PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
ESCUELA DE POSGRADO

Design of a Mobile Robot's Control System for Obstacle Identification and Avoidance using Sensor Fusion and Model Predictive Control

TESIS PARA OPTAR EL GRADO ACADÉMICO DE

MAGISTER EN INGENIERÍA DE CONTROL Y AUTOMATIZACIÓN

AUTOR

JEAN PAUL BARRETO GUERRA

ASESORES

Dr.-Ing. ANTONIO MANUEL MORÁN CÁRDENAS

Dr.-Ing. SIEGBERT HOPFGARTEN

SETIEMBRE, 2017

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

TECHNISCHE UNIVERSITÄT ILMENAU

Technische Universität Ilmenau

Fakultät für Informatik und Automatisierung

Master Thesis

Design of a Mobile Robot's Control System for Obstacle Identification and Avoidance using Sensor Fusion and Model Predictive Control

To achieve the Degree of:

Master of Science (M.Sc.)

in Technische Kybernetik und Systemtheorie

Submitted by:	Jean Paul Barreto Guerra
Date and Place of Birth:	04.08.1990, Lima, Peru
Supervisor (TU Ilmenau):	Dr.-Ing. Siegbert Hopfgarten
Responsible Prof. (TU Ilmenau):	Prof. Dr.-Ing. habil. Pu Li
Supervisor (PUCP):	Dr. Antonio Morán Cárdenas
Date and Place:	18. 09. 2017, Ilmenau, Germany

Statement

I assure that I have done the work without the help of others and without using any sources other than those indicated, and that the work has not been submitted to the same other or similar form by any other examination body and has been accepted as part of an examination. All statements, which have been taken literally or meaningfully, are marked as such.

Jean Paul Barreto Guerra
Ilmenau, Germany, 18. 09. 2017

Acknowledgments

The autor of this thesis work, Jean Paul Barreto Guerra, thanks the FONDECYT-CONCYTEC grant through the agreement 2015-034 FONDECYT, in the framework of which the present thesis was developed "Design of a Mobile Robot's Control System for Obstacle Identification and Avoidance using Sensor Fusion and Model Predictive Control".

First of all, my special thanks to my advisor Dr. Siegbert Hopfgarten for his patience, dedication and valuable support, for helping to get the best of me during all this time and also for trusting in my skills. Another special thanks to my advisor Dr. Antonio Moran, whom I know years ago and has influenced in my professional development in the control and automation fields since then, all my gratitude for his advices and my admiration to him. Likewise, my deepest gratitude to Prof. Johann Reger and Dr. Javier Sotomayor that trusted in me and collaborated to fulfill my dream of being able to study in Germany. My gratitude also to Prof. Pu Li for giving me the opportunity to work in the field of autonomous vehicles.

My greatest thanks to my fiancée Patty for her patience and support in this adventure, for being with me when I needed the most, for understand how important was for me to grow professionally and tell me that I am always the best. My greatest thanks also to my parents Virginia and Orlando for always wishing the best for me, even if it means being away from them, for their advices, for having raised me as I am and for having accompanied me in every triumph achieved in my life.

Thanks to my close friends here from Ilmenau, Coco, Eduardo and Jorge, that helped me in every possible way since my arrival and for their true friendship. My sincere thanks to my family, because they always believed in me and also to my friends that worried about me and wished the best.

Finally special thanks to Dr. Oscar Penny, who taught me how to be a good engineer, to help me create dreams and make them come true, I know that from the sky he watches me.

Abstract

The aim of this master thesis is to design a control system based on model predictive control (MPC) with sensor data fusion for obstacle avoidance. Since the amount of obtained data is larger due to multiple sensors, the required sampling time has to be larger enough in comparison with the calculation time of the optimal problem. Then it is proposed a simplification of the mobile robot model in order to reduce this optimization time.

The sensor data fusion technique uses the range information of a laser scanner and the data of a mono-camera acquired from image processing techniques. In image processing different detection algorithms are proposed such as shape and color detection. Therefore an estimation of the obstacles dimension and distance is explained obtaining accurate results.

Finally a data fusion for obstacle determination is developed in order to use this information in the optimization control problem as a path constraint. The obtained results show the mobile robot behavior in trajectories tracking and obstacle avoidance problems by comparing two different sampling times. It is concluded that the mobile robot reaches the final desired position while avoiding the detected obstacles along the trajectory.

Kurzfassung

Ziel dieser Masterarbeit ist, einen Steuerungsentwurf auf Basis der modellprädiktiven Regelung (MPC) mit Sensordatenfusion und zur Hindernisvermeidung. Da die Menge der erhaltenen Daten aufgrund mehrerer Sensoren größer ist, muss die erforderliche Abtastzeit im Vergleich zur Rechenzeit des optimalen Problems größer sein. In der Arbeit wird eine Vereinfachung des mobilen Robotermodells vorgeschlagen, um diese Optimierungszeit zu reduzieren.

Die Sensordaten-Fusionstechnik verwendet die Bereichsinformation eines Laserscanners und die Daten einer Monokamera, die durch Bildverarbeitungstechniken gewonnen werden. Bei der Bildverarbeitung werden verschiedene Erfassungsalgorithmen vorgeschlagen, wie z. B. Muster- und Farbdetektion. Eine Schätzung der Hindernisdimension und -distanz wird erklärt, um genaue Ergebnisse zu erzielen.

Schließlich wird eine Datenfusion zur Hindernisbestimmung entwickelt, um diese Information im Optimalsteuerungsproblem als Pfadbeschränkung zu nutzen. Die erzielten Ergebnisse zeigen das Verhalten des mobilen Roboters bei Trajektorienverfolgungs- und Hindernisvermeidungsproblemen, indem zwei verschiedene Abtastzeiten verglichen werden. Es wird gefolgert, dass der mobile Roboter die endgültige gewünschte Position erreicht, während die erkannten Hindernisse entlang der Trajektorie vermieden werden.

List of Abbreviations

ACC	Adaptive Cruise Control
CHT	Circular Hough Transform
EKF	Extended Kalman Filter
GPS	Global Positioning System
HSV	Hue-Saturation-Value
HT	Hough Transform
IMU	Inertial Measuring Unit
IPM	Inverse Perspective Mapping
MIMO	Multiple Inputs, Multiple Outputs
MLP	Multi-Layer Perceptron
MPC	Model Predictive Control
MSE	Mean Square Error
NLP	Nonlinear optimization Problem
PTZ	Pan-Tilt-Zoom
RGB	Red-Green-Blue
RMSE	Root Mean Square Error

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

In today's world, it is seen that the technology has evolved in an exponential manner over the years. That is to say, the case that human intervention performs various activities is declining, either to reduce the execution time of a job or the difficulty of controlling a system. Because of this phenomenon, corresponding technology has been introduced as well to industries as to the daily life of people.

In the field of engineering, these technological changes can be seen. Since the beginning of the last century systems have been implemented which exclusively relied on the management of human beings, but in the last decades, ways were sought that these systems are intelligent, i.e., they can function on their own without or very little human intervention, and this is how the concept of artificial intelligence was created.

If one talks about artificial intelligence, there is by definition a system that has to make a decision to reach some goals under consideration of its situation and environment. That system can tend to create an autonomous system. Autonomous systems have been used in various fields of application with regard to civil, rural, military areas, etc. Thus, unmanned vehicles whether on land, sea and air begin to play an important role in the execution of various tasks and they are developed further. Specifically, if there is an unknown area, these unmanned vehicles must behave autonomously, i.e., that the control system must capable of performing the sensing, processing and decision-making tasks in real-time considering the characteristics of the system itself, the existing constraints and physical resources.

Talking about exploration, the environment in which they move greatly influences the performance of the task. Ground vehicles have problems regarding the geography of the terrain and obstacles along the way. Tracking trajectories and obstacle avoidance are the most common tasks of these ground vehicles, i.e. the vehicle goes from an ini-

tial position to a final position avoiding all the stationary and moving obstacles that are in the unknown area.

Many control strategies and techniques have been implemented for solving these problems like Model Predictive Control (MPC) that is a robust and reliable advanced control strategy that can handle dynamic Multiple Inputs Multiple Outputs (MIMO) systems. It is shown to be more effective than classical control methods because the main feature of MPC is that the desired behavior as well as the constraints on the system can be directly specified in the problem formulation.

In trajectories tracking the external constraints are defined as the stationary or moving unknown obstacles, then it is important and necessary to capture all the information coming from these unknown obstacles in order to avoid them and achieve the goal position. By adding devices and fusing the sensors data one can improve the obstacle identification.

Sensor fusion refers to the synergistic use of information from different sensors so that the system can achieve a required task. Fusion of data is especially important in any application where a large amount of data must be combined, merged and grouped to obtain the appropriate quality and integrity of the decisions to be made. The advantages of multi-sensor data fusion are redundancy of the information can reduce uncertainty, increase of the accuracy and increase of the reliability in case of sensor failure. Thus the data fusion process allows to combine the measurements and information in order to deliver the sufficient knowledge to achieve the required task.

For these reasons, the central goal of this thesis is the implementation of a sensor fusion algorithm for obstacle identification and application of MPC for trajectories tracking and obstacle avoidance with the mobile robot SUMMIT. Since sensor fusion sometimes requires a large computation time due to the data processing of each sensor, then the problem to compensate this computation time arises. Because of this, a mathematical 3-state model is proposed to simplify the system behavior.

1.2 Previous works and initial situation

As mentioned before, recently the number of applications of autonomous mobile robots has been increasing noticeably in scientific research. Because of this the necessity of sensors arises, such as laser scanners and cameras. The possibility of extracting basic information from the robot environment by means of small and low-power integrated triangulation laser scanner was demonstrated by (Fu, Corradi, Menciassi, & Dario, 2011). Also a method which uses an angle histogram to create a path to improve the accuracy of navigation in a corridor environment by using a laser scanner was proposed by (J. K. Park, Kim, & Park, 2015). On the other hand, talking about obstacle detection, (Hussin, Juhari, Kang, R.C.Ismail, & Kamarudin, 2012) presented color processing techniques as primary filtering and Circular Hough Transform (CHT) as a shape detection technique, (Lee, Yi, & Cho, 2016) used the inverse perspective mapping (IPM) method and (T.N.R.Kumar, 2015) proposed a real time approach for road boundary and lane detection, and also pothole and object detection in a road by using Hough transform (HT). Often when using image processing techniques some parameters of the camera must be determined, i.e. the case in (Zhang, 2000) and (Heikkilä & Silvén, 1997) where different calibration techniques and procedures are proposed.

Mobile robots have not only to execute predefined programmed tasks, but also they must explore the unknown working environment. A new model for mobile robot 3D environment using a mono-vision system was proposed, implemented and tested by (Al-Jarrah, 2016). Some conditions for modeling the environment are the obstacles dimensions and the distance estimation between these obstacles and the mobile robot. A measuring system based on a single non-metric CCD camera and a laser projector was implemented by (Wang, Lu, Wang, & Tsai, 2007). A measurement estimation algorithm for two cases of obstacles width by using the detected lane information and a pinhole camera model was applied by (Han, Heo, Park, Kee, & Sunwoo, 2016). A vision-based Adaptive Cruise Control (ACC) system for computing range and range-rate from a single camera was described by (Stein, Mano, & Shashua, 2003). Also (Mrovlje & Vrancic, 2008) used a stereoscopy technique in order to create an illusion of the depth using two pictures taken at slightly different positions. A Markov random field-based obstacle segmentation using IPM results to estimate distances was performed by (Lee et al., 2016), while a range estimation method with a monocular camera considering the pitch angle due to vehicle motion and road inclination was

proposed by (K.-Y. Park & Hwang, 2014).

In general, since the combination of data from both sensors is required, there has been research also in this area. An approach to fuse 3D and 2D information in a driver assistance setup to perform obstacle detection and tracking was presented by (Gruyer, Cord, & Belaroussi, 2013), while (Mahajan, Bhosale, & Kulkarni, 2013) proposed a cooperative system based on a mono-vision camera and laser scanner in order to detect the obstacles more precise and accurate.

Finally in previous works with the mobile robot SUMMIT there have been important contributions in contemplation of making the robot autonomous. A non-linear single track model of the mobile robot was designed, parametrized and validated, and an optimization algorithm based on collocation method and multiple shooting for time and energy optimal control problems was applied by (Müller, 2013). The efficiency of MPC for tracking trajectory and obstacle avoidance problems were improved by (Thieme, 2014). The obstacles (detected by a laser scanner and described by ellipses) as constraints was introduced, and the optimization problem using the optimizer tool Ipopt was solved by (Drozdova, 2015). The position determination of the mobile robot by using Kalman filter and a neural network was improved by (Belgradskaia, 2016).

1.3 Objective of the work

As has been mentioned, several previous works dealt with the modeling, parameter estimation, obstacle detection using laser scanner data, constraint formulation of obstacles, and application of MPC avoiding obstacles. The main focus in the proposed master thesis lies on the involvement of camera data for obstacle detection and description as well as the fusion with the laser scanner data to aim at a better and more reliable description of the obstacles to be avoided within an autonomous driving using MPC. Different driving scenarios will be investigated and practically tested at the mobile robot SUMMIT. To accomplish this objective the following topics will be thoroughly examined in the work presented:

- Improve the mobile robot SUMMIT mathematical model based on a counter-steering driving model, correct the control variables and estimate the state variables due to this modeling.
- Acquire, analyze and process the data obtained from the laser scanner and also

from the video camera in order to use sensor fusion technique and get a better information about the position and dimension of the obstacles in the unknown environment.

- Propose the new optimization problem to be solved based on the previously processed data of the environment and use MPC in order to accomplish the avoidance of all obstacles along the way.
- Embed the extensions (sensor data fusion) and improvements in the control system of the mobile robot SUMMIT and analyze its behavior by testing in different situations.
- Make conclusions about the upgrade of the system and propose future works to look at.

1.4 Outline of the chapters

The content presented in this thesis is organized as follows:

- *Chapter 2* describes the mobile robot SUMMIT technical specifications, and the sensors characteristics and functioning. Likewise, the development of 3-state mathematical model is explained as well and its validation by comparing it with the previously used mathematical state model.
- *Chapter 3* presents the correction of the control variables velocity and steering angle by using a linear approximation and a neural network, respectively. Moreover the estimation of the state variables is proposed by the approach of the Extended Kalman Filter (EKF). Simulation and experimental tests are performed in order to test the accuracy of the proposed techniques.
- *Chapter 4* introduces the concepts of computer vision, and presents the detection and cluster classification processes. In the first process, the shape and color detection algorithms are explained. Then in the second process the cluster classification techniques are considered for obstacle detection. Experimental tests are performed in order to validate the image processing algorithms.
- *Chapter 5* presents the data acquisition of the laser scanner and the 2D camera. Also, the concept of the pinhole camera model is introduced. Then the estimation of the obstacle width and height are explained and developed. For obstacle

avoidance an ellipse form is proposed and its parameters are described based on data fusion of both sensors. Experimental tests are performed in order to test the precision of the obstacle determination.

- *Chapter 6* presents the application of MPC considering the trajectory tracking and obstacle avoidance problems. The chapter starts with a the definition of the continuous nonlinear optimal control problem. Then a solution using multiple-shooting and collocation methods with MPC is proposed. Experimental tests are performed in order to test the accuracy and the efficiency of the optimization control problem.
- *Chapter 7* concludes the thesis with a summary of the main results and discusses the outlook for possible future topics of research.

2 Description of the Mobile Robot SUMMIT

The SUMMIT is a medium-sized autonomous robot with high mobility and it is ideal for both indoor and outdoor applications. It has Ackerman (car-like) kinematics and a crawler chassis design specially suited for rough outdoor terrain. Also it has a mechanical system that is equivalent to a 4x4 remote control car and uses a high quality aluminum chassis. Both front and rear axles are has a counter-steerable behavior in order to have a greater rotation angle. To further improve it's stability, each wheel has a drive motor mounted on each axis with an independent damping system and pendulum counterweight. Fig. 2.1 and 2.2 (obtained from (Robotnik Automation, S.L.L, 2012b)) show the SUMMIT robot and its most important components.

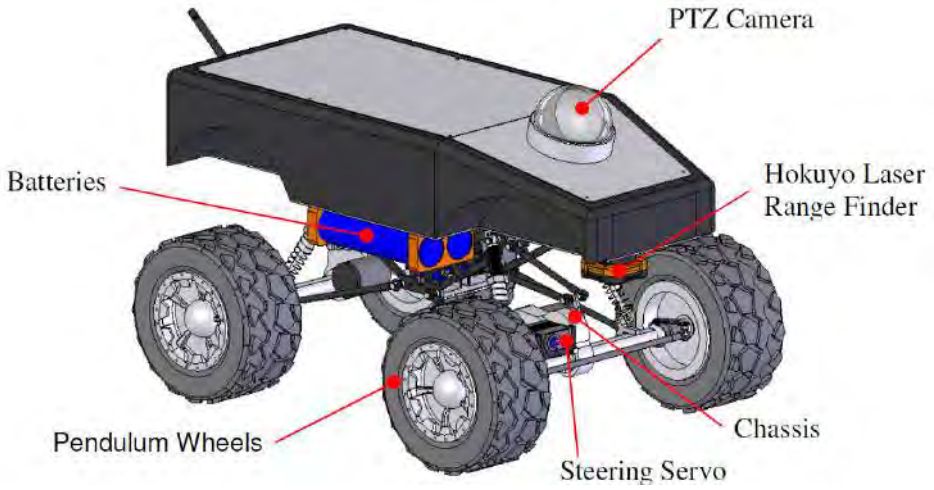


Figure 2.1: Main parts of the SUMMIT robot

The main parts of the robot are:

- *Housing:* Holds the upper and rear covers. The electrical components are placed inside, while the motor drivers are the only components that are outside.

- *Upper Cover:* Allows the access to the interior of the robot where some of the control components are placed.
- *Rear Cover:* Holds the control panel, buttons and the WiFi antenna.
- *Pendulum Wheels:* Aluminum wheels with removable pendulum weights.
- *Batteries:* 4 LiFePO4 3.2 V, 16 Ah cells.
- *Chassis:* Aluminum chassis with strong shock absorbers.
- *Motors:* Two brushless motors, one in the front axle and the other in the rear axle.
- *Steering Servo:* HITEC HS-7980TH servo with 38 Kg/cm torque at 6.0 V in each axle.
- *PTZ Camera:* Logitech Sphere AF camera with microphone and protected by a replaceable dome.
- *Hoyuko Laser Range Finder:* Laser range scanner with the specific adaptor.
- *Encoder:* High-speed rotary magnetic encoder designed for use in harsh environments.

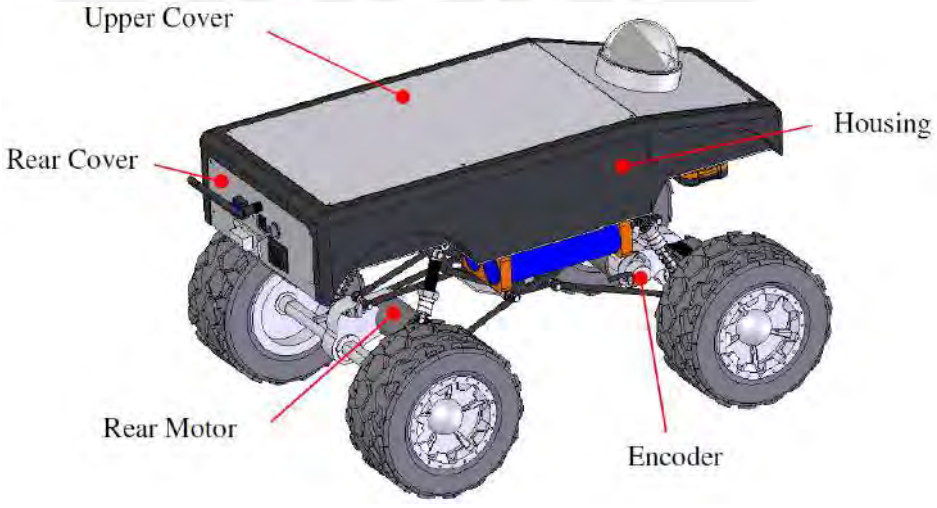


Figure 2.2: Main parts of the SUMMIT robot (rear view)

Every main piece of the robot and more detailed description of it can be found in (Robotnik Automation, S.L.L, 2012b).

2.1 Technical specifications

The robot comes with operating system (Linux Real Time) and complete software architecture and tools, fully installed and working. Complete installation of a new system starting from scratch can be found on (Robotnik Automation, S.L.L, 2012c). The robot has its own WiFi network, this allows to communicate with the PC via TCP/IP protocol. In order to control the robot the Player/Stage software is used. It allows to send and receive control and sensor data signals using C++ language. In general, Player provides a network interface to a variety of robot and sensor hardware. The Player/Stage architecture and information can be found in (Robotnik Automation, S.L.L, 2012a) and (Hedges et al., 2006), respectively. All the programming was made in Ubuntu version 1.0, QtCreator 1.3, Player version 2.0 and Stage version 3.2.1. Then the main specifications can be seen in Tab. 2.1, while in Fig. 2.3 (obtained from (Robotnik Automation, S.L.L, 2012b)) it is shown the dimensions in frontal, lateral and top views.

	Mechanical
External dimensions	570 x 345 x 320 mm
Weight	12.9 Kg
Speed	3 m/s
Traction system	4 wheels
Batteries	4 x 3.3V LiFePO4
Traction motors	2 brushless motors
Kinematics	Dual or single axis Ackerman steering
	Control
Controller	Player/Stage Embedded PC with Linux Real Time
Communication	WiFi 802.11n

Table 2.1: Technical specifications of the SUMMIT robot

Since the main objective of the work is sensor fusion for obstacle identification and avoidance, it is necessary to detail the function of the elements involved in the fusion algorithm. So in this work the laser scanner and the PTZ camera (Pan-Tilt-Zoom camera) are used for this task. In the next subsections are detailed the characteristics and functioning of them, respectively.

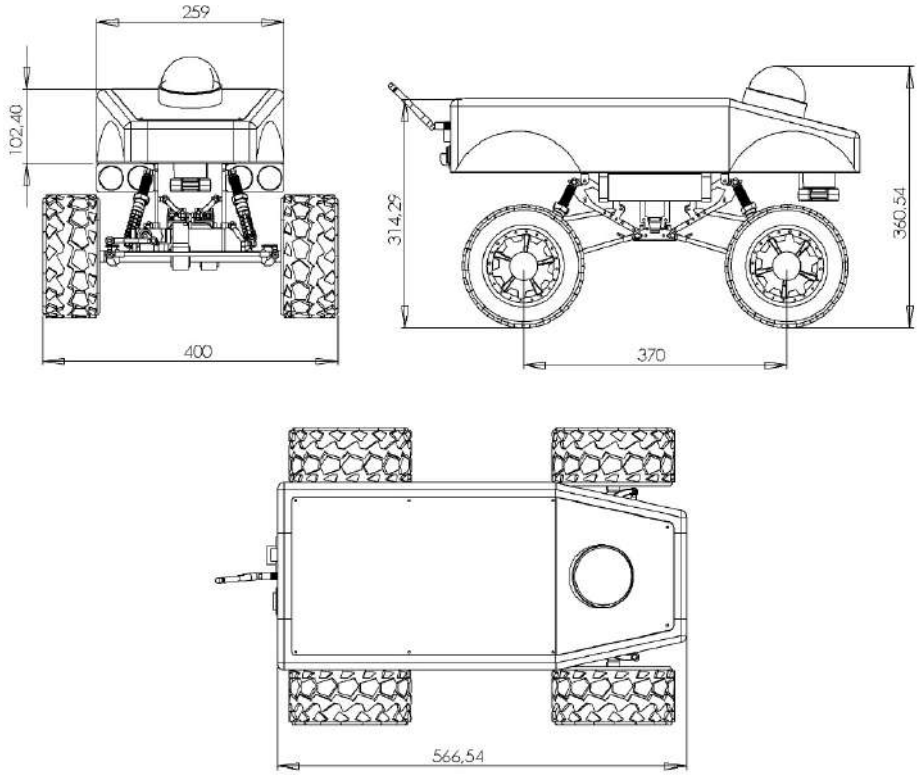


Figure 2.3: External SUMMIT robot drawings

2.1.1 Laser Scanner

One of the main elements or devices that the robot has is the laser range finder or usually called laser scanner. The Hokuyo URG-04LX-UG01 scanning laser range-finder is a small, affordable and accurate laser scanner that is perfect for robotic applications.

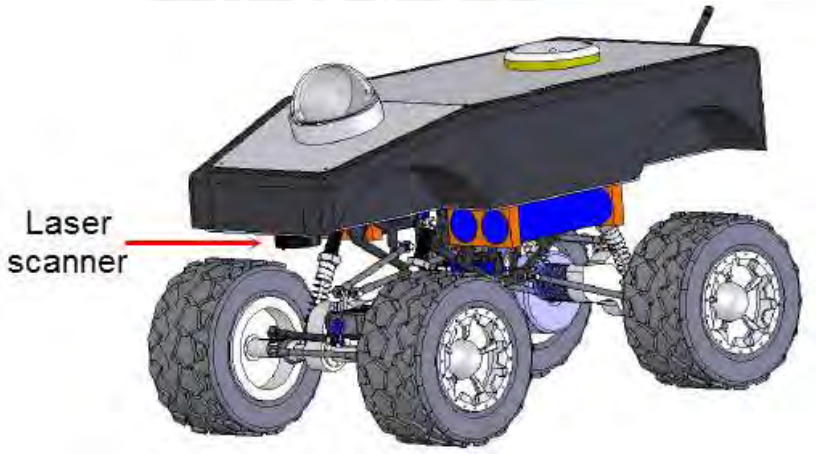


Figure 2.4: Laser scanner and PTZ camera mounted on the SUMMIT robot

As can be seen from Tab. 2.2, the total scan angle is 240°. So for this scan angle, the scan point goes from 1 to 682. Since the goal of the robot is to move forward detecting and avoiding obstacles ahead, the whole scanning range is not necessary, for this reason in this work 120°, is used as the scan angle (scan point from 170 to 511). The default detection range of the laser scanner is 5.6 m, but for practical reasons 3.3 m is used.

	Laser Scanner
Model No.	URG-04LX-UG01
Power Source	USB Bus power
Light Source	Semiconductor laser diode λ=785nm;Laser safety class 1
Measuring area	20 to 5600mm, 240°
Angular resolution	Step angle: approx. 0.352°(360°/1024 steps)
Scanning time	100 ms/scan
Weight	Approx. 160 g

Table 2.2: Laser scanner specifications

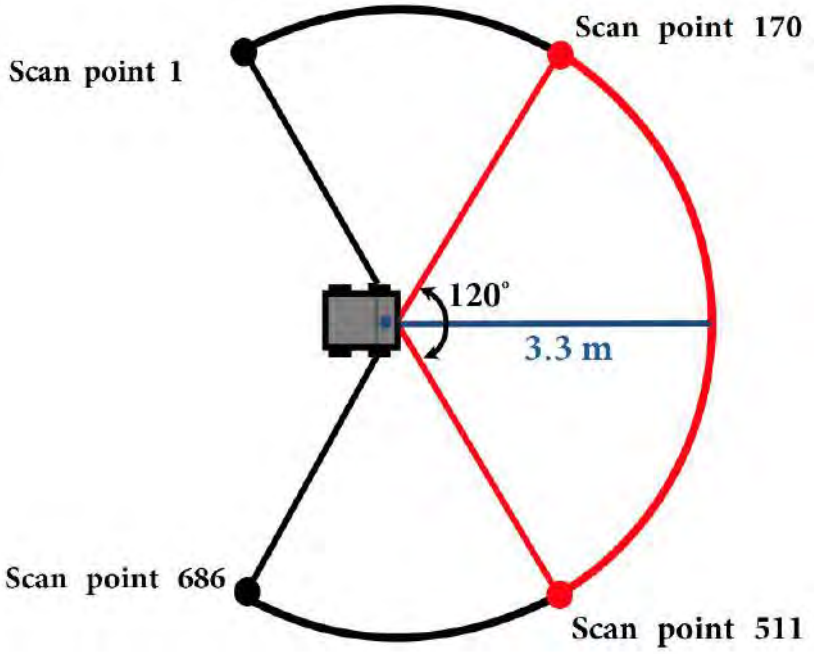


Figure 2.5: Laser scanner selected area (angle and range)

2.1.2 PTZ Camera

The other main element that the robot has is the PTZ camera. The Logitech Sphere AF is a 2D mono-camera with 2 Megapixels of maximum resolution, it has up to 30fps and RightLight 2 Teechnology which adjusts intelligently to produce true-to-life clear images in dim or poor backlight settings.

In Tab. 2.3 most of the camera specifications can be seen. In this work, the 2D mono-camera is used for taking pictures while the robot is moving (working together with the laser scanner). The resolution selected for every picture is 320 x 240 pixels in order to have less computation time for the image processing algorithm that will be seen in chapter 4.

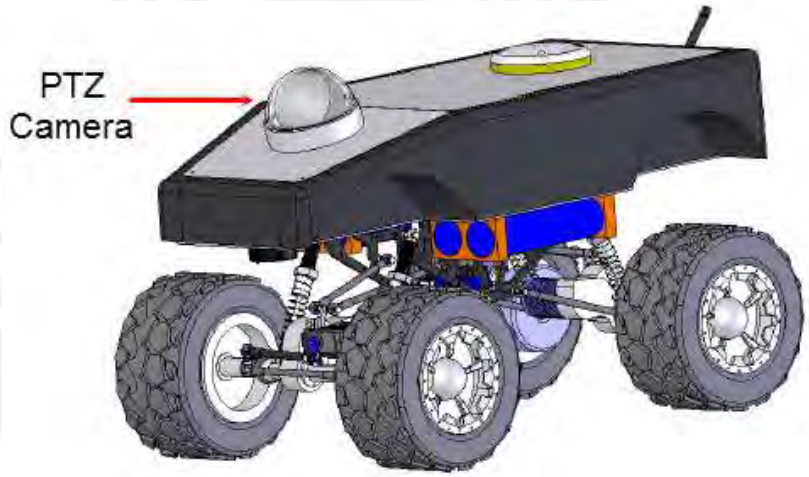


Figure 2.6: PTZ camera mounted on the SUMMIT robot

	PTZ Camera
Motorized tracking	189°horizontal and 102°vertical
Focus system	Autofocus lens
Resolution	2-megapixel sensor
Color depth	24-bit true color
Video capture	Up to 1600 by 1200 pixels (HD quality)
Frame rate	Up to 30 frames per second
Focal length	3.7 mm

Table 2.3: PTZ Camera specifications

2.2 Mathematical model

2.2.1 Establishment of the mathematical model

A mathematical model based on a single-track model was developed and validated in (Thieme, 2014) and (Müller, 2013). So the state and control variables are defined as,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} \beta \\ \\ \dot{\psi} \\ x_{pos} \\ y_{pos} \end{bmatrix} \quad , \quad u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} v \\ \delta \end{bmatrix} \tag{2.1}$$

where β is the side slip angle, the yaw angle, $\dot{\psi}$ the yaw angular velocity, x_{pos} and y_{pos} are the position coordinates in the XY-plane. v is the velocity of the robot and δ is the steering angle. Then the dynamics of the SUMMIT robot are governed by the following equations,

$$\begin{aligned} \dot{x}_1 &= \frac{c_\alpha \cos(\delta)\left(-2\beta + \frac{\dot{\psi}}{v}(l_r - l_f)\right)}{m\, v \cos(\beta)} - \dot{\psi} \\ \dot{x}_2 &= \dot{\psi} \\ \dot{x}_3 &= \frac{c_\alpha \cos(\delta)\left(\delta(l_r + l_f) + \beta(l_r - l_f) - \frac{\dot{\psi}}{v}(l_r^2 + l_f^2)\right)}{J} \\ \dot{x}_4 &= v \cos(\beta + \psi) \\ \dot{x}_5 &= v \sin(\beta + \psi) \end{aligned} \tag{2.2}$$

where c_α is the cornering stiffness, m is the mass of the vehicle, J is the inertia moment, l_f is the distance from the COG to the rear axle and l_r is the distance from the COG to the front axle. All these parameters are numerical values and are calculated in (Thieme, 2014). This mathematical model can be used, but since the robot's velocity is relatively low, the side slip angle β is considered to be near 0, i.e. that a new mathematical model based on its dynamics must be proposed.

First of all, the SUMMIT robot moves in XY-plane, and it is important to consider that it has a counter-steering wheel's system. So the mobile robot moves with a velocity and a steering angle that affects both front and rear wheels. In Fig. 2.7 the first sketch of the mobile robot and the system variables that influence the robot's movement can be observed and are considered in the new model.

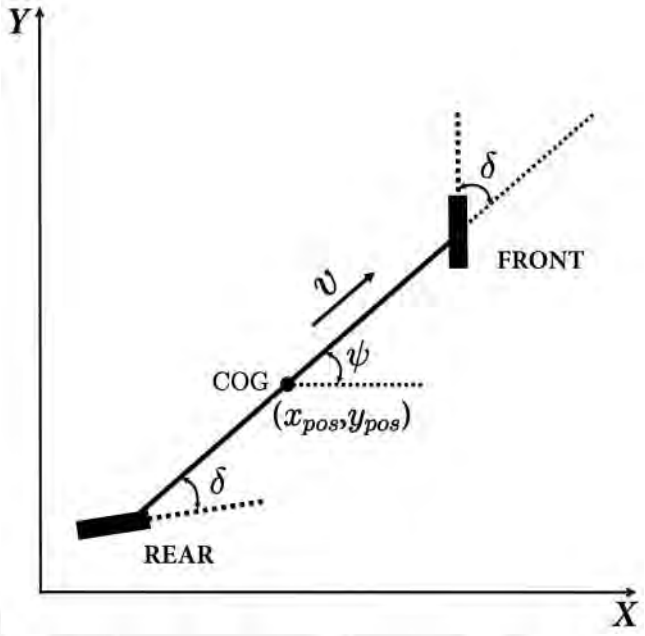


Figure 2.7: Counter-steering wheel's system behavior in the XY-plane

Fig. 2.8 is more detailed and in it can be observed that the system will be analyzed in the discrete time interval $[k, k+1]$. Later it will be seen that analyzing the robot movement from one position to another will help to get all the system equations.

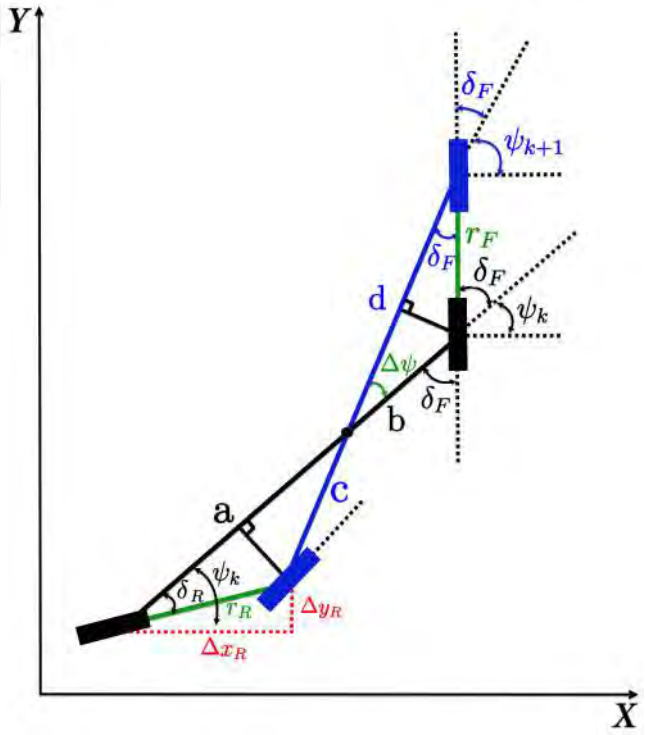
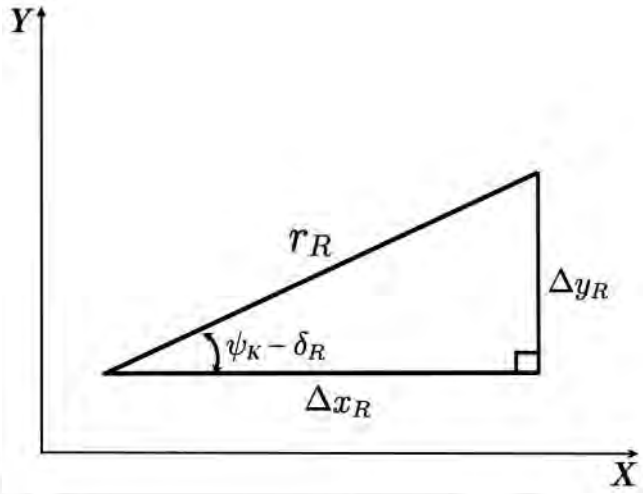
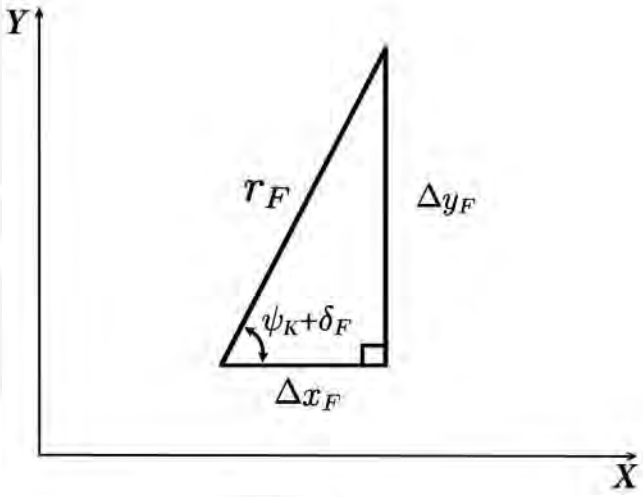


Figure 2.8: Mobile robot movement in discrete time interval $[k, k+1]$

From Fig. 2.8a and 2.8b it can be observed that the rear wheel moves a distance r_R and the front wheel moves r_F.



(a) Rear wheel



(b) Front wheel

Figure 2.8: Wheels position due to the movement

Thus the traveled distance in the XY-plane in the time interval $[k, k+1]$ can be determined as follows,

$$\begin{aligned} \Delta x_F &= r_F \cos(\psi_k + \delta_F) \\ \Delta y_F &= r_F \sin(\psi_k + \delta_F) \\ \Delta x_R &= r_R \cos(\psi_k - \delta_R) \\ \Delta y_R &= r_R \sin(\psi_k - \delta_R) \end{aligned} \tag{2.3}$$

It can be observed that the traveled distances depend on the actual value of ψ. Also, since the steering angle in both axles are equal, then the traveled distances by the front and rear wheels are the same.

$$\psi = \psi_k \tag{2.4}$$
$$\delta = \delta_F = \delta_R \tag{2.5}$$
$$r = r_F = r_R \tag{2.6}$$

i.e. that the robot wheels move Δx and Δy in the X and Y-direction, respectively, in the discrete time interval $[k, k+1]$. Thus from (2.3),

$$\begin{aligned} \Delta x &= r\left(\frac{\cos(\psi+\delta)+\cos(\psi-\delta)}{2}\right) \\ \Delta y &= r\left(\frac{\sin(\psi+\delta)+\sin(\psi-\delta)}{2}\right) \end{aligned} \tag{2.7}$$

Then by trigonometric identity,

$$\begin{aligned} \Delta x &= r\ \cos(\psi)\ \cos(\delta) \\ \Delta y &= r\ \sin(\psi)\ \cos(\delta) \end{aligned} \tag{2.8}$$

Now transforming from discrete time to continuous time (dividing both sides by Δt), the first 2 state equations are found,

$$\dot{x} = v\ \cos(\psi)\ \cos(\delta) \tag{2.9}$$
$$\dot{y} = v\ \sin(\psi)\ \cos(\delta) \tag{2.10}$$

where the position coordinates can be written as $\dot{x} = \dot{x}_{pos}$ and $\dot{y} = \dot{y}_{pos}$.

Now, it is necessary to find the third state equation. So from Fig. 2.8 it can be seen that a and b determine the length between both axles in the time instant k, while c and d determine the same length but in the time instant $k+1$.

$$L = a + b = c + d \tag{2.11}$$
$$2\,L = (a+d) + (b+c) \tag{2.12}$$

Since the robot is moving forward, the yaw angle changes.

$$\Delta\psi = \quad \psi_{k+1} - \psi_k \tag{2.13}$$

where $\Delta\psi$ is the difference between the yaw angle in the time interval $[k, k+1]$.

The relationship between the length of the mobile robot and $\Delta\psi$ can be obtained from Fig. 2.8 by looking at the movement of the front and rear wheels in the time interval $[k, k+1]$ respect to the body of the mobile robot. Then for the front wheels movement one has the following relationship,

$$2b\,\cos(\Delta\psi) + r_F\,\cos(\delta_F) = d \tag{2.14a}$$

$$b\,\sin(\Delta\psi) = r_F\,\sin(\delta_F) \tag{2.14b}$$

as well for the rear wheels movement one has the following

$$2c\,\cos(\Delta\psi) + r_R\,\cos(\delta_R) = a \tag{2.15a}$$

$$c\,\sin(\Delta\psi) = r_R\,\sin(\delta_R) \tag{2.15b}$$

Taking (2.14a) and (2.15a), and considering that $\Delta\psi$ is relatively small, one gets the following equation,

$$(b+c) + r_F\,\cos(\delta_F) + r_R\,\cos(\delta_R) = (a+d) \tag{2.16}$$

Then replacing (2.5), (2.6) and (2.12) in (2.16),

$$(b+c) = L - r\,\cos(\delta) \tag{2.17}$$

Now by taking (2.14b) and (2.15b),

$$(b+c)\,\sin(\Delta\psi) = r_F\,\sin(\delta_F) + r_R\,\sin(\delta_R) \tag{2.18}$$

And replacing (2.5), (2.6), (2.16) in (2.18),

$$(L - r\,\cos(\delta))\,\sin(\Delta\psi) = 2\,r\,\sin(\delta) \tag{2.19}$$

Since the mobile robot is moving from one position to another (distance r) in a very

short time interval and also considering that $\Delta\psi$ is nearly 0, then

$$L\,\Delta\psi = 2\,r\,\sin(\delta) \tag{2.20}$$

And the final state equation is,

$$\dot{\psi} = \frac{2\,v}{L}\,\sin(\delta) \tag{2.21}$$

Finally the state variables and control variables are shown in (2.22)

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_{pos} \\ y_{pos} \\ \psi \end{bmatrix} \quad , \quad u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} v \\ \delta \end{bmatrix} \tag{2.22}$$

where x_{pos} and y_{pos} are the position coordinates in the XY-plane, and ψ the yaw angle. v is the velocity of the robot, δ is the steering angle. Then the dynamics of the SUMMIT robot are governed by,

$$\begin{aligned} \dot{x}_1 &= v\,\cos(\psi)\,\cos(\delta) \\ \dot{x}_2 &= v\,\sin(\psi)\,\cos(\delta) \\ \dot{x}_3 &= \frac{2\,v}{L}\,\sin(\delta) \end{aligned} \tag{2.23}$$

where L is the length of the mobile robot and is defined as follows,

$$\begin{aligned} L &= l_F + l_R \\ &= 0.1776\,m + 0.1924\,m \\ &= 0.37\,m \end{aligned} \tag{2.24}$$

As it can be seen, (2.2) refers to the previous mobile robot's model (the system has 5 state variables), while (2.23) refers to the new mobile robot's model (the system has 3 state variables). This reduction of the states entails less computational time and effort.

2.2.2 Validation of the model

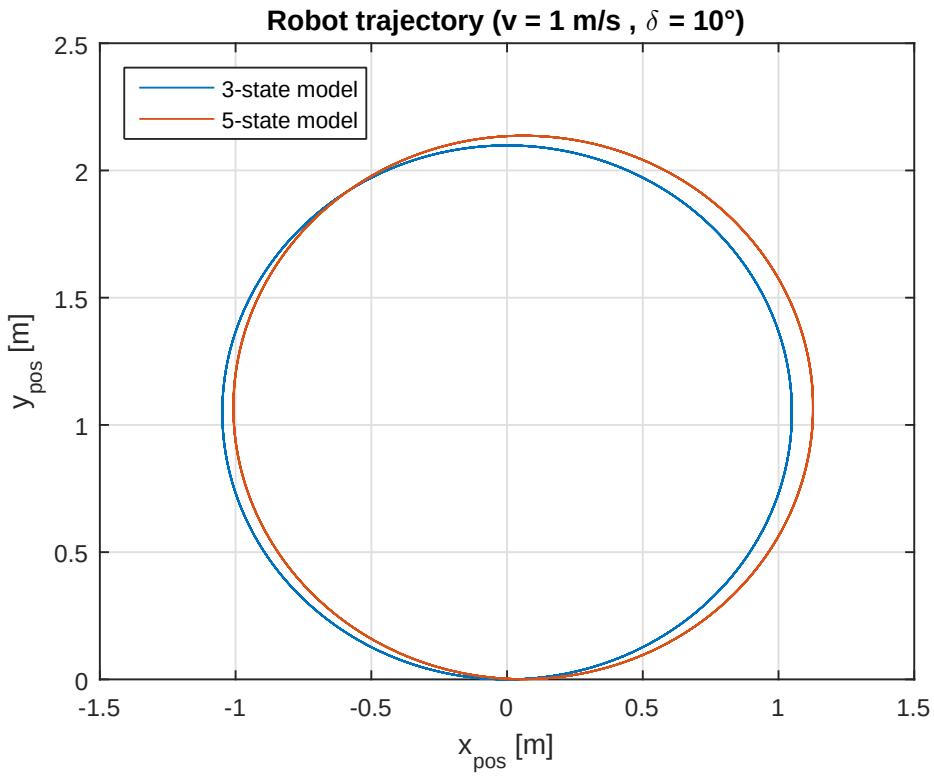
In order to validate the new mobile robot's model, a comparison between the 3-state and 5-state models was made using MATLAB®[1]. In case 1 ($v = 1\,m/s$ and $\delta = 10°$),

[1]MATLAB®is a registered trademark of The Mathworks Inc.

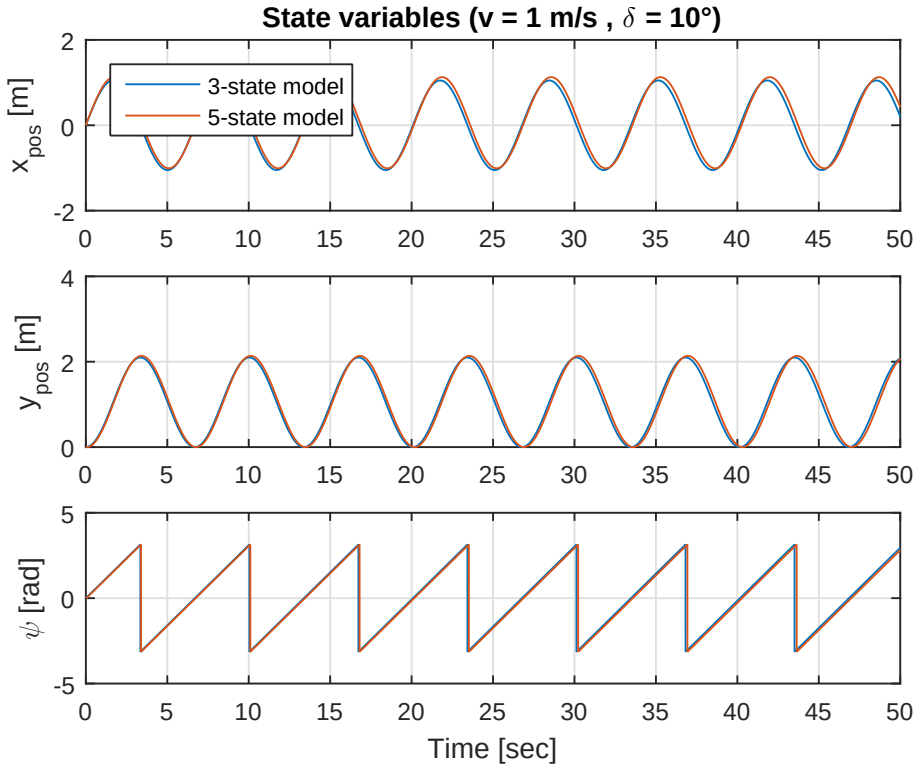
both the velocity and steering angle are constant. In Fig. 2.9a it is observed that the position in the XY-plane for both models are very similar but with a little deviation. Also in Fig. 2.9b it can be seen that the state variables are very similar. Finally in Fig. 2.9c the error of every state variable is shown, where the resulting error values are increasing due to the dynamics of the mobile robot. Nevertheless the mean-square errors (MSE) and the root-mean-square errors (RMSE) are relatively low for a long time horizon.

Case 1		
v	1 m/s	(constant)
δ	10°	(constant)
	MSE	RMSE
x_{pos}	175.7459	13.2569
y_{pos}	10.7929	3.2853
$_{pos}$	3.3390	1.8273

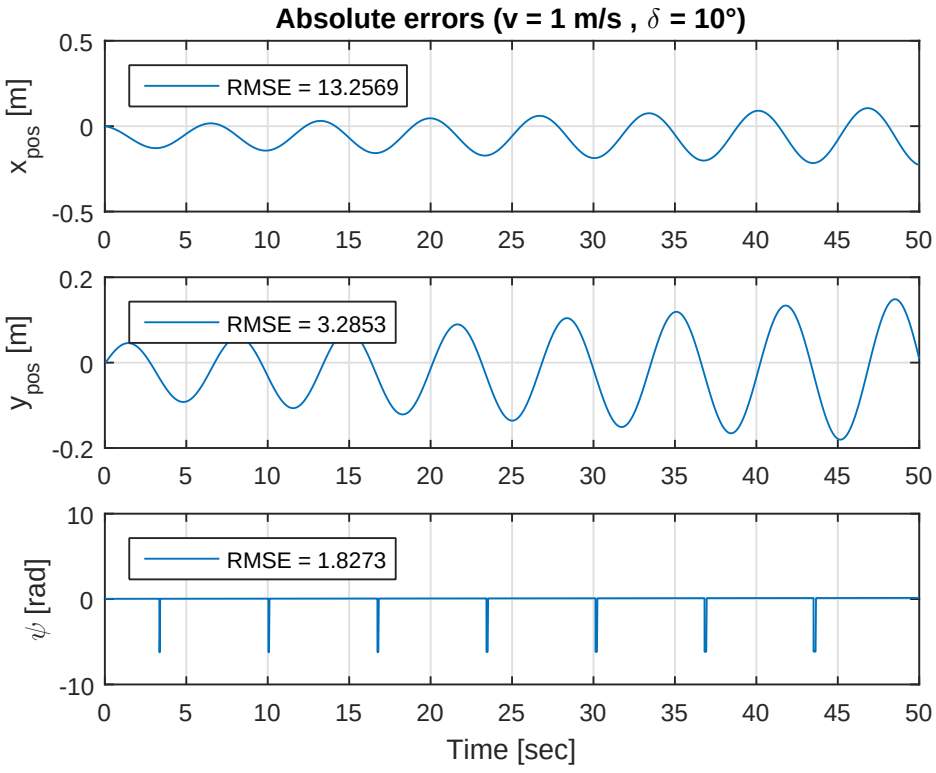
Table 2.4: MSE and RMSE for case 1



(a) Robot trajectory in XY-plane



(b) State variables



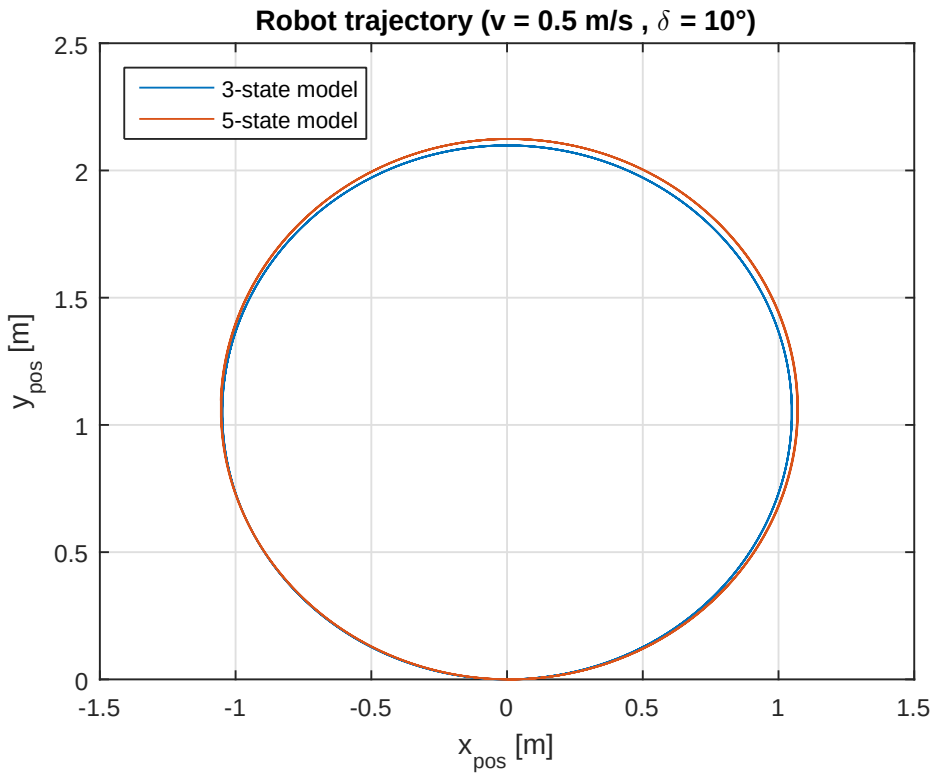
(c) Absolute and RMS errors

Figure 2.9: Results for case 1 ($v = 1\ m/s$ and $\delta = 10°$)

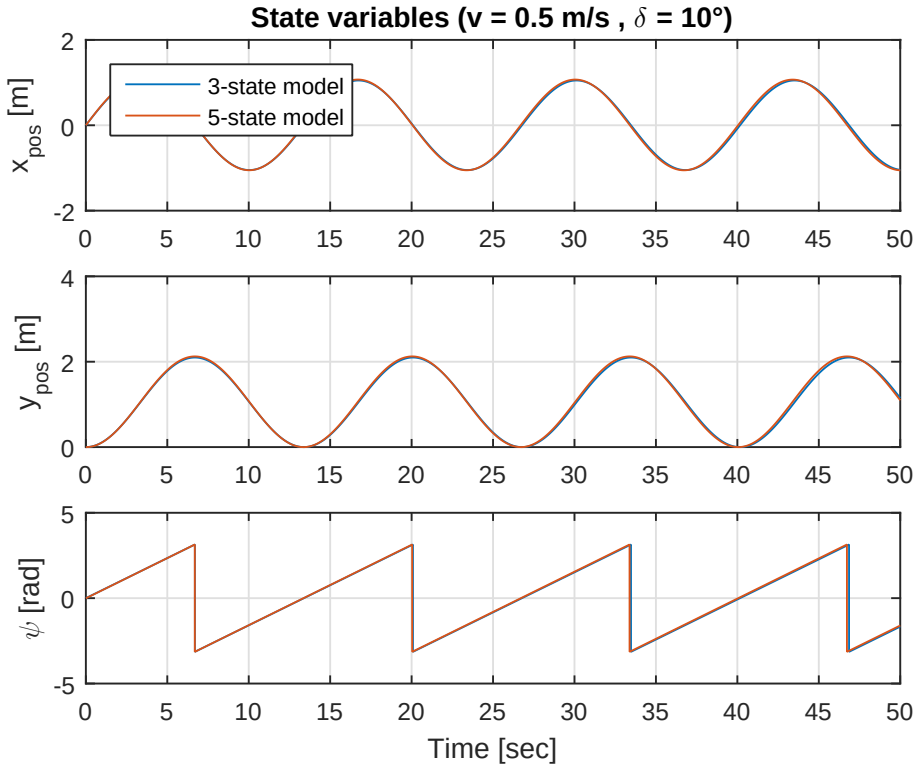
In case 2, now the velocity is reduced ($v = 0.5\,m/s$ and $\delta = 10°$) and both control variables are constant. In Fig. 2.10a it is observed that the position in the XY-plane for both models are more accurate i.e. the error deviation between both models is less. Also in Fig. 2.10b it can be seen that the state variables are more accurate as well. Finally in Fig. 2.10c the error of every state variable is shown, where the resulting values are lower in comparison with the previous case due to the reduction of the velocity, i.e. that the side slip angle is nearly zero.

Case 2		
v	0.5 m/s	(constant)
δ	10°	(constant)
	MSE	RMSE
x_{pos}	1.5960	1.2633
y_{pos}	8.5378	2.9220
ψ_{pos}	1.1290	1.0625

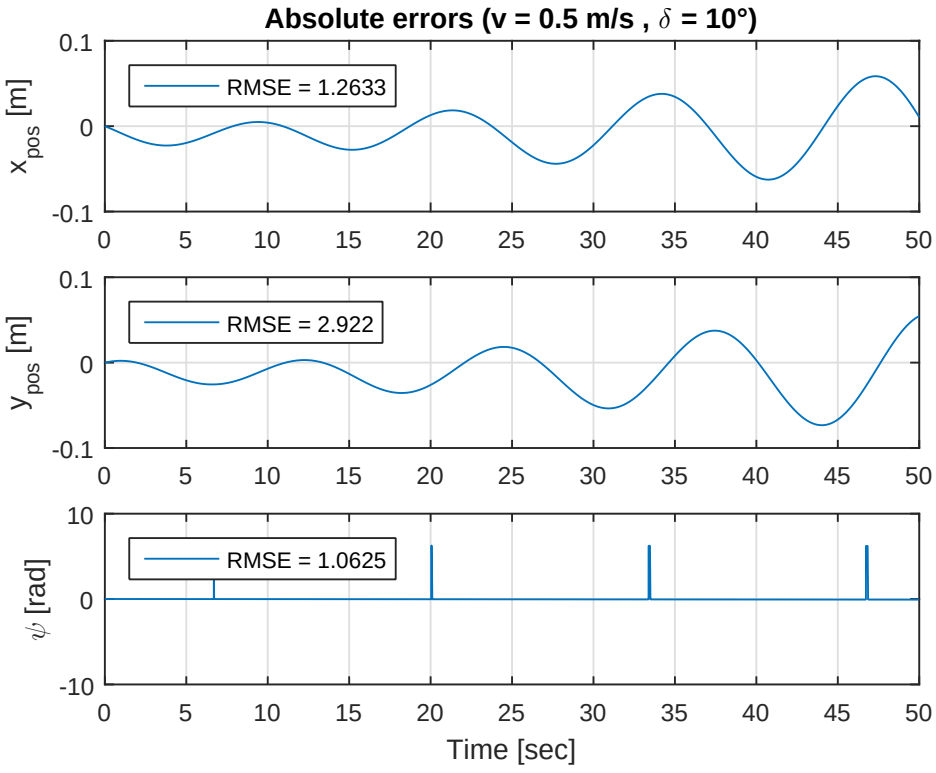
Table 2.5: MSE and RMSE for case 2



(a) Robot trajectory in XY-plane



(b) State variables



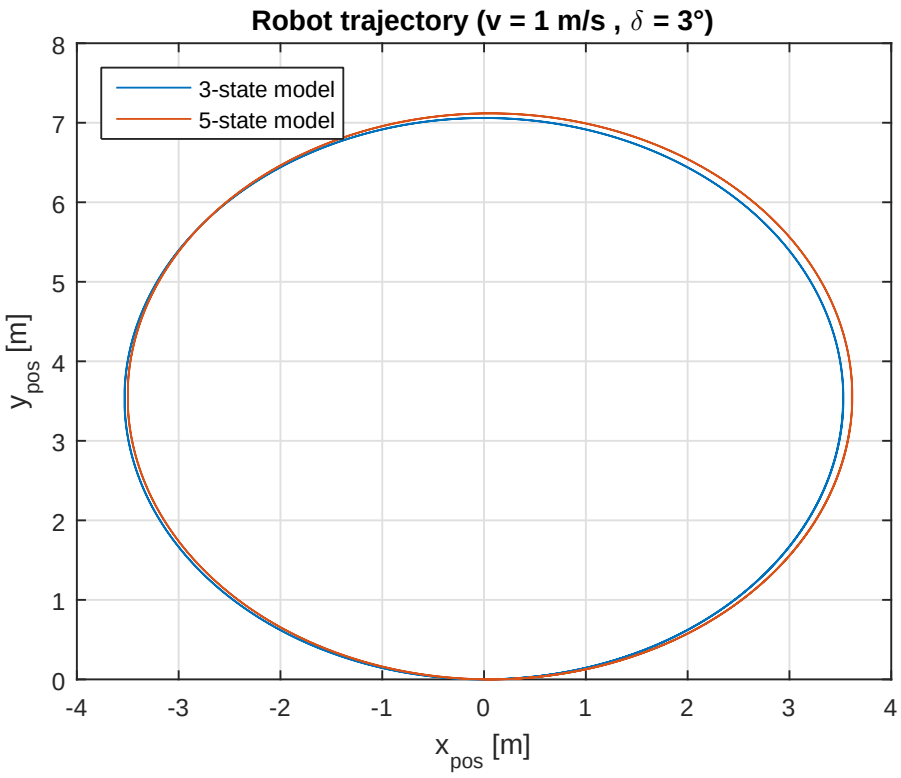
(c) Absolute and RMS errors

Figure 2.10: Results for case 2 ($v = 0.5\ m/s$ and $\delta = 10°$)

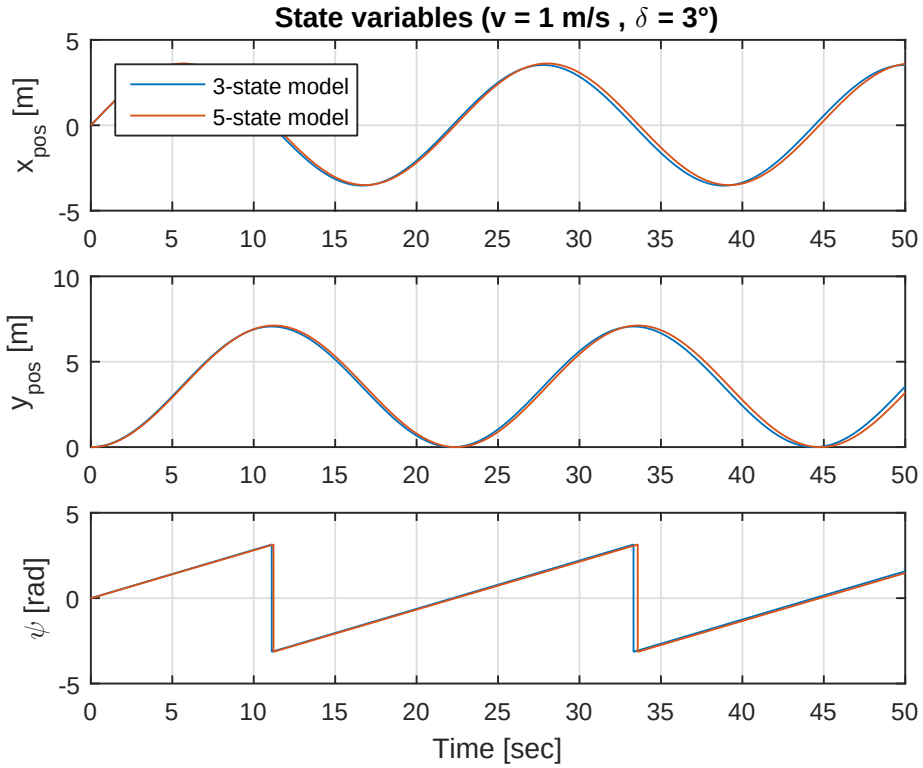
In case 3, now the steering angle is reduced ($v = 1\,m/s$ and $\delta = 3°$). In Fig. 2.11a it is observed that the position in the XY-plane for both models are more accurate than the case 1 but not as the case 2. Also in Fig. 2.11b it can be seen that the state variables are accurate as well but again with a very little deviation. Finally in Fig. 2.11c the error of every state variable is shown as well, where the resulting values are greater than case 2 and lower than case 1 due to the reduction of the steering angle, i.e. that reducing the steering angle is not as relevant as reducing the velocity.

Case 3		
v	1 m/s	(constant)
δ	3°	(constant)
	MSE	RMSE
x_{pos}	61.1445	7.8195
y_{pos}	25.2590	5.0258
ψ_{pos}	5.7378	2.3954

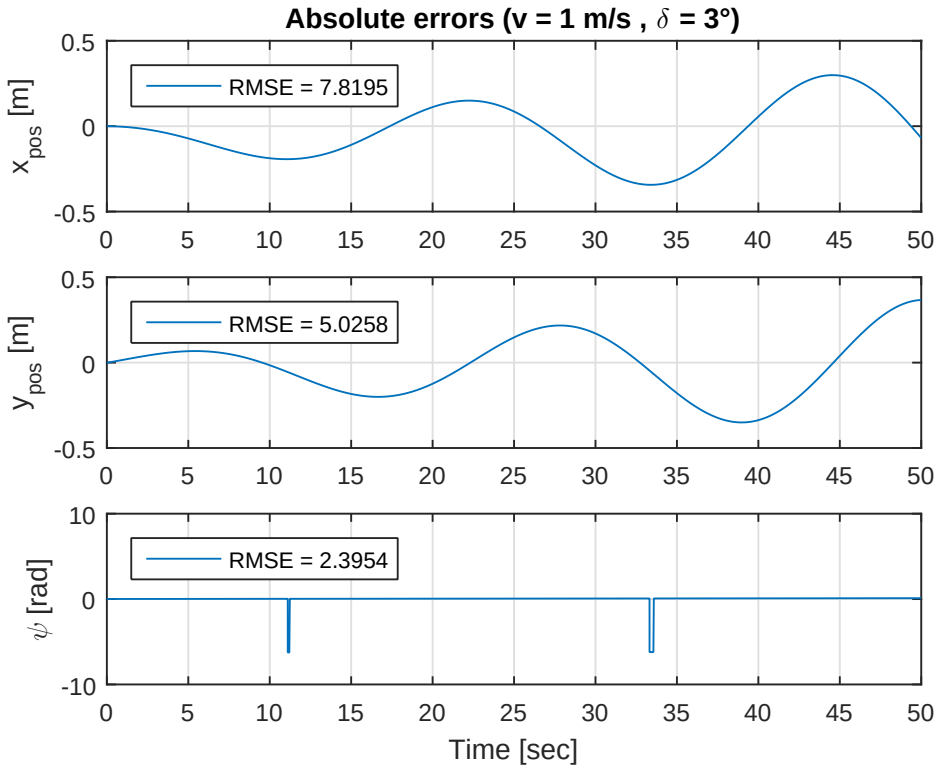
Table 2.6: MSE and RMSE for case 3



(a) Robot trajectory in XY-plane



(b) State variables



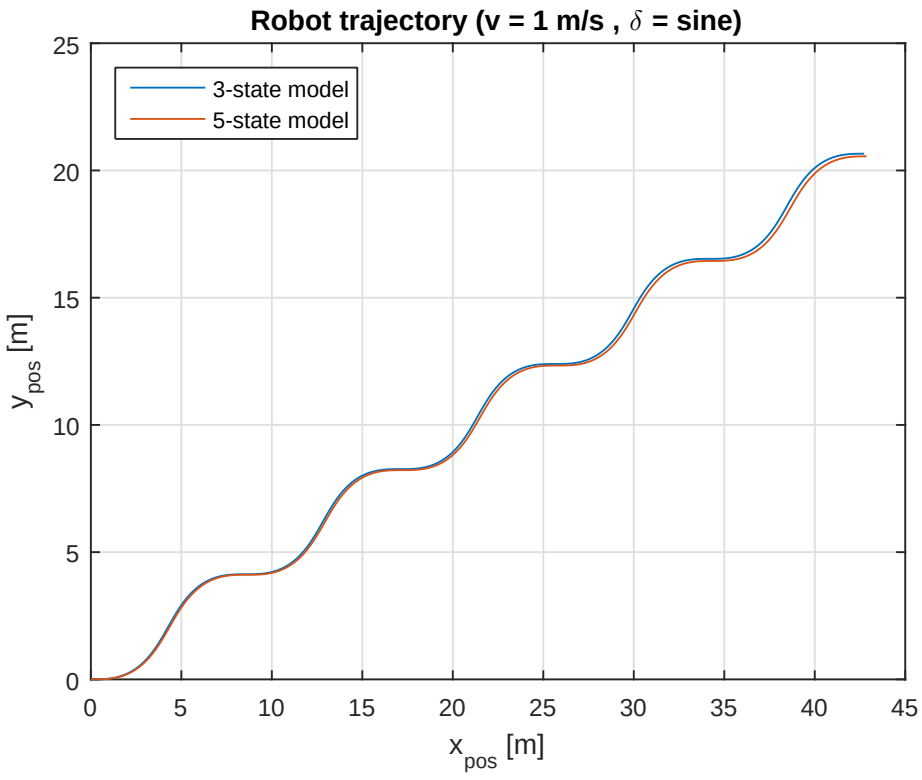
(c) Absolute and RMS errors

Figure 2.11: Results for case 3 ($v = 1\ m/s$ and $\delta = 3°$)

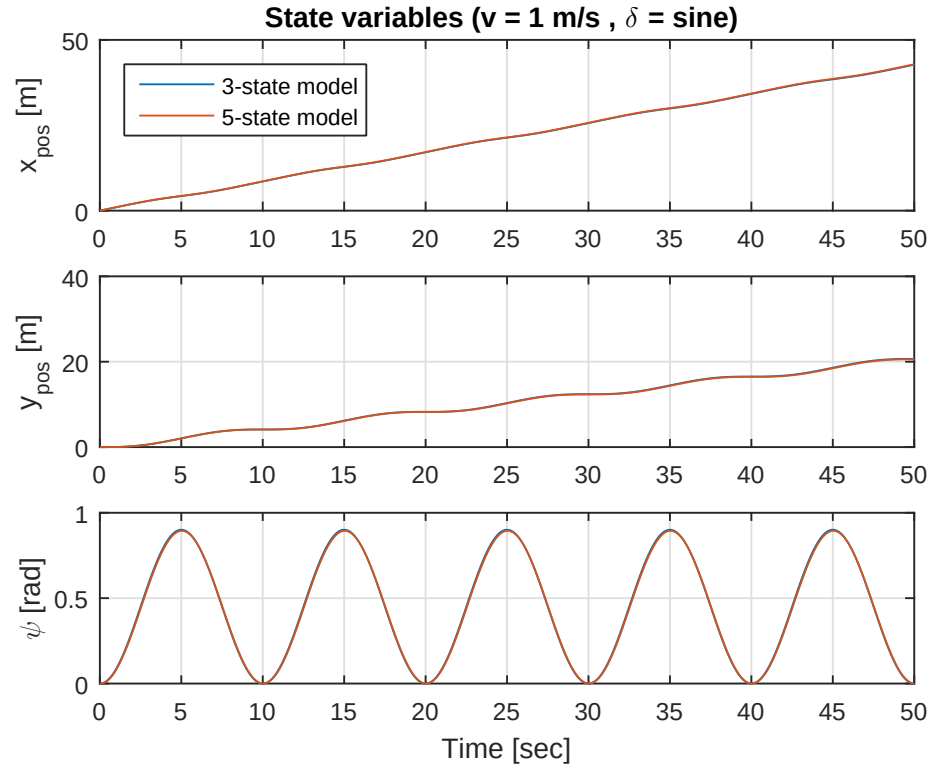
In case 4, now the steering angle is not constant ($v = 1\,m/s$ and $\delta = sine$). In Fig. 2.12a it is observed that the position in the XY-plane comparing both models is accurate. Also in Fig. 2.12b it can be seen that the state variables are very similar. Finally in Fig. 2.12c the error of every state variable is shown, where the resulting error values are increasing due to the dynamics of the mobile robot. Nevertheless the RMS errors are relatively low for a long time horizon. In conclusion, after all the tests were made the 3-state model is validated due to the results observed in every case.

Case 4		
v	1 m/s	(constant)
δ	$3° \sin(wt)$	(variable)
	MSE	RMSE
x_{pos}	255.1136	15.9723
y_{pos}	284.9554	16.8806
ψ_{pos}	0.4791	0.6921

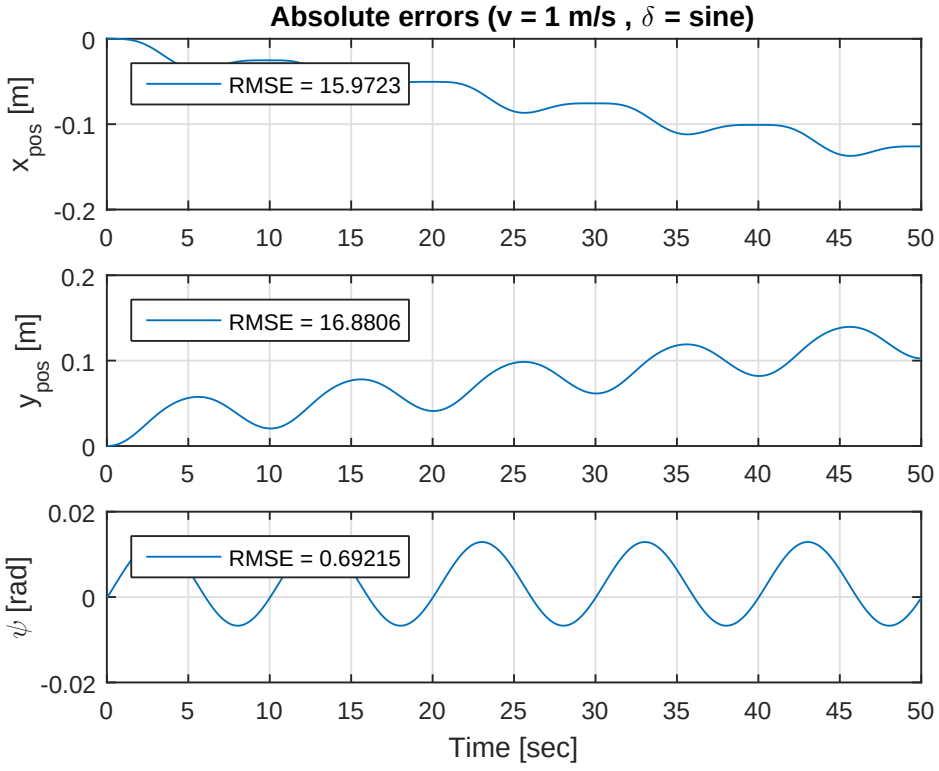
Table 2.7: MSE and RMSE for case 4



(a) Robot trajectory in XY-plane



(b) State variables



(c) Absolute and RMS errors

Figure 2.12: Results for case 4 ($v = 1\ m/s$ and $\delta = sine$)

3 Correction of control variables and estimation of states

In the previous chapter from the system equations it was shown that the robot will move depending on the value of the control variables (velocity and steering angle). Since one is working in a real environment, the position coordinates in the XY-plane and yaw angle depend also on the data of the sensors (given by the encoders) and mechanical inaccuracies related to its construction. So first, it is necessary to check that the transmitted control variables (written values in the program) are the same as the real control variables (received values of the mobile robot), i.e. that a correction of factors must be applied. And second, the data given by the sensors may not be as exact as expected. Therefore, an estimation of the state variables is proposed in order to get a better accuracy. Solving these problems will lead to reduce the error between the calculated and the real positions, i.e. that it will be possible to determine the mobile robot state variables as expected by testing the correction of the control variables in different cases.

3.1 Correction of control variables

During real test executions it was found that the transmitted values of the velocity and the steering angle were not the same as the real received values by the mobile robot. As described below, the velocity has a direct linear relation, while the steering angle has a non-linear relation.

3.1.1 Determination of the velocity

Since the transmitted and real robot's velocity do not match, it is necessary to find a correction factor. For determining this, different test were made using different speed values. So the transmitted velocity (v_{trans}) is the one that is given by the program, and the real velocity (v_{real}) is based on real measurements in distance and time. So

the real velocity is defined as follows,

$$v_{real} = \frac{x_{real}}{t_{real}} \tag{3.1}$$

where v_{real} is the real velocity, x_{real} is the traveled distance in X-direction and t_{real} is the total time of the travel. In Fig. 3.1 it is shown the relation between both velocities and also the approximation line which gives the correction constant.

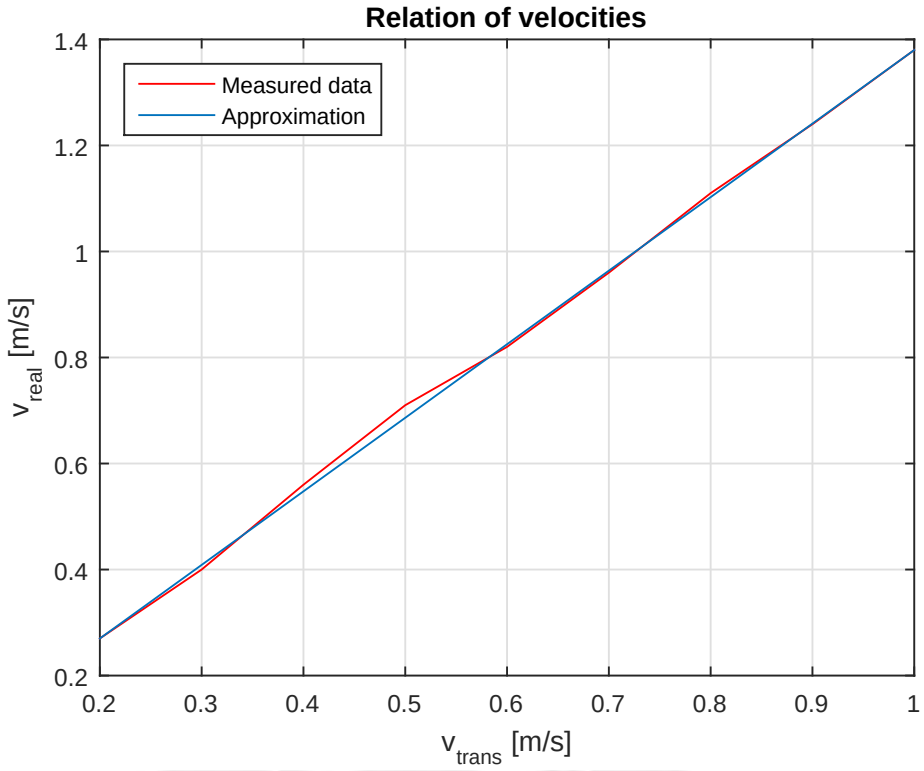


Figure 3.1: Relation between velocities and its linear approximation

$$k_v = \frac{v_{real}}{v_{trans}} = 1.3875 \tag{3.2}$$

From (3.2), the constant value is approximately 1.4 and it is the correction factor of the velocity. So this value must be applied in the program,

$$v_{real} \approx 1.4\, v_{trans} \tag{3.3}$$

3.1.2 Determination of the steering angle

As well as the velocity correction factor, different tests were made using different steering angle values. So the transmitted steering angle (δ_{trans}) is the one that is given by the program and the real steering angle (δ_{trans}) is based on real measurements. In Fig. 3.2 the relation between both steering angle is shown. It can be seen that the relation in this case is non-linear. Thus a solution based on neural networks is proposed as explained also in (Belgradskaia, 2016). Is important to notice that the mobile robot is not affected for small values of the steering angle.

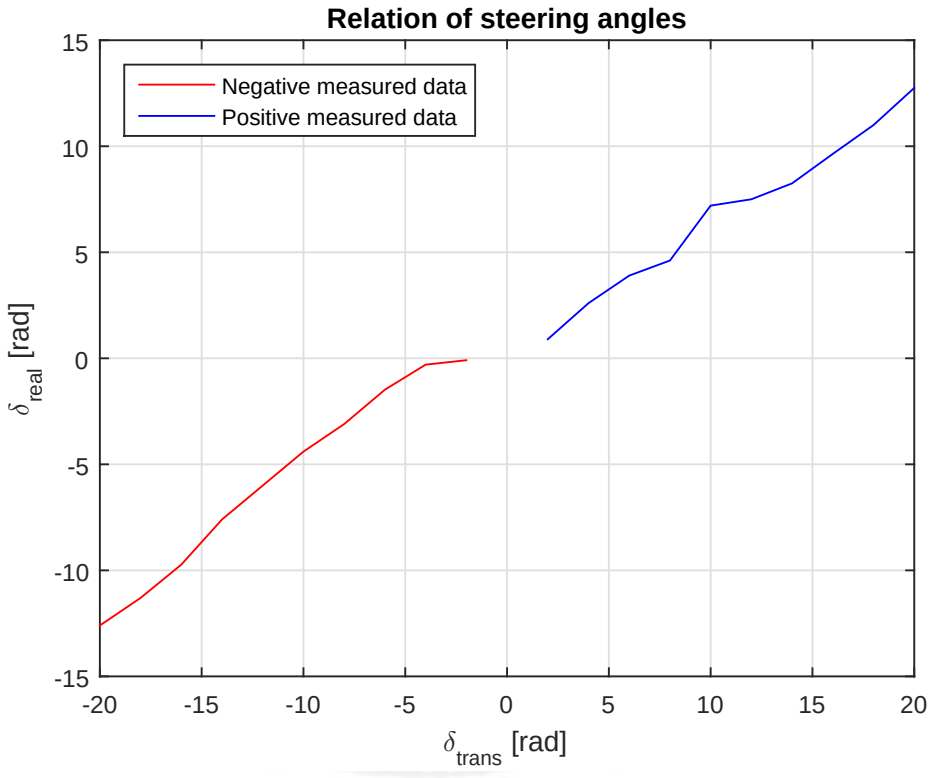


Figure 3.2: Relation between steering angles

An artificial neural network is able to recognize complex dependencies between the input and output data, based on a learning process. This process consist in determining certain weighting factors for each neuron. Each layer is composed by an artificial neuron. The structure of this neuron can be seen in Fig. 3.3. The inputs of the neuron are the weighting values from previous ones, these values are added together and finally the result enters to the activation function. This activation function is a hyperbolic tangent function (non-linear function) because the steering angle has both positive and negative values.

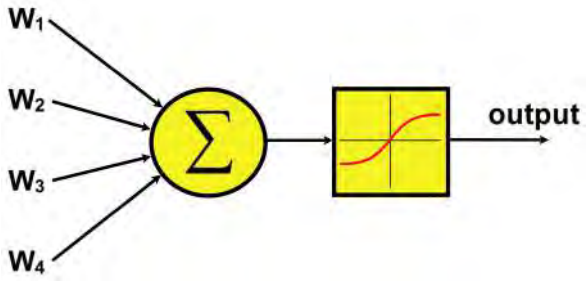


Figure 3.3: Structure of an artificial neuron

The structure of the chosen neural network is the multilayer perceptron (MLP) with a neuron in the input and output layer and six neurons in the hidden layer. So the structure of the entire neural network is as follows,

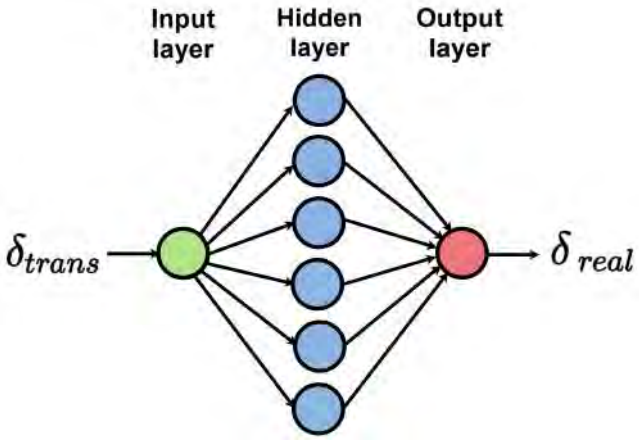


Figure 3.4: Structure of the neural network

The proposed solution deals with the learning method of backpropagation. First, an input value is applied to the neural network and passes through it (from the input layer to the output layer). Then the output value is compared with the desired value, so the deviation between these two values is regarded as the error. This error is now propagated from the output layer to the input layer (backwards), this is the so called backpropagation method. So the weighting factors are altered due to the influence of the error and finally the output value is obtained (δ_{real}). This process is constantly repeated until the error leads to 0. More detailed information in (Haykin, 2011), (Du & Swamy, 2013) and (Gurney, 2003). Thus as observed in Fig. 3.4, having δ_{trans} as the input of the neural network, δ_{real} is obtained at the output. Then this obtained result needs to be transmitted to the mobile robot in order to achieve the desired steering angle.

$$\delta_{real} = F(u) = F(\delta_{trans}) \tag{3.4}$$

3.1.3 Results using correction of speed and steering angle

After the explanation of the correction of the speed and steering angle, different tests were done in order to see the accuracy of the mobile robot trajectory. In case 1, the reference trajectory is a sine function with amplitude of 0.3 m and correction of the control variables is not considered. It can be seen from Fig. 3.5 that the mobile robot tends to go to the left direction and this happens due to its mechanical construction and dynamics.

Case 1	
Reference trajectory	$0.3\ \sin(wt)$
	Correction
v	No
δ	No

Table 3.1: Considerations for case 1

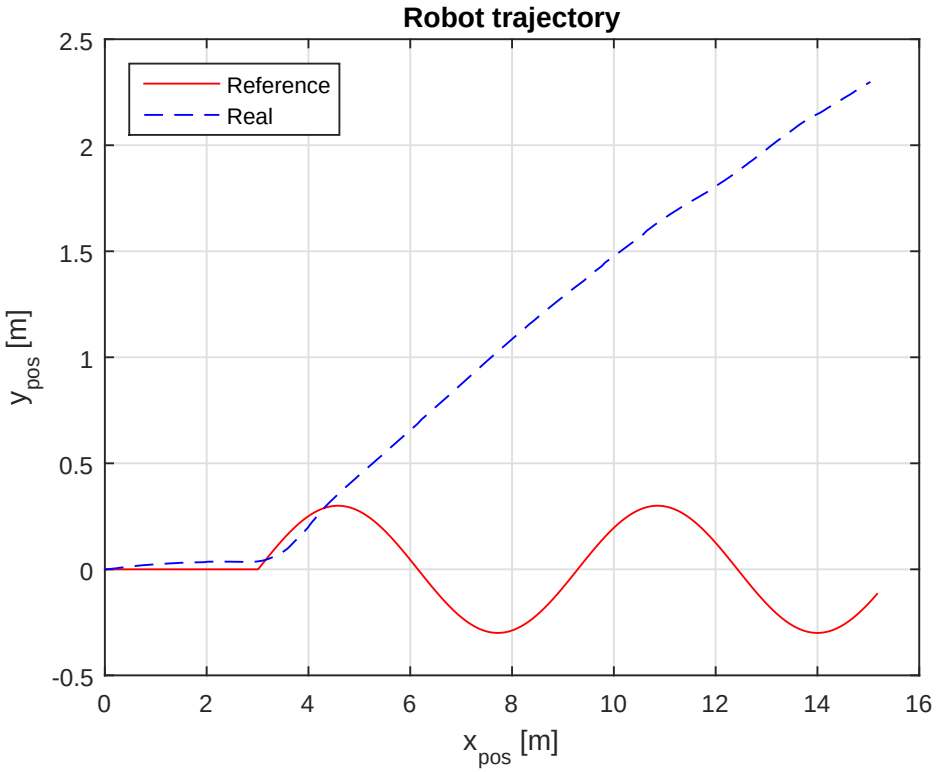


Figure 3.5: Robot trajectory in the XY-plane for case 1

Similarly, in case 2 the reference trajectory is a sine function with amplitude of $-0.3\ m$ and correction of the control variables is not considered. It can be seen from Fig. 3.6 that first the mobile robot tends to go to the right and the ends turning to the left. As same as case 1, this happens due to its mechanical construction and dynamics.

Case 2	
Reference trajectory	-0.3 $\sin(wt)$
	Correction
v	No
δ	No

Table 3.2: Considerations for case 2

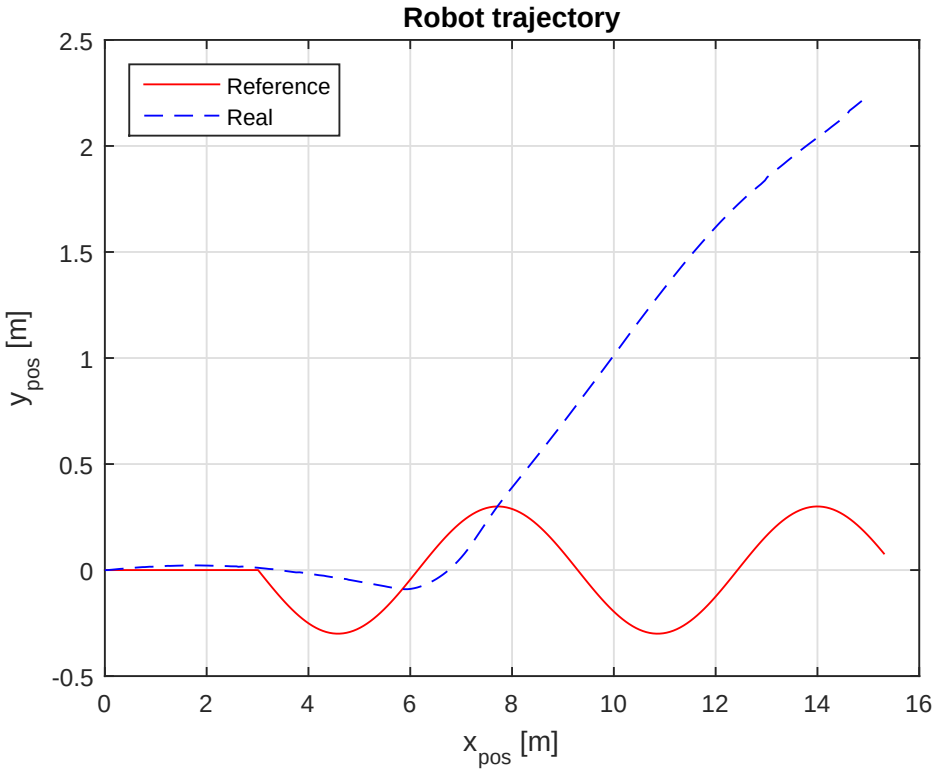
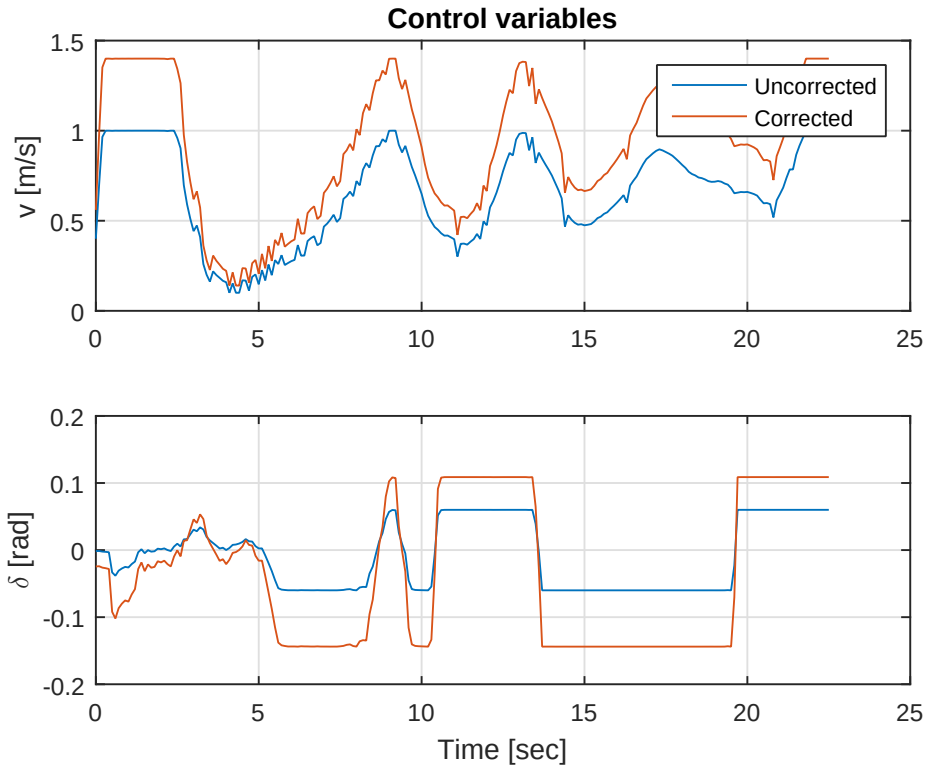


Figure 3.6: Robot trajectory in the XY-plane for case 2

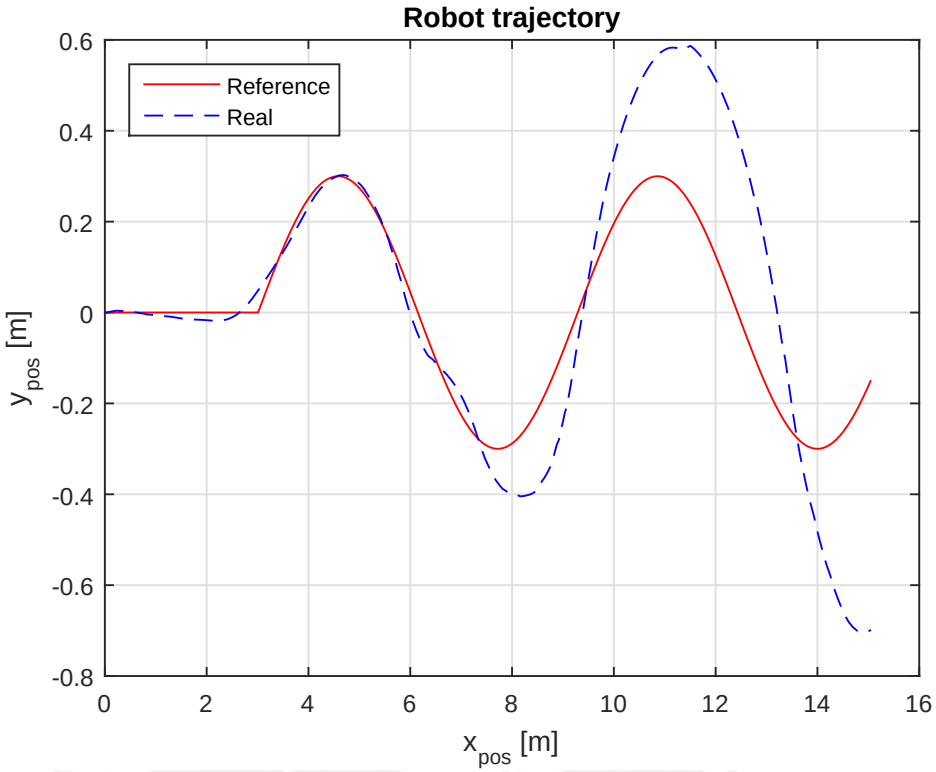
In case 3 the reference trajectory is a sine function with amplitude of 0.3 m and now the correction for both control variables is considered. In Fig. 3.7a the correction of the control variables is shown. In Fig. 3.7b it can be seen that the real trajectory now has the sine form. In Fig. 3.7c the position in the X-direction is equal, while the position in the Y-direction and the yaw angle are not as exact, i.e. that the state variables need to be estimated in order to get an accurate solution.

Case 3	
Reference trajectory	$0.3 \sin(wt)$
	Correction
v	Yes
δ	Yes

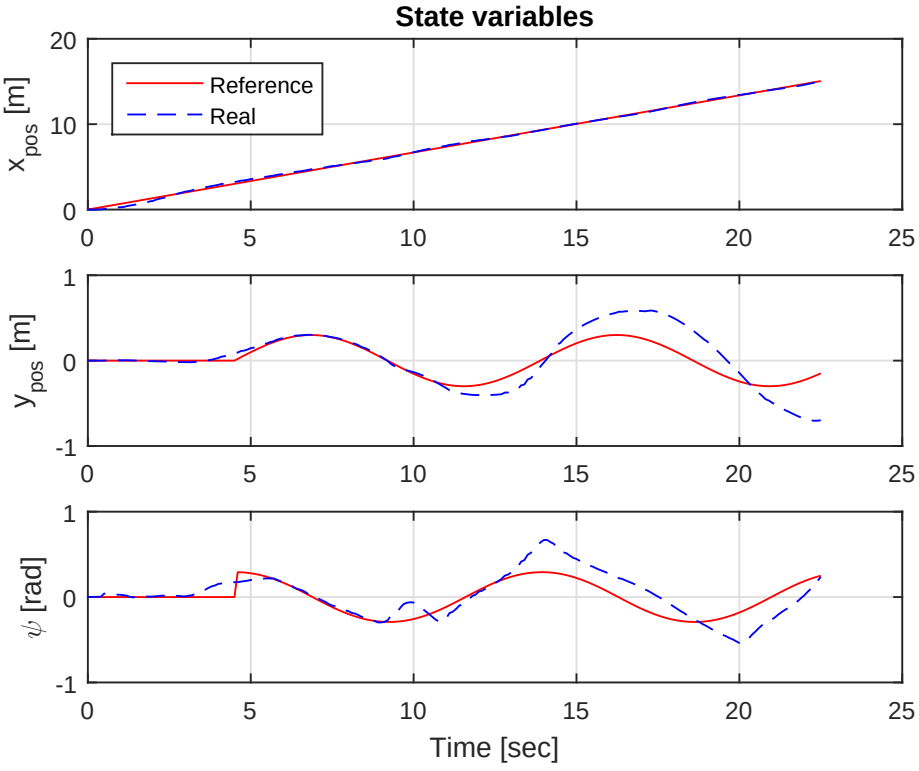
Table 3.3: Considerations for case 3



(a) Corrected and uncorrected control variables



(b) Robot trajectory in the XY-plane



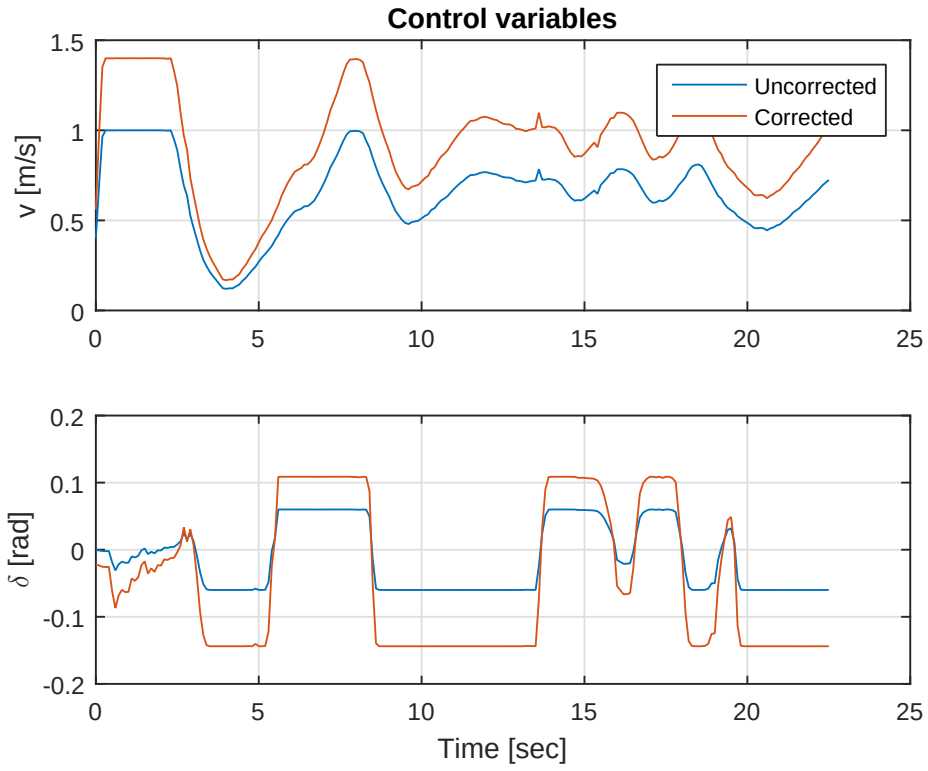
(c) State variables

Figure 3.7: Mobile robot behavior results for case 3

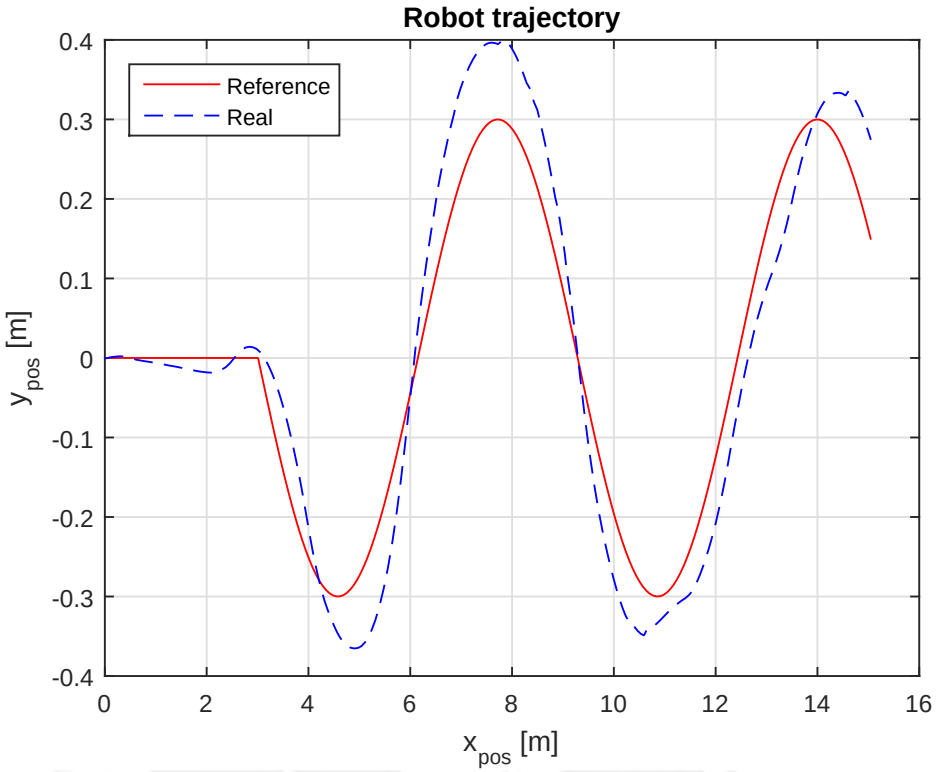
In case 4 the reference trajectory is a sine function with amplitude of $-0.3\ m$ and now the correction for both control variables is again considered. In Fig. 3.8a the correction of the control variables is shown. In Fig. 3.8b it can be seen that the real trajectory has the sine form and is more accurate than the case 3. In Fig. 3.8c the position in the X-direction is equal, while the position in the Y-direction and the yaw angle are almost equal. In order to get an accurate solution the state variables need to be estimated. Finally, it can be seen from all the cases that correcting the control variables improve the behavior of the mobile robot.

Case 4	
Reference trajectory	-0.3 $\sin(wt)$
	Correction
v	Yes
δ	Yes

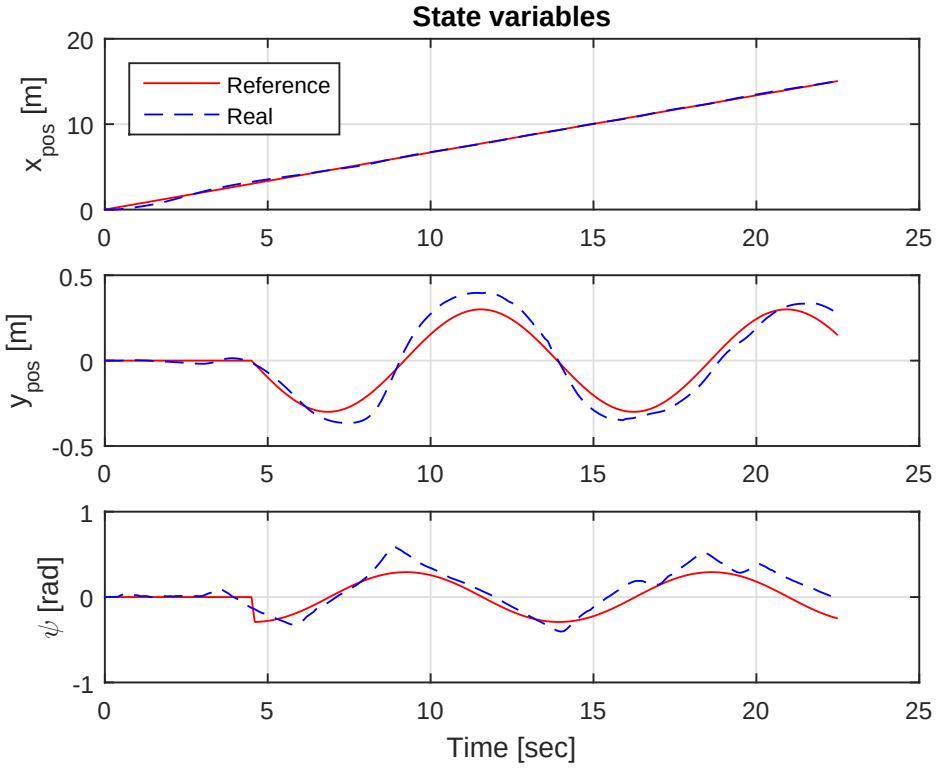
Table 3.4: Considerations for case 4



(a) Corrected and uncorrected control variables



(b) Robot trajectory in the XY-plane



(c) State variables

Figure 3.8: Mobile robot behavior results for case 4

3.2 Estimation and prediction using Extended Kalman Filter

As commented in the initial part of this chapter, another task that arises is that the data of the sensors must be corrected because it affects the position determination. A suitable solution is to use filters. The extended Kalman filter (EKF) will leads to improve the accuracy of mobile robot position.

3.2.1 Main idea of the Extended Kalman Filter

The Kalman filter is an algorithm developed by Rudolf E. Kalman in 1960 and it helps to identify that non-measurable states of a dynamic system, also can have statistical noise and other inaccuracies. This filter is a recursive filter that uses a series of measurements observed over time, containing statistical noise and other inaccuracies. It produces a more precise estimation of the unknown variables in comparison with the ones that were just measured alone. The Kalman filter has many applications, but commonly it is often used in guidance, navigation, and control of vehicles. More information available in (Zarchan & Musoff, 2000), (Haykin, 2004), (Grewal & Andrews, 2001) and (Simon, 2006).

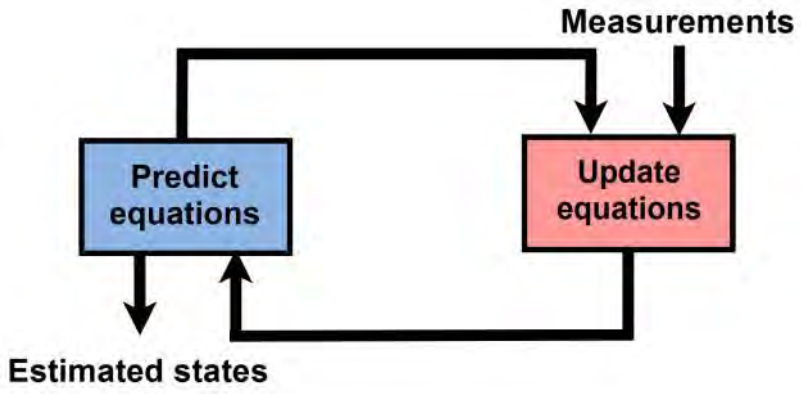


Figure 3.9: Structure of the EKF

The algorithm works in two steps. The first step is the prediction, where the Kalman filter produces estimates of the current state variables, along with their uncertainties. The second step is the correction, where the estimates are updated using a weighted average using the measured data from sensors. The algorithm is recursive. It can run in real time, using only the present input measurements, the previously calculated state and its uncertainty matrix, so the advantage is that no additional past information is

required. The EKF is an extension of this method, and is the nonlinear version of the Kalman filter which linearizes about an estimate of the current mean and covariance.

3.2.2 Extended Kalman Filter applied to the mobile robot

In the EKF, the state transition and observation models don't need to be linear functions of the state but may instead be differentiable functions. So (3.5) and (3.6) show the relationship between the states variables and the measured values in the time interval $[k-1, k]$.

$$x_k = f(x_{k-1}, u_{k-1}) + w_k \tag{3.5}$$

$$z_k = h(x_k) + p_k \tag{3.6}$$

where x_k is the vector of actual states, z_k is the vector of measured values, w_k and p_k are the process and observation noises (zero mean value) with covariance matrices Q_k and R_k, respectively, $f(x_{k-1}, u_{k-1})$ is the system function based on previous state and control values and $h(x_k)$ is the function that relates the vector of actual states with the vector of measured values.

Regarding the 3-state model described in (2.22) from chapter 2, the state and control variables vectors in the time instant k are,

$$x_k = \begin{pmatrix} x_k \\ y_k \\ \psi_k \end{pmatrix} \quad , \quad u_k = \begin{pmatrix} v_k \\ \delta_k \end{pmatrix} \tag{3.7}$$

Then using the mathematical model described in (2.23) with (3.7), the state variables vector in the instant time k is determined as follows,

$$x_k = f(x_{k-1}, u_{k-1}) = \begin{pmatrix} x_{k-1} + \Delta t\, v_{k-1}\, \cos(\psi_{k-1})\, \cos(\delta_{k-1}) \\ y_{k-1} + \Delta t\, v_{k-1}\, \sin(\psi_{k-1})\, \cos(\delta_{k-1}) \\ \psi_{k-1} + \Delta t\, \frac{2\, v_{k-1}}{L}\, \sin(\delta_{k-1}) \end{pmatrix} \tag{3.8}$$

where Δt is the step size of the time interval $[k-1, k]$.

The algorithm of the EKF is as follows. First in the predictive stage, two predicted

vectors need to be calculated,

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_{k-1}) \tag{3.9}$$
$$P_{k|k-1} = F_{k-1}\, P_{k-1|k-1}\, F_{k-1}^T + Q_{k-1} \tag{3.10}$$

where $\hat{x}$ is the predicted state estimate vector, P is the predicted covariance matrix, and F is the state transition matrix.

In the EKF equations the Jacobian is used with the current predicted states at each time step in order to linearize the nonlinear function around the current state. Then the Jacobian is defined by,

$$F_{k-1} = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1|k-1}, u_{k-1}} \tag{3.11}$$

then,

$$F = \begin{pmatrix} 1 & 0 & -v\,\sin(\hat{\psi})\cos(\delta)\Delta t \\ 0 & 1 & v\,\cos(\hat{\psi})\cos(\delta)\Delta t \\ 0 & 0 & 1 \end{pmatrix} \tag{3.12}$$

Also from (3.10), the covariance matrix Q is by,

$$Q = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 50 \end{pmatrix} \tag{3.13}$$

After all the prediction calculations were made, the next step is to measure the states, find the Kalman gains and correct updating all the state values. So starting with the measurement residual vector and the residual covariance matrix,

$$\tilde{y}_k = z_k - h(\hat{x}_{k|k-1}) \tag{3.14}$$
$$S_k = H_k\, P_{k|k-1}\, H_k^T + R_k \tag{3.15}$$

where H is the observation matrix and is defined also by the Jacobian as follows,

$$H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{k|k-1}} \tag{3.16}$$

But since the values for all the state variables (position in the XY-plane and yaw angle)

are given by the mobile robot, it is not necessary to calculate them again, this means that

$$h(x_k) = x_k \tag{3.17}$$

then,

$$H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.18}$$

Also from (3.15), the covariance matrix R is defined by,

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.19}$$

It is important to remark that both covariances matrices Q and R were defined empirically. The criterion used for the choice of values of the diagonal of the weight matrix Q depended on the system response and the performance index value. In the case of the matrix R, the same weight was placed on the velocity and steering angle.

So the optimal Kalman gain matrix needs to be calculated as follows,

$$K_k = P_{k|k-1}\, H_k^T\, S_k^{-1} \tag{3.20}$$

And finally it is needed to update the values of the estimated state vector and the estimated covariance matrix,

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\, \tilde{y}_k \tag{3.21}$$

$$P_{k|k} = (I - K_k\, H_k)\, P_{k|k-1} \tag{3.22}$$

3.2.3 Results using Extended Kalman Filter

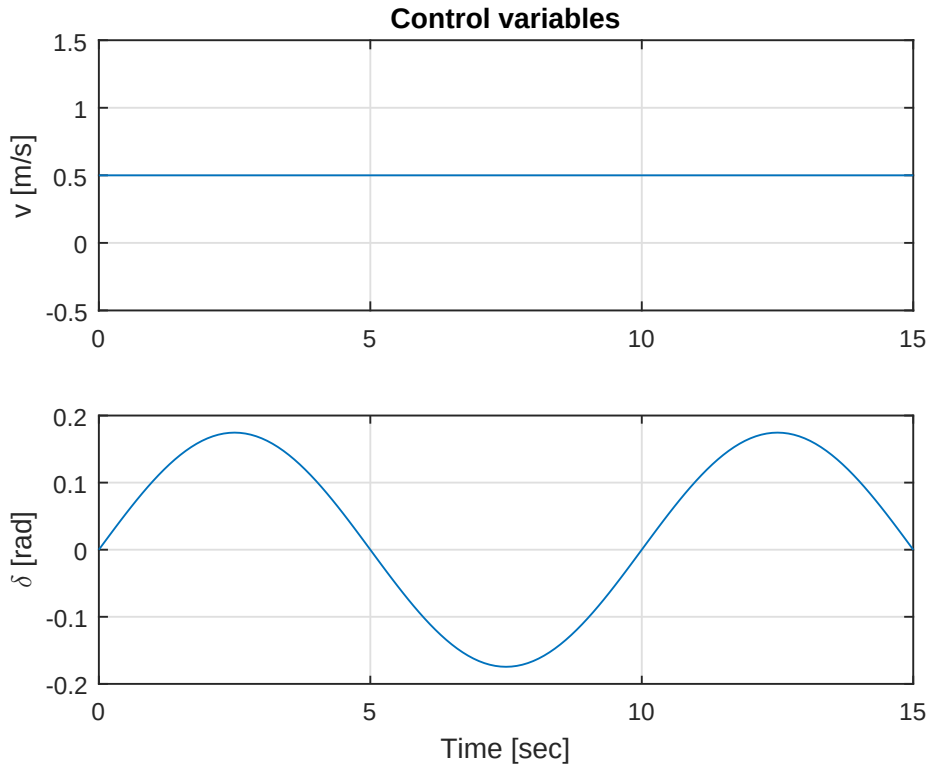
As same as previous results using just correction of control variables, different tests were needed in order to see the accuracy of the mobile robot trajectory applying the EKF explained previously.

In case 1 a simulation of the mobile robot behavior is presented, where the velocity is constant, the steering angle is a sine function (see Fig. 3.10a), and noise from sensors

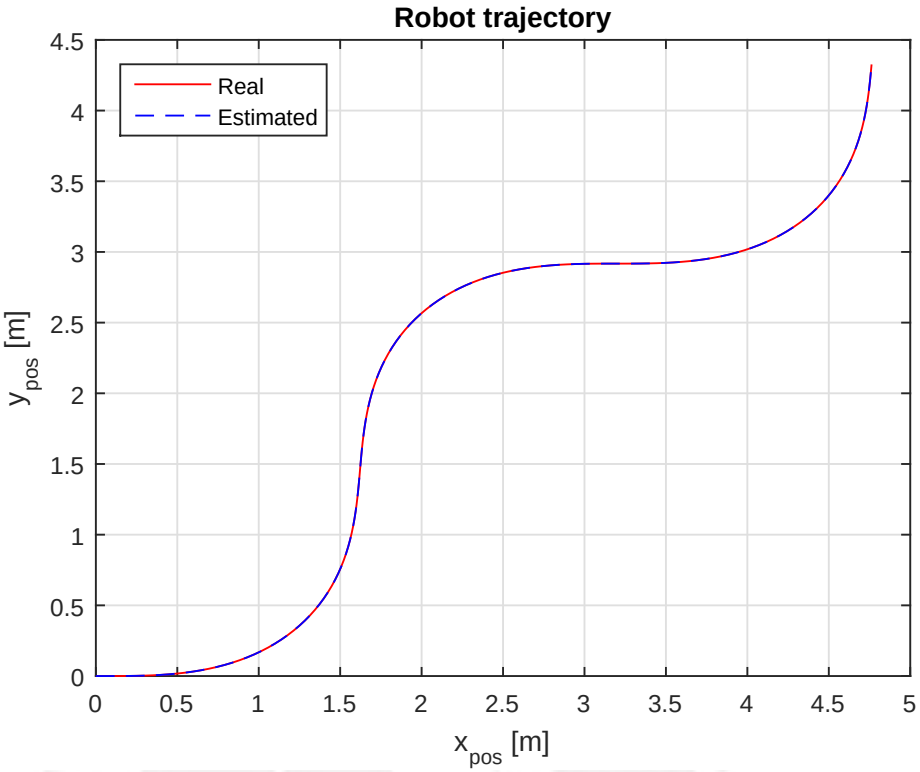
is not considered in the system. From Fig. 3.10b and 3.10c it can be observed that the estimated and real state variables are equal, i.e. that the errors between these variables converge to zero. Is important to notice that the initial states influence on the mobile robot behavior, for this case the mobile robot has all its initial states in zero.

Case 1	(Simulation)
v	0.5 m/s
δ	$10° \sin(wt)$
Noise	No
	Initial States
$x_{pos,0}$	0 m
$y_{pos,0}$	0 m
ψ_0	0°

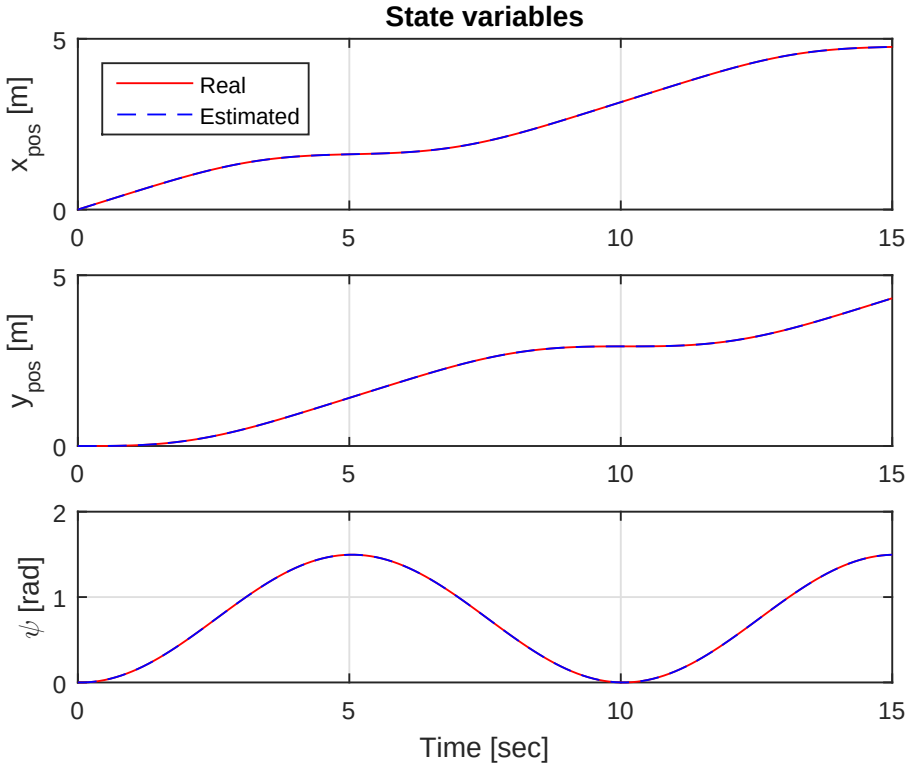
Table 3.5: Considerations for case 1



(a) Control variables



(b) Robot trajectory in the XY-plane



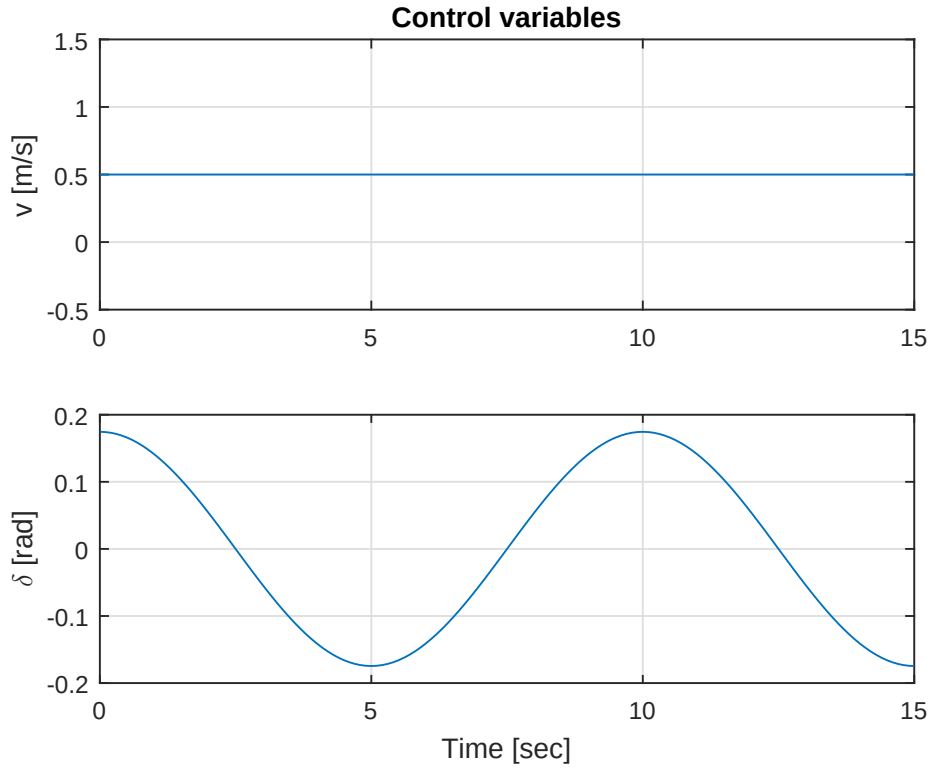
(c) State variables

Figure 3.10: Mobile robot behavior results for case 1

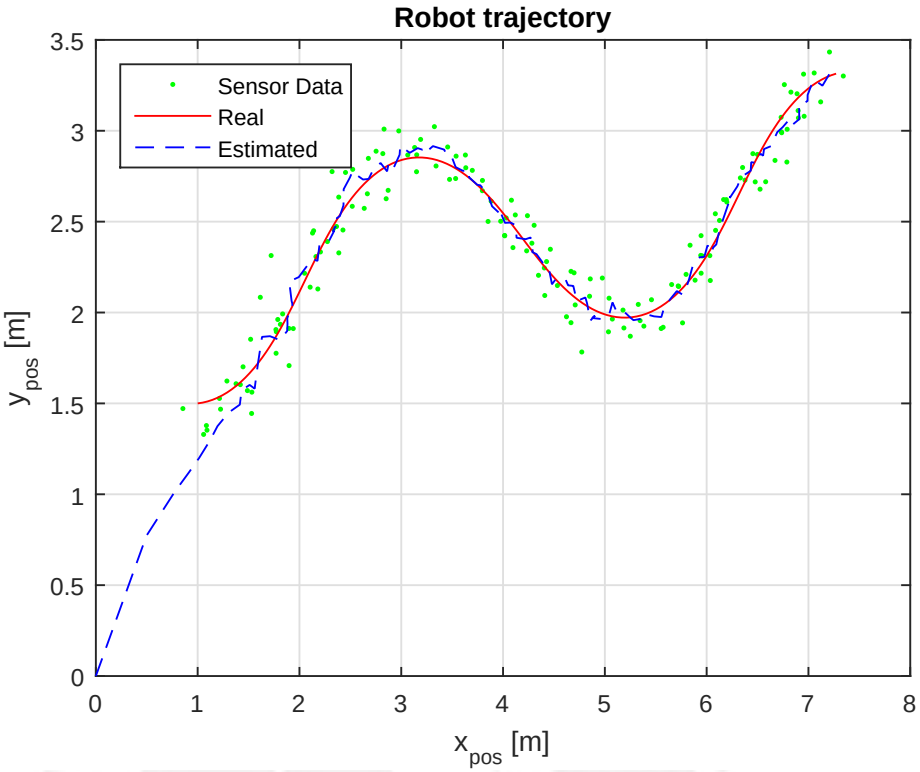
In case 2 a simulation of the mobile robot behavior is presented as well, where the velocity is constant, the steering angle is a cosine function (see Fig. 3.11a), and now noise from sensors is considered in the system due to behavior in a real environment. From Fig. 3.11b and 3.11c it can be observed that estimated state variables follow very well to the real trajectory even the initial state variables are different from zero.

Case 2	(Simulation)
v	0.5 m/s
δ	$10° \cos(wt)$
Noise	Yes
	Initial States
$x_{pos,0}$	1 m
$y_{pos,0}$	1.5 m
ψ_0	5°

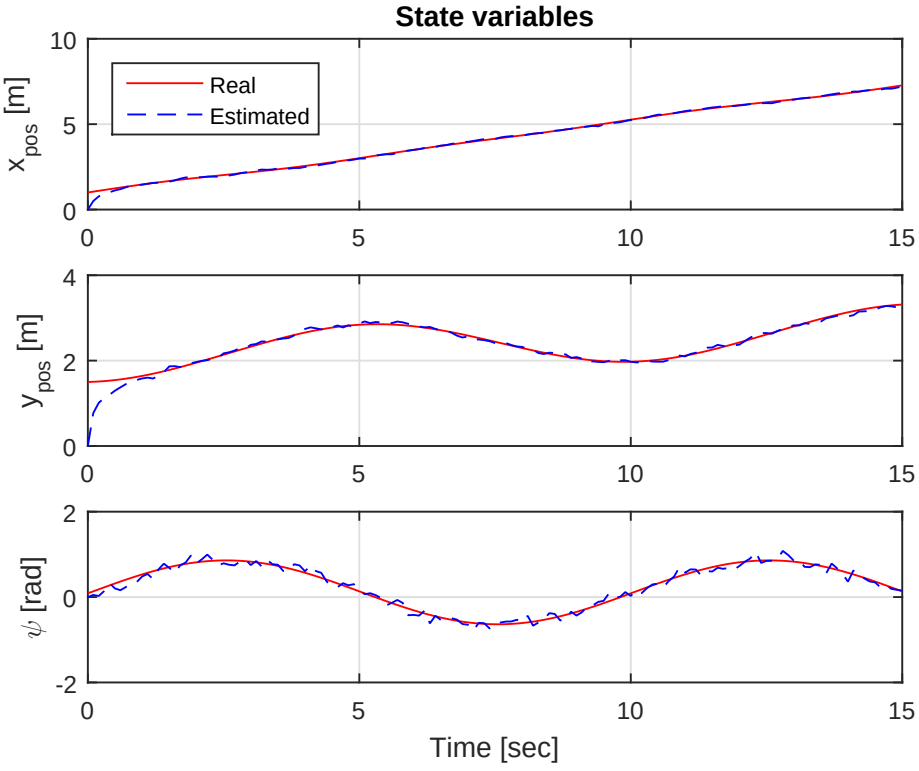
Table 3.6: Considerations for case 2



(a) Control variables



(b) Robot trajectory in the XY-plane



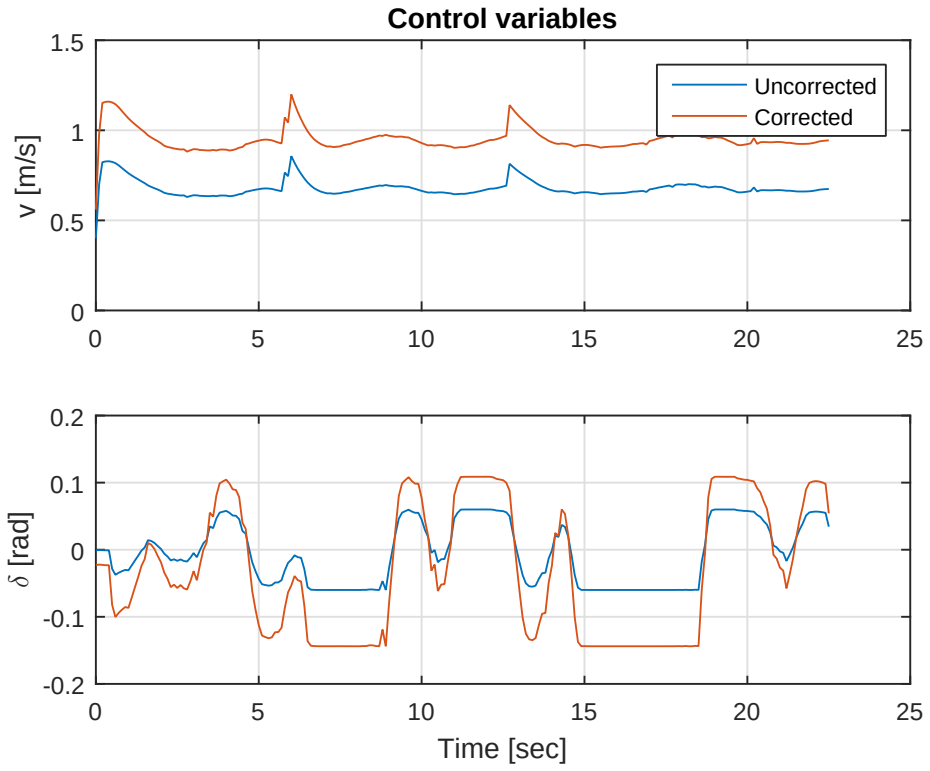
(c) State variables

Figure 3.11: Mobile robot behavior results for case 2

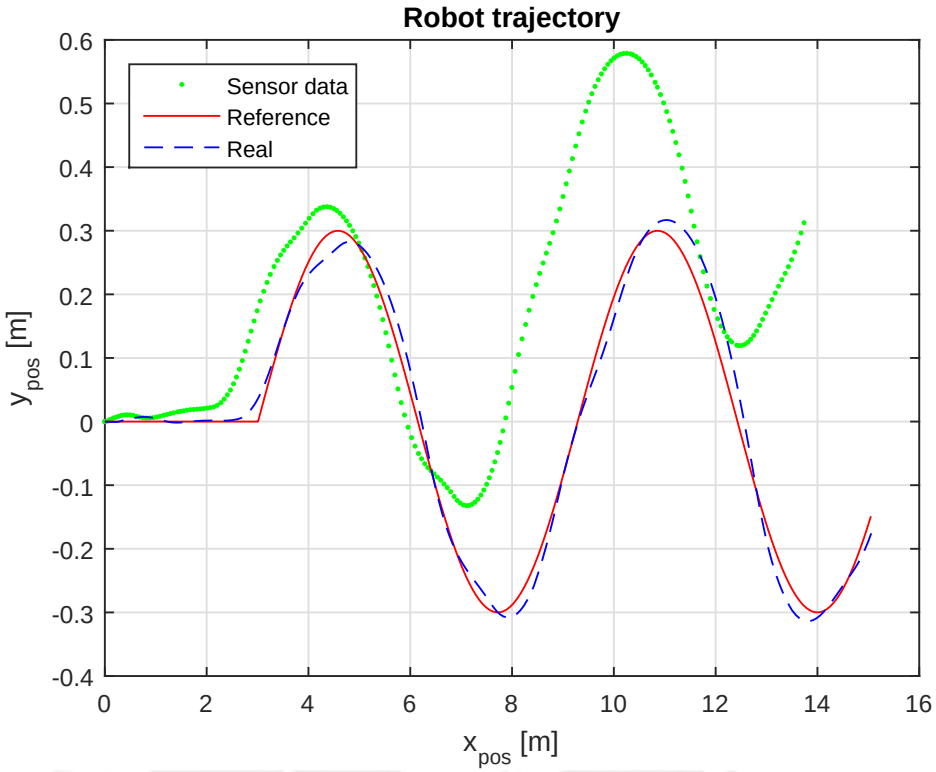
In case 3 a real test of the mobile robot behavior is presented, where the reference trajectory is a sine function with amplitude of 0.3 m. For this case the control variables are not constant (see Fig. 3.12a). From Fig. 3.12b it can be seen that the data given by the sensor has the same form as the reference trajectory. From Fig. 3.12c it can be seen that the position in the X and Y-directions are accurate, while the yaw angle has some differences between the real and the reference values due to the dynamics of the mobile robot.

Case 3	(Real)
Reference trajectory	$0.3\sin(wt)$
	Initial States
$x_{pos,0}$	0 m
$y_{pos,0}$	0 m
ψ_0	0°

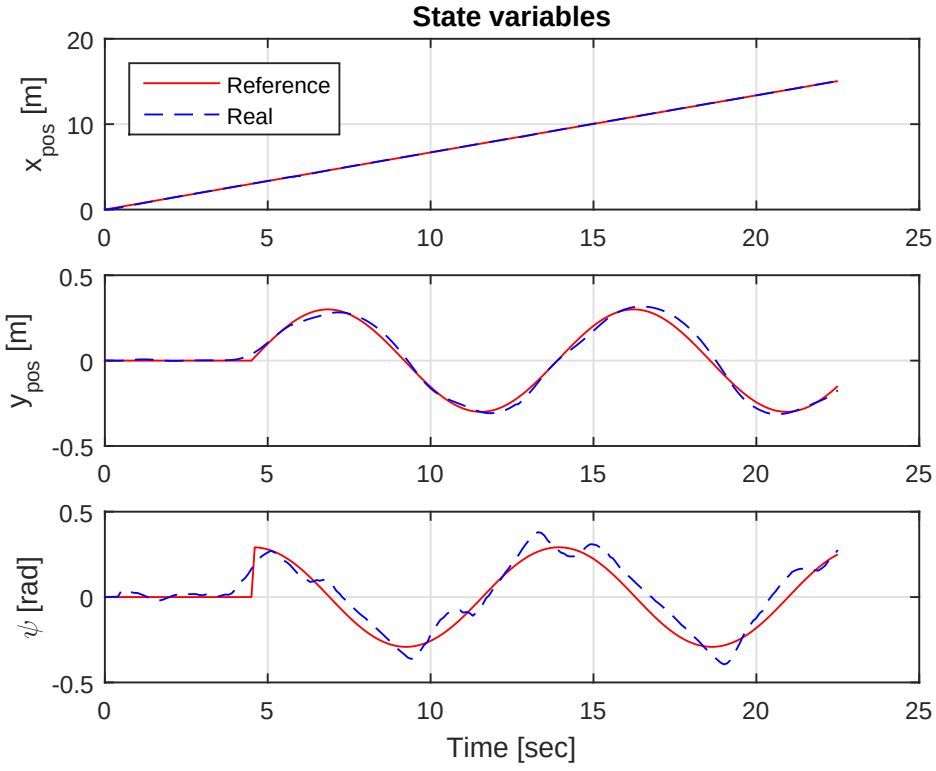
Table 3.7: Considerations for case 3



(a) Control variables



(b) Robot trajectory in the XY-plane



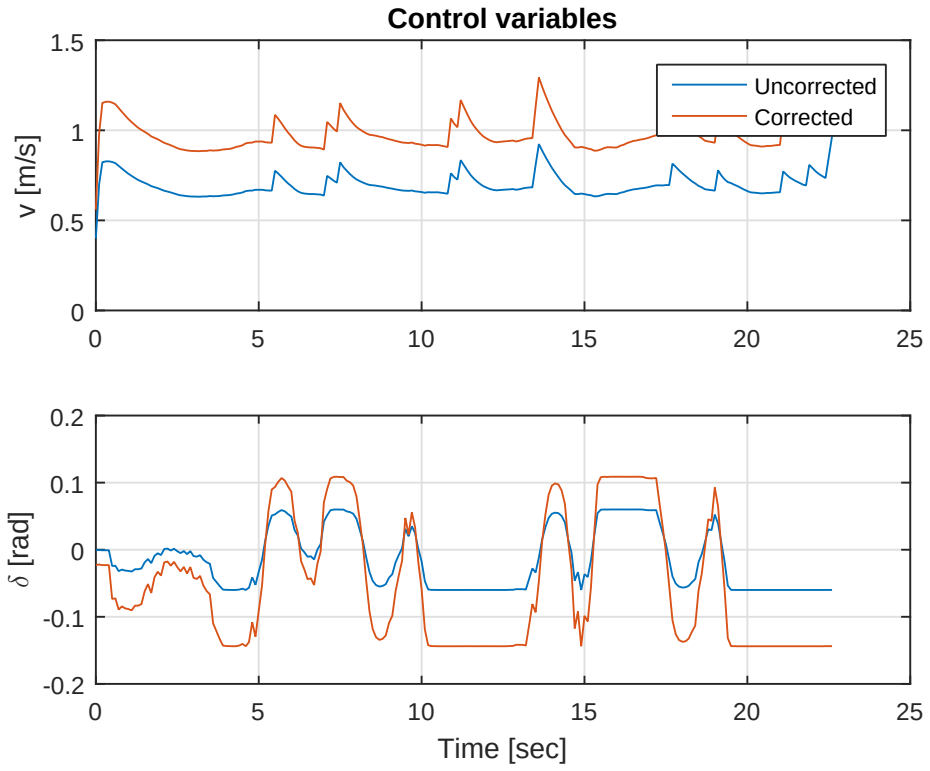
(c) State variables

Figure 3.12: Mobile robot behavior results for case 3

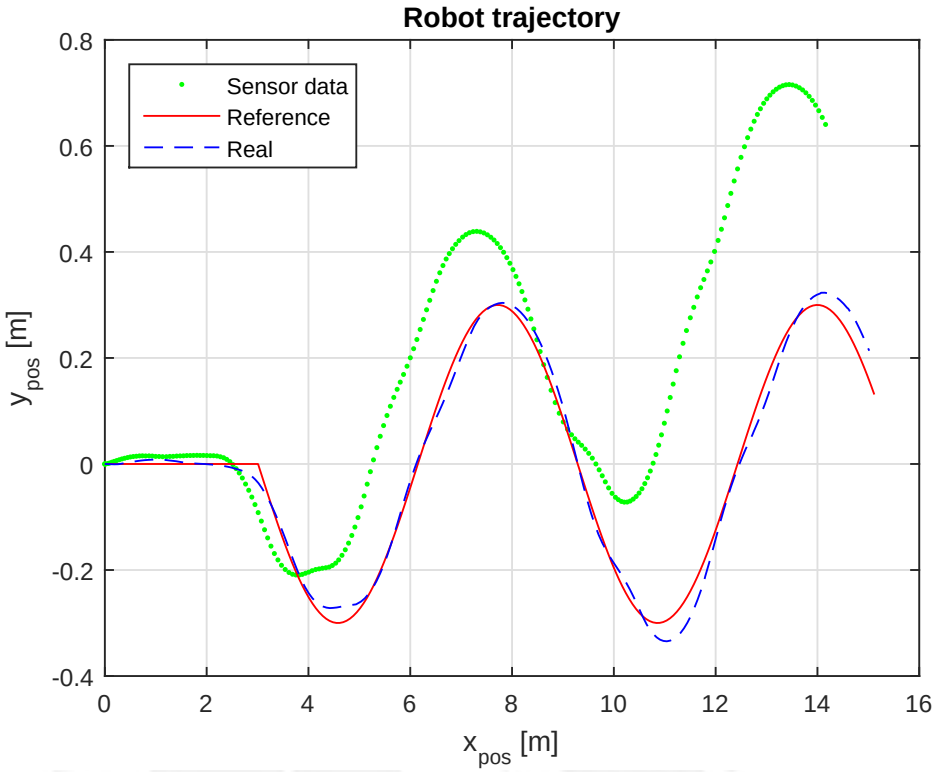
In case 4 a real test is presented as well, where the reference trajectory is a sine function with amplitude of -0.3 m. For this case the control variables are not constant (see Fig. 3.13a). From Fig. 3.13b it can be seen that the data given by the sensor has the same form as the reference trajectory. From Fig. 3.13c it is observed that the position in the X and Y-directions are accurate, as well as the yaw angle. Finally it is demonstrated that using EKF for prediction and estimation helps to get an accurate estimation of the state variables.

Case 4	(Real)
Reference trajectory	-0.3 sin(wt)
	Initial States
$x_{pos,0}$	0 m
$y_{pos,0}$	0 m
ψ_0	0°

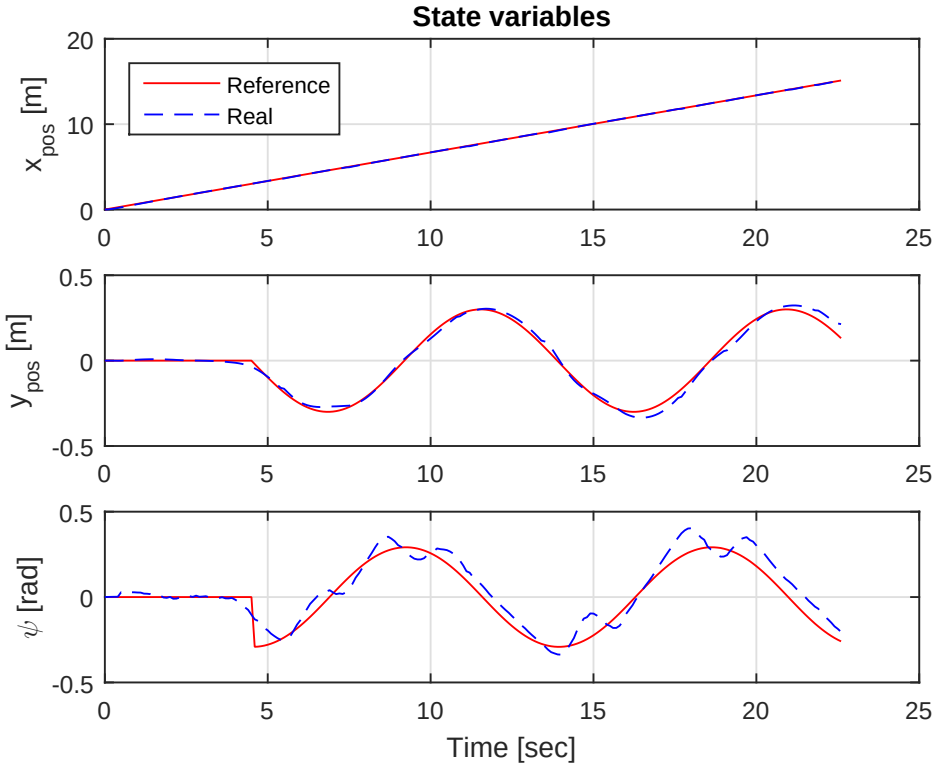
Table 3.8: Considerations for case 4



(a) Control variables



(b) Robot trajectory in the XY-plane



(c) State variables

Figure 3.13: Mobile robot behavior results for case 4

4 Computer vision

Computer vision is a scientific discipline that includes different methods e.g. acquire, process, analyze and understand real-world images in order to produce numerical or symbolic information so that they can be manipulated by a computer. As humans use their eyes and brains to understand the world around them, computer vision tries to produce the same effect so that computers can perceive and understand an image or sequence of images and behave as appropriate in a given situation. This understanding is achieved through different fields such as geometry, physics, statistics, and other disciplines. In general, the acquisition of data is achieved by various means such as image sequences, views from several video cameras or multidimensional data from a medical scanner (in the medicine's field). There are many technologies that use computer vision, among which are the recognition of objects, the detection of events, the reconstruction of a scene (mapping) and the restoration of images. Nowadays image processing is bringing great contributions to actual technology e.g. in mobile robot's autonomous driving.

In this chapter, it will be discussed the recognition of obstacles along the desired trajectory of the mobile robot. Later it will be seen that this information will be combined with the laser scanner data in order to get a better knowledge and information about the characteristics of these obstacles. Since there are many ways to detect obstacles, in this work the process for identifying them is divided in two steps. The first step is the detection process, as its name says, an algorithm is applied here (shape detection, color detection or the combination of both) for detecting the obstacle in an image. The second step is the cluster classification process, where all the detected obstacles in the image are separated by clusters or groups based on Euclidean distance algorithm. Fig. 4.1 shows a block diagram of the entire process for identifying an obstacle. OpenCV[2] library version 3.2 is used for image processing. More information detailed in (Kaehler & Bradski, 2016) and (*OpenCV Library*, 2017).

[2]OpenCV is a library of programming functions mainly aimed at real-time computer vision

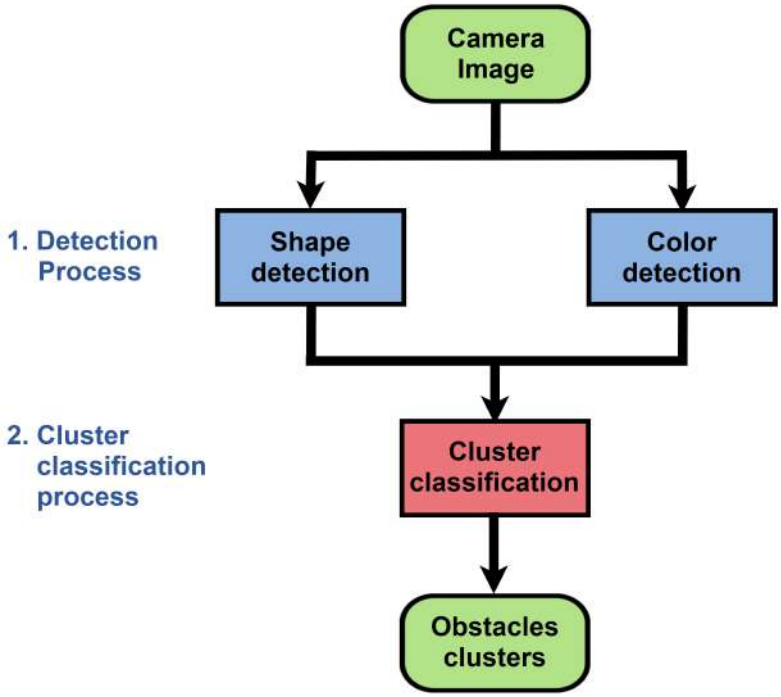


Figure 4.1: Block diagram for identifying an obstacle

4.1 Detection process

The detection process is the first step for identifying obstacles. The most used algorithms are the shape and color detection that are discussed in this sub chapter. The shape detection algorithm has the advantage that detects all the shapes that are present in the image including shadows. The disadvantage is that depends on the environment because it can detect some shapes that are not wanted or expected. On the other hand, the color detection algorithm has the advantage that detects more accurate the desired obstacles based on a predefined color. Since previously it is needed the color information, the disadvantage is that this information may not be available.

	Advantages	Disadvantages
Shape detection	Detects all kind of objects	May fail due to lighting
Color detection	Detects objects more accurate	Predefined color is needed

Table 4.1: Comparison of detection algorithms

In Tab. 4.1 it is shown the main advantages and disadvantages of the shape and color detection algorithm, respectively. Looking at the given information, one can use on of these algorithms or the combination of them in order to have a more accurate result.

Although the camera has a long range resolution, the minimum resolution was chosen in order to use less image processing time. In Tab. 4.2 it can be seen the image information to be processed.

	Image Data
Width	320 pixels
Height	240 pixels
Size	230400 pixels
Format	RGB (8 bits per color)

Table 4.2: Image information

Figure 4.2: Image taken by the 2D camera

4.1.1 Shape detection algorithm

In order to detect any object by using the shape detection algorithm, first it is necessary to pass the image by a filter and then get a blurred image. The filtering is done by making a convolution between every part of the image and an operator, this operator is the so called kernel matrix. A kernel matrix is essentially a fixed-size array of numerical coefficients along with an "anchor point" in that array, which is typically located at

the center. So this kernel matrix is defined by,

$$kernel = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{4.1}$$

Then with the $filter2D()$ function and the kernel specified in (4.1) and applying the difference between the original image and the filtered image, one can get a blurred image that specifies the objects borders as shown in Fig. 4.3.

Figure 4.3: Blurred image of the obstacle

Then the image is converted into a gray-scale image using $cvtColor()$ function and reducing noise with $blur()$ function.

Figure 4.4: Blurred image in grayscale

As it can be seen in Fig. 4.4, the obstacle is almost clear in comparison from the original figure, then the problem to detect just the edges of this obstacle arises and the Canny algorithm is proposed.The Canny edge detector is an operator developed by John F. Canny in 1986 that uses a multi-stage algorithm for detecting a wide range of edges in images (Mokrzycki & Samko, 2012). The general criteria for edge detection includes:

- Detection of edge with low error rate, which means that the detection should accurately catch as many edges as possible shown in the image.
- The edge point detected from the operator should accurately be localized on the center of the edge.
- A given edge in the image should only be marked once, and where possible, image noise should not create false edges.

In order to satisfy these requirements Canny algorithm uses the calculus of variations, finding the function which optimizes a given functional. So basically the process for the Canny edge detection algorithm has 5 steps:

- Remove the noise by applying a Gaussian filter to smooth the image.
- Find the intensity gradients of the image.
- Get rid of fake response to edge detection by applying non-maximum suppression (edge thinning technique).
- Determine the potential edges by applying double threshold.
- Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges (edge tracking by hysteresis).

The function *Canny*() is the algorithm for the Canny edge detector and one can choose the kernel size, the low and high thresholds. The kernel size is chosen of size 3 and the thresholds values are not chosen manually, instead they are chosen depending on the image characteristics. The better way to do this is by using Otsu's thresholding (Greensted, 2010). In simple words with this method the threshold values are calculated automatically based on an image histogram. Then after all the shape detection algorithm, the final obtained image is as shown in Fig. 4.6.

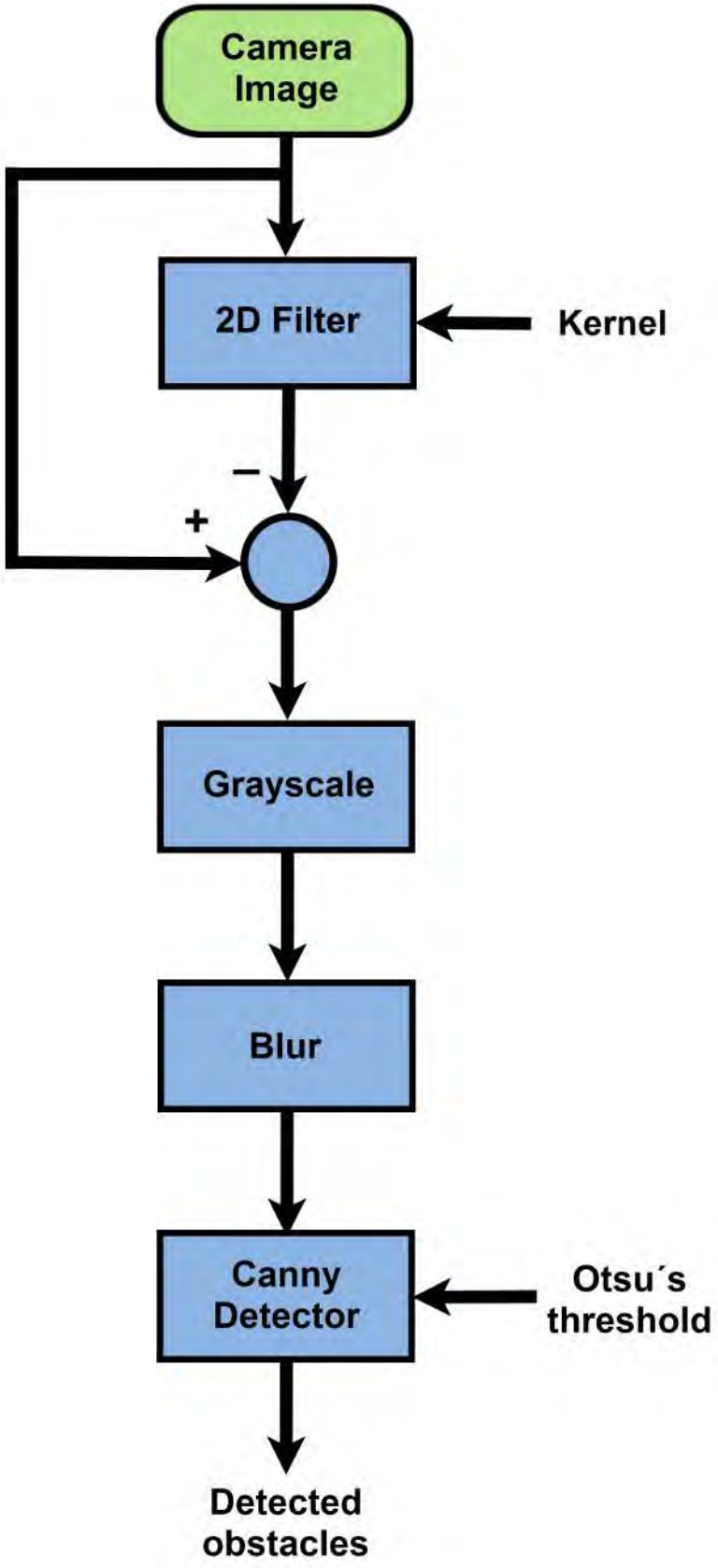


Figure 4.5: Block diagram for shape detection algorithm

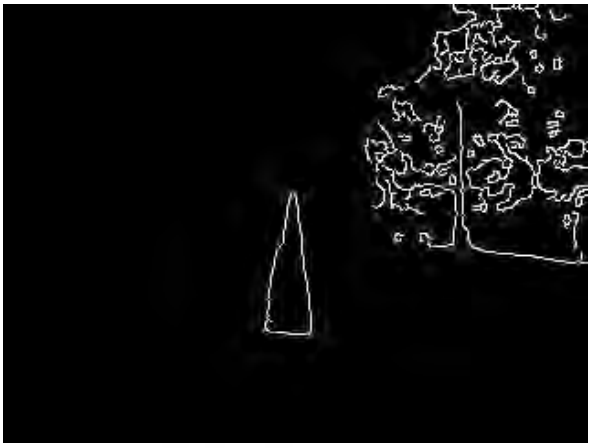


Figure 4.6: Image with detected edges

4.1.2 Color detection algorithm

The color detection algorithm is simpler than the shape detection algorithm. First it is needed to reduce the noise of the original image with the *blur*() function and then using *cvtColor*() function the image is changed from RGB (Red-Green-Blue) to HSV (Hue-Saturation-Value) color as shown in Fig. 4.7.

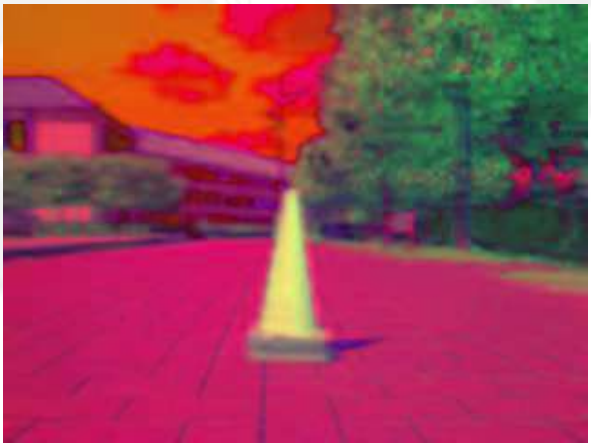


Figure 4.7: Image in HSV values

Then with the function *inRange*() the elements of the array that don't lie between the minimum and maximum HSV threshold values are discarded. As was said before, these threshold values are predefined since the characteristic of the desired objects are known, i.e. if the color of the desired object changes then it is necessary to adapt new minimum and maximum HSV threshold values.

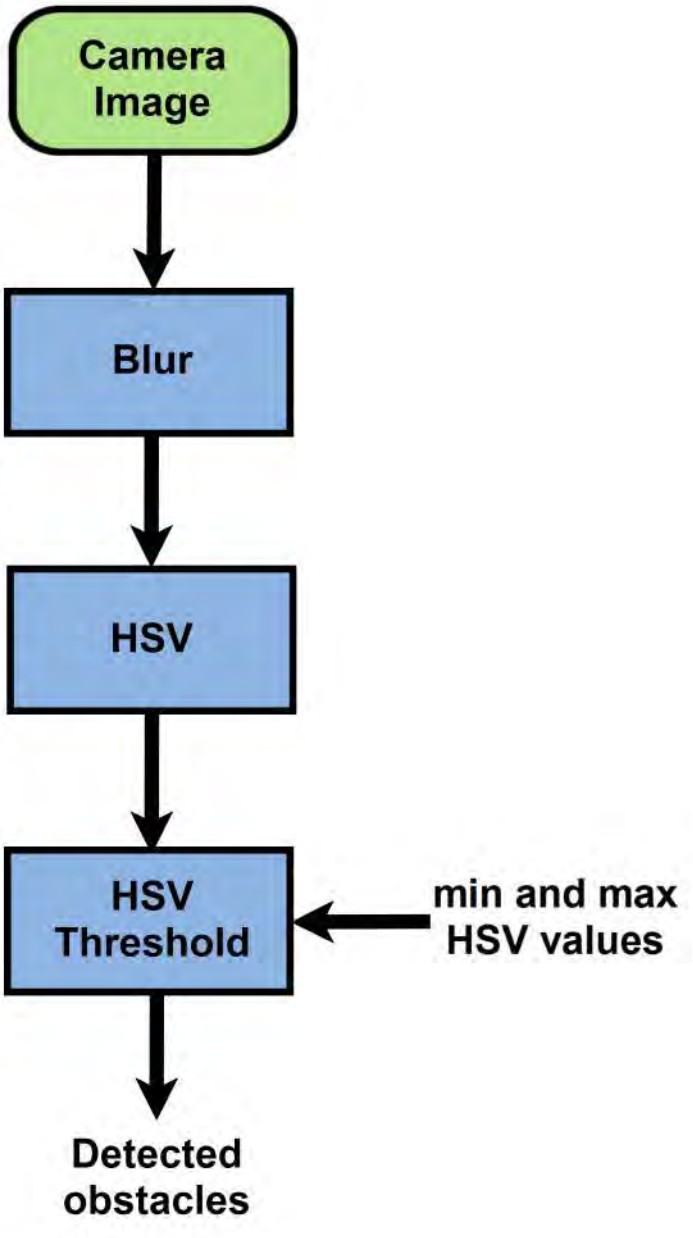


Figure 4.8: Block diagram for color detection algorithm

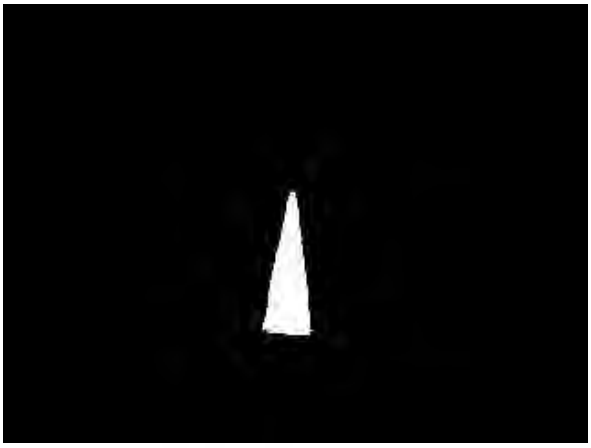


Figure 4.9: Image with detected obstacles

4.2 Cluster classification process

The cluster classification process is the second step for identifying obstacles. Since the final image obtained in the previous process is a black and white image with a lot of detected obstacles and points, one needs to filter it and classify how many obstacles are in there. Therefore the proposed algorithm is the Euclidean distance. In mathematics, the Euclidean distance is the straight line between two points in Euclidean space. In Fig. 4.10 it can be seen an example of an Euclidean space with random points.

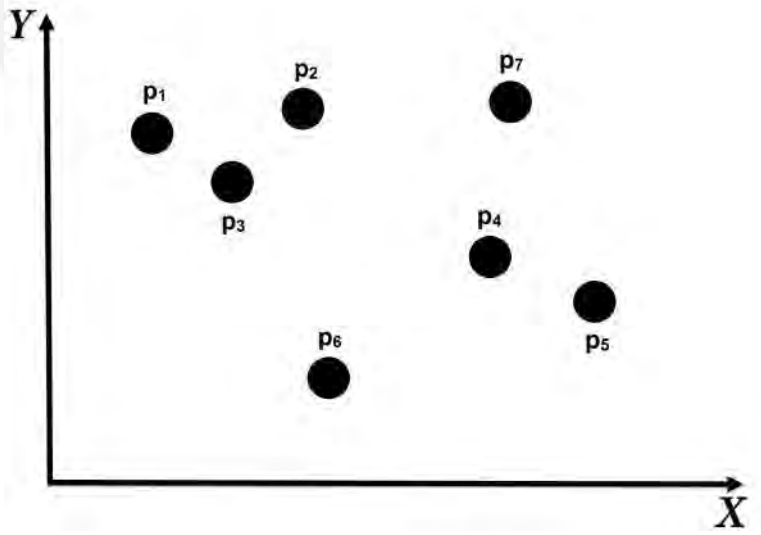


Figure 4.10: Random points in the Euclidean space

Every point has its coordinates in the XY-plane, i.e. $p_i = (x_i, y_i)$ where $i = 1...7$.

Then the distance between two points is given by

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{4.2}$$

It can be seen that (4.2) is equivalent to the Pythagoras theorem. The distances between points will help to determine whether a point belongs to a cluster or not by defining a euclidean distance threshold.

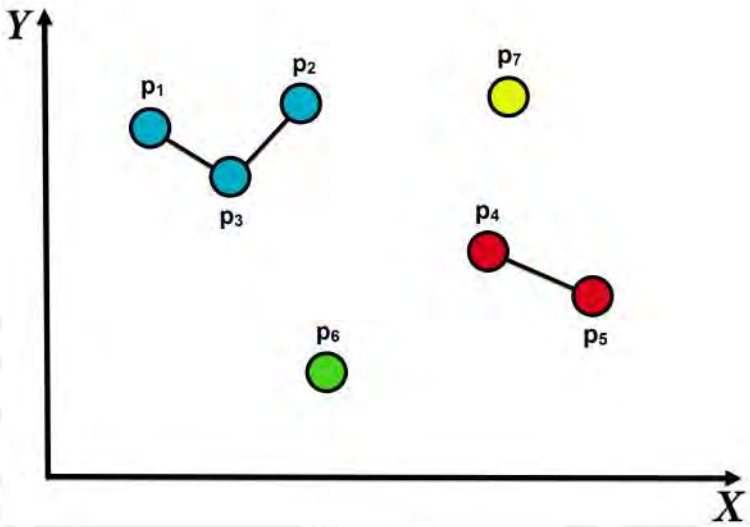


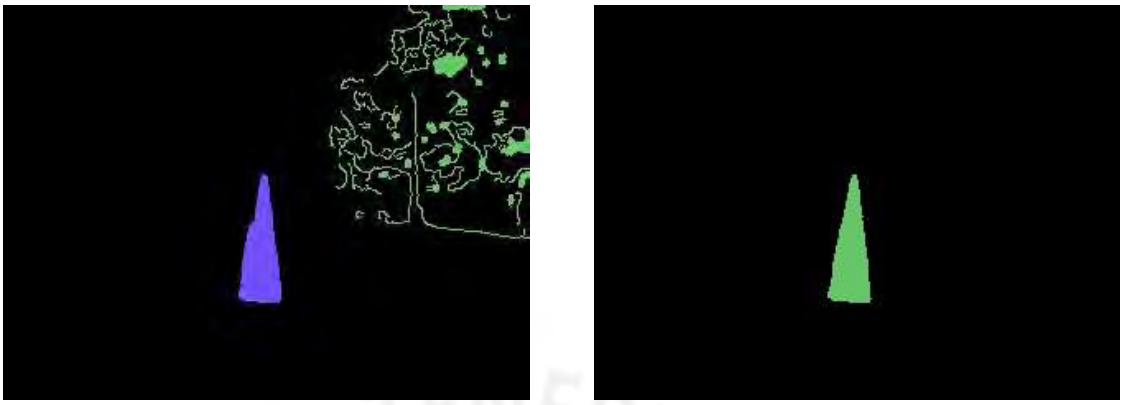
Figure 4.11: Points classified in clusters in the Euclidean space

From Fig. 4.11 it can be observed that there are four clusters, instead of one and are the following,

$$\begin{aligned} C_1 &= \{p_1\,,\, p_2\,,\, p_3\} \\ C_2 &= \{p_4\,,\, p_5\} \\ C_3 &= \{p_6\} \\ C_4 &= \{p_7\} \end{aligned} \tag{4.3}$$

So the solution begins first by getting all non black points from the black-white output image obtained in the previous process. After this the image pixels are partitioned and analyzed with respect to the other ones. Then the euclidean distances between points are calculated and each pixel is assigned to its respective cluster according to the predefined euclidean distance threshold. Experimental values give the best results with a euclidean distance of 15 pixels. This threshold value should not vary so much even if the environment conditions change since the tested environment is very abstract. It is important to clarify that apart from the Euclidean distance threshold, the cluster information depends almost from the previous process, i.e. if the color or

shape detection are not accurate, then the obtained clusters may be not accurate as well.



(a) Using shape detection algorithm (b) Using color detection algorithm

Figure 4.12: Cluster classification

In order to improve the results of the final obtained clusters, two considerations had been taken into account:

- Neglect little clusters based on their sizes. This can be done by choosing a threshold size based of the amount of pixels.
- Neglect all clusters above the horizon line. To be above the horizon line means that the obstacles are too far.

The horizon line is assumed to pass through the center of the image, if the camera is parallel to a flat terrain. Finally with the desired information and since the maximum and minimum pixels of the clusters are known, one can now obtain the obstacle width and height in pixels of every obstacle.

(a) Using shape detection algorithm (b) Using color detection algorithm

Figure 4.13: Detected obstacle

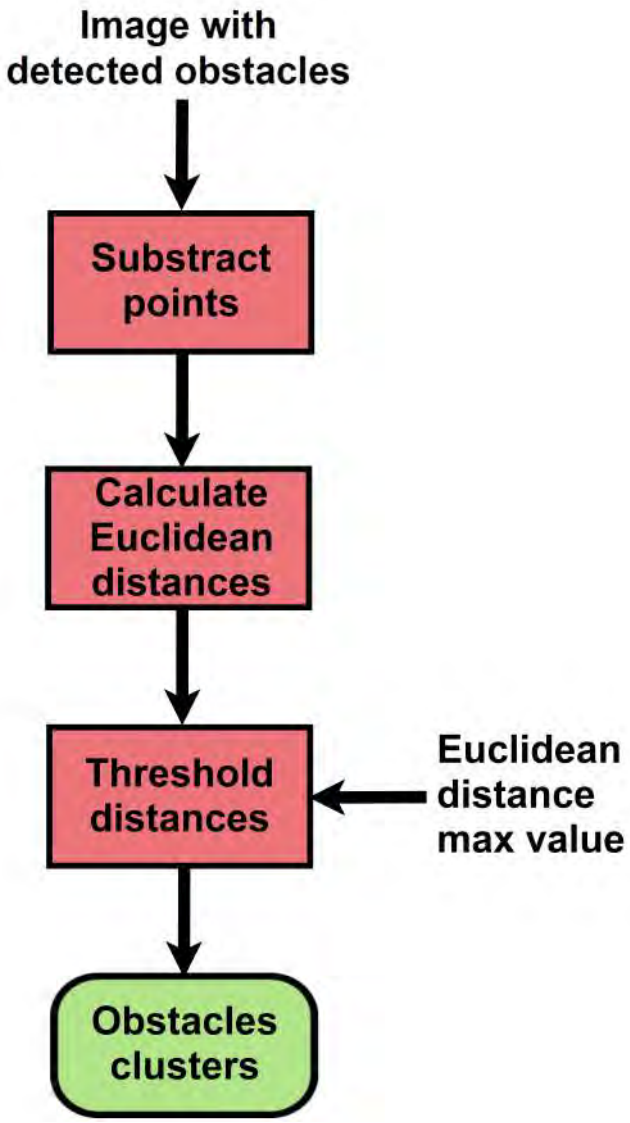


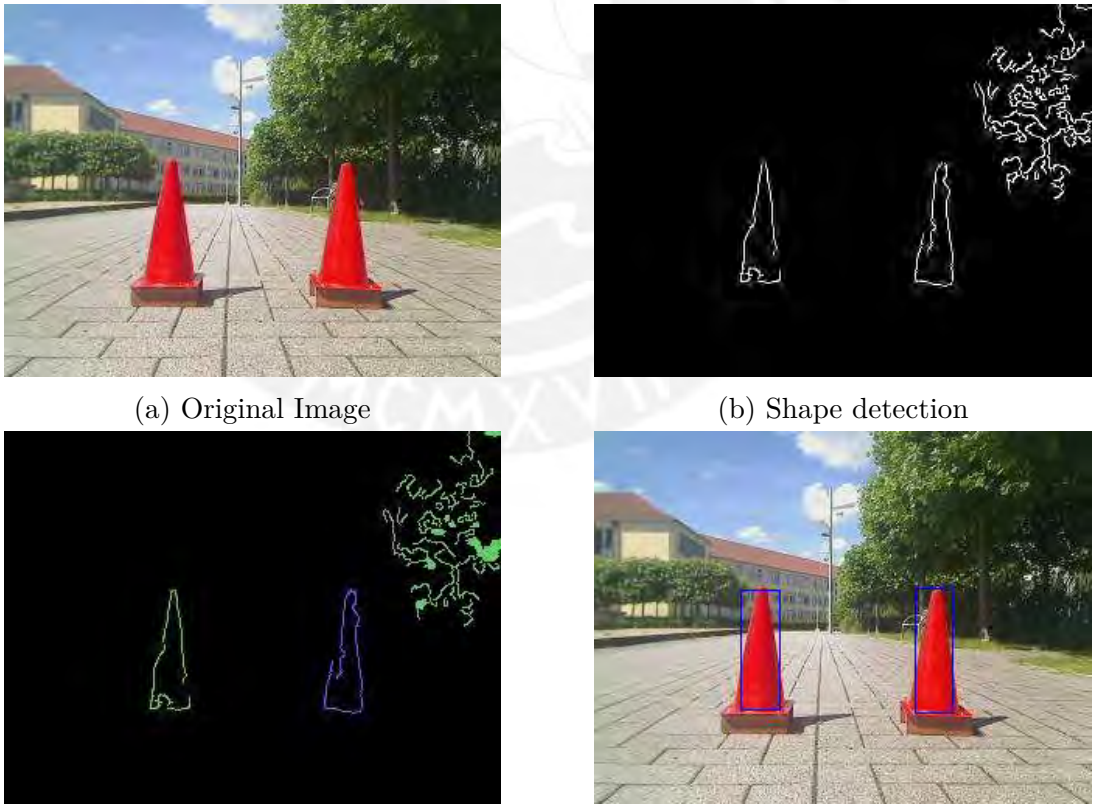
Figure 4.14: Block diagram for obstacle clusters classification

4.3 Results for more than one obstacle

In case 1 the shape detection algorithm is used in scenario A. It can be seen that both obstacles and the tree are detected. Then they are separated into clusters and so the tree cluster is rejected because is above the horizon line. Finally the width and height of the obstacles are calculated.

Case 1		
Algorithm	Shape	
Scenario	A	
	Width	Height
1st obstacle	24 pixels	76 pixels
2nd obstacle	24 pixels	78 pixels

Table 4.3: Considerations for case 1



(a) Original Image (b) Shape detection

(c) Detected clusters (d) Detected obstacles

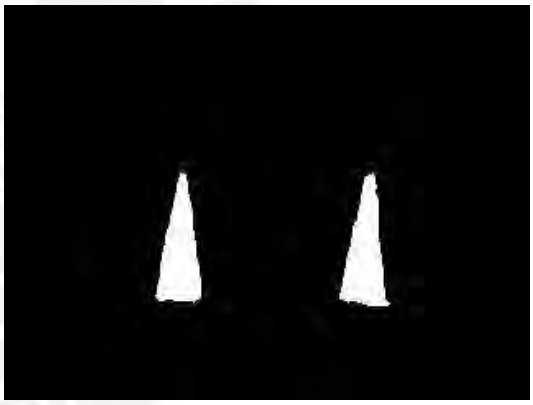
Figure 4.15: Image processing for case 1 (Scenario A)

In case 2 the color detection algorithm is used in scenario A. It can be seen that now just the two obstacles are detected. In the cluster detection process these detected obstacles are separated into clusters. Then the width and height of the obstacles are calculated as well. Finally comparing both the shape and color detection algorithms for this scenario, the results are mostly equal and accurate.

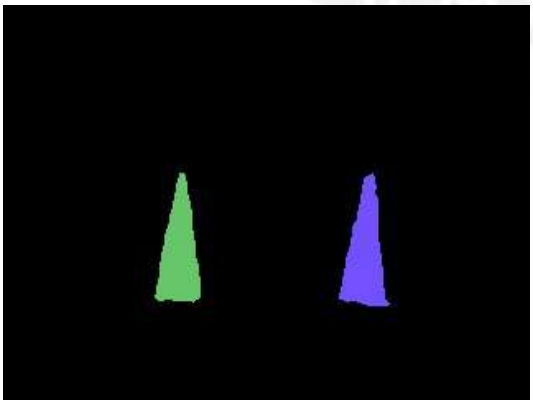
Case 2		
Algorithm	Color	
Scenario	A	
	Width	Height
1st obstacle	27 pixels	78 pixels
2nd obstacle	30 pixels	80 pixels

Table 4.4: Considerations for case 2

(a) Original Image



(b) Color detection



(c) Detected clusters

(d) Detected obstacles

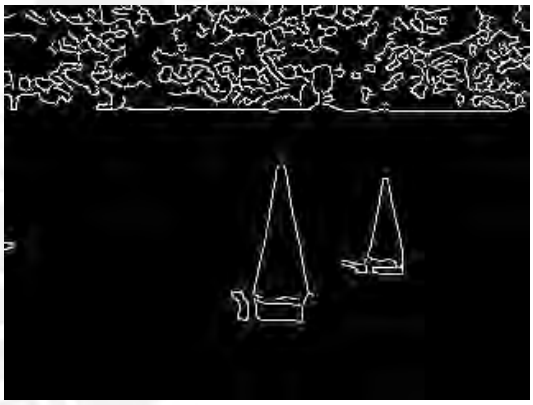
Figure 4.16: Image processing for case 2 (Scenario A)

In case 3 the shape detection algorithm is used in scenario B. It can be seen that both obstacles and the tree are detected. Then they are separated into clusters and so the tree cluster is rejected because is above the horizon line. Is important to notice that the obstacles shadows are detected as well, this may lead to a not so accurate result. Finally the width and height of the obstacles are calculated.

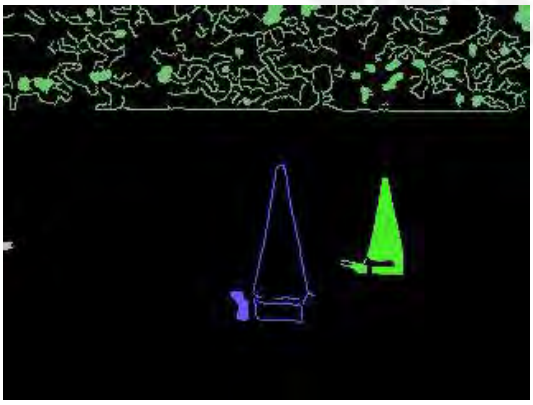
Case 3		
Algorithm	Shape	
Scenario	B	
	Width	Height
1st obstacle	50 pixels	95 pixels
2nd obstacle	37 pixels	58 pixels

Table 4.5: Considerations for case 3

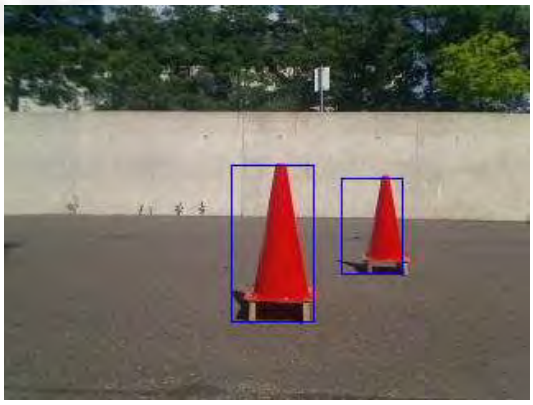
(a) Original Image



(b) Shape detection



(c) Detected clusters



(d) Detected obstacles

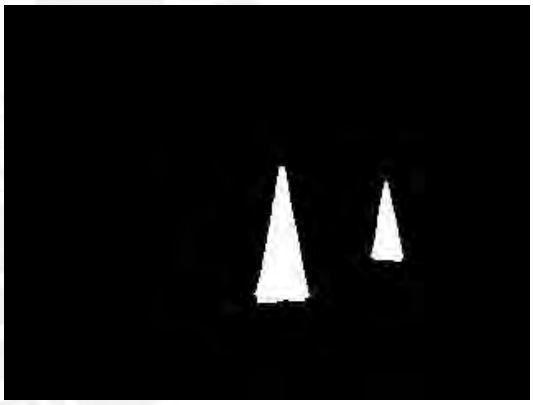
Figure 4.17: Image processing for case 3 (Scenario B)

In case 4 the color detection algorithm is used in scenario B. It can be seen that just the two obstacles are detected. In the cluster detection process these detected obstacles are separated into clusters. Then the width and height of the obstacles are calculated as well. Finally comparing both the shape and color detection algorithms for this scenario, using color detection is more accurate due to the obstacle shadows.

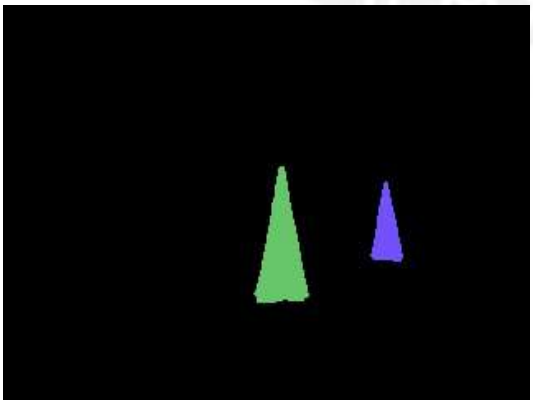
Case 4		
Algorithm	Color	
Scenario	B	
	Width	Height
1st obstacle	33 pixels	82 pixels
2nd obstacle	19 pixels	48 pixels

Table 4.6: Considerations for case 4

(a) Original Image



(b) Color detection



(c) Detected clusters

(d) Detected obstacles

Figure 4.18: Image processing for case 4 (Scenario B)

5 Sensor fusion for obstacle determination

In previous works like (Belgradskaia, 2016), (Drozdova, 2015), (Müller, 2013), (Thieme, 2014) and (Correa, 2016), the mobile robot SUMMIT used just the laser scanner for obstacle identification. Using laser scanner is the easiest way to know that something is in front of the mobile robot but it doesn't give a deep information about the environment surrounding it. Because of this, sensor fusion arises and then one can get a better information about the detected obstacle as mentioned in (Mahajan et al., 2013). Sensor fusion is the combination of data derived from different sources or sensors such that the resulting information is more accurate (less uncertain),in comparison, if these sources or sensors were individually used (Mitchell, 2007). The data sources may not be from identical sensors, so one can distinguish two kind of fusions:

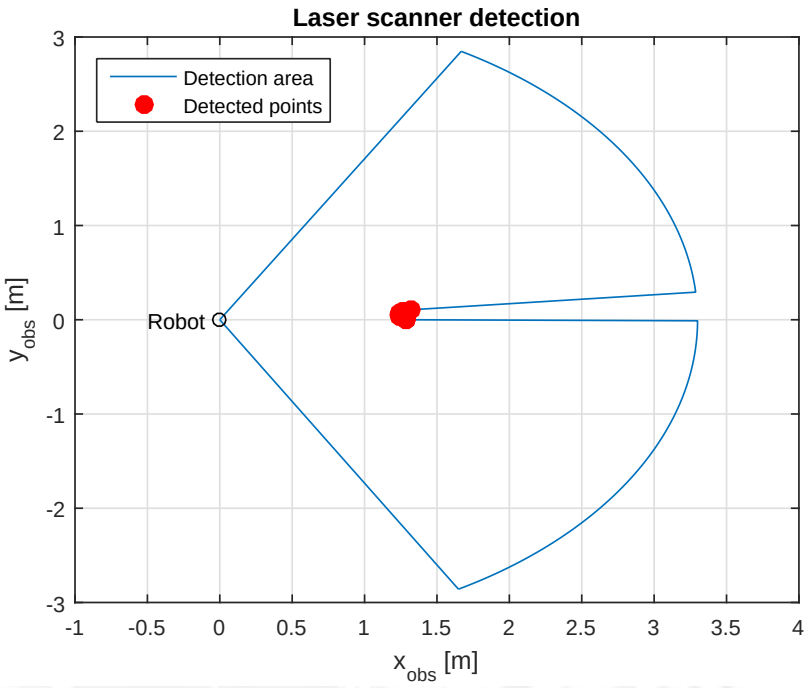
- Direct fusion: uses a set of heterogeneous or homogeneous sensors, soft sensors and history values of sensor data.
- Indirect fusion: uses information sources like a priori knowledge about the environment and human input.

In this chapter the main task is to determine the obstacle position and dimensions via direct fusion based on the information of the 2D camera and the laser scanner. The laser scanner gives the obstacle position and width information, while the 2D camera gives the obstacle width and height information. Since the camera is a mono camera, the depth information is no available but there are some methods and approaches that help to estimate the distance from the camera to the obstacle.

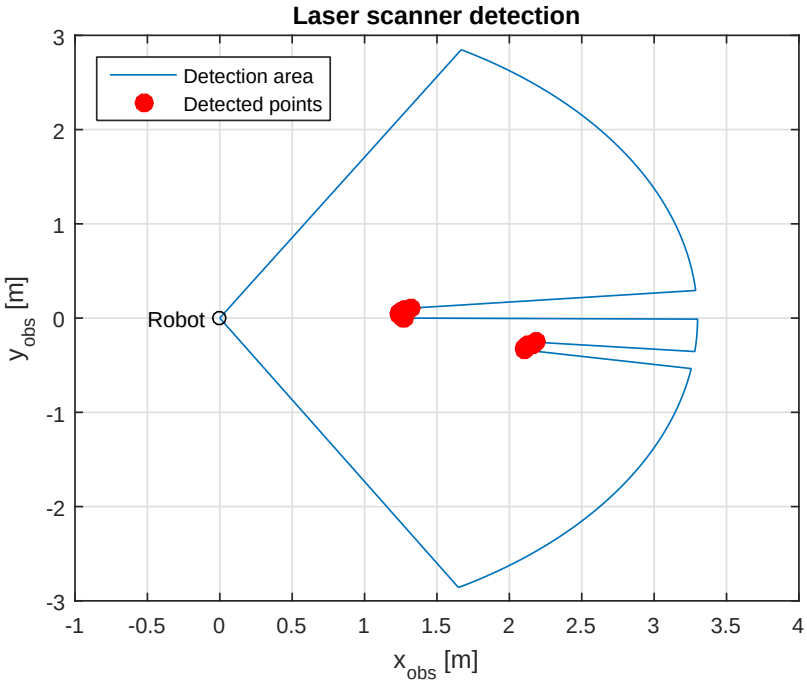
5.1 Data acquisition by the laser scanner

As it was described in Fig. 2.5 from chapter 2, the selected detection range is 3.3 m and the scan angle is 120°. So the laser scanner detects all the obstacles that are inside

this area. Fig. 5.1a and 5.1b show the laser scanner detected points in the predefined area.



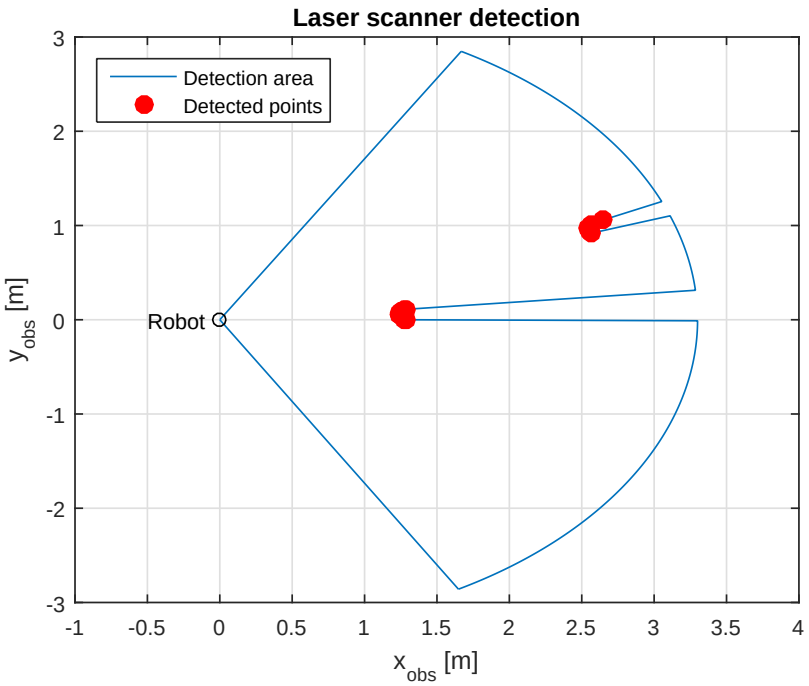
(a) For one obstacle



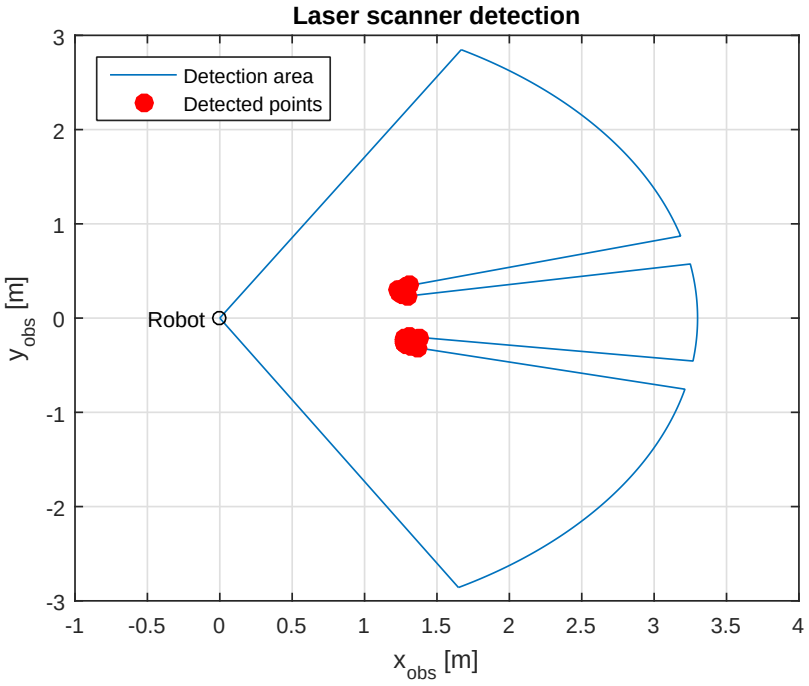
(b) For two obstacles

Figure 5.1: Data acquired by the laser scanner

If the laser scanner detects more than one obstacle, depending on the distance between them one can take two obstacles as one.



(a) For a distance between obstacles > 1.2 m



(b) For a distance between obstacles < 1.2 m

Figure 5.2: Data acquired by the laser scanner

The distance that defines if two obstacles are considered as one is the distance in the XY-plane between the most right point of the first obstacle and the most left point of the second obstacle. In Fig. 5.2a it can be seen that the distance between both obstacles is greater than 1.2 m, i.e. it is considered just the first obstacle. On the other hand, in Fig. 5.2b it can be seen that the distance between both obstacles now is less than 1.2 m, i.e. both obstacles are considered as one.

The most left point coordinate is defined as $(x_{obs,1}, y_{obs,1})$, while the most right point coordinate is defined as $(x_{obs,f}, y_{obs,f})$. Then the relative position of the obstacle in the XY-plane is obtained as follows,

$$x_{rel,las} = \frac{x_{obs,1} + x_{obs,f}}{2}$$
$$y_{rel,las} = \frac{y_{obs,1} + y_{obs,f}}{2} \tag{5.1}$$

5.2 Data acquisition by the 2D camera

As it was described in chapter 4, the final information that is obtained from the image after the detection and cluster classification processes is the obstacle width and height in pixels. So there is the necessity to transform this obstacle dimensions from pixels to a real measure, in this case meters.

Therefore first the pinhole camera model is explained in order to understand the later estimations of distance and obstacle dimensions that can be seen in (Han et al., 2016) and (K.-Y. Park & Hwang, 2014). Then the estimation of the distance, height and width are explained as well. Finally some results for different images are shown.

5.2.1 Pinhole camera model

The pinhole camera model describes the mathematical relationship between the coordinates of a point and its projection onto the image plane of an ideal pinhole camera. Since the pinhole camera has no lenses and just a tiny aperture, the model does not include geometric distortions or blurring. Its validity depends on the camera quality.

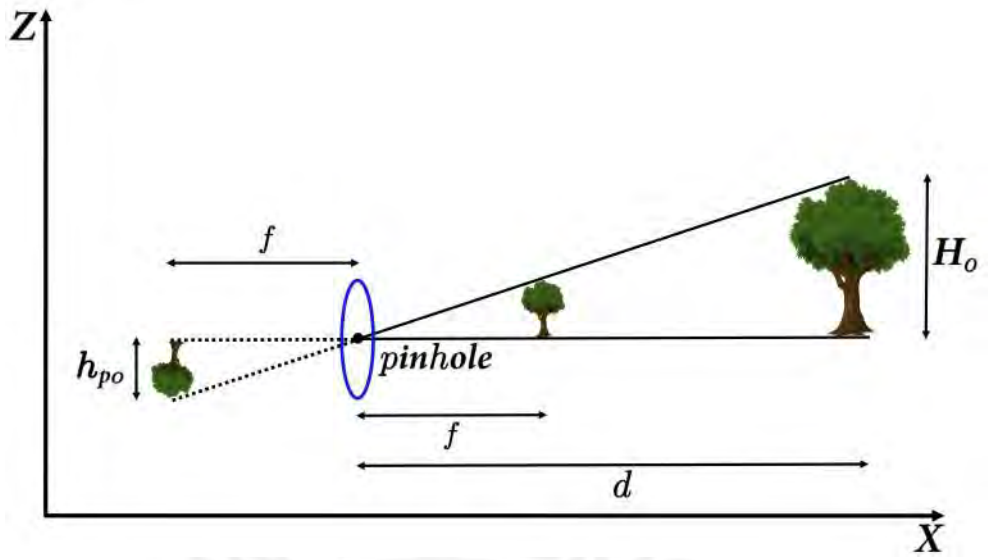


Figure 5.3: Pinhole camera model

From Fig. 5.3, a pinhole camera model equation can be obtained,

$$d = f \frac{H_o}{h_{po}} \tag{5.2}$$

where d is the object distance in the X-direction, f is the focal length of the camera, H_o is the object height and h_{po} is the projected object height.

5.2.2 Estimation of the distance to the obstacle

In order to estimate the distance between the obstacle and the mobile robot by using the pinhole camera model, two methods can be described.

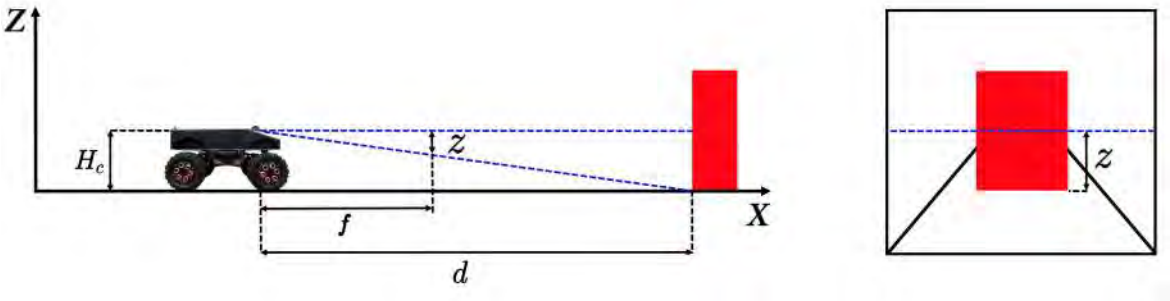


Figure 5.4: Distance estimation method using obstacle image height

The first method uses the obstacle image height and it is described in Fig. 5.4. Thereby from the equation of the pinhole camera model, an equation of the method using the

obstacle image height can be derived and it is shown in (5.3).

$$d = f\,\frac{Hc}{z} \tag{5.3}$$

where Hc is the camera height and z is the obstacle image height between the horizon line and the bottom edge of the obstacle. As said before, since the road is assumed to be plain the optical axis of the camera is parallel to the road surface and the horizon line is assumed to pass through the center of the image.

The second method uses the obstacle image width and it is described in Fig. 5.5. Thereby from the equation of the pinhole camera model, an equation of the method using a obstacle image width can be derived and it is shown in (5.4)

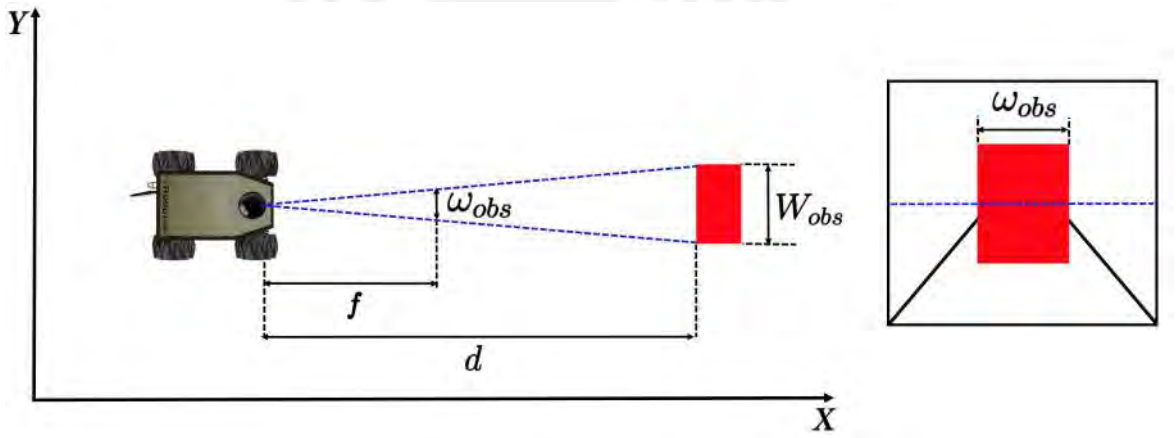


Figure 5.5: Distance estimation method using obstacle image width

$$d = f\,\frac{W_{obs}}{\omega_{obs}} \tag{5.4}$$

where W_{obs} is the obstacle physical width and ω_{obs} is the obstacle image width.

In conclusion with one of the previously presented methods the distance between the mobile robot and the obstacle can be obtained, but by now from (5.3) and (5.4) the only known values are Hc and ω_{obs}, respectively.

5.2.3 Estimation of obstacle width and height

In order to estimate the obstacle width and height a relationship between the previously methods is described by matching (5.3) and (5.4),

$$W_{obs} = \omega_{obs} \frac{Hc}{z} \tag{5.5}$$

where W_{obs} depends on the ω_{obs}, Hc, and z.

From (5.5), the known values are ω_{obs} and Hc, while the unknown value is z. Then this unknown value z can be represented as,

$$z = z_{bottom} - z_{horizon} \tag{5.6}$$

where z_{bottom} is the bottom edge of the detected obstacle and $z_{horizon}$ is the horizon line.

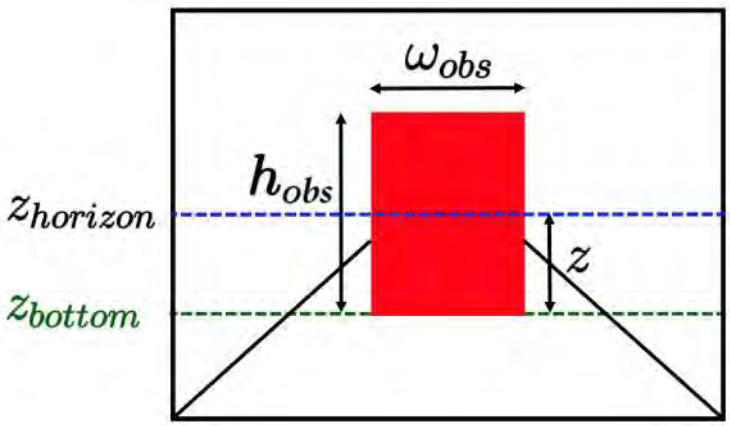


Figure 5.6: Image parameters to find obstacle width

Since both z_bottom and $z_horizon$ are known values then replacing (5.6) in (5.5) and using the known values Hc and ω_{obs} the obstacle width W_{obs} can be obtained. Finally, using a cross relation H_{obs} can be obtained as well.

$$H_{obs} = h_{obs} \frac{W_{obs}}{\omega_{obs}} \tag{5.7}$$

where h_{obs} is the obstacle image height and H_{obs} the obstacle physical height.

5.2.4 Results for estimation of obstacle width and height

As explained previously, the obstacle width and height can be determined from (5.5) and (5.7), respectively. In case 1, there is just one obstacle, so the width and height are calculated and one can see that the final cluster is the obstacle itself.

Case 1	Width	Height
Obstacle real measure	19.6 cm	37.4 cm
1st obstacle estimation	13.8667 cm	41.0667 cm
2nd obstacle estimation	0 cm	0 cm
Total cluster estimation	13.8667 cm	41.0667 cm

Table 5.1: Obstacle dimensions for case 1

(a) Original Image

(b) Detected clusters

(c) Final detected obstacle

Figure 5.7: Obstacle determination for case 1

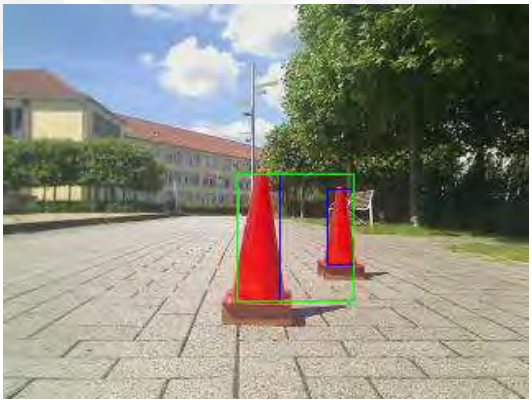
In case 2 there are two obstacles with a distance between them less than 1.2 m, i.e. that both obstacles are considered as one. The cluster width is calculated with the most left point of the first obstacle and the most right point of the second obstacle. Finally the cluster height is calculated from the nearest obstacle.

Case 2	Width	Height
Obstacle real measure	19.6 cm	37.4 cm
1st obstacle estimation	13.8667 cm	41.0667 cm
2nd obstacle estimation	13.4737 cm	38.7368 cm
Total cluster estimation	37.8667 cm	41.0667 cm

Table 5.2: Obstacle dimensions for case 2

(a) Original Image

(b) Detected clusters



(c) Final detected obstacle

Figure 5.8: Obstacle determination for case 2

In case 3 there are two obstacles as well, but now the distance between them is greater than 1.2 m, this leads to consider just the nearest obstacle as the final cluster, i.e. that the width and height of the final cluster are the dimensions of the nearest obstacle.

Case 3	Width	Height
Obstacle real measure	19.6 cm	37.4 cm
1st obstacle estimation	13.6393 cm	40.918 cm
2nd obstacle estimation	12.1379 cm	40.8276 cm
Total cluster estimation	13.6393 cm	40.918 cm

Table 5.3: Obstacle dimensions for case 3

(a) Original Image

(b) Detected clusters

(c) Final detected obstacle

Figure 5.9: Obstacle determination for case 3

In case 4 there are two obstacles with a distance between them less than 1.2 m, i.e. that both obstacles are considered as one. The cluster width is calculated with the most left point of the first obstacle and the most right point of the second obstacle. Finally the cluster height is calculated from the nearest obstacle.

Case 4	Width	Height
Obstacle real measure	19.6 cm	37.4 cm
1st obstacle estimation	14.4 cm	41.6 cm
2nd obstacle estimation	15.4839 cm	41.2903 cm
Total cluster estimation	73.2903 cm	41.2903 cm

Table 5.4: Obstacle dimensions for case 4

(a) Original Image

(b) Detected clusters

(c) Final detected obstacle

Figure 5.10: Obstacle determination for case 4

5.2.5 Results for estimation of distance to the obstacle

The distance between the camera and the obstacle can be estimated with the pinhole camera model equation presented previously, i.e. with the obstacle width or height information, (5.4) or (5.3), respectively. Nevertheless the focal length value in pixels is necessary to calculate this distance. Thus this unknown value can be found experimentally by making different tests using the following relationship,

$$f = d\,\frac{\omega_{obs}}{W_{obs}} \tag{5.8}$$

where d is the measured distance between the camera and the obstacle, ω_{obs} is the obstacle width in pixels and W_{obs} is the real obstacle width.

It can be seen from Tab. 5.5 different values for the focal length for each measured distance. Then the experimental value for the focal length can be defined by the mean value as follows,

$$f = \frac{1}{N}\sum_{i=1}^{N} f_i = 259.4811 \tag{5.9}$$

where N is the number of taken data. Now with the value of the focal length is possible to estimate the distance from the camera to the obstacle using (5.4). The previously cases are presented as well for distance estimation from the camera to the final cluster.

In case 1 there is just one obstacle, i.e. that the final cluster is the obstacle itself.

In case 2 there are two obstacles with distance between them less than 1.2 m. The final cluster is a huge obstacle that starts from the most left point of the first obstacle to the most right point of the second obstacle.

In case 3 there are two obstacles with a distance between them more than 1.2 m. The final cluster is the nearest obstacle.

In case 4 there are two obstacles with a distance between them less than 1.2 m. The final cluster is a huge obstacle that starts from the most left point of the first obstacle to the most right point of the second obstacle.

i	d (cm)	ω_{obs} (pixels)	W_{obs} (cm)	f (pixels)
1	70	78	20.9748	260.3123
2	80	63	19.7647	255.0001
3	90	39	13.7143	255.9372
4	100	39	15.2195	256.2502
5	110	34	14.7	254.4217
6	120	31	14.5882	255.0006
7	130	28	14.4516	251.8752
8	140	25	13.7931	253.75
9	150	24	14.2222	253.1253
10	160	21	13.1765	254.9994
11	170	18	12	255
12	180	17	12.0889	253.1247
13	190	15	11.1628	255.3122
14	200	13	10.1463	256.251
15	210	12	9.8461	255.9376
16	220	12	10.1053	261.249
17	230	13	11.5556	258.749
18	240	11	10.0571	262.5011
19	250	10	9.4117	265.6251
20	260	9	8.7272	268.125
21	270	9	9	270
22	280	9	9.2903	271.25
23	290	8	8.5333	271.8751
24	300	8	8.8275	271.8748

Table 5.5: Experimental data for focal length estimation

Case 1	W_{obs}	ω_{obs}	d
Obstacle real measure	19.6 cm	-	124.29 cm
Obstacle estimation	13.8667 cm	26 pixels	138.389 cm

Table 5.6: Distance calculation for case 1

Figure 5.11: Final detected obstacle for case 1

Case 2	W_{obs}	ω_{obs}	d
1st obstacle real measure	19.6 cm	-	124.29 cm
2nd obstacle real measure	19.6 cm	-	210.49 cm
1st obstacle estimation	13.8667 cm	26 pixels	138.389 cm
2nd obstacle estimation	13.4737 cm	16 pixels	218.509 cm

Table 5.7: Distance calculation for case 2

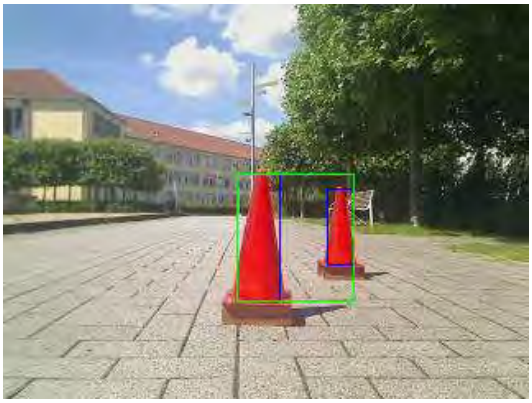


Figure 5.12: Final detected obstacle for case 2

Case 3	W_{obs}	ω_{obs}	d
1st obstacle real measure	19.6 cm	-	124.11 cm
1st obstacle estimation	13.6393 cm	26 pixels	136.121 cm

Table 5.8: Distance calculation for case 3

Figure 5.13: Final detected obstacle for case 3

Case 4	W_{obs}	ω_{obs}	d
1st obstacle real measure	19.6 cm	-	123.37 cm
2nd obstacle real measure	19.6 cm	-	127.53 cm
1st obstacle estimation	14.4 cm	27 pixels	138.389 cm
2nd obstacle estimation	15.4839 cm	30 pixels	133.925 cm

Table 5.9: Distance calculation for case 4

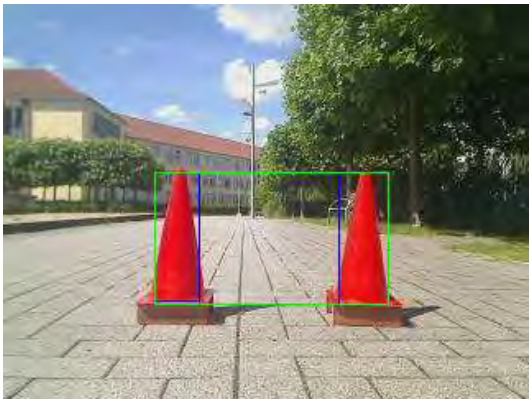


Figure 5.14: Final detected obstacle for case 4

5.3 Data fusion of the laser scanner and the 2D camera

Up to here both the laser scanner and the 2D camera have acquired independent data. First with the laser scanner it is possible to estimate the obstacle position and width; and then with the 2D camera it is possible to estimate the obstacle width and height.

Since the main task of these sensors is to detect obstacles, the problem to avoiding them arises. Then one method for obstacle avoiding is by creating an ellipse around it, i.e. means that some ellipse parameters must be defined,

- Inclination angle.
- Position in the XY-plane.
- Semi-major axis and semi-minor axes.

5.3.1 Ellipse inclination angle

The first parameter is the ellipse inclination angle and it just depends on the laser scanner, since the 2D camera does not give depth information. So it is defined as follows,

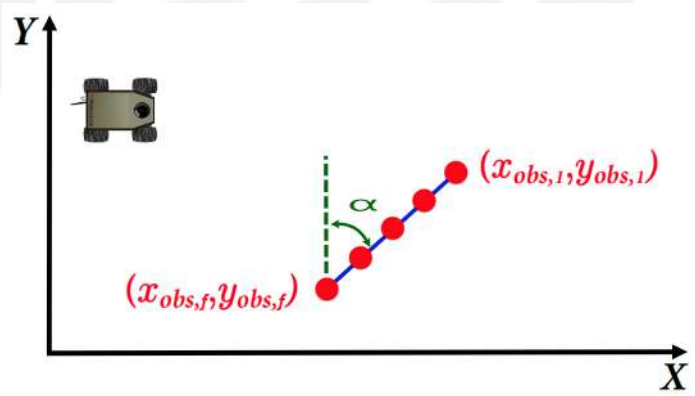


Figure 5.15: Inclination angle defined by the laser scanner detected points

where α is the ellipse inclination angle between the first and last detected points. Then the equation is defined as

$$\alpha = \arctan\left(\frac{x_{obs,1} - x_{obs,f}}{y_{obs,1} - y_{obs,f}}\right) \tag{5.10}$$

5.3.2 Ellipse absolute position in the XY-plane

The second parameters are the ellipse position in the XY-plane that depends on the laser scanner and the 2D camera data. It is important to clarify that the information given by the 2D camera is only used for the distance estimation in the X-direction. So the obstacle relative position is defined by,

$$\begin{aligned} x_{rel} &= \frac{x_{rel,las} + x_{rel,cam}}{2} \\ y_{rel} &= y_{rel,las} \end{aligned} \tag{5.11}$$

where $(x_{rel,las}, y_{rel,las})$ are the position coordinates in the XY-plane given by the laser scanner, and $x_{rel,cam}$ is the distance in the X-direction given by the camera. Then the obstacle absolute position depends on its relative position and mobile robot absolute position. Fig. 5.16 shows the obstacle and robot position in the XY-plane.

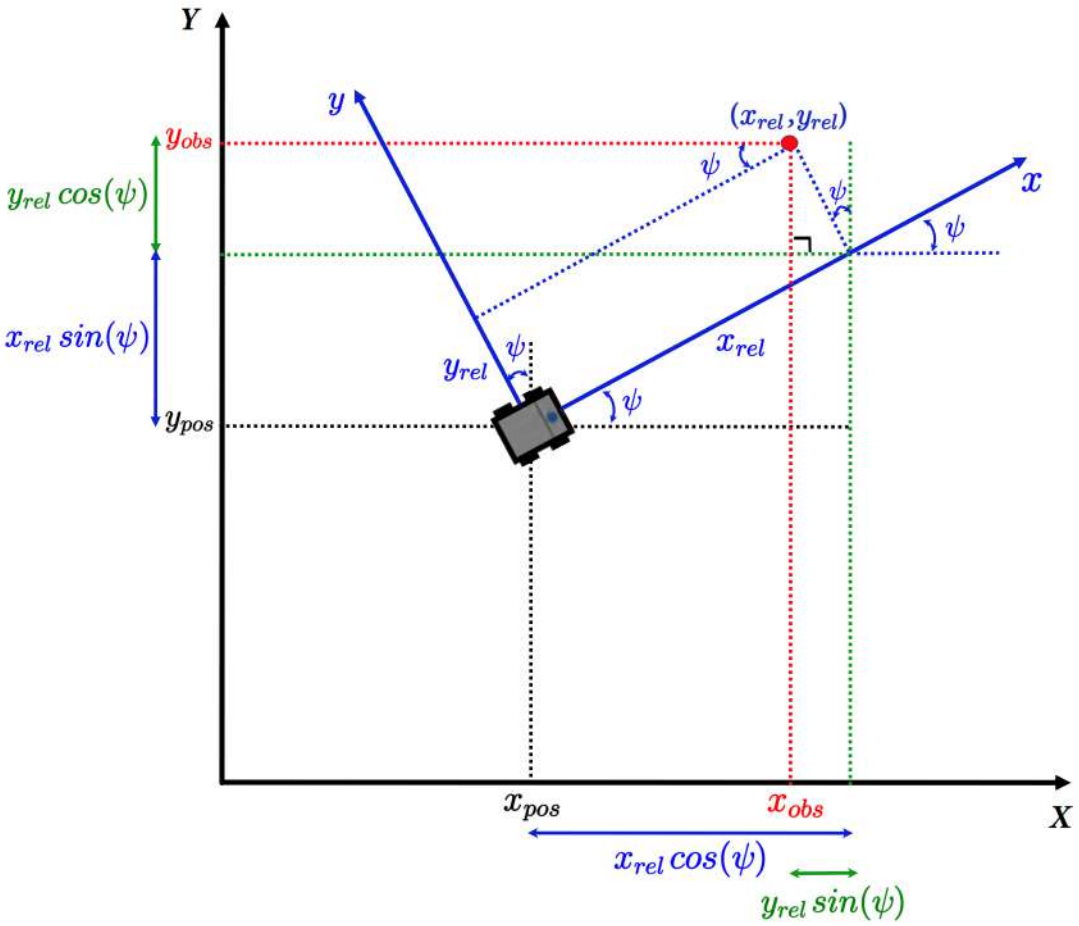


Figure 5.16: Obstacle position in absolute coordinates in the XY-plane

As a result, the absolute obstacle position coordinates in the XY-plane are defined as,

$$\begin{aligned} x_{obs} &= x_{pos} + x_{rel}\ \cos(\psi) - y_{rel}\ \sin(\psi) \\ y_{obs} &= y_{pos} + x_{rel}\ \sin(\psi) + y_{rel}\ \cos(\psi) \end{aligned} \tag{5.12}$$

where x_{pos} and y_{pos} are the robot position, x_{rel} and y_{rel} the relative obstacle position calculated previously and ψ is the yaw angle.

5.3.3 Ellipse axes

The third parameters are the ellipse semi-major and semi-minor axes depending directly on the obstacle width information given by the laser scanner and the 2D camera data.

As described before, the obstacle width using the 2D camera is determined by (5.5) for each obstacle. Since the mobile robot can detect more than one obstacle on its way inside the detection range, a solution is presented.

In the first case there is just one obstacle on the image, so this means that the obstacle width is W_{obs} as one can see in Fig. 5.17.

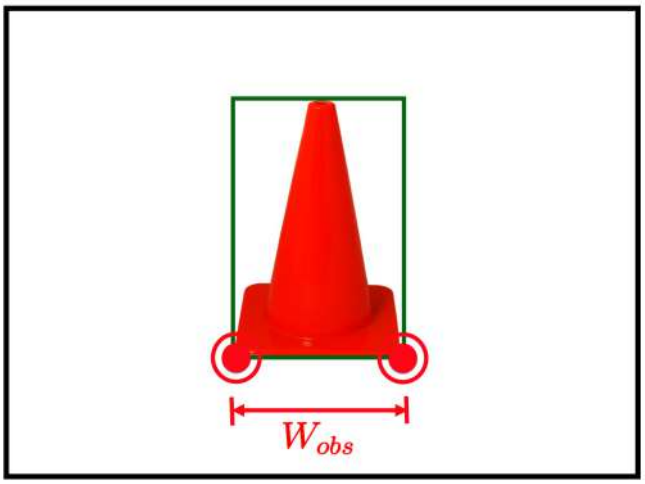
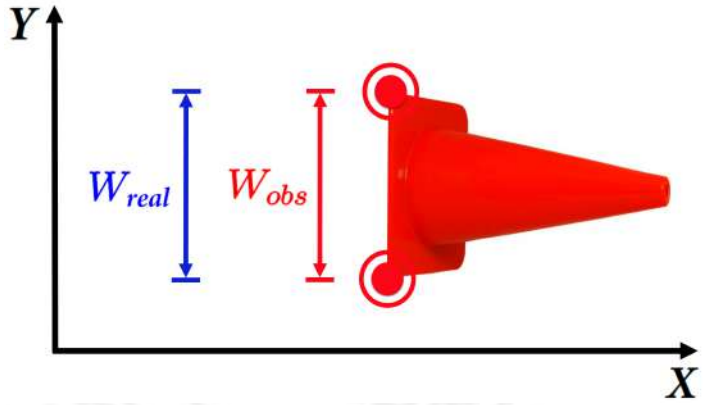


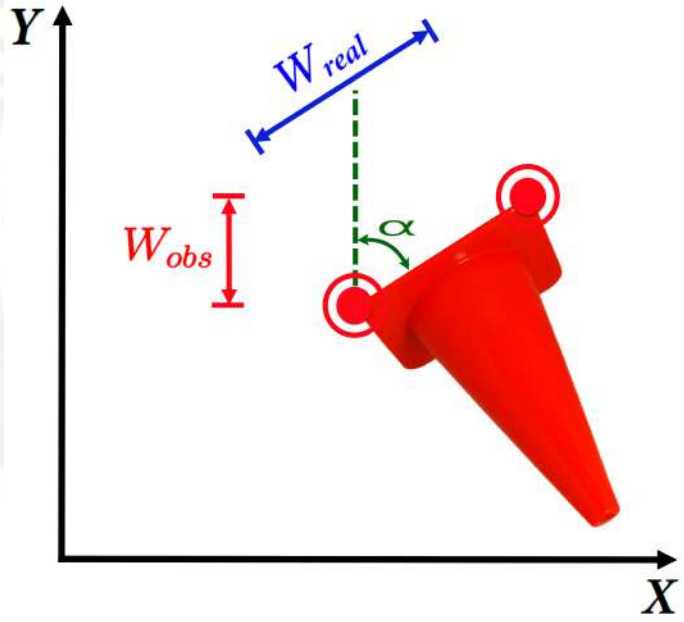
Figure 5.17: One obstacle detected by the camera

As said before, since the camera doesn't give depth information, then the obstacle width W_{obs} may not be the real width and this is because of the inclination angle α.

Therefore two situations can be seen in Fig. 5.18a and 5.18b related to the obstacle real width.



(a) For $\alpha = 0°$



(b) For $\alpha \neq 0°$

Figure 5.18: Obstacle real width (one obstacle)

Thus the relation between W_{obs} and W_{real} is,

$$W_{real} = \frac{W_{obs}}{\cos(\alpha)} \tag{5.13}$$

where W_{real} is the obstacle real width and depends on the obstacle width and the inclination angle. If there is no inclination angle ($\alpha = 0°$) then $W_{real} = W_{obs}$.

In the second case there is more than one obstacle on the image, so this means that the obstacle width now depends on each W_{obs} of each obstacle as one can see in Fig. 5.19.

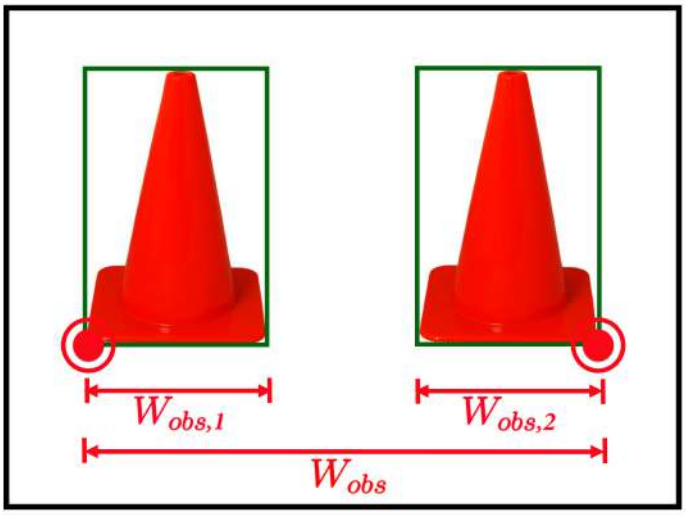
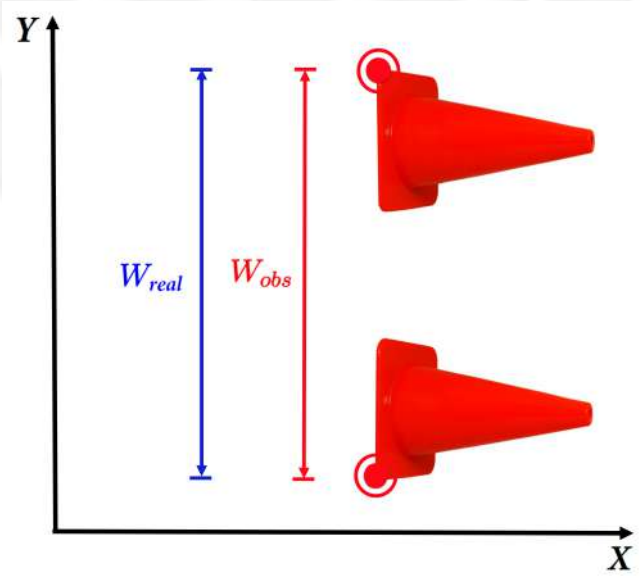
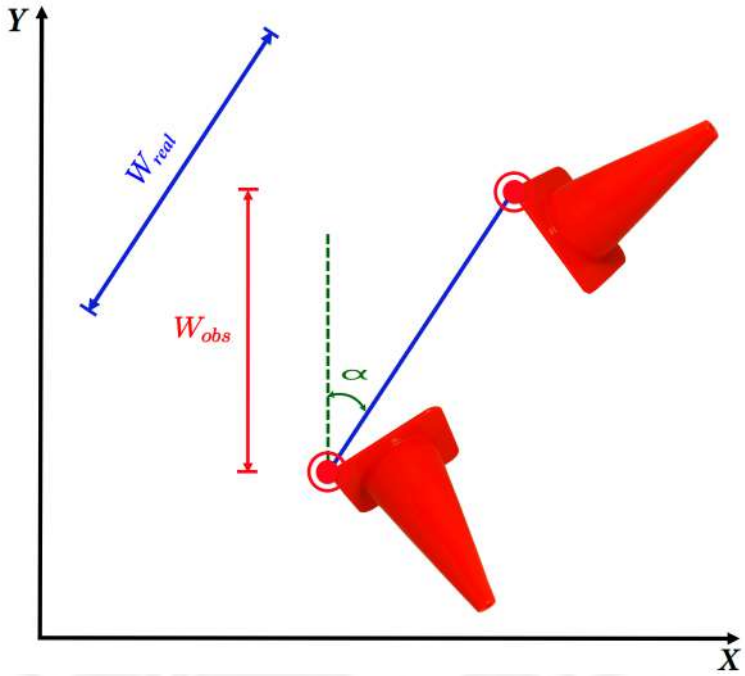


Figure 5.19: Two obstacles detected by the camera

In the same way as before, the obstacle width W_{obs} may not be the real width and this is because of the inclination angle α. Therefore two situations can be seen in Fig. 5.20a and 5.20b related to the obstacle real width.



(a) For $\alpha = 0°$



(b) For $\alpha \neq 0°$

Figure 5.20: Obstacle real width (two obstacles)

Thus the relation between W_{obs} and W_{real} is the same as (5.13). Now the ellipse semi-minor axis can be determined by the laser scanner and the 2D camera. One semi-minor axis value is calculated for each sensor in order to get an accurate solution. Eq. (5.14) and (5.15) show these values.

$$b_{las} = |y_{obs,1} - y_{obs,f}| + 0.6 \tag{5.14}$$

$$b_{cam} = W_{real} + 0.6 \tag{5.15}$$

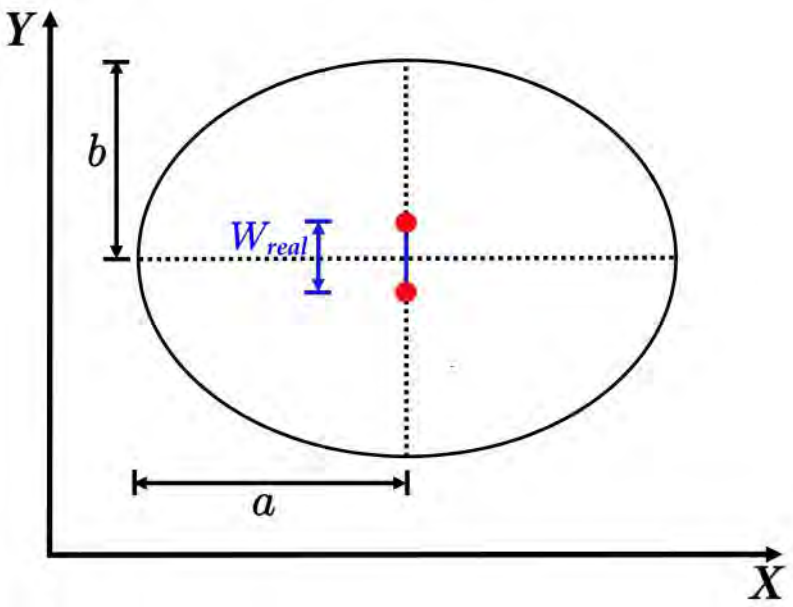
where b_{las} is the first calculated semi-minor axis that depends on the first and last detected points of the laser scanner and b_{cam} is the second calculated semi-minor axis that depends on the obstacle real width W_{real} of the 2D camera. The value of 0.6 was chosen to ensure that the ellipse is sufficiently large even with small-sized obstacles. With both semi-minor axes, one can combine these information in order to get a better result using the mean value. On the other hand, the semi-major axis is also calculated based on the semi-minor axis.

$$b = \frac{b_{las} + b_{cam}}{2} \tag{5.16}$$

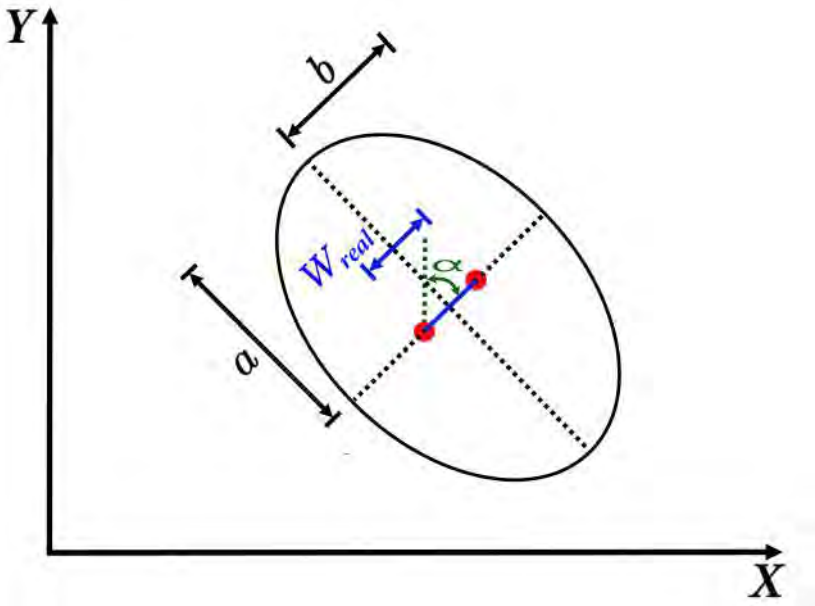
$$a = 1.3\, b \tag{5.17}$$

where b is the ellipse semi-minor axis and a is the ellipse semi-major axis. A restriction must be applied to the ellipse axis to limit the size of the ellipse, then

$$a = \begin{cases} 1.3\,b, & \text{if } b \geq 1.2 \\ 1.2, & \text{otherwise} \end{cases} \tag{5.18}$$



(a) For $\alpha = 0°$



(b) For $\alpha \neq 0°$

Figure 5.21: Ellipse representation

5.4 Results of data fusion

In case 1 there is just one obstacle and it is surrounded by an ellipse defined by the parameters b and a as shown in Tab. 5.10. The values of b_{las} and b_{cam} are almost equal, i.e. the calculation for each sensor was made correctly. Also it can be seen that $x_{rel,las}$ and $x_{rel,cam}$ are almost equal as well. The obstacle relative and absolute coordinates are the same because the robot is in the position (0,0).

Case 1	$x_{rel,las}$ (cm)	$y_{rel,las}$ (cm)	$x_{rel,cam}$ (cm)	x_{rel} (cm)	x_{obs} (cm)	y_{rel} (cm)	y_{obs} (cm)
	130.24	5.26	138.389	134.31	134.31	5.26	5.26
	α (rad)	W_{obs} (cm)	W_{real} (cm)	b_{las} (cm)	b_{cam} (cm)	b (cm)	a (cm)
	0.2495	13.87	14.31	70.52	74.31	72.41	120

Table 5.10: Data fusion parameters for case 1

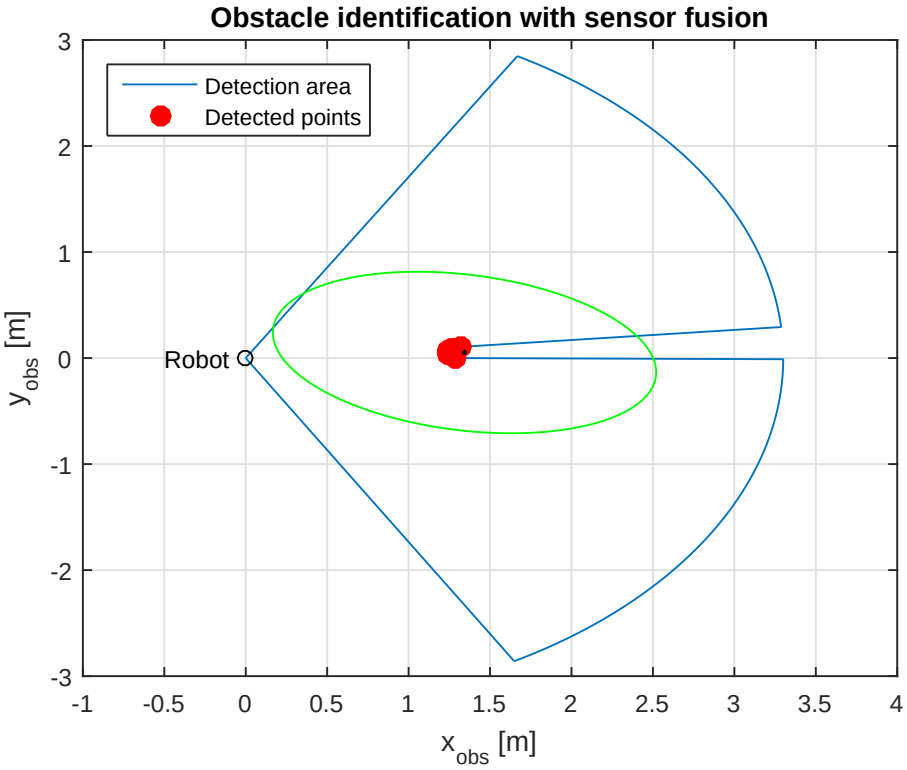


Figure 5.22: Obstacle determination and identification with sensor fusion for case 1

In case 2 there are two obstacles but taken as one and they are surrounded by an ellipse defined by the parameters b and a as shown in Tab. 5.11. It can be seen that now the ellipse is bigger due to the distance threshold restriction between obstacles explained before. The values of b_{las} and b_{cam} are almost equal, i.e. the calculation for each sensor was made correctly. Also it can be seen that $x_{rel,las}$ and $x_{rel,cam}$ are almost equal as well. The obstacle relative and absolute coordinates are the same because the robot is in the position (0,0) of the XY-plane.

Case 2	$x_{rel,las}$ (cm)	$y_{rel,las}$ (cm)	$x_{rel,cam}$ (cm)	x_{rel} (cm)	x_{obs} (cm)	y_{rel} (cm)	y_{obs} (cm)
	171.09	-11.67	178.45	174.77	174.77	-11.67	-11.67
	α (rad)	W_{obs} (cm)	W_{real} (cm)	b_{las} (cm)	b_{cam} (cm)	b (cm)	a (cm)
	-0.7854	37.87	53.55	104.39	113.55	108.97	120

Table 5.11: Data fusion parameters for case 2

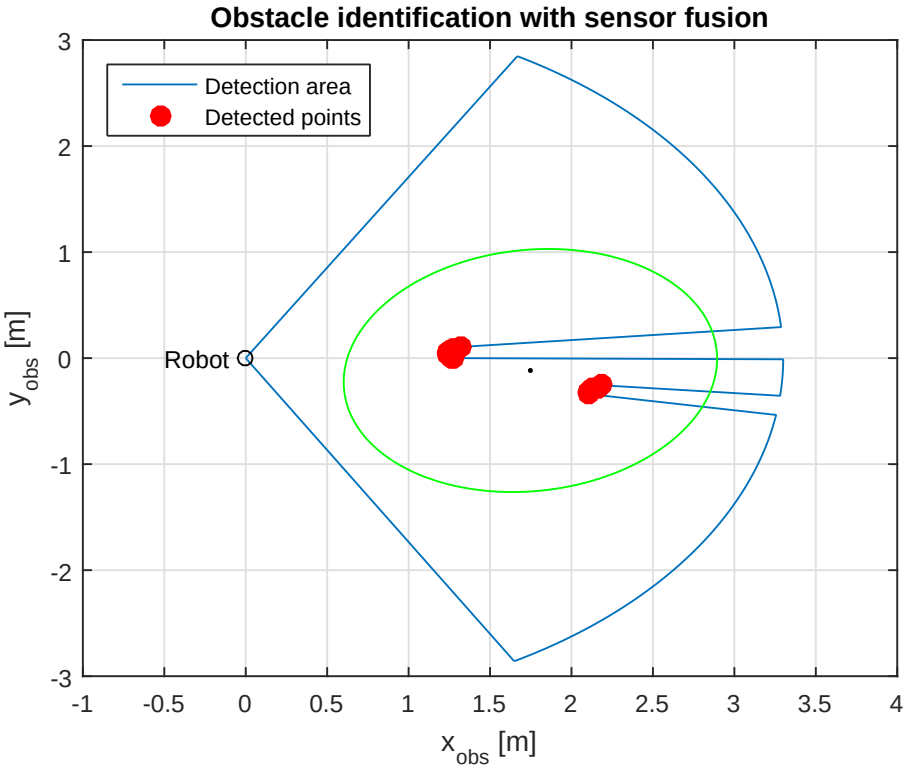


Figure 5.23: Obstacle determination and identification with sensor fusion for case 2

6 Dynamic optimal control problem

The dynamic optimization problem is a problem in which the states and control variables change over a predefined period of time. The main goal of the optimization problem, is to minimize the cost function by finding the optimal control values and taking in account certain equality, inequality and constraints conditions.

6.1 Continuous nonlinear optimal control problem

Since the main task is to minimize the values of the control variables, then the continuous nonlinear optimization problem is defined as follows,

$$\min_{u(t)} \left\{ J = \phi(x(t_f), t_f) + \int_{t_0}^{t_f} f_0(x(t), u(t), t)dt \right\} \tag{6.1}$$

$$s.t. \qquad \dot{x}(t) = f(x(t), u(t), t), \;\; t \in [t_0, t_f] \tag{6.2}$$

$$0 \leq g(x(t), u(t), t) \tag{6.3}$$

$$x(t_0) = x_0 \tag{6.4}$$

$$x_{min} \leq x(t) \leq x_{max} \tag{6.5}$$

$$u_{min} \leq u(t) \leq u_{max} \tag{6.6}$$

The cost function known as a Bolza problem is shown in (6.1), the system equations in (6.2), the inequality constraints in (6.3), the initial state in (6.4). The state constraints in (6.5) and the control constraints in (6.6).

The state and control variables as seen in chapter 2, is defined as follows,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_{pos} \\ y_{pos} \\ \psi \end{bmatrix} \quad , \quad u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} v \\ \delta \end{bmatrix} \tag{6.7}$$

and so the system equations,

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} v\ \cos(\psi)\ \cos(\delta) \\ v\ \sin(\psi)\ \cos(\delta) \\ \frac{2v}{L}\ \sin(\delta) \end{bmatrix} \tag{6.8}$$

Then the state constraints are defined according to the real environment,

$$\begin{aligned} -\infty \leq x_1(t) \leq \infty \\ -\infty \leq x_2(t) \leq \infty \\ -\pi \leq x_3(t) \leq \pi \end{aligned} \tag{6.9}$$

where $x_1(t)$ is the position x_{pos} in the X-direction, $x_2(t)$ is the position y_{pos} in the Y-direction, and $x_3(t)$ is the yaw angle ψ. It can be seen that the position in the XY-plane is not limited, because the mobile robot can move in any direction in this plane.

Also the control constraints are defined as follows,

$$\begin{aligned} 0.1\ \frac{m}{s} \leq u_1(t) \leq 1\ \frac{m}{s} \\ -0.1\ rad \leq u_2(t) \leq 0.1\ rad \end{aligned} \tag{6.10}$$

where $u_1(t)$ is the velocity v and $u_2(t)$ is the steering angle δ. The minimum value of the speed is different from 0 in order to fix the trajectory if the mobile robot enters to a constrained area and the maximum value is set as 1 m/s due to the accuracy of the reduced 3-state model. The values of the steering angle are defined taken into account the mechanical construction and the correction factors that are needed to be done.

The inequality constraint is defined by the ellipses of the detected obstacles, i.e. if the robot is moving from the initial to the desired final position and then it detects an obstacle, the robot must avoid it taking into account the following ellipse equation,

$$1 \leq \frac{(x_{pos} - x_{obs})^2}{a^2} + \frac{(y_{pos} - y_{obs})^2}{b^2} \tag{6.11}$$

where x_{pos} and y_{pos} are the robot position coordinates in the XY-plane, x_{obs} and y_{obs} are the obstacle position coordinates in the XY-plane, a is the ellipse major axis and b is the ellipse minor axis. A variation of this ellipse equation can be done by adding

the inclination angle α (referring to Fig. 5.21b) as follows,

$$1 \leq \frac{\left[(x_{pos} - x_{obs})\cos(\alpha) + (y_{pos} - y_{obs})\sin(\alpha)\right]^2}{a^2} + \frac{\left[(x_{pos} - x_{obs})\sin(\alpha) - (y_{pos} - y_{obs})\cos(\alpha)\right]^2}{b^2} \tag{6.12}$$

6.2 Transformation into a nonlinear optimization problem

The optimization problem described previously is a continuous nonlinear optimal control problem, since it is formulated in the continuous time domain, the complexity in obtaining the solution to this problem increases considerably. In general, the solution methods for optimal control problems are divided into three categories: dynamic programming, indirect methods and direct methods as shown in Fig. 6.1 (Li, 2017).

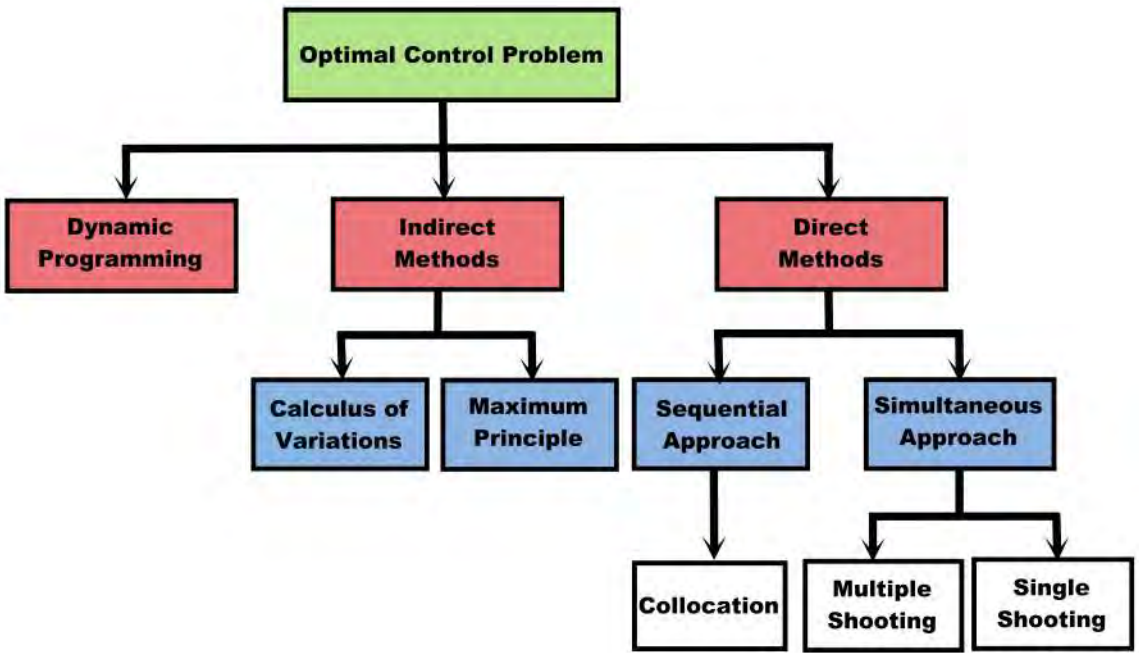


Figure 6.1: Solution methods for optimal control problems

In summary, dynamic programming method is restricted to small-scale dimension problems. Indirect methods, cannot solve large-scale problems within an acceptable time, which is a crucial factor in online optimization. On the other hand, direct methods can solve the optimal control problem by transforming it into a nonlinear optimization

problem (NLP) which is then solved using numerical optimization methods. So first the problem must be discretized, i.e. transform the continuous dynamics of the system into a series of discrete equations. Then the optimization of the NLP can be done. Compared to indirect methods, these methods can easily manage linear and nonlinear inequality constraints, making them very useful for real applications, i.e. direct methods can solve very large-scale and nonlinear problems by using a wide range of solvers.

In direct methods, the most popular variant of the sequential approach is the direct single-shooting method, while for the simultaneous approach the collocation on finite elements,and the multiple-shooting methods are available. This thesis deals with a solution based on multiple-shooting with a three-point collocation method. More information and application of this method can be found in (Correa, 2016), (Müller, 2013), (Geletu, 2016) and (Li, 2017). As a result of using the direct methods, the optimization task is reduced to solving a NLP in order to obtain the optimal control variables. Then the resulting NLP can be described as follows,

$$\min_{w} F(w) \tag{6.13}$$

$$G(w) = 0 \tag{6.14}$$

$$H(w) \leq 0 \tag{6.15}$$

$$w_{min} \leq w \leq w_{max} \tag{6.16}$$

where $F(w)$ is the objective function, $G(w)$ is the equality constraint, $H(w)$ is the inequality constraint, and w is the vector of the optimization variables which considers the parametrized control variables and also the approximation of the state variables at each discrete time.

So first to perform this transformation the time horizon $[t_0, t_f]$ must be discretized into N time intervals, i.e. divide the time horizon in intervals $[t_k, t_{k+1}]$ where $k = 0, 1, 2, ...N-1$ with three collocation points in each interval.

$$t_k = t_{k,0} < t_{k,1} < t_{k,2} < t_{k,3} = t_{k+1} \tag{6.17}$$

where $t_{k,i}$ for $i = 0, ..., 3$ are the collocation points.

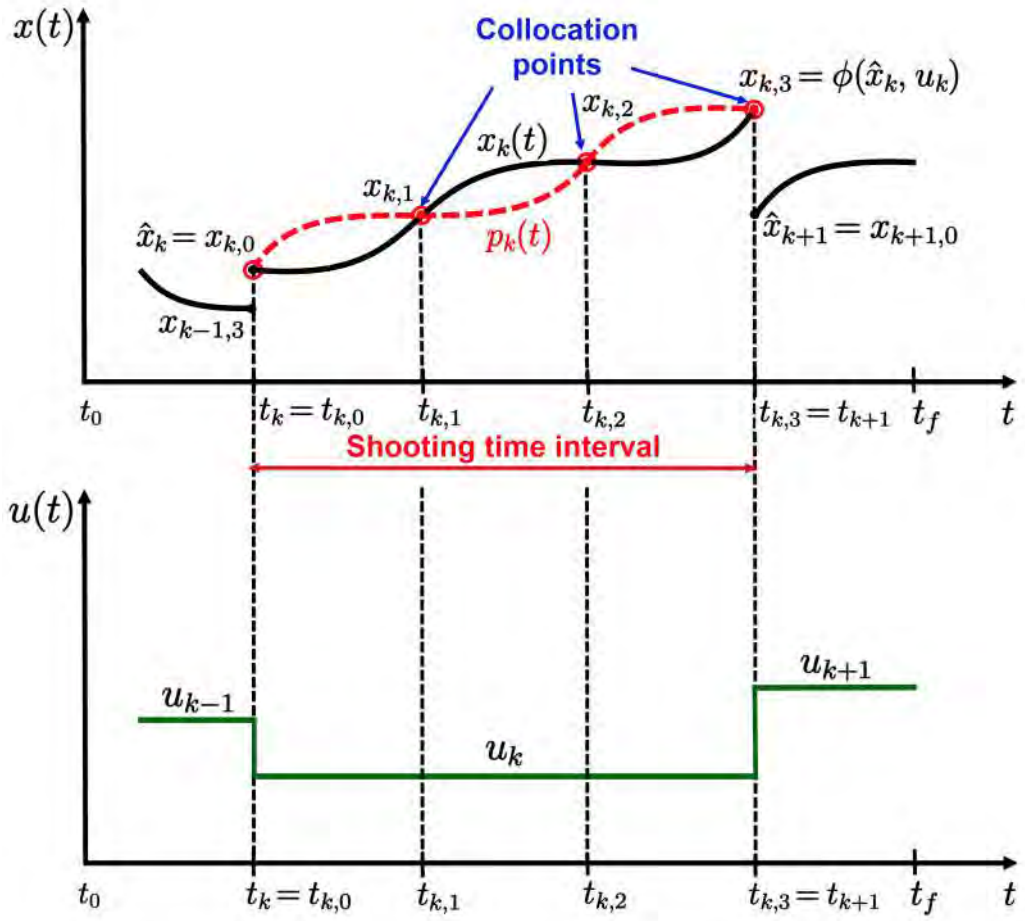


Figure 6.2: Collocation points, collocated state and control in time interval $[t_k, t_{k+1}]$

It can be seen from Fig. 6.2 (Drozdova, 2015) that $x_{k,i}$ are the discretized values of the state trajectory at these collocation points. $x_{k,0}$ is the initial point of the time interval $[t_k, t_{k+1}]$ and its value is given by the optimization variable $\hat{x}_k$. In a similar way, $x_{k,3}$ is the final point of the time interval $[t_k, t_{k+1}]$ and its value represents the value of $\phi_k(\hat{x}_k, u_k)$. The control variable is constant in each shooting time interval, while the state variable trajectory is approximated by a linear combination of the Lagrange polynomials, i.e.,

$$\begin{aligned} u_k &= u(t) \quad , \quad p_k(t) = \sum_{i=0}^{3} l_{k,i}(t)\, x_{k,i} \\ k &= 0, 1, ..., N-1 \\ t &= [t_k, t_{k+1}] \end{aligned} \tag{6.18}$$

where $p_k(t)$ is the approximated polynomial for the time interval $[t_k, t_{k+1}]$. It can be

seen that this polynomial has the same values of the state trajectory at the collocation points. Then the Lagrange polynomials are defined as follows,

$$l_{k,i}(t) = \prod_{i=0, i\neq j}^{3} \frac{(t - t_j)}{t_i - t_j} \tag{6.19}$$

and the time derivative of $p_k(t)$ from (6.18) is given by,

$$\dot{p}_k(t) = \sum_{i=0}^{3} \frac{dl_{k,i}(t)}{dt} x_{k,i} \tag{6.20}$$

Thus using (6.18) in the state equation (6.2), the state variables values $x_{k,i}$ can be calculated by Newton's method.

The vector of optimization variables with the previous information can be defined as,

$$w = \begin{bmatrix} \hat{x}_0 & u_0 & \hat{x}_1 & u_1 & \hat{x}_2 & u_2 & \dots & u_{N-1} & \hat{x}_N \end{bmatrix}^T \tag{6.21}$$

Then the inequality constraint $H(w)$ consists of a set of equations that depend on the variables $\hat{x}_k$ and u_k.

$$H(w) = \begin{bmatrix} h(\hat{x}_0, u_0) \\ h(\hat{x}_1, u_1) \\ h(\hat{x}_2, u_2) \\ \vdots \\ h(\hat{x}_N, u_N) \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{6.22}$$

where h is defined as the obstacle ellipse equation shown in (6.12).

For the determination of the equality constraint $G(w)$, first is important to specify that the state trajectory shown in Fig. (6.2) must be continuous in the time horizon, i.e. that the initial state value $\hat{x}_k$ of each interval must be equal to the final value of the trajectory of the previous interval.

$$\begin{aligned} \hat{x}_{k+1} &= x_k(t_{k+1}, \hat{x}_k, u_k) \\ &= \phi_k(\hat{x}_k, u_k) \end{aligned} \tag{6.23}$$

where $\phi_k(\hat{x}_k, u_k)$ is the value of the polynomial $p_k(t)$ at the last collocation point. Then the equality constraint $G(w)$ is obtained taken into account the initial state $\hat{x}_0 = x(0)$

and (6.23) with $k = 0, ..., N-1$.

$$G(w) = \begin{bmatrix} \hat{x}_0 - x(0) \\ \hat{x}_1 - \phi_0(\hat{x}_0, u_0) \\ \hat{x}_2 - \phi_1(\hat{x}_1, u_1) \\ \vdots \\ \hat{x}_N - \phi_0(\hat{x}_{N-1}, u_{N-1}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{6.24}$$

Finally, the cost function is presented as follows,

$$F(w) = x_N^T \, P \, x_N + \sum_{k=0}^{N-1} (x_k^T \, Q \, x_k + u_k^T \, R \, u_k) \tag{6.25}$$

with

$$x_N = \begin{bmatrix} x_{pos,N} & y_{pos,N} \end{bmatrix}^T \tag{6.26}$$

$$x_k = \begin{bmatrix} x_{pos,k} & y_{pos,k} \end{bmatrix}^T \tag{6.27}$$

$$u_k = \begin{bmatrix} v_k & \delta_k \end{bmatrix}^T \tag{6.28}$$

where P is the weighting matrix for the desired final states, while Q and R are the weighting matrices for the states and control values during the optimization task, respectively. x_N and x_k are the vectors of the state variables in the discrete time N and k, respectively, while u_k is the vector of the control variables in the time interval $[k, k+1]$. A detailed procedure of this transformation can be found in (Lazutkin, Geletu, Hopfgarten, & Li, 2014).

After the transformation of the continuous optimal control problem into an NLP, a numerical solution is obtained with the numerical solver Ipopt[3]. More information about Ipopt can be found in (Wächter & Biegler, 2006).

6.3 Model predictive control

The main idea of the model predictive control (MPC) is that it computes the optimal controls within an control time horizon and predicts the future states/outputs in the prediction time horizon taking into account the actual states and possibly a

[3]Ipopt (Interior Point Optimizer) is an open-source software package for solving very large-scale NLP. It is based on a primal-dual interior-point line-search filter method.

prediction of disturbances in the prediction horizon. Often the control horizon equals the prediction horizon. Because of this, MPC provides a quasi-online solution to the optimization problem. Then the control/prediction horizon is defined as follows,

$$t_c = t_p = N\, t_s \tag{6.29}$$

where N is the number of intervals in the control/prediction horizon and t_s is the sampling time.

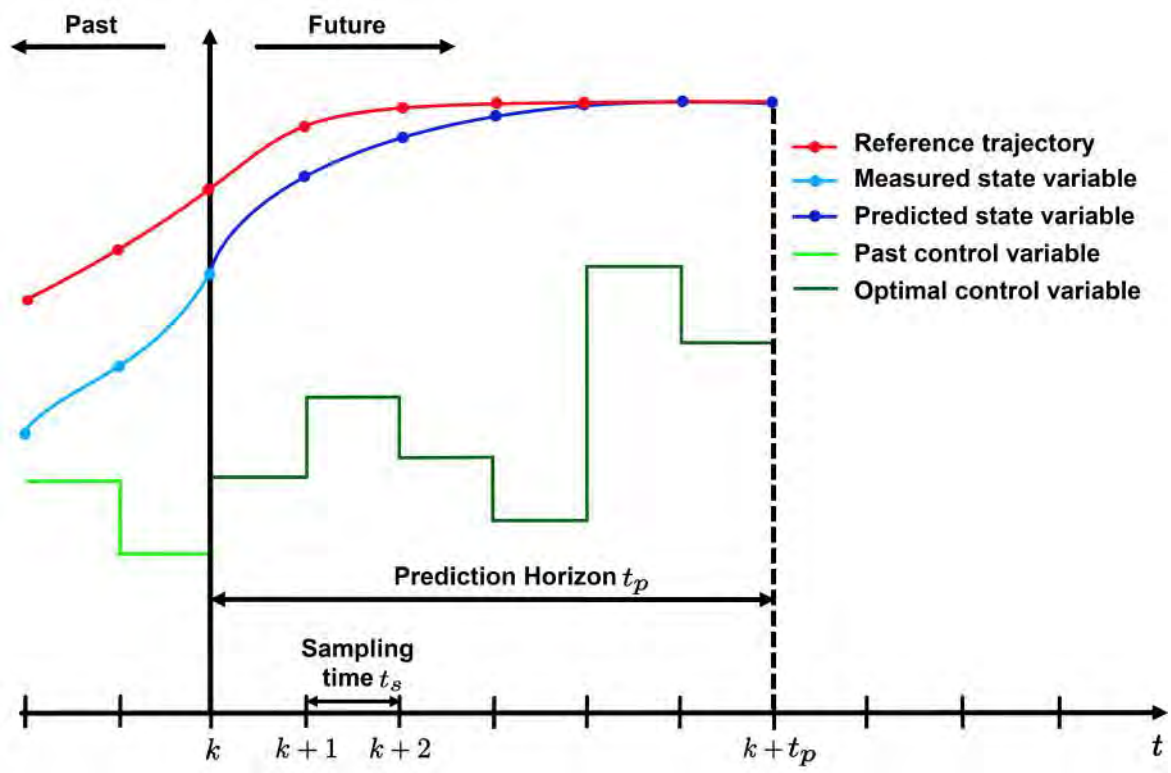


Figure 6.3: The MPC principle

Fig. 6.3 based on (Thieme, 2014) shows the MPC principle. So first the state variable $x(k)$ is measured or estimated at the discrete time k. Then the optimization problem is solved over the entire prediction horizon considering the constraints. After this, the calculated optimal control $u(k)$ is applied in the interval $[k, k+1]$. Meanwhile the prediction horizon is shifted to the right by one time interval and the state variable is again measured or estimated at the discrete time $k+1$. Then again the optimization problem is solved and the new calculated optimal control $u(k+1)$ is applied in the interval $[k+1, k+2]$. These steps are repeated until the reference is reached, i.e. the error between the desired value and the current state is small enough. The advantages of the MPC are the possibility to determine an optimal control and future behavior of

the system considering constraints, and to take into account predictable or measurable disturbances. The disadvantage is the high numerical calculation complexity, but it is reduced as a result of the combination of mutiple-shooting and collocation method explained previously.

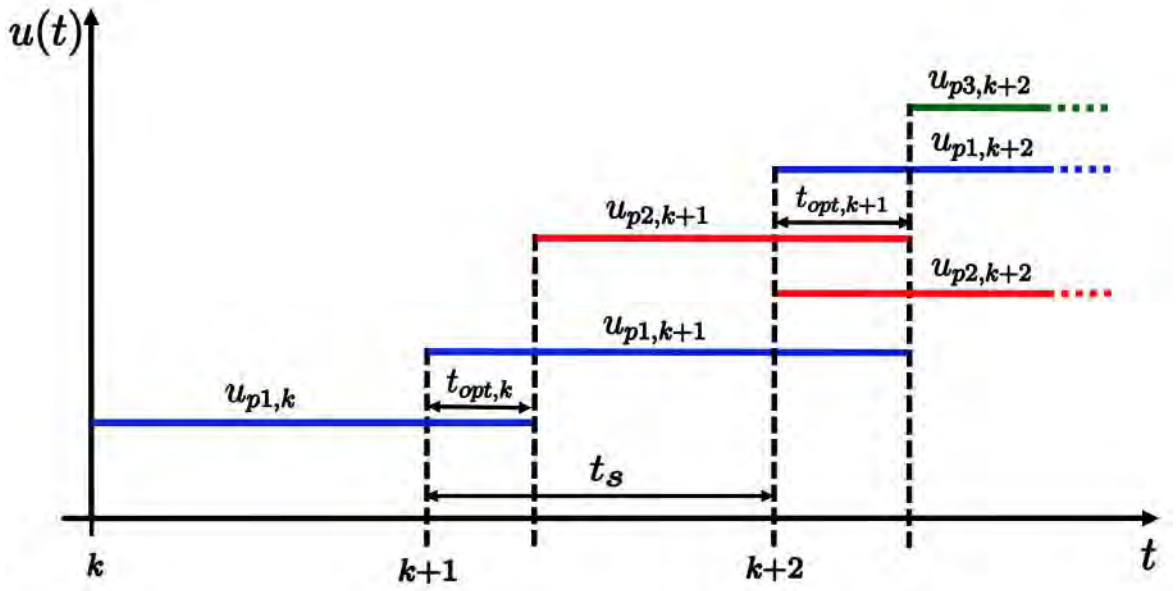


Figure 6.4: Control variable behavior in the prediction horizon

Fig. 6.4 based on (Belgradskaia, 2016) shows the behavior of the control variable in the prediction horizon depending on the calculated time t_{opt}. So first is calculated the optimal control variable u_{p1} for the entire prediction horizon $p1$ due to the mathematical model. Then in the first time interval $[k, k+1]$ the optimal control value $u_{p1,k}$ is used. At the end of this time interval, i.e. at the discrete time $k+1$ the state variables and the sensor data are obtained. Now, based on this information the optimal control variable u_{p2} is calculated again for the entire prediction horizon $p2$. However during the calculation time $t_{opt,k}$ of this new optimal control variable, $u_{p1,k}$ is used from the previous time interval. When $t_{opt,k}$ ends, the new calculated control variable $u_{p2,k+1}$ is used for the rest of the time interval $[k+1, k+2]$. Finally at the discrete time $k+2$ the state variables and the sensor data are obtained again and then the optimal control variable u_{p3} is calculated. This procedure is repeated until the target is reached. It is important to notice that t_{opt} must be lower than the sampling time t_s in order to make the MPC works as expected.

6.4 Solution of the optimization problem

The entire control structure of the mobile robot SUMMIT can be seen in Fig. 6.5. The input of the control system is the reference trajectory or desired final position. Then in the optimization block the MPC with multiple-shooting and collocation method are used in order to solve the optimization problem, the constraints are specified by the sensor fusion data from both the laser scanner and the 2D camera, while the cost function is predefined by the MPC algorithm. The optimal control values then are corrected with the linear relationship and the Neural network of the velocity and steering angle, respectively. Later, these corrected values are sent to the mobile robot SUMMIT and finally with the EKF one can get the estimated state variables.

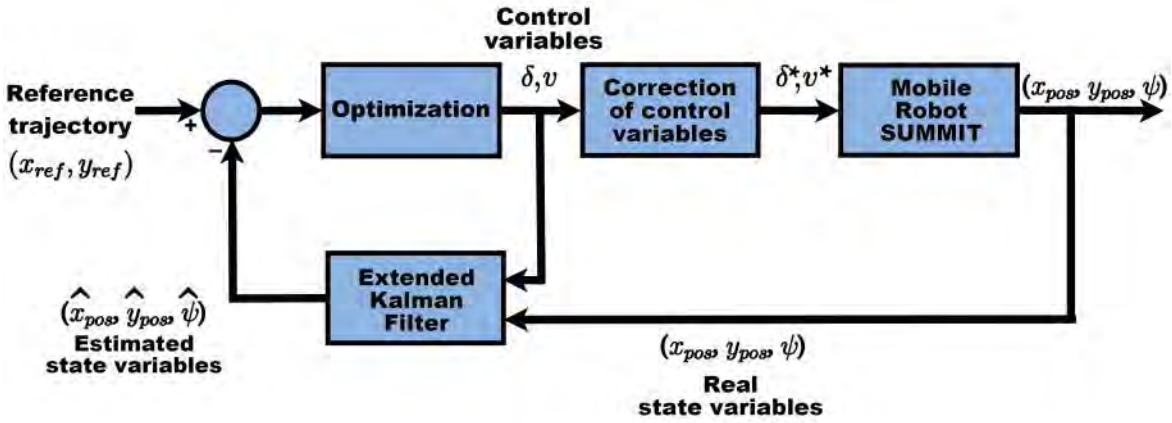


Figure 6.5: Block diagram of the control system architecture

As it can be seen the main objective is to reduce the deviation between the input and the output of the control system via optimization procedures. Thus in order to verify the accuracy of the MPC, two kind of optimization problems are discussed: trajectory tracking problem and obstacle avoidance.

6.4.1 Trajectory tracking problem

The trajectory tracking problem is a classic problem in the field of autonomous mobile robots. This problem consists in finding the optimal control variables in order to follow a reference trajectory (x_{ref}, y_{ref}). Within the scope of MPC, this problem is solved by minimizing the cost function J_{track} in the prediction horizon, i.e. minimize the deviation between the current and the reference position in the XY-plane at each

discrete time k. Then the cost function to be minimized is as follows,

$$\begin{aligned} J_{track} =& P_x \left(x_{pos,N} - x_{ref,N}\right)^2 + P_y \left(y_{pos,N} - y_{ref,N}\right)^2 \\ &+ \sum_{k=1}^{N-1} \left[Q_x \left(x_{pos,k} - x_{ref,k}\right)^2 + Q_y \left(y_{pos,k} - y_{ref,k}\right)^2 + R_v \, v_k^2 + R_\delta \, \delta_k^2 \right] \end{aligned} \tag{6.30}$$

where $N = 20$ is the selected number of time intervals of the prediction horizon. Then the weights are defined as,

	P_x	P_y	Q_x	Q_y	R_v	R_δ
Weights	8.0	8.0	0.7	0.7	0.1	0.1

Table 6.1: Weighting values for trajectories tracking problem

The weights of R_v and R_δ are low because the main goal in this kind of problem is the reduction of the deviation between the reference and the current position. The values of Q_x and Q_y are chosen in order to adjust the current trajectory to the reference. Finally the values of P_x an P_y are relatively big for make the current position reach the last value of the reference trajectory. Thus the resulting NLP is defined as follows,

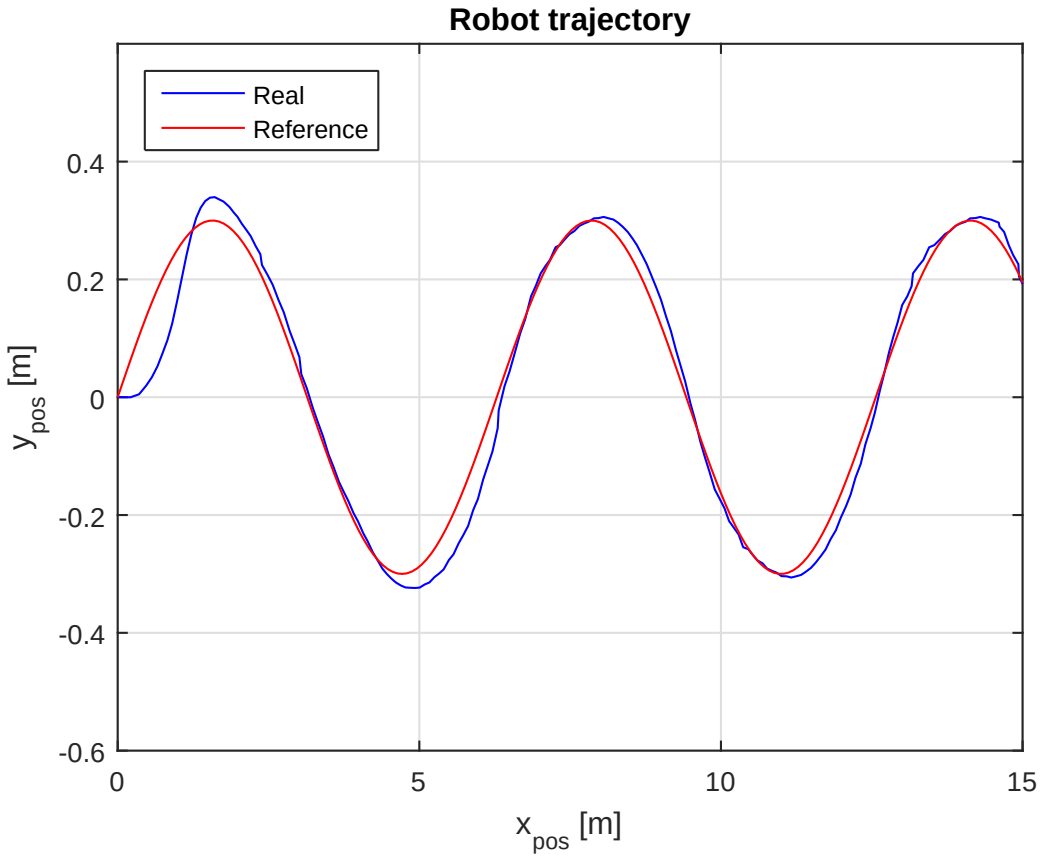
$$\begin{aligned} \min_{u} \quad & J_{track}(x, u) \\ \text{s.t.} \quad & x(0) = x_0 \\ & \text{system equations (6.8)} \\ & \text{state constraints (6.9)} \\ & \text{control constraints (6.10)} \end{aligned}$$

In order to test the performance of the proposed optimization problem, two cases are presented: case 1 ($t_s = 0.12$ s with sinusoidal reference trajectory), and case 2 ($t_s = 0.17$ s with sinusoidal reference trajectory). Since it is a trajectory tracking problem, there is no need to use the laser scanner or the camera information. The value of $t_s = 0.17$ s was choosen because this sampling time was used by (Belgradskaia, 2016), while the value of $t_s = 0.12$ s was choosen because it is a lower value than the previous one proposed.

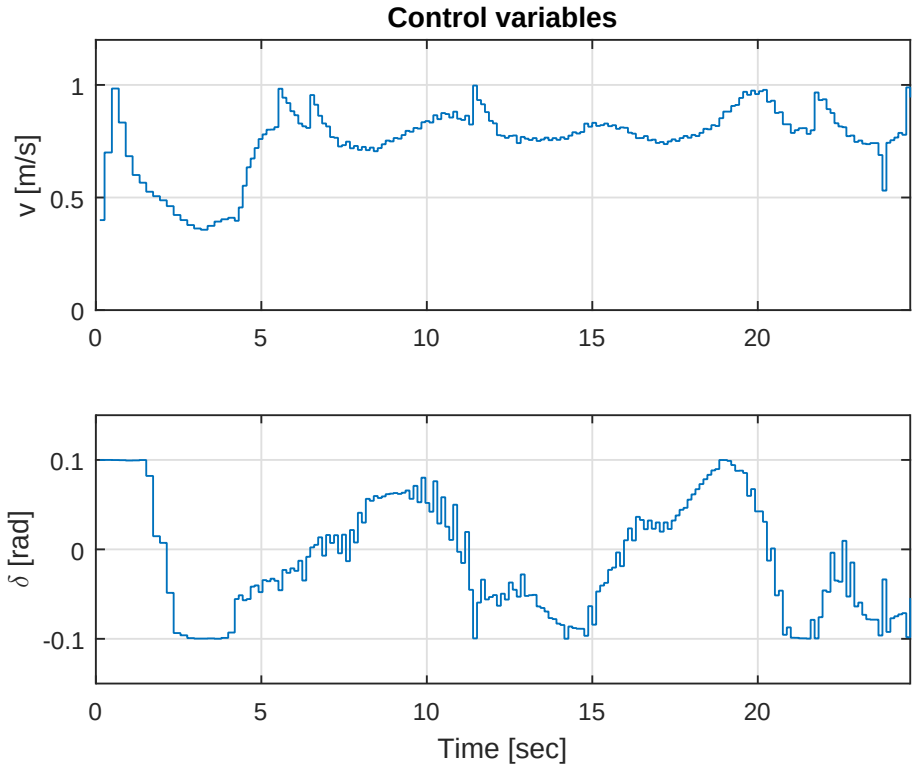
Case 1 is a trajectory tracking problem with a sinusoidal reference trajectory and the selected sampling time is 0.12 s. Fig. 6.6a shows the real trajectory in the XY-plane, Fig. 6.6b shows the control variables along the trajectory and Fig. 6.6c shows the computation time that the solution of the optimal control problem needs to be solved.

Case 1	
Type of problem	Trajectory tracking
Reference	$0.3\sin(wt)$
Sampling time	$t_s = 0.12$ s

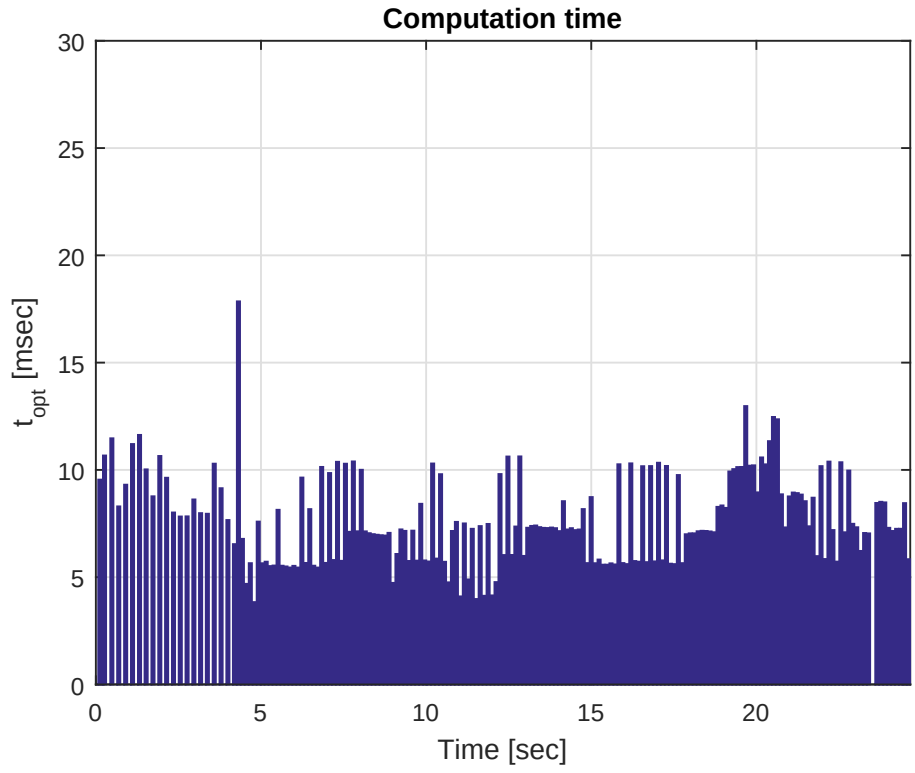
Table 6.2: Considerations for case 1



(a) Robot trajectory in the XY-plane



(b) Control variables



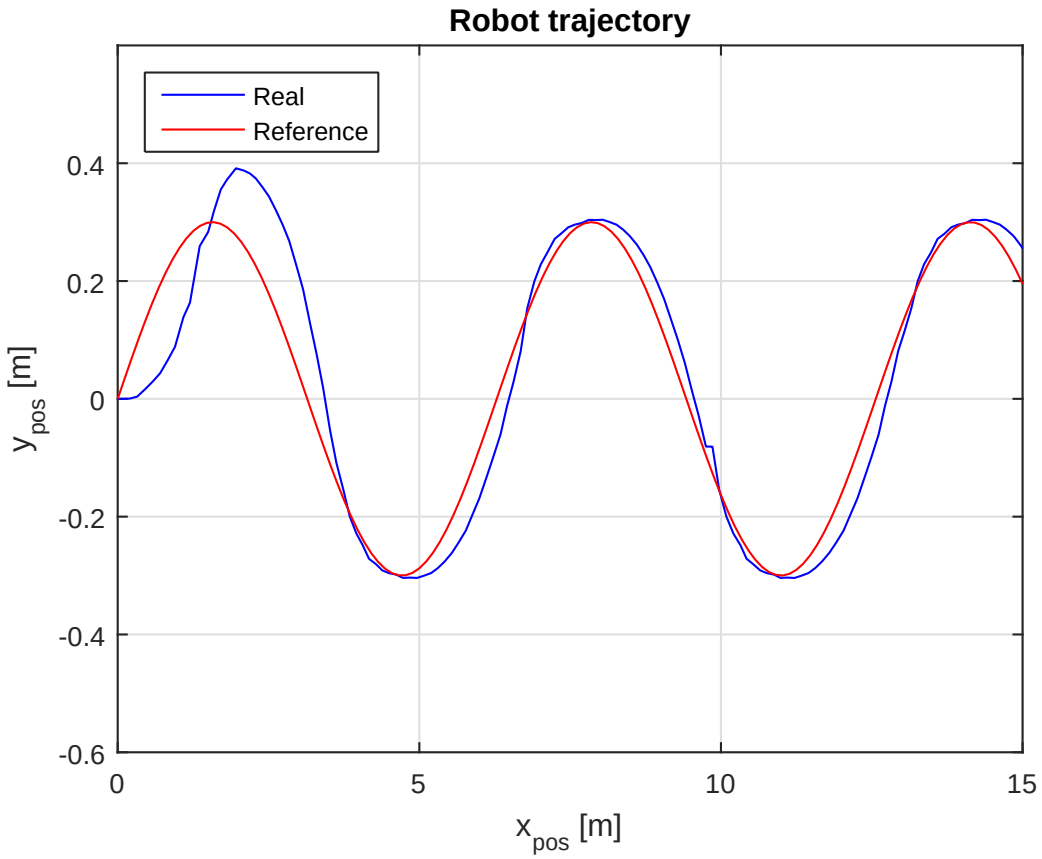
(c) Computation time

Figure 6.6: Results for case 1. Tracking problem with $t_s = 0.12$ s

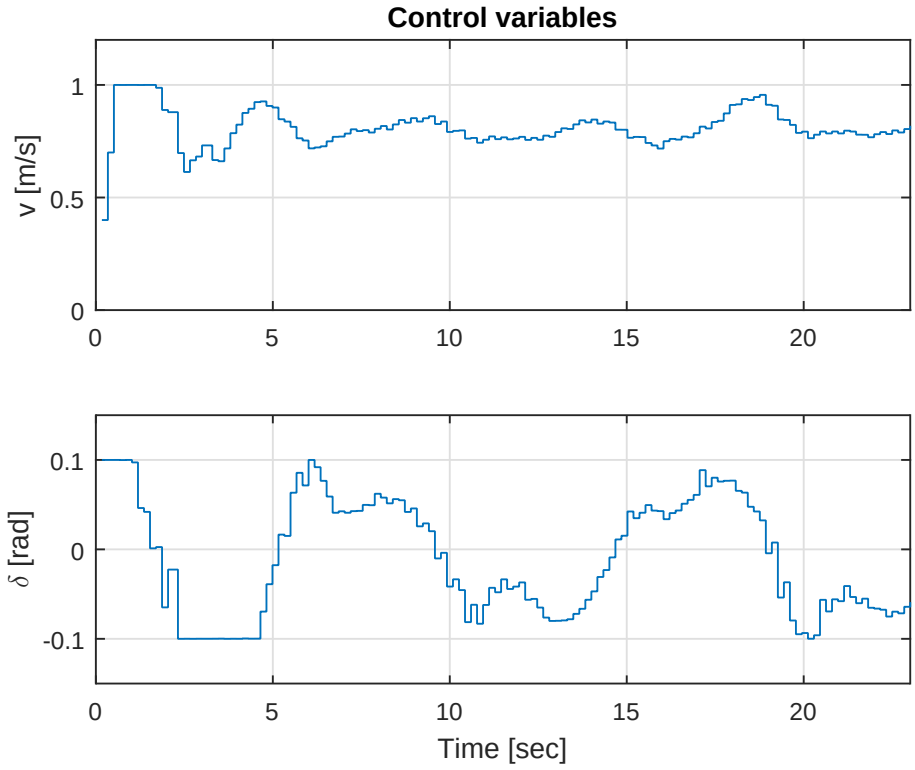
Case 2 is also a trajectory tracking problem with the same sinusoidal reference trajectory but now the selected sampling time is 0.17 s. Fig. 6.7a shows the real trajectory in the XY-plane, Fig. 6.7b shows the control variables along the trajectory and Fig. 6.7c shows the computation time that the solution of the optimal control problem needs to be solved.

Case 2	
Type of problem	Trajectory tracking
Reference	$0.3\sin(wt)$
Sampling time	0.17 s

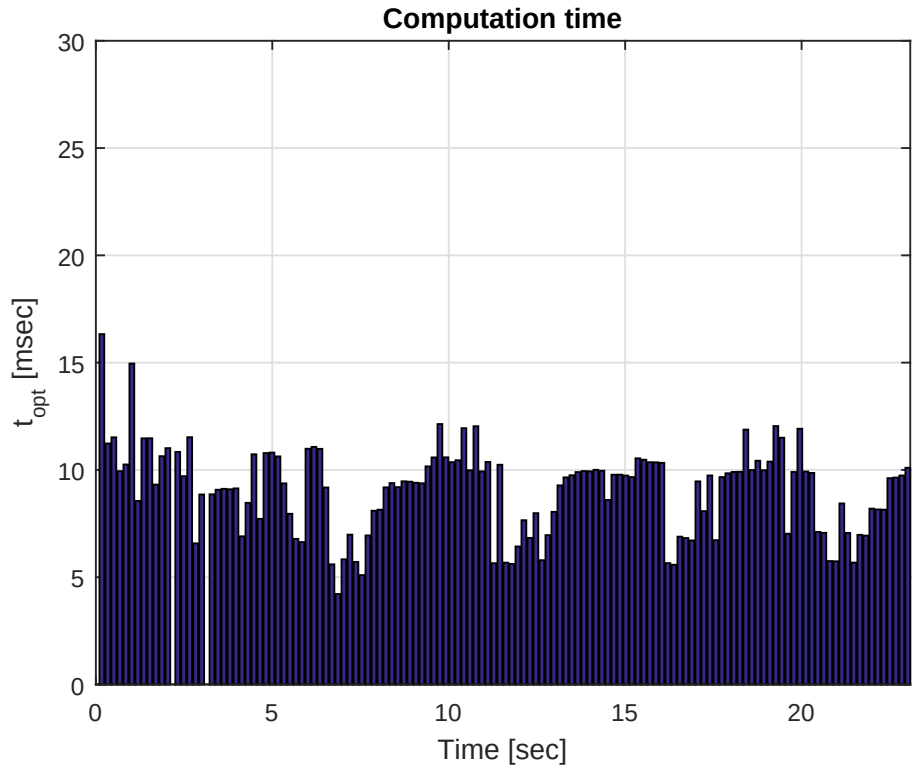
Table 6.3: Considerations for case 2



(a) Robot trajectory in the XY-plane



(b) Control variables



(c) Computation time

Figure 6.7: Results for case 2. Tracking problem with $t_s = 0.17$ s

6.4.2 Obstacle avoidance problem

The obstacle avoidance problem is also a classical problem in the field of autonomous mobile robots. This problem consists in finding the optimal control variables in order to reach at a desired final position by following a trajectory (x_d, y_d) while avoiding all the obstacles along it. Then the cost function to be minimized is as follows,

$$\begin{aligned} J_{avoid} =& P_x \left(x_{pos,N} - x_{end,N}\right)^2 + P_y \left(y_{pos,N} - y_{end,N}\right)^2 \\ &+ \sum_{k=1}^{N-1} \left[Q_x \left(x_{pos,k} - x_{end,k}\right)^2 + Q_y \left(y_{pos,k} - y_{end,k}\right)^2 + R_v \, v_k^2 + R_\delta \, \delta_k^2 \right] \end{aligned} \tag{6.31}$$

where $N = 20$ is the selected number of time intervals of the prediction horizon. Then the weights are defined as,

	P_x	P_y	Q_x	Q_y	R_v	R_δ
Weights	0.5	2	0.1	1	0.05	0.05

Table 6.4: Weighting values for obstacle avoidance problem

The weights of R_v and R_δ are low because the main goal in this kind of problem is the reduction of the deviation between the desired and the current position. The value of Q_y is greater than Q_x in order to reduce the movement of the mobile robot in the Y-direction. Likewise the value of P_y is greater than P_x in order to penalize the deviation between the desired and current position in the discrete time N. Thus the resulting NLP is defined as follows,

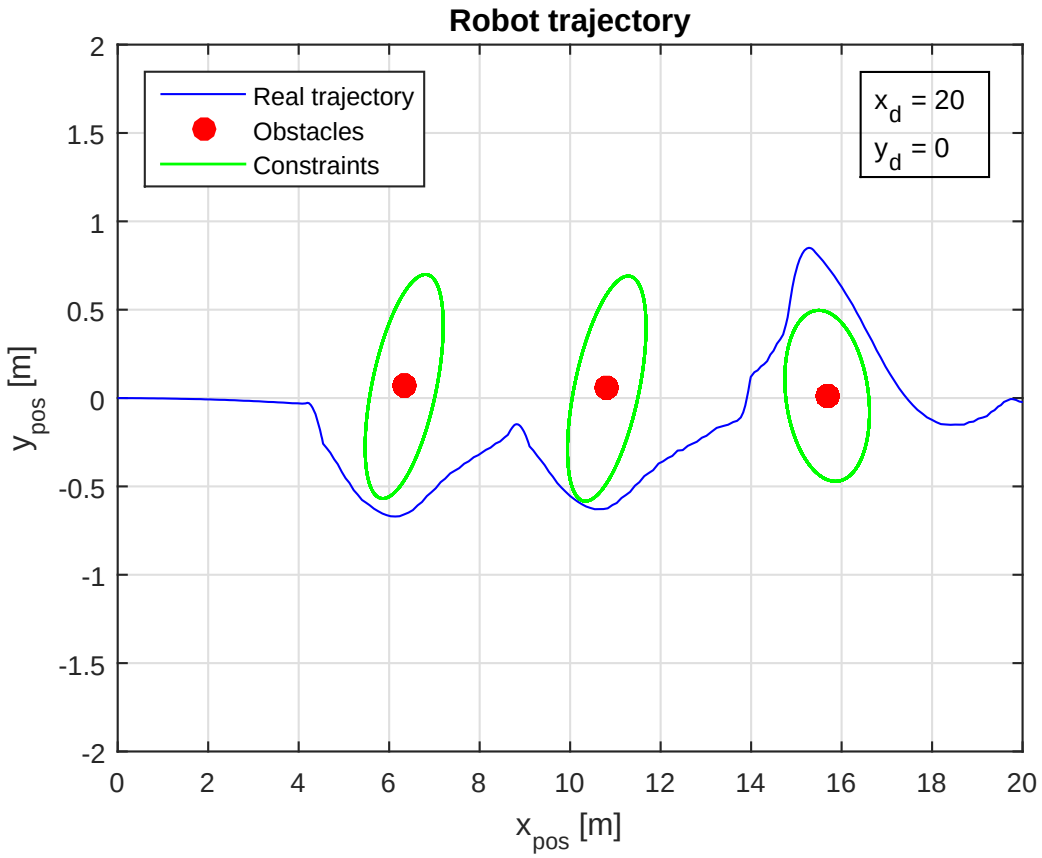
$$\begin{aligned} \min_{u} \quad & J_{avoid}(x, u) \\ \text{s.t.} \quad & x(0) = x_0 \\ & \text{system equations } (6.8) \\ & \text{state constraints } (6.9) \\ & \text{control constraints } (6.10) \\ & \text{obstacle path constraints } (6.12) \end{aligned}$$

In order to test the performance of the proposed optimization problem, four cases are presented: case 3 ($t_s = 0.12$ s with 3 obstacles), case 4 ($t_s = 0.17$ s with 3 obstacles), case 5 ($t_s = 0.12$ s with 5 obstacles), and case 6 ($t_s = 0.17$ s with 5 obstacles).

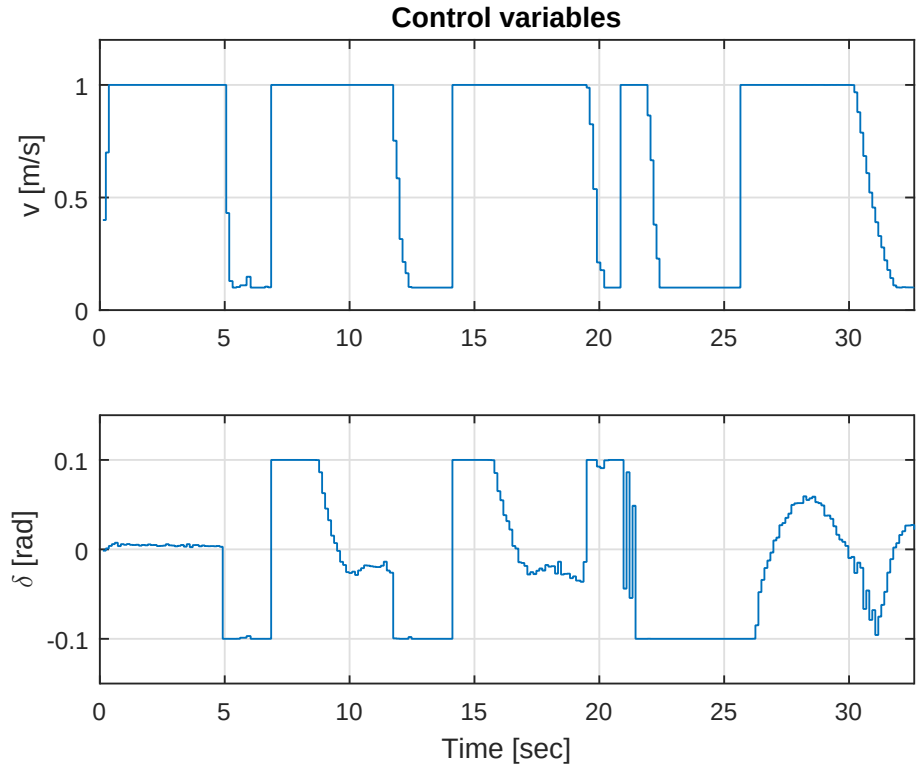
Case 3 is an obstacle avoidance problem with a fixed end position in the XY-plane and a selected sampling time of 0.12 s. Fig. 6.8a shows the mobile robot trajectory with three obstacles along the way in the XY-plane, Fig. 6.8b shows the control variables along the trajectory and Fig. 6.8c shows the computation time that the solution of the optimal control problem needs to be solved.

Case 3	
Type of problem	Obstacle avoidance
Reference	$(x_d, y_d) = (20, 0)$
Sampling time	0.12 s

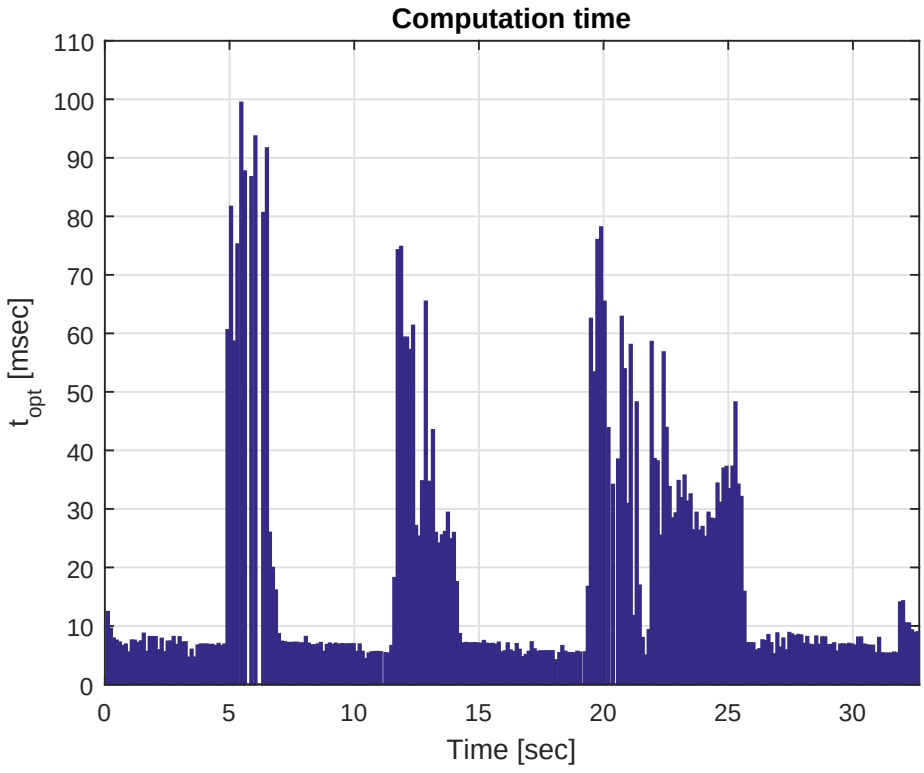
Table 6.5: Considerations for case 3



(a) Robot trajectory in the XY-plane



(b) Control variables



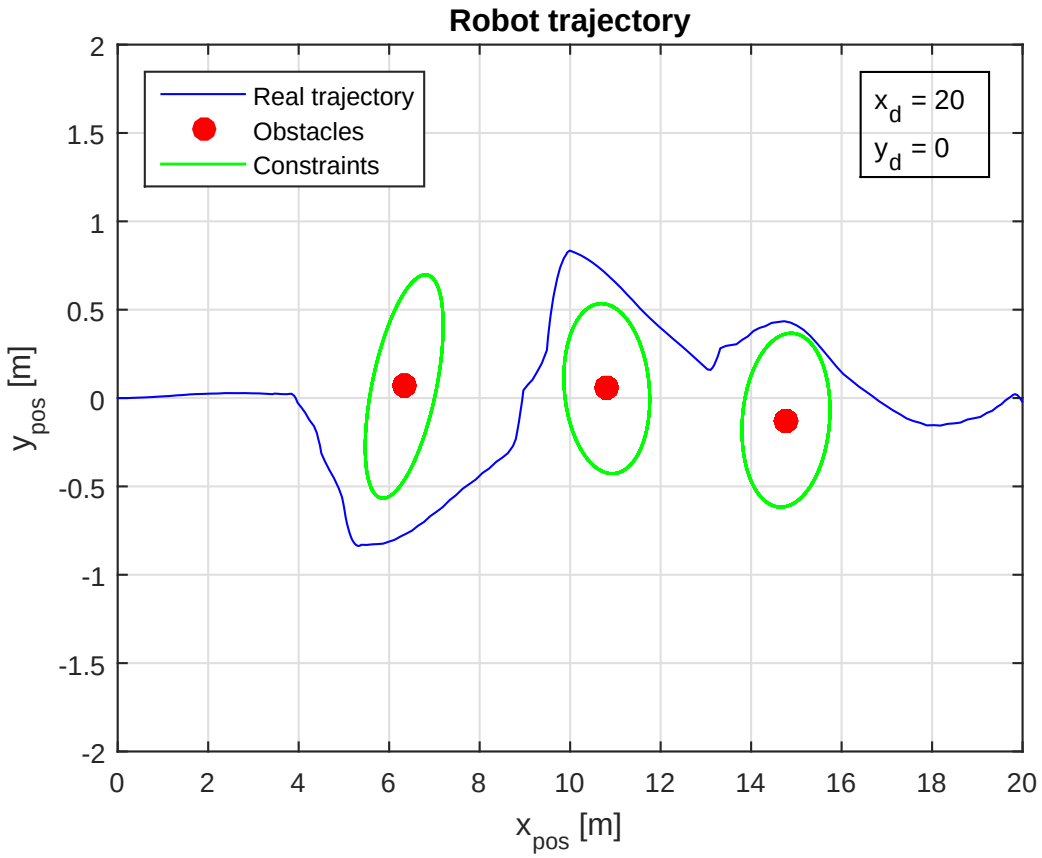
(c) Computation time

Figure 6.8: Results for case 3. Obstacle avoidance with $t_s = 0.12$ s

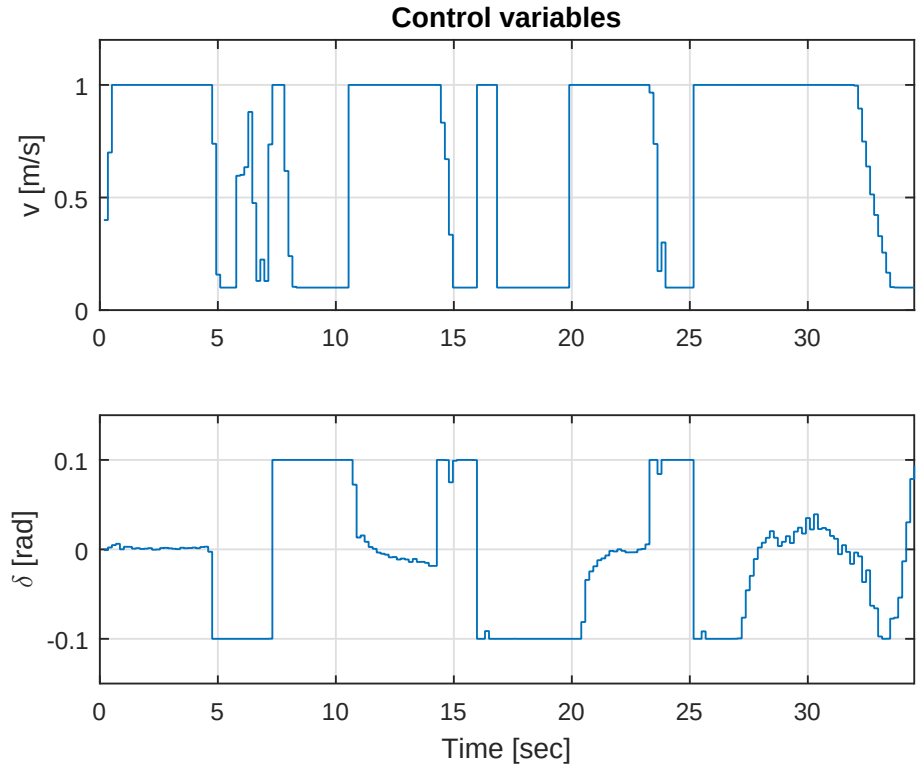
Case 4 is also an obstacle avoidance problem with a fixed end position in the XY-plane but now the selected sampling time is 0.17 s. Fig. 6.9a shows the mobile robot trajectory with three obstacles along the way in the XY-plane, Fig. 6.9b shows the control variables along the trajectory and Fig. 6.9c shows the computation time that the solution of the optimal control problem needs to be solved.

Case 4	
Type of problem	Obstacle avoidance
Reference	$(x_d, y_d) = (20, 0)$
Sampling time	0.17 s

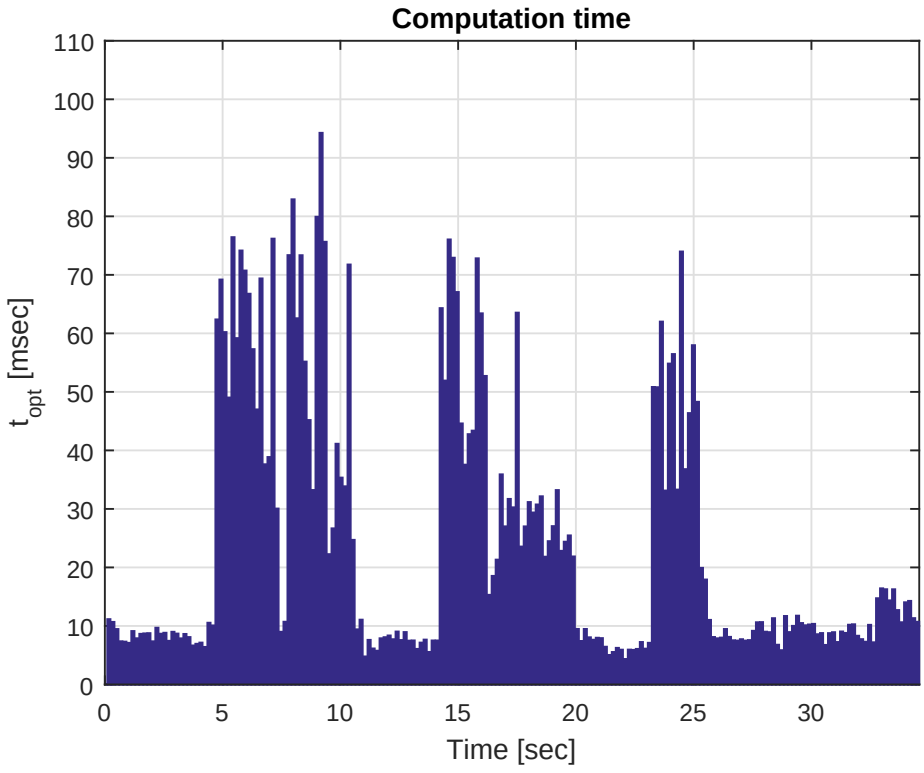
Table 6.6: Considerations for case 4



(a) Robot trajectory in the XY-plane



(b) Control variables



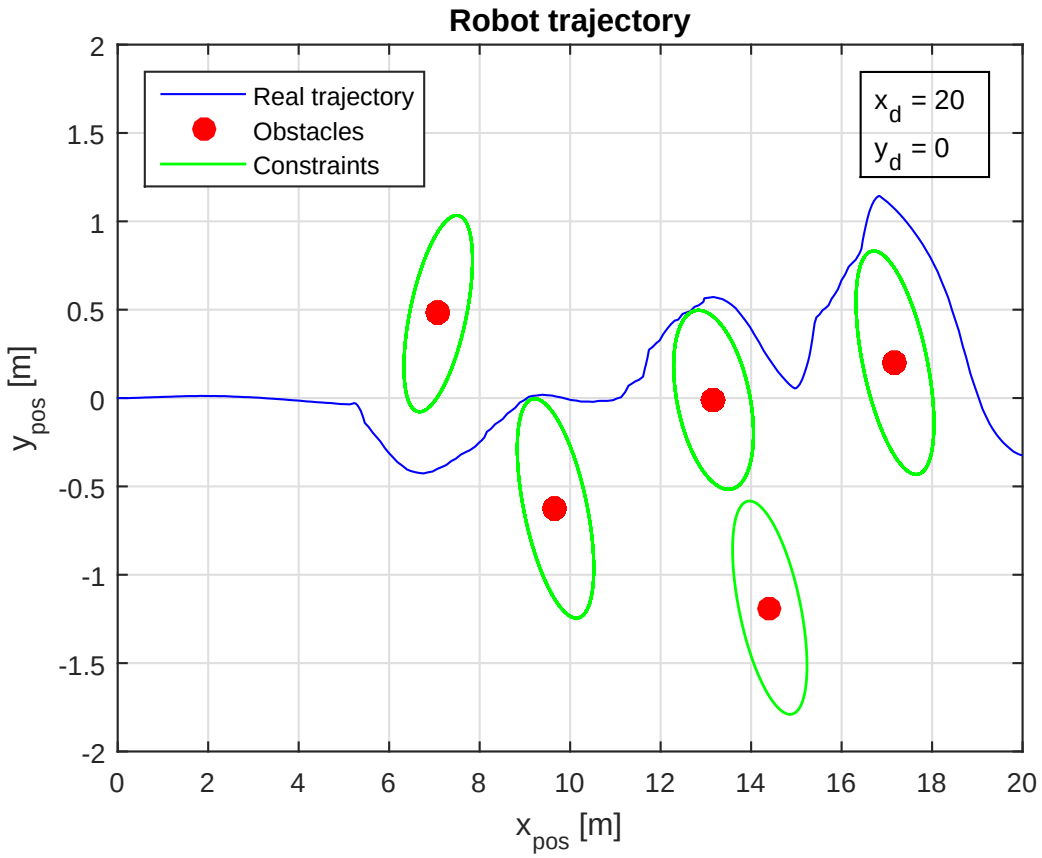
(c) Computation time

Figure 6.9: Results for case 4. Obstacle avoidance with $t_s = 0.17$ s

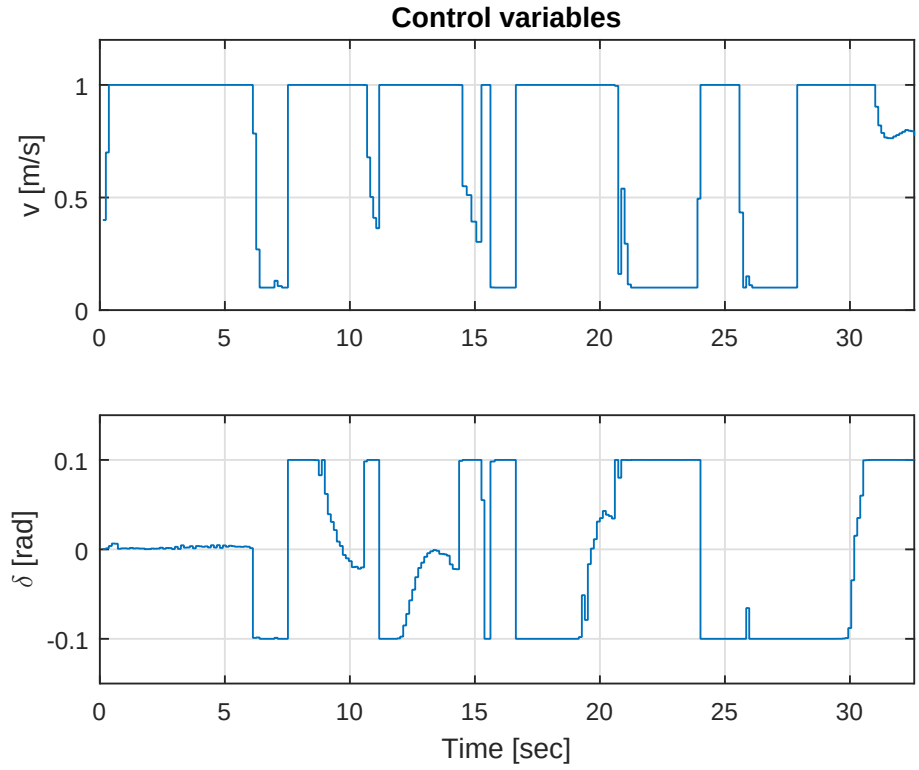
Case 5 is an obstacle avoidance problem with a fixed end position in the XY-plane and a selected sampling time of 0.12 s. Fig. 6.10a shows the mobile robot trajectory with five obstacles along the way in the XY-plane, Fig. 6.10b shows the control variables along the trajectory and Fig. 6.10c shows the computation time that the solution of the optimal control problem needs to be solved.

Case 5	
Type of problem	Obstacle avoidance
Reference	$(x_d, y_d) = (20, 0)$
Sampling time	0.12 s

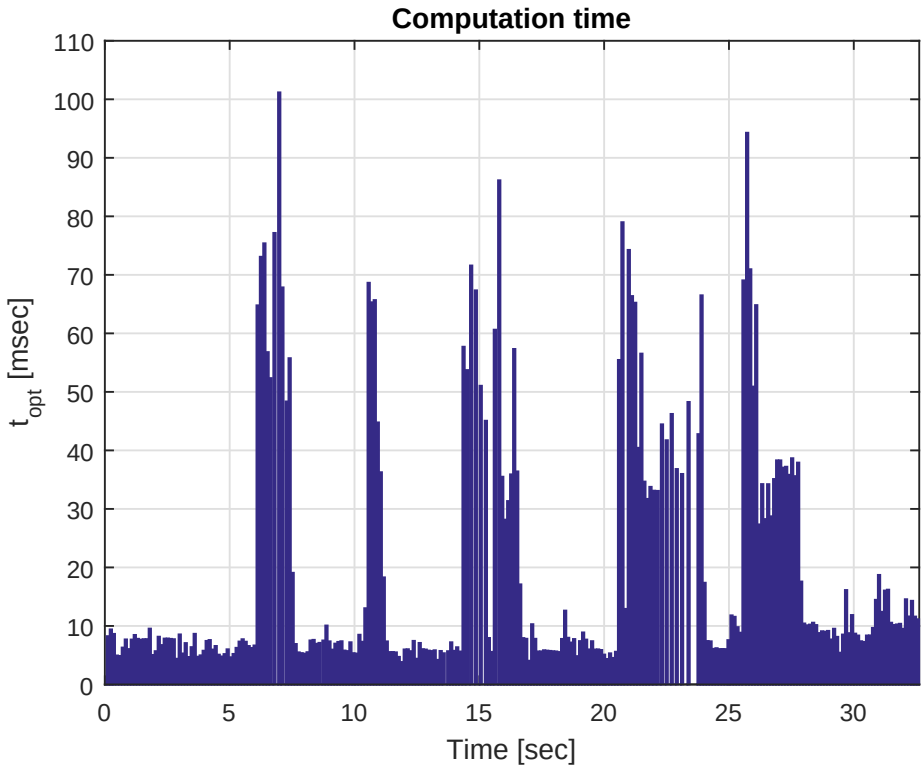
Table 6.7: Considerations for case 5



(a) Robot trajectory in the XY-plane



(b) Control variables



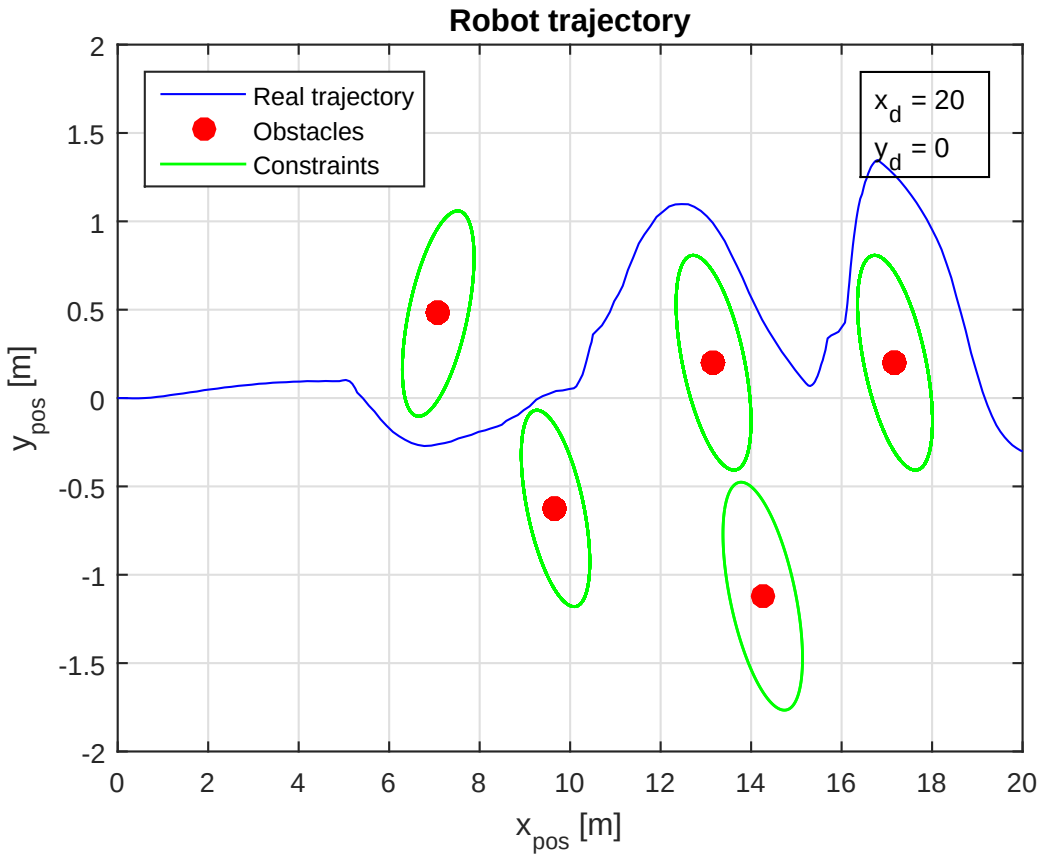
(c) Computation time

Figure 6.10: Results for case 5. Obstacle avoidance with $t_s = 0.12$ s

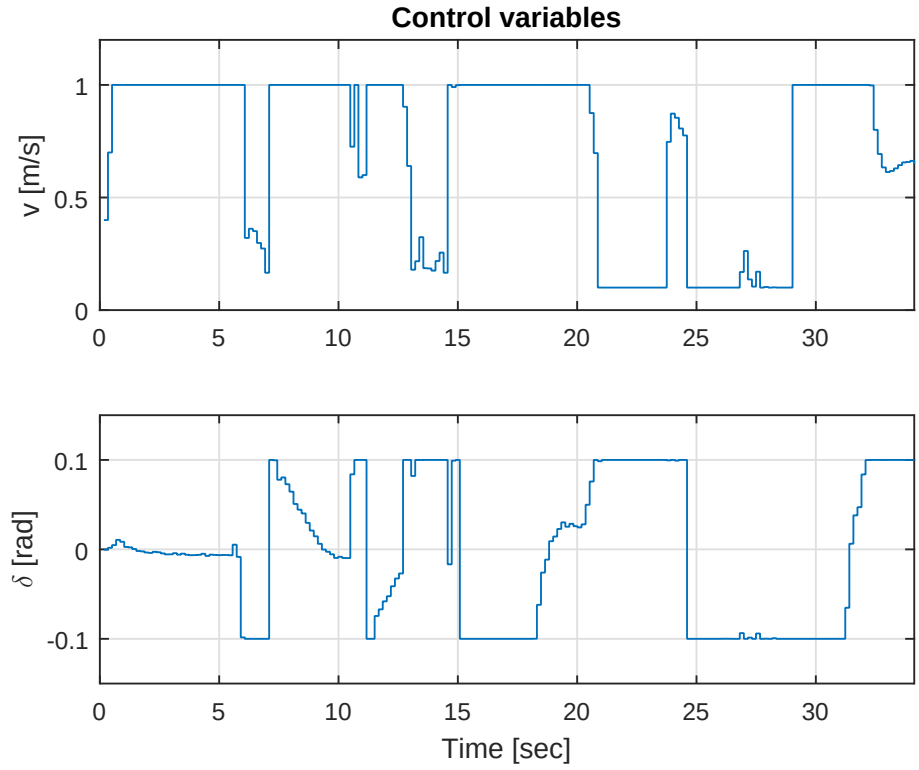
Case 6 is also an obstacle avoidance problem with a fixed end position in the XY-plane but now the selected sampling time is 0.17 s. Fig. 6.11a shows the mobile robot trajectory with five obstacles along the way in the XY-plane, Fig. 6.11b shows the control variables along the trajectory and Fig. 6.11c shows the computation time that the solution of the optimal control problem needs to be solved.

Case 6	
Type of problem	Obstacle avoidance
Reference	$(x_d, y_d) = (20, 0)$
Sampling time	0.17 s

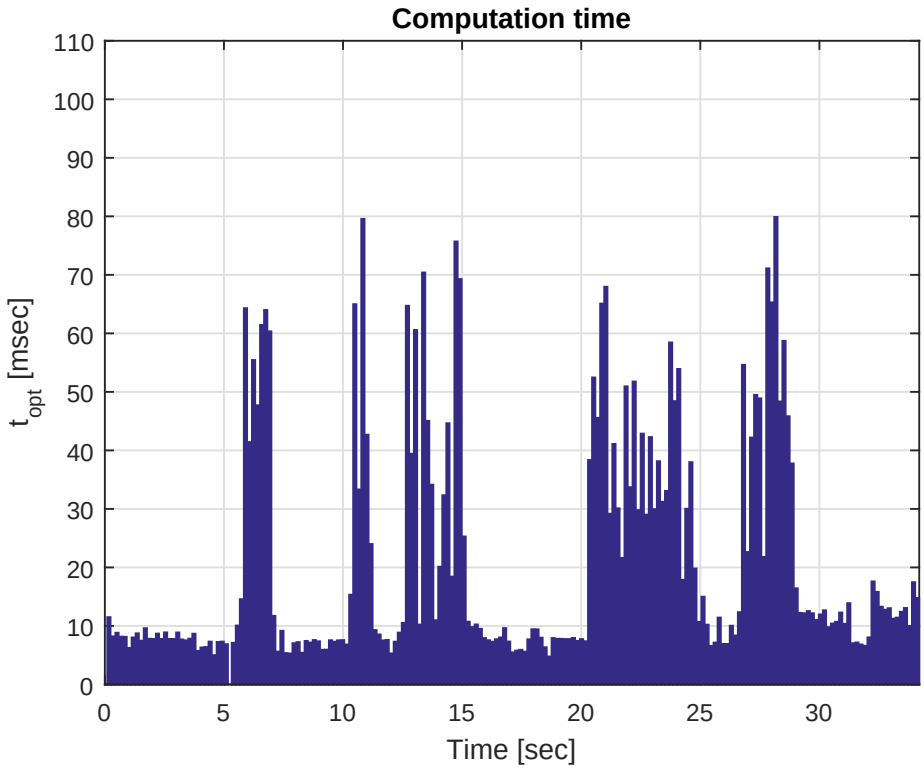
Table 6.8: Considerations for case 6



(a) Robot trajectory in the XY-plane



(b) Control variables



(c) Computation time

Figure 6.11: Results for case 6. Obstacle avoidance with $t_s = 0.17$ s

6.5 Evaluation of the results

For the trajectory tracking problem, there are case 1 and 2, where sampling times of 0.12 s and 0.17 s are used, respectively. In Fig. 6.6a one can observe that for the lower sampling time (120 ms) the mobile robot follows very well the reference trajectory (sine function). In the first 2 m, the mobile robot tries to follow the trajectory and because of the fast change of the values in the Y-direction, it fails. Then the mobile robot is capable of reaching the trajectory and begins to follow it closely until the final position in the X-direction is reached (15 m). The maximum value of t_{opt} for case 1 is approximately 18 ms (shown in Fig. 6.6c), and it is lower than the sampling time (120 ms). On the other hand, for the greater sampling time (170 ms) the mobile robot follows not as well as in case 1. In Fig. 6.7a in the first mostly 4 m, the mobile robot tries to follow the trajectory and because of the fast change of the values in the Y-direction, it fails. Then the mobile robot is capable of reaching the trajectory but with more striking deviations due to the longer sampling time. The maximum value of t_{opt} (shown in Fig. 6.7c) is approximately 16 ms, and it is much lower than the sampling time (170 ms). It can be noticed that for both cases there are some little deviations between the real and reference trajectories due to the mechanical construction and the dynamics of the mobile robot.

Tab. 6.9 shows the comparison between the two cases, where $t_{opt,max}$ is the maximum value of the computation time and they are almost equal, but the great difference is that in case 1, the mobile robot starts following the reference trajectory earlier than in case 2, i.e. having less sampling time for trajectories problem is more accurate.

Results	t_s	$t_{opt,max}$	
Case 1	120 ms	18 ms	(started at 2 m)
Case 2	170 ms	16 ms	(started at 4 m)

Table 6.9: Results for trajectory tracking problem

For the obstacle avoidance problems, in first scenario (three obstacles) there are case 3 and 4, where sampling times of 0.12 s and 0.17 s are used, respectively. In Fig. 6.8a one can observe that for the lower sampling time (120 ms) the mobile robot goes from the initial position to the final desired position avoiding all the obstacles without problem. In Fig. 6.8c one can observe that there are three peaks of computation time at 5 s, 12 s and 20 s approximately, each one corresponds to the detection and consideration of

an obstacle. The maximum value of t_{opt} is approximately 100 ms, and it is lower than the sampling time (120 ms). On the other hand, in Fig. 6.9a for the greater sampling time (170 ms) the mobile robot also avoids all the obstacles along the way, but in comparison with case 3, when the mobile robot reaches the first and second obstacle it moves further in the Y-direction and passes very close to the third obstacle. For this case looking at the computation time (shown in Fig. 6.9c) one can observe that there are three peaks of computation time at 5 s, 14 s and 22 s, each one corresponds to the detection and consideration of an obstacle. The maximum value of t_{opt} is approximately 94 ms, and it is lower than the sampling time (170 ms).

Tab. 6.10 shows the comparison between both cases where $t_{opt,max}$ is the maximum peak value of computation time for each detected obstacle. It can also be seen that the finally reached position is shown. Then, because of the values of computation time and the final position of the mobile robot, case 4 is more accurate than case 3.

Results	t_s	$t_{opt,max,1}$	$t_{opt,max,2}$	$t_{opt,max,3}$	$x_{pos,N}$	$y_{pos,N}$
Case 3	120 ms	100 ms	75 ms	78 ms	20 m	0.04 m
Case 4	170 ms	94 ms	76 ms	73 ms	20 m	0.01 m

Table 6.10: Results for obstacle avoiding problem (3 obstacles)

Now for the second scenario (five obstacles) there are case 5 and 6, where sampling times of 0.12 s and 0.17 s are used, respectively. In Fig. 6.10a one can observe that for the lower sampling time (120 ms) the mobile robot goes from the initial position to the final desired position avoiding all the obstacles without problem but when it reaches the first and second obstacle it moves further in the Y-direction and passes very close to the third obstacle (too close to the constrained area but without collision). In Fig. 6.10c one can observe that there are five peaks of computation time at 6 s, 10.5 s, 14.8 s, 20.3 s and 25.2 s approximately, each one corresponds to the detection and consideration of an obstacle. The maximum value of t_{opt} is approximately 101 ms, and it is lower than the sampling time (120 ms). On the other hand in Fig. 6.11a for the greater sampling time (170 ms) the mobile robot also avoids all the obstacles along the way but when it reaches the third and fifth obstacle it moves further in the Y-direction. In Fig. 6.11c there are five peaks of computation times at 5.2 s, 10.1 s, 12.5 s, 20.2 s and 27 s approximately, each one corresponds to the detection and consideration of an obstacle. The maximum value of t_{opt} is approximately 80 ms, and it is lower than the

sampling time (170 ms).

Tab. 6.11 shows the comparison between both cases where $t_{opt,max}$ is the maximum peak value of computation time for each detected obstacle. It can also be seen that the final reached position is shown. Then, because of the values of computation time and the final position of the mobile robot, case 6 is more accurate than case 5.

Results	t_s	$t_{opt,max,1}$	$t_{opt,max,2}$	$t_{opt,max,3}$	$t_{opt,max,4}$	$t_{opt,max,5}$	$x_{pos,N}$	$y_{pos,N}$
Case 5	120 ms	101 ms	69 ms	87 ms	79 ms	93 ms	20 m	0.38 m
Case 6	170 ms	64 ms	79 ms	75 ms	69 ms	80 ms	20 m	0.29 m

Table 6.11: Results for obstacle avoiding problem (3 obstacles)

In conclusion, one can see that from the results and comparison of all the cases, for the trajectory tracking problem the better results were obtained by using a lower sampling time (120 ms), while for the obstacle avoidance problem the better results were obtained (for both scenarios) by using a greater sampling time (170 ms).

7 Summary, conclusions and outlook

7.1 Summary

In this master thesis the main focus lies on the involvement of camera data for obstacle detection and description as well as the fusion with the laser scanner data to aim at a better and more reliable description of the obstacles to be avoided within an autonomous driving using model predictive control.

First the technical specifications of the mobile robot SUMMIT main devices such as the 2D mono-camera and the laser scanner were presented. Then a 3-state mathematical model was presented, compared and validated with the 5-state model developed in (Thieme, 2014) considering some limitations like the velocity maximum value of 1 m/s. After this a correction of the control variables is explained based on (Belgradskaia, 2016) because the transmitted value of the control variables are not the same as the one received by the mobile robot due to its mechanical construction and dynamics. The velocity correction has a linear relationship, while the steering angle correction has a nonlinear relationship (solved with neural networks). Then the estimation of the state variables are explained using the extended Kalman filter. It was observed that after the correction of the control variables and the estimation of the states the mobile robot behaved very well.

Later, computer vision was explained and applied where one had the first stage that was the detection process: shape detection algorithm (using Canny edge detection) and color detection algorithm (using Hue-Saturation-Values). Then, the cluster classification process based on the Euclidean distance between pixels was performed in the second stage. Subsequently the data acquisition by the laser scanner and the 2D camera, was explained. The estimation of the obstacle dimensions and the distance were founded based on the pinhole camera model. Furthermore a sensor fusion technique for obstacle determination was proposed demonstrating three important parameters: ellipse inclination angle, ellipse absolute position and its dimensions.

Finally a continuous nonlinear optimal control problem inlcuding cost functional, state equations, and constraints was presented. Then a transformation into a nonlinear optimization problem was necessarily explained in order to solve such a complex problem. The multiple-shooting with collocation method was illustrated and so the model predictive control principle in order to solve two types of classical problems: trajectory tracking and obstacle avoidance problems, where very good results were obtained considering two different sampling times ($t_s = 0.12\ s$ and $t_s = 0.17\ s$).

7.2 Conclusions

Since the 3-state mobile robot's mathematical model was validated in simulation environment by comparing its behavior with the 5-state model, this 3-state mathematical model is an accurate model for the mobile robot when its maximum speed is limited to 1 m/s, i.e. using a greater speed may lead to less accurate results. The correction of the control variables depend on the dynamics of the mobile robot and its mechanical construction. It is important to remark that these corrections were done based on certain conditions of the mobile robot like the usage, working terrain, mechanical structure, etc. The estimation of the state variables with the extended Kalman Filter may affect the results if the covariance values of the matrix Q are modified, i.e. the most significant weight is the one that belongs to the yaw angle.

Referring to computer vision, the shape and color detection algorithms have their advantages and disadvantages, depending on the case one should choose the best option for obstacle determination and identification. On one hand is better to use the shape detection algorithm to detect different kind of obstacles or objects, but it fails due to lighting (not usable for very dark environment or when there are shadows). In order to improve the obstacle identification there are two significant variables, the first one is the kernel matrix (since it is constructed for image filtering) and the second one is the threshold and kernel size of the Canny edge detector. In this work the thresholds were calculated dynamically based on the image histogram. On the other hand is better to use the color detection algorithm to detect obstacles and objects more accurate (immune to lightning), but previously it is needed the information about the color. The most significant variables in this algorithm are the HSV thresholds value that depends exclusively on the color characteristic. In the cluster classification process is important to remark the importance of the predefined Euclidean distance and it depends directly on the camera resolution.

The obstacle position and dimensions are calculated with a sensor fusion algorithm based on a data combination from the laser scanner and the 2D camera. Both the position and dimensions depend on the laser scanner angular resolution, the characteristic of the terrain (in this master thesis it is assumed a flat terrain for practical purposes), the camera resolution, focal length estimation and external disturbances e.g. the weather, obstacles material, etc. Finally in the dynamic optimal control problem, specifically in the trajectory tracking problem, the accuracy of the results and the computing time depend on the sampling time, the weighting values of the cost function and the reference trajectory. Besides in the obstacle avoidance problem, the accuracy of the results and the computing time depend on the sampling time, the weighting values of the cost function, number of obstacles and used obstacle detection algorithm (color or shape). For both cases all the previously mentioned variables influence in the results obtained in this master thesis.

7.3 Future work

Since the 3-state mathematical model is reduced (considering a maximum velocity of 1 m/s), the robot behavior is not described exactly. Therefore in order to achieve a further improvement of the position determination, more devices such as GPS (Global Positioning System) receiver, IMU (inertial measuring unit) or infrared cameras can be added.

Talking about computer vision, two detection algorithms (shape and color) were explained and tested. It is important to consider a robust detection algorithm for obstacles determination (lighting or darkness immunity). Although in this master thesis, a mono-camera was used and the distance between the detected obstacle and the camera was estimated, one can use a stereo-camera or two mono-cameras in order to get the depth information, and the correct obstacles dimension. The first one can help to construct a 3D environment based on laser scanner information and camera depth data, while the second one can estimate more precisely the obstacle dimensions by using a triangulation technique.

Finally in trajectories tracking problems, one can use a lane detection algorithm for tracking, i.e. using the lane information as a path constraint in the optimization problem. One can also consider to detect moving obstacles along the desired trajectory.

Bibliography

Al-Jarrah, M. A. (2016). Developing 3d Model for Mobile Robot Environment Using Mono-Vision System. *7th International Conference on Computer Science and Information Technology (CSIT). Amman-Jordan.*

Belgradskaia, E. (2016). *Verbesserte positionsbestimmung mittels Kalman Filter und neuronalem Netz bei der modellprädiktiven Regelung eines mobilen Roboters* (Master thesis). TU Ilmenau, Germany.

Correa, M. (2016). *High Performance Implementation of MPC Schemes for Fast Systems* (Master thesis). TU Ilmenau, Germany.

Drozdova, E. (2015). *Hinderniserkennung und -vermeidung durch einen mobilen roboter im Rahmen einer modellprädiktiven Regelung* (Master thesis). TU Ilmenau, Germany.

Du, K.-L., & Swamy, M. (2013). *Neural Networks and Statistical Learning* (Illustrated ed.). Springer Science and Business Media.

Fu, G., Corradi, P., Menciassi, A., & Dario, P. (2011). An Integrated Triangulation Laser Scanner for Obstacle Detection of Miniature Mobile Robots in Indoor Environment. *IEEE/ASME Transactions on Mechatronics, vol. 16, pp. 778-783.*

Geletu, A. (2016). *Systems Optimization* (Lectures collection). TU Ilmenau, Germany.

Greensted, A. (2010). *Otsu Thresholding.* Retrieved 2017-02-13, from `http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html`.

Grewal, M., & Andrews, A. (2001). *Kalman Filtering: Theory and Practice Using MATLAB* (3rd ed.). John Wiley and Sons.

Gruyer, D., Cord, A., & Belaroussi, R. (2013). Vehicle Detection and Tracking by Collaborative Fusion Between Laser Scanner and Camera. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5207-5214. Tokyo, Japan.*

Gurney, K. (2003). *An Introduction to Neural Networks.* CRC Press.

Han, J., Heo, O., Park, M., Kee, S., & Sunwoo, M. (2016). Vehicle distance estimation using a mono-camera for FCW/AEB systems. *International Journal of Automotive Technology, vol. 17, Issue 3, pp. 483–491.*

Haykin, S. (2004). *Kalman Filtering and Neural Networks*. John Wiley and Sons.

Haykin, S. (2011). *Neural Networks and Learning Machines* (3rd ed.). Pearson Education.

Hedges, R., Stoy, K., Bers, J., Douglas, J., Jinsuck, K., & Martignoni, A. (2006). *The Player Project Free software tools for robot and sensor applications.* Retrieved 2017-03-13, from `http://playerstage.sourceforge.net`.

Heikkilä, J., & Silvén, O. (1997). A Four-step Camera Calibration Procedure with Implicit Image Correction. *IEEE Computer Society Conference, pp. 1106-1112. San Juan, Puerto Rico, USA.*

Hussin, R., Juhari, M. R., Kang, N. W., R.C.Ismail, & Kamarudin, A. (2012). Digital Image Processing Techniques for Object Detection From Complex Background Image. *International Symposium on Robotics and Intelligent Sensors (IRIS), Procedia Engineering 41, pp. 340-344. Kuching, Sarawak, Malaysia.*

Kaehler, A., & Bradski, G. (2016). *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library.* O'Reilly Media, Inc.

Lazutkin, E., Geletu, A., Hopfgarten, S., & Li, P. (2014). Modified Multiple Shooting Combined with Collocation Method in JModelica.org with Symbolic Calculations. *10th International Modelica Conference, pp. 999-1006. Lund, Sweden.*

Lee, T.-J., Yi, D.-H., & Cho, D.-I. D. (2016). A Monocular Vision Sensor-Based Obstacle Detection Algorithm for Autonomous Robots. *Sensors 2016, vol. 16, pp. 311-330*.

Li, P. (2017). *Dynamische Prozessoptimierung* (Foliensammlung zur Vorlesung). TU Ilmenau, Germany.

Mahajan, S., Bhosale, R., & Kulkarni, P. (2013). Obstacle Detection Using Mono Vision Camera and Laser Scanner. *International Journal of Research in Engineering and Technology, vol. 2, pp. 684-690*.

Mitchell, H. (2007). *Multi-sensor Data Fusion: An Introduction* (Illustrated ed.). Springer Science and Business Media.

Müller, V. (2013). *Modellvalidierung und modell-prädiktive Regelung an einem mobilen Roboter* (Master thesis). TU Ilmenau, Germany.

Mokrzycki, W., & Samko, M. (2012). Canny Edge Detection Algorithm Modification.

International Conference on Computer Vision and Graphics, vol. 7594, pp. 533-540. Warsaw, Poland.

Mrovlje, J., & Vrancic, D. (2008). Distance measuring based on stereoscopic pictures. *9th International PhD Workshop on Systems and Control: Young Generation Viewpoint. Izola, Slovenia.*

OpenCV Library. (2017). Retrieved 2016-10-03, from `http://opencv.org/`.

Park, J. K., Kim, T. H., & Park, T. H. (2015). Autonomous Navigation System for a Mobile Robot Using a Laser Scanner in a Corridor Environment. *IEEE/SICE International Symposium on System Integration (SII), pp. 512-516. Nagoya, Japan.*

Park, K.-Y., & Hwang, S.-Y. (2014). Robust Range Estimation with a Monocular Camera for Vision-Based Forward Collision Warning System. *The Scientific World Journal, vol. 2014. 9 pages.*

Robotnik Automation, S.L.L. (2012a). *Summit mobile robot: System Architecture Manual.* (Version 3.0)

Robotnik Automation, S.L.L. (2012b). *Summit mobile robot: System Elements and Maintenance Manual.* (Version 3.0)

Robotnik Automation, S.L.L. (2012c). *Summit mobile robot: System Installation and Configuration Manual.* (Version 3.0)

Simon, D. (2006). *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches.* John Wiley and Sons.

Stein, G. P., Mano, O., & Shashua, A. (2003). Vision-based ACC with a Single Camera: Bounds on Range and Range Rate Accuracy. *IEEE Intelligent Vehicles Symposium. Columbus, OH, USA.*

Thieme, A. (2014). *Effizienzsteigerung modell-prädiktiver Regelungen und Anwendung auf einen mobilen Roboter* (Master thesis). TU Ilmenau, Germany.

T.N.R.Kumar. (2015). A Real Time Approach for Indian Road Analysis using Image Processing and Computer Vision. *IOSR Journal of Computer Engineering (IOSR-JCE), vol. 17, issue 4, pp. 1-10.*

Wang, T.-H., Lu, M.-C., Wang, W.-Y., & Tsai, C.-Y. (2007). Distance Measurement Using Single Non-metric CCD Camera. *7th WSEAS International Conference*

on Signal Processing, Computational Geometry and Artificial Vision, pp. 1-6. Athens, Greece.

Wächter, A., & Biegler, L. (2006). *Introduction to ipopt - a tutorial for downloading, installing, and using ipopt.* Retrieved 2016-11-20, from `http://www.coin-or.org/Ipopt/documentation`.

Zarchan, P., & Musoff, H. (2000). *Fundamentals of Kalman Filtering: A Practical Approach* (Illustrated ed.). American Institute of Aeronautics and Astronautics.

Zhang, Z. (2000). A Flexible New Technique for Camera Calibration. *IEEE Transactions on pattern analysis and machine intelligence, vol. 22, issue 11, pp. 1330-1334*.