POLITECNICO DI TORINO
Aerospace Engineering

Master Degree Thesis

Identification Methods and Simulation Modeling of a small UGV for Indoor Applications

Supervisors
Prof. Elisa Capello
Dr. Davide Carminati
Dr. Iris David Du Mutel

Candidate
Enza Incoronata Trombetta
m.256845

April 2021

To my family,
Mum, Dad and Ni
To my grandparents,
Andrea and Antonietta

Acknowledgments

The most sincere thanks are dedicated to Professor Elisa Capello for having carefully followed the development of this thesis and for the opportunity to explore such an interesting topic. I would like to thank Dr. Matteo Scanavino for helping me in the first steps of this work and Dr. Davide Carminati and Dr. Iris David Du Mutel de Pierrepont Franzetti who, with patience and kindness, guided and supported me from the beginning to the end. Their advice and knowledge have been useful and valuable to me.
I would like to thank my life companions Sabina and Fede for always walking by my side and my special motivator Marco. Our discussions on mathematical issues were an important starting point for reflection.
I would also like to thank my "Northern" family Malvina, Greta, Alexandra, Valeria, Amina and Ilaria. Our breakfasts and snacks together gave me the energy to face the difficulties encountered.
I cannot fail to mention also my colleagues-friends thanks to whom the years of University have been a wonderful adventure, and my companions of special dinners and evenings: Martina, Riccardo, Alessia, Perry, Peppe, Elisa, Daniel, Fabio, Fabiano, Federico, Mario, Fabio, Menni, Edo, Martina, Giulio, Miki.

Abstract

The last few years have seen a growing interest in the development of intelligent machines capable of moving autonomously in space and being aware of their surroundings. Their great potential makes them ideal for the most varied fields of application: agriculture, manufacturing, land and aerial surveillance, naval operations, commercial transport, space exploration.
A focus on mobile robots can be observed, since they are systems able to integrate technologies related to sensing, information processing, movement on wheels, obstacle avoidance technique and Artificial Intelligence. Their versatility makes them the test bench suitable not only for testing high level solutions of Guidance, Control and Navigation techniques (GNC), but also at a strictly mechanical level with the possibility of testing configurations that include robotic arms, payloads for environmental analysis, hybrid motion mechanisms.
In a space framework, *rovers* can be considered as evolution of these mobile robots: these robotic systems for the exploration of planetary surfaces are designed to move independently in unknown environments, to conduct analysis on the ground that can identify composition and properties, to photograph the surrounding environment and allow its study on Earth. Given the cost associated with space operations and the rare possibility of repair, the reliability of such systems must be of the highest level. It is therefore clear the importance of testing their effectiveness in similar situations to those in which they will operate on the designed mission. The tests are possible thanks to numerical simulations.
This thesis is addressed to the system identification of a mobile robot to obtain a model that can be used in a simulation environment. The advantage offered by simulation is linked to the possibility of carrying out tests that do not damage the real device and do not require its continuous use. *Devastator* robotic platform, a tracked mobile robot controlled by an on-board companion computer, is the reference used for identification and experimental testing.
The analysis of the DC motors behavior is first performed with both a lumped parameter approach and a data-driven approach. A comparison between the two methodologies is carried out. A kinematic model is then adopted for the reproduction of the robot motion, but the identification of the inertial parameters of the system has also been performed as a basis for future development and integration of a dynamic model.
The thus obtained model is recreated in MATLAB/Simulink environment. A controller and a path planning algorithm are also implemented to test and validate the behavior of the plant. The selected control strategy is based on a Proportional–Integral–Derivative or PID controller while a simple but effective Artificial Potential Field strategy is adopted for path planning. Their performances are subsequently established through Simulink-ROS/Gazebo

co-simulation; Gazebo is in fact a simulation tool for algorithms, configurations and control strategies testing in a realistic scenario.
Finally, they are translated into executable code supported by on-board systems thanks to the MATLAB Code Generation functions and tested in indoor applications.

Contents

List of Figures

List of Tables

List of Acronyms

AIC	Akaike's Information Criterion
ALFUS	Autonomy Levels for Unmanned Systems
APF	Artificial Potential Field
ARMAX	Auto-Regressive and Moving Average with eXogenous input
ARX	Auto-Regressive with eXogenous input
ASTM	American Society for Testing and Materials International
ATT	Advanced Teleoperator Technology
BIC	Bayesian Information Criterion
BJ	Box-Jenkins
CoM	Center of Mass
DARPA	Defense Advanced Research Projects Agency
EKF	Extended Kalman Filter
FMKF	Finite-Model Kalman Filter
FPE	Final Prediction Error
RAM	Random Access Memory
ROM	Read Only Memory
GATERS	Ground/Air TeleRobotic System
GNC	Guidance, Navigation and Control
GND	Ground
GPS	Global Positioning System
GSR	Ground Surveillance Robot
HMI	Human Machine Interface

HTML Hypertext Markup Language

IAPF Improved Artificial Potential Field

IMU Inertial Measurement Unit

IR Infrared

LED Light Emitting Diode

LiPo Lithium Polymer

LS Least Squares

LTI Linear Time Invariant

MAV Micro Air Vehicle

MDL Minimum Description Length Criterion

MIMO Multi-Input Multi-Output

ML Maximum likelihood

MISO Multi-Input Sigle-Output

MMSE Minimum Mean Square Error

NARMAX Nonlinear Auto-Regressive and Moving Average with eXogenous input

NARX Nonlinear Auto-Regressive with eXogenous input

NASA National Aeronautics and Space Administration

NCAGV Network-Centric Autonomous Grounf Vehicle

ODE Open Dynamics Engine

OE Output Error

OGRE Open-source Graphics Rendering Engine

PCAGV Platform-Centric Autonomous Ground Vehicle

PEM Prediction Error Methods

PID Proportional Integral Derivative

PWM Pulse With Modulation

ROS Robotic Operating System

SISO Single-Input Single-Output

SLAM Simultaneous Localization And Mapping

TGV	TeleOperated UGV
TOV	TeleOperated Vehicle
TUGV	Gladiator Tactical Unmanned Ground Vehicle
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
UGV JPO	Unmanned Ground Vehicles Joint Program Office
UKF	Unscented Kalman Filter
URDF	Unified Robot Description Format
URI	Uniform Resource Identifier
UUV	Unmanned Underwater Vehicle
UV	Unmanned Vehicle
VCS	Version Control System
XML	eXtensible Markup Language

Chapter 1

Introduction

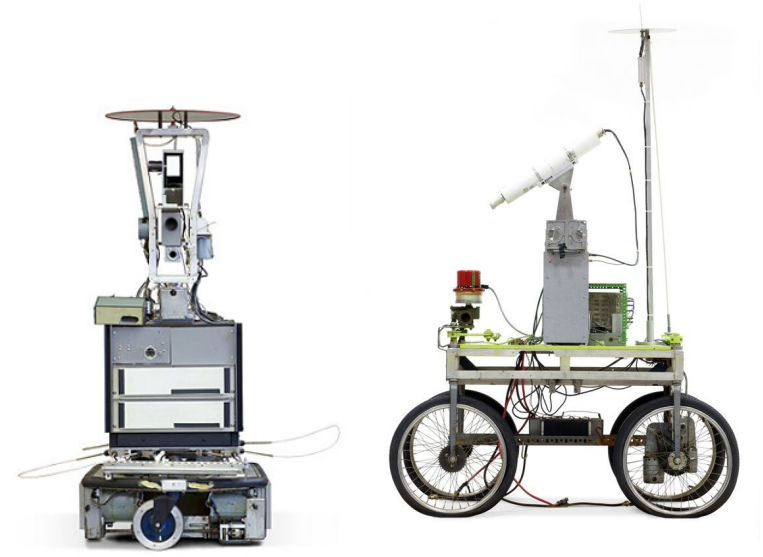
UVs or Unmanned Vehicles: Any vehicle which operates without a human in direct physical contact with that vehicle [34]

What in the past would have aroused amazement and unbelief is now a consolidated reality: the existence of smart machines capable of operating without direct human intervention. Characterized by different degrees of autonomy in navigation and in the perception of the surrounding environment, these machines are the result of years of technological developments and scientific innovations that have reached such a maturity as to allow the absence of direct human control on board. Despite a history strongly linked to the military environment, the use of these systems in the civilian field has grown in recent years making UVs the answer for the most varied needs, from security to intervention accuracy, from agriculture to commercial transport, from the exploration of remote areas not otherwise accessible to monitoring and surveillance. The growing availability of powerful low-cost hardware platforms and the possibility of exploiting increasingly "intelligent" and autonomous forms of control has led to the growth of interest in UVs and paved the way for the exploration of all kinds of possibilities, from land mobility (*Unmanned Ground Vehicles* or UGVs) to marine (*Unmanned Underwater Vehicles* or UUVs), up to the aerial one (*Unmanned Aerial Vehicles* or UAVs).
Although the spotlight nowadays is on the fascinating world of UAVs, and this is not surprising given the opportunities offered by a "top view" of things, another category deserves as much attention: the world of small UGVs.

1.1 Historical background

Sometimes also called *mobile robots*, UGVs are distinguished by their ability to move on a surface and, if necessary, to transport a payload. Their versatility is such to have made them an excellent development platform for testing algorithms related to path planning, obstacle avoidance, control strategies, sensor testing, Artificial Intelligence, making them over time more and more autonomous.
The concept of "autonomous vehicle" is not exactly current: as far back as 400 BC, Greeks and Romans speculated on steam-powered robot birds and used unmanned fire ships as

weapons to strike enemy fleets [26]. In 1849 the Austrians used unmanned balloons to attack the city of Venice, loading them with explosives, while 20 years later, in the American Civil War, balloons were equipped with cameras to observe enemy positions from above. Intelligent self-propelled missiles with on board guidance and control were developed during the World Wars by Americans and Germans, such as the Flying Bombs of the former and the V-1 missiles of the latter, improving and establishing the use of UVs in warfare. While a considerable boost was given to the field of UAV technology, not as much luck seems to have had the UGV sphere. The reason identified is essentially one: the complexity of the operating and navigation environment with respect to that of their aerial analogues ([26] and [34]). The first UGVs were tanks loaded with explosives capable of approaching close to enemy fortifications and detonated remotely. The first examples of autonomous UVs were instead developed around 1948 in England with vehicles capable to respond to contact with objects and able to perceive the light.
One of the first examples of a mobile robot with a notable impact was *Shakey*, a general-purpose test-bed developed in the Artificial Intelligent Center at the Stanford Research Institute in the 1960s, capable of perceive and reason about its surroundings (Figure 1.1a). It was propelled by two stepping motors independently driving a wheel on each side of the robot and equipped with a TV camera and ultrasonic range finder in a movable "head". Touch sensors captured information from the environment and sent them to a mainframe computer over special radio links for the elaboration of navigation and exploration tasks. It was not yet an autonomous robot in the sense of completely independent and self-governed, i.e. capable of processing information from sensors on board, but it is the first notable example of an attempt to integrate the functionality and performance of a mobile robot with control algorithms even if performed by an external digital computer [69]. In the 1980s



(a) *Shakey the Robot [91]* (b) *Stanford Cart [100]*

Figure 1.1: Notable mobile robots

at the Stanford University AI Lab it was developed the *Stanford Cart*, a project born to exploring navigation and obstacle avoidance tasks (Figure 1.1b). Through the use of a TV camera mounted on the top of the head, the robot took pictures from several different angles and sent them to a main computer to compute the distance between itself and the obstacles and to obtain instructions from the computer in order to avoid the impact. Although the system took up to 15 minutes to make each one-meter move, it was successful in the planning of an obstacle-free path to the destination and formed the basis for future developments in this field. Also in the 1980s, the *Autonomous Land Vehicle* was developed as part of DARPA's Strategic Computing Program, an example of a mobile robot that, through the processing of video and distance data, was able to generate a model of the scene in front of it and use it for navigation. In those years, however, the attention on UGVs did not grow exclusively from a scientific point of view, but also from a military one. The possibility of using these vehicles to penetrate enemy lines without endangering human lives and exploit them for surveillance, reconnaissance and target acquisition tasks, favored the emergence of programs such as Ground Surveillance Robot (GSR) and the Advanced Teleoperator Technology (ATT) whose successes led to the birth of the Ground/Air TeleRobotic Systems (GATERS) Program in 1985 and the development of a TeleOperated vehicle (TOV) to test the use of UGV in the military field. TOV was a remotely piloted vehicle controlled by a human operator in a Control Station. Visual and audio devices provided information about what surrounds the vehicle and hand and foot controls managed to the human operator constituted the means through which control the robot. Data were exchanged via optical fiber to ensure fast and robust communication even when not in line of sight. About its equipment, cameras, laser rangefinders and sensors capable of detecting chemicals, as well as a 50-caliber machine gun, were installed [107]. However, the development of such programs came to a halt in 1987 when Congressional direction prohibited the use of weapons systems on robot: all the projects were re-targeted to other missions.
The development of UGV continued with the birth of the Unmanned Ground Vehicles Joint Program Office (UGV JPO), an attempt by the Department of Defense (DoD)[1] to standardize and consolidate robotic projects into a single program, and between 1990 and 1996 with the DEMO programs by DARPA, that demonstrated the adaptability of UVs in real missions with supervised autonomous collaboration between multiple units.
Many other military examples could be cited and their number is not surprising, given the success of such technologies in all those situations that lead to the risk of losing human lives, in particular for the detection of explosives, chemical, biological or radioactive agents, for the breaking into enemy territory for exploration, surveillance. Alongside this, however, there is a group of research programs created for rescue and first aid, for underwater exploration, for space exploration, for agriculture aims, born thanks to the favorable conditions offered by the lowering of machine prices and the increase in public interest in this research. Precisely, as reported in [34], this transition from military to civilian use was dictated by factors such as

- Inter-agency transfers, since UV hardware prototypes were no longer extremely expensive but mainstream military vehicles with the possibility to be "lent" for civilian

[1] DoD is an executive branch department of the federal USA government

purposes.

- Market competition, such that the price of components has dropped to the point that they were also accessible to civil entities, not just a few military organizations.

- Public research support.

- Increase in hardware power and possibility of using "off the shelf" components.

Furthermore, among what has certainly contributed to the development of such technologies were the enormous advances in the field of robotics. In 2007 at Stanford University the idea of a single framework that could manage the different aspects of the discipline was developed: the Robot Operating System. ROS is an open source tool organized with libraries and packages to support the development of robot applications, providing hardware abstraction, visualization tools, communication and message exchange between processes, information management etc. In 2014, NASA announced the first robot to use ROS in space aboard the International Space Station, and in the following years the attention of tech giants such as Amazon and Microsoft was captured [83]. The ability to manage robotic programming more easily and to use packages and libraries with already available functions to focus on experimentation and innovation was certainly a significant factor in the spread of this tool in the context of mobile robots and UVs programming in general.
Today UVs operate in a wide variety of forms and cost: they range from UAVs large enough to fit in a bag to those comparable to manned aircraft, from small remote-controlled UGVs to tanks of several tons. Just think of the Predator, a Medium Altitude Long Endurance UAV, or the Global Hawk, capable of flying for 30 hours at an altitude exceeding 19000 m [34], and at the same time of the Micro Air Vehicles or MAVs with a flight duration of just 30 minutes [26], UGVs like the Gladiator TUGV tank and small remotely controlled tanks like Packbots (Figure 1.2).

(a) *RQ-4 Global Hawk [87]*

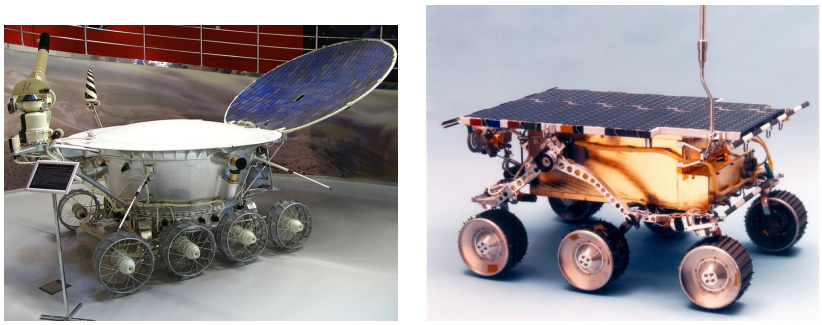
(b) *RQ-16 T-Hawk (MAV) [86]*

(c) *Gladiator Tactical UGV [33]*

Figure 1.2: UVs of today

1.2 Robot and Aerospace

Among others there is a field that has not been purposely investigated in the last section in which UVs have been particularly successful. In the 1970s another organization had placed the focus on the development of unmanned vehicles, attracted to the possibility of drastically reducing the cost of space exploration and the risks associated with human presence on board: the National Aeronautics and Space Administration, also known as NASA.
The knowledge associated with UGVs allowed the development of mobile robots capable of exploring the surface of a planet to collect information useful for its study on Earth called *rovers*. Among the earliest examples of successful rovers was *Lunokhod 1*, a Soviet Union robot that landed on the Moon in 1970 as part of the Lunokhod Program (Figure 1.3a). Equipped with 8 independently controllable wheels, it reached speeds of 100 meters/hour for a weight of 756 kg. It was commanded by a team of five operators on Earth while on board it mounted tools such as solar cells and chemical batteries, an imaging system and countless sensors for soil characterization. It traveled around 10 km collecting more than 20.000 between TV images and high resolution panoramas with various soil analyzes before interrupting communications in September 1971. It was a successful mission: the rover, designed to survive only 3 lunar days (approx. 21 terrestrial days), worked for 11 [93].
The USA proposal for robot space employment came with the Pathfinder Program: it was



(a) *Lunokhod 1 rover [55]* (b) *Sojourner rover [97]*

Figure 1.3: Mobile robots in the Space Exploration

an ambitious mission consisting in the application of a lander and a remotely controlled rover for the exploration of the Mars surface and an attempt to demonstrate the effectiveness of using innovative low-cost technologies in space framework.
The *Sojourner* rover was a 6-wheeled vehicle powered by solar cells plus a non-rechargeable lithium thionyl chloride battery, served as a backup system, equipped with a sensor system capable of acquiring images from the surrounding environment (Figure 1.3b). An internal computer compressed and stored an image on board: black and white images were used in order to study the terrain, the distribution of soil and rock, the properties of the soil, the condition of the machines after the touchdown. Communication with the Earth took place through the Lander: the captured images were transmitted to a ground station which, for its part, monitored the rover and the lander by giving them specific commands. The slowness of this information exchange system marked the decline of teleoperations, in fact

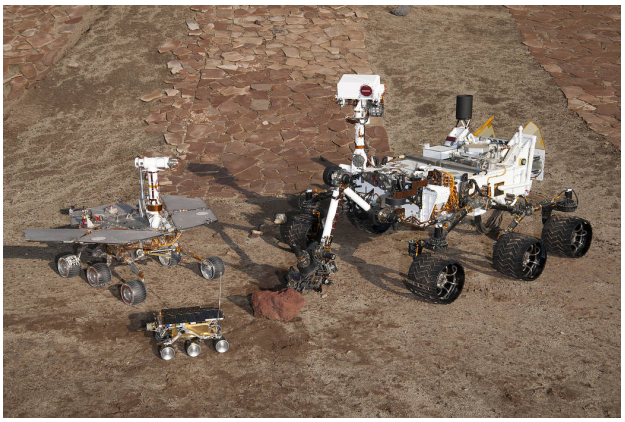


Figure 1.4: Three generations of Mars rovers [60]

impractical over long distances given the need for the ground operations team to continuously monitor the systems. For the Pathfinder mission the control station operators had the necessity to adjust themselves to Mars time, sleeping and waking 37 minutes later each day, for the entire mission [93]. The rover was unable to act autonomously but entered a safe mode in case of anomalous situation and waited for the uplink of the new sequence of commands from Earth. Sojourner's mission ended in September 1997, when communication was lost and never recovered, but it was a success in highlighting the aspects to be improved for future missions [93].
Experience with Sojourner provided, in fact, the basis for the development of subsequent rovers as part of the Mars Exploration Rover, a robotic space mission involving two rovers, *Spirit* and *Opportunity*, began in 2003 with the aim of continuing the analysis of the Mars surface started years earlier. On their example it was realized *Curiosity*, a Mars Science Laboratory, for the collection of martian soil samples and rock cores, their analysis for the research of organic compounds and the exploration of environmental conditions that could have supported microbial life now or in the past (Figure 1.4).
The experience of planetary exploration beyond the Moon has demonstrated the need for UGV systems to improve key aspects of their behavior such as

- Robustness and flexibility in operation, choosing a behavior according to the surrounding environment.

- Management of resources in terms of power, data storage, communication bandwidth.

- Failure recovery, the ability to recover system from faults given the costs associated with space missions.

Everything is strongly linked to the necessity of on board autonomy in operation. Great strides have now been made thanks to advances in path planning, obstacle avoidance and navigation control.

1.3 Any robot is as good as another?

1.3.1 General arrangement

The brief look at the history of UGVs has suggested how the shapes and configurations explored, as well as the solutions adopted for solving mobility and control problems, are many and varied. As the degree of autonomy increases, the development of an integrated system that can effectively manage the functions of hardware and software elements becomes essential [26]. Although in fact it is possible to use components that individually perform their function in the best possible way, attention must be paid to the effectiveness with which they are placed in relation to each other in order to obtain a globally efficient system. A series of macro-areas essential to the definition of an *autonomous system* are listed, both for the execution of high level tasks (sensing and perception, navigation and planning, behavior, communication) and low level ones (energy, monitoring and diagnosis, propulsion).

Sensing and Perception
The robot collects and interprets data from sensors or from the network in which it is inserted. This information is necessary for the awareness of the situation, i.e. for the creation of a representation of the environment in which the UV operates and its recognition through elements such as targets, obstacles, friendly units [105]. Perception is essential to achieve autonomous mobility for which the vehicle must be equipped with on-board sensors capable of orienting itself even in a complex environment. It is precisely the type of operating environment, on the other hand, that determines the type of sensors required: is the mission in sunlight? At night? In a wood? On a flat road? The identification of the operating scenario is essential to equip the robot with the tools necessary for the identification, classification and positioning of the elements in the space and to start its navigation performance.
What it needs is a set of [26]

- *Internal sensors*: for wheel velocity such as odometers and encoders, for the steering angle, for depth or altitude as pressure sensors, etc. They are used for low-level closed loop control and dead-reckoning navigation.

- *External sensors*: such as IMU, GPS, provide position and orientation with respect to an absolute reference frame.

- *Environmental sensors*: radar, IR and acoustic sensor, to create a more detailed map of the surrounding environment.

The purpose is to estimate the vehicle's position, orientation, speed and acceleration.

Navigation, Planning and Decision
Navigation System is concerned with the ability to move in a space toward a path of waypoints or to the final destination detected in the meanwhile hazards and obstacles. It takes input from perception functions and interacted with the other system to execute the mission [26], in particular with the Planning and Decision System. The latter is related to the trajectory to be followed and the actions to be execute to accomplish the mission,

i.e. to the generation of a safety path from a specified starting position to a final position [105]. Input of the planning algorithm are a map of the environment in which obstacles have been identified, the starting point and the final one. The major goal of both is to provide enough information to allow near-autonomous mobility for the UGV [105].

Behavior end mobility
While Behavior is related to the combination of the outputs from Navigation, Planning and Perception system to translate them in actuator commands for the robot, Mobility is linked with the ability to implement these commands to really move the robot around the environment [26]. Behavior can be also defined as *the observable response of a single robot vehicle to internal and external stimuli* [105].

Communication and Human Interaction
Communication System provides the channel to exchange data between the robot and other units, both operators or other robot in the network. When a user need to understand what the system is doing and correct or monitor the situation, a Human Machine Interface or HMI need to be developed [26]. Even for the highest levels of autonomy, in fact, human intervention is required to set high-level goals, any additional limitations, problem monitoring and diagnosis, resource management [105].

Energy an Propulsion
It may seem trivial but even the smartest machine is unable to act unless powered. The storage of energy on board and the efficiency with which it is spent are two key points in the development of UVs. For small robots, electrical energy is the most used form; with increasing size, hybrid systems are also allowed [26].

Health, Monitoring and Diagnosis
Monitoring and Diagnosis systems are used to detect and isolate failures within UVs subsystems. Although it is not widespread in small UGVs, it is widespread in larger ones [71].

Learining/Adaptation and Network Collaboration
Learning and Adaptation skills are essential to improve the robot performance over time and they can have a great impact on navigation, planning and behavior tasks. The same occurs with the possibility of exchanging information with a network of other vehicles engaged in the execution of the same mission.

1.3.2 Degree of autonomy

Since the 1960s there has been a significant miniaturization of hardware and enormous advances have been made in the fields of sensors, computer processing, image and signal processing, communication and control techniques, allowing the achievement of an ever higher degree of autonomy. It is precisely the level of autonomy possessed a first indicator to create a classification: the one considered, although it refers to UGVs used in the military field, highlights the essential characteristics that distinguish one type of UGV from the other ([105] and [26]):

- *Tele-operated UGVs or TGVs*: robot is controlled by human operators from distance.

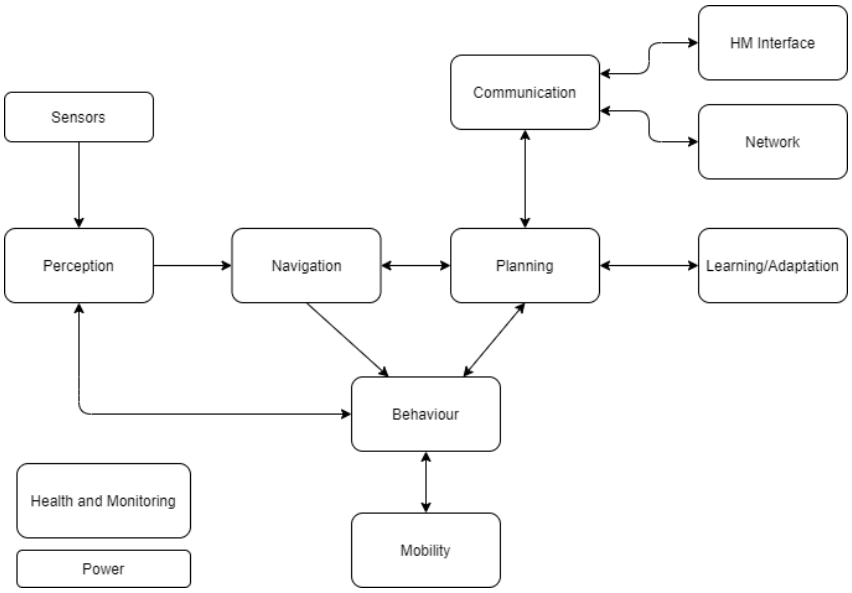


Figure 1.5: General Structure and Elements of a UV

If the distance is such that the operator cannot directly see the vehicle, it is the information acquired through the sensors on board it and shown through a display system that allows the display of the environment in which it is moving. The operator is responsible for the actions of the robot, which he commands via a control interface, and for the success of the mission, given that it is indeed the conductor of all cognitive processes. For teleoperated robots, it is necessary to ensure a robust man-machine communication channel, a valid interface between them, excellent mobility and excellent energy management. High operational performance is not required from the robot itself, as well as tactical skills or cooperation with other units, learning and adaptation or even mission and path planning (except in the case of loss of communication, where minimum performance is required) since it is the operator who takes care of all decision-making processes.

- *Semiautonomous*: the robot moves using cognitive processes to select the best way to move from one point to another. It is therefore capable of carrying out path planning and obstacles avoidance tasks, with a more articulated suite of sensors than the TGVs, including sensors for geolocation, night and day vision cameras, laser and spectral sensors. It must have sufficient autonomy to select the best path and processes capable of identifying its position independently, completing the mission with a few calls to the operator. High performance is therefore required in the areas of navigation, mobility and energy management, together with the need to guarantee excellent performance also in path planning, adaptation to the environment and communication.

- *Platform-centric Autonomous Ground Vehicles or PCAGVs*: from DARPA definition *autonomous* refers to *A mode of control of a UGV wherein the UGV is self-sufficient. The UGV is given its global mission by the human, having been programmed to learn*

from and respond to its environment, and operates without further human intervention. In order to be autonomous, the vehicle must therefore be able to independently travel the distance from point A to point B, showing ability to detect obstacles, navigation, reliability and robustness in crossing unknown environments, execution of tactical maneuvers, awareness of the situation, while there is no need to ensure excellent communication with the human operator.

- *Network-centric Autonomous Ground Vehicles or NCAGVs*: autonomous robots that must stand out for their ability to operate autonomously and in a network, exploiting the information received from the communication network and incorporating it into their own mission execution.

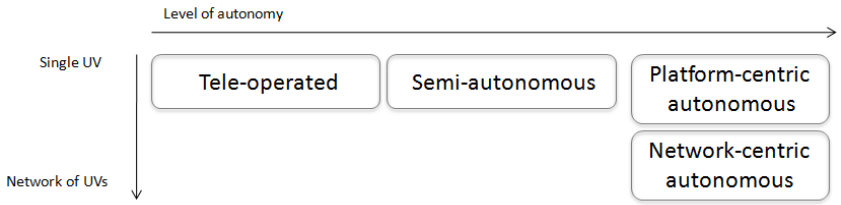


Figure 1.6: Degree of autonomy and interaction with other UVs

The rapid evolution of technologies and their transition to the civil sphere have led to the birth of other classifications capable of facing the multiplicity of nuances that such systems have assumed, such the one of Sheridan and Verplank [92], the ALFUS (Autonomy Levels for Unmanned Systems) classification [41] or the one defined by ASTM Commitee F45 on Driverless Automatic Guided Industrial Vehicles [13]. The choice of one level of autonomy over another is linked to the type of mission and surrounding conditions (operating environment, mobility within the same, hostility, sensors available): it is therefore a real design trade-off.

1.3.3 Mobility strategies

The choice of the type of mobility is also a key point of the design trade-offs. Aside from the possibility of airborne and submarine systems, the most accredited and developed alternatives for UGV are

- *Wheel-enabled systems*: one of the most used form of mobility in motion on the ground.
- *Leg-enabled systems*: inspired to biological systems, they are designed to operate in dangerous and rough terrain.
- *Track-enabled systems*: common in military vehicles, the tracks allow movement on difficult terrain.
- *Hybrid systems*: they are a combination of more kind of mobility form exploiting the advantages of each of them. Four possibilities have been studied: legs-wheels, legs-tracks, wheels-tracks and legs-wheels-tracks [14].

A comparison between different strategies can be made based on a series of parameters as reported in [90] and [14]

- Maximum speed capability: maximum speed on flat surface without obstacles.
- Obstacle traverse capability: capability to cross obstacles with random shapes.
- Slope climb capability: capability of climbing sloping surfaces maintaining a sufficient degree of adhesion.
- Soil sinkage: ability to move on soft soil by having the least contact pressure without large slip.
- Payload mass fraction capacity.
- Mechanical complexity: level of complexity of the system elements.
- Energy efficiency: how energy consumption is efficient related to normal operating condition.
- Technology readiness level: level of maturity of the technologies.

The first five voices are linked to pure mobility performance (they are quantitatively measurable) and the last three to a more general vision of system (considerations upon reliability, for example, can be made based on them).

(a) *Wheel-enabled system [19]* (b) *Track-enable system [27]* (c) *Leg-enabled system [89]*

Figure 1.7: Different mobility strategies

Wheeled systems
Wheeled systems relie on a well established technology and on the ability to reach high speeds on flat ground, with moderate sinkage on soft ground. These systems are generally cheaper than other solutions and have shown high reliability, but their ability to overcome obstacles is generally limited.

Tracked system
Tracked system has a better ability to climb and overcome obstacles, at the cost of a higher energy expenditure. Generally their reliability is lower than other systems due to the possibility of track failure which brings to partial or total immobility. Only moderate speeds

are allowed compared with wheeled systems.

Legged systems
Legged systems are relatively recent and studies have shown high stability of movement when overcoming obstacles [90] but higher power consumption. They are relatively slow and characterized by high mechanical complexity which translates into the need for a complex Control System.

Hybrid systems
Hybrid systems possess favorable characteristics due to the possibility of changing mobility strategy depending on the situation. However they are extremely complex.

Summarizing and focusing on the basic three categories, the use of wheels has the advantage of guaranteeing high speed, low consumption, adequate redundancy and payload carrying capacity but low slope climb capacity due to slippage. Tracks have good mobility in all terrains, high traction capacity even in yielding ones and good payload carrying capacity, but they are not energy efficient due to the high friction of the tracks and characterized by low speed and low reliability. Legged systems are distinguished by a good ability to overcome obstacles but they require high energy to perform operations. With reference to [14], a graphical comparison is proposed in Figure 1.8: along the x-axis, speed and energetic efficiency are reported; along the y-axis, mobility in unstructured environments is considered. Considering more properly the mechanical arrangement of the system that

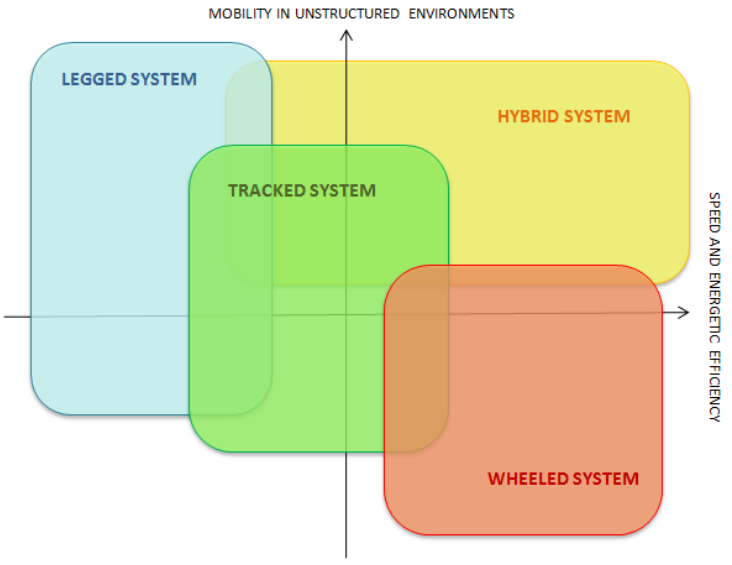


Figure 1.8: Comparison between different mobility strategies

allows the mobility of the robot, different configurations for wheeled and tracked system in particular are possible:

- *Car-like vehicles*

- *Bi-wheeled vehicles*
- *Tracked vehicles*
- *Omni-wheeled vehicles*

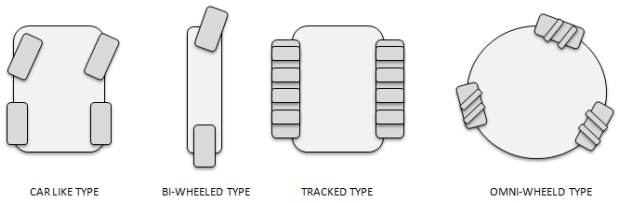


Figure 1.9: Comparison between different mechanical configurations

Again the mission conditions, costs, operating environment and technology availability among others will guide the choice.

1.4 Mobile Robot Control and Navigation

An "intelligent" component in the robot is necessary to process information coming from the environment and obtain commands to be sent to the actuators, responsible for the physical movement of the body. Navigation is associated with the generation of a model of the surrounding world in the form of a map, calculation of a safe trajectory from the starting position to the final one and control of the movement along this trajectory avoiding collisions with obstacles [88]. It is essential to

- provide enough information about robot's states: Perception and Localization task.
- plan a trajectory free of obstacles: Trajectory planning task.
- control the movement along the trajectory selected: Control task.

1.4.1 Perception and Localization task

The ability of a robot to move autonomously in the environment is linked to its ability to "perceive" what surrounds it. This is possible thanks to the use of sensors. As well as in [88], two groups of sensors are identified:

- *Proprioceptive-exteroceptive*
  - Proprioceptive sensors read information internal to the robot such as motor speed, battery voltage etc.
  - Exteroceptive sensors acquire information from the environment such as distances, sound and light intensity etc.

- *Passive-active*
  - Passive sensors measure energy coming from surroundings such as microphones, cameras, temperature probes etc.
  - Active sensors radiate energy to measure the reaction of surroundings elements.

In [106], the most common sensors are divided in:

- *Mechanical sensors*: they require a physical contact with objects.
- *Acoustic sensors*: they employ ultrasound frequencies to sent and receive signals in order to compute angular and linear position of the body.
- *Electomagnetic sensors*: they use the directionality and the time-of-flight information to compute robot position.
- *Magnetic sensors*: they exploit static magnetic fields to obtain information.
- *Optical sensors*: they include cameras and other device sensible to light.

Within these groups they stand out

- *Tactile sensors*, they sense objects at a small distance around the robot.
- *Encoders*, they measure the angular speed of the motor shaft and constitute basic element for motor control.
- *Ultrasonic sensors*, they are sound based sensors used to measure distance.
- *Accelerometers*, they feel acceleration.
- *Gyroscopes*, they measure angular velocities and orientation.
- *Laser range finders*, they calculate the time between the emission of a signal and its return, obtaining the distance from the obstacle.
- *Vision-based sensors*, they use the electromagnetic spectrum to produce images.
- *Depth sensors*, able to reconstruct a 3-D view of the environment.

Localization, on the other hand, is related to the knowledge of the robot's position. Different techniques [12] are available to estimate it, grouped in

- *Relative position measurements or dead-reckoning*, such as odometry and inertial navigation;
- *Absolute position measurements*, such as magnetic compasses, Global Positioning System (GPS), landmark navigation.

The choice is usually associated with the availability of sensors on board, the required accuracy, complexity, application purposes (GPS, one of the most used techniques, is not useful for indoor missions, for example).

Odometry
One of the most widely used navigation method, it provide good short term accuracy with a very low cost. Odometry is based on simple equations where the speed of the wheels is related to the movements of the robot and the position is obtained by integrating incremental motion information over time [12]. It is however subject to systematic and non-systematic errors: the former result from the kinematic imperfections of the robot (difference in wheel diameters, different length of the tracks), the latter from dynamic interaction with the ground.

Inertial Navigation
Gyroscopes and accelerometers are used to measure rates of rotation and accelerations, obtaining through integration position information. They have the advantage of not requiring external references but at the same time the disadvantage of accumulating and increasing any small constant error after integration. The combination of gyroscopes and odometry allows to immediately detect and correct orientation errors that would otherwise be increased leading to a wrong lateral position information.

Global Positioning System
Technology available for outdoor applications. It provides to a GPS receiver information about its location by transmitting a radio signal from a Satellites System and elaborating it using advanced trilateration methods.

Landmark Navigation
Navigation is performed by identifying landmarks, i.e. distinct features of the environment that robot can recognize (geometric shapes o specific code) relative to which it can localize itself.

1.4.2 Trajectory planning task

Trajectory planning is related to find the force inputs to move actuators in order to follow the best path and reach the target without obstacles collision [88]. There are numerous methods developed in an attempt to solve the problem of motion planning of a mobile robot. In [77] off-line and on-line algorithms are identified:

- *Off-line path planning*: complete information about stationary obstacles and moving obstacles trajectory is known in advance.
- *On-line path planning*: information is acquired from the environment through sensors and the path planned updated taking this into account.

They include ([88])

- *Classical methods*: roadmaps such as Visibility graphs and Voronoi diagram, potential functions, Cell Decomposition. It is necessary to introduce the initial position, the

position of the goal, that of the static obstacles and the motion model of the dynamic ones. The goal of the algorithm is the generation of the next position that the robot must reach in order to arrive at the goal. In *roadmap* approach, positions that can actually be occupied by the robot are mapped onto a network and the planning problem degenerates to a graph-searching problem. In *Cell Decomposition*, the space is decomposed into a series of cells and the path that joins the cell occupied by the robot initially with the one containing the final arrival point is calculated, avoiding collisions with the cells where there are obstacles. Movement is allowed based on the relationships between adjacent cells. In the *Potential Fields* approach, the robot is identified by a moving point influenced by the potential field defined by the sum of an attractive potential towards the goal and a repulsive potential towards obstacles [80].

- *Probabilistic methods*: they try to solve problems of the classical methods such as limitations related to the complexity of computation in high-dimensional problems and entrapment in local minima; they include probabilistic roadmap planners, randomized potential planners, etc.

- *Heuristic planners*: they guarantee completeness, efficiency and optimality with respect to the classical planner; A* algorithm is an example.

- *Evolutionary algorithms*: since classical approaches tend to fail in a dynamic environment with multiple obstacles [62], evolutionary algorithms have been developed. Genetic Algorithm and Particle Swarm Optimization are examples among the others.

- *Sensor-based planners*: movement is planned on the base of sensors information.

1.4.3 Control task

Controllers allow the plant to obtain the desired results in terms of achieving or maintaining a position, a speed, an orientation. A *plant* is a physical entity which takes any form of energy as input and produces an output; a *system* is more than a plant, it consists of a plant together with other components around it [46]. The overall assembly constitutes the system to be controlled. The control action is effective if it acts according to the specific characteristics of the system. The analysis of the plant is necessary to understand which controller is suitable based on the nature of the inputs and outputs or based on the nature of the system itself, linear or non-linear for example. Not all controllers are suitable for all situations.
There are two basic types of Control Systems [11]:

- *Open loop* in which the output from the system has no effect on the input signal to the plant or process.

  – Advantage: a simpler and cheaper plant with good reliability in case the functioning of the system and its characteristics are well known.

  – Disadvantage: no correction for errors in output which may result from external disturbances, so lower accuracy.

- *Closed loop* in which the output of the system is fed back to the input and compared with a reference. The error is used to control that the output is compliant with the required value.
  - Advantage: more accurate than the open loop solution in achieving and maintaining the required value.
  - Disadvantage: more complex and more costly than the open loop solution.

Basic elements of an open loop Control System (Figure 1.10) include:

- *Control*: it establishes the action to be taken as a result of the input to the system.
- *Actuation*: it performs the action to change the variable being controlled on the basis of the input from the controller.
- *Plant*: it is the element on which the actuation block acts to obtain the desired output.

Figure 1.10: Open loop system

Basic elements of a closed loop Control System (Figure 1.11) include:

- *Comparison element*: the reference value is compared with the actual output measurement and an error signal is generated.
- *Control*: it establishes the action to be taken as a result of the error signal coming from the comparison element.
- *Actuation*: it performs the action on plant on the basis of the input from the controller.
- *Plant*: it is the element on which the actuation block acts.
- *Measurement element*: it fed back a signal containing information about the actual value of plant output in order to let it available as input for the comparison element.

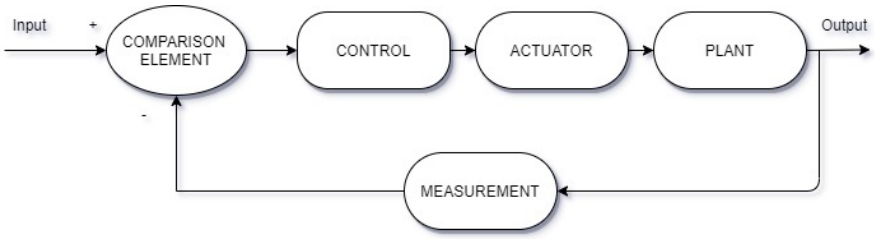


Figure 1.11: Closed loop system

In general, *feedback control* is the control for which the signal relating to the current conditions is brought back to modify the input signal [11]; *forward control* is the control for which the control signal acts on the plant without any type of retraction.
There are many solutions for controlling a system. Standard robot controllers include

- *PID controller*.

- *Computed torque controller*, effective in the reduction of the effects of uncertainties in the dynamic model.

- *Sliding mode controller*, based on the definition of a sliding surface in space and of a control law that keeps the trajectory of the plant on the surface itself, causing it to "slide" on that, possibly in a finite time [79].

- *Adaptive controller*, whose feature is the automatic adjustment in real time to achieve and maintain a desired performance, especially suitable with unknown or change in time behavior of the plant [79].

- *Neural networks controller*, based on the use of neural networks, algorithm able to learn from observational data and find a solution to the problem.

- *Fuzzy logic controller*, using a collection of fuzzy conditional statements in the form *if* $(\cdot)$ *then* $(\cdot)$ to decide what action perform to obtain the desired output. The advantage in its choice is that does not depend on any mathematical model, so that its application on non-linear systems is also effective [79].

Again, the choice of one type of control over another is a design choice.

1.5 Motivation and Contributions

This is intended to be an introduction into the world of small UGVs in order to underline their potential: mobile robots such as Turtlebot, Husky, Jackal UGVs are widely used in research for testing guidance strategies, control methods, Artificial Intelligence, Simultaneous localization and mapping or SLAM algorithms. The Devastator platform stands among them with the aim of achieving autonomous navigation in GPS denied environments, using only the information taken from sensors and processed by the on-board computer.
The goal is to obtain the model of the robot so that its behavior can be reproduced into a simulation environment, taking advantage of the possibility of simulating more realistic and complex missions than those that would be physically possible to perform in a research laboratory.
The behavior of mobile robots has been extensively studied from many points of view:

- Kinematic description of the robot behavior

  – Cerkala and Jadlovska [17] focuses on the mathematical modelling of mobile robot with differentially driven two-wheel chassis considering the non-holonomic constraints that characterized it.

  - Majkut, Giergiel and Kohut [58] analyze the kinematics of a small crawler robot assuming that it can be properly described using a two-wheeled parametrical model. The "virtual wheels" location is found in those points where there is no slip between the ground and the tracks.
  - Nagatani, Endo and Yoshida [68] proposed a method to improve odometry accuracy including consideration on slippage; the kinematic model is completed with slip values identification through a relationship that links them to the speed of the tracks, whose parameters have been empirically calculated.
  - Martinez at all [63] employ a kinematic approach for tracked mobile robots in order to improve motion control and pose estimation based on the research of the Instantaneous Center of Rotation position for the whole vehicle and for both the tracks.

- Including dynamic consideration in the modeling

  - Dar and Longoria [20] proposed a combination of kinematic and dynamic model to describe robot behavior; GPS measurements and torque values obtained through the motor model are used with an Extended Kalman Filter to estimate the unknown parameters, especially the friction coefficients, demonstrating their relationship with the vehicle velocity and turning radius.
  - Cerkala and Jadlovska [17] integrate a dynamic model together with the kinematic one and introduce friction effects demonstrating the serious impact on the mobile robot final position.
  - Dhaouadi and Hatab [38] develop a dynamic model for differential-drive mobile robots based on Lagrangian mechanics and on Newton-Euler mechanics.

- Considering it as a tracked vehicle, even if in a small version

  - Wong [115] extensively discusses the mechanics of vehicles equipped with wheels as well as tracks and the complex interaction between them and the terrain.
  - Wong and Chiang [114] elaborate a general theory for skid steering of tracked vehicles under steady state conditions on firm ground, exploring the shear stress-shear displacement relationship between the track and the ground.

- Considering robots as a input-output entities, moving towards a black box modeling

  - Raafiu and Darwito [76] use input and output data sets coming from experiments to create a MISO black box model of ARX and ARMAX type, characterized by PWM-current inputs and angular speed of the wheel output.
  - Sjijberg et al. [96] presented an unified overview of nonlinear black-box modeling in System Identification, fundamental in the development of the mobile robot model.
  - Granja et al. [35] employ System Identification to create a controller that improves the performance of the autonomous mobile robot. A second order model is obtained with motors voltage as input and center of mass position in the acquired image as output, since the robot is equipped with a camera.

  - Iglesias et al. [43] propose to perform the System Identification of the robot to obtain a transparent model to be used in control development and training; the obtained NARMAX model allows the study of the system stability and the individuation of the main factors involved in the execution of robot's task.
  - Kyriacou et al. [42] show how the modeling approach can be used to describe the robot-environment-task interaction, making realistic predictions of robot's behavior.

- Considering them as platforms for testing Guidance, Control and Navigation algorithms

  - Pebrianti et al. [74] propose a mathematical modelling and a controller design for autonomous wheeled mobile robot based on PID strategy; both a kinematic model and a state-space model have been realized, together with a P, PD and PID controller versions.
  - Barsan [5] proposes PID controllers for the position control of a mobile robot with differential steering and reports the implementation and tuning steps.
  - G. Li and X. Li [52] exploit a sliding mode control method for trajectory tracking, using an exponential reaching law instead of conventional laws to reduce chatter and ensure reaching time as well.
  - Ahmadi, Polotski and Hurteau [2] propose a path planning and control solution based on a path in every point of which it is also defined the desired speed. The objective of the controller is that of finding the tracks speed such as to minimize the error between the desired and the actual velocity possessed by the robot and this is reached first finding traction forces required and then the corresponding track speed to be reached applying that forces.
  - Guldner et al.[36] propose an artificial potential fields algorithm with a sliding mode control strategy to improve the performance of non-holonomic mobile robots navigation.
  - Gupta et al. [37] employ an outer controller acting on positional error and a inner PID controller acting on velocity error for the control of a non-holonomic wheeled mobile robot.
  - Ferrara and Rubagotti [25] propose a second order sliding mode control strategy of the mobile robot based on a harmonic potential field.
  - Tzafestas [106] provides a global overview of mobile robot control and navigation methodologies developed over the last decades.
  - Rubio et al. [88] present instead the state of the art, trends and novel application upon locomotion, perception, cognition, and navigation fields.
  - Sabudin et al. [66] review the traditional artificial potential field theory and the variety of algorithms based on potential field method that have been implemented to upgrade the potential function performance in obstacle avoidance and local minima problem.

Modeling and Control of mobile robots are very complicated problems, where a general solution has not yet been reached: this is due to the great variety of configurations that a mobile robot can assume, to the multiple sensors it can be equipped with, to the different applications for which it is intended. The modeling strategy followed in the course of this thesis is a black box modeling of the behavior of the robot associated with a kinematic model. The future integration of a dynamic model has been hypothesized given the non-negligible presence of slippage, but the achievement of high tracking and control performance is beyond the scope of this study and the effort has been directed rather to the creation of a simple algorithm of Guidance and Control for the validation of the implemented model.
The aim is to demonstrate how through the creation of a simulation model it is possible to perform the tuning of the parameters and obtain the direct translation of the algorithm into code that can be implemented on board the platform, with the considerable advantages of not requiring continuous tests with the robotic platform, performing tests in safety, freedom of experimentation and simplicity of integration of the solutions considered valid.

1.5.1 Outline

This thesis is structured as follows:

- In chapter 2 the mathematical model of the mobile robot is presented; first of all a kinematic model characterized by non-honolomic constraints, subsequently a model including the effects of slippage of the tracks and finally a dynamic model including the friction affecting the system. An innovative method in the characterization of the inertia of the system is presented using a composite pendulum and finally the lumped parameter model of the DC motors, actuators of the system, is introduced.

- In chapter 3 the System Identification discipline is introduced, with a general overview of the most used methods and in particular by introducing non-linear black-box modeling; the steps necessary to create the model are summarized here, describing the entire process from the data collection to the validation phase.

- In chapter 4 the algorithms selected for the implementation of GNC System are widely discussed. The APF method is presented as path planning strategy while a PID controller has been implemented for the reference tracking.

- In chapter 5 the experimental setup is presented together with all the experiments conducted in order to obtain the robot model. For DC motors, both the model obtained through a lumped parameter approach and that obtained through a data-driven approach are reported. The inertia of the system is also calculated through the pendulum experiment.

- In chapter 6 the model developed in the MATLAB/Simulink environment is introduced. After a general overview of its structure, each block is analyzed and each parameter in G&C algorithm studied in order to obtain acceptable performance.

- In chapter 7 the Gazebo environment is presented together with a timid introduction to the ROS world, indispensable tool for robotics. The goal is to ensure a good

behavior of the Guidance algorithm by testing it on a pre-existing model of the Devastator implemented in Gazebo environment through a Gazebo-Simulink co-simulation, in order to be able to proceed with its transformation into code through the Code Generation function offered by MATLAB.

- In chapter 8 simple indoor missions are performed using the MATLAB/Simulink model and Gazebo model. A real application on Devastator platform is also presented to demonstrate the success of code generation.

Finally, a summary of what has been done and further considerations regarding the future improvement of the presented algorithms have been reported.

Chapter 2

Mathematical model

Virtually every system we can think of can be described by mathematical model [8]

Cars, biological organisms, economic data flows, power plants: all these systems, although very different for characteristics and properties, can actually be represented by mathematical relationships combined in an appropriate way inherent to the type of phenomenon underlying each one. The set of these relations constitutes the *mathematical model* of the system and the purpose of this chapter is to present the one identified for the representation of the mobile robot behavior.

2.1 Basic concepts

Physical entities such as forces, torques, velocities and accelerations can be described by vectors [23]. Let a Cartesian coordinate frame a (Figure 2.1) to be introduced: it can be defined by three orthogonal unit vectors $\hat{\boldsymbol{a}}_1$, $\hat{\boldsymbol{a}}_2$ and $\hat{\boldsymbol{a}}_3$ along the x, y, z axes of a. A vector

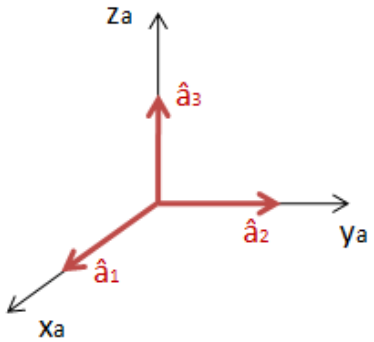


Figure 2.1: Coordinate Reference System a

$\boldsymbol{v}$ in a can be expressed as

$$\boldsymbol{v} = v_1^a \hat{\boldsymbol{a}}_1 + v_2^a \hat{\boldsymbol{a}}_2 + v_3^a \hat{\boldsymbol{a}}_3$$

where

$$v_i^a \;=\; \boldsymbol{v} \cdot \hat{\boldsymbol{a}}_i$$

with $i \in \{1,\ 2,\ 3\}$ are the unique components of $\boldsymbol{v}$ in a. In a more compact form,

$$\boldsymbol{v}^a = \begin{pmatrix} v_1^a \\ v_2^a \\ v_3^a \end{pmatrix}$$

If another Cartesian coordinate frame b with orthogonal unit vectors $\hat{\boldsymbol{b}}_1$, $\hat{\boldsymbol{b}}_2$ and $\hat{\boldsymbol{b}}_3$ along axes is introduced, $\boldsymbol{v}$ can equally be described by

$$\boldsymbol{v} = v_1^b \hat{\boldsymbol{b}}_1 + v_2^b \hat{\boldsymbol{b}}_2 + v_3^b \hat{\boldsymbol{b}}_3$$

where

$$v_i^b \;=\; \boldsymbol{v} \cdot \hat{\boldsymbol{b}}_i$$

with $i \in \{1,\ 2,\ 3\}$ are the unique components of $\boldsymbol{v}$ in b. Again, it is possible to compact notation as

$$\boldsymbol{v}^b = \begin{pmatrix} v_1^b \\ v_2^b \\ v_3^b \end{pmatrix}$$

So superscript a or b denotes that the vector $\boldsymbol{v}$ is given by the coordinates in a or in b.
It is possible to find a relation that binds the components of the vector $\boldsymbol{v}$ written in the reference a, i.e. $\boldsymbol{v}^a$, directly with those written in the reference b, i.e. $\boldsymbol{v}^b$

$$\boldsymbol{v}^a = \boldsymbol{R}_b^a \boldsymbol{v}^b$$

where

$$\boldsymbol{R}_b^a = [\hat{\boldsymbol{a}}_i \cdot \hat{\boldsymbol{b}}_j]$$

is the *rotation matrix* $\boldsymbol{R}_b^a$ from a to b and its elements r_{ij} the *direction cosines*. Given that the rotation matrix from a to b transforms a coordinate vector in b to one in a, the matrix can also be called *coordinate transformation matrix* from b to a. An important property of the rotation matrix is its orthogonality, so that the inverse passage from the coordinates in the frame a to those in the frame b is achieved through

$$\boldsymbol{R}_a^b = (\boldsymbol{R}_b^a)^{-1} = (\boldsymbol{R}_b^a)^T$$

A rotation about a fixed axis is denoted as *simple rotation* and the coordinate transformation matrix is a function of the angle of which the two systems are rotated. Rotation from one frame to another may also be described by a *composite rotation* made up of several simple rotations between systems. Let suppose a rotation from a frame a to a frame c: it can be obtained through a rotation from a to b and then one from b to c, i.e.

$$\boldsymbol{v}^a = \boldsymbol{R}_b^a \boldsymbol{v}^b$$

$$\boldsymbol{v}^b = \boldsymbol{R}_c^b \boldsymbol{v}^c$$

Combining these two equations it is possible to write

$$\boldsymbol{v}^a = \boldsymbol{R}_b^a \boldsymbol{R}_c^b \boldsymbol{v}^c = \boldsymbol{R}_c^a \boldsymbol{v}^c$$

so that

$$\boldsymbol{R}_c^a = \boldsymbol{R}_b^a \boldsymbol{R}_c^b$$

The rotation matrix for the composite rotation $\boldsymbol{R}_c^a$ is simply the product of the rotation matrices from a to b and from b to c.
Consider two Cartesian reference systems a and b consisting of a set of three right-hand axes. The rotation matrix describing the respective orientation of the frames is a 3×3 matrix with nine elements of which six are constrained given the orthogonality of the matrix, so only 3 independent parameters are sufficient to identify it. One of the most common parameterizations is that given by the Euler angles. The *Euler angles* are used to describe the motion of rigid bodies that move freely in space: the rotation from a to b is described as a rotation of ψ about the z_a-axis, a rotation of θ about the current rotated y-axis and finally a rotation of ϕ about the current rotated x-axis (Figure 2.2).

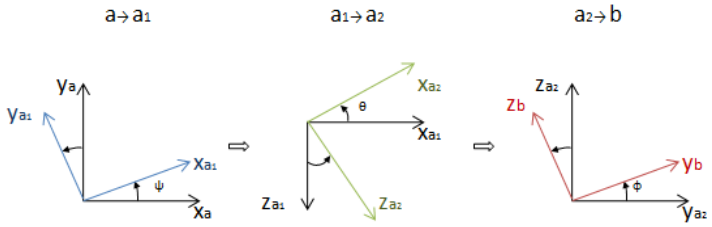


Figure 2.2: Set of rotations from a to b

- From a to a_1, rotation of ψ around $z_a \rightarrow \boldsymbol{v}^a = \boldsymbol{R_z}(\psi)\boldsymbol{v}^{a_1}$

$$\boldsymbol{R_z}(\psi) = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- From a_1 to a_2, rotation of θ around $y_{a_1} \rightarrow \boldsymbol{v}^{a_1} = \boldsymbol{R_y}(\theta)\boldsymbol{v}^{a_2}$

$$\boldsymbol{R_y}(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

- From a_2 to b, rotation of ϕ around $x_{a_2} \rightarrow \boldsymbol{v}^{a_2} = \boldsymbol{R_x}(\phi)\boldsymbol{v}^b$

$$\boldsymbol{R_x}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi & cos\phi \end{bmatrix}$$

So

$$\boldsymbol{v}^a = \boldsymbol{R_z}(\psi)\boldsymbol{R_y}(\theta)\boldsymbol{R_x}(\phi)\boldsymbol{v}^b \;=\; \boldsymbol{R}_b^a\boldsymbol{v}^b \tag{2.1}$$

where $\boldsymbol{R}_b^a = \boldsymbol{R_z}(\psi)\boldsymbol{R_y}(\theta)\boldsymbol{R_x}(\phi)$ is the coordinate transformation matrix that transforms a vector $\boldsymbol{v}^b$ to a vector $\boldsymbol{v}^a$. The order of the rotations is important and not random.
Suppose that the aim is to identify the change in the rotation matrix over time, that is, define the angular velocity of one system with respect to the other. Considering a set of simple rotations

- Angular velocity of a_1 with respect to a written in frame $a \rightarrow \boldsymbol{\omega}_{aa_1}^a = \boldsymbol{\omega}_z = \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix}$

- Angular velocity of a_2 with respect to a_1 written in frame $a_1 \rightarrow \boldsymbol{\omega}_{a_1a_2}^{a_1} = \boldsymbol{\omega}_y = \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix}$

- Angular velocity of b with respect to a_2 written in frame $a_2 \rightarrow \boldsymbol{\omega}_{a_2b}^{a_2} = \boldsymbol{\omega}_x = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix}$

The angular velocity of frame b relative to frame a of the composite rotation written in frame a will be

$$\boldsymbol{\omega}_{ab}^a = \boldsymbol{\omega}_{aa_1}^a + \boldsymbol{\omega}_{a_1a_2}^a + \boldsymbol{\omega}_{a_2b}^a$$

equal to

$$\boldsymbol{\omega}_{ab}^a = \;\boldsymbol{\omega}_z + \boldsymbol{R_z}(\psi)\boldsymbol{\omega}_y + \boldsymbol{R_z}(\psi)\boldsymbol{R_y}(\theta)\boldsymbol{\omega}_x$$

Differentiation of a vector $\boldsymbol{v}$ with respect to a reference system can be expressed as

$$\frac{^a d}{dt}\boldsymbol{v} := \dot{v}_1^a\hat{\boldsymbol{a}}_1 + \dot{v}_2^a\hat{\boldsymbol{a}}_2 + \dot{v}_3^a\hat{\boldsymbol{a}}_3$$

where the superscript a on the differentiation operator denotes a differentiation with reference to frame a. The same can be written for differentiation with reference to frame b:

$$\frac{^b d}{dt}\boldsymbol{v} := \dot{v}_1^b\hat{\boldsymbol{b}}_1 + \dot{v}_2^b\hat{\boldsymbol{b}}_2 + \dot{v}_3^b\hat{\boldsymbol{b}}_3$$

A direct relationship can be expressed through the involvement of the angular velocity between the two reference systems $\boldsymbol{\omega}_{ab}$

$$\frac{^a d}{dt}\boldsymbol{v} = \frac{^b d}{dt}\boldsymbol{v} + \boldsymbol{\omega}_{ab} \times \boldsymbol{v} \tag{2.2}$$

These basic concepts will be exploited for the description of the kinematic relationships of which the model is constituted.

2.2 Motion in the space

Consider a rigid body free to move in space. Let there be two reference systems, an *Inertial Reference Frame* i and a *Body Reference Frame* b, such that (Figure 2.3)

- Inertial Reference Frame i, fixed in the space, with a right-handed set of orthogonal axes X, Y and Z
- Body Reference Frame b, joined to the body, with a right-handed set of orthogonal axes x, y and z

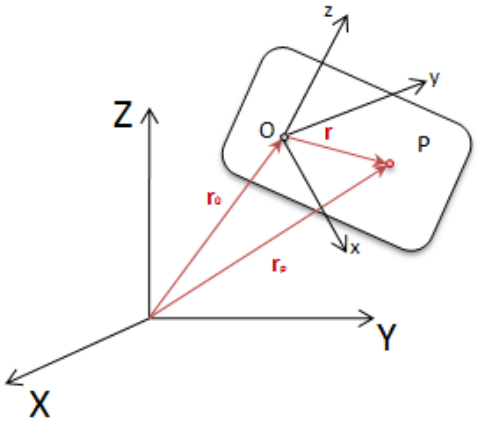


Figure 2.3: Rigid body in the Inertial Reference Frame

The position of any point P in the rigid body is given by

$$\boldsymbol{r}_P = \boldsymbol{r}_0 + \boldsymbol{r} \tag{2.3}$$

where $\boldsymbol{r}_0$ is the position of one reference point in the rigid body with respect to the origin of the frame i and $\boldsymbol{r}$ is the vector from that point to P.
The speed of the point P with respect to the Inertial system can be expressed by differentiating[1] the expression (3.16)

$$\frac{d}{dt}\boldsymbol{r}_P = \frac{d}{dt}\boldsymbol{r}_0 + \frac{d}{dt}\boldsymbol{r} \tag{2.4}$$

to obtain

$$\boldsymbol{v}_P = \boldsymbol{v}_0 + \frac{d}{dt}\boldsymbol{r} \tag{2.5}$$

Similarly to what is expressed in (2.2), it is possible to differentiate $\boldsymbol{r}$ with respect to the Inertial frame as

$$\frac{d}{dt}\boldsymbol{r} = \frac{{}^b d}{dt}\boldsymbol{r} + \boldsymbol{\omega}_{ib} \times \boldsymbol{r} \tag{2.6}$$

[1]For simplicity of writing, the superscript i will be omitted to indicate a differentiation with respect to the Inertial reference system

introducing the angular speed $\boldsymbol{\omega}_{ib}$ of the two references.
The complete expression is shown below

$$\boldsymbol{v}_P = \boldsymbol{v}_0 + \frac{{}^b d}{dt}\boldsymbol{r} + \boldsymbol{\omega}_{ib} \times \boldsymbol{r} \tag{2.7}$$

It is also possible to express the acceleration of P with respect to the origin of the frame i differentiating two times (2.4).

$$\begin{aligned} \frac{d^2}{dt^2}\boldsymbol{r}_P &= \frac{d^2}{dt^2}\boldsymbol{r}_0 + \frac{d}{dt}\Big(\frac{d}{dt}\boldsymbol{r}\Big) \\ &= \frac{d^2}{dt^2}\boldsymbol{r}_0 + \frac{d}{dt}\Big(\frac{{}^b d}{dt}\boldsymbol{r} + \boldsymbol{\omega}_{ib} \times \boldsymbol{r}\Big) \\ &= \frac{d^2}{dt^2}\boldsymbol{r}_0 + \frac{{}^b d}{dt}\Big(\frac{{}^b d}{dt}\boldsymbol{r} + \boldsymbol{\omega}_{ib} \times \boldsymbol{r}\Big) + \boldsymbol{\omega}_{ib} \times \Big(\frac{{}^b d}{dt}\boldsymbol{r} + \boldsymbol{\omega}_{ib} \times \boldsymbol{r}\Big) \end{aligned} \tag{2.8}$$

By performing simple algebraic steps from (2.8) it is possible to obtain the final expression

$$\boldsymbol{a}_P = \boldsymbol{a}_0 + \frac{{}^b d^2}{dt^2}\boldsymbol{r} + 2\boldsymbol{\omega}_{ib} \times \frac{{}^b d}{dt}\boldsymbol{r} + \frac{d}{dt}\boldsymbol{\omega}_{ib} \times \boldsymbol{r} + \boldsymbol{\omega}_{ib} \times (\boldsymbol{\omega}_{ib} \times \boldsymbol{r}) \tag{2.9}$$

with

- $\boldsymbol{a}_P$ acceleration of P
- $\boldsymbol{a}_0 = \frac{d^2}{dt^2}\boldsymbol{r}_0$ acceleration of 0
- $\frac{{}^b d^2}{dt^2}\boldsymbol{r}$ second derivative of $\boldsymbol{r}$ in b
- $2\boldsymbol{\omega}_{ib} \times \frac{{}^b d}{dt}\boldsymbol{r}$ Coriolis acceleration
- $\frac{d}{dt}\boldsymbol{\omega}_{ib} \times \boldsymbol{r}$ transversal acceleration
- $\boldsymbol{\omega}_{ib} \times (\boldsymbol{\omega}_{ib} \times \boldsymbol{r})$ centripetal acceleration

2.3 Assumptions and kinematic relationship

The objective of the discussion is the modeling of the robot behavior in its motion on a flat surface. The following assumptions are accepted:

- The robot is considered a rigid body on tracks, that is, all the dynamics relating to the elements inside the chassis are ignored.
- The robot motion is attribute to that of its center of mass CoM.
- The degrees of freedom of the body are three, two of position in the plane and one of orientation along the vertical axis which is orthogonal to the surface on which the motion occurs.

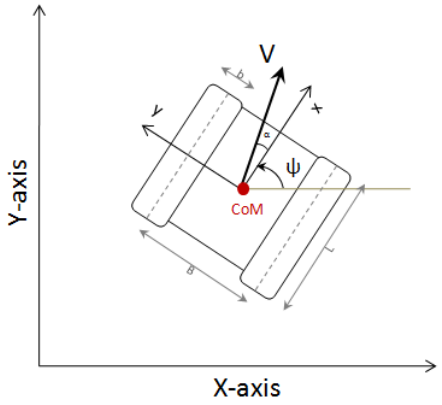


Figure 2.4: Reference Frames of the robot

- The Inertial Reference System is assumed such that the plane on which the X and Y axes lie is parallel to that of the motion (the floor), while the Body Reference System is such that the x-axis is in the frontal direction of advancement, the z-axis parallel to Z, y in the lateral direction to complete the right-hand triad (Figure 2.4).
- rotations around the Z-axis are assumed to be positive if counterclockwise.

The position of the CoM in the Inertial Reference Frame is expressed through the pair of coordinates X and Y; the speed V is expressed through the components V_X and V_Y referring to it, while through the components V_x and V_y referring to the local one. As seen in Section 2.1, the relationship between them can be easily obtained by referring to (2.1) and considering that the only significant rotation is that of entity ψ around the Z-axis. By gathering the information necessary to establish the velocity and the angular speed of the body in a single vector, it is possible to express it as

$$\begin{pmatrix} V_X \\ V_Y \\ \dot{\psi} \end{pmatrix} = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} V_x \\ V_y \\ \dot{\psi} \end{pmatrix} \tag{2.10}$$

which, in a more compact form, becomes

$$\boldsymbol{\xi}^i = \boldsymbol{R}_b^i \boldsymbol{\xi}^b \tag{2.11}$$

The forward speed of the robot depends on the speed of the tracks, which in turn is a function of the rotational speed of the driving wheels located on the right and left side of the body. Given r the radius of the driving wheels, B the distance between the center lines of the tracks, ω_L and ω_R the angular speeds of the left and right driving wheels respectively, it is possible to predict the speed of the body in the global reference by a relationship of the type

$$\boldsymbol{\xi}^i = \begin{pmatrix} V_X \\ V_Y \\ \dot{\psi} \end{pmatrix} = f(B,\ r,\ \psi, \omega_L,\ \omega_R) \tag{2.12}$$

With the same behavior, the two tracks allows the motion forward/backward of the robot while with the difference in speed between them its rotation; furthermore neither tracks can contribute to sideways motion in the robot's reference frame.
By making the relation (2.12) explicit, the kinematic model in the local frame is obtained:

$$\boldsymbol{\xi}^b = \begin{pmatrix} V_x \\ V_y \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} \frac{r}{2}(\omega_L + \omega_R) \\ 0 \\ \frac{r}{B}(\omega_R - \omega_L) \end{pmatrix} \tag{2.13}$$

Through (2.11), it is possible to refer the model to the Inertial frame, thus obtaining

$$\boldsymbol{\xi}^i = \begin{pmatrix} V_X \\ V_Y \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} \frac{r}{2}(\omega_L + \omega_R)cos\psi \\ \frac{r}{2}(\omega_L + \omega_R)sin\psi \\ \frac{r}{B}(\omega_R - \omega_L) \end{pmatrix} \tag{2.14}$$

2.4 Kinematic constraints

It is important to identify and define the type of kinematic constraints given that they limit the admissible configurations in the space of the robot. A *holonomic* kinematic constraint is a constraint that can be expressed as an explicit function of position variables only [94]. A *nonholonomic* kinematic constraint is a constraint that requires a differential relationship to be expressed: it cannot be integrated to provide a constraint in terms of the position variable. The sliding constraint belongs to the latter category.
Approximate the behavior of the track to that of an equivalent wheel of angular velocity ω, whose axis of rotation always remains contained in a plane parallel to that of the motion of the robot. Suppose that this equivalent wheel has only one point of contact with the ground and that it does not slip with respect to the latter: this is equivalent to affirming that pure rolling motion is performed and no lateral slippage occurs. Consider the generic configuration shown in Figure 2.5 in which a wheel constrained to a robot chassis is reported. It is indicated with β the angle between the x-axis of the body and the velocity vector V, while with l the distance of the body CoM to the wheel ground point of contact. It can be a fixed wheel, i.e. β is fixed, unable to rotate with respect to an axis perpendicular to the plane of motion, or a car-like wheel which has this further possibility (β is a function of time).
Mathematically the condition of pure rolling and no lateral slippage for both the type of wheel are translated in kinematic constraints by imposing that

$$\begin{bmatrix} cos\beta & sin\beta & -lsin(\alpha+\beta) \end{bmatrix} \cdot \boldsymbol{R}_i^b \boldsymbol{\xi}^i - r\omega = 0 \tag{2.15}$$

$$\begin{bmatrix} sin\beta & -cos\beta & lcos(\alpha+\beta) \end{bmatrix} \cdot \boldsymbol{R}_i^b \boldsymbol{\xi}^i = 0 \tag{2.16}$$

Suppose the robot has N wheels: N_f standard fixed wheels and N_s steerable ones. Indicating the rotational speeds of these wheels respectively with ω_f and ω_s and defining the vector $\boldsymbol{\omega}$ such as

$$\boldsymbol{\omega} = \begin{pmatrix} \omega_f \\ \omega_s \end{pmatrix}$$

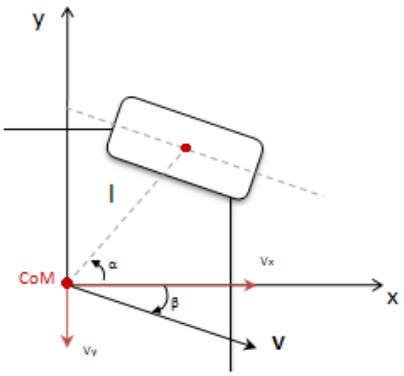


Figure 2.5: General configuration of a wheel constrained to a robot chassis

the rolling constraints of all wheels can be expressed in a single equation as

$$\boldsymbol{J_1 R_i^b \xi^i} - \boldsymbol{J_2 \omega} = 0 \tag{2.17}$$

where

$$\boldsymbol{J_1} = \begin{bmatrix} \boldsymbol{J_{1f}} \\ \boldsymbol{J_{1s}} \end{bmatrix} \tag{2.18}$$

with $\boldsymbol{J_{1f}}$ constant $N_f \times 3$ matrix made up of rows containing the three terms of matrix in (2.15) and $\boldsymbol{J_{1s}}(\beta_s)$ $N_s \times 3$ matrix made at the same way but for a steerable wheel (so depending of β_s variable over time); J_2 is a constant diagonal $N \times N$ matrix with the radii of all the wheels.
The sliding constraints of all wheels can be instead expressed in a single equation as

$$\boldsymbol{C_1 R_i^b \xi^i} = 0 \tag{2.19}$$

where

$$\boldsymbol{C_1} = \begin{bmatrix} \boldsymbol{C_{1f}} \\ \boldsymbol{C_{1s}} \end{bmatrix} \tag{2.20}$$

with $\boldsymbol{C_{1f}}$ constant $N_f \times 3$ matrix made up of rows containing the three terms of matrix in (2.16) and $\boldsymbol{C_{1s}}(\beta_s)$ $N_s \times 3$ matrix for a steerable wheel.
In summary, (2.17) represents pure rolling constraint while (2.19) the constraint for which no motion orthogonal to the wheel plane is allowed. Assuming a parallelism between the tracks of a mobile robot and an equivalent wheels representation, the latter are constrained so that $\beta = 0$ and for the right wheel $\alpha = -\pi/2$ while for the left one $\alpha = \pi/2$. Reformulating the relations (2.17) and (2.19) with these values, model in equation (2.13) can be obtained, thus demonstrating that the combination of wheel pure rolling and sliding constraints can provide the kinematic behavior of the system.
From the analysis of the rank[2] of the matrix $\boldsymbol{C_1}$ it is possible to trace the number of independent constraints that characterize the motion. In general, it is between 0 and 3:

[2] *Rank* is the maximum number of linearly independent rows or columns of a matrix

- It is 0 only if there are no independent kinematic constraints and it is possible if neither fixed or steerable standard wheels are attached to the chassis.
- It is 3 only if each of the three degrees of freedom is constrained.

It is therefore defined the *degree of mobility* δ_m as

$$\delta_m = 3 - rank[\boldsymbol{C_1}] \tag{2.21}$$

and it is a measure of the degrees of freedom modifiable by changing the speed of rotation of the wheels. For the differential-drive robot model equivalent to the tracked one examined earlier, matrix $\boldsymbol{C_1}$ can be expressed as

$$\boldsymbol{C_1} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.22}$$

Rank of $\boldsymbol{C_1}$ is 1, so $\delta_m = 2$. It is deduced that by acting on the angular speed of the wheels it is possible to modify both the angular speed and the velocity of forward/backward motion. For the sake of completeness, it is reported that for vehicles with steering wheels a *degree of steerability* δ_s can be defined: δ_s is equal to the rank of the $\boldsymbol{C_{1s}}$ matrix.
The total degrees of freedom that a robot can manipulate are defined by the sum of the terms of mobility and steerability and defined as *degree of maneuverability*

$$\delta_M = \delta_m + \delta_s \tag{2.23}$$

It is important to take into account the degree of maneuverability as this is connected to the control of the robot and the possibility of achieving specific positions in space.

2.5 Kinematic constraints violation

Pure rolling and no-slip conditions considered in the realization of the model (2.13) are assumptions whose admissibility needs to be checked given the presence of a large contact area of the tracks with the ground and given that they cannot be applied in skid-steering case[59]. Skid steering principle is based on controlling the relative velocities of both left and right tracks such as for differential drive wheeled vehicles: the difference lies in the presence of a not negligible slippage during steering [62].
Suppose to expand the previous treatment by denying the condition of pure rolling and considering the presence of longitudinal slippage through the *slip ratios* i defined as

$$i_L = \frac{v_L - v'_L}{v_L} \tag{2.24}$$

$$i_R = \frac{v_R - v'_R}{v_R} \tag{2.25}$$

for the right and left tracks respectively, with $v_L = r\omega_L$ and $v_R = r\omega_R$ theoretical left and right velocities of tracks and v'_L and v'_R left and right actual velocities of tracks respect to the ground. The denial of the condition of no lateral motion is also possible defining a *side slip angle* α, i.e. the angle between longitudinal orientation of the mobile robot and its

actual running direction (Figure 2.4), that is non zero if lateral slippage occurs. Anyway, assuming that vehicle's velocity is small enough and friction force large enough, lateral slippage can be neglected and $\alpha = 0$ [68].
By integrating these considerations, equations (2.14) can be reformulated as

$$\begin{aligned} V_X &= \frac{r}{2}(\omega_L(1-i_L)+\omega_R(1-i_R))cos\psi \\ V_Y &= \frac{r}{2}(\omega_L(1-i_L)+\omega_R(1-i_R))sin\psi \\ \dot{\psi} &= \frac{r}{B}(\omega_R(1-i_R)-\omega_L(1-i_L)) \end{aligned} \tag{2.26}$$

If no slippage occurs, $i_L = 0$ and $i_R = 0$ and the model degenerates in the previous reported.
Slip ratios are determined by physical interactions between the tracks and the ground [24], they are very complex to predict and require dynamical consideration to be investigated.

2.6 Dynamics

The dynamics of a tracked mobile robot is linked to the forces developed by the interaction of the tracks with the ground. Neglecting the aerodynamic resistance and the effect of suspension, the vehicle is subject to [102]

- tractive forces F_L and F_R
- longitudinal resistance forces R_L and R_R
- lateral force F_y
- moment of turning resistance induced by resistance forces M_r

Tractive forces
The introduction of basic concepts of *terramechanics* is essential to understand track-terrain interactions. For certain types of terrain the behavior of the soil is similar to that of an ideal elastoplastic material [102]. Below a certain stress, the behavior is elastic; when the yield point is reached, this behavior becomes plastic and the transition leads to soil failure. The stress for which this occurs is calculated by the Mohr-Coulomb theory and is equal to

$$\tau = c + \sigma tan\phi \tag{2.27}$$

with τ shear stress, c apparent cohesion, σ normal stress on the sheared surface and ϕ angle of internal shearing resistance.
Cohesion of the material is the bond that cements particles together irrespective of the normal pressure between them. When it is negligible, the result is that only a normal pressure can held together particles. Thus, for example, shear stress does not depend on normal pressure for clayey soil, while for dry sand it clearly increases with normal load; this difference is why the equation (2.27) includes both contributions, since the ground surface particles usually have both cohesive and frictional properties [115].

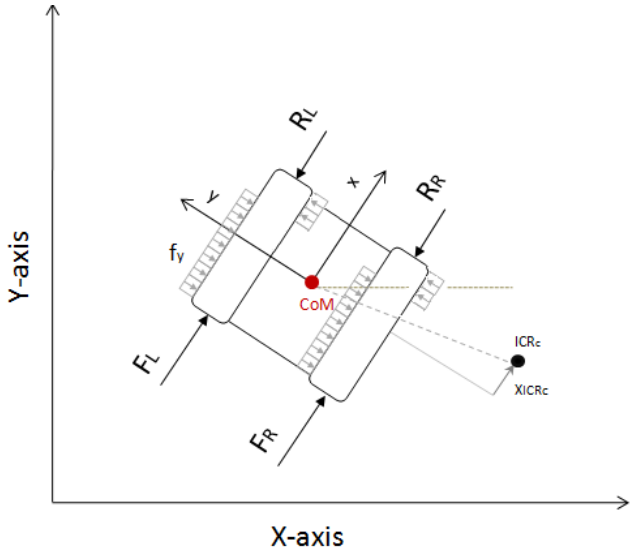


Figure 2.6: Free-body diagram of tracked vehicle

When a torque is applied to the sprocket of a track, shearing action appears at track-terrain interface. To predict vehicle thrust and associated slip, the shear stress-shear displacement relationship of the terrain is required.
Homogeneous soil properties are assumed, as well as the rigidity of the tracks and the impossibility of stretching. Slip has been defined in equation (2.24) and (2.25) for left and right track. A general expression is here reported:

$$i = \frac{v - v'}{v} = \frac{v_j}{v}$$

where v is the theoretical speed of the track defined by the sprocket rotational speed and its radius, v' the forward speed of the track and v_j the speed of slip with respect to the ground. *Shear displacement* j at a distance x from the front of the track-terrain contact area can be found with the equation

$$j = v_j t$$

with $t = x/v$ contact time of a considered point with the terrain. Rearranged the equation

$$j = ix$$

i.e. shear displacement increases linearly with the distance from the front of the contact area [102].
Different consideration must be done according to the type of terrain (Figure 2.7):

- loose sand, saturated clay, dry fresh snow: the shear stress initially increases rapidly with an increase in shear displacement, and then approaches a constant value with a further increase. This behavior is represented by Janosi and Hanamoto exponential law

$$\tau = \tau_{max}(1 - e^{-j/K}) = (c + \sigma tan\phi)(1 - e^{-j/K}) \tag{2.28}$$

with K *soil shear deformation modulus*. It can be considered as a measure of the magnitude of the shear displacement required to develop the maximum shear stress [115].

- organic terrain (muskeg) with a mat of living vegetation on the surface: the shear stress initially increases rapidly with the increase of shear displacement, it reaches a threshold of maximum shear stress and then it continuously decreases. This is due to the fact that the lower layers offer lower shear strength than the surface carpet. The equation that describes this behavior is

$$\tau = \tau_{max}(j/K_{max})e^{1-j/K_{max}} \tag{2.29}$$

where K_{max} is the shear displacement where the maximum shear stress occurs.

- compact sand, silt, loam and frozen snow: the shear stress has the same behavior of that for organic terrain, but instead of continuously decreasing it approaches a more or less constant residual value. This behavior is represented by the equation

$$\tau = \tau_{max}K_r\{1 + (1/K_r(1 - 1/e)) - 1)e^{1-j/K_{max}}\}(1 - e^{-j/K_{max}}) \tag{2.30}$$

with K_r ratio between the residual stress shear and the maximum one, K_{max} as above.

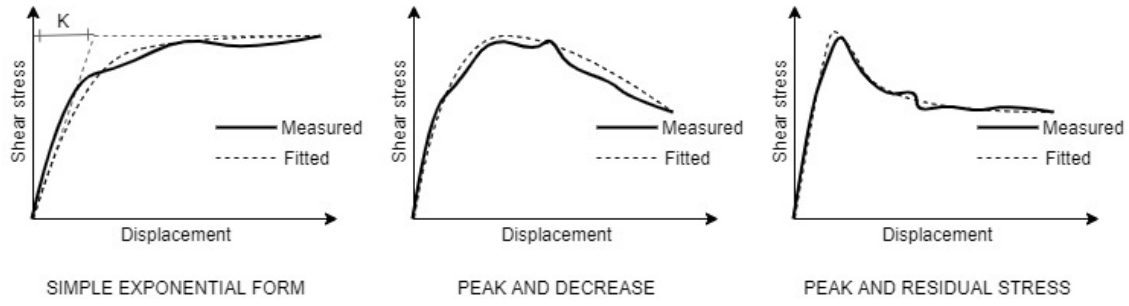


Figure 2.7: Shear stress for various terrains

The overall generated traction force is given by the integration of the stress along the track length L extended to its width b:

$$F = b\int_0^L \tau dx \tag{2.31}$$

where τ has the expression reported in equations (2.28), (2.29) or (2.30) according to the type of soil. If it is assumed that the ground is non-deformable, i.e. that the contact between the track and the ground is comparable to a rigid footprint, and that the distribution of the normal pressure produced by the track is uniform, τ takes the expression

$$\tau = \Big(c + \frac{W}{bL}tan\phi\Big)(1 - e^{-ix/K}) \tag{2.32}$$

where W is the normal load and W/bL the normal pressure, independent of x given the uniformity of the distribution. This allows to rewrite the expression (2.31) and get

$$F = (Ac + Wtan\phi)\Big(1 - \frac{K}{iL}(1 - e^{-iL/K})\Big) \tag{2.33}$$

with A contact area.

Longitudinal resistance forces
The longitudinal resistance contribution is calculated as proportional to the normal load acting on the track through a coefficient μ_r such as to be in the opposite direction with respect to the direction of travel of the track:

$$R_i = \begin{cases} -sign(v_i)\ \frac{mg}{2}\mu_r & \text{if } v_i \neq 0, \\ 0 & \text{if } v_i = 0 \end{cases} \tag{2.34}$$

where v_i is the speed transmitted by the tracks to the robotic platform and i stands for L or R, left or right track respectively.

Lateral force
The lateral friction force is calculated similarly to the longitudinal resistance force. It is proportional to the normal pressure per unit length of the track through a coefficient μ_l, then integrated taking into account the distribution of lateral frictional force due to the turning maneuver. The distribution is such as to depend of the center of instantaneous rotation [3] position, so

$$F_y = -sign(V_y)\ 4\mu_l \frac{mg}{2L} x_{ICR_c} \tag{2.35}$$

with x_{ICR_c} the position of the vehicle's ICR along the x-axis.

Turning Resistance Moment
The resistance moment is due to the generation of a lateral friction force on the tracks. Its value can be obtained by the integration of the distribution of this force over the track length:

$$M_r = -sign(\dot{\psi})\ 4\mu_l \frac{mg}{2L}\Big(\frac{L^2}{4} - x_{ICR_c}^2\Big) \tag{2.36}$$

where again x_{ICR_c} is the position of the vehicle's ICR along the x-axis. It is such as to oppose the rotation of the platform.

In this context, given the indoor application of the mobile robot on a flat and not deformable soil and the difficulty linked to identifying all the parameters necessary for the discussion, the forces that will be responsible for traction are those generated by the driving torque applied by the sprocket to the track. Traction torques will be indicated as $\boldsymbol{\tau} = \{\tau_L,\ \tau_R\}^T$ for the left and right track respectively.
Let m be the mass of the body, J its inertia of rotation about its center of mass along the z-axis. Dynamic equations in Body fixed frame are reported:

$$\begin{aligned} ma_x &= F_L + F_R - R_L - R_R \\ ma_y &= F_y \\ J\ddot{\psi} &= (F_R - R_R)\frac{B}{2} - (F_L - R_L)\frac{B}{2} - M_r \end{aligned} \tag{2.37}$$

[3] The instantaneous center of rotation (ICR) is defined as the point in the horizontal plane respect to which the motion can be represented by a rotation without translation [61]

with a_x, a_y and $\ddot{\psi}$ acceleration along the x-axis, y-axis and angular acceleration around the z-axis. Similarly to what done for the kinematic model in Section 2.3, the equations can be traced from the Body system to the Inertial system:

$$\begin{pmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{\psi} \end{pmatrix} = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} a_x \\ a_y \\ \ddot{\psi} \end{pmatrix} \tag{2.38}$$

Defining as $\boldsymbol{M}$ the matrix with mass and inertia contribution

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix}$$

and as $\boldsymbol{R}_b^i$ the rotation matrix,

$$\boldsymbol{M} \begin{pmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{\psi} \end{pmatrix} = \boldsymbol{M}\boldsymbol{R}_b^i \begin{pmatrix} a_x \\ a_y \\ \ddot{\psi} \end{pmatrix} = \boldsymbol{R}_b^i \begin{pmatrix} F_L + F_R - R_L - R_R \\ F_y \\ (F_R - R_R)\frac{B}{2} - (F_L - R_L)\frac{B}{2} - M_r \end{pmatrix} \tag{2.39}$$

By rearranging the last term, a more compact expression can be reported:

$$\boldsymbol{M} \begin{pmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{\psi} \end{pmatrix} = \boldsymbol{B} \begin{pmatrix} \tau_L \\ \tau_R \end{pmatrix} - \boldsymbol{C} \tag{2.40}$$

with

$$\begin{aligned} \boldsymbol{B} &= \frac{1}{r} \begin{bmatrix} cos\psi & cos\psi \\ sin\psi & sin\psi \\ -B/2 & B/2 \end{bmatrix} \\ \boldsymbol{C} &= \begin{bmatrix} (R_L + R_R)cos\psi + F_y sin\psi \\ (R_L + R_R)sin\psi - F_y cos\psi \\ (R_R - R_L)B/2 + M_r \end{bmatrix} \end{aligned} \tag{2.41}$$

Summing up, the dynamic model is obtained:

$$\boldsymbol{M}\ddot{\boldsymbol{\xi}}^i = \boldsymbol{B}\boldsymbol{\tau} - \boldsymbol{C} \tag{2.42}$$

2.6.1 Inertia of the System

One of the fundamental parameters for the development of the dynamic model is the robot inertia to rotation around the Z-axis. In analogy to what reported in [16], the pendulum method is adopted in order to identify the value of J. Let introduce a composite pendulum made up of two bodies, respectively of mass m and m_1, as well as inertia I and I_1, free to oscillate together around a hinge. Introducing an Inertial Reference System, their position is described by a right-hand triad $x,\ y,\ z$ with origin in the hinge and z-axis parallel to the rotation axis (Figure 2.8). In this formulation it is neglected the motions along x and z directions and it is indicated with θ the angle of oscillation of the pendulum. Furthermore the effects of friction between the hinge and the support are assumed as irrelevant. The equation of motion is evaluated through the Lagrangian dynamics as

$$\mathcal{L} = K - V \tag{2.43}$$

with K and V respectively kinetic energy and potential energy of the system. K is the

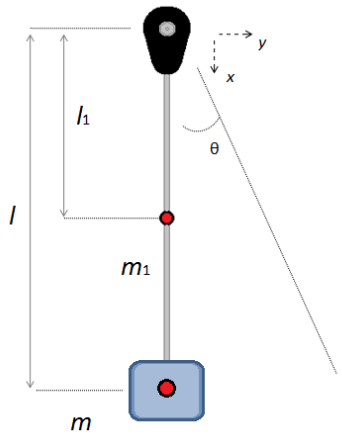


Figure 2.8: Pendulum system

sum of the kinetic energy of the centers of mass of the two bodies, consisting of a linear and a rotational component

$$K = \frac{1}{2}m\dot{y}^2 + \frac{1}{2}m_1\dot{y}_1^2 + \frac{1}{2}I\dot{\theta}^2 + \frac{1}{2}I_1\dot{\theta}^2 \tag{2.44}$$

reformulated as

$$K = \frac{1}{2}(ml^2 + m_1l_1^2 + I + I_1)\dot{\theta}^2 \tag{2.45}$$

where l is the center of gravity-hinge distance of the first body, l_1 the center of gravity-hinge distance of the second.
V is the sum of the potential energy of each body

$$V = mgl(1 - cos\theta) + m_1gl_1(1 - cos\theta) \tag{2.46}$$

Applying the Lagrangian approach,

$$\frac{d}{dt}\Big(\frac{\partial K}{\partial \dot{\theta}}\Big)+\frac{\partial V}{\partial \theta} = 0 \tag{2.47}$$

that is

$$(ml^2 + m_1 l_1^2 + I + I_1)\ddot{\theta} - (mgl + m_1 g l_1)sin\theta = 0 \tag{2.48}$$

Considering oscillations of small amplitude it is possible to assume the validity of the theory of small perturbations for which $sin\theta \approx \theta$ and the expression can be reformulated as

$$A\ddot{\theta} + B\theta = 0$$

second order equation for undamped systems with natural oscillation frequency

$$\omega_n^2 = \frac{B}{A} \tag{2.49}$$

directly related to the period of oscillation

$$T = \frac{2\pi}{\omega_n} \tag{2.50}$$

By substituting A and B with the respective expressions and isolating the unknown parameter of the system, it is possible to obtain the identification of the inertia of the system:

$$I = \Big(\frac{T}{2\pi}\Big)^2 (mgl + m_1 g l_1) - ml^2 - m_1 l_1^2 - I_1 \tag{2.51}$$

2.7 Actuation model

Robot motion is allowed by the movement of the two tracks; in turn, their movement is allowed by two DC motors. Another important element for the kinematic and dynamic models is in fact the behavior of the actuators in terms of angular speed and driving torque. An electric motor is an electro-mechanical device responsible of the conversion of electric energy into mechanical energy [47]. Electric machines can be classified into two big groups [64]: *brushed* and *brushless* machines. The difference relays in the use of mechanical contact elements to power the machine for the former and in their absence in the latter. Brushed DC motors are very widely used and this is linked to the easy understanding of their operations and therefore to the ease of their control: simple speed/torque control can be implemented with excellent drive performance [47].
To explain the working principle, let consider a "simplified motor" with one coil passing through a space between two permanent magnets with opposite polarization [50]. Referring to Figure 2.9, through brush X the current flows in section A of the commutator and through the coil arrives to the other commutator half ring B and brush Y. Because of the existence of a magnetic field generated by the permanent magnets, the current passing through the coil generates a force that is downwards for the right side and upwards for the left one, given that according to Lorentz law

$$\boldsymbol{F} = i\boldsymbol{L} \times \boldsymbol{B} \tag{2.52}$$

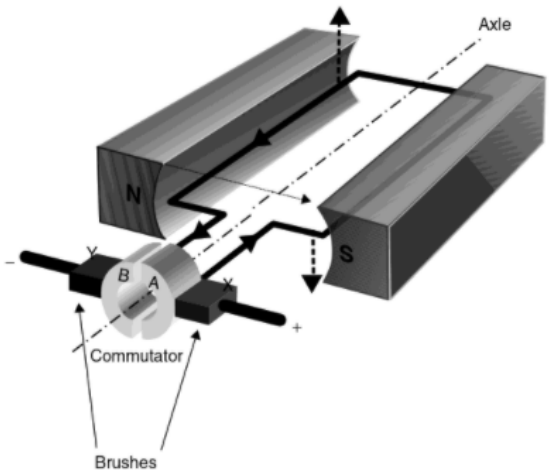


Figure 2.9: Simple permanent magnet DC motor-[50]

with i flowing current, $\boldsymbol{L}$ vector with the length of the wire and direction of current flow, $\boldsymbol{B}$ magnetic field vector. The coupling of the two forces generates a moment that turns the coil and the commutator. To ensure that rotation always occurs in the same direction, the direction of the current flowing in the commutator is switched.
In a real DC motor the rotating wire coil is wound round a piece of ferromagnetic material; more than one coil can be used, thus having a segmented commutator and not only a two half rings one; also more than one pair of magnets can be employed, stabilizing the output torque [50].
An electric motor is composed of two main parts:

- a stationary part or *stator*
- a moving part or *rotor*

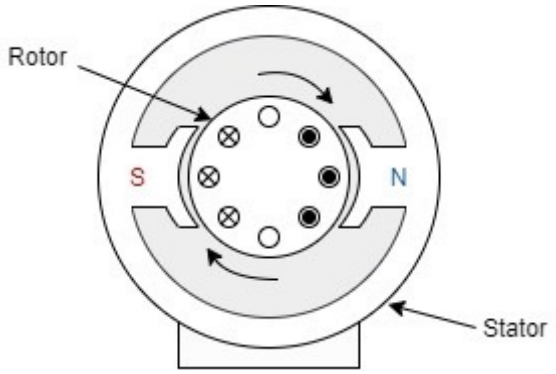


Figure 2.10: PM DC motor

Mechanical commutation devices are also present and consist of brushes and commutators. Similarly to the simplified motor previously described, the stator magnetic flux, generated

by the presence of permanent magnets, interacts with the rotor one, generated by the current flowing through the brushes and the collector, causing the birth of a force that tends to move the rotor and create a drive torque proportional to the armature current. The relationship between them is quantified by a coefficient, K_c, defined *torque constant*. The movement of the rotor has as a secondary effect the generation of a *back electromotive force*, proportional to the rotation speed ω_m through a coefficient K_{em} depending on the characteristics of the motor, defined *constant of Back EMF or Back EMF*, whose growth determines a decrease in the output torque. Against the motion there are also countless other phenomena that are difficult to identify, friction above all, linked to the losses in the materials, to the heat generated, to the sparking of the collectors.
A Permanent Magnet or PM brushed DC motor is analyzed and modeled. Assume a simplified model in which mechanical, iron and additional losses, as well as the influence of temperature and environment on machine parameters, are neglected and a mechanical friction is considered as a linear function of motor speed. The electric motor can be represented by two functional blocks:

- *Electromagnetic part*, that represents the behavior of the stator and rotor windings, described by a system of differential equations linked to electrical quantities.
- *Mechanical part* that represents the mechanical behavior of the moving parts, described by laws of dynamic equilibrium, linked to the inertia of the rotating masses and to the internal resisting torques.

Electromagnetic part
The DC motor is characterized by inductance L_a and resistance R_a. The equation describing the relationship between supply voltage V_a, current i_a and back EMF e is

$$V_a = L_a \frac{di_a}{dt} + R_a i_a + e \tag{2.53}$$

where it is suggested that the applied voltage V_a is the result of the voltage drop at the inductance, at the resistance and due to the presence of the counter electromotive force [39]. The back electromotive force e is shown to be proportional through the Back EMF constant to the motor rotation speed ω_m

$$e = K_{em}\omega_m \tag{2.54}$$

while the driving torque C_m at the intensity of the armature current through the torque constant

$$C_m = K_c i_a \tag{2.55}$$

Although the two constants have different names, they numerically and theoretically coincide: the phenomenon underlying the generation of the back electromotive force is the same underlying the production of the driving torque and is linked to the specific characteristics of the motor itself. In real application this is not always true due to mechanical imperfection, losses etc.

Mechanical part
The behavior of the rotating parts of the motor is simplified in a linear relationship by inserting

- a term representing the *driving torque* C_m
- a term representing the *resistant torque* C_1
- a term proportional to the speed of rotation through a coefficient f_m of friction
- a term proportional to the derivative of the rotational speed through the moment of inertia J_m

$$C_m = C_1 + f_m\omega_m + J_m\frac{d\omega_m}{dt} \tag{2.56}$$

Because of the presence of a gear box, a gear ratio need to be defined:

$$\tau = \frac{\omega}{\omega_m} \tag{2.57}$$

with ω_m angular speed of the motor shaft and ω angular speed of the external one. By imposing a unitary efficiency, for conservation of power

$$C_1\omega_m = \ C_2\omega \tag{2.58}$$

so that the torque C_1 is proportional at the one applied on the external shaft of the gear box C_2 through the gear ratio τ. When an external load is applied, at the external shaft

$$C_2 = C_r + f\omega + J\frac{d\omega}{dt} \tag{2.59}$$

where C_r is the resistant load torque, f is the friction coefficient and J the load inertia. Reducing to the motor shaft, it is possible to find

$$\begin{aligned} C_m &= C_2\tau + f_m\omega_m + J_m\frac{d\omega_m}{dt} \\ &= (C_r + f\omega + J\frac{d\omega}{dt})\tau + f_m\omega_m + J_m\frac{d\omega_m}{dt} \\ &= C_r\tau + (f\tau^2 + f_m)\omega_m + (J\tau^2 + J_m)\frac{d\omega_m}{dt} \\ &= C_r\tau + f_{tot}\ \omega_m + J_{tot}\frac{d\omega_m}{dt} \end{aligned} \tag{2.60}$$

Summarizing, the mathematical model of the DC motor-load system can be obtained:

$$\begin{cases} V_a &= L_a\frac{di_a}{dt} + R_a i_a + e \\ e &= K_{em}\omega_m \\ C_m &= K_c i_a \\ C_m &= C_r\tau + f_{tot}\ \omega_m + J_{tot}\frac{d\omega_m}{dt} \end{cases} \tag{2.61}$$

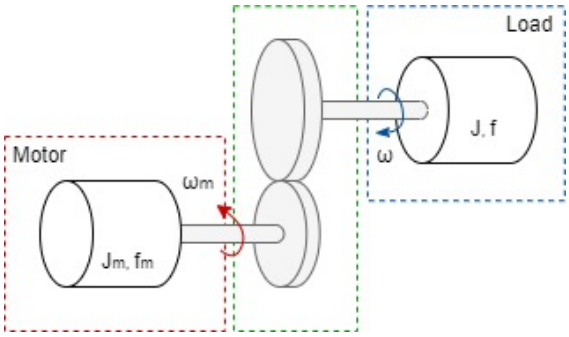


Figure 2.11: DC motor with gear and external load

Relating the motor torque and the angular velocity, the relation in Figure 2.12 is found: this relation is also known as the *DC motor characteristic*. The motor identification through a lumped parameter approach will be addressed to its realization.

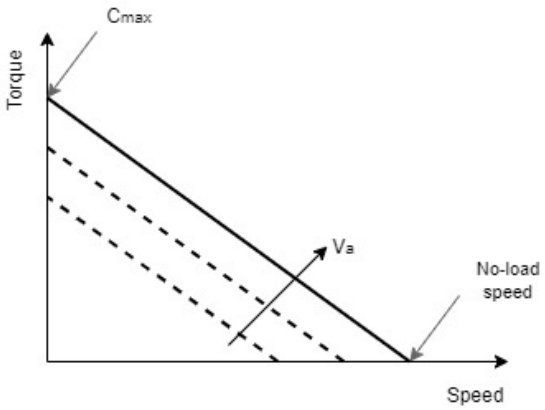


Figure 2.12: DC motor characteristic

Chapter 3

System Identification

Given the mathematical model and the input to a system, the system response can be computed: this is the simulation problem. [...] Given measurements of the system inputs and outputs, determine what the mathematical model of the system should be: this is 'system identification' [8]

System modeling is a procedure that concerns the identification of system characteristics and the possibility of reproducing its behavior in a simulation environment. For a good working model [111]

- data quality must be good enough
- model selection must be based on the intended use
- complex model does not mean better model, trade offs on appropriate complexity must be conducted
- data must be collected in a "smart" way, i.e. with the appropriate timescale
- validation must give acceptable results

Models can be classified as

- *White Box*: at the bases of the system there are known physical laws which are translated into a mathematical model.
- *Grey Box*: the physical laws underlying the system are known, but not all the parameters that characterize it.
- *Black Box*: the processes are not transparent and the physical laws are not immediately verifiable, no a priori information about the system is available.

The first type of model is linked to an approach based on the knowledge of the physical phenomenon, the second is a mixed approach that involves both physical laws and data analysis techniques, the last instead provides exclusively for a data-driven approach without the need of identifying the basic physical phenomenon.

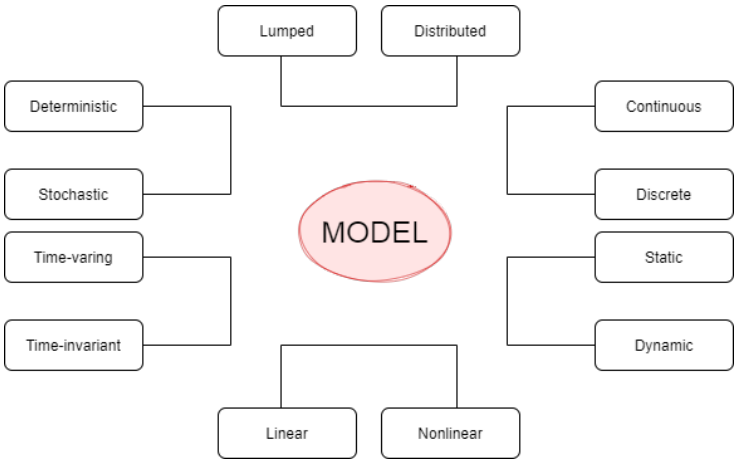


Figure 3.1: Model properties

From these considerations two fundamental approaches are highlighted [101]:

- *Analytical Modeling*
- *Experimental Identification*

The first is based on differential equations for the description of the physics of the process and for the characterization of the mechanical, electrical, thermal and fluid response; the second considers the system as an entity that, when prompted by data inputs, responds with certain outputs.
Experimental identification is linked to the discipline of *System Identification*. System identification is the mathematical description of a system obtained on the basis of available information, such as the knowledge of inputs and the measurement of outputs. The identification procedure includes the following steps [51]:

- Record a data set of input-output
- Choose the class of model or model structure
- Estimate the model coefficients on the basis of certain criteria
- Validate the obtained model

A discrete time treatment is introduced given the discrete nature of the sampling performed on the real system. A simplified scheme of the data recording process is shown in Figure 3.3: the external input given through a computer is converted from digital to analog actuators input that physically act on the system. The state of the former is then measured by sensors and the information reaches the computer again through an analog-digital conversion, thus obtaining the system outputs.
The sampling period T_s is important since it is linked to the information that can be obtained from a signal. The availability of information defines the model identifiability, i.e. the possibility to obtain a unique model. Identifiability is guaranteed, at least for linear time-invariant systems, when the input is *persistently exciting* [104], that is when it

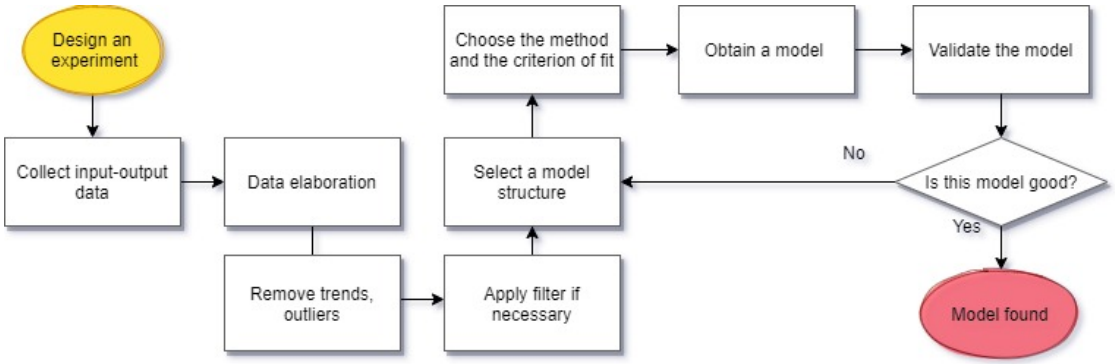


Figure 3.2: Identification procedure

contains almost all frequencies necessary to identify system dynamics.

For simplicity, let us consider linear time invariant systems with single output, single

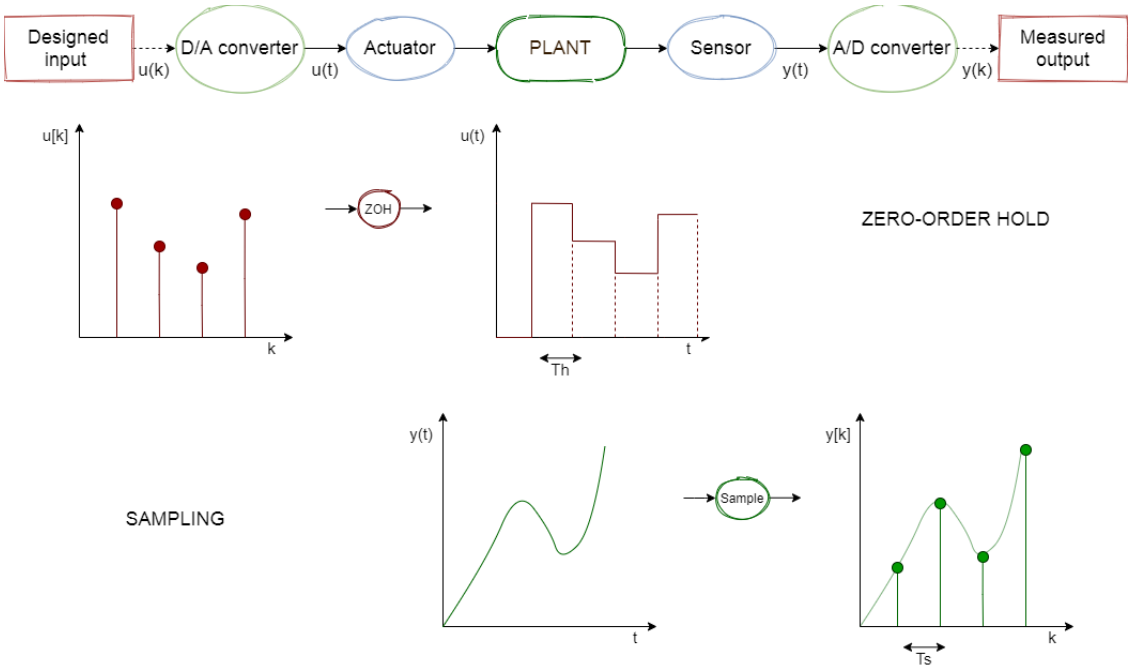


Figure 3.3: Sample data system

input (SISO) structure. In *time-domain* space, the mathematical model of a physical system is based on a set of differential/algebraic equations and can be represented both with a *Difference equation model* and with a *State-space model*. The former focuses upon input-output relationship, underlining the link between present and past inputs and output. The latter includes variables or *states* providing an internal description of the system. The state-variables vector $x(k)$ is defined as the minimal set of linearly independent variables such that knowledge of the states at any time k_0 plus the information on the input $u(k)$

for subsequently applied $k \geq k_0$ are sufficient to determine the state of the system and the output $y(k)$ at any time $k \geq k_0$ [22]. For system identification, an input-output model is preferred [22].

3.1 Model Structures

Basic structures can be employed to describe system dynamics, even if it is characterized by strong non-linearity. The objective is in fact to develop a model in order to predict the behavior of the system with an approximation: in some applications the shape of the model is not even important [8], as long as this is functional for the designer's purpose.
System output depends on the input, on the disturbances and on the measurement noise. Signals, noises or more generally the uncertainties of the system can be managed with two different approaches:

- *Deterministic approach*
- *Stochastic approach*

Deterministic signals include constant signals, exponential ones, sinusoids and their combinations. The basic deterministic signal capable of generating an output $y(k)$ is an input $u(k)$ modeled as a delta function $\delta(k)$ equal to

$$u(k) = \delta(k) = \begin{cases} 1 & if\ k = 0 \\ 0 & else \end{cases} \tag{3.1}$$

Stochastic signals are signals that at a fixed time instant k are random and their behavior is determined by their PDF of Probability density function[1]. The elementary signals analogous to the delta function for the deterministic case is the *white noise*, a sequence of independent and identically distributed random variables

$$e \sim WN(0,\ \sigma^2) \tag{3.2}$$

with zero mean ($E[e] = 0$) and autocorrelation[2] matrix in the form $\sigma^2 \boldsymbol{I}$, $\boldsymbol{I}$ identity matrix [113]. White noises are used to model the uncertainties that characterize the system.
The basic idea in modeling is to identify a process by splitting itself in a *predictable* and an *unpredictable* component [104]:

$$y(k) = \tilde{y}(k) + \nu(k) \tag{3.3}$$

with $\tilde{y}(k)$ the predictable part and $\nu(k)$ the ideal random process with white noise characteristics.

[1] PDF is a function whose value at any given sample in the sample space (the set of possible values taken by the random variable) can be interpreted as providing a relative likelihood that the value of the random variable would equal that sample [75]

[2] Autocorrelation is the cross-correlation of the signal with itself; considering a sample and another one of the same signal delayed by a quantity τ of time, autocorrelation can verify how similar the two values are to the progress of time [4]

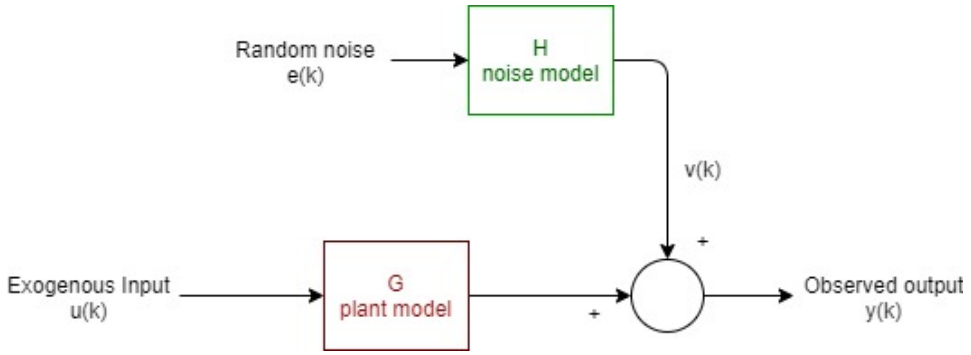


Figure 3.4: Model representation

3.1.1 LTI and SISO systems

In a more extended form LTI systems can be described by difference equations such as

$$y(k) + a_1 y(k-1) + \ ... \ + a_{n_a} y(k-n_a) = b_0 u(k) + b_1 u(k-1) + \ ... \ + b_{n_b} u(k-n_b) \quad (3.4)$$

with $[a_i]$ and $[b_i]$ multiplicative coefficients, $y(k)$ the output of the system, $u(k)$ the input, n_a and n_b non negative integers denoted as orders of the polynomials or model orders. In particular n_a is linked to the number of past outputs that have a link with the actual one, n_b is the number of past inputs that affect the output. Introducing the backward shift operator q^{-1} such that

$$q^{-1}x(k) = x(k-1) \quad (3.5)$$

and the forward shift operator q as

$$qx(k) = x(k+1) \quad (3.6)$$

with x a generic signal, equation (3.4) can be re-written as

$$(1 + a_1 q^{-1} + \ ... \ + a_{n_a} q^{-n_a}) y(k) = \ (b_0 + b_1 q^{-1} + \ ... \ + b_{n_b} q^{-n_b}) u(k) \quad (3.7)$$

Returning to a synthetic form,

$$y(k) = \ G(q^{-1}) u(k) \quad (3.8)$$

where

$$G(q^{-1}) = \frac{(b_0 + b_1 q^{-1} + \ ... \ + b_{n_b} q^{-n_b})}{1 + a_1 q^{-1} + \ ... \ + a_{n_a} q^{-n_a}} = \frac{B(q^{-1})}{A(q^{-1})} \quad (3.9)$$

is also known as *plant model*. From the analysis of the poles and zeros of the plant model it is possible to determine the *stability* of the discrete system:

- *Asymptotic stability* if all poles lie inside the unit circle.
- *Marginal stability* if all poles lie strictly inside or on the unit circle and the ones on the unit circle are simple.
- *Instability* if at least one pole lies outside the unit circle.

Noise in the system can be represented in a similar way of equation (3.8) as

$$\nu(k) = \ H(q^{-1})e(k) \tag{3.10}$$

where the white noise signal $e(k)$ is considered as an "input" of the stochastic process $\nu(k)$ and $H(q^{-1})$ the *noise model*.
Assuming that the process evolves as a linear combination of M past values of the input plus an uncertainty, a *Moving Average* (MA) model can be obtained

$$\nu(k) = \ H(q^{-1})e(k) = \ \Big(1 + \sum_{i=1}^{M} c_i q^{-i}\Big)e(k) \tag{3.11}$$

Supposing that the process evolves instead as a linear combination of P past values of the output plus an uncertainty, an *Auto Regressive* (AR) model can be obtained

$$\nu(k) = \ H(q^{-1})e(k) = \ \frac{1}{1 + \sum_{j=1}^{P} d_j q^{-j}}e(k) \tag{3.12}$$

More often a combination of the two is found: it is the *Auto Regressive Moving Average* (ARMA) model, with

$$\nu(k) = \ H(q^{-1})e(k) = \ \frac{1 + \sum_{i=1}^{M} c_i q^{-i}}{1 + \sum_{j=1}^{P} d_j q^{-j}}e(k) \tag{3.13}$$

A superimposition of the predictable and unpredictable parts is made to obtain a more realistic behavior of the system: in other words, it is equivalent to say that the output measurement $y(k)$ is the result of a combination between the response of the deterministic process $\tilde{y}(k)$ and the noise process $\nu(k)$ [104]

$$y(k) = \ G(q^{-1})u(k) + H(q^{-1})e(k) \qquad e(k) \sim WN(0,\ \sigma^2) \tag{3.14}$$

On the basis of the form presented in the equation (3.14) and assuming possible a parametric representation for $G(q^{-1})$ and $H(q^{-1})$ as rational polynomial transfer functions, it can be written that

$$\begin{aligned} A(q^{-1})y(k) &= \frac{B(q^{-1})}{F(q^{-1})}u(k) + \frac{C(q^{-1})}{D(q^{-1})}e(k) \\ G(q^{-1}) &= \ \frac{B(q^{-1})}{A(q^{-1})F(q^{-1})} \\ H(q^{-1}) &= \ \frac{C(q^{-1})}{A(q^{-1})D(q^{-1})} \end{aligned} \tag{3.15}$$

with $y(k)$ and $u(k)$, $k = 1,\ 2,\ ...$, the system output and input signals, $e(k)$ the noise sequence with white noise characteristics and $A(q^{-1})$, $B(q^{-1})$, $C(q^{-1})$, $D(q^{-1})$ and $F(q^{-1})$ polynomial expressed as

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \ ... \ + a_{n_a} q^{-n_a} \\ B(q^{-1}) &= b_1 q^{-n_k} + \ ... \ + b_{n_b} q^{-n_b+1-n_k} \\ C(q^{-1}) &= 1 + c_1 q^{-1} + \ ... \ + c_{n_c} q^{-n_c} \\ D(q^{-1}) &= 1 + d_1 q^{-1} + \ ... \ + d_{n_d} q^{-n_d} \\ F(q^{-1}) &= 1 + f_1 q^{-1} + \ ... \ + f_{n_f} q^{-n_f} \end{aligned} \tag{3.16}$$

with $[a_i]$, $[b_i]$, $[c_i]$, $[d_i]$ and $[f_i]$ coefficients, n_a, n_b, n_c, n_d and n_f the model orders, n_k the input-output delay.
Different models are available:

- Auto-Regressive with eXogenous input model or ARX
- Auto-Regressive and Moving Average with eXogenous input model or ARMAX
- Output Error model or OE
- Box-Jenkins model or BJ

ARX $\rightarrow$ noise filter has the same characteristics as the process

$$y(k) = \frac{B(q^{-1})}{A(q^{-1})}u(k) + \frac{1}{A(q^{-1})}e(k) \tag{3.17}$$

or in difference equation form

$$\begin{aligned} y(k)+a_1 y(k-1) + \ ... \ + a_{n_a} y(k-n_a) = \\ b_1 u(k-n_k) + \ ... \ + b_{n_b} u(k-n_k-n_b+1) + e(k) \end{aligned} \tag{3.18}$$

This structure is also denoted as ARX(n_a, n_b) with n_k as delay.

ARMAX $\rightarrow$ a moving average part for the noise model is considered

$$y(k) = \frac{B(q^{-1})}{A(q^{-1})}u(k) + \frac{C(q^{-1})}{A(q^{-1})}e(k) \tag{3.19}$$

or in difference equation form

$$\begin{aligned} y(k)+a_1 y(k-1) + \ ... \ + a_{n_a} y(k-n_a) = \ b_1 u(k-n_k) + \ ... \ + \\ b_{n_b} u(k-n_k-n_b+1) + c_1 e(k-1) + \ ... \ + c_{n_c} e(k-n_c) + e(k) \end{aligned} \tag{3.20}$$

This structure is also denoted as ARMAX(n_a, n_b, n_c) with n_k as delay.

OE $\rightarrow$ noise directly affects the output

$$y(k) = \frac{B(q^{-1})}{F(q^{-1})}u(k) + e(k) \tag{3.21}$$

or in difference equation form

$$\begin{aligned} y(k)+f_1 y(k-1) + \ ... \ + f_{n_f} y(k-n_f) = b_1 u(k-n_k) + \ ... \ + \\ b_{n_b} u(k-n_k-n_b+1) + f_1 e(k-1) + \ ... \ + f_{n_f} e(k-n_f) + e(k) \end{aligned} \tag{3.22}$$

This structure is also denoted as OE(n_b, n_f) with n_k as delay.

BJ $\rightarrow$ independent parametrization of plant and noise model

$$y(k) = \frac{B(q^{-1})}{F(q^{-1})}u(k) + \frac{C(q^{-1})}{D(q^{-1})}e(k) \tag{3.23}$$

or in difference equation form

$$\begin{aligned} \tilde{y}(k) + f_1\tilde{y}(k-1) + \ ... \ + f_{n_f}\tilde{y}(k-n_f) &= b_1 u(k-n_k) + \ ... \ + b_{n_b} u(k-n_k-n_b+1) \\ \nu(k) + d_1\nu(k-1) + \ ... \ + d_{n_d}\nu(k-n_d) &= c_1 e(k-1) + \ ... \ + c_{n_c} e(k-n_c) + e(k) \\ y(k) &= \ \tilde{y}(k) + \nu(k) \end{aligned} \tag{3.24}$$

This structure is also denoted as BJ(n_b, n_c, n_d, n_f) with n_k as delay.

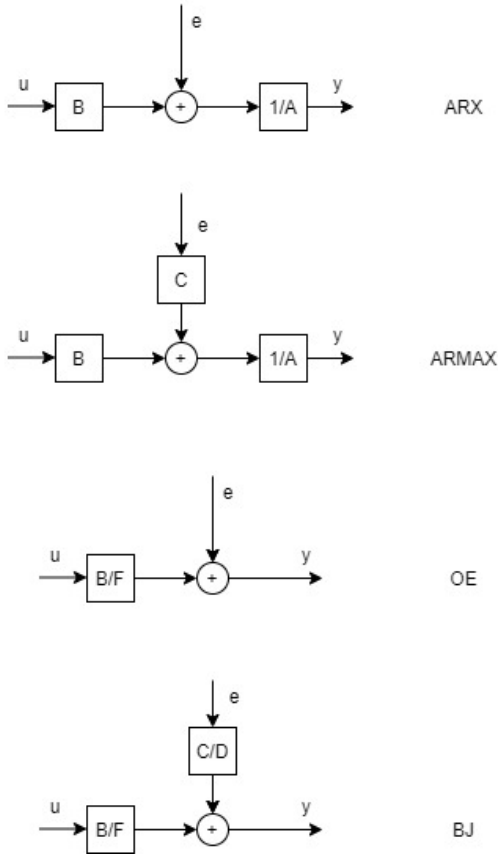


Figure 3.5: Linear Black-Box Model structures

3.1.2 Nonlinear and SISO systems

Models for linear time invariant SISO type systems have been analyzed. A linear model is often sufficient to capture the dynamics of the system but sometimes nonlinear models are required. When does it happen? [54]

- Linear model is not good enough, i.e. poor fit between measured output and model output is provided.
- Weakly nonlinear characteristics affect the system, for which dynamics is not well captured.

- Physical system is inherently non linear, such as for dry friction, saturation.

For each of these situations, a non-linear solution is offered: NARMAX, NARX, Hammerstein-Wiener models, Nonlinear State-Space models, etc.

A **NARMAX** model is defined as

$$\begin{aligned} y(k) = \; & F(y(k-1),\ y(k-2),\ ...,\ y(k-n_y), \\ & u(k-n_k),\ u(k-n_k-1),\ ...,\ u(k-n_k-n_u), \\ & e(k-1),\ e(k-2),\ ...,\ e(k-n_e)) + e(k) \end{aligned} \tag{3.25}$$

where $y(k)$, $u(k)$ and $e(k)$ are the system output, input and noise, respectively; n_y, n_u and n_e are the orders for system output, input and noise respectively; $F(\cdot)$ is a nonlinear function and n_k the system delay [8].

A **NARX** model is defined as

$$\begin{aligned} y(k) = \; & F(y(k-1),\ y(k-2),\ ...,\ y(k-n_y), \\ & u(k-n_k),\ u(k-n_k-1),\ ...,\ u(k-n_k-n_u)) + e(k) \end{aligned} \tag{3.26}$$

where again $y(k)$, $u(k)$ and $e(k)$ are the system output, input and noise respectively, n_y and n_u are the orders for system output and input, $F(\cdot)$ a nonlinear function and n_k the system delay [1].

Hammerstein-Wiener models describe dynamic systems using a nonlinear and linear blocks in series as shown below [54]:

$$u(k) \rightarrow \textit{ input nonlinearity } F \rightarrow \textit{ linear block} \rightarrow \textit{ output nonlinearity } H \rightarrow \ y(k) \tag{3.27}$$

When the model contains only nonlinearity in input, it is called a Hammerstein model; when only in output, it is called a Wiener model. It is particularly useful when there is the need to model a nonlinear element while keeping the dynamics of the system linear. For example, for modeling of saturations or almost linear inputs/outputs that can be approximated with a piece-wise linear function [54].
Many types of model structures are available to approximate the unknown function $F(\cdot)$, such as polynomial models, neural networks, fuzzy logic-based models, wavelet expansions [8].

3.1.3 Nonlinear and MIMO systems

In subsection 3.1.1 the linearity of the system and a SISO structure of it have been assumed. In subsection 3.1.2 the discussion has been extended to nonlinear systems; in the following section, MIMO or multi-input multi-output systems are considered. Focus the attention on a system with m outputs and r inputs. In analogy with the equation (3.25), it is possible to write [6]

$$\begin{aligned} \boldsymbol{y}(k) = \; & \boldsymbol{F}(\boldsymbol{y}(k-1),\ \boldsymbol{y}(k-2),\ ...\ ,\ \boldsymbol{y}(k-n_y), \\ & \boldsymbol{u}(k-n_k),\ \boldsymbol{u}(k-n_k-1),\ ...\ ,\ \boldsymbol{u}(k-n_k-n_u), \\ & \boldsymbol{e}(k-1),\ \boldsymbol{e}(k-2),\ ...\ ,\ \boldsymbol{e}(k-n_e)) + \boldsymbol{e}(k) \end{aligned} \tag{3.28}$$

where

$$\begin{aligned} \boldsymbol{y}(k) &= [y_1(k) \ ... \ y_m(k)]^T \\ \boldsymbol{u}(k) &= [u_1(k) \ ... \ u_r(k)]^T \\ \boldsymbol{e}(k) &= [e_1(k) \ ... \ e_m(k)]^T \end{aligned} \tag{3.29}$$

are the system output, input and white noise respectively; n_y, n_u and n_e are the orders for system output, input and noise; $\boldsymbol{F}(\cdot)$ is a matrix of nonlinear functions $F_1(\cdot)$, $F_2(\cdot)$, ..., $F_m(\cdot)$ that can be implemented based on any of the model types discussed and expressed in a more extended form as [104]

$$\begin{aligned} y_1(k) = \ & F_1(y_1(k-1), \ ... \ , \ y_m(k-n_y), \\ & u_1(k-n_k), \ ... \ , \ u_r(k-n_k-n_u), e_1(k-1), \ ... \ e_m(k-n_e)) + e_1(k) \\ y_2(k) = \ & F_2(y_1(k-1), \ ... \ , \ y_m(k-n_y), \\ & u_1(k-n_k), \ ... \ , \ u_r(k-n_k-n_u), e_1(k-1), \ ... \ , \ e_m(k-n_e)) + e_2(k) \\ & ... \\ y_m(k) = \ & F_m(y_1(k-1), \ ... \ , \ y_m(k-n_y), \\ & u_1(k-n_k), \ ... \ , \ u_r(k-n_k-n_u), e_1(k-1), \ ... \ , \ e_m(k-n_e)) + e_m(k) \end{aligned} \tag{3.30}$$

3.2 Identification process

The problem of System identification is to find the optimal estimate of the system given N observations of the input-output data. Estimation is about inferring unobserved data from a given information set using a mathematical relationship and a precise estimate criterion [104]. Important elements are

- the information set of input-output data Z
- the model $\mathcal{M}$ chosen (see subsection 3.1.1, subsection 3.1.2 and subsection 3.1.3)
- the *objective function* J, a mathematical statement used to establish the criterion for the "best" selection
- the *predictor*, a mathematical entity that generate the estimation through the previous elements

The parameter or vector of parameters to be estimated is denoted by θ, the estimated one with $\hat{\theta}$. It is

$$\hat{\theta} = \ g(Z) \tag{3.31}$$

where $g(Z)$ implicitly depends on J and $\mathcal{M}$.
Suppose the model class and its orders have been identified. Consider a set Z of N measures such that

$$Z = \{u(k), y(k)\}_{k=1}^{N} \tag{3.32}$$

and indicate with

$$\hat{y}(k, \ \theta) = f(\theta, \ Z^{k-1}) \tag{3.33}$$

the predictor resulting from the model. It is possible to express it in a form more suitable for the estimation methods application: for example, the ARX structure in equation (3.18) can be rewritten, underlining the *regressors* vector, as

$$\phi(k) = [-y(k-1),\ \dots\ , -y(k-n_a),\ u(k-n_k),\ \dots\ ,\ u(k-n_k-nb)]^T$$

and the unknown parameters vector

$$\theta = [a_1,\ \dots\ ,\ a_{n_a},\ b_1,\ \dots\ ,\ b_{n_b}]$$

as

$$\hat{y}(k,\ \theta) =\ \phi(k)^T\theta \tag{3.34}$$

In order to fit the calculated values $\hat{y}(k,\theta)$ as near as possible to the measured outputs, a series of criteria can be used:

- *Least squares* (LS) criterion: minimization of the sum of squared errors, where an error is the difference between an observed value and the fitted one proposed by the model.
- *Minimum mean square* (MMSE) criterion: minimization of the mean square error of the fitted value.
- *Maximum likelihood* (ML) criterion: maximization of the likelihood function, a measure of the agreement of the selected model with the observed data.

They involve in the minimization of the objective function J and in the identification of θ such that

$$\hat{\theta} = arg\ min_\theta\ J(\theta,\ Z) \tag{3.35}$$

The general term *Prediction Error Methods* or PEM is used for the family of estimation methods with an approach like the one in the relationship (3.35) [95].
The extension to multi-variable or non linear cases is possible by redefining the regression vector $\phi(k)$. A vector in the form of equation (3.34) is known as *linear regressor*, but other possibilities can be explored.
For nonlinear model such as NARX model, for example, a different function can be chosen: recalling the general form in equation (3.33), it can be written that

$$\hat{y}(k,\ \theta) =\ f(\theta,\ Z^{k-1}) =\ f(\phi(k),\ \theta) \tag{3.36}$$

where $\phi(k)$ is the regression vector. It creates a map between the past inputs and outputs, while the nonlinear function f creates a map between the regressor and the output space. The former can be also expressed as

$$f(\phi(k),\ \theta) =\ \sum a_k f_k(\phi)$$

where f_k are referred as *basis functions*. The key point of the identification is their selection: usually f_k is formed by a *mother basic function* and the other are generated by dilation and scaling of this. Examples are the Fourier series and Piecewise Constant Functions, but also Wavelets and Neural Networks [53].
Model structure is usually determined by the choice of the regression vector, the basic function and the number of elements or *nodes* in the expansion. Once this is done, the following steps need to be performed:

- From observed data, estimate the predictor $\hat{y}(k,\ \theta)$
- Calculate the difference between the measured output and the predictor: this is known as *prediction error*

$$e(k,\ \theta) =\ y(k) - \hat{y}(k,\ \theta)$$

- Choose a function to measure the norm of the prediction error
- Minimize the sum of these norms to find θ

If $\hat{y}(k,\ \theta)$ is not linear, minimization must be done with numerical search procedure [53]. The followed iterative scheme is like

$$\hat{\theta}_{i+1} = \hat{\theta}_i - \mu_i R_i^{-1} \hat{g}_i$$

where $\hat{\theta}_i$ is the parameter estimate after i iteration, μ_i the step size, $\hat{g}_i$ the estimate of the gradient of the function J and R_i a matrix that modifies the search direction. According to the expression of R_i, different search methods can be identified:

- Gradient method
- Gauss-Newton method
- Levenberg-Marquard method

The estimation of θ is linked to the initial choice of the model: how is it possible to "quantify" the correctness of that choice?

3.3 Model validation

Model quality is tested in the *validation* step. On the one hand, there is the need to ensure that the model output actually matches the working data (a small prediction error) through tests such as the *residual analysis* or fitting, and on the other hand that it accurately responds to a new data set, defined as *validation data*, by performing a *cross-validation* [104].
The *residual analysis* quantify the goodness of a model observing the correlation between the residuals and the time-shifted inputs. A significant correlation implies that the effects of input on system have not been completely explained. *Auto-correlation* of residuals is also useful to understand if the stochastic part of the model has been modeled in the correct way: if non-zero correlation occurs, this means that a predictable part has not been identified.
In order to understand if the model "works" in an acceptable way, a check about its ability to "reproduce" data must be performed. A set of validation data is used for this purpose. As reported in [53], in System identification it is common practice to simulate the model using validation data and inspect the agreement between the outputs. This agreement is usually measured by an approximation criterion. It measures the similarity between the model output and the actual system one, allowing to define how good the estimation of the system is [18].

Let us define the *cost function* $J(\theta)$ as a function of the error between the model output and the measured response. For a model with n_y outputs

$$J(\theta) = \frac{1}{N}\sum_{1}^{N}\varepsilon(t,\hat{\theta}_N)(\varepsilon(t,\hat{\theta}_N))^T \tag{3.37}$$

with N number of values in the estimation data set, $\varepsilon(t)$ a $n_y \times 1$ vector of prediction errors, $\hat{\theta}_N$ the estimated parameters. This expression is also known as *Loss function*. A group of criteria, also known as *information criteria*, are introduced as aids for choosing between models:

- *FitPercent*, or Normalized Root Mean Squared Error expressed as a percentage, defined as

$$FitPercent = 100\Big(1 - \frac{||y-\hat{y}||}{||y-\bar{y}||}\Big) \tag{3.38}$$

  with y measured output data, $\hat{y}$ simulated response of the model, $\bar{y}$ mean of the measured data, $||\cdot||$ the 2-norm of the vector [3].

- *Akaike's information criterion* or AIC, it describes the tradeoff between accuracy and complexity in model construction [18]; it is given by

$$AIC = N\ log(det(J)) + 2n_p + N(n_y\ log(2\pi) + 1)) \tag{3.39}$$

  with N number of values in the estimation data set, V loss function, n_p the number of estimated parameters and n_y the number of model outputs [3]. The most accurate model has the smallest AIC.

- *Final Prediction Error* or FPE,

$$FPE = det(J)\frac{1+n_p/N}{1-n_p/N} \tag{3.40}$$

  where N is again the number of values in the estimation data set, V the loss function and n_p the number of estimated parameters. The most accurate model has the smallest FPE [28]

- *Minimum Description Length Criterion* or MDL, according to which the best model is the one with good predictive performance on unseen data [18];

- *Bayesian Information Criteria* or BIC

$$BIC = N\ log(det(J)) + N(n_y\ log(2\pi) + 1)) + n_p log(N) \tag{3.41}$$

Chapter 4

Guidance, Navigation and Control

The path planning might be defined as: "A finding safe obstacle-free road from initial state to target" [49]

Path planning is a necessary skill that an autonomous mobile robot must possess to successfully accomplish its missions. It is a complex task that requires the processing of all the available information from sensors in order to trace the most appropriate and safe path from a starting point to a final one.
In the process of path panning it is necessary to define [78]

- The *state*: a model of the robot and some characteristics from the environment in which it operates. From the robot, one must mainly consider its degrees of freedom, mathematical model and sensors' errors. The environment can be identified by the number and complexity of the obstacles as well as the possible enviromental uncertainty.

- The *action*: guiding/tracking/control algorithms and their effects. It is related to the knowledge of the kinematic and dynamic constraints and uncertainty in the robot's operations.

For mobile robots it is essential to develop a path planning algorithm efficient in both computational time and complexity, since it will be executed on board by embedded processors with a specific computational power.

4.1 Navigation

Localization is the problem of individuation of a robot pose referenced to given landmarks [49]. A pose with respect to a coordinate frame in a 2D space is defined as

$$\boldsymbol{q} = \{X \; Y \; \psi\}^T \tag{4.1}$$

where X and Y are the Cartesian Coordinates of the CoM with respect to the Inertial System and ψ the orientation angle.
Localization can be classified as

- indoors or outdoors, depending of the target applications
  - *indoors*: man-created environment
  - *outdoors*: open terrains
- local or global, depending of the reference
  - *local*: use of local data, encoders, sensors
  - *global*: state in a global system

Usually a combination of the local and global localization is used in order to perform efficient global localization.
One of the most used localization systems is the Global Positioning System (GPS). It is a powerful instrument to obtain information about the position of an object on the Earth surface, but its use is however not allowed in indoor applications, where other methods must be exploited. Some of them are listed below:

- *Odometry*
  Information about the platform motion can be obtain from encoders attached to the motor shafts. The velocity of the robot and its rotational speed are obtained through the kinematic model presented in chapter 2: indicating with V_R and V_L the right and left speed of the two tracks

$$\begin{aligned} V_{xenc} &= \frac{1}{2}(V_R + V_L) \\ \dot{\psi}_{enc} &= \frac{1}{B}(V_R - V_L) \end{aligned} \tag{4.2}$$

  with r and B radius of the wheels and distance between track's centerline, respectively.
  The estimation of the robot pose by integration of these variables is referred as *odometry*. Odometry is a simple method to obtain the location of the robot, but reliable only in the short term due to the accumulation of errors. These can arise due to the approximation made in the models, e.g. wrong radius of the wheel, inaccuracy in the calculation of the geometric parameters, or due to measurement error, for example in presence of slippage [48]. A filtering action is required for long term navigation.

- *Inertial Measurement*
  Accelerometers and gyroscopes are used to measure accelerations and angular velocities. A double integration of the information from the sensors provides robot position and orientation, while a single integration linear and angular velocities. The use of the integration action is the main cause of errors accumulation, so again, the use of a filter is required to improve navigation tasks.

- *Orientation Measurement*
  Several sensors can be used to estimate the orientation of the robotic platform, such as magnetometers, gyroscopes and accelerometers. Magnetometers provide an absolute measurement of the Earth's magnetic field. An estimation of the orientation angle can be obtained through the relationship

$$\psi_{magn} = atan\Big(\frac{-B_y}{B_x}\Big) \tag{4.3}$$

  with B_x and B_y components of the magnetic vector. An accurate orientation estimate is important to assure better performance of odometry or inertial navigation [48].

4.1.1 Kalman Filter

A filter is an algorithm for the processing of noisy input data in order to generate accurate estimates of unknown variables. Kalman filters are ideal for the use in real-time applications and embedded systems due to their low computational cost [57]. The algorithm uses a series of measurements with statistical noise and inaccuracies and produces an estimation of the unknown states. It is an optimum filter in the sense that it is obtained by the minimization of the mean square error of the estimate with respect to the real value [81].
Let us describe the dynamics of a system with the expression

$$\boldsymbol{x}(k) = \boldsymbol{F}(k)\boldsymbol{x}(k-1) + \ \boldsymbol{B}(k)\boldsymbol{u}(k) + \ \boldsymbol{w}(k) \tag{4.4}$$

where

- $\boldsymbol{x} \in \mathbb{R}^N$ is the system *state* vector
- $\boldsymbol{F}(k)$ is the *state transition model* applied to the previous state $\boldsymbol{x}(k-1)$
- $\boldsymbol{u} \in \mathbb{R}^M$ is the *control vector*
- $\boldsymbol{w} \in \mathbb{R}^N$ is the *process noise*, a Gaussian random state noise vector with zero mean and covariance matrix $\boldsymbol{Q}$

The observation at time k is done according to

$$\boldsymbol{y}(k) = \boldsymbol{H}(k)\boldsymbol{x}(k) + \ \boldsymbol{v}(k) \tag{4.5}$$

where

- $\boldsymbol{y} \in \mathbb{R}^R$ is the *measurements* vector
- $\boldsymbol{H}(k)$ denotes the *observation model*
- $\boldsymbol{v} \in \mathbb{R}^R$ is the *observation noise*, a Gaussian random measurement noise vector with zero mean and covariance matrix $\boldsymbol{R}$

The initial state and noise vectors at each step are assumed to be mutually independent. The algorithm works in two steps:

- *Prediction step*: an estimation of the current state variables is produced by using the state estimate from the previous step. This is known as *a priori* estimate, indicated with the notation $\hat{\boldsymbol{x}}(k|k-1)$, because it doesn't include observation of the current time step. An *a priori* estimation of the error covariance matrix $\boldsymbol{P}(k|k-1)$ is also calculated.
  A priori state estimate

$$\hat{\boldsymbol{x}}(k|k-1) = \boldsymbol{F}(k)\hat{\boldsymbol{x}}(k-1|k-1) + \boldsymbol{B}(k)\boldsymbol{u}(k) \tag{4.6}$$

  A priori error covariance

$$\boldsymbol{P}(k|k-1) = \boldsymbol{F}(k)\boldsymbol{P}(k-1|k-1)\boldsymbol{F}(k)^T + \boldsymbol{Q}(k) \tag{4.7}$$

- *Update step*: the estimation is updated using weighted average on the basis of the observed measurement. This estimate is known as *a posteriori* estimate, indicated with the notation $\hat{\boldsymbol{x}}(k|k)$. The *a posteriori* estimate of the error covariance matrix is instead indicated as $\boldsymbol{P}(k|k)$.
  Innovation

$$\boldsymbol{e}(k) = \boldsymbol{y}(k) - \boldsymbol{H}(k)\hat{\boldsymbol{x}}(k|k-1) \tag{4.8}$$

  Kalman gain

$$\boldsymbol{K}(k) = \boldsymbol{P}(k|k-1)\boldsymbol{H}(k)^T(\boldsymbol{H}(k)\boldsymbol{P}(k|k-1)\boldsymbol{H}(k)^T + \boldsymbol{R}(k))^{-1} \tag{4.9}$$

  A posteriori estimate

$$\hat{\boldsymbol{x}}(k|k) = \hat{\boldsymbol{x}}(k|k-1) + \boldsymbol{K}(k)\boldsymbol{e}(k) \tag{4.10}$$

  A posteriori error covariance

$$\boldsymbol{P}(k|k) = (\boldsymbol{I} - \boldsymbol{K}(k)\boldsymbol{H}(k))\boldsymbol{P}(k|k-1) \tag{4.11}$$

The algorithm is recursive.
The optimality of the filter is guaranteed if the system model is linear and precisely known a priori, with process and measurement noises characterized by zero mean, completely unrelated, with known covariance matrices. In practice, this is not always possible [57] since

- Systems are usually nonlinear.
- System models may have inaccuracies in the parameters that characterize them.
- Process and measurement noises may not be white.
- Covariance matrices are not known a priori.

In particular, the main challenge in Kalman filter creation is the identification of the statistical properties of the system, i.e. the knowledge of the process and measurement noise matrix. There is usually no way to directly isolate the noise from measurement signals and this could lead to the wrong choice of $\boldsymbol{Q}$ and $\boldsymbol{R}$, thus degrading the performance of the state estimation [57]. However, some solutions are available for the problems reported above:

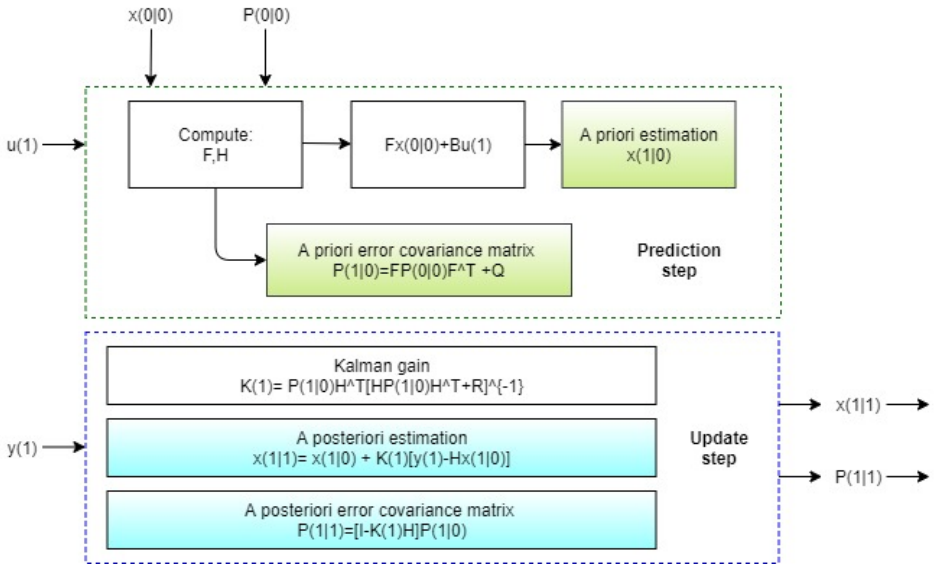


Figure 4.1: Kalman Algorithm

- EKF or *Extended Kalman Filter* can be used for non-linear systems: it is based on the idea of using a Taylor expansion to obtain a local linearization of the system.
- FMKF or *Finite-Model Kalman Filter* can be employed if large model uncertainties occur: it is based on the idea of restricting them by a finite set of known different models.
- UKF or *Unscented Kalman Filter* is used when non-linear and non-Gaussian systems are involved.

4.2 Guidance

Motion planning requires the determination of an optimal path in two or three dimensional domain such to avoid any collision with static or dynamic obstacles [45]. Optimality refers to the minimization of a parameter that works as a cost function for the motion like length of path, energy consumption, duration of the mission. In addition, a number of constraints generally exist, limiting the possible configurations.
The *configuration space* $\mathcal{C}$ or $\mathcal{C} - space$ is a n-dimensional space that includes all the configuration or system's state $\boldsymbol{q}$ that can be assumed by the robot in a operating world $\mathcal{W}$ [49] and it should be simple enough to maintain path planning computationally feasible [78]. The robot is denoted by $\mathcal{A}$ and can be represented as a point or as a body with a certain shape, depending on the size, and $\mathcal{A}(\boldsymbol{q})$ is the subset of the world occupied by the robot configuration $\boldsymbol{q}$. Obstacles are denoted by $\mathcal{O}_i$, $i = 1,\ 2,\ ...,\ N_o$, with N_0 their total number.
In motion planning

$$\mathcal{C} = \ \mathcal{C}_{free} \cup \mathcal{C}_{obs} \tag{4.12}$$

where $\mathcal{C}_{free}$ is the free space configuration and $\mathcal{C}_{obs}$ is the obstacles space configuration defined by

$$\mathcal{C}_{obs} = \ \{\boldsymbol{q} \in \mathcal{C} \mid \mathcal{A}_q \cap \mathcal{O} \neq 0\} \tag{4.13}$$

with $\mathcal{O} \cup \mathcal{O}_i$, so

$$\mathcal{C}_{free} = \ \mathcal{C} \setminus \mathcal{C}_{obs} \tag{4.14}$$

In the general case, computing the configuration space obstacle region is intractable and

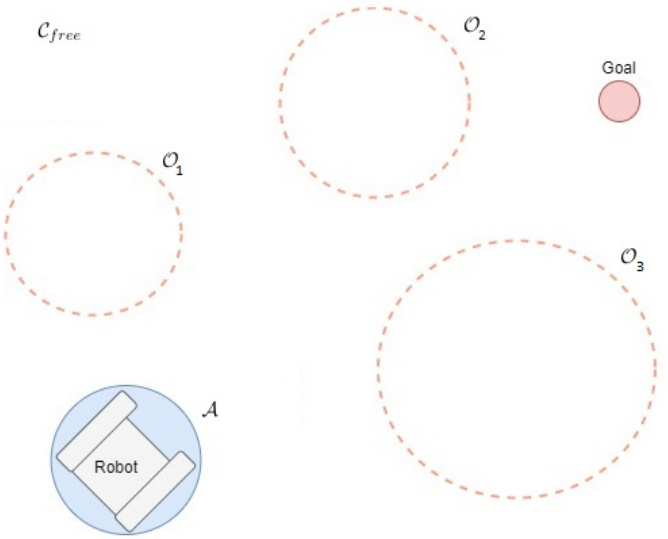


Figure 4.2: Configuration Space

one reason is linked to the exponential growth of its complexity with an increase in the number of degrees of freedom [99], so the problem of planning is instead solved without an

explicit construction of this space.
From a mathematical point of view, path planning is related to generate an optimal path from an initial configuration $\boldsymbol{q_0} \in \mathcal{C}_{free}$ to a final one $\boldsymbol{q_g} \in \mathcal{C}_{free}$ with the minimization of a cost function c [45].
There are four main situations in which the problem falls:

- Environment is perfectly known and represented by a map
- Positions of the robot and the obstacles are perfectly known
- Only simplistic geometries of the robot and the obstacles are known
- Only simplistic dynamic model of the robot is known

The focus will be on solving the path planning problem in known environments where both the position of the obstacles and that of the robot are known at all times.

4.3 Potential Field

The exploration of the $\mathcal{C}_{free}$ space can be done with a search algorithm whose aim is to discover the best path to be followed by the minimization of a cost function. It is possible to assume two cost-functions to be optimized: an *obstacle function* to help the robot to avoid the obstacles and a *goal function* that would continually attract the robot to the goal [78]. This aspect is well captured in *potential field method*.
The artificial potential field method is based on the idea of constructing a function that serves as a "surface" on which the motion occurs from a starting configuration to a final one [45]. The robot motion occurs because of the influence of an artificial potential field $\mathcal{U}$. This field consists in one component that attracts the robot to the final configuration $\boldsymbol{q}_g$ and a second one that repels it from the boundaries of $\mathcal{C}_{obs}$:

$$\mathcal{U}(\boldsymbol{q}) = \mathcal{U}_{att}(\boldsymbol{q}) + \mathcal{U}_{rep}(\boldsymbol{q}) \tag{4.15}$$

Path planning can be seen as the problem related to find the global minimum in $\mathcal{U}$, starting from the initial configuration $\boldsymbol{q}_0$, given that $\mathcal{U}$ is constructed in such a way that this minimum corresponds to the final configuration [99].
One of the easiest algorithms to solve the problem is the *gradient descent*, in which the negative gradient of $\mathcal{U}$ is calculated and considered as a force acting on the body

$$\boldsymbol{F}(\boldsymbol{q}) = -\nabla\mathcal{U}(\boldsymbol{q}) = -\nabla\mathcal{U}_{att}(\boldsymbol{q}) - \nabla\mathcal{U}_{rep}(\boldsymbol{q}) \tag{4.16}$$

One of the prerequisites for the correct functioning of the algorithm is the existence of a global minimum that coincides with the configuration of the goal and this happens only by appropriately choosing the potential fields.

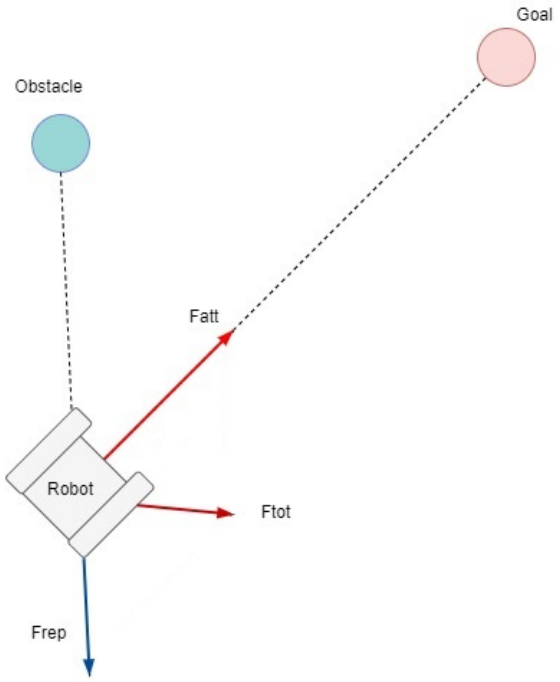


Figure 4.3: APF representation

4.3.1 Attractive Field

$\mathcal{U}_{att}(\boldsymbol{q})$ should be monotonically increasing with distance from $\boldsymbol{q}_g$ [99]. A first possible choice is a linear growth (*conic potential*) but this led to poor results; a field that grows quadratically with distance (*parabolic potential*) can instead be adopted:

$$\mathcal{U}_{att}(\boldsymbol{q}) = \frac{1}{2}K\|\boldsymbol{q}-\boldsymbol{q}_g\|^2 = \frac{1}{2}K\rho_f(\boldsymbol{q})^2 \tag{4.17}$$

where $\rho_f(\boldsymbol{q}) = \|\boldsymbol{q}-\boldsymbol{q}_g\|$ is the Euclidean distance between the actual configuration $\boldsymbol{q}$ and the final one $\boldsymbol{q}_g$ and K a parameter used to define the intensity of the attractive field. So the attractive force can be expressed as

$$\boldsymbol{F}_{att}(\boldsymbol{q}) = -\nabla\mathcal{U}(\boldsymbol{q}) = -K(\boldsymbol{q}-\boldsymbol{q}_g) \tag{4.18}$$

and it is a vector directed toward $\boldsymbol{q}_g$ linearly related to the distance. The advantage of this choice is that $\boldsymbol{F}_{att}$ linearly converges to zero as the robot approaches the final configuration. However, if on the one hand the behavior of the function is optimal, on the other, it may not be because the attractive force tends to grow when the robot moves away from the goal, risking to reach a value too high. As a consequence, it leads the robot to move too close to obstacles. A solution is the combination of the two presented fields: a conic potential when the robot is very far from the final configuration and a parabolic one when it is near it, such as

$$\mathcal{U}_{att}(\boldsymbol{q}) = \begin{cases} \frac{1}{2}K\rho_f(\boldsymbol{q})^2 & for\ \rho_f(\boldsymbol{q}) \le d \\ dK\rho_f(\boldsymbol{q}) - \frac{1}{2}Kd^2 & for\ \rho_f(\boldsymbol{q}) > d \end{cases} \tag{4.19}$$

so that

$$\boldsymbol{F}_{att}(\boldsymbol{q}) = -\nabla \mathcal{U}_{att}(\boldsymbol{q}) = \begin{cases} -K(\boldsymbol{q} - \boldsymbol{q}_g) & for\ \rho_f(\boldsymbol{q}) \leq d \\ \\ -\frac{dK(\boldsymbol{q}-\boldsymbol{q}_g)}{\rho_f(\boldsymbol{q})} & for\ \rho_f(\boldsymbol{q}) > d \end{cases} \tag{4.20}$$

where d is the distance value which discriminates against the application of one kind of potential respect to the other. The gradient is well defined at the boundary.

4.3.2 Repulsive Field

$\mathcal{U}_{rep}(\boldsymbol{q})$ should repel the robot from obstacles but should not affect motion when it is far from them [99]. One possible choice is a potential that goes to infinity at the obstacle boundaries and zero far from it. Let us define as ρ_0 the limit of the region in which the motion is affected by the presence of the obstacle:

$$\mathcal{U}_{rep}(\boldsymbol{q}) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(\boldsymbol{q})} - \frac{1}{\rho_0}\right)^2 & for\ \rho(\boldsymbol{q}) \leq \rho_0 \\ \\ 0 & for\ \rho(\boldsymbol{q}) > \rho_0 \end{cases} \tag{4.21}$$

where $\rho(\boldsymbol{q})$ is the shortest distance from the actual configuration to the obstacle space one while η is a gain coefficient used to define the intensity of the repulsive field. So the repulsive force can be expressed as

$$\boldsymbol{F}_{rep}(\boldsymbol{q}) = -\nabla \mathcal{U}_{rep}(\boldsymbol{q}) = \begin{cases} \eta\left(\frac{1}{\rho(\boldsymbol{q})} - \frac{1}{\rho_0}\right)\frac{1}{\rho(\boldsymbol{q})^2}\nabla\rho(\boldsymbol{q}) & for\ \rho(\boldsymbol{q}) \leq \rho_0 \\ \\ 0 & for\ \rho(\boldsymbol{q}) \leq \rho_0 \end{cases} \tag{4.22}$$

where

$$\nabla\rho(\boldsymbol{q}) = \frac{\boldsymbol{q} - \boldsymbol{q}_{obs}}{\|\boldsymbol{q} - \boldsymbol{q}_{obs}\|} \tag{4.23}$$

with $\boldsymbol{q}_{obs}$ is the point in $\mathcal{C}_{obs}$ that is nearest to the actual configuration. The advantage is a simple solution to obstacle avoidance.

A proper choice for K and η must be done to define the intensity of the attractive and repulsive force. Furthermore, since it is highly unlikely that in reality the condition $\boldsymbol{q} = \boldsymbol{q}_g$ will be satisfied, it is possible to define a tolerance within which the objective of the mission is considered to have been achieved. The extent of this tolerance is obviously linked to the type of application.
To simplify the discussion, the passage from the space of configurations to that of the world $\mathcal{W}$ is introduced. $\mathcal{W}$ is assumed to be the Cartesian domain, so they will be indicated as X and Y the position of the robot with respect to the Inertial Frame, with X_g and Y_g the one of the goal and with X_{oi} and Y_{oi} the one of the i-esimal obstacle.
It is possible to express the attractive potential force in equation (4.18) in components in the form [67]

$$\begin{aligned} F_{attX} &= -K(X - X_g) \\ F_{attY} &= -K(Y - Y_g) \end{aligned} \tag{4.24}$$

and in a similar way for the repulsive potential force in equation (4.22)

$$F_{repX} = \begin{cases} \eta\left(\frac{1}{\rho(\boldsymbol{q})} - \frac{1}{\rho_0}\right)\frac{1}{\rho(\boldsymbol{q})^2}\frac{X - X_{obs}}{\|\boldsymbol{q} - \boldsymbol{q}_{obs}\|} & for\ \rho(\boldsymbol{q}) \leq \rho_0 \\ 0 & for\ \rho(\boldsymbol{q}) \leq \rho_0 \end{cases}$$
$$F_{repY} = \begin{cases} \eta\left(\frac{1}{\rho(\boldsymbol{q})} - \frac{1}{\rho_0}\right)\frac{1}{\rho(\boldsymbol{q})^2}\frac{Y - Y_{obs}}{\|\boldsymbol{q} - \boldsymbol{q}_{obs}\|} & for\ \rho(\boldsymbol{q}) \leq \rho_0 \\ 0 & for\ \rho(\boldsymbol{q}) \leq \rho_0 \end{cases} \tag{4.25}$$

where $\boldsymbol{q}_{obs} = \{X_{obs}\ Y_{obs}\}^T$ is the obstacle closest to the actual position and $\rho(\boldsymbol{q})$ is the Euclidean distance from them.
The total force is the sum of the attractive and repulsive forces:

$$\boldsymbol{F}(\boldsymbol{q}) = \boldsymbol{F}_{att}(\boldsymbol{q}) + \boldsymbol{F}_{rep}(\boldsymbol{q}) \tag{4.26}$$

4.4 Reference signal generation

Reference signals of velocity and orientation can be generated at each step of the path and transmitted to the Control System as desired value to achieve.

Orientation signal

The desired orientation is defined at each point of the path by the analysis of the resultant force direction. As seen in section 4.3, it is possible to obtain a representation of the attractive and repulsive forces as components in X and Y direction. The total force on X direction is obtained by

$$F_X(\boldsymbol{q}) = F_{attX}(\boldsymbol{q}) + F_{repX}(\boldsymbol{q}) \tag{4.27}$$

and in a similar way, on Y direction

$$F_Y(\boldsymbol{q}) = F_{attY}(\boldsymbol{q}) + F_{repY}(\boldsymbol{q}) \tag{4.28}$$

The desired orientation is the angle between the resultant force components in the X direction and in the Y direction. In other words

$$\psi_d = \ atan\frac{F_Y(\boldsymbol{q})}{F_X(\boldsymbol{q})} \tag{4.29}$$

where the $atan(\cdot)$ function is such that $\psi_d \in (-\pi, \pi]$, returning the angle in a 4-quadrant representation.

Velocity signal

It is desired to modify the velocity reference signal so that it takes into account where the robot is with respect to the goal and obstacles positions: the robot should go faster

when far from obstacles or from the final point, slower in their proximity in order to avoid collision or to overtake the objective. As reported in [25], a function $\Delta(\boldsymbol{q})$ is defined as

$$\Delta(\boldsymbol{q}) = \frac{K_g}{\rho_g} + \sum_{i-1}^{N_o} \frac{K_i}{\rho_{obsi}} \tag{4.30}$$

where ρ_g is the distance from the goal point, ρ_{obs_i} the one from the i-esimal obstacle with N_o the number of obstacles, K_g and K_i positive parameters. The desired velocity is obtained as

$$V_d = \frac{V_{xmax}\ \theta^h}{\theta^h + \Delta^h(\boldsymbol{q})} \tag{4.31}$$

with h a positive integer number, θ a positive parameter and V_{xmax} the maximum desired reachable velocity.
It is interesting to note that:

- V_d is exactly V_{xmax} when $\Delta^h(\boldsymbol{q})$ tends to zero: this happens when the distances between the robot and the obstacles and between the robot and the goal are very high.

- V_d tends to zero when $\Delta^h(\boldsymbol{q})$ tends to infinity: this happens when the distances between the robot and the obstacles and between the robot and the goal are very low.

- θ and h can be chosen to model the value and the way in which the speed increases or decreases in the presence of the goal and obstacles.

4.5 Limitations and solutions

The artificial potential field is a simple but effective method for path planning, appreciated for its simplicity, mathematical elegance and low computational cost [67]. However some problems can arise:

- local minima

- oscillation in the presence of obstacles

- no path between closely spaced obstacles

- oscillation in narrow passages

Solutions for these problems have been proposed. For example, the combination of an artificial potential field method and a control strategy based on harmonic functions could contribute to the elimination of local minima. Also an IAPF, which is an improved version of the APF method for environments in which dynamic obstacles are considered could be useful in rapidly changing environments. The choice, once more, depends on the application as well as on the design requirements.

4.6 Control

The design of the Control System involves aspects related to system dynamics, actuator behavior, noise characteristics [103]. It is necessary to fix the primary goal of control and identify the major restrictions, such as nonlinearities, disturbances, process uncertainty among others in order to design the best controller for the system. It is also required to quantitatively express the control specifications:

- What are the desired time or frequency response characteristics?
- How good are the tracking performance of a reference?
- Is the Controller able to face up the uncertainties of the model?
- Is it sensitive to noise?

PID control strategies are presented given PID simplicity in design, analysis and implementation [112].

4.6.1 PID Controller

The Proportional Integral Derivative PID controller is one of the most widely used control technique in industrial control systems [98]. As the name suggests, it is made up of a

- P part: proportional control, the action is based on the current behavior of the system.
- I part: integral control, the action is based on the past values of the signal.
- D part: derivative control, the action is based on the predicted future values of the signal.

A mathematical representation of the three parts is given by

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{4.32}$$

where K_p, K_i and K_d are denoted as proportional, integral and derivative gain respectively, while $e(t)$ is the input signal and $u(t)$ the output signal. The *proportional* control generates a signal

$$u_p(t) = K_p e(t) \tag{4.33}$$

proportional to the input through the gain K_p: the choice of the proportional gain must be carefully made, since it is linked with the dynamic response characteristics and the steady state error.
The *integral* control generates a signal

$$u_i(t) = K_i \int_0^t e(\tau)d\tau \tag{4.34}$$

proportional to the integral of the input through the gain K_i. The steady state error is reduced to zero when a step reference signal is applied, thus correcting the problem of a

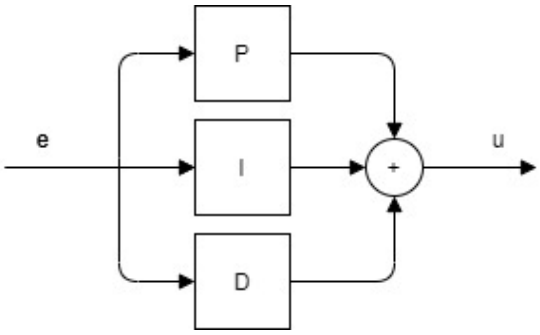


Figure 4.4: PID configuration

proportional-only control.
The *derivative* control generates a signal

$$u_d(t) = K_d \frac{de(t)}{dt} \tag{4.35}$$

proportional to the derivative of the input through the gain K_d. The derivative action mitigates the unstable behavior that an excessive value of the proportional and integrative gain can bring, improving the characteristics of the system response. Its value must be carefully chosen since it is correlated to critical issues in control practice [98], so that it is frequently unused [103].
Three different alternatives to a complete PID controller can be evaluated:

- *Proportional P controller* is usually implemented when no particular performance is required: it has the advantage of having only one parameter to be set, but it might not get rid of the steady-state error;

- *Proportional and Integrative PI controller* is usually the best choice to eliminate steady-state error. However, it has the disadvantage of accentuating the overshoot of the system response, an undesirable feature for a Control System;

- *Proportional and Derivative PD controller* is usually required to achieve stabilization or adequate damping for the closed-loop system.

The process of finding the optimal values for the PID coefficients is called *tuning*. A "manual" search of the parameters can be carried out, although in the literature there are empirical methods based on the observation of a large number of cases that help the designer in this difficult choice. The Ziegler-Nichols tuning rule, Cohen-Coon rule and the Wang-Cluett rule are widely used methods for PID tuning based on LTI system requirements. For nonlinear systems, however, these methods are not suitable and the research must be done differently.

Chapter 5

Experimental Setup

Robot: a mechanical device that works automatically or by computer control [82]

Robotics as a discipline is based on the interaction of notions deriving from the fields of electronics, mechanics, control and software. The design of a robotic system requires the study not only of high-level Software Subsystems such as the ones for Navigation or Control, but also that of Hardware Subsystems necessary for the operation of the platform such as sensors, mechanisms, materials [29]. The Devastator robotic platform is a small

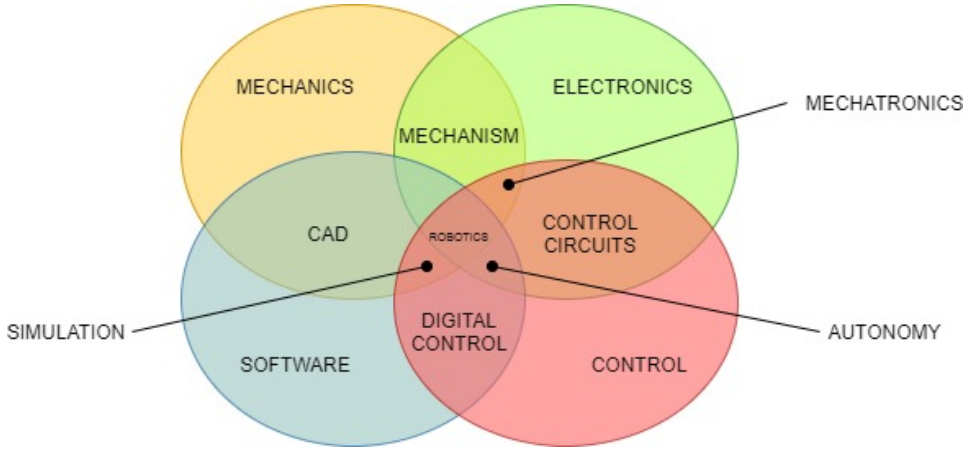


Figure 5.1: Robotics interdisciplinary

tracked vehicle made from high strength aluminum alloy and equipped with the necessary tools to carry out missions in an indoor environment in total autonomy, thanks to the presence on board of the *Lattepanda* computer and the *Freedom* microcontroller. The ground movement of the *Devastator* is allowed by the use of two DC brushed motors: each output shaft is engaged by a driving wheel on the right and left side of the robot and thanks to their rotation it is possible to have the motion of the two tracks. Consequently, through to the aid of four other driven wheels for each side, the movement of the entire platform is possible (Figure 5.3). In addition to the possibility of moving forward and backward, the use of the tracks allows the rotation in place through differential control for the two sides,

thus constituting a considerable advantage over the use of wheels in a classic configuration.

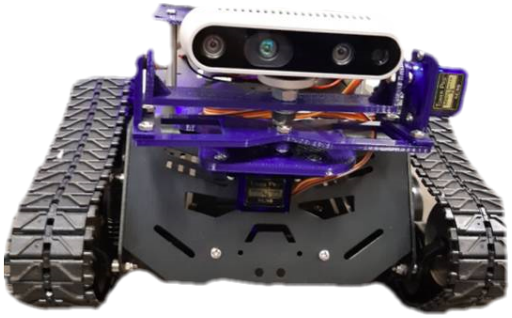


Figure 5.2: Devastator

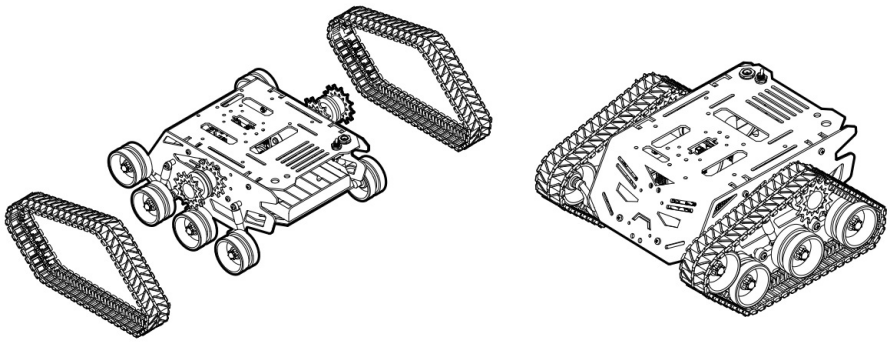


Figure 5.3: Devastator structure [21]

A list of components and on board sensors of the robot is here reported:

- *LattePanda* companion computer, it constitutes the "brain" of the system, processes the information made available by sensors and allows autonomous control of the robot.
- *NXP Freedom-k64f*, the platform which deals with the execution of lower level processes with low energy consumption, equipped with accelerometers and magnetometers.
- *LiPo battery*, it provides the energy necessary to the platform operations.
- *DC-DC module*, it allows adjustment of output voltage and current in order to supply the H-bridge with 7.5 V necessary for its ideal operation.
- *H-Bridge*, it deals with the driving of the motors and their power supply at the appropriate voltage.
- *DC brushed motors*, necessary to move the tracks and consequently the robot.
- *Encoders*, they are indispensable for obtaining information from the rotation of the driving wheels.

- *Depth detection camera*, it is employed to obtain information from the surrounding environment.

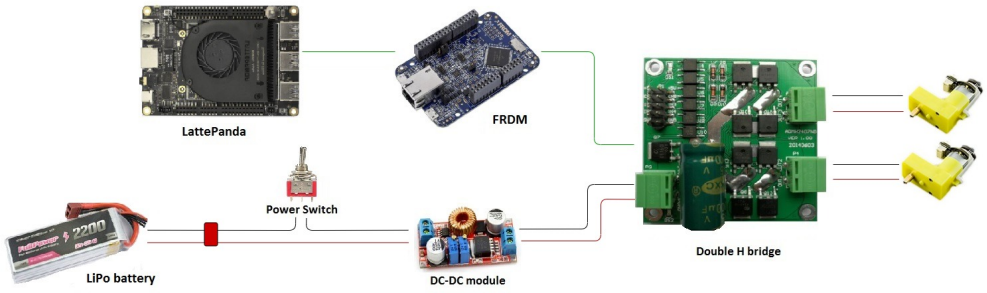


Figure 5.4: Functional scheme

5.1 More about Devastator Architecture

An *embedded* system is an electronic system designed to perform dedicated function through combined use of software and hardware. A generic embedded system is composed by

- *Microprocessor*: it deals with processing the inputs to provide the outputs on the basis of the needs required by the system.

- *Memory*: it is composed of a non-volatile part, in which the software code for the execution of the process to input the information and set the output is stored, and a volatile part, usually flash memory or read-only memory (ROM). The random-access memory (RAM) provides the run-time computation.

- *Input* and *Output*: the input is usually an information provided by sensors, the output a variable of actuation. They are managed by dedicated peripherals such as general-purpose input/output (I/O), timers, analog-to-digital converter (A/D).

When integrated in a single chip, they form a *microcontroller* [65].
The Devastator microcontroller is responsible for managing the various components of the system, including the control of the DC motors. In general, a motor drive system is composed by [47]

- *Power Supply*, i.e. the energy source of the system, usually battery.

- *Electric Motors*, chosen according to size, cost, operating condition.

- *Power Electronic Converters*, device that takes electrical energy from the power source and turning it in a form suitable to be used by the motor.

- *Digital Controllers*, based on microprocessor, to adjust the voltage provided to the motor.

- *Sensors*, necessary to obtain information about the state of the motor.

Devastator power supply is a LiPo battery chosen as a compromise to guarantee the right autonomy and ensure dimensions compatible with the platform. The installed motors are instead two geared DC brushed motors with encoders, whose characteristics have been identified during the system identification experiments. The main feature of these is their compactness and the possibility of easy integration into the structure, allowing efficient use of the space on board the robot. Regarding the practical control of motors, several suggestions are available in the literature [56]:

- *Direct driving from a microcontroller pin*: the motor is connected on one side to the voltage supply and on the other to a microcontroller digital output pin. The switching action of the pin between low and high impedance activates and deactivates the motor.

- *Current amplification with a switch*: this device, an electromechanical relay or a transistor, can be used to switch on and off the motor driving the current flow; often a protection diode is placed in the circuit to prevent the instantaneous change in current and sparking when it is opened. This configuration can be used to modulate the input voltage to the motor by rapidly changing the state of the switch. Only one direction of rotation of the motor is allowed.

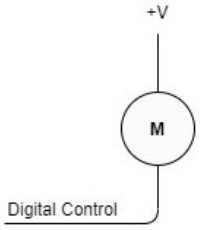
- *Bidirectional operation with switches and bipolar supply*: it is a solution to allow the motor to rotate in both directions. By employing a bipolar supply (+V,-V, GND) and two switches controlled by independent signal, it is possible to have the current flow in one direction and the other, letting clockwise or counterclockwise rotations. However, this solution is not very common due to the lack of availability of bipolar power supplies.

- *H-bridge*: it is one of the most popular choices for the motor driving. It uses a unipolar power supply but both the direction of rotation are possible thanks to the control of four switches with little voltage drop. A microcontroller is used to pilot the switches: according to Figure 5.5d,

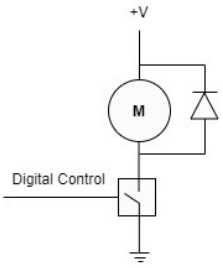
S1, S4	*Forward rotation*
S2, S3	*Reverse rotation*
S1, S3	*Short-circuit braking*
S2, S4	*Short-circuit braking*
None	*Open circuit*

  In low power applications, an integrated circuit is available.

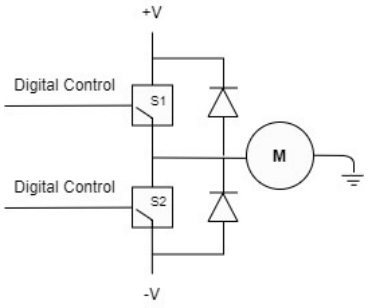
The control of the Devastator DC motors takes place through the last solution, that is the use of an H-bridge, in particular a double H-bridge giving that the controls of both the motors are integrated in a single component. In detail, the microcontroller independently drives the two motors by sending a signal through the pins to the switches in the circuits. From the combination of these signals it is possible to have operative conditions that fall into one of the previously mentioned cases, i.e. forward and reverse operation, brake or



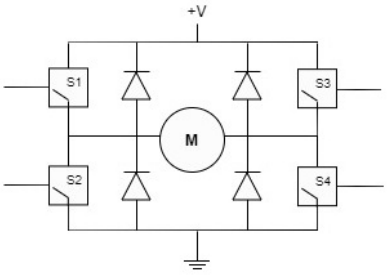
(a) *Direct driving from a microcontroller pin*



(b) *Current amplification with a switch*



(c) *Bidirectional operation with switches and bipolar supply*



(d) *H-bridge*

Figure 5.5: DC motor drive configurations

immobility. PWM signal generation is also possible thanks to the use of particular pins suitable for this application.
The goal of the *Pulse With Modulation* technique is to generate pulse signals such to produce an output voltage with the desired amplitude and frequency [47]. The Digital Control creates a square wave signal given that only two states are possible, on or off, but it is possible to get the output voltage between the power supply value and 0 by simply changing the time the signal is *on* versus the time it is off in a given sampling period. The duration of the signal *on* is called *pulse width*, while the inverse of the sampling period PWM *frequency*.

The operating condition of motors is controlled through the use of sensors. For the Devastator, two rotary encoders, one for each motor, have been installed. A *rotary encoder* is a sensor that generates digital signals as a consequence of the rotation of the shaft [47]. These signals are made up of a series of pulse that are used to measure position, speed and direction: their number is related to the resolution of the encoder. Let us consider a simple configuration to understand its working principle: an optical incremental encoder is presented. It consists in a moving disc with slits mounted on the rotating shaft, light sources and receivers: when the light emitted by the light source passes through the slits and gets to the receiver, an electric signal is generated. As result, a square-wave signal

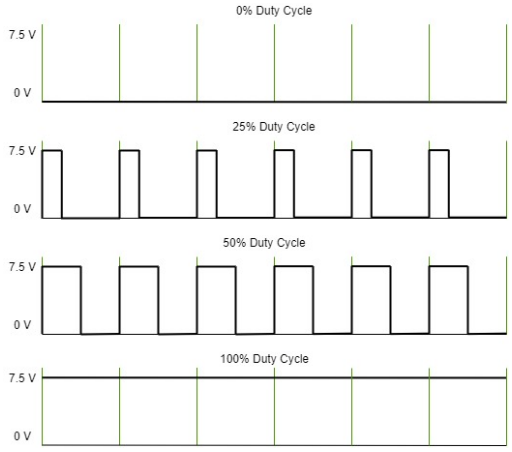


Figure 5.6: PWM duty cycle

of pulses is obtained. Usually two signals are employed: A and B. The total number of A and B pulses per revolution are used to obtain the angular position and the speed of the motor, while the comparison between them allows the identification of the rotation direction, given that they are out of phase. If the motion is forward, pulse A is ahead of pulse B; in the opposite case, the motion is reverse. There are actually various versions of encoders available in addition to the optical ones: magnetic, capacitive, inductive, etc. The working principle is almost the same.

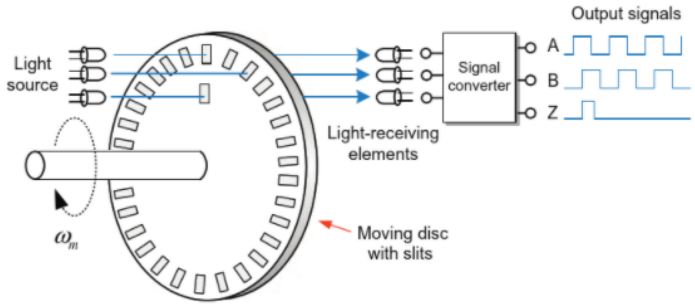


Figure 5.7: Rotary Encoder [47]

5.2 DC Motors characterization

The first step in System Identification is the characterization of the behavior of the motors. The objective is to obtain a relationship that can describe the system's output behavior based on certain inputs. In other words, the purpose of the experiments carried out on the DC motors is the identification of the necessary parameters to predict the angular speed of the driving wheel and the developed torque, so as to adequately describe the dynamics of the track and consequently that of the robot.

A lumped parameter approach has been chosen for this purpose: based on the knowledge of the motor mathematical model described in chapter 2, a series of experiments have been conducted in order to identify the main parameters and allow the creation of a Simulink model. The challenge has been that of reaching the goal by using low-cost sensors that are easily available, exploiting the potential of an Arduino microcontroller for the management of inputs and data collection. A voltage-angular velocity and voltage-torque relationship has been obtained. During the experiments, however, it was decided to change the motors and to replace them with motor units integrated with encoders: the choice was dictated by the need to include sensors capable of referring to the Control System feedback information regarding the speed of the tracks, thus allowing navigation based on odometry. This has offered the opportunity to approach the problem of identification in a new way, comparing the previously used method and the new one selected, i.e. a data-driven approach. A first section dedicated to the experiments conducted for the realization of the lumped parameter model follows. A second section concerning the data-driven approach is reported, while a third section proposing instead a comparison between the techniques and reporting the reached conclusions is also presented.

5.3 Lumped Parameters Approach

The reference model is the one presented in chapter 2 and reported below:

$$\begin{cases} V & = L\frac{di}{dt} + Ri + e \\ e & = K_{em}\omega_m \\ C_m & = K_c \\ C_m & = C_r\tau + f_{tot}\ \omega_m + J_{tot}\frac{d\omega_m}{dt} \end{cases} \tag{5.1}$$

The goal is to identify the unknown parameters in order to obtain a voltage-angular velocity and voltage-torque relationship. In light of this, it is convenient to reorganize the equations so as to highlight the inputs and outputs of the system:

- Inputs of the system are the voltage V and the resistant load C_r.
- Outputs of the system are the angular speed seen at the external shaft ω and the "useful" torque.

Some consideration must be done:

- used sensors and conducted experiments did not make possible to carefully study the transients; quantities linked to them has been neglected (as in the case of the inductance L) or indirectly estimated (as in the case of the inertia of the rotating parts J_{tot}).
- the term C_r is linked only to a possible clearly identifiable external antagonist load, while all the other phenomena not individually identifiable of friction and opposition to motion have been considered in an additional term C_0.

- great attention has been paid to the motor speed behavior in normal operating conditions rather than extreme operating situations (for very low voltage values as well as for voltages such as to bring the angular velocity close to its maximum value).

Consequently, a steady-state analysis has been conducted. Taking into account these considerations, the previously presented model is reduced to

$$\begin{cases} V &= Ri + e \\ e &= K_{em}\omega_m \\ C_m &= K_c i \\ C_m &= C_r\tau + C_0 + f_{tot}\omega_m + J_{tot}\frac{d\omega_m}{dt} \end{cases} \tag{5.2}$$

with parameters such as

- R motor resistance, to be determined [Ω]
- K_{em} Back EMF constant, to be determined [$V/rad/s$]
- K_c torque constant, to be determined [Nm/A]
- C_r resistant torque, known during the experiments [Nm]
- C_0 friction torque, to be determined [Nm]
- f_{tot} equivalent friction constant, to be determined [$Nm/rad/s$]
- J_{tot} equivalent motor inertia, to be determined [kgm^2]

The reduction gear mechanism placed as interface between the motor and the driving wheel has a gear ratio of $\tau = 1/120$.
The identification of the unknown parameters took place essentially in three steps:

- 1) Identification of the motor resistance R through the use of a bench power supply,
- 2) Measurement of the motor current, again with the help of a bench power supply,
- 3) Measurement of the angular velocity of the shaft to which the drive wheel is connected through the use of an IR sensor.

5.3.1 Experiment 1: motor resistance identification

The simplest method of finding motor resistance is blocking the shaft so that the back EMF component is canceled. By measuring through a bench power supply the steady current and knowing the power supply voltage input (a low value in order not to damage the motor), it is possible to trace the resistance of the windings. A value of R=3.1 Ω has been found.

5.3.2 Experiment 2: current measuring

Through the use of the bench power supply the motor has been supplied at different voltages and the measured current value at steady state have been recorded. The selected condition is that of zero resistant external load, with no track ($C_r = 0$). The results are shown in the table below.

Voltage *[V]*	***Current*** *[mA]*
3	170
3.5	180
4	190
4.5	200
5	210

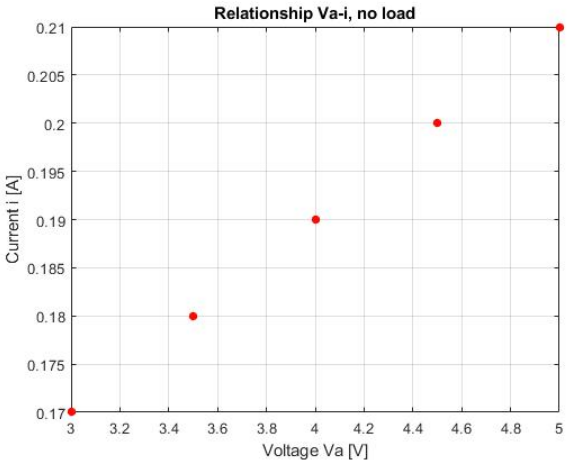


Figure 5.8: Voltage-Current characteristic

5.3.3 Experiment 3: angular speed measuring

In order to obtain the angular speed of the motor unit shaft an IR module has been used. The sensor essentially consists of an LED and an IR detector that interact with a marker positioned on the motor shaft. While the marker is made of reflective material, the surface not marked is black and inert to the action of the sensor. Since black material absorbs light, when the IR emission hits any point other than the marker, the reflected light is below the sensor activation threshold and therefore the emitted signal is a high signal and the indicator LED turns off. Conversely, when it hits the marker surface, the reflected light is such that it emits a low signal and the indicator LED turns on. The recording of these signals has been entrusted to a computer interfaced with an Arduino microcontroller, which is also necessary for sending driving signals to the motor. The angular speed of the shaft is calculated on the basis of the time that elapses between one passage of the marker and the next: in this time the shaft has made one revolution and it is therefore possible to trace its speed.
An important note concerns the sampling time: the reading from the sensor occurs every 2 milliseconds but the data on the angular velocity, being calculated on the basis of the number of marker passes, is not uniformly distributed in time. A linear interpolation was performed to obtain a more regular characteristic.

Figure 5.9: IR sensor

No load experiments
It is a series of experiments conducted with the aim of evaluating the performance of the motor at zero external resistant load. The angular velocity values in steady-state according to the voltage input are reported.

Voltage *[V]*	***Steady-state*** ω *[rpm]*
3	85.3476
3.5	101.7996
4	117.9596
4.5	132.3505
5	149.4196

Knowing the angular velocity ω, the gear ratio and the steady current i it is possible to trace the value of the back EMF constant K_{em} through the relation

$$V = R\, i + K_{em}\omega \tag{5.3}$$

getting the average value of K_{em}=0.0023 V/rad/s.

Max load experiments
It is a series of experiments conducted with the aim of evaluating the performance of the motor when the shaft is blocked, that is, evaluate the maximum torque that can be reached with a specific supply voltage (if the shaft is stationary, the contribution of the counter-electromotive force is canceled). A dynamometer has been employed. It is now possible to

Voltage *[V]*	C_m *[Nm]*
3	6.95e-04
3.5	9.0334e-04
4	9.73e-4
4.5	1.042e-03
5	1.1e-03

trace the motor torque-angular speed characteristic (Figure 5.11). From the interpolation of the points characterized by the same supply voltage it is possible to obtain the torque constant, equal to K_c=8.6626e-04 Nm/A.

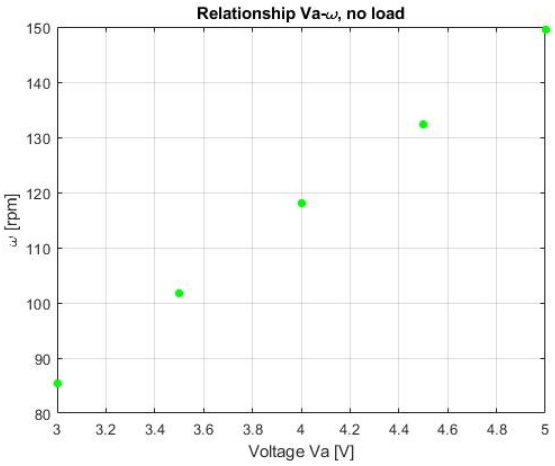


Figure 5.10: Voltage-Speed characteristic, no external load

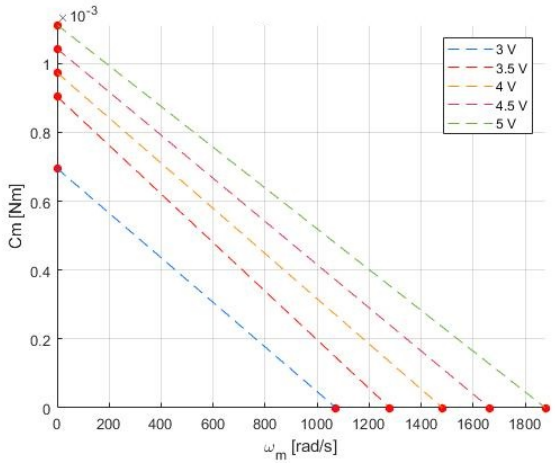


Figure 5.11: C_m-ω_m characteristic

Experiments with external loads

It is possible to express the steady-state mechanical equation such as

$$C_m = C_0 + f_{tot}\omega_m + C_r\tau \tag{5.4}$$

in which C_m is known since it is calculated as the product between the previously identified torque constant K_c and the current, in turn evaluated through the electrical relationship knowing the supply voltage, the speed reached at steady-state and the resistance; C_r is also a known quantity, so the net difference between them is equal to the unknown torque dissipated in friction and in secondary effects which are not clearly identifiable. Different experiments have been carried out introducing as a resistant load a weight hooked to the output shaft by an inextensible cable. Several voltages and six load values has been evaluated. The results of these tests are shown in the graphs of Figure 5.13 and Figure 5.14, the values of C_r in the Table 5.1. The area in evidence with respect to the background is

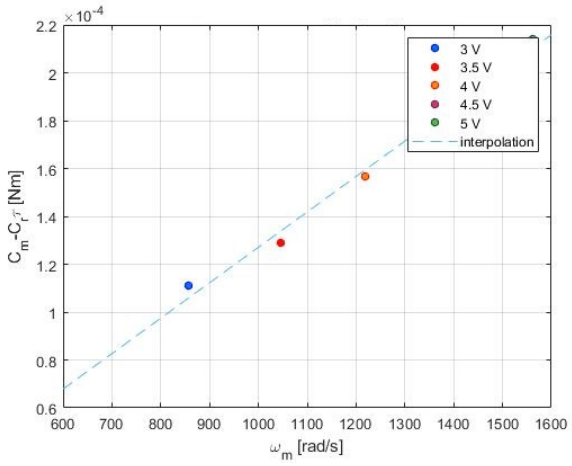


Figure 5.12: Relationship C_m-$C_r\tau$,ω

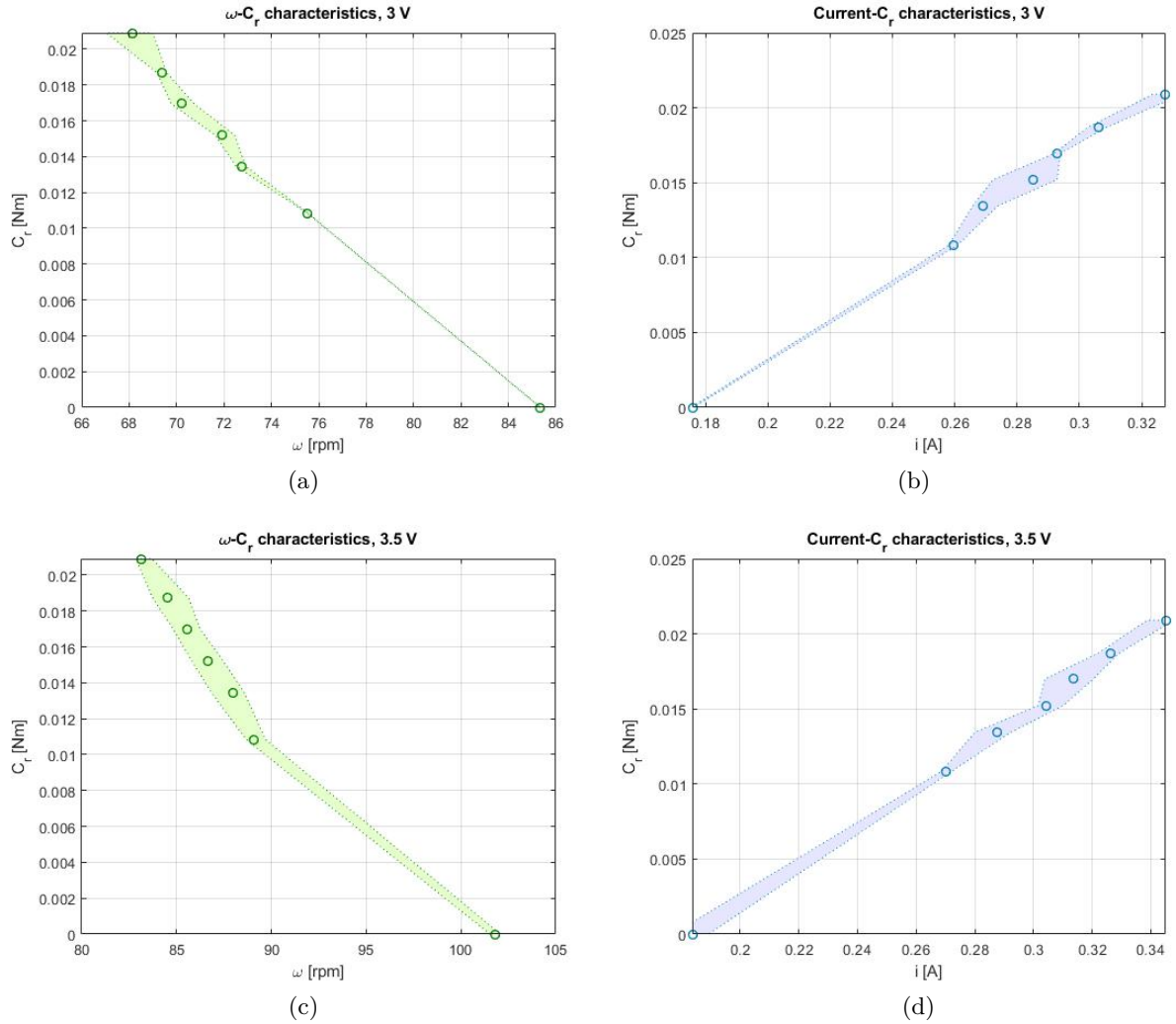
Torque C_r [Nm]	0.0108	0.0135	0.0152	0.0169	0.0187	0.0209

Table 5.1: External torques data

the region of variability of the data given by the multiple tests carried out, in other words it is a measure of the repeatability and objectivity of the data themselves.

By exploiting a least squares formulation, a linear relationship has been identified between $C_m - C_r\tau$ difference and the motor speed in order to find how they are related: in Figure 5.12 is reported the relation that has been obtained for C_r=0.0209 Nm at 3, 3.5, 4, 4.5 and 5 V. The angular coefficient is representative of the friction constant f_{tot} while the known term of the torque C_0. It can be found that f=1.0651e-07 Nm/rad/s and C_0= 2.9658e-05 Nm (mean values between the different tests).

The last parameter to identify is the inertia J: it has been estimated on the basis of the motor and system geometry and set at a value of 1e-7 kg m^2. Finally the complete characteristic of the motor is reported (Figure 5.15).



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

Figure 5.13: Tests (Part 1)

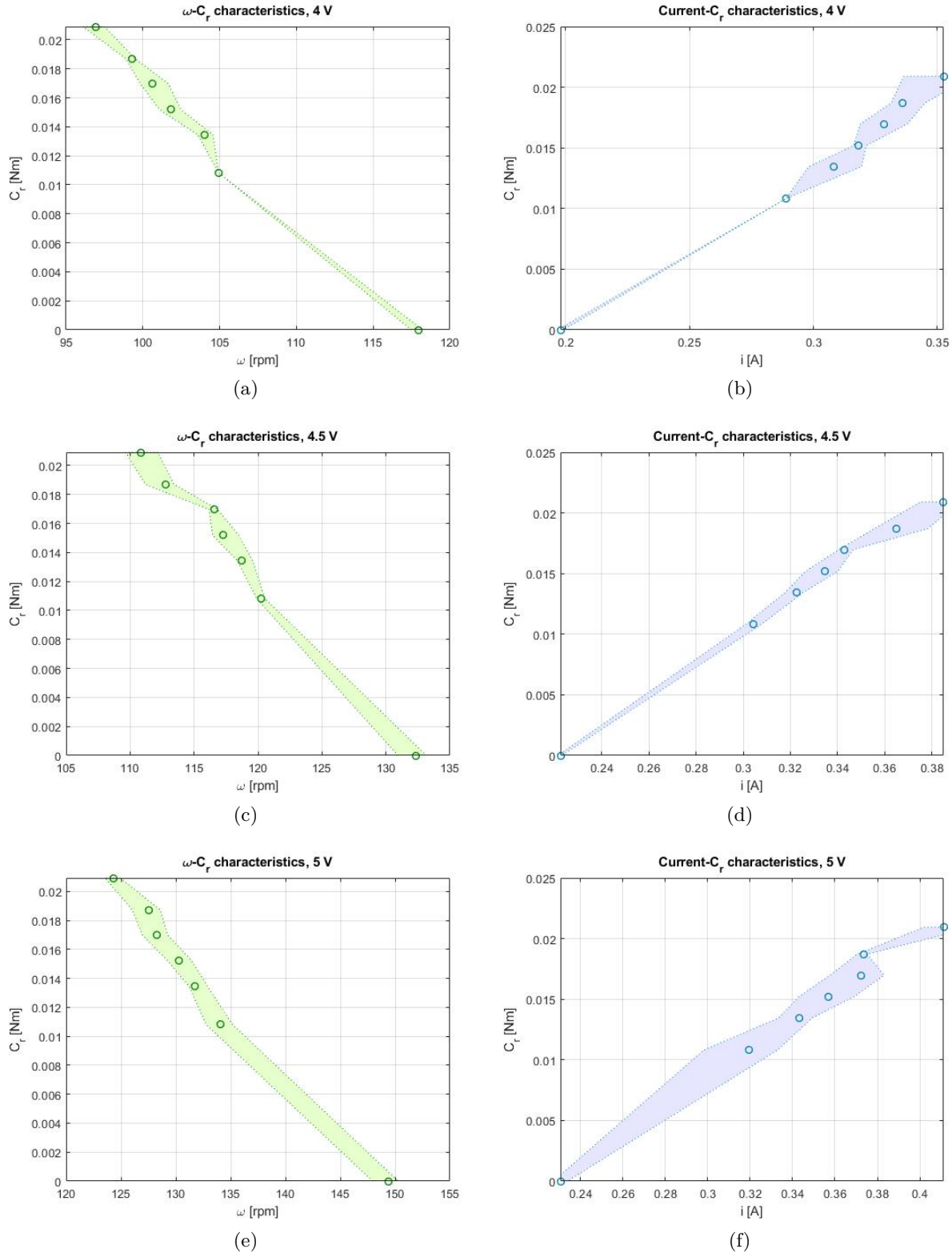


Figure 5.14: Tests (Part 2)

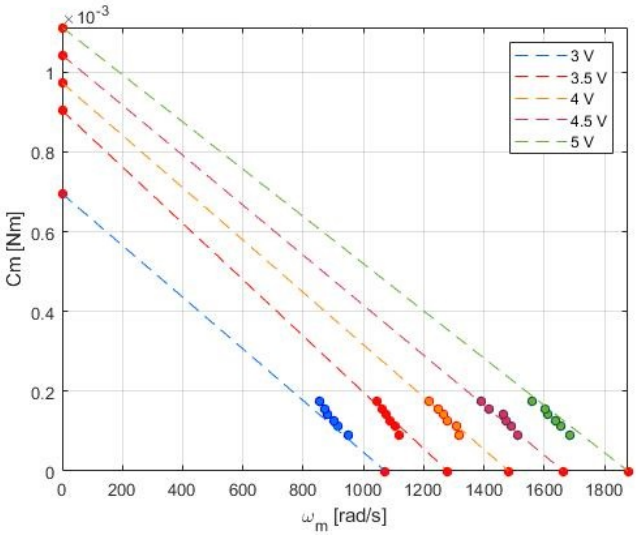


Figure 5.15: Torque-Speed characteristic

5.4 Data-driven Approach

An overview of System Identification theory has been presented in chapter 3. Methods differing in complexity, effectiveness and suitability have been analyzed. Among them, the black box modeling is the one on which the attention has been concentrated. The required procedure to identify the System Model is reported below:

- Record a data set of input-output
- Choose the class of model or model structure
- Estimate the model coefficient on the basis of certain criteria
- Obtain a model
- Validate the obtained model

5.4.1 Input-output data creation

Through the use of encoders, the response of the system in terms of angular speed of the driving wheels to a PWM input has been recorded. PWM signals have been chosen as input signals given their direct relationship with the Control System and the effectiveness of a command managed through this strategy. The angular speed of the driving wheels has been chosen as the output signal given the possibility of obtaining, through the kinematic model extensively discussed in chapter 2, both robot linear and yaw velocities. An example of input signal is shown in Figure 5.16: it is a forward command, a pause and a subsequent retraction command for the robot. A command of the type in Figure 5.17 is instead a differential command that generates a rotation on the robot's place: the PWM intensity is equal and opposite for the two motors. Input values can range from 0 to 20000 μs, both positive and negative. At a PWM input equal to 0, the platform is immobile; positive inputs lead to direct operation for the motor, negative inputs lead to retrograde operation.

The experimental data collected in output are shown in Figure 5.18 and Figure 5.19: these are the angular speeds of the right and left driving wheels.
Some observation about the *sampling time* must be done, according also with [7]:

- By increasing the sampling time, the best model tends to have more terms and require more degrees of freedom.
- By decreasing excessively the sampling time, the measures tend to be correlated and problems of bad conditioning and insufficient computational resources to record and process the data arise.
- The choice of the sampling period depends on the frequency of the system: the sampling should be such as to represent all the frequency components in the final data sets.
- Short sampling period favors the estimation of the parameters but not the correct selection of the structure; on the contrary, the selection of the structure is simplified but the accuracy of the parameters deteriorated.

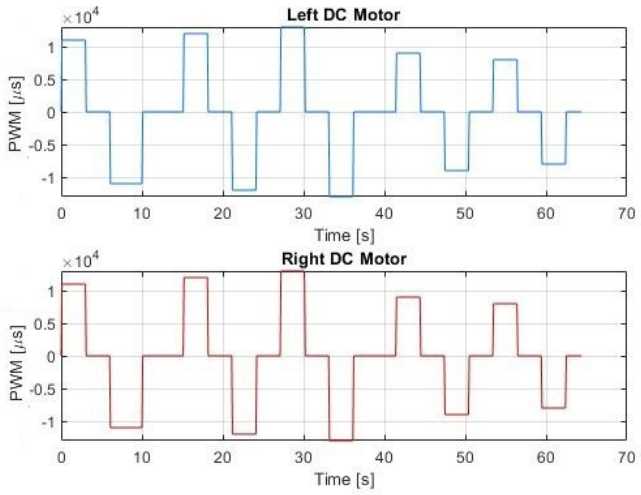


Figure 5.16: PWM input signal: forward and backward motion

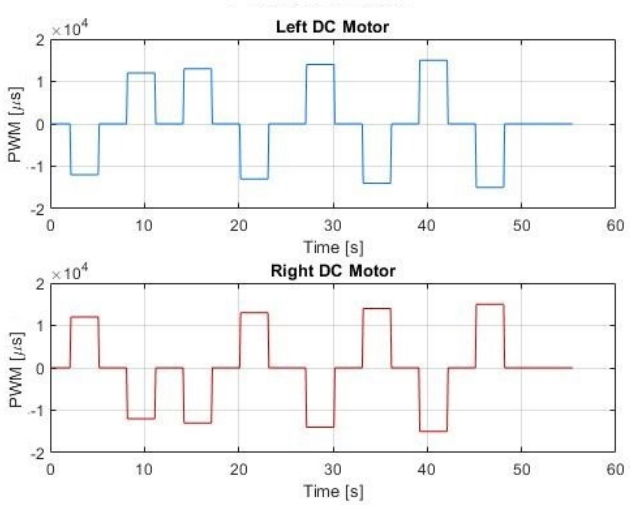


Figure 5.17: PWM input signal: rotation

Summarizing, too short sampling period means highly correlated data, i.e. *redundancy* phenomenon; too big sampling period, data jammed and chaotic, i.e. *irrelevance* phenomenon.

5.4.2 Model structure and estimation criterion selection

Model structure selection is probably among the most important step. Two paths have been explored: one based on linear identification methods and a second one on non-linear methods. For the former an ARMAX model has been selected, for the latter a NARX one.

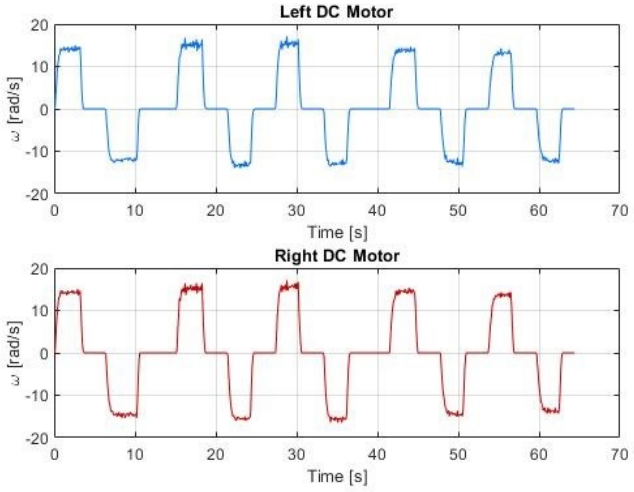


Figure 5.18: Angular speed ω output signal: forward and backward motion

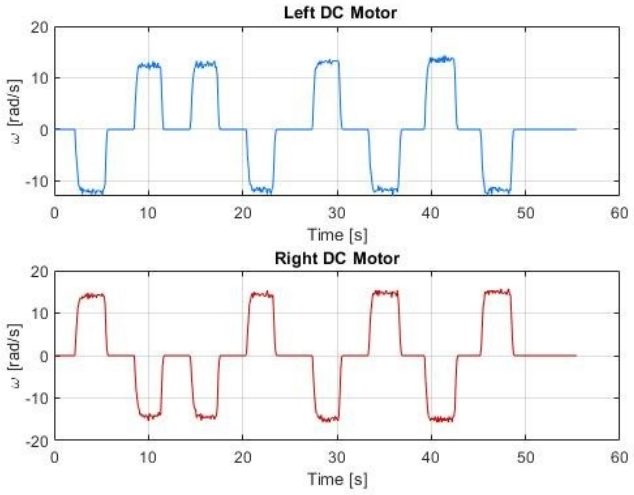


Figure 5.19: Angular speed ω output signal: rotation

According to what reported in chapter 3, the general structure of an ARMAX model is

$$y(k) = \frac{B(q^{-1})}{A(q^{-1})}u(k) + \frac{C(q^{-1})}{A(q^{-1})}e(k) \tag{5.5}$$

also denoted as ARMAX(n_a, n_b, n_c) with n_k as delay. For multi-input multi-output cases, i.e. when y and u are N_y and N_u column vectors, A, B and C are matrix of row vectors where each terms is a coefficients of the polynomial that relates input, output and noise. In particular [54]

- $A(i,j)$ contains coefficients of the relation between output y_i and output y_j. n_a is a $N_y - by - N_y$ matrix containing the degree of the corresponding A polynomial.

- $B(i,j)$ contains the coefficients of the relation between output y_i and input u_j. n_k is a $N_y - by - N_u$ matrix containing the input delay, while n_b is also a $N_y - by - N_u$ matrix.

- $C(i)$ contains the coefficients of the relation between output y_i and noise e_i. n_c is a $N_y - by - 1$ matrix containing the degree of the corresponding C polynomial.

A NARX model is instead represented by the structure

$$\begin{aligned} y(k) = \; & F(y(k-1),\ y(k-2),\ ...,\ y(k-n_y), \\ & u(k-n_d),\ u(k-n_d-1),\ ...,\ u(k-n_d-n_u)) + e(k) \end{aligned} \tag{5.6}$$

In a different way, it can be seen as the set of two components as reported in Figure 5.20 [54]:

- Regressors block, where a relationship is calculated based on the current and the past input values and past output data.

- Nonlinearity Estimator block, which maps the regressors to the model output using a combination of nonlinear and linear functions.

For the case in exam, a standard regressor has been selected, while a wavelet network has been chosen as nonlinear estimator. A PEM approach has been applied for the analysis.

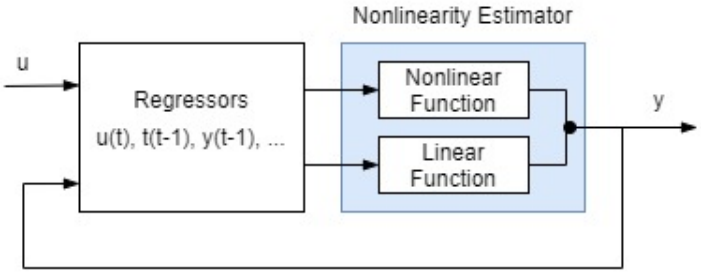


Figure 5.20: NARX structure

5.4.3 Identification procedure: operative guidelines

MATLAB System Identification Toolbox[1] has been used for the identification procedure. The essential steps listed above are reformulated based on the steps followed in the MATLAB application:

- *Loading of the input-output data*

[1]For more information [54]

  - Input data → PWM values
  - Output data → ω_L and ω_R, angular speed of the left and right drive wheels
  - Selection of the sampling interval
  - Selection of the starting time

- *Organization of the estimation data set*
  - Creation of *Working Data*
  - Creation of *Validation Data*
    Two different data sets or alternatively a single split data set.
  - Data preparation: normalization, possible removal of outliners, media, etc.

- *Estimation of Parametric Model*
  - Choice of the model structure
  - Choice of the model characteristics (orders, regressor proprieties)
  - Choice of the estimation method: PEM approach is the one adopted in this case

- *Validation*
  - Validation by using the *Validation Data*
  - Validation criteria evaluation
  - Comparison between models on the basis of the selected criteria

After several attempts, the need to turn to a non-linear identification method has emerged. The linearity path has proved unsuitable for the system in question given the many non-linearities that characterize it, such as friction between track and ground, PWM-voltage relationship, wheel speed saturation, etc.

5.5 Results and Considerations

The model with the best results obtained from the procedure is reported, where *the best* means that best satisfied the selected validation criteria (FIT, AIC, FPE, residue analysis). It is a NARX model with

$$n_a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad n_b = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad n_k = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

and

FitPercent	[91.474; 91.8968]
LossFcn	1.2013e-06
FPE	1.2013e-06
AIC	-4.2330+04

Some validation tests are reported in Figure 5.21, Figure 5.22, Figure 5.23.

It is possible to notice that the model quite faithfully simulates the behavior of the robot,

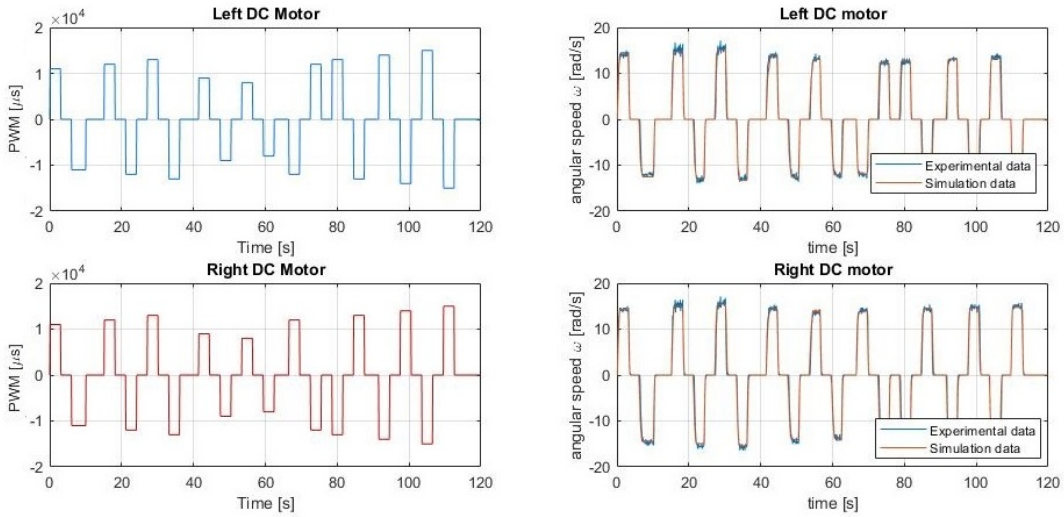


Figure 5.21: Validation test 1

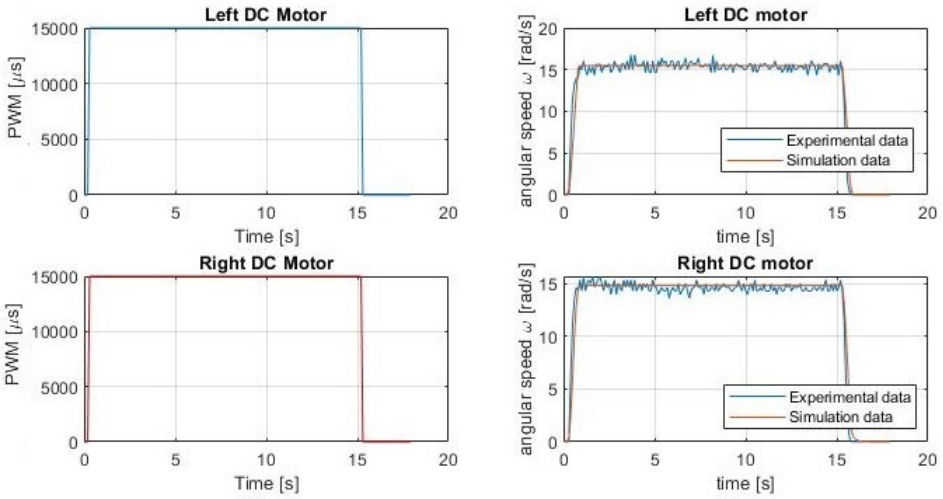


Figure 5.22: Validation test 2

both when it moves forward and backward and when it turns. It slightly overestimates its performance in reverse motion, where it provides a more immediate response to PWM input which, in reality, comes with a slight delay. The most reliable results are those concerning the PWM input range from 5000 to 18000, resulting instead less precise in the two areas of very low and very high PWM. This is not surprising: at very low PWM, the

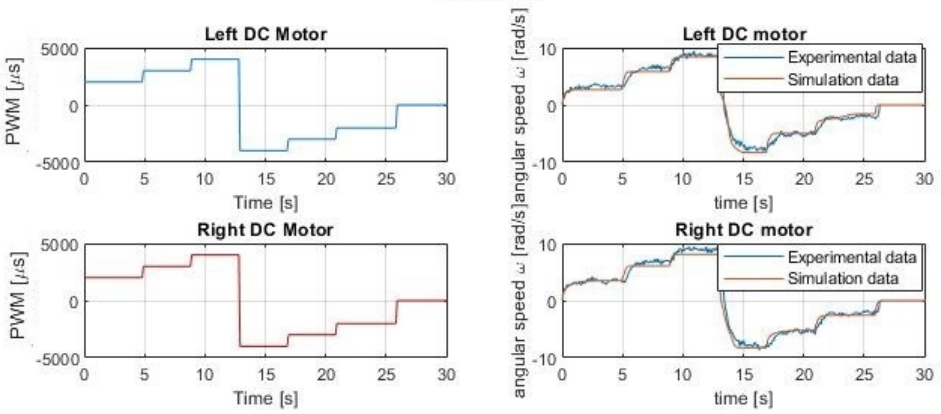


Figure 5.23: Validation test 3

friction and resistances in general that the motors must overcome to move the platform are responsible for a behavior that is difficult to predict; at high PWM, on the other hand, there are saturation phenomena.
However, the model has been accepted with the foresight to prevent reaching the maximum speed under standard operating conditions and by assuming that the GNC is robust enough to guide the robot even in the area of very low PWM in which the simulation tests are ineffective.

5.6 Comparison

Interesting points of discussion can be drawn from the comparison between the two approaches. First of all, on the one hand there is the creation of a model that has its roots in the solid physical knowledge of the phenomenon underlying the system. On the other hand the creation of a model that is independent of the individual parts of which it is composed and of the laws that govern the single elements, favoring an overall vision of the system as a unit capable to process inputs and provide outputs in response to them.
A lumped-parameter approach is preferable if the interest is in the cause-effect relationships that govern the elements of the system: it is easier than a black-box model to understand what happens on a physical level when a parameter changes and give a satisfactory explanation. The outputs are more easily interpretable since the relationships that bind them to the inputs are known and above all the background of the physical phenomenon underlying the process is known. As the complexity of the system grows, however, it becomes difficult to trace these relationships and the attempt leads to poor results given the multitude of parameters to be identified and the series of assumptions that have to be made for all those that cannot be identified. On the contrary, a data-driven approach is preferable given its ability to manage all the components of the system as a single entity and regardless of the characteristics of each, providing instead an overview. Unfortunately this happens to the detriment of the immediate understanding of the functioning of the system: for a black-box parameterization it is more difficult to interpret the input-output relationship given the

computational complexity introduced by the use of non-linear structures or wavelet-type functions, neural networks, etc. , in opposition to the case of lumped parameters in which the interpretation is usually of greater immediacy. Summarizing, the following would be the key points of discussion:

- *Ease of understanding how the system works*: generally understanding the physics of the phenomenon makes understanding the behavior of the system itself more immediate; in this, lumped-parameter systems are favored. However, when the system itself is complex, a data-driven approach is better and the model is more easily obtained.
- *Computational complexity*: lumped-parameter models are usually lighter than black-box models since physical relations are for the most part algebraic or differential relations. Black-box models usually are complex mathematical functions.
- *Quality of the model*: if correctly performed, both approaches lead to a quality result. The choice depends on the tools available, the purpose of the model, the development time required, etc.

In the case in question, the data-driven model has been preferred due to the possibility of using a PWM-angular velocity relationship without the need to characterize each element in the ideal line that connects the input to the output; a further advantage is to have obtained, in reality, not only the behavior of the DC motors, but in fact a model of the entire platform given the influence that the output of one track has on the behavior of the other.

5.7 Inertia Identification

For simulation a kinematic model has been adopted; nevertheless, in view of a future integration of the dynamic model, the identification of the system inertia J has been performed. The complex construction architecture of the real model and its innumerable components justified the adoption of an identification method that did not require the disassembly and analysis of the individual elements, but that allowed, through the measurement of the vehicle's oscillation period on a pendulum, to go back to the desired parameter. For this purpose, in analogy to what reported in [16], the composite pendulum method has been used.
In order to capture the robot CoM position, the Otus Tracker sensor has been employed: the reasons are related to the ease of installation on the robotic platform and the ease of acquiring the data detected thanks to the use of the RCbenchmark Tracking Lab software.
Otus tracker sensor uses two base stations appropriately positioned as tracking references.

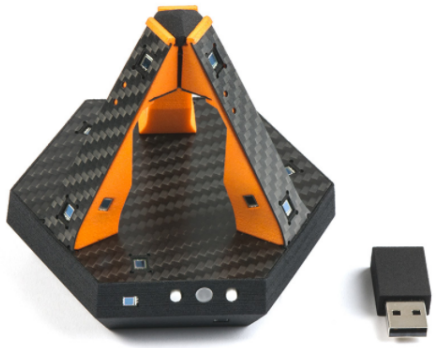


Figure 5.24: Otus Tracker sensor [73]

The base stations are passive emitters of infrared light and interacting with the sensor allow the identification in the space of the body. In detail, it is possible to obtain

- position of the CoM
- translational speeds
- angular velocities
- orientation of the body through a representation in quaternions, subsequently reconverted in terms of Euler angles ϕ, θ, ψ respectively around the x, y and z axes

captured at a specific frequency, 200 Hz in this case. The experiment has been conducted with the use of a pendulum to which the robotic platform has been connected by means of a rigid aluminum bar of known size and mass. Figure 5.26 shows the experimental setup: the support holds the bar-robot system which is free to oscillate with negligible friction around the upper hinge. The extent of the oscillation is captured by the Otus Tracker sensor whose local reference axes have been oriented parallel to the *body* axes of the robot system, i.e. with the x axis directed along the local vertical, the z axis parallel

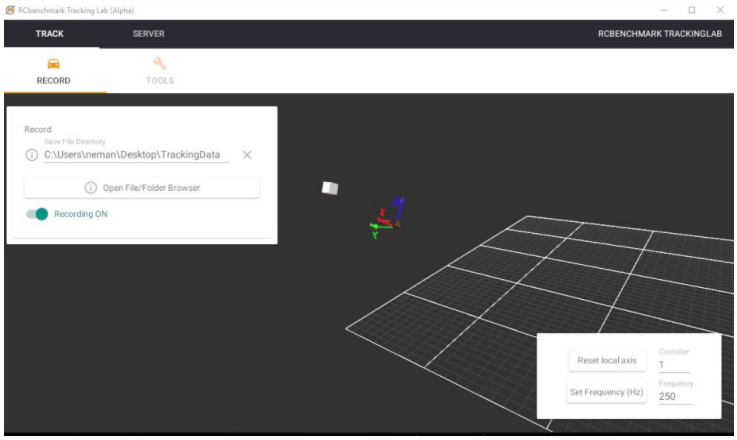


Figure 5.25: Software RCbenchmark Tacking Lab [72]

to the rotation axis and y axis to complete the triad. It is also assumed that the origin of the robot's local reference system is in the robot center of gravity. An angle close to

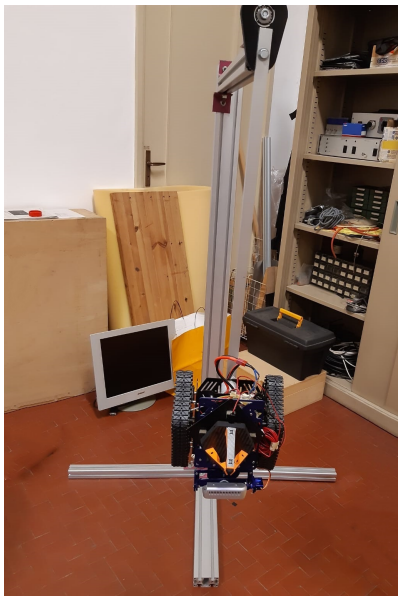
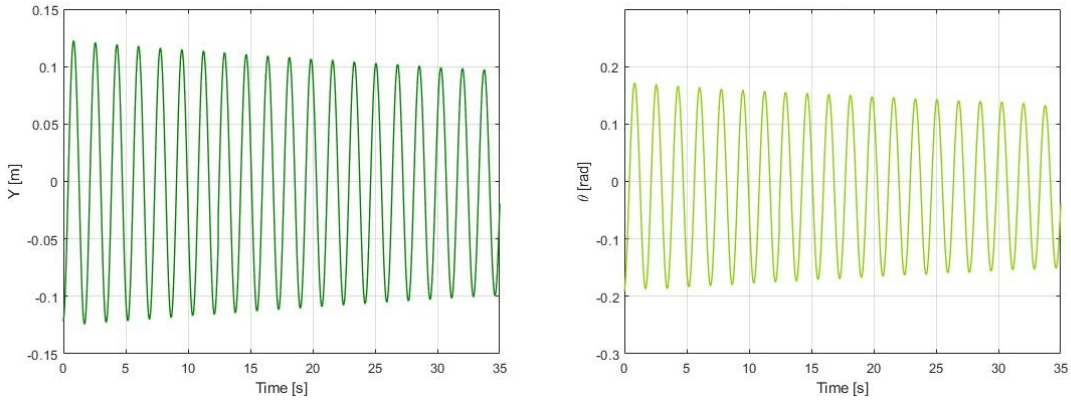


Figure 5.26: Experimental setup

10° has been provided and the system has left free to oscillate for a time of 35 seconds, while the recording of position and oscillation angle θ over time has been carried out. The analysis of the temporal variation of θ has made possible to trace the period of oscillation T: different tests have been conducted in order to obtain a more accurate result (errors linked to phenomena of friction, vibrations, uncertainties in the calculation of distances and weights have been ignored for simplicity of treatment). By the analysis of the temporal distance between the first and second positive peaks of the angular position, a period of oscillation of T=1.735 s has been found.



(a) *Y position* (b) *Yaw angle*

Figure 5.27: Data collected during the experiment

In Table 5.2 it is possible to find the data necessary to calculate the unknown I: these data are related to the geometry of the system and the mass of the elements. The inertia

Data			
robot mass	m	kg	1.592
bar mass	m_1	kg	0.146
center of gravity-hinge distance of robot CoG	l	m	0.7
center of gravity-hinge distance of bar CoG	l_1	m	0.455
Inertia of the bar	I_1	kg m^2	0.0078

Table 5.2: Geometry and inertial data

of the aluminum bar has been evaluated taking into account its mass and its geometry; homogeneity of the body has also assumed, so the center of gravity has assumed to coincide with the midpoint of the bar.
Reporting the expression obtained in subsection 2.6.1

$$I = \left(\frac{T}{2\pi}\right)^2 (mgl + m_1 g l_1) - ml^2 - m_1 l_1^2 - I_1 \tag{5.7}$$

an inertia value of $J = 0.0652$ kgm^2 has been calculated.

Chapter 6

Simulation Model

A model plays the role of a surrogate for the system it represents and its purpose is to replace the system in experimental studies [10]

MATLAB/Simulink is the modeling, simulation and analysis environment of dynamic systems chosen for the creation of the Simulation Model of the robot. Its functions are countless, including solution of algebraic equations, management of the state variables, numerical integration with several methods, "block" diagram representation of equations. The advantage is that of clearly distinguishing the different components of the system (Plant, Control, Guidance etc.) and exploiting a simple graphical interface with the potential of the MATLAB environment behind it. This makes it one of the most versatile and useful tools in the widest variety of application fields, from the aerospace sector to the electronic, mechanical, automotive, etc.
In the previous chapters the mathematical model has been introduced and the essential elements for its implementation in a simulation environment have been obtained: now it is presented the model realized in the Simulink environment together with the implementation of the selected algorithms as the robot's guidance, control and navigation strategy.

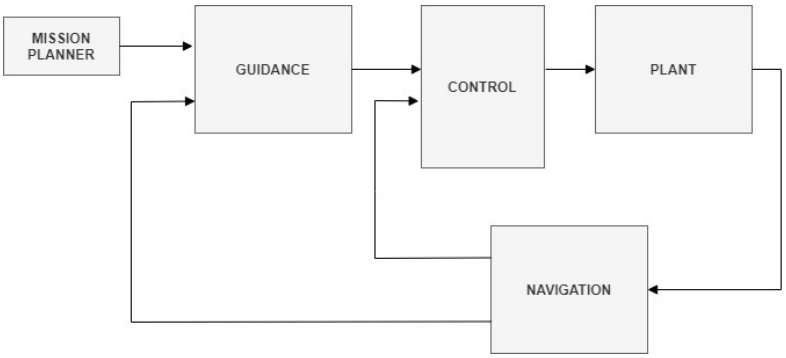


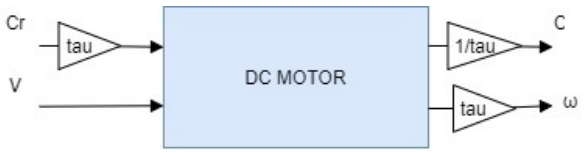
Figure 6.1: Complete Model

6.1 Plant Model

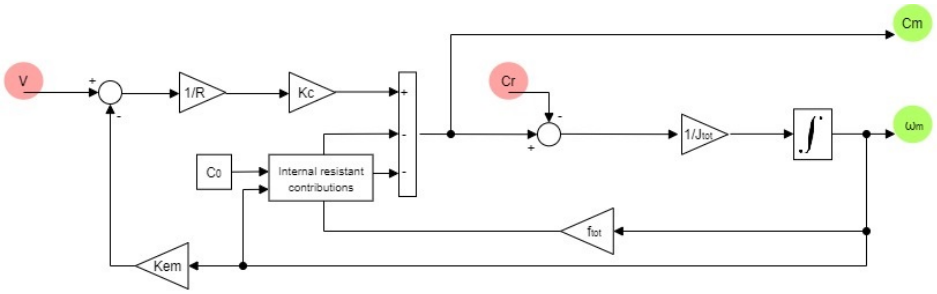
The identification of the robot model has been conducted through a lumped-parameter approach and through a data-driven one.

6.1.1 Lumped Parameter Approach

Based on the model in (2.61), the blocks diagram in Figure 6.2 has been created. Model inputs are the supply voltage V and the external resistant torque C_r, model outputs the useful torque C_m and the angular speed of the driving wheel ω. A representation of this type offers the possibility of implementing both a kinematic and dynamic model thanks to the calculation of torque and speed; however, it has the disadvantage of requiring a relationship that can convert the PWM signal into a voltage signal. Figure 6.3 shows an



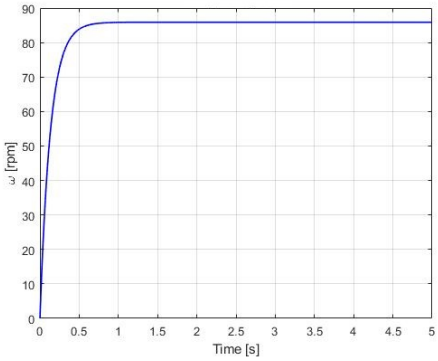
(a) *Block scheme*



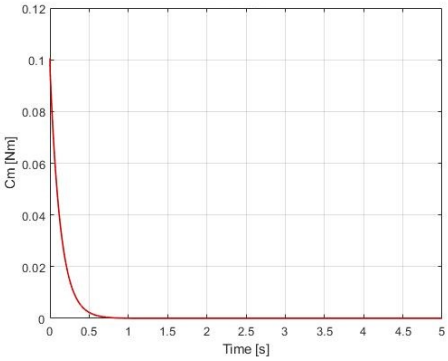
(b) *Block scheme detail*

Figure 6.2: Motor Scheme

example of the behavior of the model at a voltage step of 3 V without applying any external load. In Figure 6.4 an external load of Cr= 0.017 Nm is applied at the external shaft: it is plotted the response of the system in angular speed terms at different voltage inputs.



(a) *Angular speed*



(b) *Torque*

Figure 6.3: System response to a step of 3 V, no external load

Voltage	
3	blue
3.5	red
4	green
4.5	magenta
5	cyan

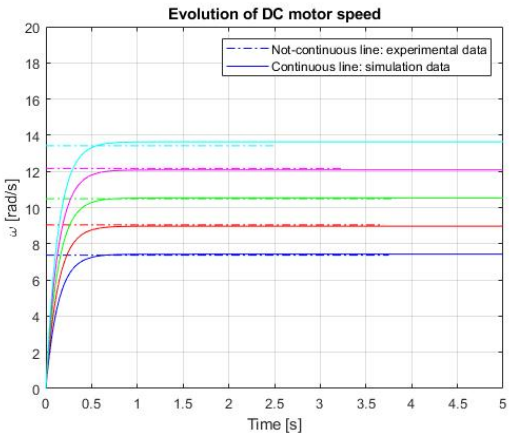


Figure 6.4: System response to a step voltage input, C_r=0.017 Nm

The graph in Figure 6.5 is a comparison between the speeds reached by the drive wheel in steady state and those predicted by the model for different voltage values and external loads applied. It can be seen that the agreement of the results is generally good, with misalignments only in the case of a higher supply voltage. The graph in Figure 6.6 shows the current and drive torque values at steady state based on the external load applied: note that although the useful torque is zero in the absence of an external load, the current is not, given the need to compensate for the internal resistance of the motor.

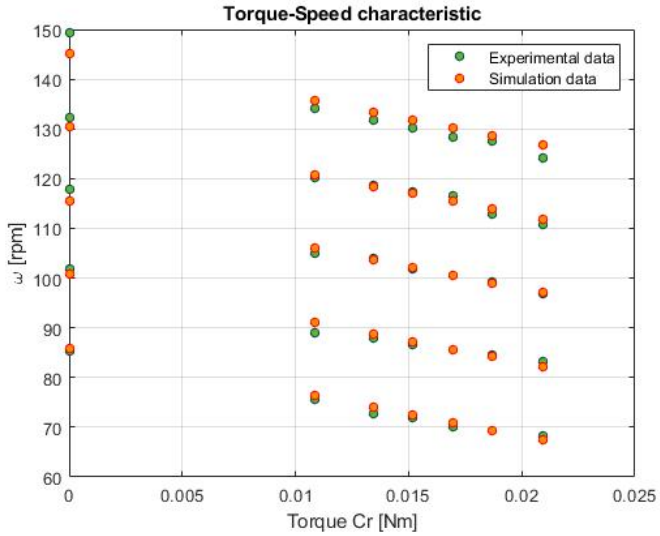


Figure 6.5: Experimental and Simulation data comparison

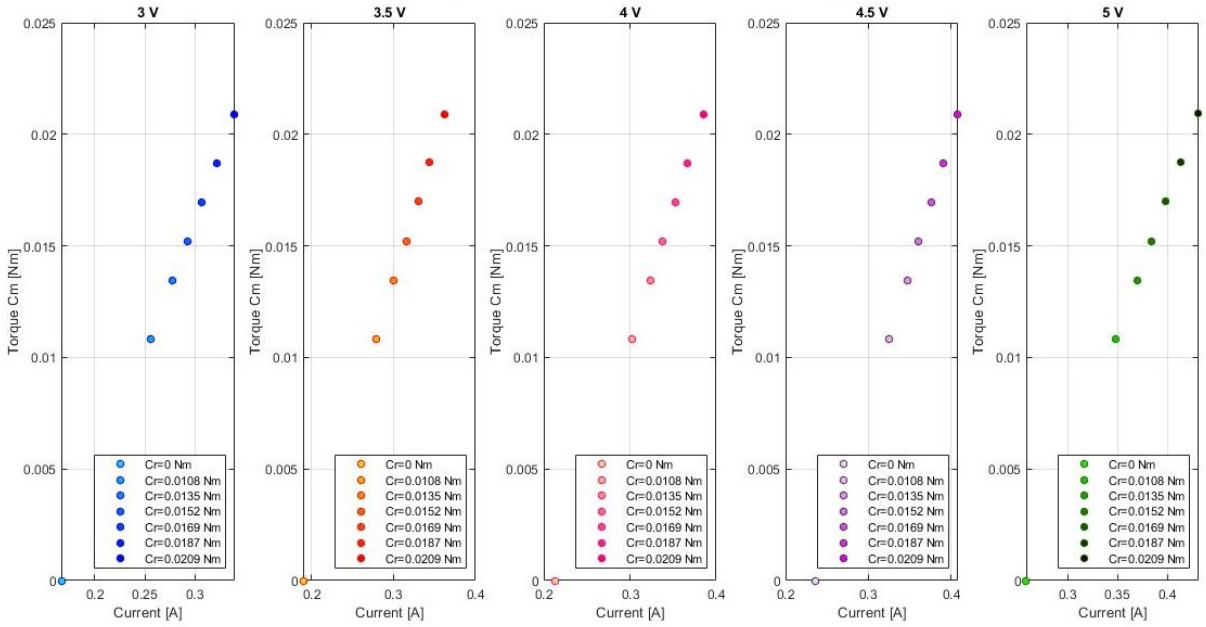


Figure 6.6: Torque-Current characteristic for different Voltages and Resistant Loads

6.1.2 Data-driven approach

The model adopted for the simulation, however, is the one obtained through a data-driven approach: the choice is due to the advantage of a direct PWM-ω relationship and the possibility of immediate use without the need to identify additional parameters such as, for example, the external resistance that is exerted on the motor as the robot advances. The NARX model has been imported in MATLAB/Simulink thanks to the appropriate non-linear block available in System Identification Toolbox Library. The Plant model is shown in Figure 6.7:

- a first block represents the behavior of the motors
  - inputs are PWM command to left and right motors
  - outputs are the left and right angular speeds
- a second block is the mathematical model of the plant (2.13)
  - inputs are left and right angular speeds
  - outputs are the body velocity vector V_x and the angular yaw speed $\dot{\psi}$

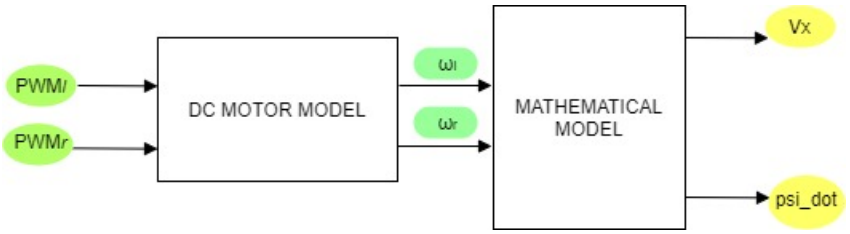


Figure 6.7: Plant Model

As an example of the plant behavior, a simulation is performed based on the PWM step input in Figure 6.8: results are shown in Figure 6.9.

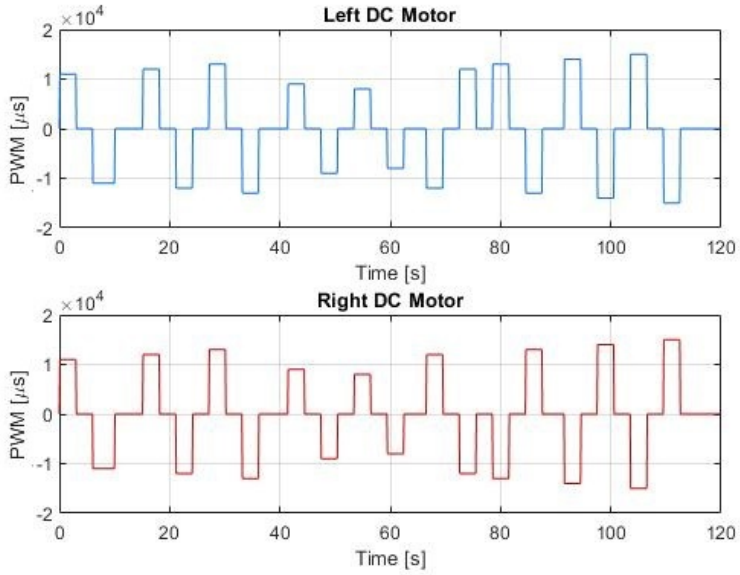


Figure 6.8: Plant Model Inputs

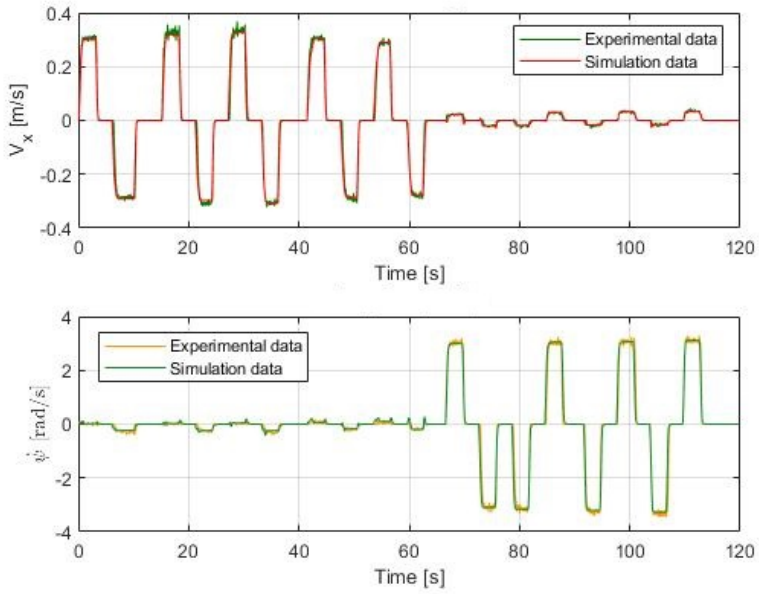


Figure 6.9: Plant Model Outputs

6.2 GNC Model

Alongside the Plant, a model of the Guidance, Navigation and Control Systems has been implemented:

- The *Control block* takes care of converting the references generated by the Guidance into commands to the system actuators: the extent of these commands depends on the information generated by the Navigation block, in particular on the comparison between the desired state and the one currently possessed by the robot.
- The *Navigation block* receives the information coming from the plant model and processes them in order to provide the status of the system.
- The *Guidance block* is responsible for generating the references in order to provide the robot with a direction to follow and the speed profile to adapt to.

The robot is considered as a point that moves in space. Obstacles are reported as points surrounded by a circular area of influence, the navigation within which is considered dangerous for the robot. Usually this area is larger than the size of the obstacle itself to ensure more safety: a value of 70 cm has been chosen calibrated on the robot's performance. The goal is also a point. The color code is reported:

- Green point: initial point, always in the origin of the Inertial Frame
- Blue point: actual position of the robot
- Red point: goal position
- Orange points: obstacles position
- Dashed orange circle: obstacles area of influence
- Blue continuous line: robot's path

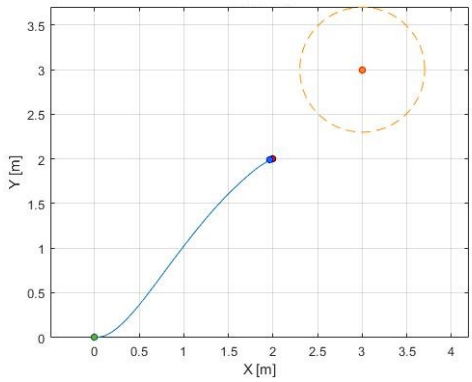


Figure 6.10: Example of a mission trajectory

6.2.1 Control

It deals with the generation of a command for the actuators, i.e. the electric motors, based on the difference between the desired and the states currently owned by the robot. It consists of two main parts:

- a first block represents the *Controller*
  - inputs:
    - Reference from Guidance: V_{xref}, reference velocity along the x-axis, and ψ_{ref}, reference orientation.
    - Navigation information: estimated V_x, velocity along the x-axis actually owned by the robot, and ψ, estimated orientation.
    - Eos signal: signal that decrees the achievement of the objective and the consequent end of the simulation.
  - outputs:
    - Command signal generated by the velocities comparison V_{xcmd}.
    - Command signal generated by the orientations comparison ψ_{cmd}.
- a second block is the *Command Module*, responsible for translating the command signal into a PWM signal for the motors
  - inputs are the Command signals
  - outputs are motors' PWM

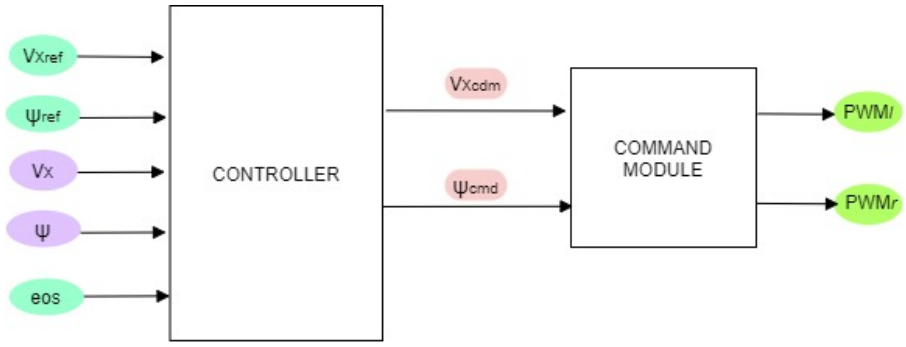


Figure 6.11: Control Block

The Controller block is made up of two PID controllers:

- A PI controller for the speed, responsible for the generation of V_{xcmd}.
  The error signal in input to the PI is the difference between the desired and the actual velocity of the robot
  $$e_V(t) = V_{xref} - V_x \tag{6.1}$$
  while its output the sum of a proportional part made up of the input signal multiplies by a gain K_{pVx} and a part equal to its integral multiply by the gain K_{iVx}. A saturation has been imposed to prevent excessive command.

- A P controller for the orientation, responsible for the generation of ψ_{cmd}.
  The error signal in input to the PID made up of only the proportional part ($K_{p\psi}$ is the value of the gain) is the difference between the desired and the actual orientation of the robot
  $$e_\psi(t) = \psi_{ref} - \psi \tag{6.2}$$
  The generated ψ_{cmd} is again subjected to saturation.

The Command Module block is characterized by the transformation of the command signals in PWM signals:

$$\begin{aligned} PWM_l &= (V_{xcmd} - \psi_{cmd}) \cdot 20000 \\ PWM_r &= (V_{xcmd} + \psi_{cmd}) \cdot 20000 \end{aligned} \tag{6.3}$$

A manual tuning of the PID parameters has been carried out in order to obtain the response with the desired characteristics for the system: in particular, the priority is to ensure *maneuver stability*, avoiding excessive commands and consequently the missing of the goal or the lengthening of the time necessary to achieve it. PID tuning in nonlinear system is not simple; the search for the most optimized solution is not the aim of this treatment, so here are presented gains values that have been demonstrated to be acceptable in simulations.
Figure 6.12 shows the response of the system to the variation of the gain value K_{pVx} given as input to the system a reference speed of 0.25 m/s. With the proportional term alone, it is necessary to considerably increase the entity of the proportional part of the PID to get closer to the reference; with the addition of the integrative term, for example a value of K_{iVx} of 0.5 like the one adopted in Figure 6.13, the situation improves allowing the achievement of the objective with a lower K_{pVx}. In Figure 6.14, setting the proportional gain at 2, it is possible to observe how the response of the system changes as the integrative term increases. As can be seen, the effect is twofold: first, the cancellation of the error at steady state; secondly, the increase of K_{iVx} brings the speed up of the system thus reaching the desired speed in a shorter time. On the basis of these considerations, the gain values

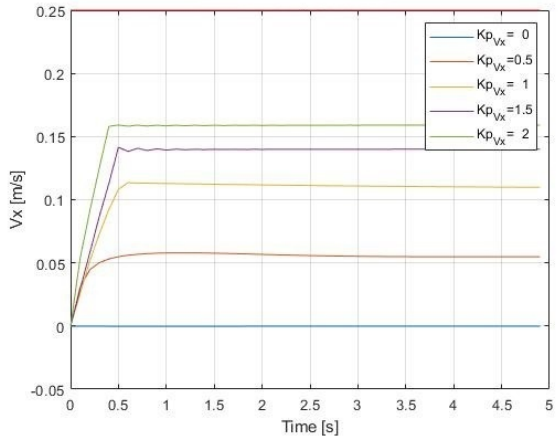


Figure 6.12: K_{pVx} Tuning

have been set as 2 for K_{pVx} and 1.85 for K_{iVx}; the derivative term has instead been set to zero.

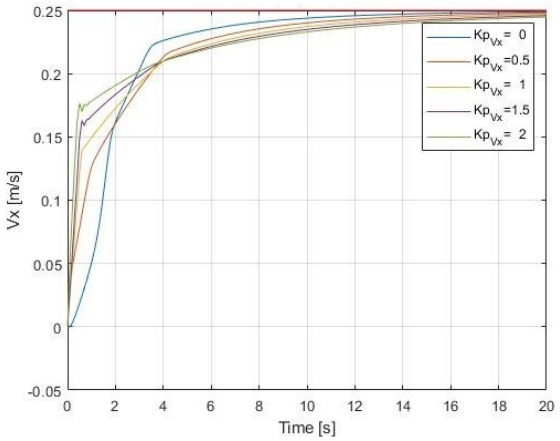


Figure 6.13: Proportional and Integrative Part Tuning

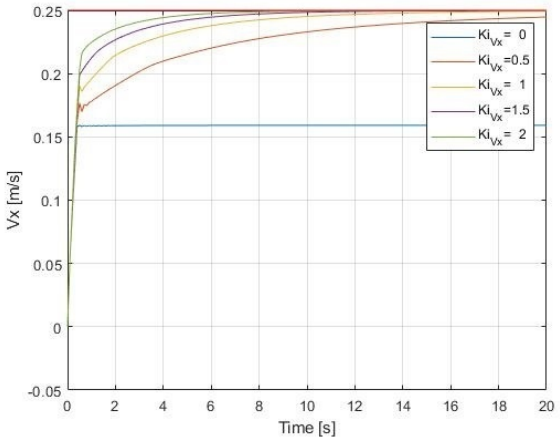


Figure 6.14: K_{iVx} Tuning

Figure 6.15 shows a series of curves representing the responses of the system to a step reference (in dashed black) obtained for different values of the proportional gain of the PID controller. If only the first part of the graph is observed, the response of the system speeds up as the gain $K_{p\psi}$ increases and it would therefore seem advantageous to adopt a high gain; on the other hand, if the reference inversion zone is observed, for high $K_{p\psi}$ an unwanted overshoot appears and this is not favorable considering that the risk would be to collide with obstacles in the proximity.

Let us observe what happens using a lower and a higher $K_{p\psi}$ value in the case of a simple

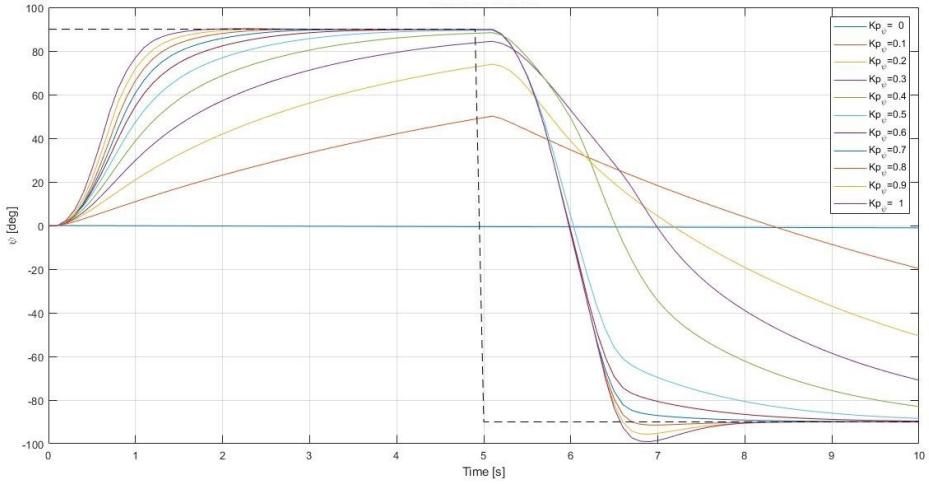
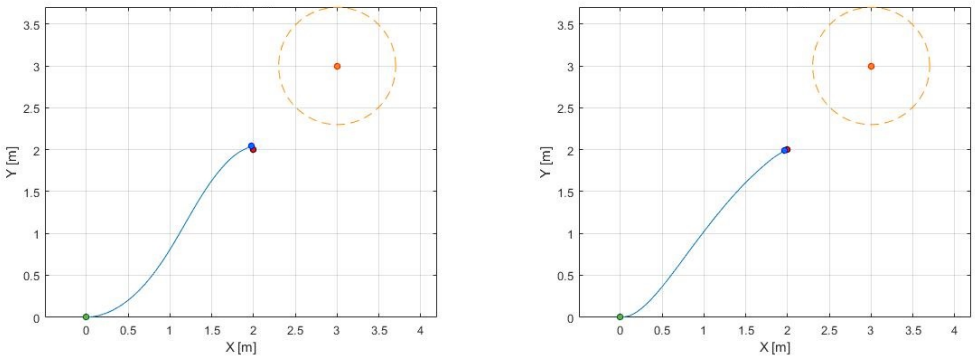


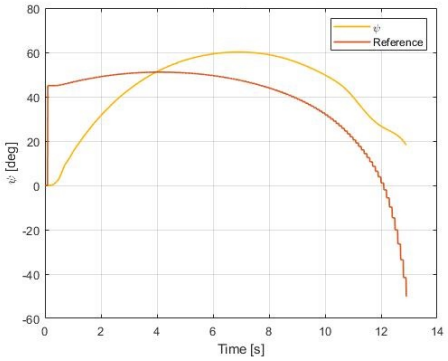
Figure 6.15: $K_{p\psi}$ Tuning

mission without obstacle and one with an obstacle along the robot trajectory. Figure 6.16 reports the two path obtained in the former case with a value of 0.3 and a value of 0.6 for $K_{p\psi}$, Figure 6.18 the two path in the latter case again with values of 0.3 and 0.6 for $K_{p\psi}$.

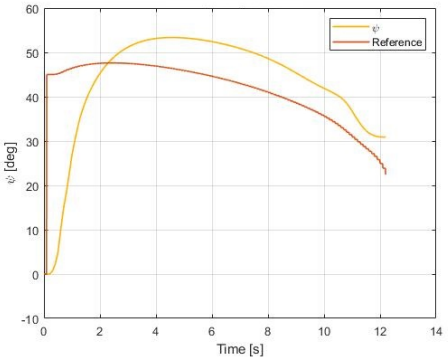


(a) $K_{p\psi}$=0.3 (b) $K_{p\psi}$=0.6

Figure 6.16: Robot path, no obstacle

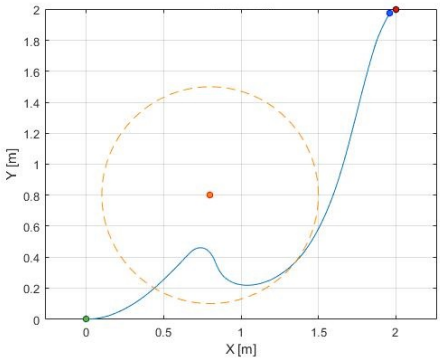


(a) $K_{p\psi}$=0.3

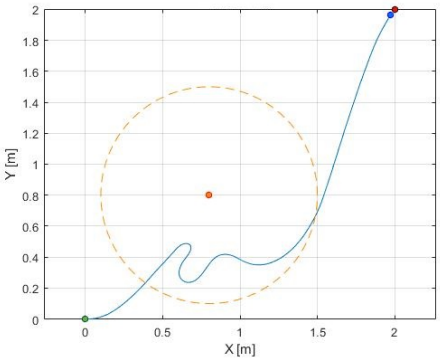


(b) $K_{p\psi}$=0.6

Figure 6.17: Robot angular position, no obstacle



(a) $K_{p\psi}$=0.3



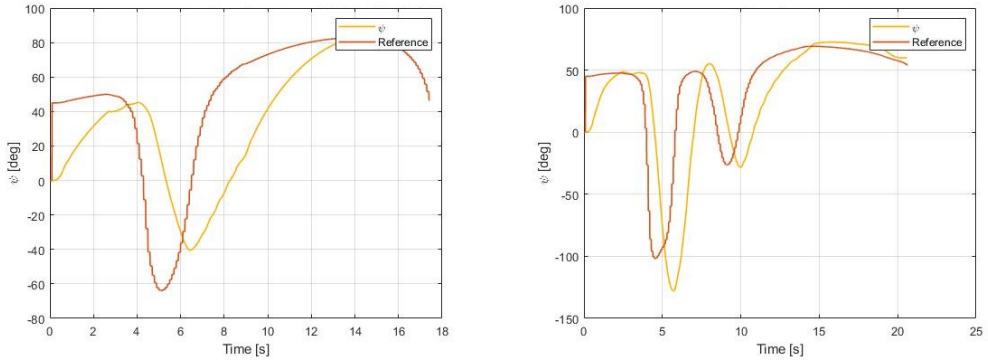
(b) $K_{p\psi}$=0.6

Figure 6.18: Robot path with an obstacle

The variation of the orientation is instead shown in Figure 6.17 and Figure 6.19 both for the first and for the second case. Here are some considerations:

- *Case without obstacles*: the most evident difference is in the time it takes for the system to approach the reference. It is observed that while with a $K_{p\psi}$ equal to 0.6 in the first two seconds there is the entrance in a band of about ± 10 degrees from the reference, for 0.3 this happens beyond and maintaining a generally greater error. The trajectory is less optimized in terms of distance from the shortest path for a value of 0.3 of $K_{p\psi}$, while the result is better in the 0.6 case.

- *Case with an obstacle*: for a value of 0.6 of $K_{p\psi}$ the excess of overshoot leads the system to perform more than one turn to assure the robot to reach the goal, so more time to complete the mission. A value of 0.3 ensures the achievement of the reference and the completion of the mission in a shorter time even if the reference is followed less faithfully.



(a) $K_{p\psi}$=0.3 (b) $K_{p\psi}$=0.6

Figure 6.19: Robot angular position with an obstacle

Based on these considerations, an intermediate value of 0.4 for $K_{p\psi}$ has been considered. The possibility of using the other gains has been also evaluated, but the poor results have led to the conclusion that the proportional part alone is sufficient to ensure the desired performance.

6.2.2 Navigation

It deals with the determination of robot actual state and the closing of the system loop through its feedback. It takes as inputs the speed in the body system V_x and the angular speed $\dot{\psi}$ and translate them into quantities referring to the Inertial system. Through integration, it deals with tracing the position of the body and its orientation, information which it then feeds back so to be exploited by the Guidance and Control Systems. By considering the future integration of a dynamic model, a Kalman filter can be implemented for merging data from the two models (kinematic and dynamic) in order to provide greater accuracy in determining the state of the robot.

6.2.3 Guidance

The objective of the robot's mission is to reach a certain position avoiding collision with any obstacles present in the environment in which it operates. The creation of the path from the initial to the final position is entrusted to the Guidance block: thanks to the information on the current state of the robot, the guidance algorithm is able to generate a reference in terms of orientation and speed to satisfy the achievement of the goal. This block is made up of

- *Initial Orientation Block*
  - Inputs are the goal position and the estimated orientation
  - Outputs are the reference for the orientation and a signal to enable the APF guidance algorithm once certain criteria have been satisfied
- *APF Block*
  - Inputs are the goal position, obstacles position and the estimated position
  - Outputs are the reference for orientation, velocity and a signal that establishes the arrival to the goal

6.2.4 Initial Orientation

The initial correction of the orientation allows to shorten the time necessary to reach the goal if it is located in an unfavorable point with respect to the initial orientation of the robot (behind, for example). The idea behind the algorithm is that the robot is always in the initial position at the origin of the Inertial Reference System and that its initial orientation is entered by the user (estimated by on-board sensors in the real case). The space is divided into 3 areas:

- a front area, characterized by orientations between $-\pi/2$ and $\pi/2$
- a first back area, characterized by orientations between $\pi/2$ and π
- a second back area, characterized by orientations between $-\pi/2$ and $-\pi$

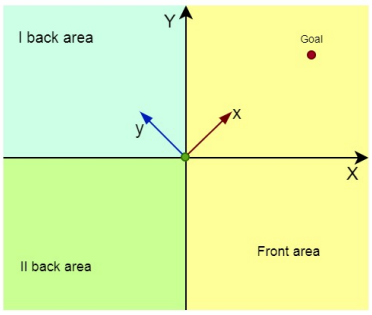
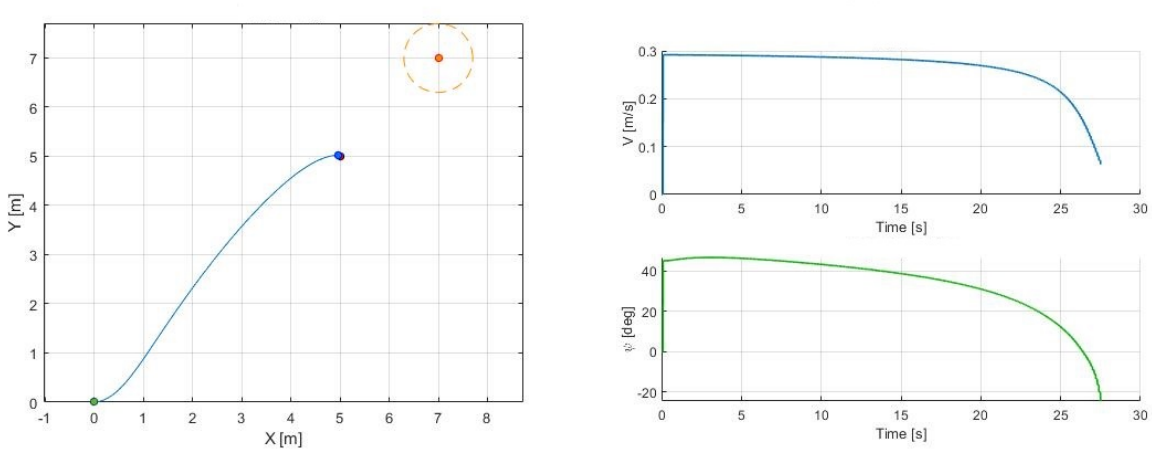


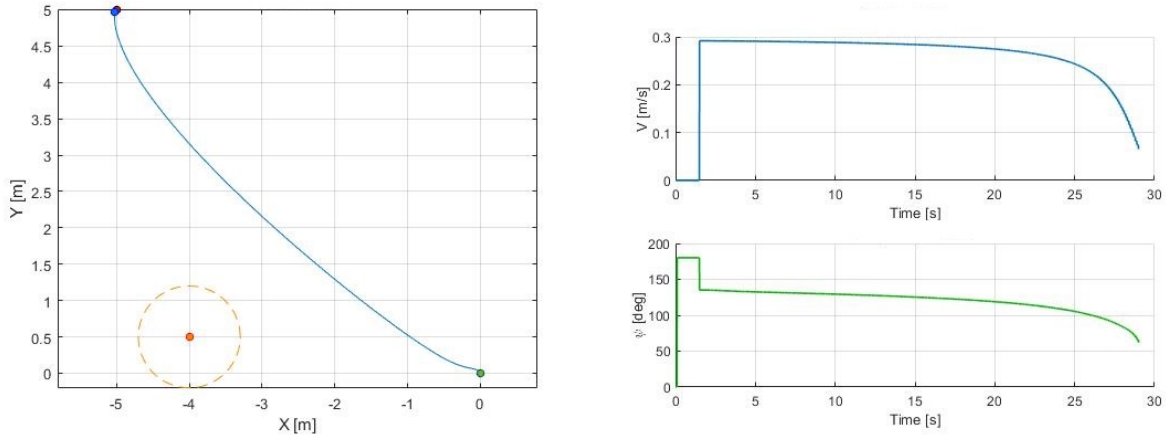
Figure 6.20: Space of robot mission

Based on the goal position, the robot receives as a reference an orientation to follow: none if it is already oriented in the most favorable area to achieving the goal, $\pm\pi$ if this is not verified. Once returned to the correct area, the enable signal activates the actual guidance algorithm and disables the functioning of the block.
Below there are some examples by which to observe what is the behavior of the Guidance block based on the different positioning of the goal: Figure 6.22 for the first back area, Figure 6.21 and Figure 6.24 for the front one, Figure 6.23 for the second back area.



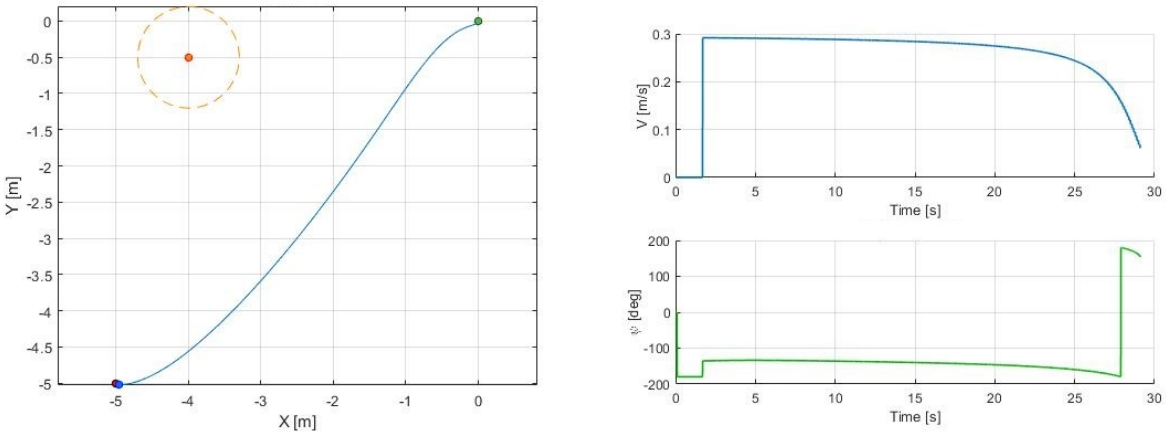
(a) *Trajectory* (b) *References*

Figure 6.21: Robot path, Goal in the front up area



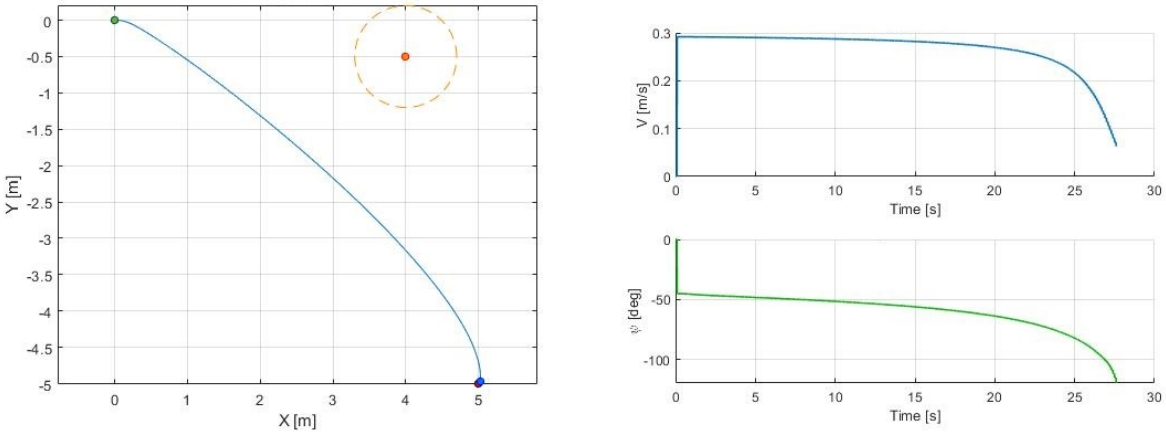
(a) *Trajectory* (b) *References*

Figure 6.22: Robot path, Goal in the first back area



(a) *Trajectory* (b) *References*

Figure 6.23: Robot path, Goal in the second back area



(a) *Trajectory* (b) *References*

Figure 6.24: Robot path, Goal in the front down area

6.2.5 Reference signals

Once the optimal state for achieving the goal in terms of initial orientation is reached, the guidance algorithm is activated. It is based on an Artificial Potential Field method already introduced in chapter 4: it receives information on the current position and on the goal and obstacles positions and traces the path to follow by exploiting the intensity of the potential field generated by the elements present in environment. In particular, for the generation of the orientation signal reference, the relationship is

$$\psi_d = \ atan\frac{F_Y}{F_X} \tag{6.4}$$

where F_Y and F_X are the sum of repulsive and attractive force along the X and Y axes. The values of the attraction and repulsion coefficients K and η have been identified through a series of tests, favoring those for which navigation proceeded smoothly without sudden changes in orientation. Below there are some graphs that show the studies conducted in order to determine the most appropriate values for the case in question.
Suppose that the aim is to determine what is the force of attraction that the goal exerts on the robot. For simplicity, let us assume that the robot moves exclusively along the X-axis and that the goal is located 4 m in this direction from the starting point of navigation, that is 0. The force that the robot feels is reported at each point as the K coefficient changes (Figure 6.25). First of all, to avoid that the attraction is excessive when the robot is very far from the goal, a function of the type described in section 4.3 is adopted, indicating a distance of 2 meters as the one in which the potential becomes parabolic again. The choice of the value K depends on the needs of the project: one could, for example, establish a value of K equal to 0.15 and impose that the speed follows this reference; in this case over a K equal to 0.2 it makes no sense to go as the maximum speed of the robot is around 43 cm/s. A value of K=0.15 has been chosen. Suppose now that the aim is to determine

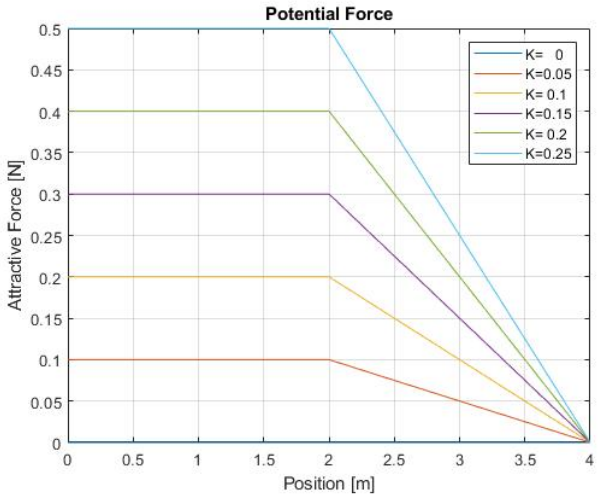


Figure 6.25: Attraction Force

what is the force of repulsion that the obstacles exerts on the robot. For simplicity, let us

assume that the robot moves exclusively along the X-axis and that the obstacle is located 1 m in this direction from the starting point of navigation, that is again 0. The force that the robot feels is reported at each point as the η coefficient changes (Figure 6.26). It is

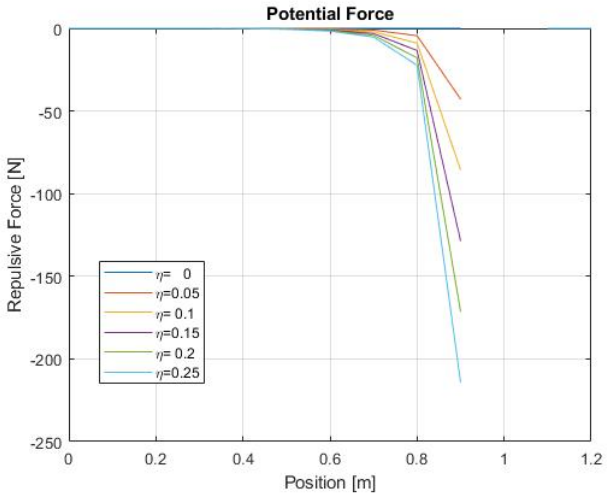


Figure 6.26: Repulsive Force

possible to notice how as the robot approaches the area of influence of the obstacle, setting at 70 cm, the felt repulsion force increases hyperbolically. In light of this, an eta coefficient of 0.05 has been considered sufficient.
For the velocity reference, a different approach has been employed: a relation like the one in section 4.4 has been adopted and an in-depth study has been conducted in order to identify the most suitable values for the parameters involved.
It is here reported the expression of V_d:

$$V_d = \frac{V_{xmax}\ \theta^h}{\theta^h + \Delta^h(\boldsymbol{q})} \tag{6.5}$$

with

$$\Delta(\boldsymbol{q}) = \frac{K_g}{\rho_g} + \sum_{i-1}^{N_o} \frac{K_i}{\rho_{obsi}} \tag{6.6}$$

where $\boldsymbol{q}$ is the current position of the robot, ρ_g is the distance from the goal point, ρ_{obs_i} the one from the i-esimal obstacle with N_o number of obstacles, K_g and K_i positive parameters, h positive integer number, θ positive parameter and V_{xmax} the maximum desired reachable velocity. The term $\sum_{i-1}^{N_o} \frac{K_i}{\rho_{obsi}}$ disappears if the robot is not in the area where there is the obstacle influence.
Some considerations are reported:

- V_{xmax} is set at 0.3 m/s to ensure acceptable mission times and prevent a not too high destabilising speed.

- K_g, K_i, h and θ have been chosen so as to have a speed profile not too strongly decreasing in the vicinity of obstacles and goal.

Suppose a goal 5 meters in a straight line from the robot. For simplicity, consider the path free of obstacles. All the coefficients are fixed except for the θ parameter (Figure 6.27, with V_{xmax}=0.3 m/s and a value of 1 for K_g and h). It can be noticed that as the parameter

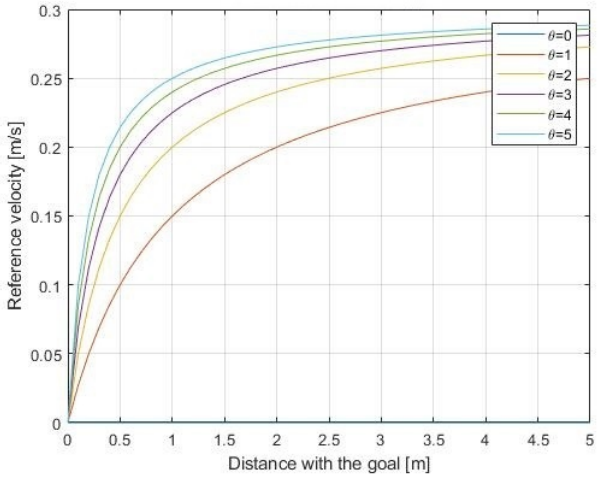
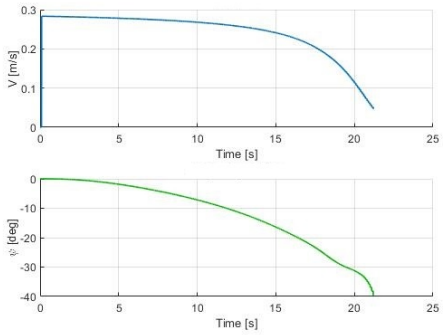
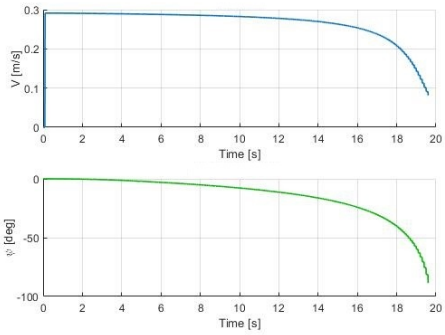


Figure 6.27: Variation of the Velocity Reference for $\theta \in [0,5]$

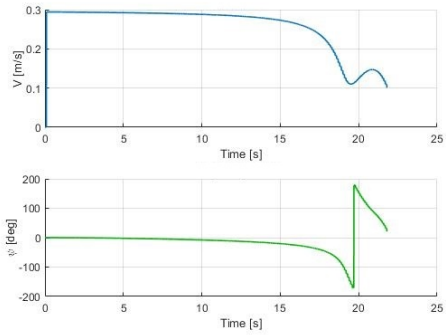
increases, the steepness of the curve increases near 0, i.e. the deceleration required to stop at the goal increases. A trade off is necessary: if the aim is to shorten the travel time of the stretch that separates the robot from the goal, then a higher value must be chosen. However, it must be taken into account that the probability of goal overcoming increases. When the aim is the reaching of a physical object in order to picked it up with a robotic arm, then the risk of collision increases if the system is not promptly reactive in stopping. Consider three references for comparison (Figure 6.28): they have been obtained by simulating a mission with the objective of reaching a goal at 5 m from the robot in the X direction, with initial orientation equal to 0. Values of 0.3 and 1 for K_g and h and V_{xmax}=0.3 m/s have been set. Figure 6.28a has been obtained for a value of 1 for θ, Figure 6.28b for a value of 2 and Figure 6.28c for a value of 3. From the first mission to the second the time necessary to reach the goal has been shortened (21.5 seconds in the first case, while 19.6 in the other), but by increasing up the θ value problems in reaching the goal arise: the latter is in fact surpassed in the third case and a turning maneuver has been required to complete the mission. A value of 2 has been chosen as a compromise.



(a) θ=1



(b) θ=2



(c) θ=3

Figure 6.28: Variation of the References for $\theta \in [1,3]$

Let us repeat the same mission, this time varying the value of h (Figure 6.29). The deceleration effect near the goal is even more pronounced. A value of 1 has chosen such as to ensure an adequate decline of the curve. The last two parameters investigated have been the coefficients K_g and K_i. For the sake of simplicity, only the curve relating to a path without obstacles is shown again (Figure 6.30). As the coefficient increases, the slope of the velocity curve near the origin softens. A value of 0.4 for K_g has been assumed so that at a distance of 20 cm the speed drops to around 50% of the maximum established. A value of 1 for K_i has been instead assumed in order to have a slowdown near the obstacle without causing the robot to stop.
Finally, a tolerance value has chosen in achieving the goal: the mission ends when the robot reaches an area around the goal of 10 cm in radial width.

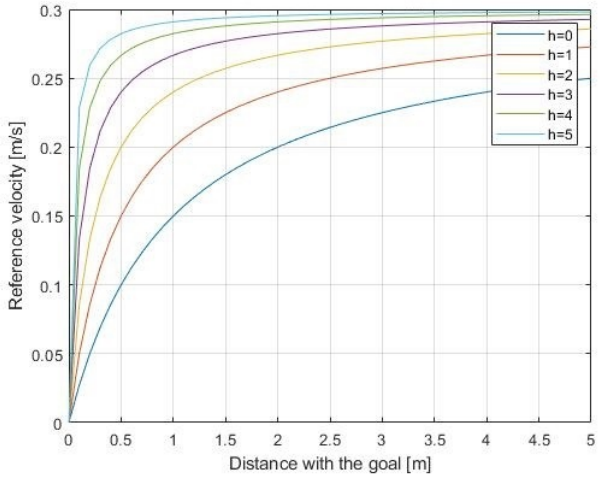


Figure 6.29: Variation of the Velocity Reference for $h \in [0,5]$

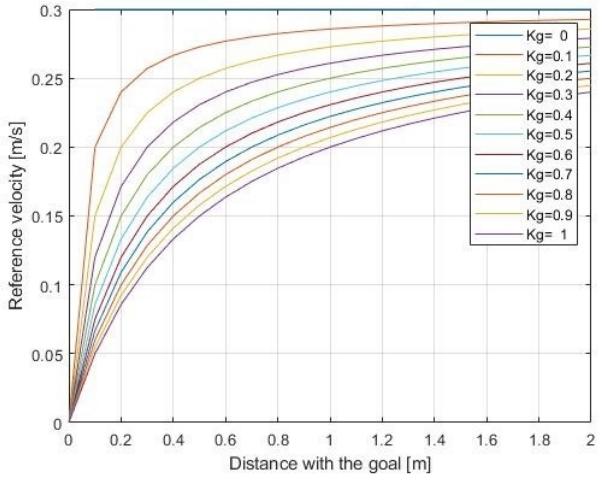


Figure 6.30: Variation of the Velocity Reference for $K_g \in [0,1]$

6.3 Mission Planner

A *Mission Planning* block has been included in order to provide the interface through which to establish the mission goal: in order to implement the guidance algorithms it is necessary to enter the coordinates of the final point to be reached and the position of the obstacles that must instead be avoided.

Chapter 7

ROS/Gazebo and Code Generation

From the robot's perspective, problems that seem trivial to humans often vary wildly between instances of tasks and environments [...]. As a result, ROS was built from the ground up to encourage collaborative robotics software development. [84]

Robotic simulation is a process of emulation of the real world behavior of a robot in a virtual environment. The aim is to test its design and programming code without the necessity to use a real expensive prototype [44], with the possibility to modify its configuration with no additional cost, change its characteristics to meet the specifications, perform tests in different scenarios.
Gazebo is a a multi-robot simulator environment which offers such possibilities in complex indoor and outdoor applications [30]. Its strength is its compatibility with the *Robot Operating System*, one of the most popular frameworks for robotic software.

7.1 ROS background

ROS official definition is [85]

An open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

More in detail, it is a Meta-Operating System [85], i.e. a system that performs scheduling, loading, error handling and monitoring running on the existing operating system and offering robot application programs as libraries.
At the base of its success are the possibilities of

- *Distributed computation*: since different processes can run across several different computers, ROS can provide communication between them allowing an orderly exchange

of information.

- *Software reuse*: if a basic algorithm has already been implemented for a certain need, why not reused it when it reappears? The idea of ROS is to provide basic functions for navigation, motion planning, mapping, etc. via *packages* to leave space for experimentation with new ideas [70].

- *Rapid testing*: experimenting with high-level algorithms is easier since it is not necessary to have the real prototype available.

- *Writing easily*: Python and C++ can be used as program languages.

ROS is composed by three levels of concepts: File-system level, Computation Graph level, Community level.
Components of *File-system level* are [85]

- *Packages*: collection of run time processes, data-sets, configuration files and anything else organized in an orderly way.

- *Metapackages*: set of packages with common purposes.

- *Repositories*: collection of packages with a common VCS[1] system.

- *Package Manifests*: documents providing metadata about a package, i.e. name, version, description.

- *Message (msg)*: data exchanged between ROS elements, they can be variables such as integer, floating point, and Boolean [9].

- *Service (srv)*: a synchronous bidirectional communication between the service client and the service server that is responsible for responding to requests. In particular

  – *Service client*: it is a client that requests service to the server and receives a response as an input in form of message.

  – *Service server*: it is a server that receives a request as an input and transmits a response as an output in the form of message.

  Services are one-time message communication: when the request and response of the service is completed, the connection between two nodes is disconnected [9].

Components of *Computation Graph level* are [85]

- *Master*: it is responsible for node-to-node connections and messages communication. The command roscore is used to run the master: without it, nodes would not be able to find each other and exchange messages. A URI address is also configured (ROS_MASTER_URI) to let slave nodes to access and register their own information or request information from other nodes.

[1]Version Control System, class of systems responsible for managing changes to computer programs, documents, etc. [110]

- *Nodes*: they are the smallest unit in ROS [9]. They are processes that perform computation. A variable called ROS_HOSTNAME, which is stored on the computer where the node is running, is created and used as the URI address to communicate with Master.

- *Parameter Server*: store of the data required by the execution of nodes.

- *Topic*: a name used to identify the content of the message. Like in a conversation, there are nodes that publish on a given topic and nodes that subscribe to it.

  – *Publisher*: it is a node that registers its own information with the master and sends messages to subscriber nodes interested in the same topic.

  – *Subscriber*: it is a node that registers its own information with the master and receives publisher information about a specific topic.

  Topic communication is an asynchronous communication: there is a continuous transmission and reception of data between the connected nodes.

- *Bag*: format for saving and playing back ROS messages data.

- *Action*: message communication method used for an asynchronous bidirectional communication. The structure of action file is also similar to that of service and it is used where it takes longer time to respond after receiving a request and intermediate responses are required until the result is returned [9].

Community level is linked to the possibility of relying on a large community active in the exchange of software, knowledge and information benefiting users in the creation of their projects.
The ROS Master acts a nameservice in ROS Computation Graph that stores topics and services registration information for nodes. Nodes communicate with the Master to give and obtain information.

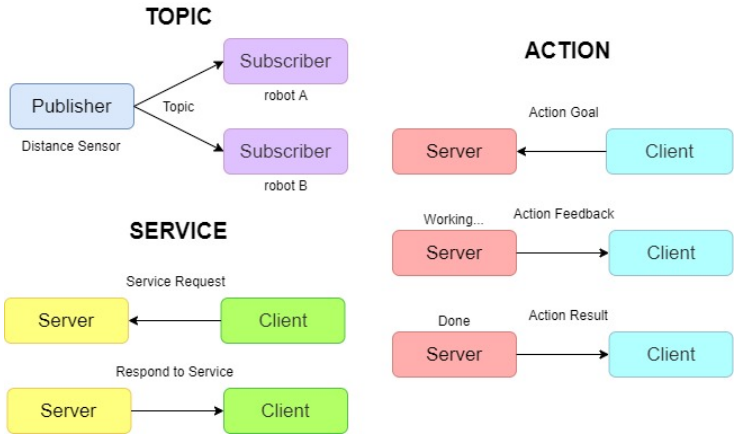


Figure 7.1: ROS communication strategies

7.2 Gazebo

Gazebo is an open source robot simulator built in a robust physics engine that offers a very practical programmatic and graphical interface. The interesting features of Gazebo for the purposes of this thesis are:

- *Dynamic simulation*, obtained through the use of physics engine such as Open Dynamics Engine (ODE), Bullet, Simbody, DART.

- *3D Graphics*, with high quality rendering of light, shadow and texture using Open-source Graphics Rendering Engines (OGRE).

- *Sensors and Noise Simulation*, given that it is possible to exploit a wide range of sensors importing their features as *plug-ins*.

- *Robot Model*, with the possibility of using a preset or a customized one. An already available virtual model of the Devastator robot has been used (Figure 7.2).

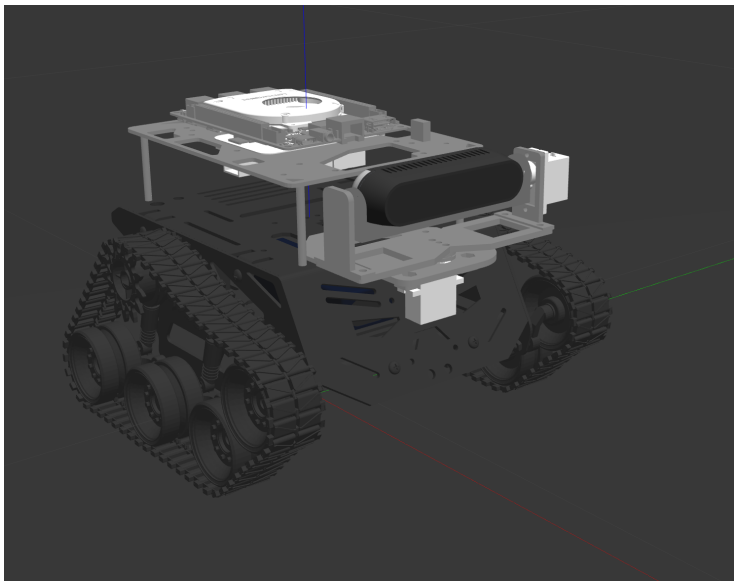


Figure 7.2: Devastator Model

Robot model in ROS contains packages to reproduce in the virtual environment the real robot configuration and proprieties: in particular, the package required to construct an object model is the URDF or Unified Robot Description Format, containing

- Kinematic and dynamic description of the robot

- Visual representation on the robot

- Collision model of the robot

Xacro files are used along with URDF files to simplify them; Xacro is a XML[2] macro language that makes it easier to manage robot description files [108]. *Robot_state_publisher* allows to publish the state of the robot and make it available to all the components in the system. What is called a robot is a collection of *link* parts and *joint* elements that connected them: for every joint it is possible to defined some characteristics and store them in the Robot model package.
Let us analyze the main package file extensions and elements:

- *.urdf.xacro*: it contains the kinematic model of the robot and attaches meshes to each link.
- *.gazebo.xacro*: it contains Gazebo model representation and properties of each link.
- *meshes* folder: it contains all the robot 3D part elements such as structure, motors, sensors.
- *.xacro* files: they are used to define other characteristics, such as color for each meshes and so on.

In these files all the essential features of the robot model are specified, in particular collision and inertial properties are necessary to introduce the robot model to the Gazebo environment because they are required by the physics engine for simulation [9]. A generic element in *.urdf.xacro* file is usually defined with a

- *Collision section*:

```
<link name="cube_base">
      <collision name="cube_base_collision">
        <origin xyz="0 0 0 " rpy="0 0 0"/>
        <geometry>
          <box size="0.19 0.134 0.047"/>
        </geometry>
      </collision>
</link>
```

- *Inertial section*:

```
<link name="cube_base">
      <inertial>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <mass value="8.2573504e-01"/>
        <inertia ixx="2.2124416e-03"
                 ixy="-1.2294101e-05" ixz="3.4938785e-05"
                 iyy="2.1193702e-03" iyz="-5.0120904e-06"
                 izz="2.0064271e-03" />
      </inertial>
</link>
```

[2]eXtensible Markup Language, it defines a set of rules for encoding documents in a format that is both human-readable and machine-readable [116]

It is also possible to define how the links behave when they are in contact with one another and how the joint moves in Gazebo environment by defining

- friction properties
- stifness properties
- dampening properties

```
<gazebo reference="wheel_right_link">
    <mu1>0.1</mu1>
    <mu2>0.1</mu2>
    <kp>500000.0</kp>
    <kd>10.0</kd>
    <minDepth>0.001</minDepth>
    <maxVel>0.1</maxVel>
    <!-- <material>Gazebo/FlatBlack</material> -->
  </gazebo>
```

with $mu1$ and $mu2$ the friction parameter in the first and second direction respectively, kp and kd the coefficients for stiffness and damping characteristics, $MinDepth$ the parameter indicating the minimum allowable depth before contact correction action is applied, $MaxVel$ the maximum contact correction velocity truncation term [44]. Once the model is available, a *.launch* file can be created to include all its component in the Gazebo environment.
Another element of fundamental importance is the modeling of the environment in which the mission will be carried out. By default this environment is an empty world: it appears as reported in Figure 7.3, i.e. essentially as a gray surface with basic reference. It constitutes the space in which the mission takes place and can be populated with elements, faithfully reproducing a real environment.

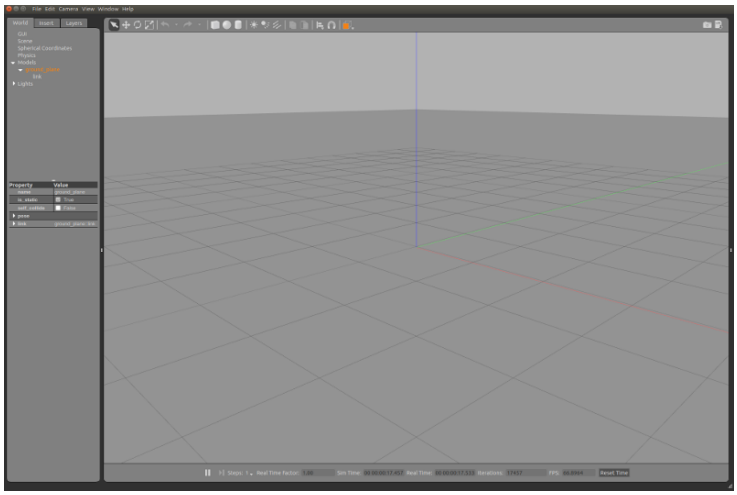


Figure 7.3: Gazebo interface [31]

7.3 ROS/Gazebo-MATLAB/Simulink Co-simulation

Gazebo is a very useful tool for testing algorithms in a realistic simulation environment; MATLAB/Simulink on the other hand is a very powerful modeling tool capable of faithfully reproducing the real behavior of the robot. It is therefore clear the advantage offered by a co-simulation that allows to integrate the functionalities of the two environments. By exploiting the potential offered by communication through ROS framework and the Robotics System Toolbox in Simulink, a link has been established between the two softwares in order to perform rapid tests on the functionality of the GNC algorithms.
ROS Toolbox creates a network of ROS nodes linked to Simulink models constituting the communication layer that allows the exchange of information through messages. A message in Simulink is represented by a *bus signal* [15] and manages by

- *Bus Creator block*: it creates a bus signal.

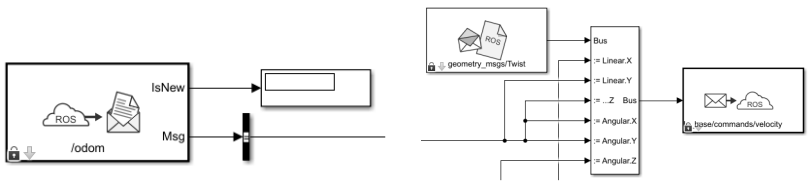
- *Bus Assignment block*: it receives as input a bus and allows the assignment of the selected signal in it with new values coming from Simulink environment.

- *Bus Selector block*: it accepts a bus as input and allows to select the output signals to be used in Simulink environment.

It is possible both to send messages by publishing them on an particular topic and to receive them by subscribing to that topic [32]. This happens thanks to the use of publisher and subscriber blocks:

- *Publisher block*: it sends message to ROS network. It is necessary to specify the topic and the message type.

- *Subscriber block*: it receives messages from ROS network. Two port are available: Msg port which outputs the new message and IsNew port that indicates if a new message is received (0 if not, else 1). It is necessary to specify the topic and the message type.



(a) *Subscriber* (b) *Publisher*

Figure 7.4: Simulink ROS blocks

The time of Simulation has been instead managed thank to the *Clock Message block*, whose output is a signal corresponding to the current time in seconds and nanoseconds.

7.3.1 Creation of the connections in the Simulink Model

The Guidance block used for the Devastator Simulink model has been adapted to make it suitable for communication with the ROS network. In particular the following blocks have been added:

- a subscriber for *Time* information: it receives a signal about the /clock topic, with rosgraph_msgs/Clock message type.
- a subscriber for *State* information: it receives a signal from /odom topic, with nav_msgs/Odometry message type.
- a blank message for the *Reference Signal* to be send back to Gazebo: the message type is geometry_msgs/Twist.
- a publish block for *Reference*: it sends the reference signals elaborated by the Guidance Block to Gazebo, under the custom message /simulink_references with geometry_msgs/Twist as message type.

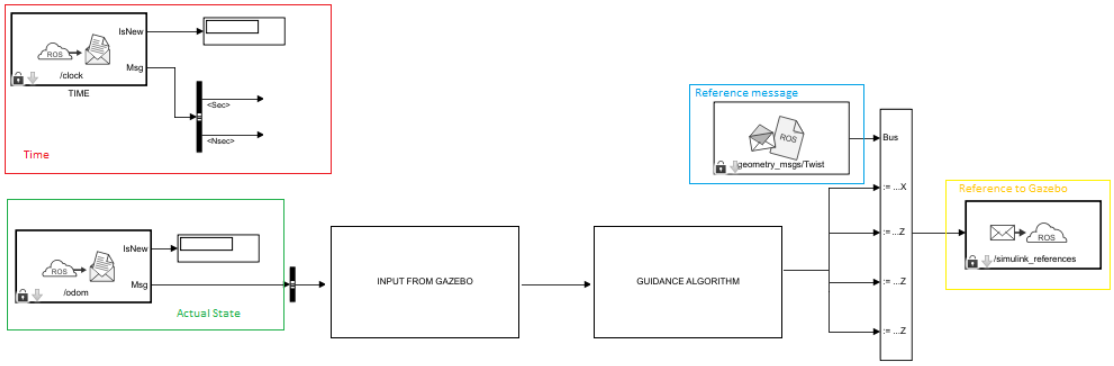


Figure 7.5: Elements for MATLAB/Simulink communication with ROS/Gazebo environment

The main elements necessary to start a joint simulation have been introduced. With the command

```
rosinit
```

in the workspace, the connection is effectively established allowing communication between the elements of the ROS network. Guidance algorithm tests are ready to be performed.

7.4 Code Generation

Another important possibility offered by the use of MATLAB/Simulink framework is the obtainment of the code of the developed algorithms ready to be directly integrated with Devastator system. The *Code Generation* function is part of the series of processing algorithms available thanks to MATLAB Coder and Simulink Coder: starting from a Simulink model it is possible to obtain the algorithms transcription in C or C ++ code and deploy it in a wide variety of applications, including the loading of GNC algorithms on the robot's on-board computer so to make its navigation autonomous.
First of all, the GNC blocks have been rearranged so to have a single Guidance and Control block. The Navigation block has been completed by the introduction of a Kalman filter to provide system status. The compatibility of the blocks that make up the system has been checked, bearing in mind that plot or communication functions with the workspace such as Scope or To and From workspace are ignored.
In order to configure a model for Code Generation, in the Model Configuration panel it has been specified a solver that is compatible with the system target; the mainly used solver type is the *Fixed-step* solver (few systems support variable-step ones). After selecting the solver, the Hardware Implementation panel must be opened to select [109]

- Hardware board, where the code generated from the model will run
- Device Vendor, for the selection of the available microprocessors
- Device type, associated with the device vendor

In Code Generation panel the System target file browser and configuration parameter must be set. It is possible to choose between

- grt.tlc generic real-time target
- ert.tlc embedded real-time target
- ert_shrlib.tlc embedded real-time target shared library
- Other

The second one is chosen for Devastator GNC conversion. C++ language is also selected. In the Interface panel the voice Reusable function is selected: in this way I/O values are passed as arguments to the function and not as global data such as for Non-reusable function. The way in which inputs and outputs are managed is defined choosing between Individual arguments, Structure reference and Part of model data structure: in the first case input and output arguments are passed individually, in the second as part of separate structures while in the last as model data structure.
Once all the parameters have been set, it is possible to proceed with the generation of the code.
Several files are created from the Code Generation process, the function of which is [109]

- to provide the public interface to the model entry points
- to list the types corresponding to built-in data types

- data structures describing model signals, states and parameters

In the generated folder there are a series of model.* files that have been created in order to support shared utilities and model references. Among them, a set of HTML[3] files are created containing the description of every source file plus a general summary. In particular, when the model_contents is opened, two section are present:

Contents

- the *Summary* contains information on the model and the code:
  - Model information such as author, model version, tasking mode.
  - Code information such as system target file, hardware device type, Simulink Coder version, type of build.
- the *Subsystem Report* reports information about the model subsystems and their reusability.
- the *Code Interface Report* contains the structures of
  - Entry-Point Functions
  - Inports
  - Outports
  - Interface Parameters
  - Data Stores
- the *Traceability Report* provides a complete mapping between model elements and code thanks to its structure of the type Traceable Simulink Blocks/Stateflow Objects/MATLAB Scripts.
- the *Static Code Metrics Report* provides generated code statistics.
- the *Code Replacements Report* contains information about code replacement library functions that were used during code generation.
- the *Coder Assumptions* is a list of assumptions respected in the generation of the code for the selected target environment, in particular about the type of data, their characteristics, etc.

Generated Code

- the Main file
  - ert_main.cpp, an example main program for the model, it controls model code execution.
- the *Model files*

[3]Hypertext Markup Language (HTML) is the standard markup language for documents [40]

  - model.cpp, the model file.
  - model.h, it defines model data structures and the interface to the model entry points.
  - model_private.h, with the local define constants and local data for the model structure and the constant block I/O data structure.
  - model_types.h, with forward declarations for the real-time model data structure and parameters.

- the *Data files*
  - model_data.cpp, with declarations for the parameters data.

- the *Utility files*
  - *rt.** files for initialization of inf, minus inf and nan, and the management of the parameters required by the code.

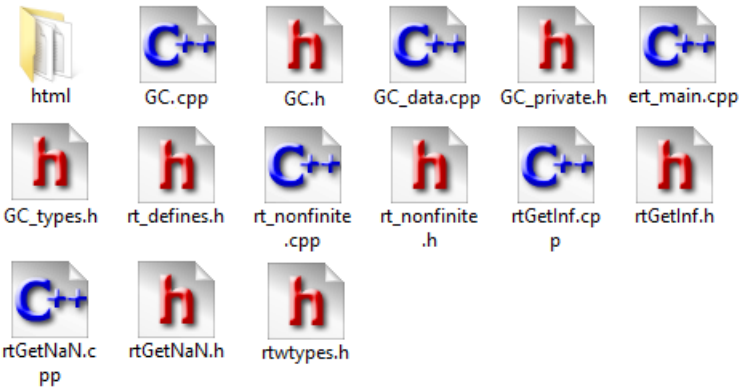


Figure 7.6: Code Generation Files

The obtained files can be recalled and integrated into the processes that characterize the robotic platform. Once the information exchange interface has been configured, i.e. the inputs assigned and the outputs defined, it is possible to immediately test the functionality of the translated algorithm.

7.4.1 Guidance and Control Code Generation

As previously introduced, the Guidance and Control blocks have been merged into a single subsystem that is responsible for generating the speed and orientation references starting from the estimated state of the system and able to control the actuators in order to follow the references. Goal and obstacles positions as well as the value of all the controller parameters have been defined in the data file. As input the G&C block receives the state estimated by the Kalman filter (fusion of the odometry data from the encoder, accelerometers, magnetometers and the visual odometry from the depth detection camera), i.e. position X, Y,

orientation ψ and speed of the robot V_x, while in output provides PWM values for the left and right motors.

Chapter 8

Simple missions in Indoor Environment

In chapter 6 the Robot model realized in the MATLAB/Simulink environment has been presented. In chapter 7 the possibility of testing the Guidance algorithm on a pre-existing model in ROS/Gazebo framework has been introduced and its effectiveness evaluated. The Code Generation of the Guidance and Control algorithm has been performed so to have its integration into the on-board processes of the robotic platform.
Let us consider two simple missions for the Devastator:

- Reaching a Goal point in a free save environment

- Reaching a Goal point in an environment with obstacles

Both missions are carried out indoors, thus excluding the possibility of navigating using tools such as GPS (widely used in outdoor applications). A combination of the information coming from sensors performed by the Kalman filter is required for the activation of the Guidance and Control algorithms. In particular, the data fusion involves encoders, accelerometers, magnetometers and a depth detection camera with the aim of obtaining the position in the Inertial Frame of the robot, its linear and angular velocity and its orientation. Thanks to the possibility of exploiting a set of different sensors, it is possible to correct the errors intrinsically linked to their use. Information coming only from the encoders would lead to an incorrect calculation of the position given the presence of slips that they are not able to identify, but the use of the depth detection camera allows the integration of the position information to obtain a more reliable status. The filter performance optimization path has been undertaken, which is why the mission subsequently presented performed with the Devastator platform shows little accuracy in achieving the goal.

8.1 MATLAB/Simulink simulations

Some tests have been carried out using the model presented in chapter 6. The obtained results and some considerations are reported below.

8.1.1 Obstacles free path

A simple mission is considered: the aim is to achieve a goal placed 1 meter in front of the robot. Below there are the simulation input parameters, i.e. the coordinates of the goal and obstacles, and the gain values that characterize the orientation and speed controllers, i.e. the proportional and integrative gains for the PI that involves speed and only proportional gain for the P that involves orientation.

```
%% Mission Parameters
X_goal=1;                % Desired X position [m]
Y_goal=0;                % Desired Y position [m]
X_0=0;                   % Initial X position [m]
Y_0=0;                   % Initial Y position [m]
psi_0=0;                 % Initial orientation [rad]
ox=-;                    % Obstacle X position [m]
oy=-;                    % Obstacle Y position [m]

%% Controller
% PID section, Speed
Kp_Vx=2;
Kd_Vx=0;
Ki_Vx=1.85;

% PID section, Orientation
Kp_psi=0.4;
Kd_psi=0;
Ki_psi=0;
```

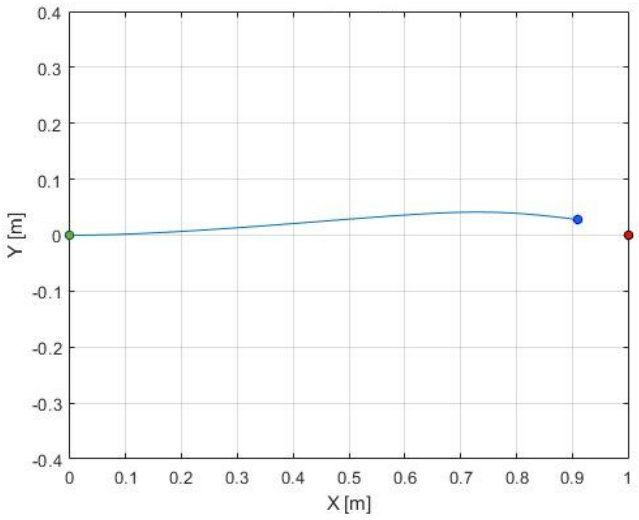
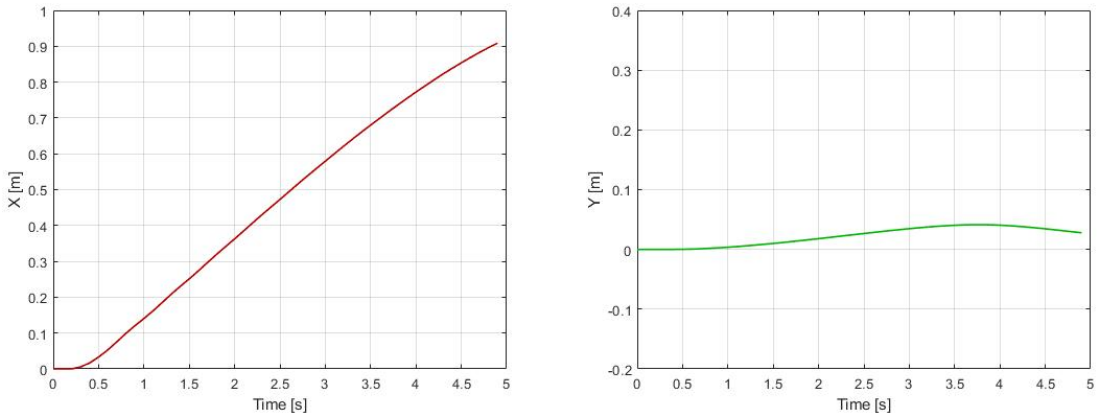


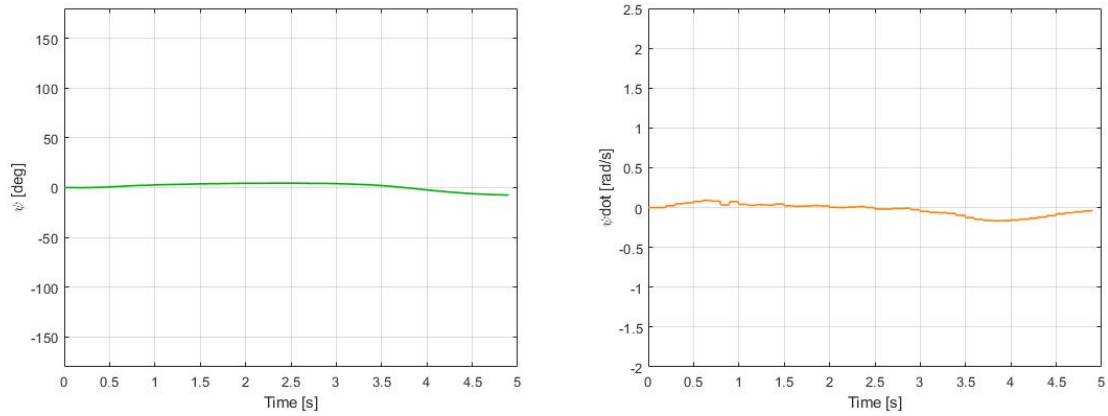
Figure 8.1: MATLAB/Simulink Trajectory in an obstacles free environment

The path followed by the robot to complete the mission is shown in Figure 8.1. The reason why the final position is not exactly in the goal is linked to the tolerance imposed on reaching the final point: once a distance contained in a circle of 10 cm from the desired arrival is reached, the mission is considered concluded. The total time taken by the robot is 4.9 seconds, with a displacement of approximately 91 cm in the X direction and a final position of approximately 2.8 cm in the Y direction (Figure 8.2). The lateral movement during the path is a characteristic of the robot itself: due to a different tension of the tracks, a slight difference in their length, a distribution of mass that is not perfectly symmetrical and so on, an identical command to the right and left motors generates a slight rotation of the robot to the left, bringing a small shift of some centimeters in the Y direction (in this case about 4 cm at most). As it can be seen, this shift increases in the first 4 seconds approximately, then decreases: this can be explained by observing what happens in the orientation reference (Figure 8.5). In the first seconds the ideal orientation and the robot one coincide, so no inputs are applied (it can be observed in Figure 8.4 that the PWM signals for the right and left motors are identical). When the orientation error begins to increase, the Control System is activated to increase the speed of the left track and decrease that of the right one: this generates the creation of a slight negative angular speed which leads to the clockwise rotation of the robot (Figure 8.3) and the decrease of the error with the reference. In such a short mission and for such a close distance, given the limitations on the maximum speed imposed, the "jagged" behavior of the speed is not surprising: on the one hand, there is the data sampling period (0.1 second), on the other hand the model itself, not perfectly reliable for PWM below 5000 μs.



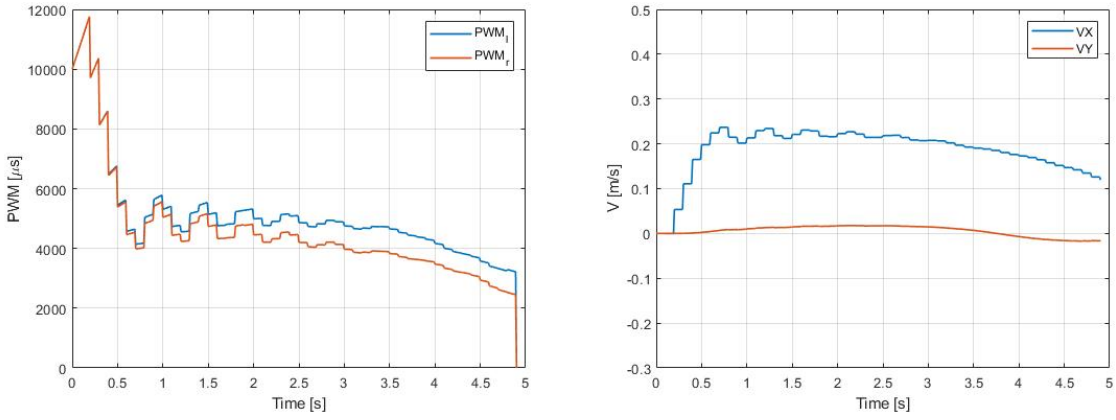
(a) *Position X* (b) *Position Y*

Figure 8.2: MATLAB/Simulink Position in an obstacles free environment



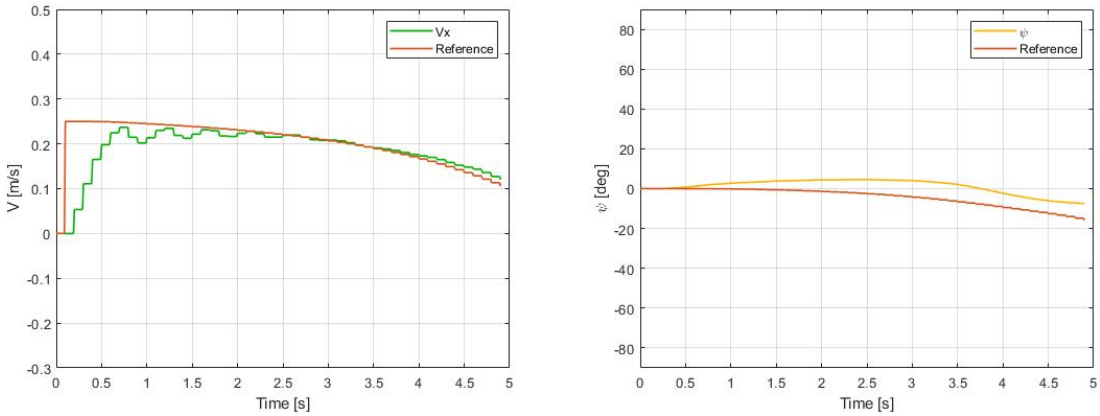
(a) *Orientation* ψ (b) *Angular speed* $\dot{\psi}$

Figure 8.3: MATLAB/Simulink Orientation and Angular speed in an obstacles free environment



(a) *PWM* (b) *Velocity*

Figure 8.4: MATLAB/Simulink Velocity in an obstacles free environment



(a) *Velocity reference* (b) *Orientation reference*

Figure 8.5: MATLAB/Simulink Reference in an obstacles free environment

8.1.2 Environment with obstacles

It is imposed the presence of obstacles in the environment. The goal is positioned 6 meters in the X direction and 3 m in the Y one. The controller values are the same as in the previous mission.

```
%% Mission Parameters
X_goal=6;                          % Desired X position [m]
Y_goal=3;                          % Desired Y position [m]
X_0=0;                             % Initial X position [m]
Y_0=0;                             % Initial Y position [m]
psi_0=0;                           % Initial orientation [rad]
ox=[2   3   4   5];                % Obstacle X position [m]
oy=[3.5   1   2.5   3];            % Obstacle Y position [m]

%% Controller
% PID section, Speed
Kp_Vx=2;
Kd_Vx=0;
Ki_Vx=1.85;

% PID section, Orientation
Kp_psi=0.4;
Kd_psi=0;
Ki_psi=0;
```

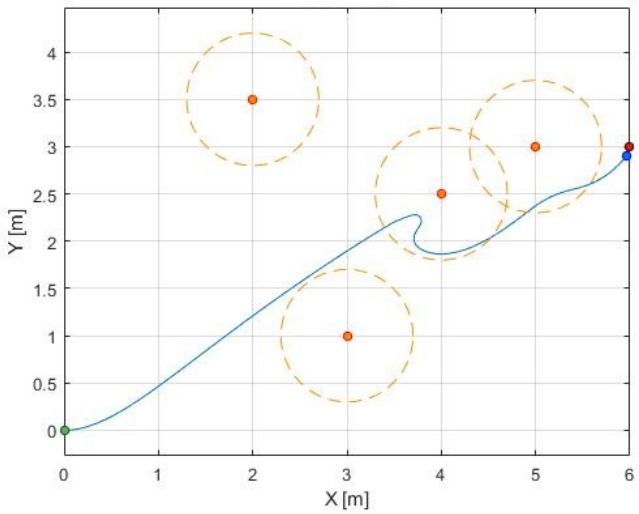
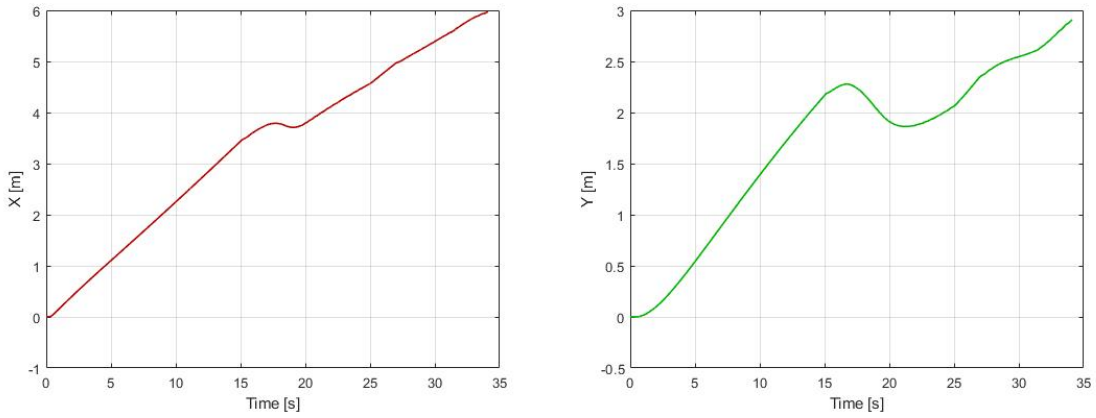


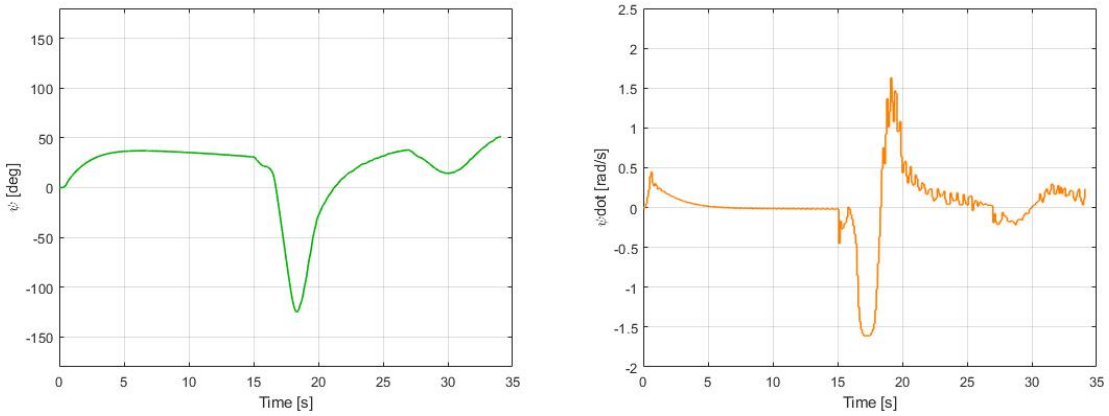
Figure 8.6: MATLAB/Simulink Trajectory in an environment with obstacles

The trajectory followed in this case is reported in Figure 8.6. The obstacle in position (3, 1) and the one in position (2, 3.5) do not in any way influence the path of the robot,



(a) *Position X* (b) *Position Y*

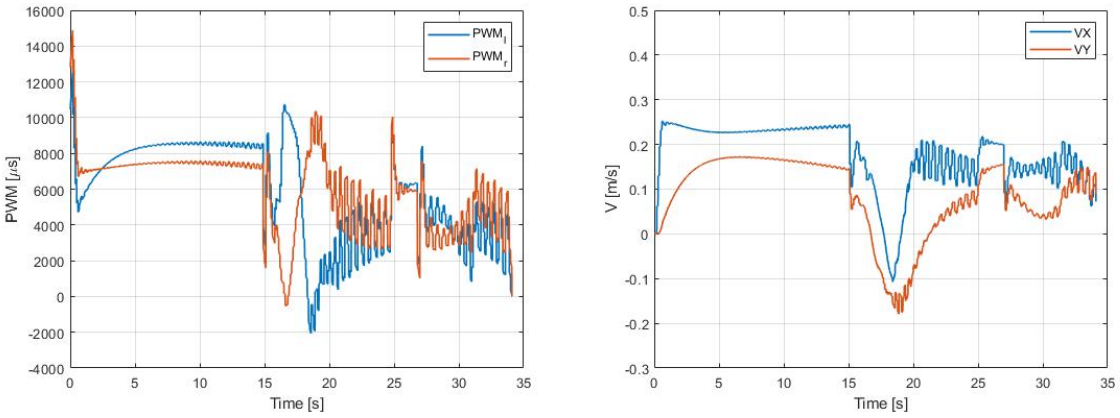
Figure 8.7: MATLAB/Simulink Position in an environment with obstacles



(a) *Orientation* ψ (b) *Angular speed* $\dot{\psi}$

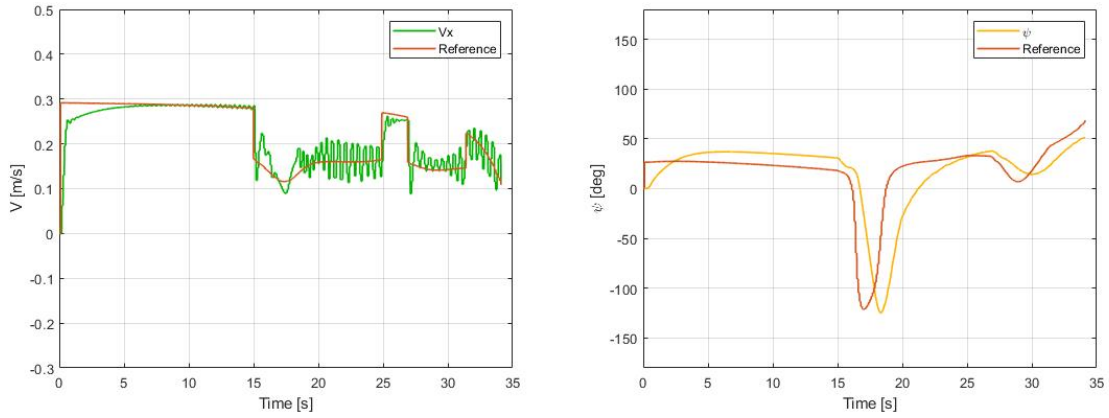
Figure 8.8: MATLAB/Simulink Orientation and Angular speed in an environment with obstacles

while the other two obstacles prevent the achievement of the goal by continuing to follow the undertaken trajectory and force the robot to deviate to avoid a collision. However, the mission is a success: the final position of the robot is 5.97 m in the X direction and 2.9 m in the Y one (Figure 8.7), reached in 34.1 seconds. Faithful tracking of the speed reference can be observed as well as, although less rapidly, that of orientation. Obstacles represent a disturbance in the path: after about 15 seconds the robot enters the area of influence of the obstacle, becomes aware of its presence and undertakes an orientation change maneuver such as to take it out of the danger area and resume navigation. The last part of the mission is characterized by a "lively" behavior of the speed signal: this is again due to the fact that the PWM signal has dropped below the threshold which guarantees



(a) *PWM* (b) *Velocity*

Figure 8.9: MATLAB/Simulink Velocity in an environment with obstacles



(a) *Velocity reference* (b) *Orientation reference*

Figure 8.10: MATLAB/Simulink Reference in an environment with obstacles

optimal behavior of the model (Figure 8.9), but on the other hand this is unavoidable given the constraints imposed by safety. If the robot enters a dangerous area and is close to an obstacle, the reference speed is considerably reduced to avoid that, when attempting to carry out the escape maneuver, it gets too close to the obstacle and does not have the time to avoid it.

8.2 ROS/Gazebo simulations

In order for the simulation to take place, it is necessary to enter the inertial parameters and the friction characteristics of the robot: a mass of 1.5 kilograms and a friction of 0.1 for the interaction of the tracks with the ground have been considered. The reason for the latter value is linked to the low grip that the track exhibits on a paved surface.

8.2.1 Obstacles free path

Again it is considered a simple mission in an obstacle-free environment. The aim remains to reach a goal located 1 meter in front of the robot, defined in Figure 8.11 with a red square.

```
%% Mission Parameters
X_goal=1;          % Desired X position [m]
Y_goal=0;          % Desired Y position [m]
X_0=0;             % Initial X position [m]
Y_0=0;             % Initial Y position [m]
psi_0=0;           % Initial orientation [rad]
ox=-;              % Obstacle X position [m]
oy=-;              % Obstacle Y position [m]
```

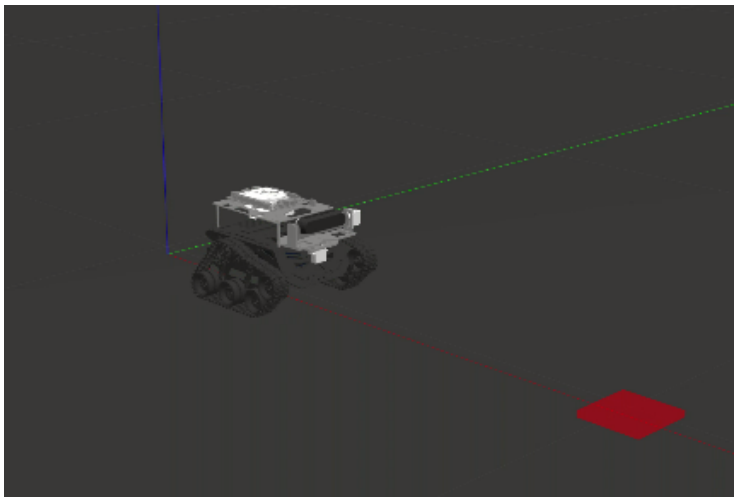


Figure 8.11: Devastator Robot and Goal

Contrary to the MATLAB/Simulink model, the Gazebo model does not exhibit the lateral displacement caused by the asymmetry of the behavior of the two tracks: it could be interpreted as the ideal behavior that the robot would have if construction imperfections or disturbances were not present. The goal is reached in about 5 seconds, with a straight trajectory and a final position of 93 cm in the X direction and 0 cm in the Y direction (Figure 8.12 and Figure 8.13). The reference is faithfully followed (Figure 8.16) and only a unusual behavior near the goal is observed, probably a small impulse of the propulsion system in an attempt to get slightly closer to the goal.

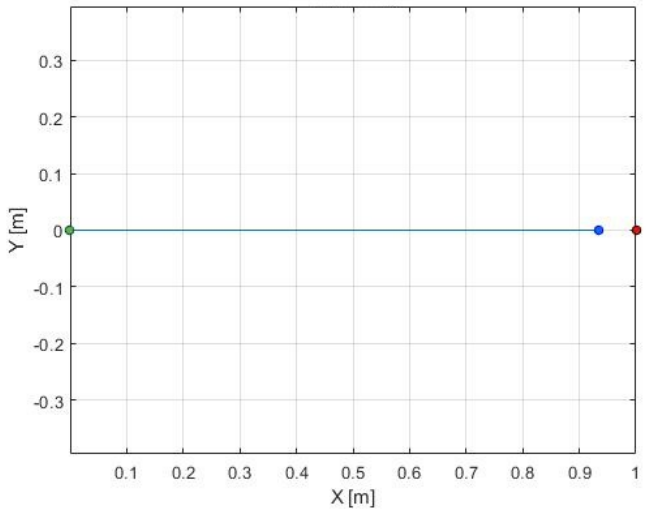
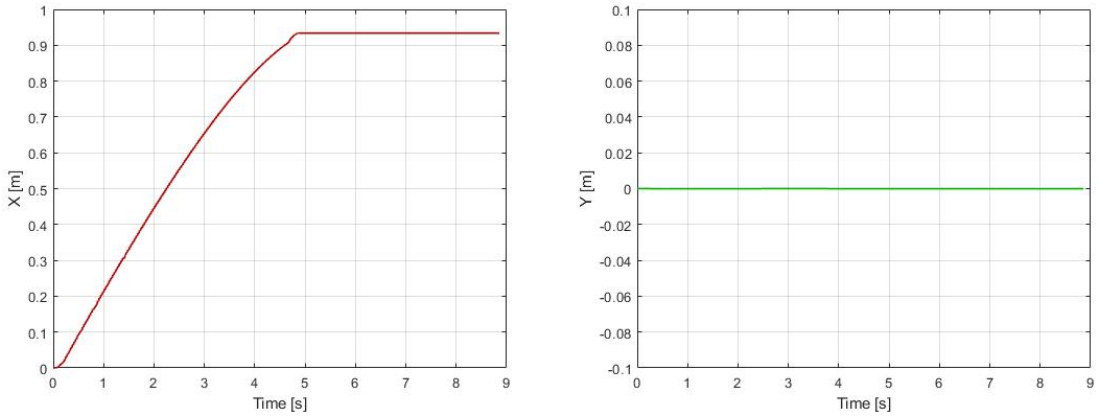
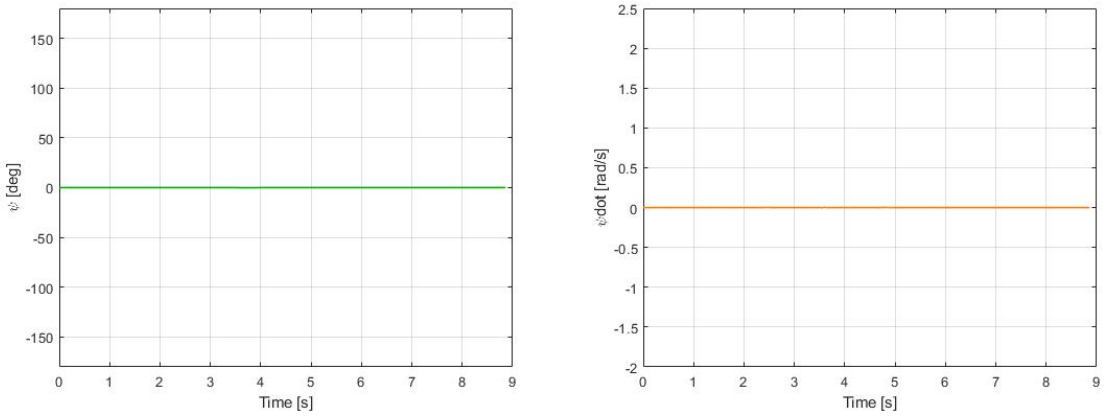


Figure 8.12: ROS/Gazebo Trajectory in an obstacles free environment



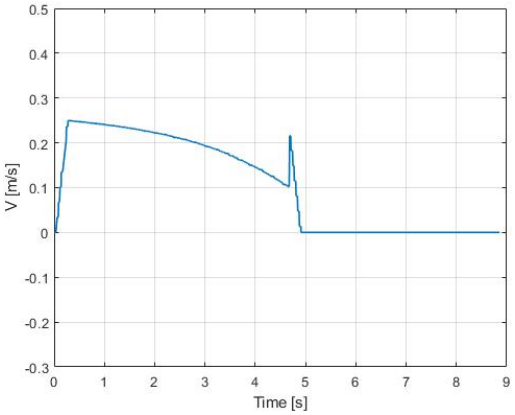
(a) *Position X* (b) *Position Y*

Figure 8.13: ROS/Gazebo Position in an obstacles free environment



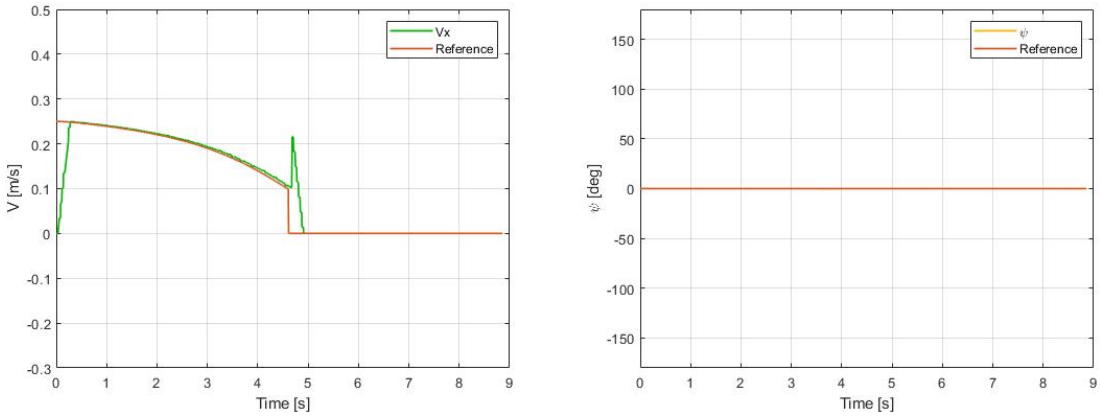
(a) *Orientation* ψ (b) *Angular speed* $\dot{\psi}$

Figure 8.14: ROS/Gazebo Orientation and Angular speed in an obstacles free environment



(a) *Velocity*

Figure 8.15: ROS/Gazebo Velocity in an obstacles free environment



(a) *Velocity reference* (b) *Orientation reference*

Figure 8.16: ROS/Gazebo Reference in an obstacles free environment

8.2.2 Environment with obstacles

The goal is positioned 6 meters in the X direction and 3 m in the Y one such as for the mission performed in MATLAB/Simulink environment. A series of obstacles with the following characteristics have been added, represented by orange cylinders of 25 cm in height and 15 cm in radius.

```
%% Mission Parameters
X_goal=6;                          % Desired X position [m]
Y_goal=3;                          % Desired Y position [m]
X_0=0;                             % Initial X position [m]
Y_0=0;                             % Initial Y position [m]
psi_0=0;                           % Initial orientation [rad]
ox=[2   3   4   5];                % Obstacle X position [m]
oy=[3.5   1   2.5   3];            % Obstacle Y position [m]
```

```
% Obstacles Definition in Gazebo world
<model name="Obstacle">
      <static>False</static>

      <link name="obstacle">
          <inertial>
            <mass>15</mass>
            <origin xyz="0.0 0 0" rpy=" 0 0 0"/>
            <inertia>
              <ixx>1.0</ixx> <ixy>0</ixy> <ixz>0</ixz>
              <iyy>1.0</iyy> <iyz>0</iyz>
              <izz>1.0</izz>
            </inertia>
          </inertial>
          <pose>2 3.5 0.5 0 0 0</pose>
          <collision name="collision">
          <geometry>
            <cylinder>
              <length>0.25</length>
              <radius>0.15</radius>
            </cylinder>
          </geometry>
          <surface>
            <friction>
              <ode>
                <mu>1</mu>
                <mu2>1</mu2>
              </ode>
            </friction>
          </surface>
        </collision>
```

```
      <visual name="visual">
        <geometry>
          <cylinder>
            <length>0.25</length>
            <radius>0.15</radius>
          </cylinder>
        </geometry>
        <material>
          <script>
            <name>Gazebo/Orange</name>
          </script>
        </material>
      </visual>
    </link>
</model>
```

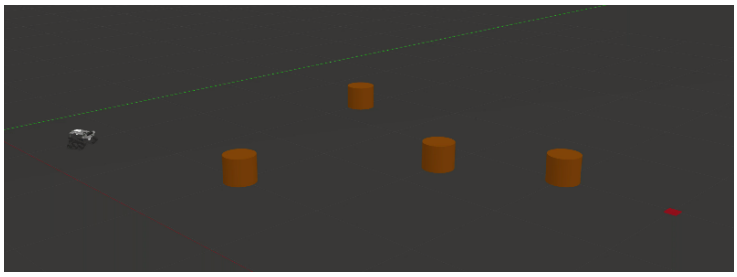


Figure 8.17: Devastator Robot, Goal and Obstacles

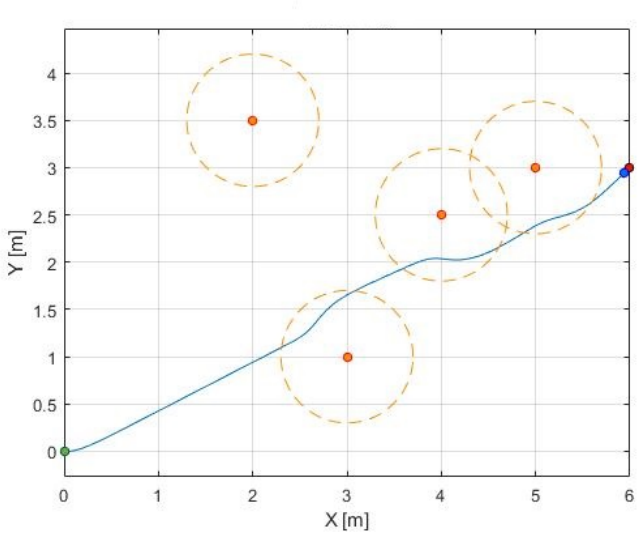
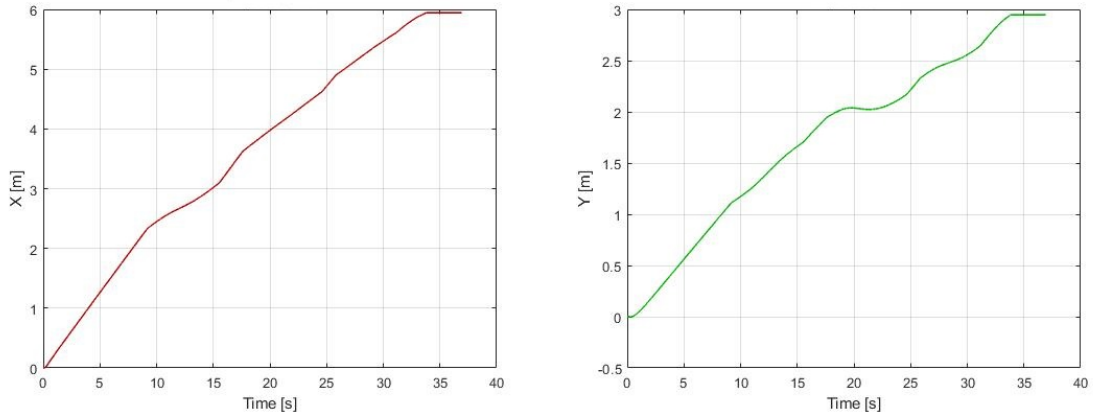


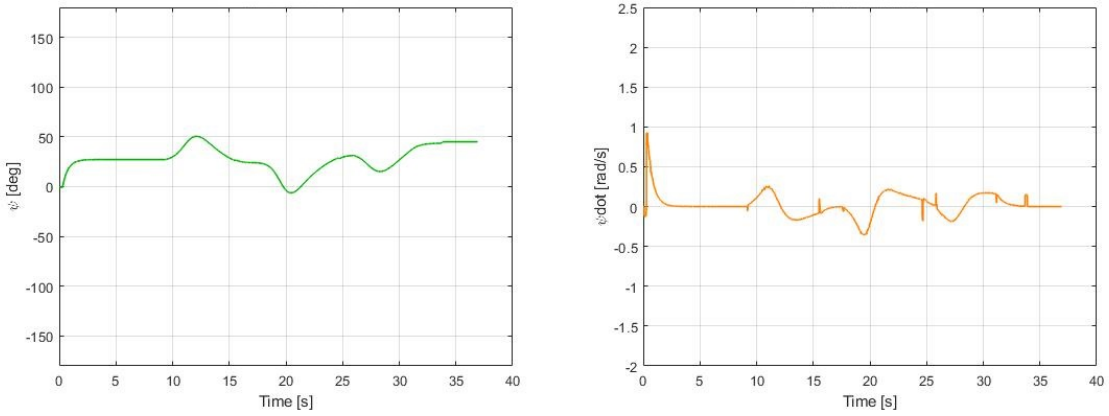
Figure 8.18: ROS/Gazebo Trajectory in an environment with obstacles

The followed path is very similar to that generated by the same mission carried out in MATLAB/Simulink. Contrary to that case, however, the lack of lateral movement leads the robot to intercept also the area of influence of the obstacle in position (3, 1), making a deviation necessary to move the robot away from the obstacle. Similarly it happens near the following obstacles. A final position of 5.94 m along X and 2.95 m along Y is reached in 34 seconds.



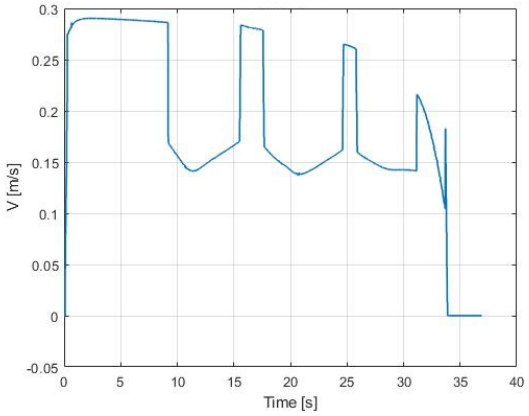
(a) *Position X* (b) *Position Y*

Figure 8.19: ROS/Gazebo Position in an environment with obstacles



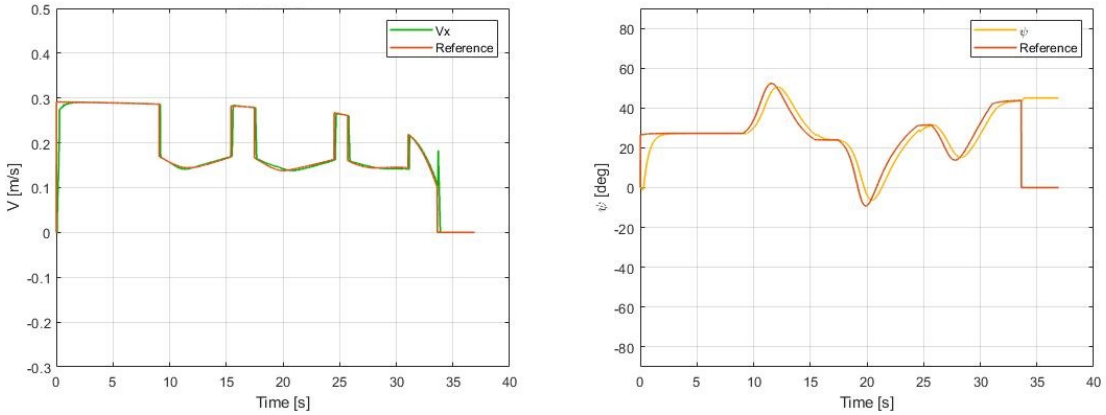
(a) *Orientation* ψ (b) *Angular speed* $\dot{\psi}$

Figure 8.20: ROS/Gazebo Orientation and Angular speed in an environment with obstacles



(a) *Velocity*

Figure 8.21: ROS/Gazebo Velocity in an environment with obstacles



(a) *Velocity reference* (b) *Orientation reference*

Figure 8.22: ROS/Gazebo Reference in an environment with obstacles

8.3 Real Robot mission

In order to validate the successful translation of the G&C algorithm and demonstrate that the tuning carried out using the simulation model is effective, a simple mission has been performed with the Devastator robot.

```
%% Mission Parameters
X_goal=1;                % Desired X position [m]
Y_goal=0;                % Desired Y position [m]
X_0=0;                   % Initial X position [m]
Y_0=0;                   % Initial Y position [m]
psi_0=0;                 % Initial orientation [rad]
ox=-;                    % Obstacle X position [m]
oy=-;                    % Obstacle Y position [m]

%% Controller
% PID section, Speed
Kp_Vx=2;
Kd_Vx=0;
Ki_Vx=1.85;

% PID section, Orientation
Kp_psi=0.3;
Kd_psi=0;
Ki_psi=0;
```

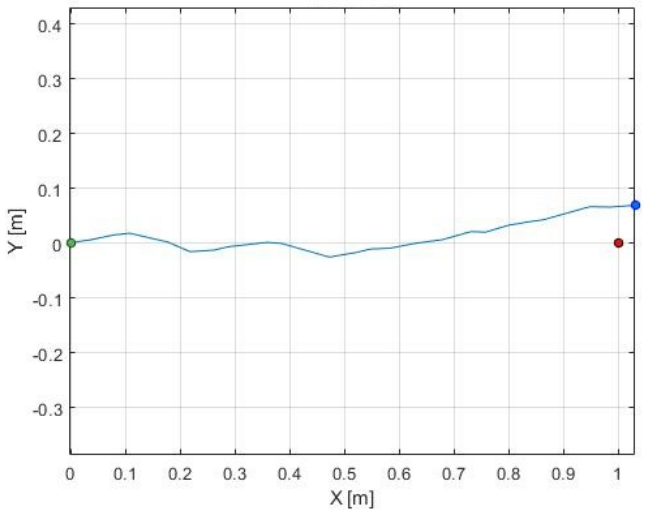
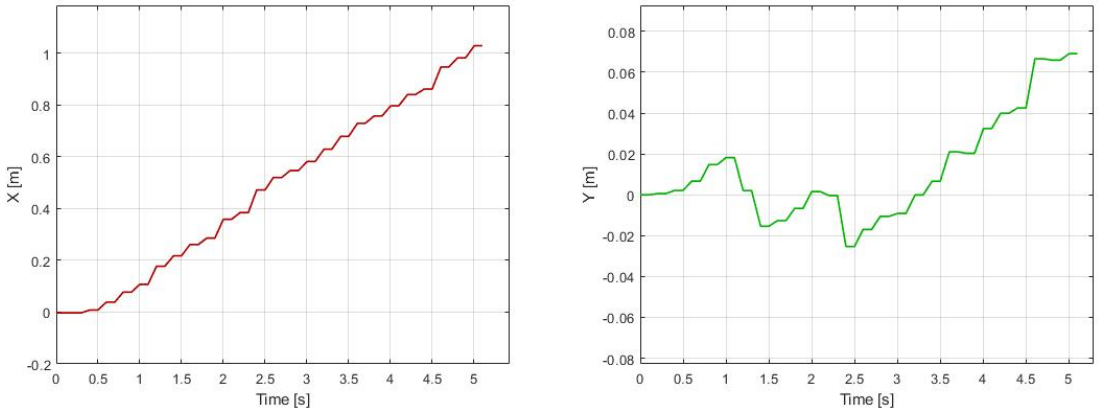


Figure 8.23: Real Robot Trajectory

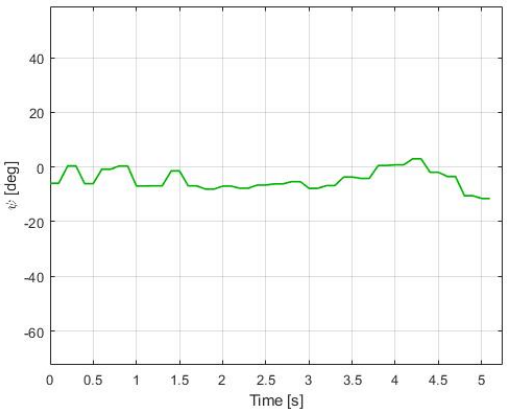
The mission is a success: the robot stops at 1.03 m in the X direction and at 0.07 m in the Y direction (Figure 8.23 and Figure 8.24), with a mission duration of about 5 seconds, in

accordance with the simulations performed in the MATLAB/Simulink and ROS/Gazebo environments. A faithful representation of the state of the robot is not available due to the Kalman filter optimization still in progress, however it is observable that the real behavior of the robot is in line with what was expected.



(a) *Position X* (b) *Position Y*

Figure 8.24: Real Robot Position



(a) *Orientation ψ*

Figure 8.25: Real Robot Orientation

8.4 Comparisons

The following discussion aims to compare simulations conducted in MATLAB/Simulink and ROS/Gazebo environments with the Devastator robotic platform mission.
For the first mission, the paths described in the three cases are shown in Figure 8.26.
The ideal trajectory is represented by a straight line connecting the starting point to the ending point. The velocity reference is such that the speed decreases along the path until it disappears near the arrival, while the orientation reference suggests to keep the initial orientation until the goal.
The trajectory that is the most faithful to the reference is the one followed by the model in Gazebo simulation: the approach is perfectly straight and the stop occurs at 93 cm, i.e. within the tolerance area around the goal in which the mission is considered completed. The trajectory followed by the Simulink model instead shows a shift to the left of a few centimeters, which is then recovered near the goal: as previously reported, this behavior is to be attributed to the imperfections of the robot found during the identification process, imperfections for which the two tracks show a slightly different behavior although the input is identical. The mission is concluded at a distance of 91 cm, i.e. as soon as the Guidance and Control System reveals that the robot is inside the arrival area. The trajectory actually followed by the robot is characterized by uncertainties due to the noise of the sensors used for navigation. The true trajectory approximates the one represented and highlights the same shift to the left already observed in the model. The finish is at 1.03 m.

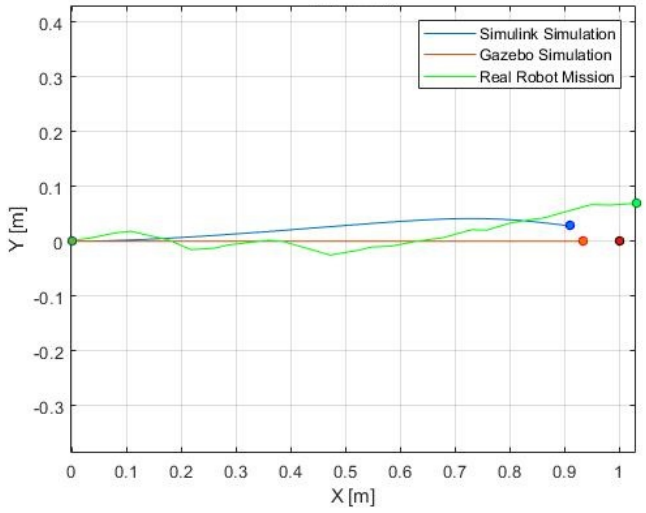


Figure 8.26: Comparison between trajectories

In particular, the Figure 8.27 reports the position in the X direction for the Simulink model simulation and for the real robot mission. The behavior of the robot is very close to that expected by the simulation. The timing of the mission is also similar for all three cases.
For the second mission, a comparison is only possible between the simulations carried out in the MATLAB/Simulink and ROS/Gazebo environments. The described trajectories are

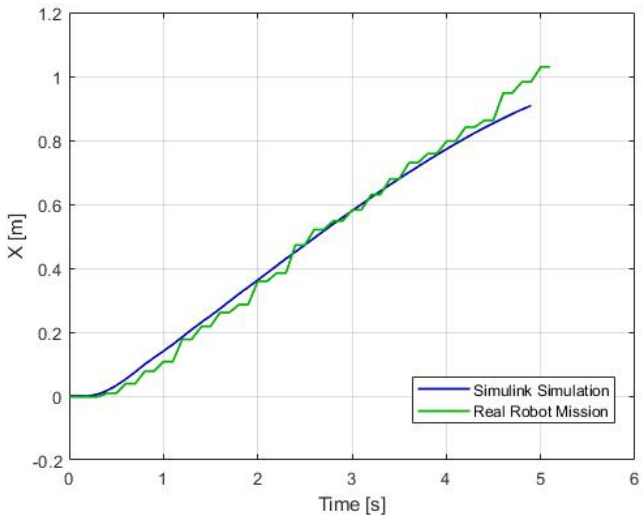


Figure 8.27: Position X comparison

reported in Figure 8.28.

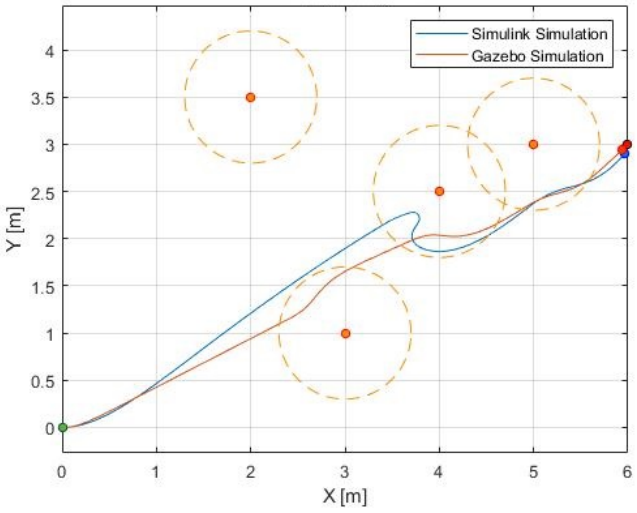


Figure 8.28: Comparison between Simulink and Gazebo simulated trajectories

Interesting considerations can be reported by observing the behavior of the Guidance System in the two cases (Figure 8.29 and Figure 8.30). In the first 6 seconds of the mission the references are identical: a speed close to 0.3 m/s and an orientation of about 27 degrees are required. The initial trajectory of the two models is in fact very similar, but as the mission proceeds, a more pronounced deviation to the left of the model created in the Simulink environment is noted. The deviation is such that it does not intercept the danger area around the first obstacle, so no discontinuity appears in the references.

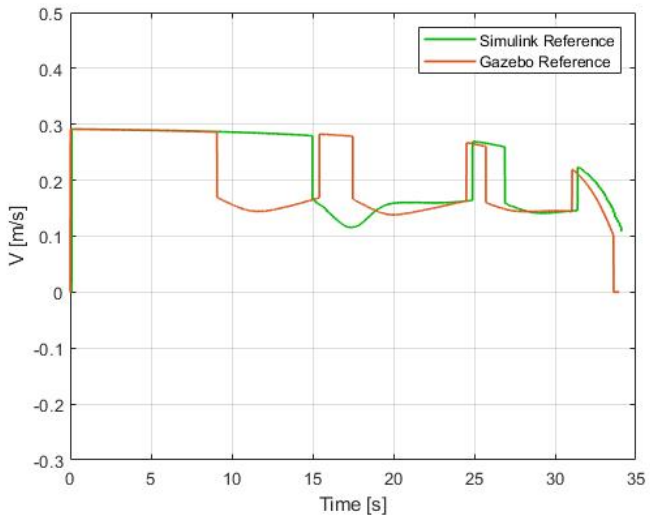


Figure 8.29: Velocity Reference comparison

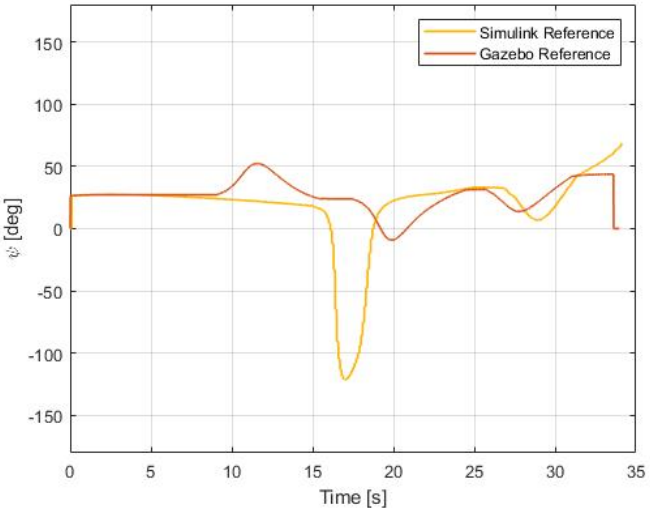


Figure 8.30: Angular Reference comparison

On the other hand in the Gazebo simulation the robot intercepts this area: the Guidance System is activated and the velocity reference is immediately lowered in an attempt to decrease the speed of the robot. The orientation reference suggests an increase of the ψ angle of about 30 degrees in order to make a left turn and move the robot away from the obstacle. In the Gazebo simulation the robot overcomes the obstacle, meanwhile in the one performed in Simulink the robot encounters its first one. Again the Guidance System suggests a decrease in speed and a change of direction. These are changes of greater entity than the ones suggested in the Gazebo simulation due to the fact that the robot risks

a head-on collision with the obstacle. Thanks to these indications, the robot is able to change its trajectory and move away from the obstacle, avoiding the impact. After around 17 seconds the robot in the Gazebo Simulation enters the area of influence of the second obstacle. Similarly the speed is decreased and a slight right turn is suggested. The same happens with the third obstacle encountered after 26 seconds. The robot in the Simulink simulation encounters its second obstacle after around 27 seconds: this time the Guidance System imposes a change of direction of about 30 degrees to the right, an orientation which then suggests to decrease in view of reaching the goal.
The two missions end at about the same time, with a final position differing by a few centimeters.

Conclusion

This thesis is intended to be an introduction into the world of small UGVs and an attempt to create a model to be studied in a simulation environment. The mathematical model of a tracked mobile robot such as the Devastator has been introduced, as well as a series of Modeling and System Identification techniques to identify its characteristics. A change in the robot's design strategy provided the opportunity to compare two completely different identification methods: the lumped parameter technique and the data-driven one. The former provided the characterization of the brushed DC motors and their response in terms of angular velocity and torque to a variation of external load and supply voltage. The latter provided the behavior of the entire platform based on a change of PWM signal in input to the motors.
The black-box approach has been preferred given the immediate availability of the PWM-angular speed relationship and in order to not require further experiments to characterize the drag exerted by the track on the motor. This led to the abandonment of a dynamic modeling of the body, the bases of which have however been defined by identifying the inertia to the rotation of the system through the experiment of the composite pendulum.
A simple Guidance and Control algorithm has been implemented with the aim of translating it into executable code for the robot's on-board computer and validating the model obtained by experimentation. An APF strategy has been developed: an attractive field towards the goal and a repulsive one towards obstacles have been defined, while the tracking of references has been ensured thanks to the implementation of a PI for the speed and a P controllers for the orientation signals.
The effectiveness of the algorithm has been validated in Gazebo environment using a pre-existing model of the Devastator with the same geometric characteristics of the real one by a ROS/Gazebo-MATLAB/Simulink co-simulation.
Through the MATLAB Code Generation function, the transformation of the Guidance and Control System into executable code has been prepared. The success of parameters tuning carried out by simulation has been also demonstrated though experimental tests.
Future projects will include the integration of a dynamic model to identify slippage and improving navigation performance, the optimization of the Kalman filter implemented for the fusion of the information coming from sensors and the inclusion of further of them in order to obtain a more precise mapping of the surrounding environment, letting different Guidance and Control algorithms to be tested (Sliding mode Controllers, for example).
The development possibilities are various and this is one of the several potentialities offered by mobile robots, i.e. lending themselves to the experimentation of many different Guidance, Control and Navigation strategies. The System Identification procedure has been

only the first step towards the birth of an autonomous and intelligent system.

Bibliography

[1] L.A Aguirre and S.A Billings. *Relationship Between the Structure and Performance of Identified Nonlinear Polynomial Models*. eng. Department of Automatic Control and Systems Engineering, 1993.

[2] M. Ahmad, V. Polotski, and R. Hurteau. "Path tracking control of tracked vehicles". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 3. 2000, 2938–2943 vol.3. DOI: 10.1109/ROBOT.2000.846474.

[3] *AIC*. https://it.mathworks.com/help/ident/ref/idgrey.aic.html.

[4] *Autocorrelation*. https://en.wikipedia.org/wiki/Autocorrelation.

[5] Alexandru Barsan. "Position Control of a Mobile Robot through PID Controller". In: *Acta Universitatis Cibiniensis. Technical Series* 71 (Dec. 2019), pp. 14–20. DOI: 10.2478/aucts-2019-0004.

[6] S. A. Billings, S. Chen, and M. J. Korenberg. "Identification of MIMO non-linear systems using a forward-regression orthogonal estimator". In: *International Journal of Control* 49.6 (1989), pp. 2157–2189. DOI: 10.1080/00207178908559767.

[7] S.A Billings and L.A Aguirre. *Effects of the Sampling Time on the Dynamics and Identification of Nonlinear Models*. eng. Department of Automatic Control and Systems Engineering, 1994.

[8] Stephen Billings. *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio–Temporal Domains*. eng. John Wiley & Sons, Inc, 2013. ISBN: 1-119-94359-0.

[9] Kumar Bipin. *Robot Operating System Cookbook*. eng. 1st ed. Packt Publishing, 2018. ISBN: 1783987448.

[10] Louis G Birta and Gilbert Arbez. *Modelling and Simulation: Exploring Dynamic System Behaviour*. eng. London: Springer London, Limited, 2007. ISBN: 184996629X.

[11] William Bolton. *Control Systems*. eng. Oxford: Newnes, 2002. ISBN: 0-7506-5461-9.

[12] J Borenstein et al. "Mobile robot positioning: Sensors and techniques". eng. In: *Journal of robotic systems* 14.4 (1997), pp. 231–249. ISSN: 0741-2223.

[13] R. Bostelman et al. "A-UGV capabilities". In: *2019 Third IEEE International Conference on Robotic Computing (IRC)*. 2019, pp. 1–7. DOI: 10.1109/IRC.2019.00130.

[14] L Bruzzone and G Quaglia. "Review article: locomotion systems for ground mobile robots in unstructured environments". eng. In: *Mechanical sciences (Göttingen)* 3.2 (2012), pp. 49–62. ISSN: 2191-916X.

[15] *BUS*. https://it.mathworks.com/help/simulink/slref/simulink-bus-signals.html.

[16] Elisa Capello and Giorgio Guglieri. *Development of a Ground Test Concept Based on Multi-Rotors for In-Flight RVD Experimentation*. eng. 2015.

[17] J. Cerkala and A. Jadlovsk'a. "NONHOLONOMIC MOBILE ROBOT WITH DIFFERENTIAL CHASSIS MATHEMATICAL MODELLING AND IMPLEMENTATION IN SIMULINK WITH FRICTION IN DYNAMICS". In: 2016.

[18] Badong Chen et al. *System Parameter Identification: Information Criteria and Algorithms*. eng. Saint Louis: Elsevier, 2013. ISBN: 012404574X.

[19] *Curiosity rover*. https://en.wikipedia.org/wiki/Curiosity_(rover)#/media/File:PIA19808-MarsCuriosityRover-AeolisMons-BuckskinRock-20150805.jpg.

[20] Tehmoor Dar and Raul Longoria. "Estimating Traction Coefficients of Friction for Small-Scale Robotic Tracked Vehicles". In: vol. 2. Jan. 2010. DOI: 10.1115/DSCC2010-4228.

[21] *Devastator*. https://www.dfrobot.com/product-1477.html.

[22] Rajamani Doraiswami. *Identification of Physical Systems: Applications to Condition Monitoring, Fault Diagnosis, Soft Sensor and Controller Design*. eng. John Wiley & Sons, Inc, 2014. ISBN: 1-119-99012-2.

[23] Olav Egeland and Jan Gravdahl. *Modeling and Simulation for Automatic Control*. Jan. 2002.

[24] D. Endo et al. "Path following control for tracked vehicles based on slip-compensating odometry". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007, pp. 2871–2876. DOI: 10.1109/IROS.2007.4399228.

[25] Antonella Ferrara and Matteo Rubagotti. "Second-order sliding-mode control of a mobile robot based on a harmonic potential field". In: *Control Theory & Applications, IET* 2 (Oct. 2008), pp. 807 –818. DOI: 10.1049/iet-cta:20070424.

[26] A. Finn and S. Scheding. *Developments and Challenges for Autonomous Unmanned Vehicles: A Compendium*. Intelligent Systems Reference Library. Springer Berlin Heidelberg, 2010. ISBN: 9783642107047. URL: https://books.google.it/books?id=TbHo4n4YrZ4C.

[27] *Foster Miller TALON SWORDS*. https://en.wikipedia.org/wiki/Foster-Miller_TALON#/media/File:Foster-Miller_TALON_SWORDS.jpg.

[28] *FPE*. https://it.mathworks.com/help/ident/ref/idmodel.fpe.html.

[29] Yang Gao. *Contemporary Planetary Robotics: An Approach Toward Autonomous Systems*. eng. Weinheim: John Wiley & Sons, Incorporated, 2016. ISBN: 9783527413256.

[30] *Gazebo*. http://gazebosim.org/tutorials?cat=install.

[31] *Gazebo interface*. http://gazebosim.org/tutorials?tut=build_world&cat=build_world.

[32] *Get started with ROS/Simulink*. https://it.mathworks.com/help/ros/ug/get-started-with-ros-in-simulink.html.

[33] *Gladiator Tactical UGV*. https://en.wikipedia.org/wiki/Gladiator_Tactical_Unmanned_Ground_Vehicle#/media/ile:Gladiator_240G.jpg.

[34] Brendan Gogarty and Isabel Robinson. *Unmanned Vehicles: A (Rebooted) History, Background and Current State of the Art*. 2011. URL: https://search.informit.org/doi/10.3316/ielapa.034249957187447.

[35] A. Granja et al. "Improving navigation of an Autonomous Mobile Robot using System Identification and Control". In: *2007 IEEE International Symposium on Industrial Electronics*. 2007, pp. 2948–2953. DOI: 10.1109/ISIE.2007.4375083.

[36] J. Guldner et al. "Tracking gradients of artificial potential fields with non-holonomic mobile robots". In: *Proceedings of 1995 American Control Conference - ACC'95*. Vol. 4. 1995, 2803–2804 vol.4. DOI: 10.1109/ACC.1995.532361.

[37] V. Gupta et al. "Three-stage computed-torque controller for trajectory tracking in non-holonomic wheeled mobile robot". In: *2018 IEEE 15th International Workshop on Advanced Motion Control (AMC)*. 2018, pp. 144–149. DOI: 10.1109/AMC.2019.8371077.

[38] Rached Hatab. "Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework". In: *Advances in Robotics & Automation* 02 (Jan. 2013). DOI: 10.4172/2168-9695.1000107.

[39] Victor Manuel Hernández-Guzmán and Ramón Silva-Ortigoza. *Automatic Control with Experiments*. eng. Advanced Textbooks in Control and Signal Processing. Cham: Springer International Publishing. ISBN: 9783319758039.

[40] *HTML*. https://en.wikipedia.org/wiki/HTML.

[41] Hui-Min Huang et al. "Autonomy levels for unmanned systems (ALFUS) framework: an update". eng. In: *Proceedings of SPIE*. Vol. 5804. 1. Bellingham WA: SPIE, 2005, pp. 439–448. ISBN: 0819457892.

[42] R. Iglesias et al. "Task identification and characterisation in mobile robotics through non-linear modelling". In: *Robotics and Autonomous Systems* 55.4 (2007), pp. 267–275. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2006.11.007.

[43] Roberto Iglesias Rodriguez et al. "Training and Analysis of Mobile Robot Behaviour Through System Identification". In: vol. 4177. Jan. 2006, pp. 470–479. ISBN: 978-3-540-45914-9. DOI: 10.1007/11881216_49.

[44] Lentin Joseph. *Learning robotics using Python: design, simulate, program, and prototype an interactive autonomous mobile robot from scratch with the help of Python, ROS, and Open-CV!* eng. 1st ed. Community experience distilled. Birmingham, UK: PACKT Publishing, 2015. ISBN: 9781783287536.

[45] Eugene Kagan, Nir Shvalb, and Irad Ben-Gal. *Autonomous Mobile Robots and Multi-Robot Systems: Motion-Planning, Communication, and Swarming*. eng. Newark: John Wiley & Sons, Incorporated, 2019. ISBN: 9781119212867.

[46] Jagannathan Kanniah, M Ercan, and Carlos Calderon. *Practical Robot Design*. eng. 1st ed. CRC Press, 2013. ISBN: 9781439810330.

[47] Sang-Hoon Kim. *Electric motor control : DC, AC, and BLDC motors / Sang-Hoon Kim*. eng. Amsterdam: Elsevier, 2017. ISBN: 978-0-12-812138-2.

[48] Gregor Klancar et al. *Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems*. eng. Oxford: Elsevier Science & Technology, 2017. ISBN: 9780128042045.

[49] Andrii Kudriashov et al. *SLAM Techniques Application for Mobile Robot in Rough Terrain*. eng. Vol. 87. Mechanisms and Machine Science. Cham: Springer International Publishing. ISBN: 9783030489809.

[50] James Larminie and John Lowry. *Electric vehicle technology explained, second edition*. eng. 2nd ed. Chichester, West Sussex, U.K: Wiley, 2012. ISBN: 111994273X.

[51] Lauer. *Hybrid System Identification*. eng. Springer International Publishing, 2019. ISBN: 3-030-00192-X.

[52] Guodong Li and Xiaolong Li. "Research on trajectory tracking of crawler robot based on sliding mode control". In: May 2014, pp. 518–523. ISBN: 978-1-4799-3708-0. DOI: 10.1109/CCDC.2014.6852203.

[53] Lennart Ljung. "System Identification". In: *Wiley Encyclopedia of Electrical and Electronics Engineering*. American Cancer Society, 2017, pp. 1–19. ISBN: 9780471346081. DOI: https://doi.org/10.1002/047134608X.W1046.pub2. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/047134608X.W1046.pub2. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/047134608X.W1046.pub2.

[54] Lennart Ljung. "System Identification Toolbox for use with MATLAB". In: 21 (Jan. 2011).

[55] *Lunokhod_1 rover*. https://en.wikipedia.org/wiki/Lunokhod_1#/media/File:Soviet_moonrover.JPG.

[56] Kevin Lynch, Nicholas Marchuk, and Matthew Elwin. *Embedded Computing and Mechatronics with the PIC32 Microcontroller*. eng. Oxford: Elsevier Science & Technology, 2015. ISBN: 0124201652.

[57] Ma. *Kalman Filtering and Information Fusion*. eng. Springer Singapore, 2020. ISBN: 981-15-0805-4.

[58] Konrad Majkut, Mariusz Giergiel, and Piotr Kohut. "Crawler robot kinematics modeling by using a two-wheeled approach". In: *Journal of Mechanical Science and Technology* 31 (Feb. 2017), pp. 893–901. DOI: 10.1007/s12206-017-0142-0.

[59] A Mandow et al. "Experimental kinematics for wheeled skid-steer mobile robots". eng. In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 1222–1227. ISBN: 9781424409112.

[60] *Mars rover generations*. https://mars.nasa.gov/resources/3792/three-generations-of-rovers-in-mars-yard/.

[61] J. L Martínez et al. "Approximating Kinematics for Tracked Mobile Robots". eng. In: *The International journal of robotics research* 24.10 (2005), pp. 867–878. ISSN: 0278-3649.

[62] Jorge Martínez et al. “Approximating Kinematics for Tracked Mobile Robots”. In: *The International Journal of Robotics Research* 24 (Oct. 2005), pp. 867–878. DOI: 10.1177/0278364905058239.

[63] Jorge Martínez et al. “Approximating Kinematics for Tracked Mobile Robots”. In: *The International Journal of Robotics Research* 24 (Oct. 2005), pp. 867–878. DOI: 10.1177/0278364905058239.

[64] Rubén Molina Llorente. *Practical Control of Electric Machines: Model-Based Design and Simulation*. eng. Advances in Industrial Control. Cham: Springer International Publishing. ISBN: 9783030347574.

[65] Rubén Molina Llorente. *Practical Control of Electric Machines: Model-Based Design and Simulation*. eng. Advances in Industrial Control. Cham: Springer International Publishing AG, 2020. ISBN: 3030347575.

[66] Sabudin elia nadira, Rosli Omar, and Che Ku Nor Hailma. “Potential field methods and their inherent approaches for path planning”. In: 11 (Jan. 2016), pp. 10801–10805.

[67] Sabudin elia nadira, Rosli Omar, and Che Ku Nor Hailma. “Potential field methods and their inherent approaches for path planning”. In: 11 (Jan. 2016), pp. 10801–10805.

[68] K. Nagatani, D. Endo, and K. Yoshida. “Improvement of the Odometry Accuracy of a Crawler Vehicle with Consideration of Slippage”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, pp. 2752–2757. DOI: 10.1109/ROBOT.2007.363881.

[69] Nils J Nilsson. *Shakey the Robot*. eng. 1984.

[70] J.M. O’Kane. *A Gentle Introduction to ROS*. Jason M. O’Kane, 2014. ISBN: 9781492143239. URL: https://books.google.it/books?id=OzFHngEACAAJ.

[71] Committee on Autonomous Vehicles in Support of Naval Operations et al. *Autonomous Vehicles in Support of Naval Operations*. eng. Washington, D.C: National Academies Press, 2005. ISBN: 9780309096768.

[72] *Otus*. https://www.rcbenchmark.com/blogs/other/otus-tracker-documentation-and-software-download-deprecated.

[73] *Otus sensor*. https://www.rcbenchmark.com/pages/otus-tracker.

[74] Dwi Pebrianti et al. “Motion Tracker Based Wheeled Mobile Robot System Identification and Controller Design”. In: *Intelligent Manufacturing {&} Mechatronics*. Ed. by Mohd Hasnun Arif Hassan. Singapore: Springer Singapore, 2018, pp. 241–258. ISBN: 978-981-10-8788-2.

[75] *Probability density function*. https://en.wikipedia.org/wiki/Probability_density_function.

[76] B. Raafiu and P. A. Darwito. “Identification of Four Wheel Mobile Robot based on Parametric Modelling”. In: *2018 International Seminar on Intelligent Technology and Its Applications (ISITIA)*. 2018, pp. 397–401. DOI: 10.1109/ISITIA.2018.8710761.

[77] P. Raja and S. Pugazhenthi. "Optimal path planning of mobile robots: A review". In: *International Journal of the Physical Sciences* 7 (Feb. 2012). DOI: 10.5897/IJPS11.1745.

[78] Jitendra R Raol. *Mobile Intelligent Autonomous Systems*. eng. CRC Press, 2016. ISBN: 1-4398-6300-8.

[79] Jitendra R Raol and Ramakalyan Ayyagari. *Control Systems: Classical, Modern, and AI-Based Approaches*. eng. 1st ed. Milton: CRC Press, 2020. ISBN: 0815346301.

[80] Jitendra R Raol and Ajith K Gopal. *Mobile Intelligent Autonomous Systems*. eng. 1st ed. Baton Rouge: CRC Press, 2013. ISBN: 1439863008.

[81] Maria Ribeiro and Isabel Ribeiro. *Kalman and Extended Kalman Filters: Concept, Derivation and Properties*. Apr. 2004.

[82] *Robot*. https://dictionary.cambridge.org/it/dizionario/inglese/robot.

[83] *ROS*. https://en.wikipedia.org/wiki/Robot_Operating_System#OSRF_and_Open_Robotics_(2013-present).

[84] *ROS*. https://www.ros.org/about-ros/.

[85] *ROS introduction*. http://wiki.ros.org/ROS/Introduction.

[86] *RQ-16 T-Hawk*. https://en.wikipedia.org/wiki/Honeywell_RQ-16_T-Hawk#/media/File:MicroAirVehicle.jpg.

[87] *RQ-4 Global Hawk*. https://en.wikipedia.org/wiki/Northrop_Grumman_RQ-4_Global_Hawk#/media/File:Global_Hawk_1.jpg.

[88] Francisco Rubio, Francisco Valero, and Carlos Llopis-Albert. "A review of mobile robots: Concepts, methods, theoretical framework, and applications". eng. In: *International Journal of Advanced Robotic Systems* 16.2 (2019), p. 172988141983959. ISSN: 1729-8814.

[89] *Scorpion Robot*. https://robotik.dfki-bremen.de/en/research/robot-systems/scorpion.html.

[90] Aravind Seeni, Bernd Schäfer, and Gerd Hirzinger. "Robot Mobility Systems for Planetary Surface Exploration – State-of-the-Art and Future Outlook: A Literature Survey". In: Jan. 2010, pp. 189–208. ISBN: 9 789537 619961. DOI: 10.5772/6930.

[91] *Shakey the Robot*. https://www.sri.com/hoi/shakey-the-robot/.

[92] Thomas B Sheridan and William L Verplank. *Human and Computer Control of Undersea Teleoperators*. eng. 1978.

[93] A.A. Siddiqi and United States. NASA History Program Office. *Beyond Earth: A Chronicle of Deep Space Exploration, 1958-2016*. NASA SP. National Aeronautics and Space Administration, Office of Communications, NASA History Division, 2018. ISBN: 9781626830431. URL: https://books.google.it/books?id=tSzeswEACAAJ.

[94] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. eng. 2nd ed. Intelligent Robotics and Autonomous Agents series. Cambridge: MIT Press, 2011. ISBN: 0262015358.

[95] A. Simpkins. "System Identification: Theory for the User, 2nd Edition (Ljung, L.; 1999) [On the Shelf]". In: *IEEE Robotics Automation Magazine* 19.2 (2012), pp. 95–96. DOI: 10.1109/MRA.2012.2192817.

[96] Jonas Sjöberg et al. "Nonlinear black-box modeling in system identification: a unified overview". In: *Automatica* 31.12 (1995). Trends in System Identification, pp. 1691–1724. ISSN: 0005-1098. DOI: https://doi.org/10.1016/0005-1098(95)00120-8. URL: https://www.sciencedirect.com/science/article/pii/0005109895001208.

[97] *Sojourner rover*. https://mars.nasa.gov/MPF/roverpwr/power.html.

[98] Yong-Duan Song. *Control of Nonlinear Systems Via PI, PD and PID : Stability and Performance*. eng. CRC Press LLC. ISBN: 1-138-31764-0.

[99] Mark W Spong. *Robot modeling and control / Mark W. Spong, Seth Hutchinson, M. Vidyasagar*. eng. Hoboken: Wiley, 2006. ISBN: 0-471-64990-2.

[100] *Stanford Cart*. https://www.computerhistory.org/revolution/artificial-intelligence-robotics/13/293/1277.

[101] Starr. *Introduction to Applied Digital Controls*. eng. Springer International Publishing, 2020. ISBN: 3-030-42809-5.

[102] Natalia Strawa et al. "On-Line Learning and Updating Unmanned Tracked Vehicle Dynamics". eng. In: *Electronics (Basel)* 10.187 (2021), p. 187. ISSN: 2079-9292.

[103] Karl J Ström. *PID controllers / Karl J. ström, Tore Hägglund*. eng. Research Triangle Park: Instrument Society of America, 1995.

[104] A.K. Tangirala. *Principles of System Identification: Theory and Practice*. CRC Press, 2018. ISBN: 9781439896020. URL: https://books.google.it/books?id=aUHOBQAAQBAJ.

[105] Committee on Army Unmanned Ground Vehicle Technology et al. *Technology Development for Army Unmanned Ground Vehicles*. eng. Washington, D.C: National Academies Press, 2002. ISBN: 9780309086202.

[106] S. G Tzafestas and Spyros G Tzafestas. *Introduction to Mobile Robot Control*. eng. Elsevier Insights. Amsterdam: Elsevier, 2014. ISBN: 0-12-417049-8.

[107] A Y Umeda, S W Martin, and J O Merritt. *Remote Vision Systems for Teleoperated Ground Vehicles*. eng. 1991.

[108] *URDF*. http://wiki.ros.org/urdf.

[109] *User Guide*. https://it.mathworks.com/products/ros.html.

[110] *VCS*. https://en.wikipedia.org/wiki/Version_control.

[111] Vyas. *Electro-Hydraulic Actuation Systems*. eng. Springer Singapore, 2019. ISBN: 981-13-2546-4.

[112] Wang. *PID Control System Design and Automatic Tuning using MATLAB/Simulink*. eng. John Wiley & Sons, Inc, 2020. ISBN: 1-119-46934-1.

[113] *White Noise*. https://en.wikipedia.org/wiki/White_noise.

[114] J Y Wong and C F Chiang. "A general theory for skid steering of tracked vehicles on firm ground". In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 215.3 (2001), pp. 343–355. DOI: 10.1243/0954407011525683. eprint: https://doi.org/10.1243/0954407011525683. URL: https://doi.org/10.1243/0954407011525683.

[115] Jo Yung Wong. *Theory of ground vehicles / J.Y. Wong*. eng. 2nd ed. New York: Wiley, 1993.

[116] *XML*. https://en.wikipedia.org/wiki/XML.