithms that only need a low-cost setup to detect and locate other cars and their surroundings. The Donkey Car has a cost price considered acceptable when we try to set up a platform where multiple robots are needed for each experiment. In comparison with the Waveshare JetRacer Pro AI, the cost for a similar setup is further reduced from €300 to €230. Another advantage of the JetRacer Pro AI is that all parts are off the shelf and combined in one kit. The Donkey Car on the other-hand needs parts from various sources and a large 3D printed base making the build process more difficult. Another disadvantage is that the size of the Donkey Car is so small that adding additional sensors can become a problem. The full comparison can be viewed in Table 2.1.

The Jetracer Pro AI could be an interesting new RC car base for a new autonomous driving platform. It could be beneficial for cooperative driving experiments, where multiple RC cars are needed and a cost reduction of one RC car can have a big impact on the whole setup.

Platform	MIT RACECAR	Design by S. van der Marel	BARC	MuSHR
Scale	1/10	1/10	1/10	1/10
Cost (with sensors)	$3663	$1720	$871	$888
Cost (without sensors)	$888	$1332	$500	$588
Computational power	Jetson TX1	Jetson TX2	Jetson Nano	Jetson Nano
Sensors	Hokuyo UST-10LX Stereolabs ZED Structure.io depth camera	Stereolabs ZED	Intel RealSense D435i	Intel RealSense D435i YDLIDAR X4 VEX Bumper switch
Communication	2.4/5 Ghz WiFi or Bluetooth	2.4/5 Ghz WiFi or Bluetooth	2.4/5 Ghz WiFi or Bluetooth	2.4/5 Ghz WiFi or Bluetooth
Pose estimation	IMU	motion capture system	-	-

Platform	**JetRacer Pro AI**	Donkey Car	Cambridge minicar	ETHZ ORCA Racer
Scale	1/12	1/16	1/24	1/43
Cost (with sensors)	$255	$333	$74	$232
Cost (without sensors)	$232	$305	$74	$222
Computational power	Jetson Nano	Jetson Nano	Raspberry Pi Zero W	ARM Cortex M4 Microcontroller
Sensors	Wide Angle Camera	Wide Angle Camera	-	-
Communication	2.4/5 Ghz WiFi or Bluetooth	2.4/5 Ghz WiFi or Bluetooth	2.4 Ghz WiFi or Bluetooth	Bluetooth
Pose estimation	-	-	IMU and motion capture system	motion capture system

Table 2.1: Different RC car platforms used in literature

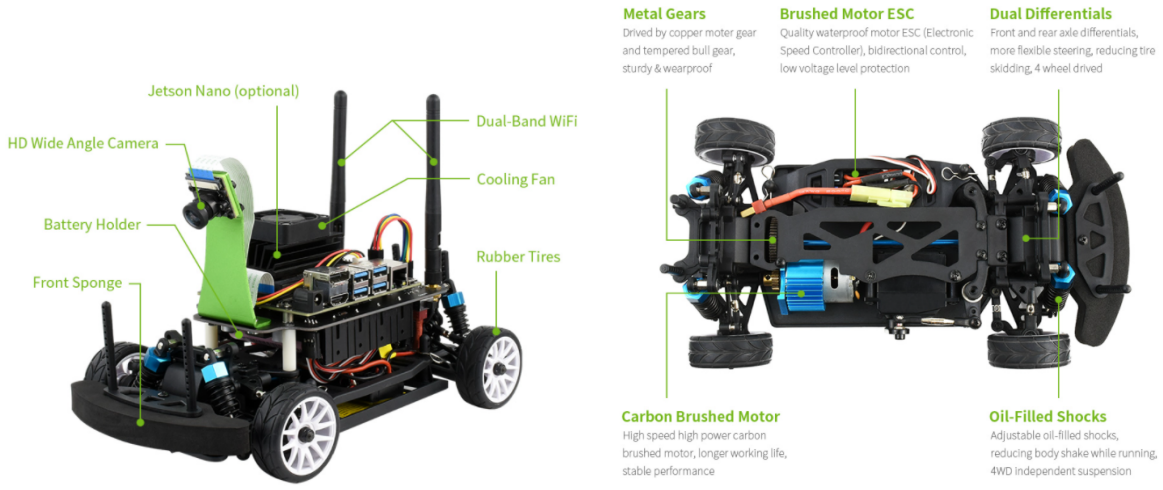
3

Hardware Design

This chapter first presents a more detailed description of the JetRacer Pro AI and its limitations. We then formulate the design criteria for the modification of the original setup. Upgrades are then discussed to comply with the design criteria. The final proposed design will then be presented, incorporating all discussed modifications.

3.1. Waveshare JetRacer Pro AI: Hardware Review

The JetRacer Pro AI chassis structure components were inspected and together with the specification given by Waveshare a review can be given on the components. The wheels are driven by a front and rear axle differential to allow better steering and reduce tire skidding. A carbon brushed motor drives both differentials using sturdy and wearproof metal gear, hard plastic gear, and the main axle. The motor is controlled by a brushed motor Electronic Speed Controller, or ESC, that is connected to the Expansion Board using a Tplug, or Dean plug, for power and a 3 pin connector for the control inputs. Steering of the front wheels is done using a servo motor, which is also connected to the Expansion Board. A figure of this chassis structure can be viewed in Figure3.1b, which is reported here again for convenience.



(a) (b)

Figure 3.1: Waveshare Jetracer Pro AI (a) and chassis structure (b) [1]

On top of the chassis structure (Figure 3.1a), the Expansion Board is placed. This board contains a holder for 4 High Capacity 18650 batteries, which provide power to the motor, servomotor, and Jetson Nano. The batteries are protected against over-charge, over-discharge, over-current, and short circuits by a battery protection circuit on the board. The expansion board also contains an OLED Display that displays the battery

life and displays monitoring of the attached Jetson Nano like CPU and GPU usage, IP address, and memory usage. Finally, the board also provides mounting of the antennas for the WiFi module and the camera mount can be attached on the front side. The camera has a 160 deg FOV (Field Of View) lens and an IMX219 8MP high-quality sensor which can provide a resolution up to 3280x2464. The Jetson Nano board is created by Nvidea and is a small but powerfull computer. The specifications are the followings:

- **CPU**: Quad-core ARM A57 @ 1.43 GHz
- **GPU**: 128-core Maxwell
- **Memory**: 4 GB 64-bit LPDDR4 25.6 GB/s
- **Storage**: microSD card slot
- **Camera**: 2 MIPI CSI-2 DPHY lanes for attaching two camera modules
- **Connectivity**: Gigabit Ethernet and a M.2 Key E slot for Wireless connectivity. The module that can be bought together with the Jetracer Pro AI is the 2.4 GHz/5 GHz Dual-band Wireless AC8265 module which also supports Bluetooth 4.2
- **Display**: HDMI and Display port
- **USB**: 4x USB 3.0 and USB 2.0 Micro-B
- **J41 Header**: 40-pin header for GPIO, I^2C, I^2S, SPI, and UART protocols, ground pins, and power pins (two 3.3V and two 5V pins).

The Jetson Nano is connected to the expansion board with a 6-pin connector cable. Two cables are for the I^2C protocol for communication between the two boards, needed to control both the servo motor and brushed motor and display information on the OLED Display. One cable is the ground cable, another cable is a 3.3V power cable for supplying power to the other pins of the J41 Header and finally two cables for supplying 5V and a maximum of 6A (3A each cable) for powering all other components of the Jetson Nano.

The Jetson Nano can be set to a 5W and 10W power mode respectively. With 5V it draws 1A and with 10W it draws 2A. The current increases when other peripherals are connected through the USB ports. In the current setup of the Jetracer Pro AI without any USB devices connected, it is already highly recommended by Waveshare to put the Jetson Nano in 5W mode. This is due to high spikes in the current draw by the brushed motor when the load is increasing, for example with high acceleration and deceleration, and the expansion board is not able to deliver enough current for both the motor and Jetson Nano. During initial test driving, the power limitations were already visible since the JetRacer Pro AI shut down during heavy breaking with the motor. The spike in the current caused the over-current protection by the battery protection circuit to be activated and cutting power to all systems. Setting the power mode to 5W decreases the load on the expansion board, but it also limits the computational power since two CPU cores are shut down and the other two are limited to 1 GHz.

3.2. Design Criteria

To improve the original JetRacer Pro AI, a list of design criteria must be specified to overcome problems with the current setup. All the proposed modifications for these design criteria should be low in cost. The four criteria are:

1. **Localization**: The current setup is fully dependent on a single monocular camera. It is used to track a road and steer accordingly, but with only one camera it is impossible to determine distances to its surroundings if dimensions are unknown. Therefore, to localize itself in its surroundings, another sensor needs to be added to the sensor suite or replace the current sensor.

2. **Feedback from the control inputs**: The JetRacer Pro AI can be controlled using a steering and throttle input for respectively the servo motor and the Electronic Speed Controller (ESC). The ESC accepts values between -1 to 1 from the provided code from Waveshare and the same holds for the servo motor. The servo motor output maps linearly to the input, so the input will translate to the steering range $-20\,\text{deg}$ to $20\,\text{deg}$. This is not the case for the output of the motor that is connected to the ESC, since the output doesn't map linearly to the input. The conservation of energy inside a motor in steady-state follows,

$$IU = I^2R + \omega\tau \tag{3.1}$$

Here I and U are respectively the input current in Ampere and voltage in Volt, R is the resistance of the armature of the electric motor in Ohms, ω denotes the angular velocity in rad/s and τ is the output torque in N/m. Generally, the torque is proportional to the current in motors,

$$\tau = pI \tag{3.2}$$

Where, p is a constant. Rewriting equation 3.1 to,

$$\tau\frac{U}{p} = \frac{\tau^2}{p^2}R + \omega\tau \tag{3.3}$$

$$\tau = \frac{pU}{R} - \frac{\omega}{p^2R} \tag{3.4}$$

$$\tag{3.5}$$

Voltage is controlled by the ESC, mapping linearly to the accepted values mentioned previously. Since $R > 0$, torque will decrease when the angular velocity increases. Therefore, to retrieve an accurate angular velocity measurement, the sensor suite of the JetRacer Pro AI should be upgraded to be able to do so.

3. **Better weight distribution and stronger shock absorbers**: For better driving capabilities, the center of mass should be moved more to the middle of the JetRacer AI Pro. Currently, it is shifted to one side, making the steering dynamics different when steering to each side. Also by evenly distributing the weight, the forces on the wheels are better distributed and thus have equal grip. Furthermore, the shock absorbers are barely handling the original weight of the JetRacer Pro AI. To overcome any problems when the weight is increased by modifications, the shock absorbers should be made stiffer.

4. **Release the full computational power of the Jetson Nano**: In the current setup, the Jetson Nano can only perform at 1/3 of its full potential. To run computationally heavy algorithms, the Jetson Nano should be able to run in 10W power mode without any problems concerning over-current protection in the expansion board.

3.3. Proposed Upgrades

Now that the design criteria are specified, upgrades to the JetRacer Pro AI can be discussed. Each of the criteria will be discussed with their final solutions. Thus first sensors are discussed for accurate localization. Then, to retrieve feedback from the JetRacer Pro AI, a sensor suite is specified for accurate velocity measurements. A better solution for the weight distribution is then presented, with a solution for the stronger shock absorbers. Finally, a modification is proposed to get the full potential of the Jetson Nano.

3.3.1. Localization

Sensors perceive their environment and give cars the capability to drive autonomously. Using the related RC car platforms, three sensors are commonly used: *Stereo Camera*, *RGB-D Camera*, and *LiDAR*. Each of the sensor types is presented below.

1. **Stereo Camera**: A single camera has limited depth information if the size of an object is not known beforehand. By using a second camera, the difference between the images can be analyzed to obtain the depth of objects in the image if the rotation and translation between the two camera frames are known. One such stereo camera can be bought from Waveshare, the IMX219-83 8MP binocular camera module for $56.99 [40]. The accuracy of depth measurements depends on the distance between the

cameras, the resolution, and the stereo matching algorithm. Especially with the close distance between the cameras in the Waveshare module, the accuracy will become a problem for higher distances. The stereo matching algorithms are also computationally demanding due to the high-resolution imagery needed.
Instead of doing calculations on the Jetson Nano, Stereo camera modules can be bought with onboard hardware that does all the computations to retrieve the depth map. These modules have a higher price point, one such module is the OAK-D-Lite HD camera for $189.99 [39]. The depth measuring range is 0.2m to 9m, for close cooperative driving, this can create a problem when determining the position on the JetRacer Pro AI.

2. **RGB-D Camera**: The previously discussed stereo cameras are passive camera systems, they receive reflected light from their surrounding. Active camera systems use a projector to light up their surrounding, the reflections are captured and processed. In practice these projectors use infrared light and are captured by stereo infrared cameras and are accompanied by a monocular camera, such systems are called RGB-D camera systems. Two frequently used cameras, also presented in Chapter 2, are the Stereolab ZED and RealSense D455. The Stereolabs ZED 2 can be bought for $449, and has a working range from 0.2 - 20m and a depth frame rate of 100 Hz [34]. The Intel RealSense D455 can be bought for around $399, has a working range of 0.3m - 10m, and a Depth frame rate of 90 Hz [16]. The depth accuracy for both systems decreases quadratically, with a depth accuracy of 1% in the near range. Both systems also require an extensive amount of computational power to process the depth frames.

3. **LiDAR**: LiDAR systems work by sending out an IR pulse, which will be reflected by an object, captured by a sensor again and the time it takes between sending and receiving can be used to calculate the distance between LiDAR and object. By rotating and taking multiple of these measurements a point cloud can be formed, showing the distances between the LiDAR center and surrounding objects. LiDAR systems can be divided into two subcategories: The systems that create only a 2D point cloud and systems creating a 3D point cloud. 3D LiDARs are very expensive and are thus not considered for the new platform. Two low cost 2D LiDARs are the YDlidar X4 for $103 [10] and RPLIDAR A1M8 $115 [11]. The YDlidar X4 has a ranging distance from 0.12 to 10m with an accuracy <0.5 mm in range$\leq 2m$ and <1% in range $> 2m$. It can measure the distance at 5kHz and has a scan frequency between 6-12 Hz, so at a scan frequency of 10Hz, the created point cloud will contain 500 data points. The RPLIDAR A1M8 has a ranging distance of 0.15 - 12m with an accuracy of <1% in this range. It can measure the distance at 8kHz and has a scanning frequency of 5-10Hz.

For our purpose, the JetRacer Pro AI is only interested in obstacles that are placed in his working space, so only using a 2D LiDAR would be sufficient to detect obstacles if it is placed at an appropriate height in this working space. Therefore, the LiDAR is a more interesting option than the RGB-D Cameras compared to the price. The stereo camera is cheaper, but it lacks accuracy and would demand a lot of computational power of the Jetson Nano to get distance measurements. The only problem in a 2D point cloud is to detect other JetRacer Pro AI's in cooperative experiments since it detects objects but it cannot recognize them. The JetRacer Pro AI comes with a monocular camera that addresses this problem and since it cannot measure depth, both sensors can assist each other. A more in-depth explanation is given in Chapter 4.

For close distance cooperative driving experiments, high accuracy and high frequency are needed. Thus the 2D LiDAR chosen for the platform is the YDlidar X4, which has better accuracy in close range, has a higher range for the scan frequency, and comes at a lower cost. The LiDAR should be placed as low as possible, just above the Expansion board, to detect most of the small objects. The YDlidar X4 has another advantage in this aspect, since the bottom plate is not part of the electric circuit and thus can be removed (see Figure 3.3). This is not the case for the RPlidar where the lower board is part of the electric circuit and thus would be positioned higher.

Vehicles are mostly used with the intention of driving forward and only driven backward for parking scenarios with a few exceptions. Therefore, the LiDAR can be placed in front on top of the Expansion board next to the Jetson Nano board, see Figure 3.3. This decreases the FoV of the LiDAR to around 220 degrees but makes it possible to place it close to the Expansion board. Finally, the LiDAR is placed backward as much as possible so it can detect objects close to the JetRacer Pro AI, in the new position of 4cm in front.

(a) (b)

Figure 3.2: (a) YDLidar X4 [29] (b) RPLidar A1M8 [28]

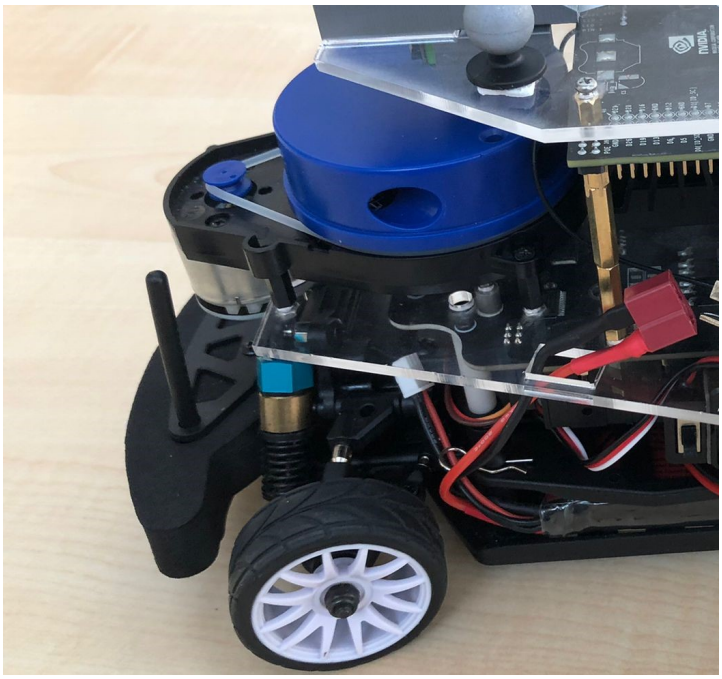


Figure 3.3: Placement of the YDLidar X4 on the JetRacer Pro AI

3.3.2. Feedback from the control inputs

To measure the velocity without off-board sensor systems like a motion capture system, the sensor suite needs to be improved. One of the cheapest and easily implementable options is using encoders, based on either optical or Hall effect, that can measure the angular velocity of the main driving shaft. This angular velocity can then be converted to the velocity of the JetRacer Pro AI relative to the ground in the case of no-slip between wheels and ground. The optical encoder makes use of an infrared light emitter and an infrared light sensor. There are two primary implementations of an optical encoder, based on either reflection or interferential. The reflection-based optical encoder uses a wheel that is divided into equally spaced reflective and less reflective surfaces. Both the emitter and receiver are next to each other pointing to the wheel as the Figure 3.4a depicts. When this wheel turns around, the encoder can detect the angular excursions. These detections can be transformed to the RPM using the geometry of the detections and the time derivative. The interferential optical encoder uses a wheel with holes in it, where the infrared emitter is placed on one side

of the wheel and the infrared sensor on the other side pointing to each other, called the gate, see Figure3.4b. Infrared light is only received by the sensor when there is a hole between the gate. By detecting the holes moving through the gate, the RPM could be calculated similarly to the interferential optical encoder. The encoder based on the Hall effect is a sensor that detects the presence of magnetic fields. By rotating small magnets on a wheel the encoder can measure the RPM similar to the optical sensor.

(a) (b)

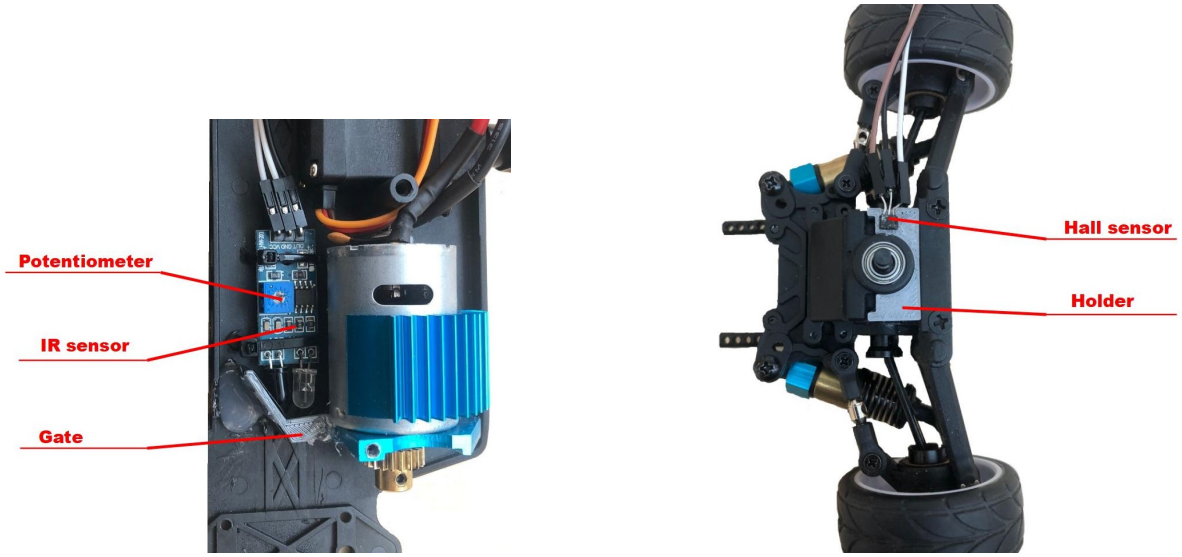
Figure 3.4: Sensor working principle with an IR emitter that sends out the IR light and a receiver that catches the light again for, (a) reflective optical sensor and (b) interferential optical sensor

Both optical and Hall encoders have different advantages and disadvantages. The optical encoder can have a higher detection rate by limiting the view of the sensor to a specific point, which is harder to do with a magnetic field. The Hall encoder detection rate is limited by the size of the used magnets and their magnetic field because they should still be strong enough to be detected by the encoder. One advantage that the Hall encoder has over the optical encoder is that the sensor can be placed in small spaces, the interferential optical sensor needs a specific disk with holes to be placed on the shaft and the reflection optical sensor needs to be placed in a small distance away from the reflective surfaces.

The interferential optical sensor is not used since the space is limited and it will be very hard to fit in a disk with holes on the main shaft. By using the main gear of the JetRacer Pro AI as a spinning wheel, the magnets or reflective and less reflective surfaces could be added to the sides. To have a velocity measurement as accurate as possible inside of a time frame, the number of detections should be maximized to minimize the noise in the sensor data. This can be done by utilizing both sides of the main gear, see Figure 3.5. The reflective optical encoder is preferred over the Hall sensor because of the higher number of detections each revolution, but it cannot be placed on both sides due to the available space. Therefore, on one side the reflective optical encoder is placed, and on the other side a Hall encoder. The reflective optical encoder is attached with tie-wraps to the JetRacer Pro AI and a 3D printed gate is made to increase the accuracy of the encoder as depicted in Figure 3.5a. The reflective optical encoder also has a potentiometer to adjust the sensitivity. The Hall sensor is attached with a 3D printed holder that is attached to the differential case as depicted in Figure3.5b .

The main gear itself is black, which is a color that absorbs most of the incoming light. White is a color that reflects most of the incoming light and is, therefore, suitable to be chosen for the reflective parts. Any flat material is suited as long as it has a white color, in our case we choose small plastic parts that could be easily glued to the main gear, see Figure 3.6. A white plastic ring was used where 10 equally spaced holes were cut out using a laser machine to show the black color of the main gear. This leaves us with a total of 20 detections each revolution since both the changes between white to black and black to white can be detected. To know if the 20 detections are equal up to a certain accuracy, a test can be performed. A constant throttle input of 0.2 was set without the R2C Car touching the ground. The period between detections is used to determine if there is any difference between the detections. The detections contain noise due to inaccuracies in the sensor, therefore, the 20 detections are averaged over 500 rotations to reduce the noise in the measurements. The results in Table 3.1 show that the accuracy without noise is below 1%, and thus both the white and black detections can be used to determine the RPM. Further work on retrieving a velocity and reducing the noise in the measurements is discussed in Section 4.2.

For the magnets, 4 strong Neodium 5x2x1.5 magnets are used. These were placed on the other side of the main gear, evenly spaced between each other depicted in Figure 3.6. To test if both detections from magnetic



(a) (b)

Figure 3.5: (a) IR sensor setup and (b) the Hall sensor setup

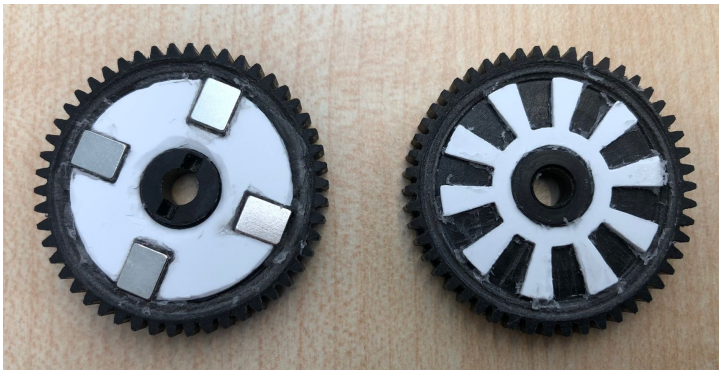


Figure 3.6: Left: One side of the main gear with the 4 Neodymium magnets. Right: White detections are added to the gear for the IR sensor, note that the white ring is not seen by the IR sensor due to the sensor gate.

detection	1	2	3	4	5	6	7	8	9	10
period [ns]	511.48	513.57	514.74	514.22	515.74	513.17	514.33	513.83	515.43	514.31
detection	11	12	13	14	15	16	17	18	19	20
period [ns]	514.14	513.84	514.37	514.46	514.38	513.96	513.00	516.82	510.28	515.86

Table 3.1: The average period is over 500 revolutions for the detections in one revolution of the IR sensor.

to non-magnetic and vice versa could be used, the same test as for the reflective optical sensor is done on the Hall sensor. The results in Table 3.2 show that the accuracy without noise is also below 1% between detections and thus both non-magnetic and magnetic detections can be used.

detection	1	2	3	4	5	6	7	8
period [ns]	1347.61	1359.20	1363.80	1362.42	1365.67	1348.26	1344.63	1344.36

Table 3.2: The average period over 500 revolutions for the detections in one revolution of the Hall sensor.

Both sensors could be attached to the J41 header of the Jetson Nano, but there is a problem. The Jetson Nano uses a software implementation to detect if the digital output of the sensors changes. This causes inaccuracies in the timing if multiple processes are running on the CPU because there is a possibility that

the detection has to wait for other processes to be finished. This inaccuracy causes problems with the fast detections of each of the sensors. Therefore an Arduino Nano microprocessor is chosen to process the fast detections and calculate the velocity which is then sent over USB to the Jetson Nano board. The Arduino Nano has two digital interrupt pins which are used for the two sensors. These two pins are connected to a hardware implementation that checks the change in digital output very fast, in our implementation a detection is done in $12ns$ but it depends on the code that is uploaded to the hardware of the interrupt pins. The main processor can then retrieve the values from the hardware of the interrupt pins to be processed further. The full setup of the Arduino Nano with the sensors and cables is around $15.

3.3.3. Weight distribution and stiffer shock absorbers

Roll dynamics are an important aspect of cars at higher velocities, the same holds for the JetRacer Pro AI. The body of a car wants to roll around its Center-of-Mass (CoM) due to centripetal forces in corners on the CoM and the friction force in the tires. The friction force creates a torque around the center of mass determined by the height of the CoM, y, see Figure 3.7. This roll torque should be lower than the torque created by the force of the weight and the distance between the wheel and CoM, x, or the car will roll over. So to prevent this rollover, y has to be minimized and x has to be maximized.

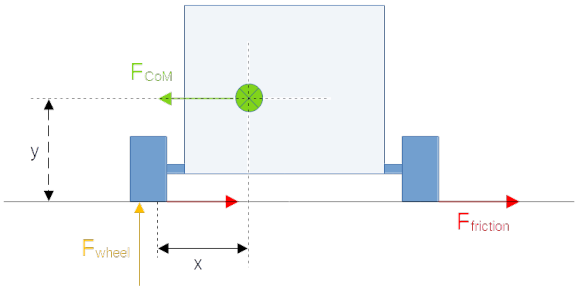


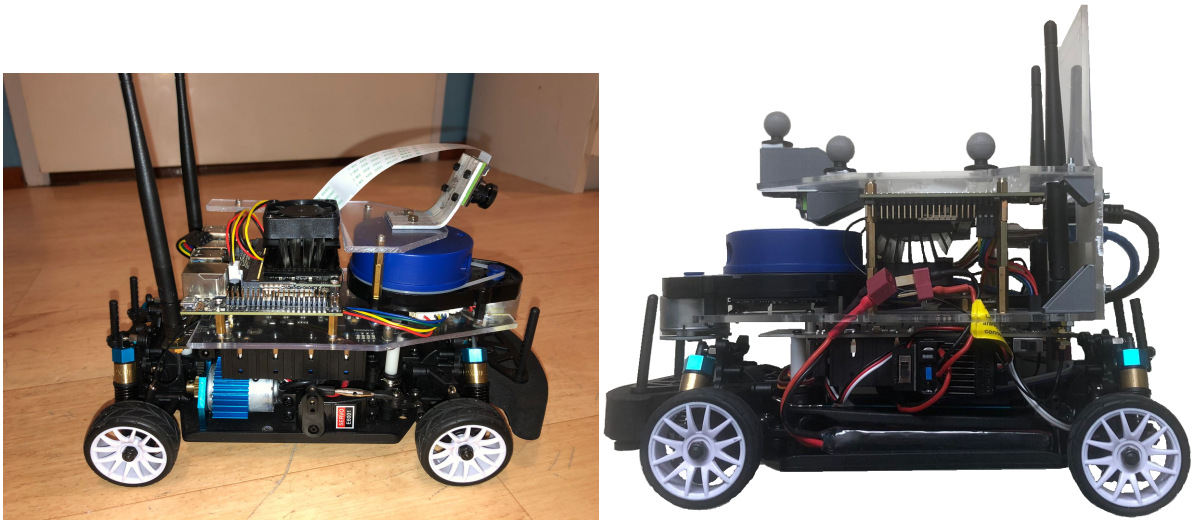
Figure 3.7: Front view schematic of a car. In green is the force on the CoM, red is the friction force, and orange is the force from the weight. x and y denote the positions of the CoM relative to the wheel on the outside of the corner

The cooling block of the Jetson Nano is on the same side as the motor and servo, making the weight distribution uneven, see Figure 3.1. This uneven weight distribution can cause problems during steering, because of the previously mentioned roll dynamics. The motor and servomotor are placed close to the ground, minimizing y and thus creating a low roll torque. This is not the case for the cooling block of the Jetson Nano, since it is placed on top of the Expansion Board and thus has a higher height y. In the original setup, the cooling block is placed to one side, see Figure 3.1. Making x smaller when turning to one side and higher turning to the other side. So there is a bigger risk of the JetRacer Pro AI rolling over when x is smaller and thus this distance should be increased. The wheels cannot be moved so the CoM of the cooling block should be moved to an even position between the wheels. By doing this, the roll torque is minimized as much as possible.

In the current orientation of the Jetson Nano board, it is hard to center the cooling block since a part of the Jetson Nano board will stick out to the side, making it prone to being hit and getting damaged. By turning the board 90 deg, the cooling block can be centered between the wheels, see Figure 3.9. Another advantage is that we can move the USB and video connectors to the rear of the car, so when the USB connectors are sticking out they can't be damaged in collisions.

By solving one problem naively another one is created, namely the 6 pin connector between the Jetson Nano and the Expansion board is too short and should be 4 times longer. The problem arises due to a voltage drop that increases with the length of the cable, the current running through it, and the copper core thickness. So instead of 5V, a lower voltage will be received by the Jetson Nano. Using the same rated cable of 24AWG, which defines how big the copper core is, the Jetson Nano gave the warning "System throttled due to Over-current." at full CPU load which caused the system to slow down. Note that the message tells something about the current, but it is also triggered for under-voltage. So higher rated cables should be used, but this increases the cost and difficulty to build the JetRacer Pro AI. By flipping the whole board upside down, the same cable can be used, the CoM of the cooling block will still lay in the middle between the wheels, and the height y remains almost equal as depicted in Figure 3.8b.

Due to the extra weight added to JetRacer Pro AI, the shock absorbers can't handle the load anymore. Thus the spring inside the shock absorber should be made tougher since the weight cannot be further reduced.



(a) (b)

Figure 3.8: (a) The first design with Jetson Nano board turned 90 deg and (b) The new setup with the Jetson Nano board flipped up side down.

Figure 3.9: The first design of the modified JetRacer Pro AI

This can be done by compressing the spring more in no-load condition, in our case using lightweight brass rings, or so-called solder rings, with an inner diameter of 10mm, an outer diameter of 12mm, and a length of 9mm, see Figure 3.10. Now that the springs are tougher, the Jetson Nano doesn't touch the ground on non-smooth surfaces and is less subjective to load displacement.

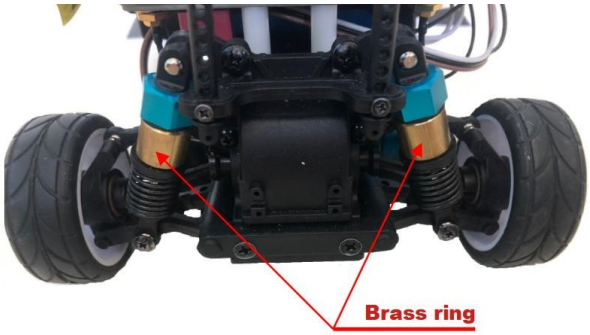


Figure 3.10: Brass rings were added to the rear shock absorbers

3.3.4. Release the full computational power of the Jetson Nano

From the Waveshare company, it is noted that the Jetson Nano board needs to be in 5W mode. This is due to too much current draw with both motor and Jetson Nano board in 10W mode. Even in the 5W mode, an incident occurred where the JetRacer Pro AI shut down because of this. When it happens the Expansion Board cuts all power, which is not desired. To overcome this problem another battery pack needs to be added.

For better performance of the motor, a LiPo battery is chosen because these batteries can deliver a high amount of current. A small and light-weight battery of 1300mAh with the dimensions 100x31x16 mm (l x b x h) [7] is chosen to fit in as seen in Figure 3.11. At the cost of only $10.40 each, extra batteries can be bought to switch out empty ones quickly and continue driving. It also depends on the connector type if further modifications are needed. An adapter cable can be bought, but the best way is to replace the connectors of the batteries so the cables aren't getting too long and the voltage drop is minimized as mentioned in the section 3.3.3. The LiPo batteries also need a special charger or so-called balancer, a low-cost version could be

bought for $15.30 [6].

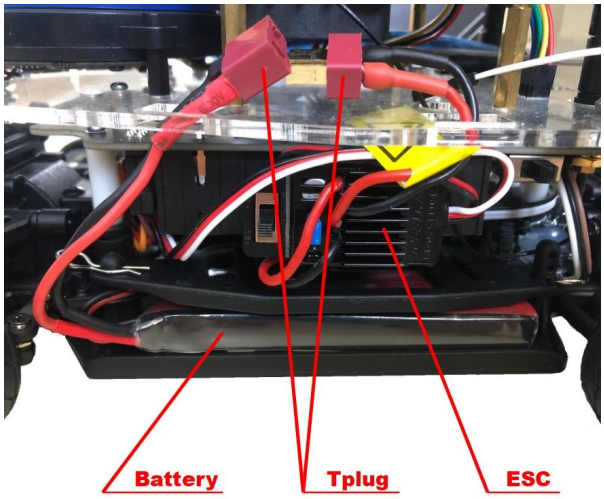
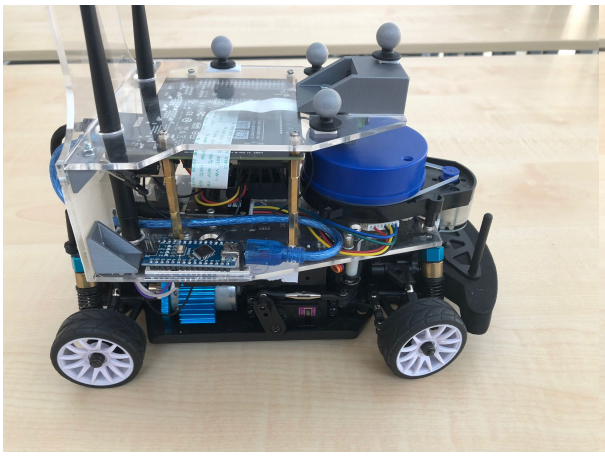


Figure 3.11: Position of the battery in the JetRacer

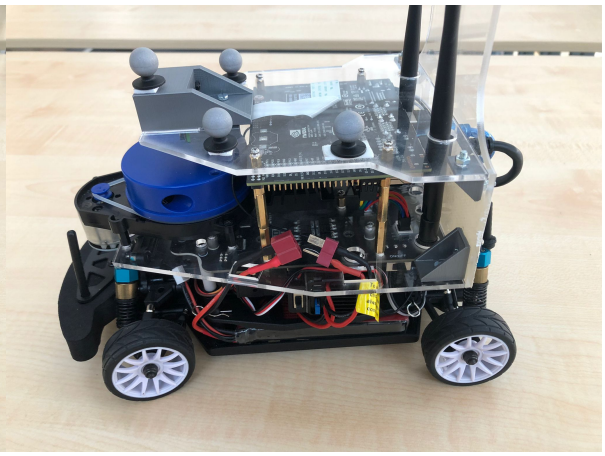
3.4. Proposed design

Now that all design criteria are met, the full final design can be viewed in Figure 3.12. The modified JetRacer Pro AI will be called **R2C Car**, named after the Reliable Robot Control (R2C) Lab at TU Delft.

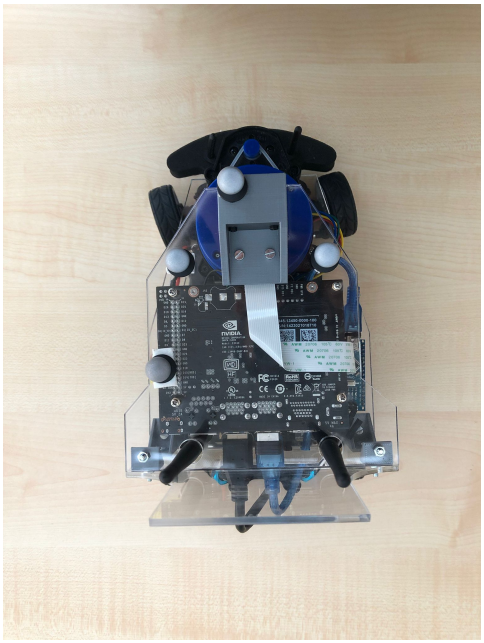
In the design, a plastic board is attached to the Expansion board to hold all components. A plastic backboard is placed at the end of the R2C Car with white tape attached as depicted in Figure 3.12d. In this way, the LiDAR from another R2C Car can detect this board and can better estimate the position. Another advantage is that markers can be placed on the backboard visible for other R2C Cars using the camera, this will be discussed in Chapter 4. On top of the R2C Car, another board is placed to protect the Jetson Nano board, create an attachment for the camera holder and provide the placement of visual markers, see Figure 3.12c. These markers are optional and can be used with a motion capture system like Optitrack, to retrieve the accurate position and orientation of the R2C Car. Note that this is optional, the R2C Car is designed to be able to drive independently with onboard sensors. All 3 boards are 3mm thick Plexiglas plates that were laser cut to the correct dimensions. The boards are attached with 3D printed holders and the Jetson Nano board is held in place using metal spacers, a box of these spacers [30] can be bought for $16.50. Note that the same box could be used to modify multiple R2C Cars. Finally, the 3D camera holder is made with a protective casing to prevent damage to the camera module itself. This holder is placed in such a way that the camera frame and LiDAR frame share a common origin with respect to the ground plane. The cost for the R2C Car is visible in the Table 3.3 and a comparison is visible with the other related RC platforms. The cost does not incorporate the 3 boards and the 3D printed parts, but these parts should not cost more than $20.



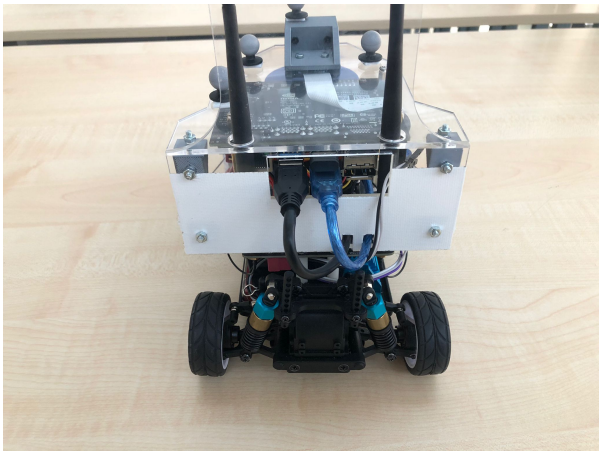
(a)



(b)



(c)



(d)

Figure 3.12: The R2C Car

Platform	MIT RACECAR	Design by S. van der Marel	BARC	MuSHR
Scale	1/10	1/10	1/10	1/10
Cost (with sensors)	$3663	$1720	$871	$888
Cost (without sensors)	$888	$1332	$500	$588
Computational power	Jetson TX1	Jetson TX2	Jetson Nano	Jetson Nano
Sensors	Hokuyo UST-10LX Stereolabs ZED Structure.io depth camera	Stereolabs ZED	Intel RealSense D435i	Intel RealSense D435i YDLIDAR X4 VEX Bumper switch
Communication	2.4/5 Ghz WiFi or Bluetooth	2.4/5 Ghz WiFi or Bluetooth	2.4/5 Ghz WiFi or Bluetooth	2.4/5 Ghz WiFi or Bluetooth
Pose estimation	IMU	motion capture system	-	-

Platform	**R2C Car**	Donkey Car	Cambridge minicar	ETHZ ORCA Racer
Scale	1/12	1/16	1/24	1/43
Cost (with sensors)	$415	$333	$74	$232
Cost (without sensors)	$232	$305	$74	$222
Computational power	Jetson Nano	Jetson Nano	Raspberry Pi Zero W	ARM Cortex M4
Sensors	Wide Angle Camera YDlidar X4	Wide Angle Camera	-	-
Communication	2.4/5 Ghz WiFi or Bluetooth	2.4/5 Ghz WiFi or Bluetooth	2.4 Ghz WiFi or Bluetooth	Bluetooth
Pose estimation	Optical encoder Magnet encoder	-	IMU and motion capture system	motion capture system

Table 3.3: Different RC car platforms used in the literature compared to the new R2C Car

4

Software Design

In this chapter, an R2C Car detection framework is proposed for cooperative driving. This framework uses a state-of-the-art fiducial tag detector which processes the 2D RGB images from the monocular camera and returns a 3D marker position in the camera frame. Fiducial tags as depicted in the Figure 4.1 are artificial visual features designed for automatic detection. A few key features make the tags useful for pose estimation and object tracking and therefore have been widely adopted by the robotics community. In our work, these tags are used to estimate the position of one R2C Car in another R2C Car's camera frame. These positions are then used to filter the 2D point cloud from the LiDAR to retrieve accurate relative R2C car positions. This cannot be done with the 2D point cloud alone since with multiple objects it is hard to determine which cluster of points denotes the other R2C Car as depicted in Figures 4.6a and 4.6b. We also propose a velocity measurement framework where both the IR and Hall sensor data are used to calculate the velocity at a rate of 20 Hz.

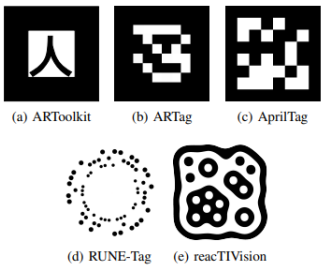
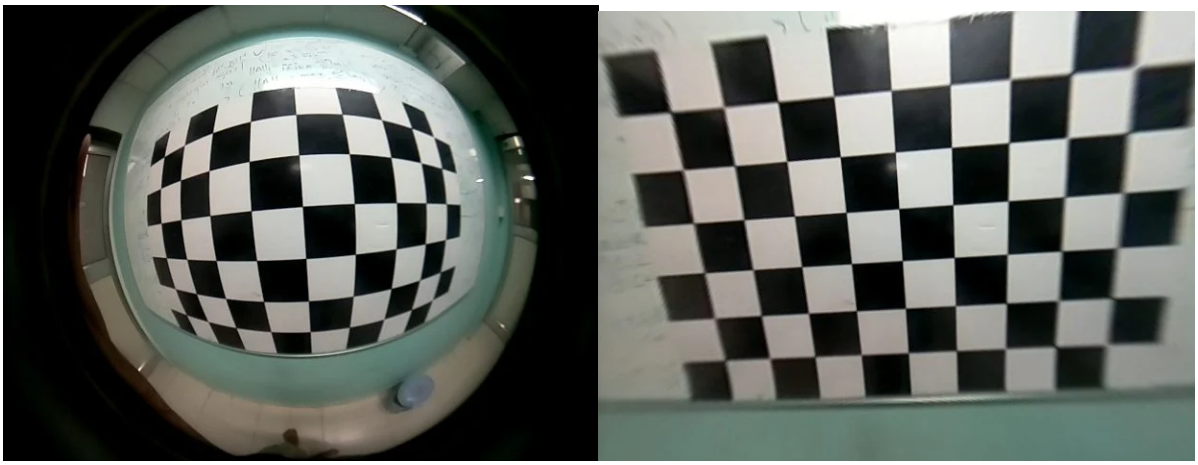


Figure 4.1: Fiducial tags comparison [38]

4.1. Detection Framework

To detect fiducial tags in the monocular camera's image, the camera must first be calibrated using the ROS camera calibration tool. This calibration is done because the Fiducial marker algorithm needs rectified images to detect tags and determine the position of the tag. The original camera has a wide 160-degree FOV, this is a problem since these wide-angle lenses have big distortion and in our case up to 14.3%. Imagine distortion occurs when straight lines in the image look unnaturally curved as depicted in Figure 4.2a and thus can result in deformed tags in the image that are hard to detect by the algorithm. The camera can be calibrated to obtain information about the camera which can be used to rectify the curved image to remove or reduce the distortions as depicted in Figure 4.2b. These levels of distortions can be hard to rectify, therefore, some distortions will remain. Also, will these distortions result in a calibration mismatch between the camera frame and world frame due to the curvature of the lens. To resolve this mismatch, the 2D LiDAR can be used to determine the relation between detections in the camera frame and the world frame which is identical to the LiDAR frame.

The AprilTag 2 library, a Fiducial marker system, is used to detect tags in the rectified images [38]. The AprilTag 2 algorithm which is the upgraded version for improved efficiency and robustness compared to the original AprilTag system, uses the detections in the rectified images to calculate the position in the camera frame. The AprilTag implementation in ROS utilizes the CPU to process the images which are expensive in computational power. Therefore, the 4K resolution is first reduced to 1088 by 860. The AprilTag algorithm would still produce results at a lower frequency than 10Hz. By cropping the image to 1088 by 400 and leaving the pixels where detections are possible as depicted in Figure 4.3, the frequency can be increased to 14Hz. Now that the position in the camera frame is known, the 2D point cloud from the LiDAR can be used to determine the position of the tag in the LiDAR frame and thus in the world frame. The PCL library [31] can be used to identify clusters in the 2D point cloud, which represents objects. Using an open area with only one cluster close by, the tag location and the difference between the location in the camera frame and the LiDAR



(a) (b)

Figure 4.2: (a) Image with big distortions. (b) The same image rectified

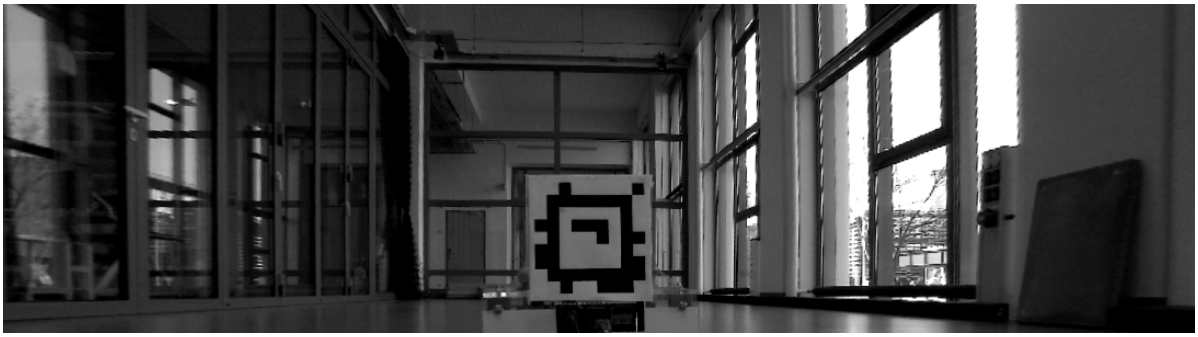
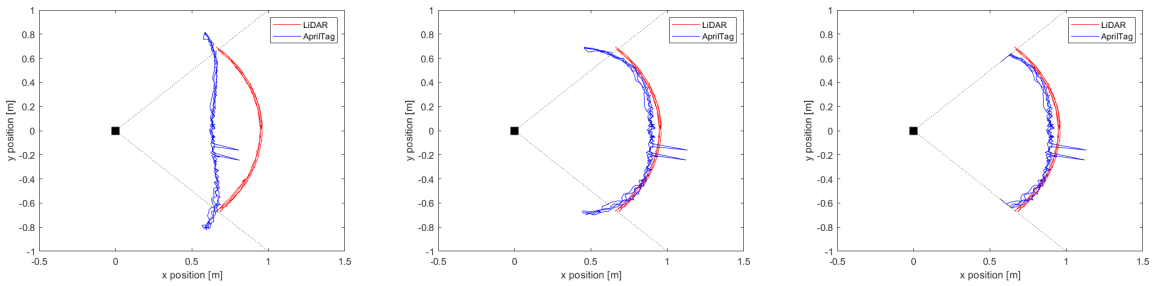


Figure 4.3: The cropped image was taken by one of the R2C Cars with a resolution of 1088 by 400.

frame can be captured. One such example is depicted in Figure 4.4a, whereby the camera was rotated around its origin. One can see that the tag moves linearly in the camera frame instead of curved around the origin. Therefore, the curvature needs to be returned to the camera frame. Note that the tag doesn't move perfectly linearly, this is due to the inaccuracies when calibrating with high distortions.



(a) (b) (c)

Figure 4.4: (a) Initial tag and cluster position detections when the camera is rotated around its center of origin. (b) The curvature of the wide-eyed lens returned to the tag location. (c) Impossible tag locations moved to the border of the possible detection region

From the example in Figure 4.4, the tag's x position can be used together with the rectified image FoV angle ϕ to determine the radius r of the curvature,

$$r = \frac{x}{\cos\frac{\phi}{2}} \tag{4.1}$$

Here, $\phi = 90\,\text{deg}$ determined by the outer cluster center's. The tag's y position determines the length along the circumference of the circle from the $x-axis$, clockwise if positive and anticlockwise if negative. Using the length y and the circumference C,

$$C = 2\pi r \tag{4.2}$$

the position in radians θ can be found on the circumference using,

$$\theta = \frac{2\pi y}{C} = \frac{y}{r} \tag{4.3}$$

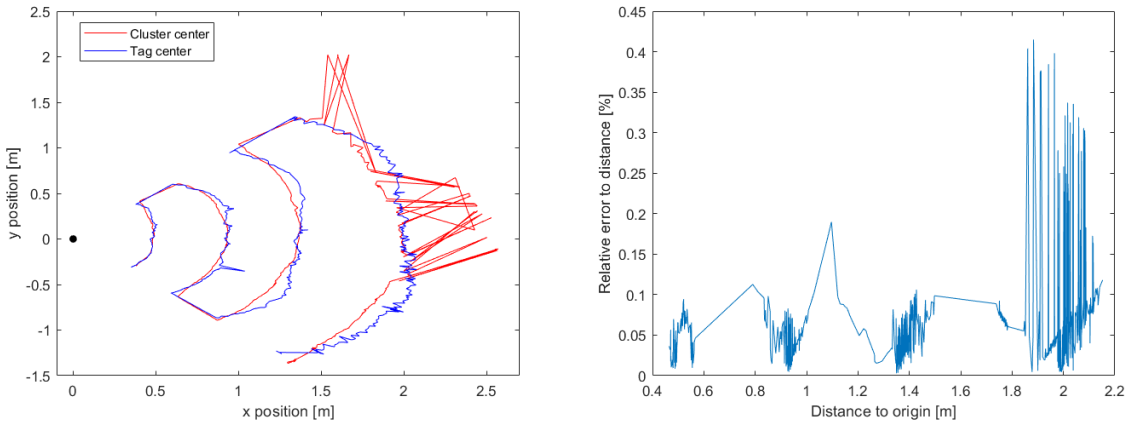
With the radius r and the angle θ, the transformation of the original tag's position can be determined by,

$$x_{new} = r\cos\frac{y}{r} \tag{4.4}$$

$$y_{new} = r\sin\frac{y}{r} \tag{4.5}$$

The transformed AprilTag tag detections are visible in Figure 4.4b. One problem remains, the AprilTag tag detections are outside the FoV range ϕ due to the inaccuracies in the calibration in the presence of high distortions. These detections are radially changed to lay on the border of the FoV, depicted in Figure 4.4c

To retrieve the error between the AprilTag tag location and the cluster center position with increasing distance, a test has been performed whereby the tag moves from side to side slowly increasing the distance relative to the origin. The followed trajectory can be found in Figure 4.5a and the error between tag cluster center position with increasing distance between origin and cluster center position can be found in Figure 4.5b. A problem occurs with detections around 2 meters and further, not enough LiDAR points detect the AprilTag tag location to form a cluster which in our case was set to a minimum of 3 points with a maximum Euclidean distance in between points of 8cm. Therefore, false detections are occurring whereby the cluster center detections are not the AprilTag tag location. For the current camera setup and AprilTag settings and size, 3 meters was the maximum range wherein detections were consistent. Increasing the distance further would cause the AprilTag tag detections to drop exponentially to almost no detections at 3.5m. The frequency setting for the LiDAR in this experiment was set to 8.66Hz, the factory setting, increasing this frequency for faster detections would cause the R2C Car detection range to drop below 2 meters as depicted in the results in Figure 4.5a. In our case is the 2m range where the R2C Car should be accurately detected. From the results can be noted that the maximum range is 2 meters since false detections occurred around 2 meters as depicted in Figure 4.5b.



(a) (b)

Figure 4.5: (a) Trajectory of AprilTag tag position and cluster center during the performed test with a constant LiDAR frequency of 8.5Hz with false cluster detections at a distance of 2 meters. (b) The error between detections with increasing distance between origin and tag detection.

Now that the tag location can be found in the world frame, one could easily use the tag to identify the R2C car in the 2D point cloud where multiple objects have similar dimensions as the R2C Car. An example can be found in Figure 4.6a, and from the point cloud data alone the R2C Car cannot be detected as depicted in

Figure 4.6b. By using the tag location, the R2C Car can already be visually detected in the point cloud data as depicted in Figure 4.6c. For the framework to know which cluster of points is closest to the tag position, the Euclidean distance between the tag center and cluster centers is calculated. The smallest distance defines the correct R2C car detection in the point cloud as depicted in Figure 4.6d.

(a) (b) (c) (d)

Figure 4.6: Setup of one R2C Car identifying and locating another R2C Car. (a) an image of the setup with (b) the corresponding 2D point cloud. (c) The green point locates the AprilTag tag location. (d) The cluster location with the smallest Euclidean distance to the tag location thus identified the correct location of the R2C Car in the point cloud.

4.2. Velocity Measurement framework

The Arduino Nano microprocessor can determine the period in microseconds between the changes in the digital output of both the IR and Hall sensor, as mentioned in the section 3.3.2. By using the details from the Table 4.1, the velocity in m/s can be calculated from the periods T in seconds,

$$v = \frac{D\pi R}{Tn} \tag{4.6}$$

Note that this velocity of the R2C Car in case no slip occurs. Now two problems arise if these velocities are directly used. One problem arises due to the change in frequency of receiving measurements when the velocity changes, an increase in velocity will increase the frequency of receiving measurements due to a smaller period between changes in the digital output. This changing frequency of receiving measurements is hard to use inside control designs. Another problem is that the periods contain noise from inaccuracies in the measurements.

To overcome these problems, a velocity measurement framework is designed in the Arduino Nano that sends out measurements through USB to the Jetson Nano at a constant frequency. This is done by averaging

Wheel diameter D [m]	Differential gear ratio R between shaft and wheels	Detections per revolution n
0.0466	$\frac{11}{39}$	IR -> 20 Magnet -> 8

Table 4.1: Details from the R2C Car

the measured velocities of each sensor over a constant time window, known as the moving average, and then finally calculate the average between the two sensors. This has the advantage of reducing the measurement noise, but still detecting fast changes at a constant frequency.

The width of the time window can be determined by the linear system dynamics of the R2C Car,

$$m\dot{v} = -Cv + F_{wheels} \tag{4.7}$$

Here, m is the mass in kg of the R2C Car, C the drag coefficient $\frac{Ns}{m}$ and F_{wheels} the force N between wheels and the ground applied by the motor. The R2C lab experimentally measured the drag coefficient to be approximately $1.5\frac{Ns}{m}$ by taking the velocity versus time of the R2C Car slowing down without any input on the motor. The mass of the R2C Car was measured to be $1.63Kg$. Now the transfer function can be determined between the velocity and the wheel force,

$$T(s) = \frac{v}{F_{wheels}} = \frac{1}{ms + C} \tag{4.8}$$

The corresponding Bode plot is visible in Figure 4.7. From the plot, it is determined that frequencies above 10Hz keep the system dynamics. In our implementation a constant frequency of 20Hz is chosen that will reduce the noise while keeping all the system dynamics and a low time delay between measurement and actual data.

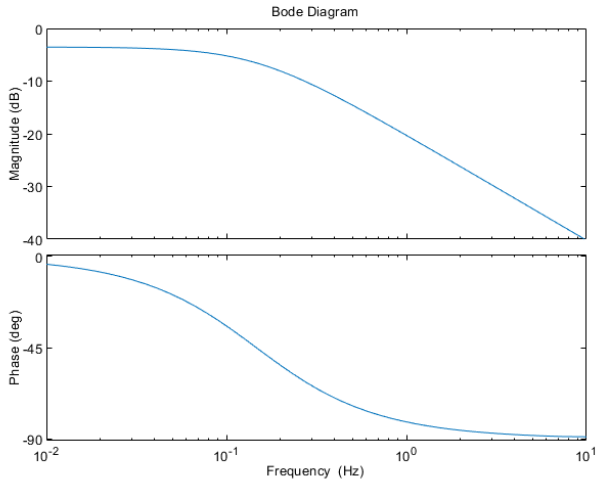


Figure 4.7: Bode plot of the transfer function 4.8

A PD-controller is designed to convert a reference velocity to throttle input for the motor using the velocity measurements. The error between the measurement and the reference velocity $e(k)$ is,

$$e(k) = v_{ref} - v_{measured} \tag{4.9}$$

This error is used by the controller to determine the throttle input. The PD-control equation becomes,

$$throttle(k) = K_P e(k) + K_D \frac{e(k) - e(k-1)}{\Delta t} \tag{4.10}$$

With the throttle input values limited to the range -0.5 to 0.5 and Δt the time between control loops, in our work $\Delta t = 0.05$. The parameters for the controller are determined in two steps, first, the K_P value was determined by using a few different values until the rise time was within a second. To dampen the oscillations of the resulting under-damped dynamics, different values for K_D were used to dampen the dynamics. After finding these values, the last minor adjustments were made to find the final values $K_P = 0.05$ and $K_D = 0.01$.

5

Experimental Cooperative Controller Design

In this chapter, a longitudinal cooperative controller is proposed for the newly designed RC Car platform, whereby the R2C Cars will drive in a platoon. This topic has been studied by many researchers in the last decades, whereby a group of semi or fully autonomous vehicles drive closely behind each other, maintaining a safe inner distance and closely matching the leader's velocity and maneuvers. One such controller is the previously mentioned CACC, see Figure 5.1. In our research, a platoon will be designed for 3 R2C Cars which will maintain a *constant distance policy* and be cooperatively controlled in the longitudinal direction for the sake of simplicity. Since the R2C Cars are similar, it is assumed that each of the vehicle models is similar, also known as a homogeneous platoon.

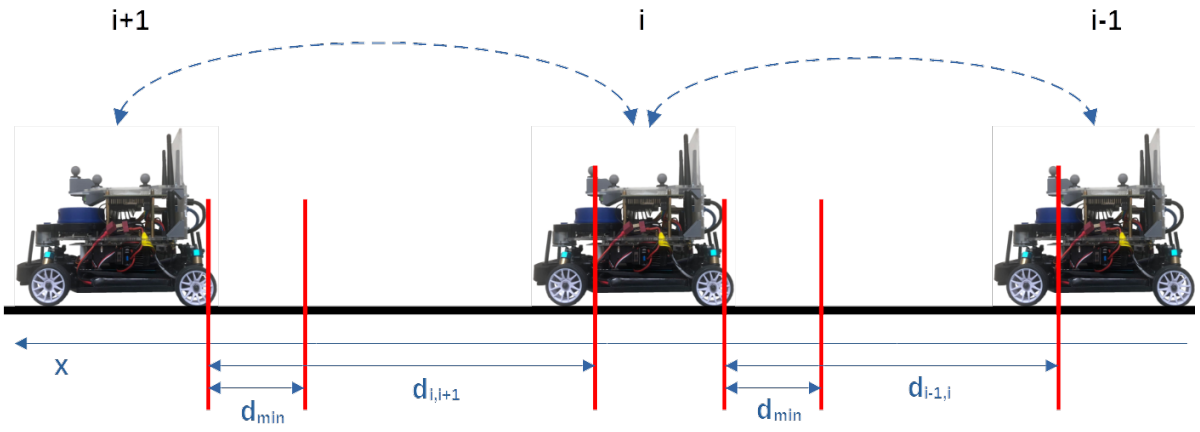


Figure 5.1: Description of a platoon with R2C Cars equipped with CACC. Where the distance between adjacent R2C Cars should be higher than the minimum distance.

One of the most promising cooperative controller designs for platoons, that will also be used in this research, is the Model Predictive Controller (MPC) [20], also known as receding horizon control, which relies on optimization algorithms. This controller design will be first presented. Then, a distributed algorithm is presented for the MPC design whereby each of the R2C Cars computes a part of the optimization problem and by communication finds the optimum. Finally, the controller framework for the R2C Cars is presented.

5.1. Model Predictive Control

Model Predictive Control relies on, generally, the discrete dynamic model of a process or a system and the measured current state [9]. From this current state and the model, future states can be predicted using control

inputs inside of a control horizon[k,k+N]. A cost-minimizing control strategy is implemented inside of this horizon to explore the future states while satisfying constraints. The input horizon [k,k+M], where $M < N$, determines how many of the predicted inputs are used before the MPC proposes new inputs. First the discrete-time dynamic model used for the R2C Cars will be discussed. Then, the cost function will be shown that will be optimized for our control strategy. Next, the constraints will be given which should be satisfied at the end of the optimization. Finally, the optimization problem for the MPC controller will be given.

5.1.1. Discrete Dynamic Model

A widely used discrete-time linear approximated model of longitudinal vehicle dynamics is the double integrator model. This model is used for simplicity in our experiment, which is described as [14] [19],

$$\underbrace{\begin{bmatrix} p_i(k+1) \\ v_i(k+1) \end{bmatrix}}_{x_i(k+1)} = \underbrace{\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} p_i(k) \\ v_i(k) \end{bmatrix}}_{x_i(k)} + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{B} u_i(k) \tag{5.1}$$

Here $p \in \mathbb{R}$, $v \in \mathbb{R}$ and $u \in \mathbb{R}$ denote the longitudinal position, velocity and acceleration. Further, $\Delta t \in \mathbb{R}^+$ denotes the sampling time of the discrete-time model and $i \in \mathbb{I}_M := \{1, ..., M\}$ the set with a total number of M R2C Cars. Future States can be expressed using equation 5.1, the current state $x_{i,0}$ and future inputs u_i,

$$x_i(1) = Ax_{i,0} + Bu_i(1) \tag{5.2}$$
$$x_i(2) = A^2x_{i,0} + ABu_i(1) + Bu_i(2) \tag{5.3}$$
$$x_i(3) = A^3x_{i,0} + A^2Bu_i(1) + ABu_i(2) + Bu_i(3) \tag{5.4}$$
$$\vdots \tag{5.5}$$

Future states in the control horizon, $\bar{x}_i$, using the input sequence $\bar{x}_i$ can then be expressed as,

$$\underbrace{\begin{bmatrix} x_i(1) \\ x_i(2) \\ \vdots \\ x_i(N) \end{bmatrix}}_{\bar{x}_i} = \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{\bar{A}} + \underbrace{x_i(0)}_{x_{i,0}} + \underbrace{\begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}}_{\bar{B}} \underbrace{\begin{bmatrix} u_i(1) \\ u_i(2) \\ \vdots \\ u_i(N) \end{bmatrix}}_{\bar{u}_i} \tag{5.6}$$

Here, $\bar{x}_i \in \mathbb{R}^{2N}$ and $\bar{u}_i \in \mathbb{R}^N$. The matrices and vectors $A \in \mathbb{R}^{2Nx2}$, $B \in \mathbb{R}^{2NxN}$ are not R2C Car specific since a homogeneous platoon is considered. This model for future states can now be used to determine the cost function that should be minimized.

5.1.2. Cost function

Each of the R2C Cars should try to drive at a reference velocity $v_{\text{ref},i} \in \mathbb{R}^+$ inside of the platoon. By setting the reference velocity of the leader $i = 1$ lower than the followers $i > 1$, the followers can keep up with the leader and thus maintain a platoon formation. So the difference between $v_i[k]$ and $v_{\text{ref},i}$ and, to prevent unwanted high acceleration and deceleration, the input $u_i[k]$ should be minimized, creating the cost function $V_i(\bar{z}_i, \bar{u}_i)$ for control horizon N,

$$V_i(\bar{z}_i, \bar{u}_i) = \frac{1}{2}\bar{z}_i^T Q \bar{z}_i + \frac{1}{2}\bar{u}_i^T R \bar{u}_i \tag{5.7}$$

Here, $Q \geq 0 \in \mathbb{R}^{NxN}$ and $R \geq 0 \in \mathbb{R}^{NxN}$ positive semi-definite and symmetric, state and input weight matrices. Also, where,

$$\bar{z}_i = \bar{v}_i - \bar{v}_{\text{ref},i} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}}_{E_v} \bar{x}_i - \underbrace{\begin{bmatrix} v_{\text{ref},i} \\ v_{\text{ref},i} \\ \vdots \\ v_{\text{ref},i} \end{bmatrix}}_{\bar{v}_{\text{ref},i}} \tag{5.8}$$

By incorporating the velocity equality equation 5.6 into the cost function, the cost function can be rewritten to be only depending on the inputs $\bar{u}_i$,

$$V_i(\bar{u}_i) = \frac{1}{2}\left(E_v\bar{A}x_{i,0} - \bar{v}_{\text{ref},i} + E_v\bar{B}\bar{u}_i\right)^T Q\left(E_v\bar{A}x_{i,0} - \bar{v}_{\text{ref},i} + E_v\bar{B}\bar{u}\right) + \frac{1}{2}\bar{u}^T R\bar{u} \tag{5.9}$$

$$V_i(\bar{u}_i) = \frac{1}{2}\bar{u}^T \underbrace{\left(\left(E_v\bar{B}\right)^T Q\left(E_v\bar{B}\right) + R\right)}_{H}\bar{u} + \underbrace{\left(\left(E_v\bar{A}x_{i,0} - \bar{v}_{\text{ref},i}\right)^T QE_v\bar{B}\right)}_{f^T}\bar{u} + \underbrace{\left(\left(E_v\bar{A}x_{i,0} - \bar{v}_{\text{ref},i}\right)^T Q\left(E_v\bar{A}x_{i,0} - \bar{v}_{\text{ref},i}\right)\right)}_{c} \tag{5.10}$$

Since c is a constant which can't be optimized, it can be removed from the optimization cost function.

5.1.3. Constraints

The prevent the first from driving higher than its reference velocity, a maximum velocity will be determined equal to the reference velocity. This increase in velocity above the reference velocity is caused by the follower R2C Cars since the optimization tries to minimize the error between the velocity and reference velocity in all the R2C Cars. When the followers are slowing down the leader will speed up, eventually finding an equilibrium between the errors in the leader and the followers. In our implementation, the leader will be given the platoon velocity and therefore may not drive faster. The constrain for the prediction horizon N becomes,

$$\bar{v}_{leader} \leq \bar{v}_{\text{ref},leader} \tag{5.11}$$

Again using the equality equation 5.6, the constrain can be written in terms of the inputs $\bar{u}_i$,

$$E_v\bar{B}\bar{u}_{leader} + E_v\bar{A}x_{leader,0} - \bar{v}_{\text{ref},leader} \leq 0 \tag{5.12}$$

The R2C Cars also have an input constrain, depicting the acceleration and deceleration range,

$$u_{\min} \leq u_i(k) \leq u_{\max} \tag{5.13}$$

For the time horizon N, this becomes,

$$\bar{u}_i - \bar{u}_{\max} \leq 0 \tag{5.14}$$

$$-\bar{u} + \bar{u}_{\min} \leq 0 \tag{5.15}$$

Now to prevent collisions in the platoon since the leader R2C Car has a lower reference velocity, the distance between two adjacent cars should always be bigger or equal to a minimum distance $d_{\min}$, see Figure 5.1. From the sensors we receive the distance between two adjacent vehicles and not the position of the two, so instead of using an initial position each car calculates how it changes the initial distance over time. The distance $\bar{d}_{i,i+1} \in \mathbb{R}^+$ in the time horizon is described by,

$$\bar{d}_{i,i+1} = E_p\bar{A}\left(x_{i,0} - x_{i+1,0}\right) + E_p\bar{B}\left(\bar{u}_{i+1} - \bar{u}_i\right) \tag{5.16}$$

with,

$$E_p = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & & & \\ 0 & 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix} \tag{5.17}$$

These distances should be bigger then the minimum distance $d_{\min}$, creating the inequality constrain,

$$-E_p\bar{A}\left(x_{i,0} - x_{i+1,0}\right) + d_{\min} - E_p\bar{B}\left(\bar{u}_{i+1} - \bar{u}_i\right) \leq 0 \tag{5.18}$$

Note that only the leader $i = 1$ and the last R2C Car $i = M$ have to satisfy one constrain and all others have to satisfy both front and rear distance constrain.

5.1.4. Optimization Problem

Now that all constrains are specified and the cost function is known, the quadratic optimisation problem with linear constrains can be specified for each R2C Car. The problem becomes,

$$\underset{\bar{u}_i}{minimize} \quad \frac{1}{2}\bar{u}_i^T H \bar{u}_i + f^T \bar{u}_i \tag{5.19}$$

$$s.t. \quad E_v \bar{B}\bar{u}_i + E_v \bar{A} x_{i,0} - \bar{v}_{\text{ref},i} \leq 0 \tag{5.20}$$

$$\bar{u}_i - \bar{u}_{\max} \leq 0 \tag{5.21}$$

$$-\bar{u}_i + \bar{u}_{\max} \leq 0 \tag{5.22}$$

$$-E_p \bar{A}\left(x_{i,0} - x_{i+1,0}\right) + d_{\min} - E_p \bar{B}\left(\bar{u}_{i+1} - \bar{u}_i\right) \leq 0 \tag{5.23}$$

$$-E_p \bar{A}\left(x_{i-1,0} - x_{i,0}\right) + d_{\min} - E_p \bar{B}\left(\bar{u}_i - \bar{u}_{i-1}\right) \leq 0 \tag{5.24}$$

Note that for the sake of simplicity only the problem for R2C Cars in the middle is shown. For the leader and last R2C Car, one of the distance constraints can be removed and the leader has a velocity constrain. The quadratic cost function is convex and with linear constrains has convergence guarantees.

5.2. Distributed Optimisation

The optimization problem is coupled with the R2C Cars through the distance constraints. Therefore, each R2C Car can perform a local optimization but has to communicate with adjacent R2C Cars to update the distance constraints. One easy implementable method for distributed optimization is the *Primal-Dual Method of Multipliers* (PDMM) [8]. The PDMM is based on the method of Lagrange multipliers, which incorporates constraints into the cost function by a penalty function. This new cost function is known as the Lagrangian function. First, the Lagrangian function will be presented for the optimization problem 5.19. Then, the PDMM algorithm will be described. Finally, the DMPC framework will be presented for the R2C Cars.

5.2.1. Lagrange Formulation

Again for explaining purposes, a R2C Car is chosen which has two distance constrains on each side. For the leader of a platoon the distance constrain in front can be removed and for the last R2C Car in the platoon the distance at it's rear can be removed. Each of the constrains in the optimisation problem 5.19 get a dual variable assigned $\mu_{j,i} \geq 0 \in \mathbb{R}^+$ for $j := \{1, ..., 5\}$,

$$\underset{\bar{u}_i}{min} \quad \frac{1}{2}\bar{u}_i^T H \bar{u}_i + f^T \bar{u}_i \tag{5.25}$$

$$s.t. \quad \mu_{1,i} : g_{1,i} = E_v \bar{B}\bar{u}_i + E_v \bar{A} x_{i,0} - \bar{v}_{\text{ref},i} \leq 0 \tag{5.26}$$

$$\mu_{2,i} : g_{2,i} = \bar{u}_i - \bar{u}_{\max} \leq 0 \tag{5.27}$$

$$\mu_{3,i} : g_{3,i} = -\bar{u}_i + \bar{u}_{\min} \leq 0 \tag{5.28}$$

$$\mu_{4,i} : g_{4,i} = -E_p \bar{A}\left(x_{i,0} - x_{i+1,0}\right) + d_{\min} - E_p \bar{B}\left(\bar{u}_{i+1} - \bar{u}_i\right) \leq 0 \tag{5.29}$$

$$\mu_{5,i} : g_{5,i} = -E_p \bar{A}\left(x_{i-1,0} - x_{i,0}\right) + d_{\min} - E_p \bar{B}\left(\bar{u}_i - \bar{u}_{i-1}\right) \leq 0 \tag{5.30}$$

Using Lagrangian augmentation, the cost function is augmented by a weighted sum of the constraints and their Dual variables. The Lagrangian becomes,

$$L(\bar{u}_i, \mu_{j,i}) = \frac{1}{2}\bar{u}_i^T H \bar{u}_i + f^T \bar{u}_i + \mu_{1,i}^T g_{1,i} + \mu_{2,i}^T g_{2,i} + \mu_{3,i}^T g_{3,i} + \mu_{4,i}^T g_{4,i} + \mu_{5,i}^T g_{5,i} \tag{5.31}$$

5.2.2. Primal-Dual Method of Multipliers

The PDMM algorithm uses the gradient of the Lagrangian in respect to the input $\bar{u}_i$, known as *primal gradient*, and to the dual variables $\mu_{j,i}$, known as *dual gradient*. Starting with the Primal gradient,

$$\frac{\partial L(\bar{u}_i, \mu_{j,i})}{\partial \bar{u}_i} = H\bar{u}_i + f + \left(E_v \bar{B}\right)^T \mu_{1,i} + \mu_{2,i} - \mu_{3,i} + \left(E_p \bar{B}\right)^T \mu_{4,i} - \left(E_p \bar{B}\right)^T \mu_{5,i} \tag{5.32}$$

The Primal gradient update can be written as,

$$\bar{u}_i^+ = \bar{u}_i + \alpha \frac{\partial L(\bar{u}_i, \mu_{j,i})}{\partial \bar{u}_i} \tag{5.33}$$

Here, α denotes the step size for updating the inputs $\bar{u}_i$. Note that this update step is local and doesn't need any information from other race cars. The gradients from the Lagrange with respect to the Dual variables are,

$$\frac{\partial L(\bar{u}_i^+, \mu_{j,i})}{\partial \mu_{j,i}} = g_{j,i} \tag{5.34}$$

The Dual gradient update function can be written as,

$$\mu_{j,i}^+ = \mu_{j,i} + \beta \frac{\partial L(\bar{u}_i^+, \mu_{j,i})}{\partial \mu_{j,i}} = \mu_{j,i}^+ = \mu_{j,i} + \beta g_{j,i} \tag{5.35}$$

Here, β denotes the step size for updating the dual variables. The two constrains $g_{i,4}$ and $g_{i,5}$ need information that the R2C Car i does not possesses. The constrain $g_{i,4}$ needs to now how the input of the adjacent R2C Car in front changes the distance between R2C Car $i+1$ and i inside of the time horizon. The same holds for the constrain $g_{i,5}$ where the information from the adjacent R2C Car from behind is needed. Both adjacent R2C Cars should communicate their initial velocity $v_{i-1,0}$ and $v_{i+1,0}$ and their updated input vector for the time horizon N, u_{i-1} and u_{u+1}. Finally, R2C Car $i-1$ should communicate the initial distance $p_{i-1}(0) - p_i(0)$, since R2C Car i can only measure the initial distance between R2C Car $i+1$ and i. Now that the dual variables are updated, the whole process of Primal and Dual gradient updates can be repeated until the change in the updates has decreased below a certain value, a maximum number of iterations is reached, or when a time limit is reached.

5.3. Proposed Distributed Model Predictive Control Framework

The DMPC algorithm uses acceleration as input to the system, but this cannot be used for the R2C Car. Therefore, instead of the acceleration as input, the velocity predicted for the next time step is used as a reference velocity for the PD-controller. To determine the maximum acceleration and deceleration, a square wave throttle input signal of 0.2 to -0.2 was used together with an IMU. Higher throttle signals caused the wheels to slip and therefore the maximum acceleration and deceleration found were 3 and -3 m/s^2. In our work the input and state matrices, Q and R, contain all ones on the diagonal. The parameters for the time Horizon N are set to 4 seconds, a control horizon M of 1, and the sampling time Δt is set to 0.1 seconds. The update parameters for the PDMM algorithm α and β were set to respectively 0.02 and 2. These values were tuned during testing to find the optimum as fast as possible without being unstable. The minimum distance, as depicted in Figure 5.1, between R2C Cars was set to 25 cm.

The DMPC algorithm will run at 10 Hz in each R2C Car. Each R2C Car will start the algorithm when a start signal is given. After the first local optimization update, each R2C Car will send out a package containing: the distance between itself and its front neighbor, the change in position in the time horizon N, the iteration number, and the vehicle identification number. When the message is received by adjacent R2C Cars, again a local optimization update is done and a package is sent again. For an R2C Car with two adjacent R2C Cars, it waits until both messages have arrived and then performs the local optimization update. So all R2C Car performs a local optimization update almost parallel since they have to wait for the correct package but differences in calculation time can cause differences in the exact start of each of the local optimization updates. The optimization is done after $0.1s$, it will reset the iteration number, send out the velocity for the next time step as a reference velocity, and will continue with a new optimization. The problem with only checking the time after each optimization update is that there are minor differences in milliseconds between the R2C Cars stopping time and this will lead to drifting after several completed optimizations. So when each package arrives from a neighbor, it first checks if the other R2C Car started a new optimization and responds by also restarting if it is true. The full behavior tree can be viewed in Figure 5.2.

The DMPC algorithm will send out the reference velocity after each optimization which is used by the velocity PD controller. After each completed optimization, the DMPC will update the distance parameter using the latest distance measurement from the sensors. The full flow chart can be viewed in Figure 5.3.

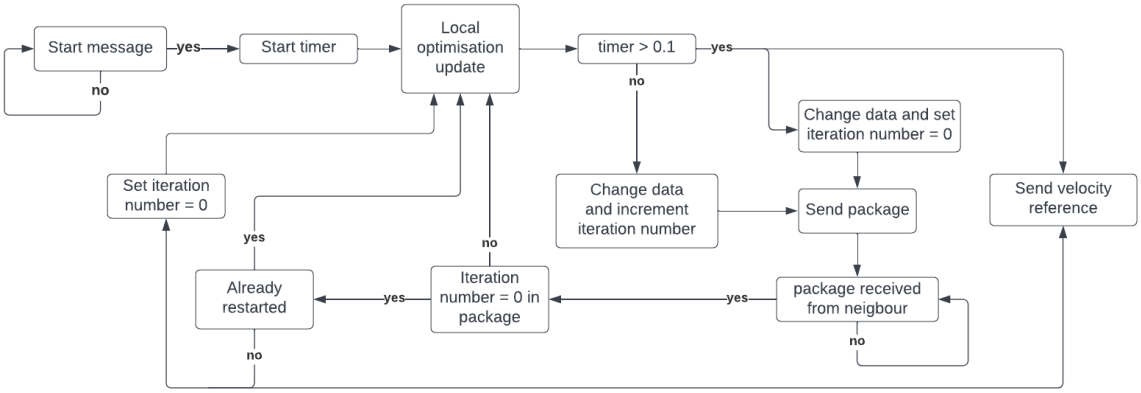


Figure 5.2: Behavior tree for the DMPC algorithm

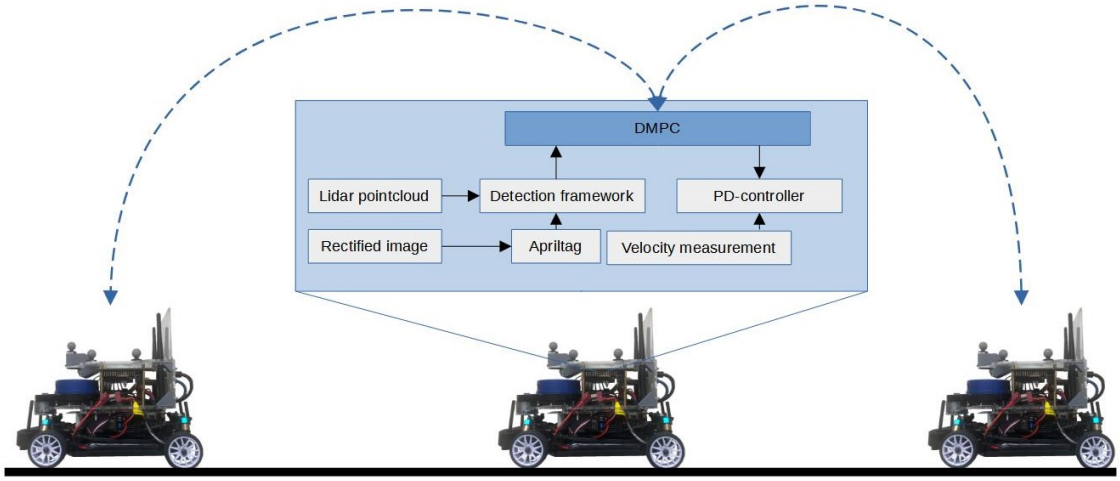


Figure 5.3: The longitudinal control flow chart for the R2C Car platoon

6

Experiments

This chapter presents the performance of the R2C Cars using the cooperative controller framework presented in the section 5.3. First, the performance of the communication between R2C Cars is discussed using different scenarios. Then, the cooperative driving experiment will be discussed whereby the performance of the R2C Car will be tested.

6.1. Communication

One of the most important aspects of a DMPC is the network connection. The delay in a network constrains the number of iterations that can be performed inside a time window. To get more insight into network stability and the minimum delay when communicating messages using the ROS environment, a communication experiment is performed that utilizes two R2C Cars for simplification purposes. The results obtained will give some insight into network stability and the minimum delay when communicating messages using the ROS environment. The DMPC framework is used to show how the controller utilizes the network to find the optimum between R2C Cars. Getting the exact delay of one message communicated between R2C Cars during each iteration is difficult due to clock differences between machines. Therefore, the average delay is calculated inside one completed optimization step. This is derived from the number of iterations, the time a message was processed, and the total time of one completed optimization step.

The DMPC framework utilizes the 5Ghz WiFi of a TP-link AC2300 Wireless MU-MIMO, Multi-User and Multi Input Multi Output, router which can process the messages of 4 different devices in parallel. This router was set to work as fast as possible by turning off security measurements. The experiment consist of three different scenarios:

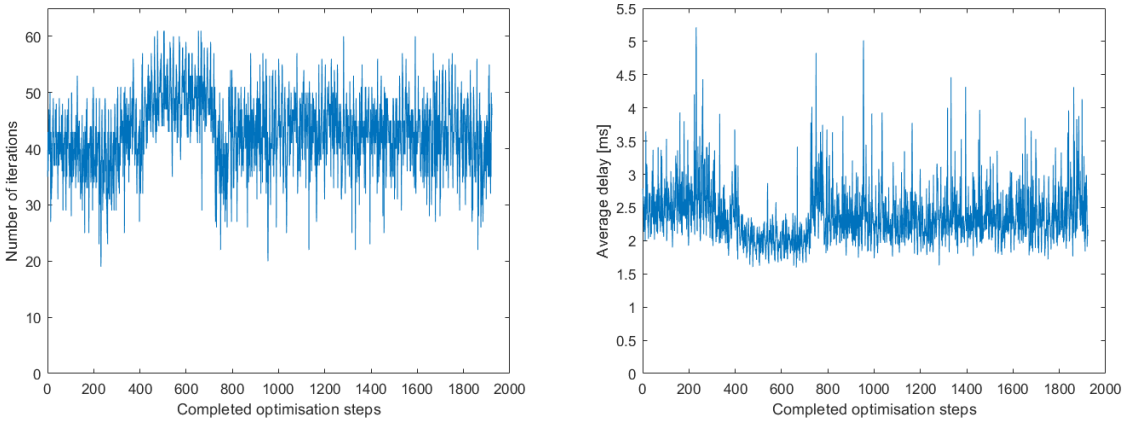
- **Minimal distance setup:**
  The two R2C Cars are within 1 meter of each other and the router, minimizing the delay. The two R2C Cars are the only devices utilizing the network.

- **Minimal distance with extra communication:**
  During the communication between the R2C Cars, a laptop is streaming a large amount of data to maximize the processing power of the router.

Note that in all cases a laptop is connected to the router, this laptop is running the Roscore that connects processes in the ROS environment. However, it only connects the processes when initialized and does not interfere until shutdown. Also, the ESC of the R2C Cars is not powered on so each of the R2C Cars is not moving when the tests were performed. Moving or standing still will not affect the delay directly, therefore standing still is chosen for simplicity.

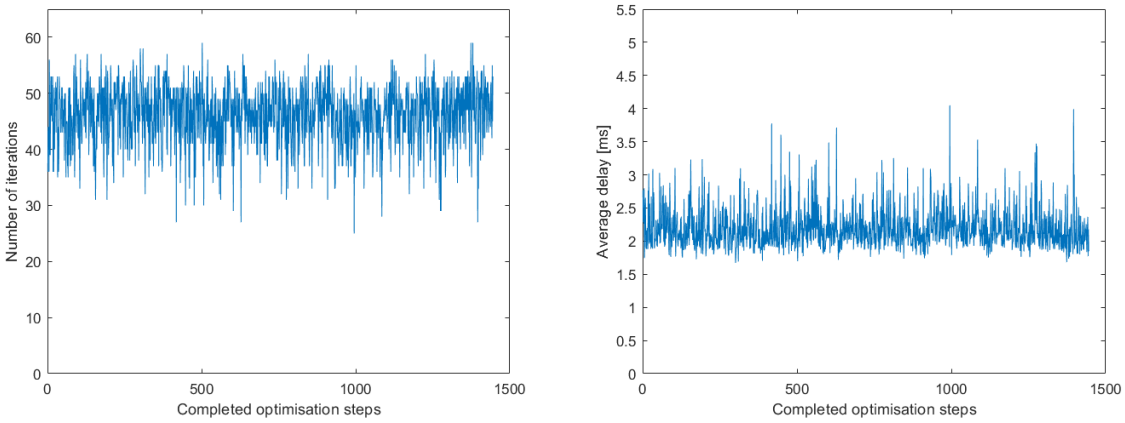
The iteration numbers and the average delay for full optimizations can be found in Figures 6.1, and 6.2. From the results, it can be noted that communication is not constant but heavily fluctuates. Heavy streaming on the Laptop did not increase the delay, due to the MU-MIMO. One should consider these network imperfections when designing a distributed controller. Additional reasons for a higher delay or package loss can be due to a clouded bandwidth with multiple other devices utilizing the same bandwidth. If more than 4 devices are connected to the router, messages will be processed as First-In-First-Out and an extra delay can

be formed. Message size can also become a problem, but it must contain a lot of data. In our case with a time horizon, N of 4 seconds the package contained 176 Bytes, using floating points that are 4 Bytes in length each. The network speed of the 5Ghz WiFi is 1625Mbps or 203MBps, this is divided over the parallel channels when more than one device is connected. With a maximum of 4 channels, the network speed is 51MBps which leaves us with a message delay of 3.4μs. This delay can be neglected when considering the delay from the two tests, therefore the delay comes from the process of sending the data and getting it to the right receiver and not from the speed itself.



(a) (b)

Figure 6.1: The completed optimization step information for the minimal distance setup with (a) the number of iterations and (b) the average delay of the messages.



(a) (b)

Figure 6.2: The completed optimization step information for the minimal distance with extra communication setup with (a) the number of iterations and (b) the average delay of the messages.

6.2. Cooperative Driving

The cooperative driving architecture can be found in Figure 6.3. The Optitrack is used to track the positions of the three R2C Cars. The positions of each of the R2C Cars are sent to the laptop and together with the reference path determine the steering input for each R2C Car. The reference path is made out of points that form an oval path. The Steering input is determined in a simple way, the previous and the new position determine the heading of the car, and this heading is used to rotate the reference track in the R2C Car frame with positive y in front of the car as depicted in Figure 6.4. The steering would be aggressive or non-smooth

if the closest reference point in front is chosen, therefore, a reference point at a minimum of $y = 0.5$ distance is chosen. The DMPC will benefit from this slow steering behavior since lateral changes are not predicted by the controller but will affect the distance measurement and thus the controller. Finally, the angle between the heading and the closest reference point is used to determine the steering proportionally.

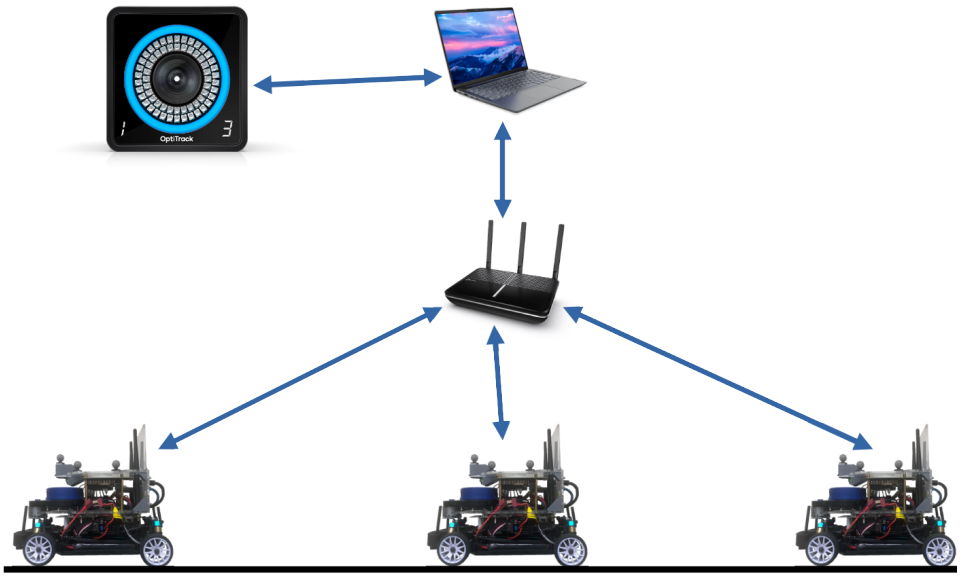


Figure 6.3: Cooperative driving architecture

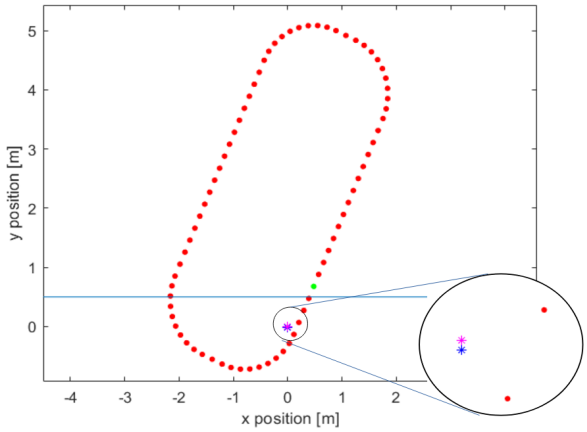
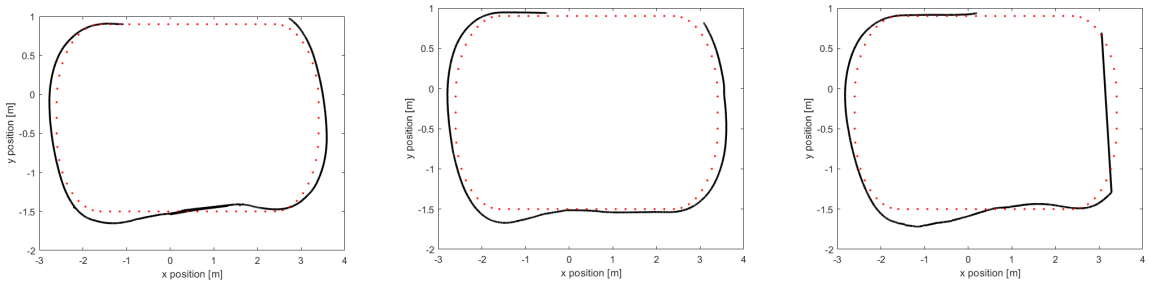


Figure 6.4: The reference track rotated in the R2C Car frame is determined by the previous position (blue star) and the new position (purple star), the blue line denotes the search constrain, and the green reference point is used to determine the steering input.

The trajectories of the three cars during the experiment can be found in Figures 6.5a, 6.5a, and 6.5a. The R2C Cars drive anti-clockwise following the reference path. Notice the strange behavior of the last R2C Car at the end of the trajectory. Here, the Optitrack had lost the visual markers of the R2C Car. Further inconsistencies of the Optitrack are depicted in the distance measurements in Figure 6.6, between the leader and middle R2C Cars around 17 seconds. Relatively high differences in the distances occur while the onboard distance measurements measure no such thing. These inconsistencies can occur when the lighting changes in the room with the Optitrack since these changes can cause problems in the detections.

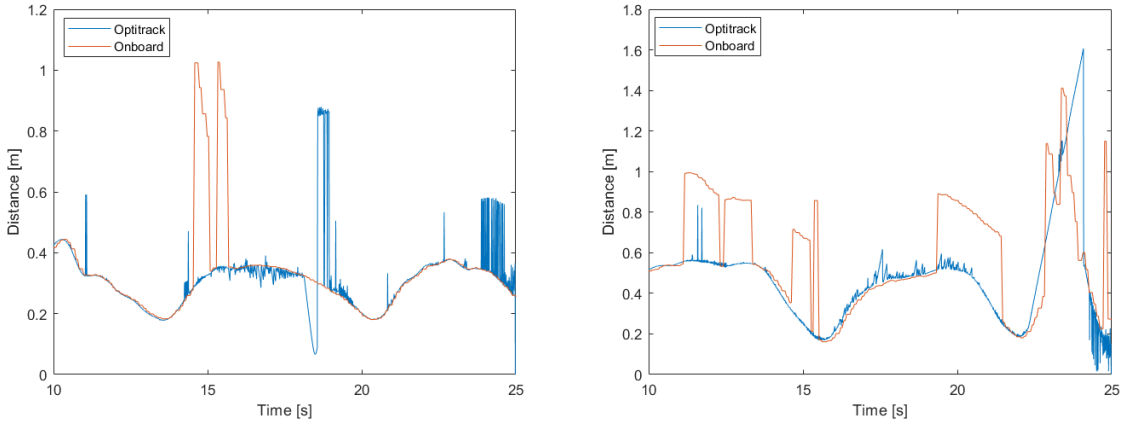
Our detection framework measures the distance accurately, compared to the Optitrack data but false detections are occurring. These false detections cause a relatively high error in the measured data. The data



(a) (b) (c)

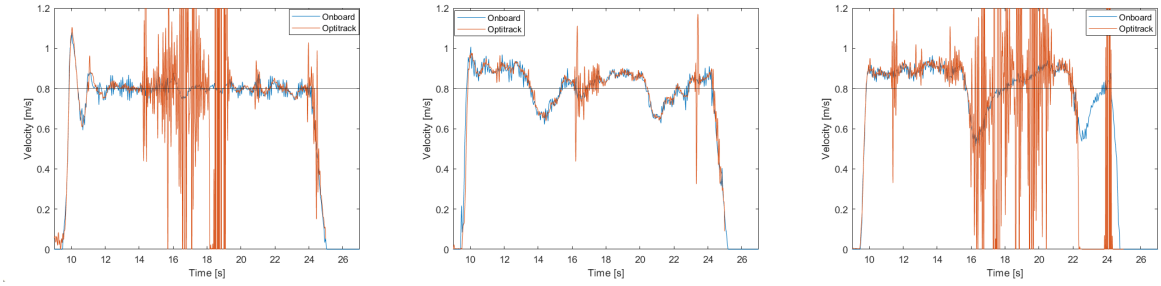
Figure 6.5: The trajectories of the R2C Cars go anti-clockwise with the reference path for (a) the first R2C Car, (b) the middle R2C Car, and (c) The last R2C Car.

measured is used by the DMPC framework to predict and optimize future states, therefore, the error also exists in the output of the DMPC. This is unwanted and can cause collisions as can be seen from the resulting minimal distances which lay well below the minimal distance setting of 0.25. When these inconsistencies are addressed, the detection framework can be perfectly suited for cooperative driving experiments. Furthermore, the velocity measurement framework performs as expected when compared to the Optitrack as depicted in Figure 6.7. Notice the high noise in the filtered data from the Optitrack due to the inconsistencies in the Optitrack detections, except for Figure 6.7b. The filtered data was obtained by using a low-pass filter at 10 Hz instead of 20 Hz due to the high amount of noise. In Figure 6.7b, the consistency between the measured data by the optitrack and by the velocity measurement framework can be seen. The PD-controller worked as expected as depicted in Figure 6.8. The reference velocity is followed by each of the R2C Cars without any inconsistencies but notices the underdamped behavior in the leader R2C Car. The middle R2C Car also has some minor underdamped behavior, but both leader and middle R2C Car need some further tuning of the parameters. The PD-controller parameters are thus R2C Car specific and need to be readjusted for each R2C Car. The DMPC framework created a platoon of R2C Cars, even with the inconsistencies in the distance detections. The constraints were violated but this can be related due to the detection inconsistencies. Therefore, further testing needs to be performed but it showed the possibilities for distributed controller designs.



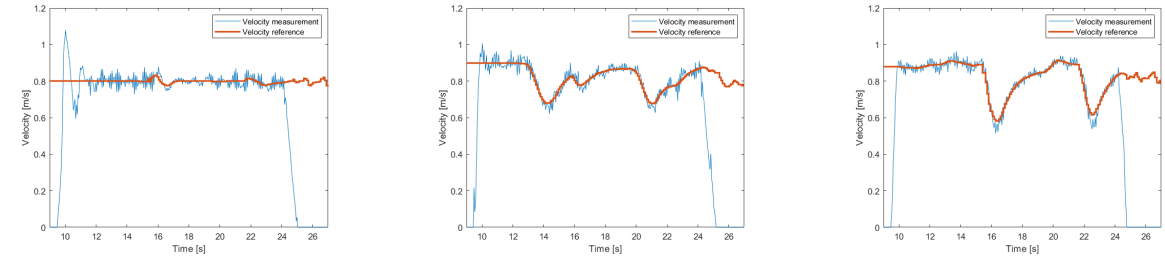
(a) (b)

Figure 6.6: The distance between two adjacent R2C Cars was measured by the onboard detections framework and the Optitrack. (a) between the first and the middle R2C Cars, and (b) between the middle and the last R2C Cars.



(a) (b) (c)

Figure 6.7: The velocities between two adjacent R2C Cars were measured by the onboard velocity framework and the Optitrack. for (a) the leader R2C Car, (b) the middle R2C Car, and (c) The last R2C Car



(a) (b) (c)

Figure 6.8: The velocities of the R2C Cars with their reference velocities obtained using the DMPC for (a) the leader R2C Car, (b) the middle R2C Car, and (c) The last R2C Car

7

Discussion and Future Work

This chapter discusses the limitations and proposes possible solutions in future work. The R2C Car and the cooperative controller, each, have their discussion section.

7.1. R2C Car

- **System Identification:**
  The R2C Car can be controlled using a reference velocity by adjusting the input to the motor and reading the velocity from the encoders. An improved controller could be made if the system dynamics are known. One example is that the R2C Car from a standstill doesn't respond to inputs in the range of -0.13 to 0.13 and when driving, the velocity will slowly decrease to zero for inputs in this range. This can negatively influence the controllers since they can become unstable in this range, due to the R2C Car not responding to inputs. By doing system identification, knowledge can be obtained to reduce these instabilities.

- **Pinion gear:**
  Currently, the motor has a hard time driving at lower velocities around $0.5\frac{m}{s}$. By lowering the number of teeth on the pinion gear, the torque needed to drive is lowered making it easier to drive at lower velocities. Note that by doing so, the maximum velocity will decrease.

- **Velocity measurement:**
  The interferential optical sensor has a higher detection rate and better accuracy due to a smaller gate than the reflective optical sensor, but it needs a wheel with holes that couldn't be placed in the R2C Car due to constraints in the available space. Although there wasn't a possible solution found in our research, more research could be done to find a solution. Also, the current reflective optical and Hall sensor could be researched more to further optimize the velocity measurement accuracy. Furthermore, during high acceleration and deceleration, there may occur slip between the tires and the ground. The RPM measurement from the main axle would then be useless to calculate the velocity of the R2C Car itself. Therefore, an Inertial Measurement Unit (IMU) can be placed on the R2C Car to help detect the occurrence of slip since it can measure the acceleration of the body of the R2C Car.

- **Offloading AprilTag algorithm to GPU:**
  The AprilTag algorithm is running on a highly inefficient CPU. The GPU can process images at a much higher and more efficient rate due to its parallel computation properties. And now that the images can be processed more easily, the size of the images can be increased. This will increase the accuracy in position estimation and tags detection range.

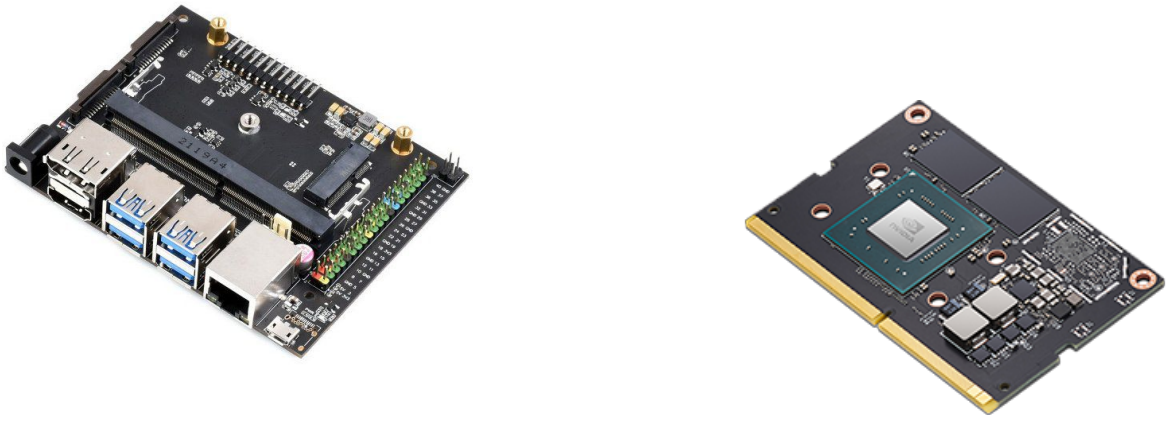
- **Wide angle camera calibration:** The 160°FoV monocular camera has a high distortion due to its wide-angle lens. It is hard to calibrate these cameras and therefore, AprilTag cannot determine the accurate position of the tag. One way to improve the accuracy of the AprilTag tag detections is to create a framework that recalibrates the camera by minimizing the error between AprilTag tag detections and the corresponding point cloud cluster detections of the 2D LiDAR.

- **Vehicle-to-Vehicle communication:**
  In our research, a TP-Link router was used to create the local 5GHz WiFi network. Communication between the R2C Cars is done via the router, making it highly inefficient since in most cases R2C Cars are closer to each other than closer to the router and the router has to process the data that is being sent. Instead, a device-to-device communication [35] could be utilized where the router acts as a link controller but the data is sent from R2C Car-to-R2C Car. One implementation could be to use WiFi extenders that are connected to the Jetson Nano boards using the UTP connection and connect them using the main router and creating a mesh, but further research is needed for this implementation.

- **Jetson modules:**
  The JetRacer Pro AI is made to be used with the Jetson Nano board. The Jetson Nano consists of the carrier board in the Figure 7.1a and the Jetson Nano module 7.1b. The Jetson Nano module is the brain of the Jetson Nano. Two modules more powerful than the Jetson Nano are the Jetson Xavier NX and Jetson TX2 NX. A small summary of the modules can be found in Table 7.1. All these modules fit on the Jetson Nano carrier board but there are limitations. The Jetson Nano carrier board gets 5v and 2.5A through each of the pins on the J41 header, making a total of 5v at 5A or 25W. Running the Jetson TX2 NX at 15W will leave us with 10W for connected components like the LiDAR which consumes 5W at startup and 2.5W continuous, so in the R2C Car with the current setup, a Jetson TX2 module can be used to increase the computational power. One disadvantage of the Jetson TX2 is that it only contains 16GB of onboard storage and doesn't have an SD card slot like the Jetson Nano module. The Jetson Xavier NX can be placed on the Jetson Nano carrier board, but can only run in the 10W or 15W mode. Therefore, it is not recommended. Also, the Jetson Xavier NX developer kit comes with the module and the Jetson Xavier NX carrier board for only $399, which is cheaper than the module itself. The dimensions are similar, but this carrier board can only be powered through the barrel jack. One solution could be to reconnect the ESC to the expansion board of the R2C Car and connect the Jetson Xavier NX to another power source.

Module	Jetson Nano	Jetson TX2 NX	Jetson Xavier NX
CPU	Quad-core ARM Cortex-A57 MPCore processor	Dual-core NVIDIA Denver 2 64-bit CPU and quad-core ARM A57 Complex	6-core NVIDIA Carmel Arm®v8.2 64-bit CPU 6MB L2 + 4MB L3
GPU	128-core NVIDIA Maxwell™ GPU	256-core NVIDIA Pascal™ GPU	384-core NVIDIA Volta™ GPU with 48 Tensor Cores
Memory	4 GB 64-bit LPDDR4 25.6GB/s	4 GB 128-bit LPDDR4 51.2GB/s	8 GB 128-bit LPDDR4x 59.7GB/s
Power modes	5W | 10W	7.5W | 15W	10W | 15W | 30W
Price	$129	$199	$479

Table 7.1: Three different Jetson modules with their specifications.



(a) (b)

Figure 7.1: (a) The Jetson Nano carrier board (b) The Jetson Nano module

7.2. Cooperative controller

- **Utilizing system dynamics in MPC:** The MPC relies on a simple linear discrete-time state-space system with no dynamics known of the R2C Car itself. An improvement will be to incorporate the dynamics from input to output into the MPC formulation, decreasing the difference between the predictions and real output. This will increase the controller's accuracy and, therefore, decrease the instabilities in the controller design.

- **Add lateral control to the distributed MPC controller:** In this research, only longitudinal control was considered for the MPC formulation due to time constraints. The lateral control in all R2C Cars was done using waypoints on a reference track and the Optitrack. By introducing lateral control in the distributed MPC controller, only the first R2C Car has to be remotely controlled.

- **Obstacle avoidance:** The cooperative controller only forms a platoon and steering of the first car is done manually thus the operator should steer the leader accordingly to avoid collisions with objects. It would be interesting to see how the cooperative controllers behave when obstacle avoidance is added to the distributed MPC formulation.

- **Communication issues:** In this research, communication is assumed to be perfect. This means, there are no package losses or communication failures. In our controller design, package loss or a communication failure would mean that the controller stops abruptly, creating an uncontrollable situation. Also if communication switches from one router to another, a high delay will occur for a small moment in the data stream. Further research is thus necessary for these communication issues to finally create a robust cooperative controller design.

8

Conclusion

This chapter will reflect on the research objectives set out in this thesis work. The research objectives as presented in Chapter 1.

- Redesign and construct a low-cost multi-modal sensor suite for the JetRacer Pro AI to allow it to drive autonomously in cooperative driving experiments.
- Design and implement a framework that can process the sensor data.
- Design a longitudinal cooperative controller for multiple RC cars that proposes an input at a frequency of 10 Hz.
- Create a test setup for testing the performance of the modified JetRacer Pro AI.
- Evaluate the performance of the modified JetRacer Pro AI and the implemented cooperative controller using the test setup.

A custom low-cost sensor suite is created for the first objective. A YDlidar X4 is added on the front of the JetRacer to have a 220°FoV for accurate object detections in its workspace. The 160°FoV monocular camera from the JetRacer is also utilized to detect details about the objects. Therefore, the JetRacer is capable of perceiving its environment front at a low cost. To have feedback from the input to the ESC, that controls the motor, two encoders are added on the sides of the main gear. One reflective optical encoder and one Hall encoder measure the rotational velocity of the main gear, which in turn could be used to calculate the velocity of the JetRacer. By redesigning the original JetRacer Pro AI, the sensors could be placed effectively and all components have a lower risk of getting damaged by crashes. The new name given to the modified JetRacer Pro AI is R2C Car.

For our second objective, The fiducial marker system R2C Car is utilized to detect objects in the images captured by the monocular camera. The AprilTag algorithm returns a 3D location of the tag in the camera frame. To increase the accuracy of the positions of the tag, the 2D point cloud of the LiDAR is used. The point cloud is filtered using the PCL library to form clusters of points that represent objects. The Euclidean distance between cluster centers and tag position is calculated and used to connect the tag position to the closest point cluster and receive an accurate position of the object with the AprilTag tag at a frequency of 8.66Hz.

To show the possibilities of the modified R2C Car for a cooperative driving platform, for the third objective, a cooperative longitudinal controller was designed for multiple R2C Cars. This controller was designed using a distributed MPC formulation with a Primal-Dual gradient algorithm proposing an input at 10 Hz. For the MPC a linear discrete-time state-space model was used for the future predictions. In each of the iterations, local information was exchanged to find the best input for all the connected R2C Cars. The distance between R2C Cars was set as a constrain and updated after each time step to count for any mismatch between predictions from the MPC and real-world implementation. The velocity obtained from the MPC in the next predicted time step is used as a reference velocity for the low-level PD controller in each of the R2C Cars. The PD controller then tries to minimize the error between the reference velocity and the velocity measurements.

The fourth objective was to show the potential of the new cooperative driving platform based on the modified JetRacer Pro AI, The R2C Car. In this objective, a platoon of multiple R2C Cars was created. The first car was controlled using a joystick for the sake of simplicity. By using a small inner distance between the R2C Cars, a good prediction can be done on the performance and stability of the implemented controller structure.

The final objective was to evaluate the performance of the R2C Car and the implemented cooperative controller. It is shown that the R2C Car can detect objects, in our case other R2C Cars, and determine their position in the workspace accurately. There are some consistencies in the detections that need to be addressed but the platform is suitable for cooperative driving experiments. By using a cooperative controller it was verified that the platform of R2C Cars was able to perform these experiments. The performance of the cooperative controller itself was also evaluated to show the possibilities for distributed controller designs.

Finally, the presented platform gives the possibility to research cooperative driving cost-effectively. With the presented distributed controller, knowledge is obtained for future research.

Bibliography

[1] Jetracer pro ai kit, high speed ai racing robot powered by jetson nano, pro version. URL https://www.waveshare.com/jetracer-pro-ai-kit.htm.

[2] Rapid autonomous complex-environment competing ackermann-steering robot (race-car). URL https://mit-racecar.github.io/hardware/.

[3] AutoX. Ride with waymo one - our autonomous ride-hailing service.

[4] Donkey Car. An opensource diy self driving platform for small scale cars.

[5] Fei Chung. Getting started with jetson nano and donkey car aka autonomous car.

[6] Conrad. Voltcraft eco lipo 1000 modelbouwoplader 230 v 1 a li-poly, .

[7] Conrad. Conrad energy lipo accupack 7.4 v 1300 mah aantal cellen: 2 25 c, .

[8] Simon Du and Wei Hu. Linear convergence of the primal-dual gradient method for convex-concave saddle point problems without strong convexity. Feb 2018.

[9] Carlos Bordons Alba Eduardo F. Camacho. *Model Predictive control*. 2007. ISBN 978-0-85729-398-5. doi: 10.1007/978-0-85729-398-5.

[10] Elektorstore. Rplidar a1m8-r6 360-degree laser scanner kit (12 m), .

[11] Elektorstore. Ydlidar x4 lidar – 360-degree laser range scanner (10 m), .

[12] Brian Goldfain, Paul Drews, Changxi You, Matthew Barulic, Orlin Velev, Panagiotis Tsiotras, and James M. Rehg. Autorally an open platform for aggressive autonomous driving, 2018.

[13] George Gunter, Caroline Janssen, William Barbour, Raphael E. Stern, and Daniel B. Work. Model-based string stability of adaptive cruise control systems using field data. *IEEE Transactions on Intelligent Vehicles*, 5(1):90–99, March 2020. ISSN 2379-8904. doi: 10.1109/TIV.2019.2955368.

[14] He Hao and Prabir Barooah. Stability and robustness of large platoons of vehicle with double-integrator models and nearest neighbor interaction. *International Journal of Robust and Nonlinear Control*, 23, Dec 2013. doi: 10.1002/rnc.2872. URL https://doi.org/10.1002/rnc.2872.

[15] Nicholas Hyldmar, Yijun He, and Amanda Prorok. A fleet of miniature cars for experiments in cooperative driving, 2019.

[16] Intel. Intel realsense depth camera d455.

[17] SAE international. Sae standards news: J3016 automated-driving graphic update.

[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, Jul 2014. ISSN 0143-2087. doi: 10.1002/oca.2123. URL http://dx.doi.org/10.1002/oca.2123.

[19] Bingyi Liu, Dongyao(Tony) Jia, Kejie Lu, Dong Ngoduy, Jianping Wang, and Libing wu. A joint control-communication design for reliable vehicle platooning in hybrid traffic. Oct 2016.

[20] Lihua Luo, Hong Liu, Ping Li, and Hui Wang. Model predictive control for adaptive cruise control with multi-objectives: Comfort, fuel-economy, safety and car-following. *Journal of Zhejiang University SCIENCE A*, 11:191–201, 03 2010. doi: 10.1631/jzus.A0900374.

[21] Jan-Niklas Meier, Aravind Kailas, Oubada Abuchaar, Maher Abubakr, Rawa Adla, Mahdi Ali, George Bitar, Richard Deering, Umair Ibrahim, Paritosh Kelkar, Vivek Vijaya Kumar, Ehsan Moradi-Pari, Jay Parikh, Samer Rajab, Masafumi Sakakida, and Masashi Yamamoto. On augmenting adaptive cruise control systems with vehicular communication for smoother automated following. *Transportation Research Record: Journal of the Transportation Research Board*, 2672:036119811879637, 09 2018. doi: 10.1177/0361198118796375.

[22] Yuri Murakami. Berkeley autonomous race car. URL `https://closestnum20.com/barc_v4-0/`.

[23] Gerrit Naus, Rene Vugts, Jeroen Ploeg, M.J.G. Molengraft, and M. Steinbuch. String-stable cacc design and experimental validation: A frequency-domain approach. *Vehicular Technology, IEEE Transactions on*, 59:4268 – 4279, 12 2010. doi: 10.1109/TVT.2010.2076320.

[24] University of Michigan. Mcity test facility.

[25] Matthew O'Kelly, Varundev Sukhil, Houssam Abbas, Jack Harkins, Chris Kao, Yash Vardhan Pant, Rahul Mangharam, Dipshil Agarwal, Madhur Behl, Paolo Burgio, and Marko Bertogna. F1/10: An open-source autonomous cyber-physical platform, 2019.

[26] Clearpath Robotics. Jackal unmanned ground vehicle, . URL `https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/`.

[27] Clearpath Robotics. Turtlebot 2 mobile robot platform, . URL `https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/`.

[28] robotshop. Rplidar a1m8 - 360 graden laserscanner-ontwikkelingskit, .

[29] robotshop. Ydlidar x4 360 ° laserscanner, .

[30] robotshop. Messing male/female spacer set - m2.5 - 120 stuks, .

[31] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[32] Siddhartha S. Srinivasa, Patrick Lancaster, Johan Michalove, Matt Schmittle, Colin Summers, Matthew Rockett, Joshua R. Smith, Sanjiban Choudhury, Christoforos Mavrogiannis, and Fereshteh Sadeghi. Mushr: A low-cost, open-source robotic racecar for education and research, 2019.

[33] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. URL `https://www.ros.org`.

[34] Stereolabs. Zed 2 stereo camera.

[35] Mohsen Nader Tehrani, Murat Uysal, and Halim Yanikomeroglu. Device-to-device communication in 5g cellular networks: challenges, solutions, and future directions. *IEEE Communications Magazine*, 52 (5):86–92, May 2014. ISSN 1558-1896. doi: 10.1109/MCOM.2014.6815897.

[36] Tesla. Tesla autopilot.

[37] Simon van der Marel. Development of a platform for stereo visual odometry based platooning. 06 2021. URL `http://resolver.tudelft.nl/uuid:1b520436-feed-4a81-8684-ea3d5b1e55e0`.

[38] John Wang and Edwin Olson. Apriltag 2: Efficient and robust fiducial detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4193–4198, Oct 2016. doi: 10.1109/IROS.2016.7759617.

[39] Waveshare. Oak-d-lite hd camera development kit, .

[40] Waveshare. Imx219-83 stereo camera, 8mp binocular camera module, .

[41] Waymo. We're building the world's most experienced driver, .

[42] Waymo. Ride with waymo one - our autonomous ride-hailing service., .

[43] Sinan Öncü, Jeroen Ploeg, Nathan van de Wouw, and Henk Nijmeijer. Cooperative adaptive cruise control: Network-aware analysis of string stability. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1527–1537, Aug 2014. ISSN 1558-0016. doi: 10.1109/TITS.2014.2302816.

Nomenclature

ACC	- Adaptive Cruise Control
CACC	- Cooperative Adaptive Cruise Control
DMPC	- Distributed Model Predictive Controller
LiDAR	- Light Detection and Ranging
FOV	- Field Of View
ESC	- Electronic Speed Controller
PCB	- Printed Circuit Board
IMU	- Inertial Measurement Unit
PDMM	- Primal Dual Method of Multipliers
ROS	- Robotic Operating System