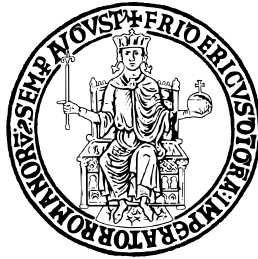


UNIVERSITY OF NAPLES FEDERICO II



SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

INDUSTRIAL ENGINEERING DEPARTMENT

DOCTORAL PROGRAM IN INDUSTRIAL ENGINEERING XXXVII CYCLE

STUDY AND DEVELOPMENT OF A SIX-WHEELED ROBOT FOR FIELD APPLICATIONS

Supervisors

Prof. Eng. Vincenzo NIOLA

Prof. Eng. Sergio SAVINO

Prof. Eng. Chiara COSENZA

PhD Candidate

Armando NICOLELLA

DR996136

Academic Year 2023–2024

Abstract

This PhD thesis focuses on the development and validation of a six-wheeled rover, designed within the AVELICA project, funded by the European REACT-EU programme, for environmental applications such as waste collection in non-cohesive terrain, such as beaches and natural environments. The rover integrates a rocker-bogie suspension architecture and a robotic arm with computer vision systems designed to operate in complex and dynamic scenarios. The approach adopted combines multibody modelling with calibration techniques, including hand-eye calibration, to improve the accuracy and effectiveness of the robotic-arm camera system.

The work involved several stages that contributed to the design and optimization of the system. One of the main achievements was the combination of modelling, simulation and practical experimentation. The development of the rover was divided into two different solutions: a Small Rover prototype to validate the numerical model, refine the design parameters and carry out initial tests, and a Full-Scale prototype to verify the operational performance at full scale. The multibody models, designed as digital twins, were used not only to simulate the system dynamics and analyse the load distribution, but also to lead the experimental test phases in the case of the Small Rover, allowing a significant reduction in development time.

Particular attention has been given to the integration of vision systems thanks to a hand-eye calibration, which are essential to enable the rover to autonomously perceive and interact with its environment. The rover is also equipped with a robotic arm to modify the attitude for aiding crossing obstacles, improving the stability and adaptability of the system. Finally, autonomous navigation applications have been explored for both single rovers and multi-agent systems, providing flexibility in environmental applications.

The methodological approach adopted in this thesis ensures that each aspect of the research directly contributes to the overarching goal of developing a fully functional robotic platform for real-world deployment in environmental contexts. The results demonstrate the effectiveness of the system in combining sustainability, technological innovation and environmental protection, in line with the objectives of the European Green Deal and national sustainable waste management strategies.

Contents

1	Introduction, Background and State of Art	1
1.1	Overview of the Research Context	2
1.2	Autonomous VEHicles for LItter Capture	3
1.2.1	Motivation and Objectives	3
1.2.2	Environmental and Technological Goals	4
1.3	Mobile Robots and Rovers: Classifications and Features	5
1.3.1	Mobility Systems: Wheels, Tracks, and Legs	5
1.3.1.1	Tracked Robot	7
1.3.1.2	Legged Robot	7
1.3.1.3	Wheeled Robot and Rover	8
1.3.2	Suspension Systems: Rocker-Bogie and Alternatives	9
1.3.3	Steering Mechanisms	12
1.3.3.1	Explicit and Skid Steering	12
1.3.3.2	Wheels	14
1.4	Perception and Computer Vision	15
1.4.1	Historical Evolution and Foundations	16
1.4.2	Fundamentals of Computer Vision	16
1.4.2.1	Perspective Model	16
1.4.2.2	Extrinsics and Intrinsics Matrix	19
1.4.2.3	Lens Distorsion	21
1.4.2.4	Stereo Vision	21
1.4.3	Techniques in Computer Vision	22
1.4.4	Applications in Robotics	22
1.4.5	Integration in Other Systems	23
1.4.6	Challenges and Future Directions	23
1.5	Mobile Manipulator Robots	25
1.5.1	Robotic Arm: Characteristic and Applications	25
1.5.1.1	Integration on Mobile Robots	25
1.5.2	Vision Systems on Mobile Robots	26
1.5.3	Challenges in Integration	26
1.6	Controllers and Autonomous Navigation Systems	27
1.6.1	Controllers	27

1.6.1.1	PID	27
1.6.1.2	Sliding Mode Control	30
1.6.2	Autonomous Navigation Systems	32
1.6.2.1	Foundations and Challenges in Robotic Navigation	34
1.6.2.2	Principles of Single-Agent Navigation	36
1.6.2.3	Strategies for Multi-Agent Navigation	39
1.6.2.4	Applications of Mobile Robots	41
1.7	Modeling and Simulation of Mobile Robots	45
1.7.1	Computer-Aided Design (CAD)	45
1.7.1.1	CAD System	45
1.7.1.2	CAD Model	46
1.7.2	Multibody Dynamics	47
1.7.3	Main Physics of Multibody Software	47
1.7.3.1	Contact	48
1.7.3.2	DC Motor	50
1.8	Communication and Signal Processing	55
1.8.1	Basic Concepts in Signal Theory	55
1.8.1.1	FFT Theory	57
1.8.2	Communication Protocol Definition	60
1.8.2.1	USB	60
1.8.2.2	I ² C	61
1.8.2.3	CAN Protocol	63
2	Integrated Design, Modelling, and Simulation of Rover Systems	65
2.1	Introduction	66
2.2	Rocker-Bogie Suspension Analysis	67
2.2.1	Fundamental Principles of the Rocker-Bogie Suspension	67
2.2.2	Obstacle Crossing: Mechanics and Dynamics	68
2.2.3	Advantages and Disadvantages of the Rocker-Bogie System	69
2.2.4	Kinematics Analysis of Suspension	70
2.2.5	Load Distribution Analysis	74
2.2.5.1	Load Distribution Without Obstacles	74
2.2.5.2	Obstacle-Induced Load Changes	75
2.2.5.3	Static Stability Analysis	80
2.2.6	Conclusions	82
2.3	Small Rover Prototype	83
2.3.1	Multibody Prototype	83
2.3.1.1	Rover Macroblock Development	85
2.3.1.2	Experimental Characterization of DC Motors	91
2.3.1.3	Contact Model: Ground and Obstacle	97
2.3.1.4	Conclusions	104
2.3.2	Rover Movement Analysis and Simulations	105

2.3.2.1	Rectilinear Motion Simulations	105
2.3.2.2	Obstacle Crossing Simulations	106
2.3.2.3	Conclusions	109
2.3.3	Vibration Analysis in Straight-Forward and Steering Configurations	110
2.3.3.1	Single-Wheel Analysis	110
2.3.3.2	Straight-Forward Configuration	112
2.3.3.3	Steering Configuration	115
2.3.3.4	Conclusions	118
2.3.4	Integration of Robotic Arm for Obstacle Crossing	119
2.3.4.1	Analysis of Integration Feasibility	119
2.3.4.2	Simulation Processes	129
2.3.4.3	Prototype Simulations	131
2.3.4.4	Conclusions	141
2.4	Full-Scale Rover Prototype	142
2.4.1	First Prototype	142
2.4.1.1	CAD Model of the Rover	142
2.4.1.2	Multibody Model: Development and Optimization . . .	152
2.4.1.3	Simulation Results for Obstacle Crossing	173
2.4.1.4	Irregular Surface Simulations	177
2.4.1.5	Energy Consumption Estimation	179
2.4.1.6	Study of Alternative Configurations	183
2.4.2	Second Prototype Development and Analysis	203
2.4.2.1	Evolution of Rover Design	203
2.4.2.2	Simulation Results	232
2.4.3	Conclusions	239
3	Development and Experimental Validation of Rover Prototypes	241
3.1	Introduction	242
3.2	Small Rover Prototype	243
3.2.1	Movement Testing of the Rover	243
3.2.1.1	Remote Management and Communication Logic . . .	244
3.2.1.2	Straight-Line Motion Test	245
3.2.1.3	Steering Performance Test	251
3.2.1.4	Obstacle Crossing Test	253
3.2.1.5	Conclusions	255
3.2.2	Vibration Spectral Analysis	257
3.2.2.1	Straight-Line Vibration Analysis	259
3.2.2.2	Obstacle Crossing Vibration Analysis	264
3.2.2.3	Steering Configuration Vibration Analysis	269
3.2.2.4	Conclusions	274
3.2.3	Robotic Arm Integration for Enhanced Obstacle Overcoming . .	275
3.2.3.1	Command Flow Optimization	276
3.2.3.2	Experimental Tests and Results	277

3.2.3.3	Conclusions	286
3.3	Full-Scale Rover Prototype	287
3.3.1	First Prototyping	288
3.3.1.1	3D Printed Components	288
3.3.1.2	Characteristics of PLA Material	291
3.3.1.3	Assembly Procedures	291
3.3.2	Second Prototyping	295
3.3.2.1	Advanced Prototyping Techniques	295
3.3.2.2	ABS Material Characteristics	296
3.3.2.3	3D Printing Techniques for ABS	297
3.3.2.4	Prototyping of Joints and Connections	298
3.3.2.5	Final Assembly Steps	299
3.3.3	Conclusions	303
4	Calibration Technique for Vision-Based Systems on the Rover	305
4.1	Introduction	306
4.2	Hand-EYE Calibration	307
4.3	Experimental Setup	311
4.3.1	Experimental Configuration	312
4.3.1.1	Reference Frame Definitions	312
4.3.1.2	Workspace Design and Setup	312
4.3.2	Robot Configurations for Testing	313
4.4	Proposed Calibration Algorithm	314
4.4.1	Mathematical Framework	314
4.4.2	Data Acquisition Strategies	314
4.4.3	Image Processing and Marker Detection	317
4.4.4	3D Point Reconstruction Techniques	318
4.4.5	Plane Fitting for Calibration Accuracy	319
4.4.5.1	Least Squares Plane Fitting Method	319
4.4.6	Projection and Registration Techniques	321
4.4.7	Optimization and Matrix Computation	323
4.4.8	Validation and Error Analysis	323
4.5	Conclusions	325
5	Applications of Robotic Navigation	327
5.1	Introduction	328
5.2	Vision-Based Algorithms for Motion Perception	329
5.2.1	Techniques and Methodologies	329
5.2.1.1	Light-Based Segmentation for Mobile Robot Tracking	330
5.2.1.2	Visual Simultaneous Localization and Mapping (vSLAM)	334
5.2.1.3	Camera Extrinsic Calibration	342
5.2.2	Experimental Results	345
5.2.2.1	External Perspective Results in Trajectory Tracking	345

	5.2.2.2	Internal Perspective Results in Trajectory Tracking . . .	348
	5.2.3	Conclusions	351
5.3		LiDAR-Based SLAM for Single and Multi-Agent Systems	352
	5.3.1	System Architecture	353
	5.3.1.1	Integration with Robotic Operating System (ROS) . . .	353
	5.3.2	Algorithm Workflow	354
	5.3.2.1	Point Cloud Mapping with LiDAR	356
	5.3.2.2	Direct LiDAR Odometry Technique	357
	5.3.2.3	GMapping for Environmental Representation	358
	5.3.2.4	Adaptive Monte Carlo Localization (AMCL)	360
	5.3.2.5	Move Base for Path Planning	362
	5.3.3	Validation Through Experiments	363
	5.3.3.1	First Experiment	363
	5.3.3.2	Second Experiment	363
	5.3.4	Configurations for Multi-Agent Collaboration	363
	5.3.5	Conclusions	366
5.4		Exploration with Multi-Agent and Collaborative Robotic Systems	367
	5.4.1	Turtlebot 4 Setups and Simulation Environment	367
	5.4.1.1	TurtleBot 4 Setup and Implementation	367
	5.4.1.2	Network Configuration	367
	5.4.1.3	Collaborative Framework for Multiple Robots	369
	5.4.1.4	Using Ignition Gazebo and Rviz2 for Visualization . . .	373
	5.4.1.5	Teleoperation for a Single Turtlebot4	374
	5.4.1.6	Leader-Follower Configurations in Multi-Agent Systems	374
	5.4.1.7	Aruco Marker-Based Localization and Navigation . . .	376
	5.4.1.8	Pose Estimation Techniques for Robotic Systems . . .	378
	5.4.1.9	Camera Frame-to-Base Link Calibration	378
	5.4.1.10	Collaborative Logic Development in Multi-Agent Systems	379
	5.4.1.11	Control Logic for Multi-Robot Coordination	380
	5.4.2	Test and Discussion	382
	5.4.2.1	Chasing Test	382
	5.4.2.2	Law of Motion Test	388
	5.4.2.3	Obstacle Avoidance Test	399
	5.4.3	Conclusions	402
5.5		Robust Control Systems for Accurate Trajectory Tracking	403
	5.5.1	Hardware Overview of Robotic Systems	403
	5.5.2	Principles of H_∞ Control Theory	404
	5.5.3	Methodologies for Controller Design	405
	5.5.3.1	Off-Center Control	406
	5.5.4	H_∞ Controller Design	408
	5.5.5	Validation of H_∞ Control Design	413
	5.5.6	Conclusions	416

6 Conclusion	417
A Hardware Components and Systems Integration	419

–1–

Introduction, Background and State of Art

CONTENTS: **1.1 Overview of the Research Context.** **1.2 Autonomous Vehicles for Litter Capture.** 1.2.1 Motivation and Objectives – 1.2.2 Environmental and Technological Goals. **1.3 Mobile Robots and Rovers: Classifications and Features.** 1.3.1 Mobility Systems: Wheels, Tracks, and Legs – 1.3.2 Suspension Systems: Rocker-Bogie and Alternatives – 1.3.3 Steering Mechanisms. **1.4 Perception and Computer Vision.** 1.4.1 Historical Evolution and Foundations – 1.4.2 Fundamentals of Computer Vision – 1.4.3 Techniques in Computer Vision – 1.4.4 Applications in Robotics – 1.4.5 Integration in Other Systems – 1.4.6 Challenges and Future Directions. **1.5 Mobile Manipulator Robots.** 1.5.1 Robotic Arm: Characteristic and Applications – 1.5.2 Vision Systems on Mobile Robots – 1.5.3 Challenges in Integration. **1.6 Controllers and Autonomous Navigation Systems.** 1.6.1 Controllers – 1.6.2 Autonomous Navigation Systems. **1.7 Modeling and Simulation of Mobile Robots.** 1.7.1 Computer-Aided Design (CAD) – 1.7.2 Multibody Dynamics – 1.7.3 Main Physics of Multibody Software. **1.8 Communication and Signal Processing.** 1.8.1 Basic Concepts in Signal Theory – 1.8.2 Communication Protocol Definition.

1.1 Overview of the Research Context

This thesis is the result of research carried out at the Department of Industrial Engineering (DII) of the University of Naples Federico II, which has developed a mobile robotic system for environmental applications. The entire project revolves around the realisation of a highly specialised autonomous rover, designed to operate in complex contexts such as waste collection on non-cohesive terrain and environmental monitoring. To ensure the success of such a system, a multidisciplinary approach was required, integrating dynamic modelling, computer vision, autonomous navigation and robotic manipulation, aspects that will be covered in the various chapters of this thesis. One of the features of this study is the design of the six-wheel rover equipped with a passive rocker-bogie suspension system, a configuration chosen to maximise stability in difficult terrain. This aspect is crucial to the operation of the rover, as it distributes the load evenly over all the wheels and ensures constant traction even in the presence of obstacles. An objective of the thesis was the development of a multibody model, validated by experimental tests. The creation of a realistic model was necessary to optimise the design and reduce the development time of the rover, allowing the analysis of the load distribution on the wheels, the interaction with the ground and the dynamic behaviour of the whole system. Validation was carried out through an experimental phase on a small rover prototype, which allowed the design solutions to be tested before the realisation of the full-scale rover prototype, which represents the final operational system. One aspect of this work was the integration of machine vision to control the robotic arm. To ensure precise and effective operation of the manipulator system, accurate calibration between the camera mounted on the rover and the manipulator was required. This hand-EYE calibration was a key element in ensuring perfect alignment between visual perception and mechanical action, improving the ability of the rover to interact autonomously with its environment. Finally, another focus of the research was the development of advanced autonomous navigation strategies, both for a single rover and for multi-agent systems. This research was essential to ensure that the rover was not just an isolated entity, but could operate in cooperative scenarios with other robots, increasing the overall efficiency of operations. The techniques developed are directly applicable to real-world scenarios, making the system ready for large-scale operations in complex and dynamic environments. In the final phase of the Ph.D. period, the topic of robotic navigation was explored in both single-agent and multi-agent configurations. This study was made possible through close collaboration with the University of Udine - Polytechnic Department of Engineering and Architecture and the Universitat Politècnica de València - Department of Systems Engineering and Automation. This overview of the research context shows how each aspect of this thesis has been developed with a clear objective: to achieve an integrated robotic system in which navigation, visual perception, manipulation and mechanical dynamics work in synergy to maximise the performance and effectiveness of the rover in environmental applications.

1.2 Autonomous VEHicles for Litter Capture

The project was born out of a need to automate the processes of collecting waste from areas of daily use, such as marine shorelines. The project is in line with the GREEN themes of environmental protection and conservation, with particular reference to the issues of preserving ecosystems, biodiversity and reducing the impact of climate change. Similar to what is happening in industry, there is a growing interest in environmental protection in the use of automated mechanical and robotic systems. In several GREEN applications, especially in waste collection, there is investment in research and development of robots that can assist or even replace humans in performing highly repetitive actions. A further step forward is the possibility of improving environmental protection measurements by equipping the vehicle with sensors that can perceive the environment, for example through machine vision systems. Robotics and automation can also contribute to a series of environmental protection measures known as green technologies. Technological innovation can be improved through control and monitoring actions, having a sustainable management of the environment and in particular the marine ecosystem. In this context, the AVELICA project involves the development, design and prototyping of an autonomous guided vehicle (rover) capable of monitoring the environment and performing various field operations when necessary. The vehicle will be equipped with sensors that allow it to perceive the working environment, a robotic arm capable of performing collection operations, and, in the future, a communication system with aircraft (drones) for trajectory planning. The AVELICA project is in line with the objectives of the REACT-EU program. In particular, the project includes the national theme of the SNSI called "Intelligent and Sustainable Industry, Energy and Environment". In addition, the project is in line with the NRP for "Food, Bioeconomy, Natural Resources, Agriculture, Environment".

1.2.1 Motivation and Objectives

In recent years, the green technology sector has been undergoing major changes to address the problems of pollution, climate change and environmental protection. The development of green technologies is one of the priorities of the National Research Plan in the area of "Food, Bioeconomy, Natural Resources, Agriculture, Environment". In this context, the proposed project aims to provide the academic Department involved and the partner company (Green Tech Solution S.r.l) with the theoretical and technical tools necessary to make the proposed innovative ideas prospectively reliable, robust and industrializable. The transversal nature of the AVELICA project brings together different themes of industrial engineering (such as robotics and mechatronics) for the development of a system perfectly framed within the scope of green technologies and environmental protection. These themes are widely present in the National Strategy of Intelligent Specialization, with particular reference to the field of "Intelligent and Sustainable Industry, Energy and Environment". The system produced as a result of the AVELICA project could be able to enter into daily environmental protection actions, contributing to the preservation of the ecosystem by going to contain unwanted waste

deposits in naturalistic, daily used or highly frequented areas. Moreover, it is interesting to note that the creation of such a system is not only usable for the collection of waste along the coastline, but can also be used for the cleaning of various environments and, with appropriate minor modifications, even for other purposes and applications in other areas outside the green sphere (e.g. monitoring and inspection of hazardous areas, etc.).

1.2.2 Environmental and Technological Goals

The main objective of the proposal is to study, design and develop an innovative mechatronic system. The AVELICA project aims to provide a concrete solution for environmental protection through the combined use of skills and technologies from different fields, including engineering, computer science, mechatronics and robotics. One of the goals is to reduce waste in open areas and thus prevent soil contamination. This approach is fully in line with the European Green Deal roadmap and contributes to the implementation of circular economy policies and protocols, biodiversity preservation and pollution reduction.

The proposed solution falls within the SNSI national thematic area "Smart and Sustainable Industry, Energy and Environment", with a focus on green technologies. It also falls within the scope of the NRP cluster "Food, Bioeconomy, Natural Resources, Agriculture, Environment", strengthening the link between technological innovation and environmental sustainability.

To date, autonomous systems dedicated to outdoor cleaning are still few and the AVELICA project would be among the first in this field. The AVELICA project is strongly focused on the creation of robust numerical simulation models, ranging from the study of dynamics to the design of control and communication algorithms. The goal is to create a simulation model that can faithfully represent the real system, acting as a digital twin. This approach offers the great advantage of being able to predict failures and make any design changes directly on the simulation model, before moving on to the prototyping and testing phases. This methodology aims to optimize the different design phases and reduce the time required to produce the finished product. Another goal of the project is to integrate a robotic arm that can interact with its environment through a vision system to perform the operations required by the project. Since these operations must be performed in complex and dynamic environments, it will be crucial to test robotic navigation algorithms in order to identify the best solutions to achieve the project's ultimate goal.

1.3 Mobile Robots and Rovers: Classifications and Features

The term rover refers to a specific mobile robot capable of moving over uneven terrain, originally used for planetary exploration. As defined by the International Organization for Standardization (ISO), a mobile robot is "an autonomous system capable of traversing terrain with natural or man-made obstacles. Its chassis is equipped with wheels, tracks, or legs, and may include a manipulator for handling work pieces, tools, or special devices. Various preplanned operations are performed based on a preprogrammed navigation strategy taking into account the current state of the environment" (1). From this definition it is possible to see the essential difference between a mobile robot and an industrial robot: a mobile robot is one that is capable of moving in the environment in which it operates, whereas industrial robots are usually more or less stationary, often consisting of an articulated arm (manipulator) and a gripper assembly (end effector) attached to a fixed surface. This definition of a mobile robot includes any intelligent machine that moves relative to the external environment with limited human interaction, either completely autonomously, i.e. capable of navigating an unknown environment without the need for guidance devices (self-driving vehicles), or alternatively relying on guidance devices to navigate a predefined navigation path in a known, controlled space (2; 3). Mobile robots can be classified according to several significant characteristics:

- Mobility system: wheels, tracks, or legs.
- Suspension: elastic or articulated systems.
- Steering system: Ackermann, explicit, or skid steering.
- Control algorithms and guidance/navigation systems.
- Operating environment and applications: rough terrain, flat surfaces, etc.

1.3.1 Mobility Systems: Wheels, Tracks, and Legs

When designing a mobile robot, the designer must be aware of the characteristics of the environment in which the robot will have to operate: the characteristics of the terrain, the height of any obstacles and the friction of the ground all influence the choice and design of the handling system to be adopted. An uneven terrain, or one that is not coherent or full of obstacles, is a real challenge for the designer, who will therefore be forced to consider increasingly complex solutions, depending on the environment and the nature of the terrain, as well as the availability of energy to power the robot. On this basis, the various mobile robot handling systems can be classified into three main categories:

- Wheeled robots (W): Robots equipped with wheels.
- Tracked robots (T): Robots equipped with tracks.

-
- Legged robots (L): Robots equipped with legs.

Combining these three categories results in additional handling systems that can be called hybrid handling systems: leg wheel (LW), leg track (LT), wheel track (WT), and leg wheel track (LWT). The three main and four hybrid categories of mobile robots are summarized in Figure 1.1.

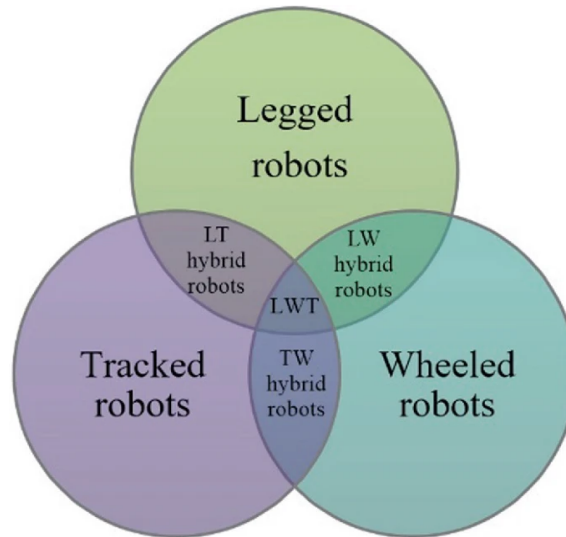


Figure 1.1: Classification of robotic mobility systems based on locomotion types (4)

Because these systems are morphologically different, a detailed comparison between them can only be made after defining a set of common characteristics that can be objectively evaluated among all handling systems. Some of these characteristics are related to mobility performance and can therefore be assessed quantitatively: maximum speed, ability to overcome randomly shaped obstacles, ability to climb steps/stairs, ability to climb slopes, ability to walk on non-cohesive ground or uneven terrain, energy efficiency, and autonomy. They allow a quick comparison between systems. In contrast, other characteristics such as mechanical complexity, control system complexity (software and hardware), technology readiness, and system reliability are not easily compared, but still require estimation to make a comparison between handling systems, also relying on the experience and judgment of the designer. Therefore, it should be pointed out that there is no single exact solution when comparing and selecting handling systems for a mobile robot. The choice should be made after a comparison of the above characteristics, an analysis of the cost of implementation and an appropriate analysis of the working environment of the robot, knowing what are the advantages and what are the disadvantages of the three previously defined categories and in which situations it is most appropriate to use them.

1.3.1.1 Tracked Robot

Tracked robots, due to their large ground contact area, are suitable for moving over uneven terrain and obstacles, but they move more slowly and consume more energy than wheeled robots. Due to the fact that the side profile of the track can be seen as a polygon with protruding vertices, the movement of the track on the ground causes considerable vibrations that are difficult to dampen. This leads to considerable complications in the design phase, while also limiting the maximum speed and reducing mechanical efficiency. Tracked robots can be classified according to the number and arrangement of their tracks. They generally have two parallel tracks with differential steering.

An example of this is Nanokhod, the miniaturized space tracker for space exploration (Fig. 1.2) based on Russian (5) technology.

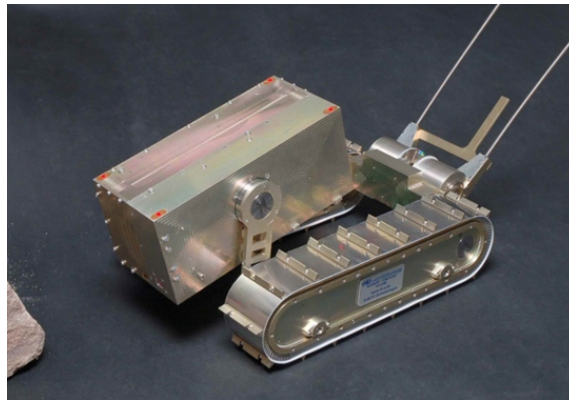


Figure 1.2: Nanokhod (5)

To further improve adaptability to rough terrain and high and uneven obstacles, the side profile of the tracks can be made with a triangular shape as shown in Fig. 1.3.

1.3.1.2 Legged Robot

In recent years, there has been a growing interest in legged mobile robots that are clearly biologically inspired. The simplest classification of legged robots is based on the number of legs. There are robots with two legs (inspired by humans), with four legs (inspired by tetrapods), and with more than four legs (inspired by insects). Originally, these robots were primarily used as technology demonstrators and research tools, while others were used for entertainment and recreation. The problem with these robots is that they are relatively slow and consume a lot of energy due to a large number of actuators. Unlike wheeled robots, they require complex control systems for positioning, since they have a much larger number of degrees of freedom. However, if we consider the advantages of such configurations in terms of adaptability to impervious and highly irregular ground, it is easy to see how much more frequently it will see legged robots in many contexts in the coming years. This is also due to the remarkable development of control algorithms and



Figure 1.3: Lobhaz DT3 (KIST)

new lightweight materials that will reduce the power consumption required to mobilize these systems. Some example are Spot and RHex as see in Fig. 1.4 and Fig. 1.5.



Figure 1.4: Spot, Boston Dynamics (6)

1.3.1.3 Wheeled Robot and Rover

Mobile robots on wheels have the advantage of being able to reach high speeds with low energy consumption. They can also carry more weight than robots that walk on mechanical legs. Another advantage is that they can be guided by controlling a few degrees of freedom. During navigation, the position and orientation of wheeled robots can be calculated more accurately than similar tracked vehicles. The disadvantage of mobile robots is their limited ability to overcome obstacles. They can be classified according

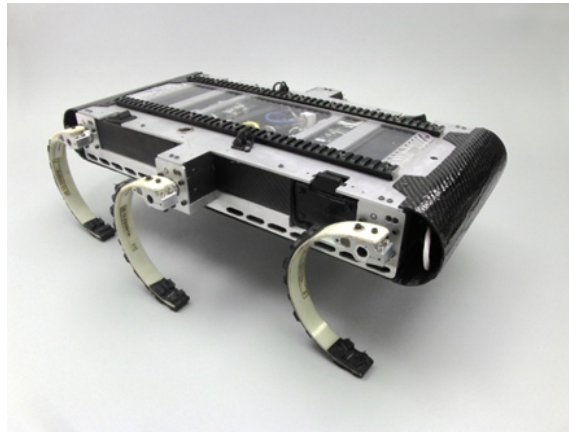


Figure 1.5: Robot RHex (7)

to the number and position of their wheels. Starting from rare and statically unstable configurations based on the use of only two wheels, such as systems based on the inverted pendulum concept or two-wheeled systems with an in-line configuration (moto-robot), they progress to statically stable systems by reaching a minimum number of wheels equal to three, where static stability is ensured when the vertical projection of the center of gravity of the robot on the ground lies inside the polygon formed by the wheel-ground contact points. Most mobile robots with wheels for rough terrain usually have 4, 6 or 8 wheels. Rough terrain wheeled mobile robots are called rovers.

More familiar examples of rovers are certainly those used for planetary exploration, as probes or vehicles for planet exploration. Historically, rovers have been used both to augment the capabilities of human crews in moving and transporting samples and to perform fully automated exploration operations. These include Lunakhod, Sojourner (Fig. 1.6), and Perseverance, which will be introduced later.

1.3.2 Suspension Systems: Rocker-Bogie and Alternatives

Since rovers are designed to move over non-cohesive and uneven terrain, 4-, 6-, or 8-wheel rover configurations are commonly used to achieve stability. One of the most important aspects is to define how to connect the wheels to the chassis so that all wheels are in contact with the ground at the same time, increasing stability and traction.

The connection systems between the wheels and the chassis are called "suspension systems", and those normally adopted for wheeled vehicles are of the elastic or articulated type. An elastic-type suspension system refers to a system made by inserting elastically deformable elements between the wheels and the rigid member to which they are connected. For a comfortable ride, cars use a combination of springs and dampers. Most heavy vehicles, such as trucks and railroad cars, use leaf springs. In contrast, rovers, which must travel highly uneven and non-cohesive terrain at low speeds, typically use articulated suspension systems.

An articulated suspension system refers to an articulated frame whose links connected

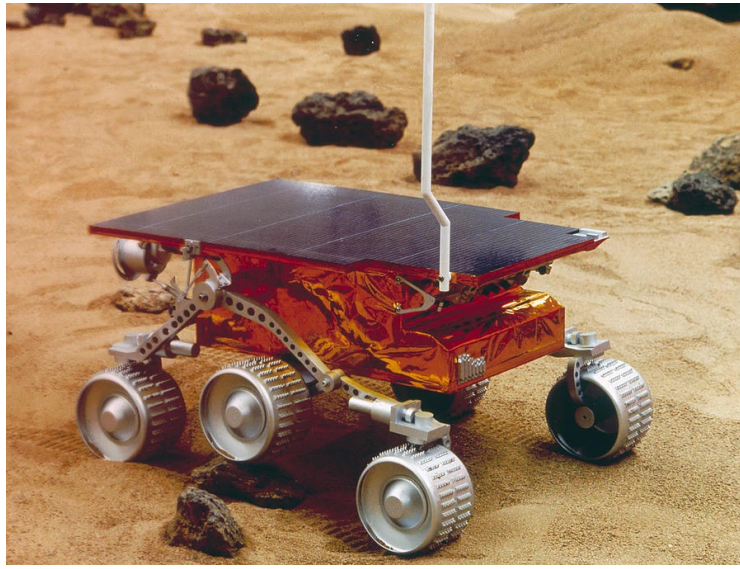


Figure 1.6: Sojourner (8)

to the wheels can move relative to the main body of the robot. This motion must be passive, not actuated, otherwise these links could be considered as legs and the robot would become a leg-wheel hybrid.

Rovers are often equipped with an articulated suspension system because these articulated frames, thanks to the mobility of passive joints, adapt their configuration to uneven terrain, reducing resistance to overcoming obstacles. Examples of this type of suspension include the CRAB system (as in Fig. 1.7) designed by the Autonomous Systems Lab (ASL) and the Rocker-Bogie system first carried by NASA on the Sojourner Mars exploration rover.

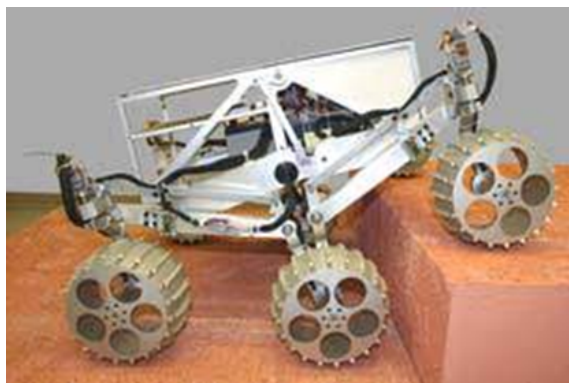


Figure 1.7: CRAB suspension equipped on a Rover (9)

The CRAB is a 6-wheel system based primarily on 4 parallel bogies, two on each side (Fig. 1.8). The two bogies of each bank are connected to the axle of the central wheel by a rotary joint. The structure is completely symmetrical and, thanks to a differential

mechanism between the two bogies, it is able to equalize the angle of inclination of the frame in such a way as to make it very stable while moving forward on uneven ground with a lot of bumps (9).

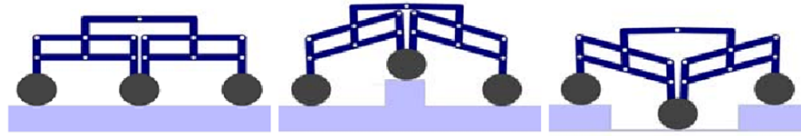


Figure 1.8: Working Scheme of the CRAB Suspension(9)

Rocker-bogie suspension (Fig. 1.9), on the other hand, is a type of passive articulated suspension that was invented by NASA in 1988 for use on the Sojourner Mars rover and has become the standard for almost all rovers used in space. The same system was used on the Opportunity and Spirit rovers in 2003, the Curiosity rover in 2012, and the Perseverance rover in 2020. The typical design of the rocker bogie has no springs or dampers, allowing the rover to overcome obstacles up to twice the diameter of the wheels while keeping all 6 wheels in constant contact with the ground. In this 6-wheel rover, the wheels are arranged symmetrically on both sides. Each side, called bank, has 3 wheels connected by two links. The main link, called the rocker, has two joints. The first joint is connected to the frame of the rover and one wheel, the other rocker joint is connected to another link called the bogie to which the other two wheels are connected (10).

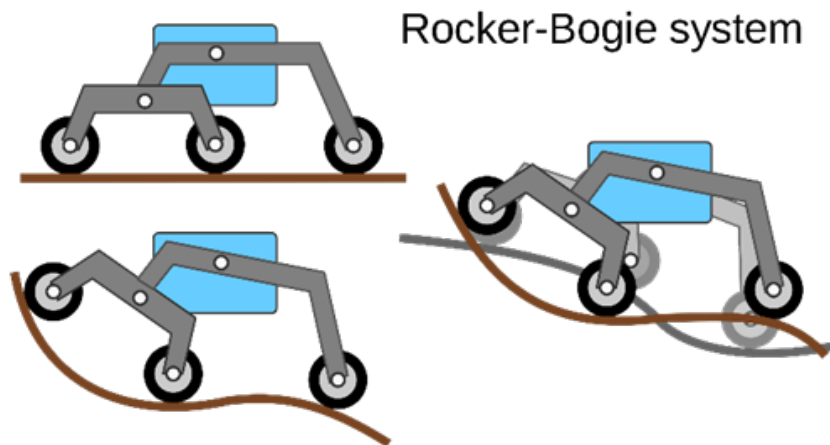


Figure 1.9: Rocker-Bogie Suspension System

The main advantage of the rocker bogie suspension is that the load on each wheel is almost identical. The two rocker-bogie mechanisms on each bank are connected by a differential mechanism constructed of two brackets and a rocker arm. In designing a suspension system for a rover, one must consider simplicity and lightness, springless connections to maintain equal tractive force on the wheels, and equal distribution of load

on each wheel in a way that prevents slippage. These considerations arouse interest in the rocker bogie suspension system, which can meet all these requirements simultaneously.

1.3.3 Steering Mechanisms

For mobile robots equipped with tracks or wheels, an appropriate steering system must be defined to allow them to follow variable trajectories.

1.3.3.1 Explicit and Skid Steering

Tracked robots, generally characterized by two parallel tracks, use a differential steering system that adjusts the rotational speeds of the two tracks. This type of system allows tracked robots to turn with small turning radii: by locking one track, the robot can turn with a turning radius equal to the width of the vehicle. Also, by reversing one track, the robot can turn in place, improving maneuverability in confined spaces.

Rovers, on the other hand, can use different steering strategies depending on the number of wheels they are equipped with. Three-wheel rovers/mobile robots use two differential steering wheels on the same axle and a third idler wheel on a second axle to provide stability to the system. They are widely used in environments with flat and uniform surfaces, such as mobile robots used for goods handling in warehouses or in robot vacuum cleaners, because of their high maneuverability, ability to rotate around a vertical axis, and simple control strategy. This three-wheeled configuration is not suitable for environments with uneven terrain, obstacles, slopes, and non-cohesive floors due to its low stability. For rovers equipped with 4, 6, or 8 wheels, steering can be achieved using several different methods.

For high speed rovers, the steering system is derived from the automotive industry and is called "Ackermann steering"(Fig. 1.10). The goal of this configuration is to avoid lateral tire slippage when following the path of a curve at high speed. Assuming we have a four-wheel rover, this goal is achieved by rotating the two steered wheels through two different angles relative to the neutral configuration during cornering. The inner wheel on the inside of the turn is turned by a larger angle than the outer wheel when steering. Ackermann's geometric solution requires that all wheels have their axles arranged as radii of circumference with a common center, which will be the center of rotation. Since the rear wheels are fixed, the center of these circumferences is on a line parallel to and coincident with the rear axle (11).

The disadvantages of this configuration are the increased complexity of the design and the need to provide additional components. Also, with this configuration, it will not be possible to rotate the robot to a fixed position, but it will be necessary for the robot to move forward in order to steer.

For rovers with low forward speeds, the most common steering modes that can be adopted are "Explicit Steering" and "Skid Steering". In these two modes, each wheel is steered and controlled separately. Assuming it is implemented this concept on a four-wheeled robot, this approach is very well suited for traveling uneven and non-cohesive

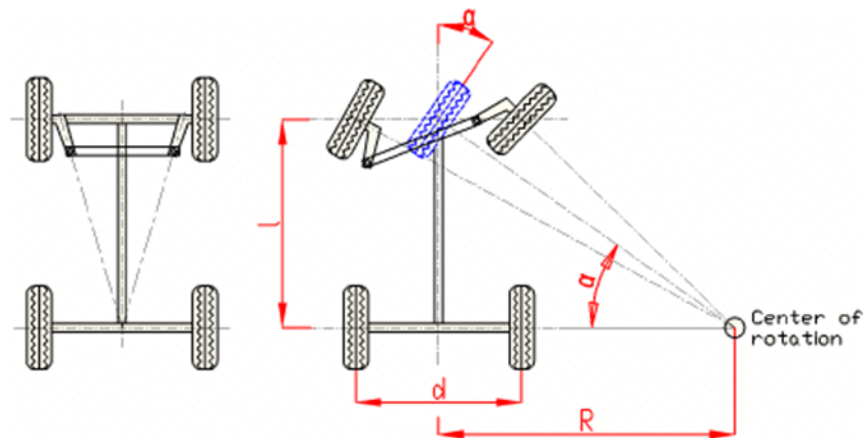


Figure 1.10: Ackermann Steering Principle (11)

terrain. Even if one of the wheels gets stuck or stops working, the remaining wheels can still guide the robot to the desired position and direction. The same concept can be applied to legged robots, where each leg works independently.

The difference between the two modes is that the first steering configuration allows the wheels to be oriented in the direction of travel in a way that minimizes slippage (Fig. 1.11 left), while the second configuration has the wheels pointing straight out and one cannot help but slip to orient the rover (Fig. 1.11 right) (12).

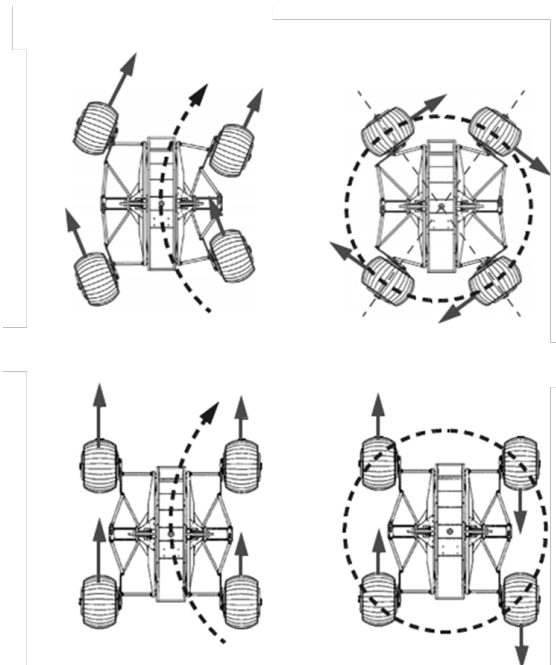


Figure 1.11: Explicit Configuration (Left) vs Skid Steering Configuration (Right) (12)

Experimental data collected on the Nomad rover allow us to highlight the main

characteristics of these two steering modes:

- Explicit Steering:
 - Reduction of all forces related to slippage.
 - Precise and predictable turns.
 - Accurate odometry.
 - Low energy consumption.
 - High turning capability (even in the presence of lateral obstacles).
 - Structural and control complexity, with increased weight and bulk due to additional motors, joints, and transmissions, which inevitably generate inefficiencies and reduce reliability.
- Skid Steering:
 - Designed with fewer components, ensuring lightweight, compactness, and construction simplicity, thereby increasing overall reliability.
 - Allows for very rapid direction changes at the cost of higher motor power and energy consumption.
 - Difficult to predict the torque required by the motors during turns.
 - Increased tire wear.
 - Reduced turning capability compared to explicit steering.
 - Very imprecise odometry.

1.3.3.2 Wheels

For the sake of completeness, it should be mentioned that most wheeled robots are equipped with deformable tires (larger sizes are usually inflatable, while smaller sizes are not) to provide fine shock reduction by avoiding suspension systems. In some cases, omni-directional wheels are used to improve robot mobility by reducing kinematic constraints during steering (13). However, these wheels are never used in unstructured environments because their mechanics can be damaged by bumps and dirt, and conventional tires provide better traction.

1.4 Perception and Computer Vision

It is clear that Computer Vision plays a crucial role in both navigation operations and tasks performed by manipulators mounted on mobile robots. Vision systems integrated on board the robot are a key component in improving the overall efficiency and autonomy of the system.

Vision systems not only improve the accuracy of operations, but also help ensure the flexibility of the robot. In unstructured environments, such as chaotic warehouses or open spaces, the ability to recognize and adapt to new situations makes the robot more versatile and effective. Computer vision enables the robot to respond to unexpected changes, such as the presence of new obstacles or a change in the position of an object being manipulated, reducing the need for human intervention.

Computer vision is an artificial intelligence discipline that enables computers and devices to extract meaningful information from a scene through image analysis. The main goal is to identify objects in the scene and determine their exact location in space. While artificial intelligence focuses on developing cognitive capabilities in computers, computer vision enables them to observe and interpret their surroundings.

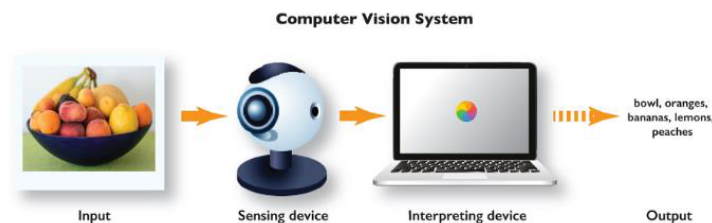


Figure 1.12: Computer vision system process, from input image to object recognition output

The main workflow of a computer vision system is shown in Fig. 1.12. Computer vision applications based on two-dimensional image processing have a significant limitation: the lack of information about the third dimension, depth. When a scene is captured by a camera, the three-dimensional world is projected onto a two-dimensional image, resulting in the loss of essential information about the geometry of the object, such as its distance and depth.

In computer vision, the ability to recover the third dimension plays a crucial role. This information not only allows for estimating the distance between the observer and the object, but also for understanding the three-dimensional properties of objects, with a focus on their depth. Such capabilities are critical for improving the accuracy and functionality of computer vision-based systems.

1.4.1 Historical Evolution and Foundations

Computer vision was born in the 1960s, with initial studies aimed at developing algorithms for interpreting digital images, such as geometric shape recognition and edge detection. In the 1970s, thanks to mathematical models of the geometry of vision, applications such as three-dimensional reconstruction and object recognition began to be explored, laying the foundations for its use in robotics.

In the 1980s and 1990s, computer vision found its way into industry with algorithms for quality control and process automation. Fundamental techniques such as edge detection (e.g. the Canny method) and image segmentation were developed.

The introduction of Convolutional Neural Networks (CNNs) in the early 2000s, together with the availability of large datasets such as ImageNet (14), revolutionised the field and significantly improved object recognition. Algorithms such as AlexNet (15) marked the beginning of a new era, making computer vision an indispensable technology.

Today, computer vision uses advanced technologies such as Vision Transformers (ViT) (16), finds applications in robotics, autonomous vehicles and environmental monitoring, and continues to be an ever-evolving discipline.

1.4.2 Fundamentals of Computer Vision

In recent decades, computer vision has played a central role in the development of autonomous and robotic systems, enabling them to perceive and interpret their environment with increasing accuracy and reliability. Understanding the mathematical and geometrical principles of image formation and perspective projection is crucial to obtaining a realistic representation of the 3D world from 2D images. The pinhole model describes the projection of 3D points onto an image plane, highlighting the loss of depth information. To correct for optical and geometric distortions, intrinsic and extrinsic matrices are used to describe the internal properties of the camera and its position relative to the scene, respectively.

These principles are fundamental to robotics applications such as autonomous navigation, object manipulation and 3D reconstruction, improving interaction with dynamic and unstructured environments.

1.4.2.1 Perspective Model

The mathematical model of a camera makes it possible to establish a relationship between a point in real three-dimensional space and its representation in the two-dimensional image produced by the camera. This process, known as image formation, involves the projection of the three-dimensional world onto a two-dimensional surface, resulting in the loss of depth information. The transformation from 3D to 2D is called perspective projection.

The simplest model that can be considered to describe this phenomenon is that of the pinhole camera. In this model, the camera is idealized as a black box with an infinitesimal hole in one of its faces. When an object is placed in front of the hole, light passes through

it and projects an inverted image of the object onto the bottom of the box, called the image plane. This model, although simplified, is the basis of many mathematical representations used in computer vision and graphics (Fig. 1.13).

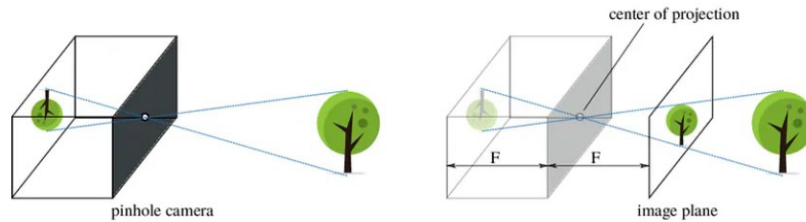


Figure 1.13: Pinhole model (17)

Consider a reference system with the origin O coincident with the camera hole. The abscissa and ordinate axes define a plane parallel to the image plane, while the third axis, called the optical axis, is perpendicular to the image plane.

Let $P(X, Y, Z)$ be a point in the three-dimensional space of the scene and $P'(X', Y')$ its projection onto the image plane. Let f be the focal length, i.e. the distance between the camera aperture and the image plane.

By applying the geometry of perspective projection and using the similarity between the right triangles involved (as shown in Fig. 1.14), it is possible to derive the mathematical relationship that links the coordinates of P to those of P' .

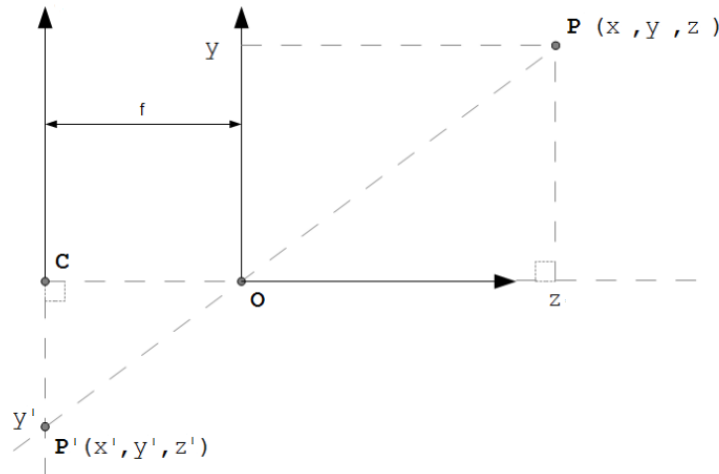


Figure 1.14: Prospective model showing the projection of 3D coordinates (P) onto the image plane (P')

The model of image formation is called the perspective model:

$$X' = -f \cdot \frac{X}{Z} \quad Y' = -f \cdot \frac{Y}{Z} \quad (1.1)$$

Similarly, we can model the perspective projection by placing the image plane in front of the hole to eliminate the minus and make the image straight:

$$X' = f \cdot \frac{X}{Z} \quad Y' = f \cdot \frac{Y}{Z} \quad (1.2)$$

Considering the coordinates of the image plane point $P(X', Y')$ as a homogeneous vector of the type $\tilde{P}(\tilde{x}; \tilde{y}; \tilde{z})$ with:

$$\tilde{x} = f \cdot X \quad \tilde{y} = f \cdot Y \quad \tilde{z} = Z \quad (1.3)$$

Then in matrix form, through the use of homogeneous coordinates, it can be written that:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = [C] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (1.4)$$

The matrix $[C]$ is called the chamber matrix. It can be written as the product of two matrices:

$$[C] = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (1.5)$$

where:

$$\Pi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (1.6)$$

The projection matrix allows 3D coordinates to be mapped to 2D coordinates. The pinhole model, while useful for its simplicity, is not practical due to its idealized nature and its inability to model optical aberrations and distortions. To overcome these limitations, thin lenses (Fig. 1.15) are studied, which offer useful properties for modeling real images:

1. Rays parallel to the optical axis are refracted by the lens and converge at a point called Focus.
2. Rays passing through the center of the lens are not refracted.

The distance at which the image is created (inverted), the distance of the object from the lens, and the focal length of the lens are related by the equation:

$$\frac{1}{p} + \frac{1}{q} = \frac{1}{f} \quad (1.7)$$

The thin lens model is valid for distances much greater than the thickness of the lens. Unlike the pinhole with infinite depth of field, thin lenses have a depth range p where the points are in focus (18).

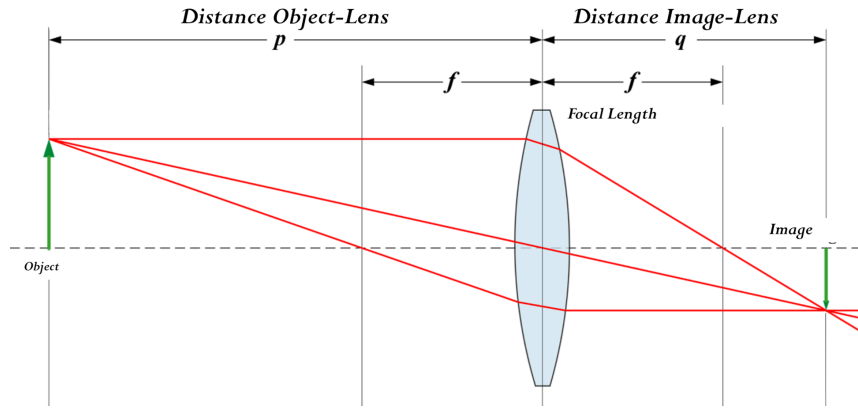


Figure 1.15: Thin Lens physics

1.4.2.2 Extrinsics and Intrinsics Matrix

The extrinsic parameter matrix contains information about the kinematic configuration of the camera in space, while the intrinsic parameter matrix contains information about the optics and sensors of the camera (Fig. 1.16).

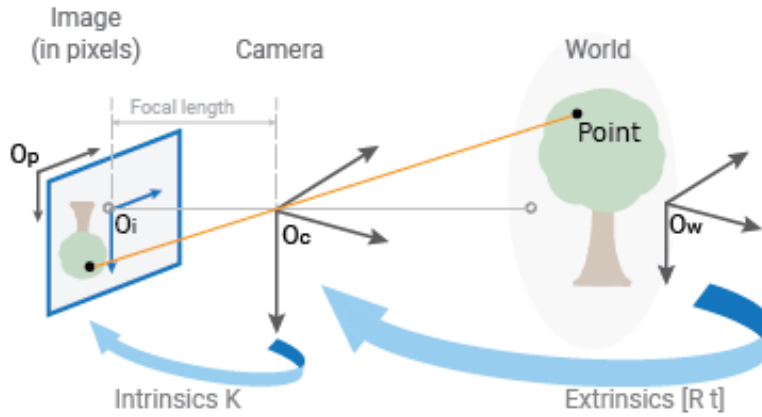


Figure 1.16: Meaning of Extrinsics and Intrinsics (19)

Extrinsic parameters transform real scene points into camera coordinates, while intrinsic parameters map these coordinates to the image plane.

The transformation matrix $[T_p]$ contains a translation vector $\{t\}$, which defines the relative position between the two reference systems, and a rotation matrix $[R]$, which aligns the respective axes.

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (1.8)$$

The transformation matrix converts a point from one reference frame (e.g. the camera) to another (e.g. the world frame). This process involves a discretization determined by the $W \times H$ grid of the camera sensor, where the photosites correspond to the pixels of the image.

The origin of the pixel frame is located in the upper left corner, and the pixel coordinates are represented by vectors (u, v) of positive integers. The principal point (u_0, v_0) is

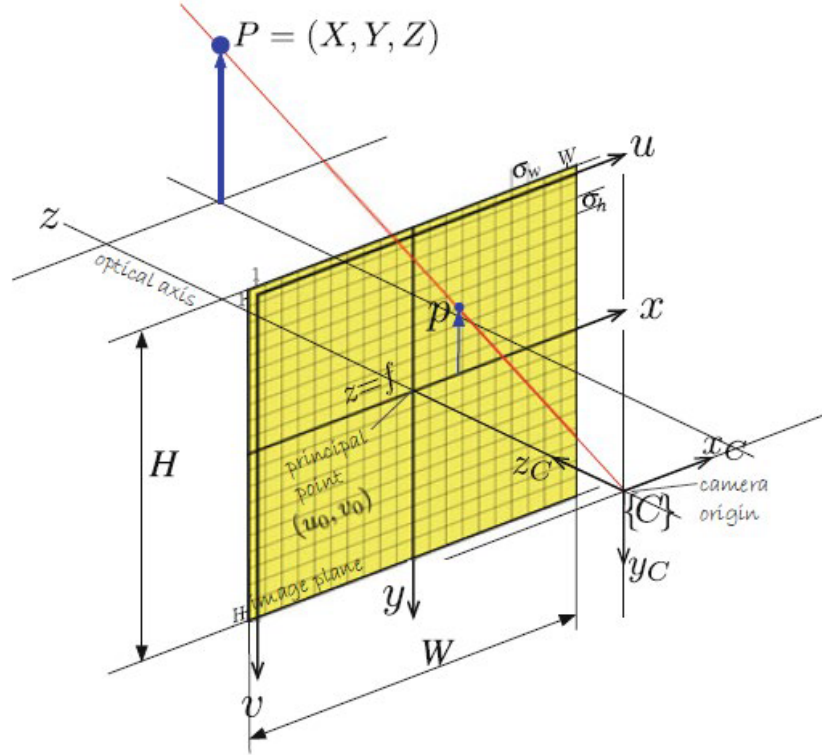


Figure 1.17: Projection Model (18)

the pixel where the optical axis intersects the image plane. Together with the coordinates of the focal point, they form the intrinsic matrix.

$$K = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1.9)$$

s is the skew parameter, which represents the angle between the x and y axes of the sensor. In an ideal system, the axes are perpendicular and s is zero. However, in some cases there may be a slight skew due to imperfections in the design of the sensor.

The intrinsic matrix is essential in many computer vision applications, such as camera calibration, 3D reconstruction, and augmented reality, because it allows us to understand and correct for distortions introduced by internal camera features (18).

1.4.2.3 Lens Distorsion

Lenses have imperfections that cause distortions such as color fringing, spherical aberration, and astigmatism (focus variation), as well as geometric distortions that shift points on the image plane from their actual position.

Lens distortion has two components: radial distortion and tangential distortion. Radial distortion causes the pixels to shift along radial lines from the focal point and is well approximated by a polynomial:

$$\delta_r = k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6 + \dots \quad (1.10)$$

where r is the distance of the pixel from the principal point.

Tangential distortion occurs perpendicular to the radius, but is generally less significant than radial distortion. Tangential distortion occurs when the lens and the shutter plane are not exactly parallel. A point $(u_d; v_d)$ in the distorted image is related to its real coordinate $(u; v)$:

$$u_d = u + \delta_u \quad v_d = v + \delta_v \quad (1.11)$$

where:

$$\begin{pmatrix} \delta_u \\ \delta_v \end{pmatrix} = \underbrace{\begin{pmatrix} u(k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots) \\ v(k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots) \end{pmatrix}}_{\text{radial}} + \underbrace{\begin{pmatrix} 2p_1 uv + p_2(r^2 + 2u^2) \\ p_1(r^2 + 2v^2) + 2p_2 uv \end{pmatrix}}_{\text{tangential}} \quad (1.12)$$

In practice, five coefficients are sufficient to describe the radial distortion and the distortion model is usually parameterized by $(k_1, k_2, k_3, p_1, p_2)$, which are considered as additional intrinsic parameters (18).

1.4.2.4 Stereo Vision

In the previous paragraph, the perspective projection relations were defined:

$$\begin{cases} u = f_x \cdot \frac{x}{z} + u_0 \\ v = f_y \cdot \frac{y}{z} + v_0 \end{cases} \quad (1.13)$$

The transformation of a point in three-dimensional space (X, Y, Z) into the coordinates (u, v) of the image plane can be obtained using the projection equations. However, the problem becomes more complicated if you want to trace the three-dimensional position (X, Y, Z) of a point from its image (u, v) . In this case, the system has three unknowns in two equations, resulting in an indeterminate system. This obstacle can be overcome by using at least two cameras. Stereo vision, or stereopsis, is a technique that allows us to perceive the depth of a scene through images captured by two or more cameras placed slightly apart, replicating the way the human eye works. The basic principle of this technique is triangulation. When two cameras capture images of the same scene from different perspectives, they produce a pair of images, called a stereo pair, that have

disparities: differences in the positions of corresponding points in the two images. By analyzing these disparities, it is possible to calculate the depth of objects in the scene. Objects closer to the cameras have larger disparities, while those farther away have smaller disparities. To apply triangulation, it is necessary to calibrate the stereo equipment, which consists in identifying the intrinsic and extrinsic parameters of the cameras.

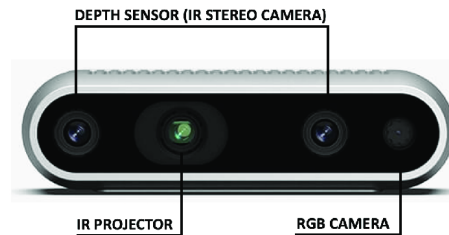


Figure 1.18: Stereo Camera

1.4.3 Techniques in Computer Vision

In recent decades, computer vision has benefited from advanced techniques that combine classical algorithms with machine learning-based approaches. Among the classical methods, feature detection using algorithms such as SIFT (20), SURF (21) and HOG (22) has been an important step in object recognition and image segmentation. However, these traditional approaches have been largely superseded by the advent of Convolutional Neural Networks (CNNs), which automate feature extraction and improve recognition accuracy (15).

Today, advanced architectures include models such as R-CNN and its variants (Fast R-CNN, Faster R-CNN, Mask R-CNN), which have revolutionised object detection and segmentation (23; 24; 25). In addition, Generative Adversarial Networks (GAN) (26) have opened up new possibilities in synthetic image generation, improving data augmentation and environmental simulation.

More recently, Vision Transformers (ViT) (16) have introduced the use of autoregressive attention, an approach that eliminates the reliance on convolutions, leading to significant improvements in classification and segmentation performance. The integration of multimodal techniques, such as CLIP (27), has allowed visual and textual data to be combined, extending the capabilities of systems based on computer vision.

These advanced techniques have allowed computer vision to be applied in increasingly complex contexts, improving the accuracy, scalability and generalisability of models.

1.4.4 Applications in Robotics

Computer Vision has a significant impact on many industries due to its ability to analyse and understand images and videos with increasing precision. Some of the applications related to this thesis include

- Machine vision systems are widely used for quality control, guiding robot manipulators and monitoring production lines. Similarly, in tasks performed by manipulators, computer vision is essential for recognizing objects, locating them accurately, and performing complex tasks. Manipulators use the data acquired by vision systems to perform pick-and-place operations, assembly, gentle manipulation, or interaction with moving objects (28; 29; 30). For example, during an automated picking session, the vision system can locate an object in a crowded or disorganized environment and determine the position and orientation required for picking with pinpoint accuracy.
- In the context of autonomous navigation, vision systems enable the robot to understand its environment by identifying obstacles, maps, or landmarks. This capability is essential for performing tasks such as trajectory tracking, obstacle avoidance, and navigation in dynamic environments. Advanced algorithms enable the detection of pedestrians, road signs and dynamic obstacles, improving the system's perception and decision-making capabilities (31; 32).
- Drones equipped with computer vision scan crops to monitor plant health, detect pests or optimise irrigation, improving agricultural efficiency (33).

These applications demonstrate how computer vision not only improves operational efficiency, but also enables innovative solutions in areas of critical importance to society.

1.4.5 Integration in Other Systems

These applications demonstrate how computer vision not only improves operational efficiency, but also enables innovative solutions in areas of critical importance to society.

- Computer vision is related to machine learning, with supervised and unsupervised models improving recognition and classification capabilities. Clustering algorithms or self-supervised learning methods are gaining popularity, reducing the dependence on large labelled datasets (34; 35).
- In collaborative robots (cobots), computer vision enables safe interaction with humans. Vision-based systems recognise movements and gestures, prevent collisions and dynamically adapt operations to the context (36).
- Computer vision is essential for tracking motion, recognising surfaces and integrating virtual objects into the real world. This synergy is crucial for applications in areas such as entertainment, education and industrial design (37).

1.4.6 Challenges and Future Directions

Despite significant progress, computer vision still faces several challenges. One of the most important is the variability of lighting, which can adversely affect the performance

of models in uncontrolled lighting environments. The presence of partially hidden objects or particularly crowded scenes also complicates accurate recognition, despite the use of advanced techniques such as Mask R-CNN (25).

Another important limitation is the bias in the datasets, which can lead to poor performance in contexts not represented during training (38). This problem is particularly critical in sensitive domains such as face recognition, where studies have shown significant differences in accuracy between different demographic groups (39). In addition, modern convolutional neural networks require large computational resources, limiting their use on embedded devices or in applications that require real-time responses (40).

Finally, the lack of interpretability of deep learning models is a significant obstacle, especially in contexts where it is essential to understand and justify decisions, such as in medicine and security (41). Generalization also remains a challenge, with many models showing excellent performance in controlled environments, but struggling to adapt to new or unpredictable situations.

1.5 Mobile Manipulator Robots

The increasing need to interface with the environment to perform specific tasks has made the integration of robotic arms into mobile robots, especially rovers, indispensable. This combination makes it possible to perform complex tasks in dynamic environments, such as the RB-VOGUI+ robot equipped with a robotic arm (Fig. 1.19).



Figure 1.19: RB-VOGUI+ (42)

1.5.1 Robotic Arm: Characteristic and Applications

Articulated robots, also known as manipulators, are key tools in mobile robotics. Some of them can provide six/seven degrees of freedom, performing complex movements with extreme precision. This makes them ideal for applications that require flexibility and accuracy, such as arc welding, painting, assembly of intricate components or even robotic surgery.

In an industrial context, manipulators mounted on mobile platforms are used to optimise logistics operations. On assembly lines, for example, they can be used to handle heavy or delicate materials, increasing production efficiency and reducing the risk of human error.

1.5.1.1 Integration on Mobile Robots

The integration of manipulators on mobile platforms is an innovative solution for extending the operational capabilities of robots. This synergy makes it possible to combine the

ability to navigate autonomously with the precise manipulation of objects, making the system particularly suitable for operations in dynamic and unstructured environments. For example, a mobile robot equipped with a manipulator can be used for storage operations in automated warehouses, improving the management and transport of materials. Other applications include collecting objects in agricultural environments or cleaning public spaces. An important implementation of this technology is described in (43), where an autonomous ground vehicle equipped with a robotic arm and a vision system was designed to detect and remove obstacles, increasing the safety and autonomy of the system.

1.5.2 Vision Systems on Mobile Robots

Vision systems play a crucial role in improving the autonomy and precision of mobile robots. Using cameras and advanced visual recognition algorithms, these systems enable robots to identify objects, estimate their position and adapt to changes in the environment. Machine vision is particularly useful for applications such as automatic material recognition in a warehouse or navigation in complex urban environments.

1.5.3 Challenges in Integration

Despite significant advancements, integrating manipulators with mobile platforms presents several technical challenges. Key issues include system stability, synchronization of movement and manipulation, and energy efficiency.

Handling heavy loads can compromise system stability, particularly on uneven terrain. Coordinating the movements of the mobile platform with the manipulator's operations is inherently complex and demands advanced control algorithms. Furthermore, the simultaneous operation of both the platform and the manipulator necessitates highly efficient energy management.

Prospects for the near future of mobile robotics focus on the development of lightweight but robust materials, the use of artificial intelligence algorithms to optimize movement planning, and the integration of collaborative technologies to make interactions with humans safer. In addition, the increasing use of advanced sensors and neural networks is expected to improve the perception of the environment and the autonomous decision-making capabilities of robots.

The combination of manipulators and mobile platforms is one of the most promising innovations in robotics. This combination makes it possible to combine precision, flexibility and autonomy to successfully address complex operational challenges and bring significant benefits to sectors such as industry, agriculture and public services.

1.6 Controllers and Autonomous Navigation Systems

1.6.1 Controllers

Control is a fundamental aspect in robotic systems as it ensures the proper functioning of the devices in terms of stability, precision and reactivity. In the context of this thesis, an in-depth analysis of the control strategies used to regulate the dynamics of the robotic system has been carried out, focusing on two widely used control methods: PID control and Sliding Mode Control (SMC).

The approach adopted for the discretisation of the controllers was the Forward-Euler method, which allows continuous equations of dynamic systems to be transformed into discrete form, making them implementable on real-time computing platforms. This discretisation method was chosen for its computational simplicity and its effectiveness in approximating the behaviour of the controlled system with adequate accuracy.

In this chapter, only basic control theory is presented, providing an overview of the main techniques and mathematical foundations of control systems.

1.6.1.1 PID

PID control is used to facilitate the appropriate adjustment of the speed of the rover to ensure that the desired trajectory is maintained. In this section, the operational mechanics of the control system will be explained, after which the practical aspects of its real-time use on the rover will be explored. In general, control can be defined as an automatic, semi-automatic, or manual mechanism that regulates a process. Control systems can be classified according to their functionality. Some are designed to regulate a single physical quantity, such as temperature, pressure, or flow. Others are capable of managing multiple variables simultaneously, affecting the state or behavior of multiple elements within a system. This type of control belongs to the category of negative feedback systems, which are widely used in automatic control systems to improve the regulation of a process. The feedback mechanism uses a computational algorithm to implement a corrective action based on the difference between the actual value and the desired value of the controlled quantity, with the ultimate goal of achieving stability.

A proportional-integral-derivative (PID) controller consists of three main components:

- The proportional term (P) is directly proportional to the current error. The magnitude of the corrective action applied is directly proportional to the magnitude of the error. This component responds quickly to changes in the error, providing an immediate response.
- The integral term (I) accumulates the error over time and applies a correction proportional to the sum of these errors. This term helps eliminate residual bias that proportional control cannot correct. As a result, the error is reduced to a negligible level over a long period of time.
- The derivative term (D) responds to the rate of change of the error by predicting the future behavior of the error and applying an anticipatory correction. This component is useful for damping and preventing excessive oscillations or instability in the system.

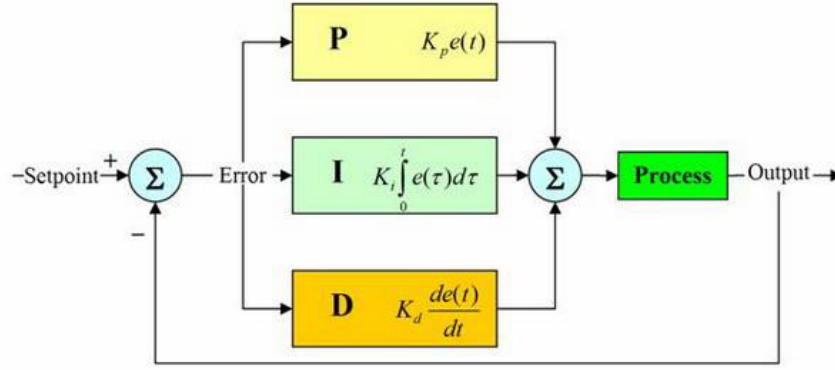


Figure 1.20: Schematic representation of a basic PID control scheme

The combination of these three terms allows PID control to be implemented in a variety of applications, including controlling the speed of a rover to maintain an accurate velocities.

Accordingly, the PID controller signal obtained by a controller as the one seen in Fig. 1.20 is defined as the sum of the three terms:

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{de(t)}{dt} \quad (1.14)$$

where K_P , K_I , and K_D are the proportional gain, integral gain, and derivative gain of the PID controller, respectively.

As shown in Fig. 1.20, the control process begins with a comparison between the setpoint, which represents the desired value of a specific quantity, and the process value, which represents the actual measured value. The difference between these two values, denoted as $e(t)$, represents the error. The sign of this error indicates whether the process value is below (positive error) or above (negative error) the setpoint.

The PID control mechanism responds to this deviation by gradually reducing the error, making the system highly adaptable to different scenarios and operating contexts. The proportional term in PID control plays a critical role by providing an immediate response to the current error magnitude, helping to quickly correct deviations.

In proportional control, the reaction to the error value is direct and immediate. The feedback control signal $u(t)$ is calculated in proportion to the current error $e(t)$ according to the following formula:

$$u(t) = K_p \cdot e(t) \quad (1.15)$$

In this context, K_p represents the proportional gain factor that determines the strength of the control response to the error. A higher K_p value results in a more responsive controller, which reduces the error more quickly. However, this increased responsiveness can also lead to overshoot and potential instability in the system. On the other hand, a lower K_p value slows down the controller response, but may improve system stability.

Selecting an appropriate K_p value is critical to effectively balancing response speed and system stability. This balance ensures that the controller is well suited to the specific

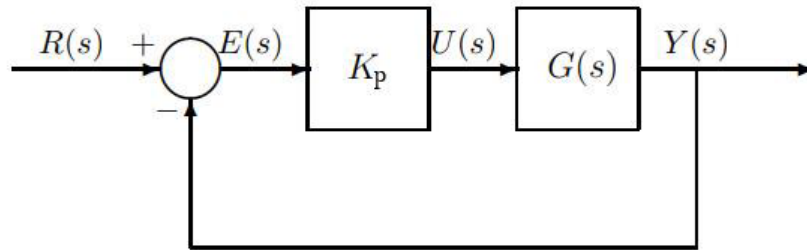


Figure 1.21: Generic proportional controller scheme

requirements of the application and the dynamic characteristics of the process being controlled.

A more detailed representation of a generic scheme of a proportional controller is shown in Fig. 1.21.

In the field of automatic control, the Laplace transform is a powerful tool for analyzing and designing control systems in the frequency domain, providing deeper insight into process dynamics. In this context:

- $R(s)$ is the Laplace transform of the reference signal, which corresponds to the desired value of the system output.
- $E(s)$ is the Laplace transform of the feedback error, defined as the difference between the reference signal and the actual system output ($E(s) = R(s) - Y(s)$).
- $U(s)$ is the Laplace transform of the control signal, which is the adjusted input from the controller to the system to minimize the error and achieve the desired system behavior.
- $Y(s)$ is the Laplace transform of the system output

The use of Laplace transforms allows control strategies, such as PID control, to be formulated in a mathematical framework that simplifies the analysis and design of stable and responsive control systems.

One of the main limitations of a simple proportional controller is its inability to completely eliminate system error in a closed-loop system, especially in scenarios that require high accuracy or operate under varying conditions. As a result, proportional control alone is often insufficient to achieve the desired control goals. Additional components are integrated into the control system to overcome these limitations:

- **Integral Control:** This component accumulates errors over time, giving greater emphasis to persistent errors. It helps eliminate static error (offset) that proportional control alone cannot resolve. The integral time constant determines the rate at which the controller responds to the accumulated error.

- **Derivative control:** This component takes into account the rate of change of the error and provides an early response to rapid changes. It is particularly effective in damping oscillations and increasing system stability by acting more decisively when the error changes rapidly.

The choice of which PID component to include or exclude depends on the specific characteristics of the process being controlled. Some processes may require a more reactive approach, while others may benefit from greater stability. The PID configuration is adjusted accordingly to optimize system performance and response (44).

1.6.1.2 Sliding Mode Control

PID is one of the classic control methods, still widely used today because of its simplicity and wide applicability. It is chosen in about 90 % of the cases. However, a notable drawback of PID control is its reduced performance in nonlinear systems. Sliding Mode Control (SMC) is a controller specifically designed to handle the nonlinear conditions of such systems.

As highlighted in (45), SMC offers several advantages, including robustness, the ability to handle nonlinear and time-varying systems, fast dynamic response, and strong performance over a wide range of operating conditions. In the current analysis, the DC motor exhibits variations in electrical parameters due to factors such as temperature rise, current and voltage variations, variable load conditions, and changing driving and operating scenarios. These factors introduce nonlinear characteristics, which require a nonlinear control method.

In this context, SMC is used to control the speed of the DC motor. Sliding mode control works by causing the system state to "slide" along a predefined surface, known as the sliding surface, with the goal of steering the system state from this surface to a desired central point, as shown in Fig. 1.22.

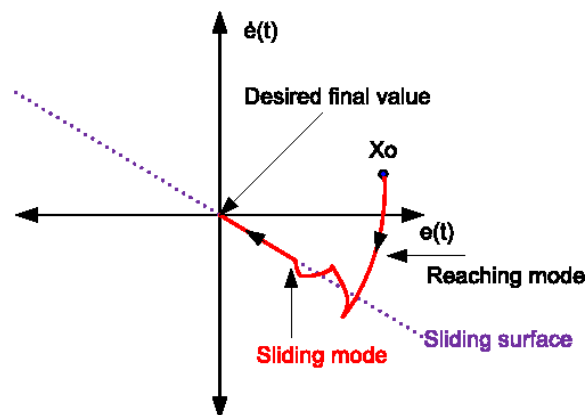


Figure 1.22: Phase portrait illustrating the dynamics of Sliding Mode Control (46)

while the Fig. 1.23, illustrates the switching of the control signal, which follows the sliding surface to the origin.

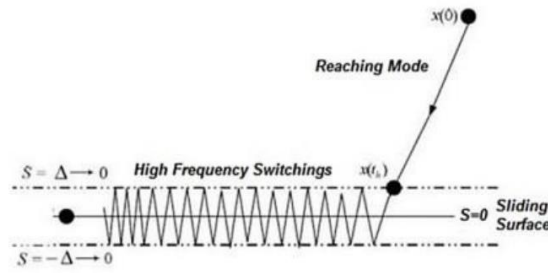


Figure 1.23: Sliding surface, reaching mode, and high-frequency switching in Sliding Mode Control (47)

The sliding surface in Sliding Mode Control (SMC) is defined by the condition where the switching function s is zero ($s = 0$). There are limit switching values $\pm\Delta$ above and below the sliding surface. If the state $x(t)$ exceeds Δ , the switching is deactivated; if $x(t)$ falls below $-\Delta$, the switching is activated. The system reaches the slip plane by alternating between stable and unstable trajectories, eventually converging to zero error on the slip plane.

The goal of SMC is to have the system output track the desired reference while minimizing the tracking error. The control signal in SMC consists of two components: the reaching mode and the sliding mode.

- Reaching Mode: This phase drives the system state toward the sliding surface and includes a switching control (u_{sw}) to force the system onto the surface.
- Sliding Mode: Once on the sliding surface, the state of the system is maintained along this trajectory using equivalent control (u_{eq}). This control ensures system stability and helps maintain the desired dynamic behavior.

The combined efforts of u_{sw} and u_{eq} enable SMC to effectively manage nonlinear system dynamics, ensuring robust and stable performance (48).

Figure 1.24 illustrates the Sliding Mode Control (SMC) diagram, where the control signal consists of two components: the equivalent control (u_{eq}) and the switching control (u_{sw}). The sliding mode control signal is thus provided by a dual voltage input defined by these components:

- Switching control (u_{sw}): This component drives the system to reach the sliding surface within a finite time and ensures that the system remains on the surface, preventing it from deviating.
- Equivalent control (u_{eq}): Once the system reaches the sliding surface, u_{eq} takes over, guiding the dynamics of the system along the surface and ensuring stable operation and adherence to the desired behavior.

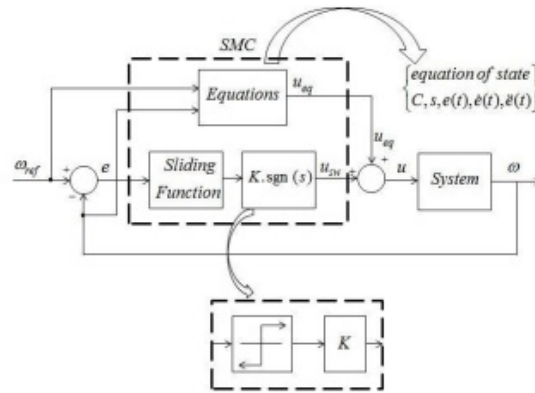




Figure 1.25: Autonomous Rover (Left) and Teleoperated Rover (Right)

Many of the autonomous navigation systems for AGVs (Automated Guided Vehicles) rely on the use of multiple sensors and thus fall, at least in part, into the realm of Sensor Fusion. However, the primary sensors for AGV navigation are often devices that use images or, more generally, light. This approach to localization and autonomous driving is known as Simultaneous Localization and Mapping (SLAM).

SLAM-equipped devices track their own motion relative to the world, rather than detecting the motion of objects within the Field of View of the camera or laser. Thanks to major advances in computer processing speed and the availability of low-cost sensors such as cameras and laser (LiDAR), SLAM is now being used in a growing number of practical applications. This method allows autonomous vehicles to construct a map of an unknown environment and locate the vehicle within that map, with a strong focus on real-time operation.

SLAM algorithms make it possible not only to map unknown environments, but also to use the collected information to perform tasks such as path planning and obstacle avoidance. Over the years, several methods have been developed to implement SLAM using different sensors, including 3D LiDAR, 2D LiDAR, radar, and monocular, stereo, or RGB-D cameras. Each technique has advantages and disadvantages, and a universal solution to all SLAM problems has not yet been achieved.

Currently, the main SLAM technologies fall into two categories: Lidar-based SLAM and Vision-based SLAM.

- Lidar-based SLAM: Uses 2D or 3D LiDAR for mapping and tracking. LiDAR (Light Detection and Ranging) emits eye-safe laser light by detecting laser reflected from surrounding objects. By measuring the time between sending and receiving the laser, it calculates the distance between the sensor and objects. This relatively mature method has been used successfully in many commercial products and has proven to be highly reliable. However, it has some drawbacks: it requires more computing power due to the large amount of data handled, and incurs significant costs for infrastructure and associated hardware.
- Vision-based SLAM: uses vision-based sensors such as monocular, stereo, or RGB-D cameras to map and locate the robot. Cameras are generally less expensive and provide detailed information about the environment, enabling accurate localization. With less onerous hardware requirements and ever-improving algorithms, Visual

SLAM is gaining popularity, especially in the design of low-to-medium-speed autonomous robots. In addition, the fact that the camera can also be used for other perception tasks makes Visual SLAM particularly attractive.

The methods described are discussed in more detail in the following chapters, where some of their possible applications are explored.

1.6.2.1 Foundations and Challenges in Robotic Navigation

Robotic navigation represents a fundamental aspect of contemporary robotics, facilitating the ability of machines to perceive their surrounding environment, plan optimal trajectories, and execute actions to achieve predefined objectives in an autonomous manner. This capability is integral to a number of fields, including autonomous vehicles, industrial automation, search and rescue missions, and planetary exploration. The field of robotic navigation is broadly divided into two categories: single-agent navigation, which involves a solitary robot interacting with its environment, and multi-agent navigation, which includes the coordinated actions of multiple robots operating collaboratively or competitively in shared spaces.

The use of robotic navigation in self-driving vehicles (Fig. 1.26) is a clear example of its application in everyday life. These vehicles rely on sophisticated navigation systems to interpret complex urban landscapes, make real-time decisions, and ensure safety. Key components include sensors such as LiDAR, cameras, and radar, which allow for obstacle detection and efficient navigation. The concept of obstacle avoidance is of particular importance, ensuring that vehicles can detect and avoid obstacles on their way to their destinations (49).

In manufacturing environments, automated guided vehicles (AGVs in Fig. 1.27) employ precise navigation algorithms to facilitate efficient transportation of materials and products. Such systems improve productivity by adapting to changes in their surrounding environment. The integration of robotic navigation into industrial processes has had a significant impact on logistics, reducing the need for human intervention.

In disaster response scenarios, robots equipped with advanced navigation capabilities are deployed to travel through hazardous environments, as the one in Fig. 1.28, with the objective of finding and providing assistance to victims. For example, autonomous drones have been used in unstructured indoor environments to carry out search and rescue operations, facilitating the real-time transmission of data to emergency teams (52).

The implementation of robotic navigation is revolutionizing the agricultural sector, as it enables autonomous systems to navigate fields and perform a range of tasks, including planting, weeding, and harvesting (Fig. 1.29). Robots equipped with sophisticated navigation and machine vision systems are capable of identifying crop rows, avoiding obstacles, and optimizing the utilization of resources. To illustrate, field robots employ SLAM-based systems to map and navigate expansive, unstructured environments, enhancing operational efficiency (53).

In the field of environmental operations, robots are employed for a variety of tasks, including cleaning beaches (Fig. 1.30). Autonomous surface and underwater robots are



Figure 1.26: Waymo-operated Jaguar I-Pace (50)



Figure 1.27: Automated Guided Vehicles during a facility task (51)

deployed in rivers, lakes and oceans to collect data on pollution levels and ecosystem health. Drones equipped with navigation capabilities are also utilised for large-scale environmental assessments, including the monitoring of forests and the mapping of carbon (54).

Robotic navigation plays a crucial role in space exploration. Mars rovers, such as the Perseverance rover, in Fig. 1.31, operated by NASA, are designed to navigate challenging terrains in order to conduct scientific experiments and collect data. The operation of these robots is contingent upon their ability to function autonomously, given the considerable communication delays with Earth. Furthermore, developments in navigation technologies have enabled autonomous access to near-Earth objects, thereby expanding the scope for exploration (56).



Figure 1.28: Robot in hazardous environment (42)



Figure 1.29: Mobile robot for agricultural operations

1.6.2.2 Principles of Single-Agent Navigation

The fundamental objective of single-agent navigation is for a robot to be able to identify its position within an environment and plan a route to a desired location while avoiding obstacles. This process requires the integration of modules responsible for perception, decision-making, and motion control.

One of the fundamental problem in single-agent navigation is Simultaneous Localization and Mapping (SLAM) (58). SLAM presents a dual challenge: the construction of a map of an unfamiliar environment and the identification of the position of the robot within that map. This simultaneous process is crucial for autonomous robots operating in environments



Figure 1.30: Robot cleans up litter (55)



Figure 1.31: NASA Rover Perseverance (57)

where the use of pre-existing maps is either impractical or unreliable.

The framework of the algorithms can be outlined as follows:

1. Localization: the robot is capable of estimating its own position and orientation in relation to the surrounding environment. This process requires the integration of data obtained from various sensors, including those that measure odometry, LiDAR, camera images, and IMUs.
2. Mapping: the robot creates a spatial representation of the environment, which is

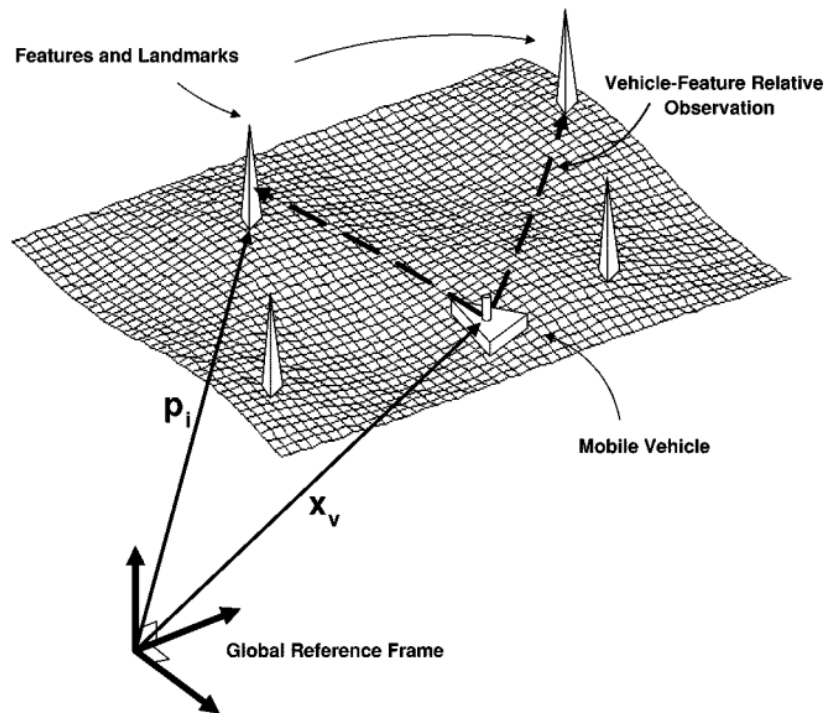


Figure 1.32: A vehicle taking relative measurements to environmental landmarks (58)

often in the form of a grid map, a feature map, or a point cloud. The map provides the robot with a reference point for navigation.

3. **Data Association:** a fundamental aspect of SLAM is the ability to identify whether the features observed by the robot at different times correspond to the same points in the environment. This prevents the robot from misinterpreting its location or duplicating map features, ensuring the accuracy and integrity of the map. Some of the main techniques used for this are: Nearest Neighbor (NN), Probabilistic Approaches, and Appearance-Based Matching.
4. **Loop Closure:** in order to ensure the continued accuracy of the map, SLAM systems are designed to detect when the robot returns to a location that has previously been visited. This process enables the correction of any accumulated errors that may have occurred in both the map and the estimated position of the robot.

Some of the most traditional approaches used for SLAM include the use of Kalman filters for probabilistic estimation of the position and orientation of the robot, Particle filters which effectively handle non-linearities in motion and sensor models, and graph-based SLAM, which optimises a graph representation of poses and landmarks.

Once the position has been defined, it is essential to plan the optimal route to reach the desired destination. Path planning algorithms span a range of approaches, from

classical techniques such as A* and the Dijkstra algorithm to modern methods including Rapidly-exploring Random Trees (RRT) and their optimized variants. These algorithms typically account for both static and dynamic obstacles.

It is evident that perception constitutes a pivotal element of navigation algorithms, with sensors playing a crucial role in this process. Single-agent systems rely on devices such as LiDAR, cameras, and IMUs to perceive their surrounding environment. Techniques such as the Extended Kalman Filter (EKF) are of great importance in the integration of multi-modal data, which is essential for the formulation of robust decisions.

1.6.2.3 Strategies for Multi-Agent Navigation

Multi-agent navigation represents a critical and advanced area within robotics, stressing the interplay and collective behaviour of multiple robots working together in a shared environment. Unlike single-agent systems, which focus on the autonomy of a single robot, multi-agent systems involve the complexity of interactions between multiple entities. These interactions introduce unique challenges that are absent or less pronounced in single-agent scenarios. For example, the presence of other agents requires constant awareness of their positions, intentions and actions to ensure effective cooperation or competition.

One of the fundamental challenges in multi-agent navigation is inter-agent communication. Effective communication allows robots to share information about their positions, goals and the environment to improve collective decision making. This can take various forms, ranging from explicit data exchange over wireless networks to more implicit methods such as interpreting the movements or behaviours of neighboring agents. The need for communication becomes particularly critical in dynamic or unpredictable environments, where robots must quickly adapt their plans based on the actions of others. However, communication itself can become a bottleneck, especially in large-scale systems, due to bandwidth limitations, latency, and the risk of communication failures. Addressing these issues often requires robust networking protocols and redundancy mechanisms to ensure system resilience (59).

Another critical aspect is collision avoidance, which is essential to maintain safety and operational integrity in densely populated environments. Unlike single-agent systems, where the robot only has to consider static and dynamic obstacles, multi-agent systems introduce additional complexity due to the presence of other robots that are also actively navigating the space. Each robot must predict the trajectories of the others and adjust its path accordingly. This requires sophisticated algorithms capable of real-time decision making, often using models such as Velocity Obstacles (VO)(60) or Reciprocal Velocity Obstacles (RVO)(61). These models allow robots to calculate feasible velocities that ensure collision-free movement while maintaining progress towards their goals. In more complex scenarios where the environment is highly dynamic, machine learning approaches are increasingly being used to enable robots to learn collision avoidance behaviours through simulated or real-world experience (62).

A key feature of multi-agent navigation is the focus on cooperative task allocation. In

many applications, robots do not simply navigate independently, but are part of a larger system working towards common goals. This cooperation may involve tasks such as collaboratively mapping an environment, transporting heavy or large objects that no single robot can handle alone, or efficiently searching a large area during a rescue mission. Cooperative task allocation requires sophisticated algorithms to distribute tasks among the robots in a way that maximizes overall efficiency and minimizes redundancy. Centralized approaches can optimize task distribution but often struggle with scalability, while decentralized methods rely on local decision making and peer-to-peer communication, offering better scalability but at the potential cost of sub-optimal global performance (63).

Finally, dynamic reconfiguration is a hallmark of multi-agent navigation systems. Unlike static systems, where the roles and positions of agents are predetermined, multi-agent systems often need to adapt to changes in the environment, the tasks at hand, or the availability of individual agents. For example, in a search-and-rescue operation, a robot that encounters an obstacle or loses communication may trigger a reallocation of tasks among the remaining agents. Dynamic reconfiguration ensures the robustness and adaptability of the system, allowing it to continue working in the face of unexpected disruptions.

These capabilities have made multi-agent navigation indispensable in many real-world applications. In warehouse automation, fleets of robots coordinate to move goods efficiently, optimising storage and retrieval processes while avoiding collisions in small spaces. Swarm robotics (see Fig. 1.33), inspired by natural systems such as ant colonies and flocks of birds, uses large groups of simple robots to perform complex tasks through decentralized control and emergent behaviour (64). In search and rescue operations, multiple robots work together to explore and map disaster areas, significantly reducing the time needed to locate survivors or assess damage (65).



Figure 1.33: Flight of the robobee

Overall, multi-agent navigation is not just an extension of single-agent navigation,

but a transforming framework that requires addressing a set of unique challenges while opening up new opportunities for efficiency, scalability, and robustness in robotics.

1.6.2.4 Applications of Mobile Robots

The most common use of rovers is as exploration probes or vehicles for planetary exploration. Historically, rovers have been used both to augment the capabilities of human crews in moving and transporting samples and to perform fully automated exploration operations. In the current scenario of planetary exploration, the latter is the category that boasts the largest number of rovers: this is due to the possibility of integrating increasingly sophisticated scientific instruments into the robot, creating true mobile laboratories that do not require operators, resulting in lower risks and costs. In addition, the lack of a fixed base on the surface to return to greatly increases the range of the probe. To date, the rovers that have operated on the lunar surface are those of the Apollo program, called LRVs (Lunar Roving Vehicles), which allowed crew transportation, and those of the Lunokhod (Fig. 1.34) program, which were automated instead.



Figure 1.34: Rover Lunokhod

Lunokhod 1 was the first remotely operated rover to land on the Moon in November 1970. It consisted of a hermetically sealed tank containing the equipment and a motorized chassis equipped with 8 independent wheels. Each wheel had a diameter of 51 cm and a width of 20 cm. The equipment of the rover included two antennas, four cameras, and an extendable arm for testing the texture and properties of the lunar soil. Also on board were an X-ray spectrometer, an X-ray telescope, and a cosmic ray detector. Power was provided by a battery that was recharged by solar panels during the lunar day. The rover, 135 cm high, 170 cm long, 160 cm wide, with a mass of 840 kg, had a motor and a brake for each wheel. It could move at two different speeds: 1 km/h and 2 km/h.

The Marsokhod rover (Fig. 1.35) was originally designed for robotic missions to Mars, but ended up being used only as a test bed for rover experiments in rough terrain (66). Marsokhod was intended to follow in the tradition of the Lunokhod rover, which

successfully travelled long distances to the Moon. It consists of an articulated frame divided into three parts connected by rotary joints; this gives the Marsokhod rover the ability to walk nimbly over rough terrain. Thanks to the hinges of the frame and a special drive to bend or lift the frame sections, it is able to overcome obstacles whose height is twice the diameter of the wheels. The sections can move alternately, allowing the rover to crawl over fragile terrain at angles of up to 35 degrees. The mass of the Marsokhod is 360 to 450 kilograms, and the diameter of the wheels is 500 millimeters.



Figure 1.35: Rover Marskhod 1 (66)

Sojourner (Fig. 1.6) was the first rover used in the exploration of Mars (67). It was a small remotely operated vehicle that in December 1996 explored the soil of the "Red Planet" by driving a few meters around the landing site to study the rocks there, following a procedure very similar to that used only by the Lunokhod rover on the Moon 30 years earlier. The Sojourner rover was the first to use the rocker-bogie suspension, designed to keep all six wheels in contact with the ground over rough terrain. The robot is 65 cm long, 30 cm wide, 30 cm high and weighs 10.6 kg. It has an X-ray spectrometer and 3 video cameras that provide stereo images. Each wheel moves independently. The vehicle is equipped with a computer that is capable of executing 100,000 instructions per second, allowing the vehicle to move autonomously on the Martian surface, choosing the most advantageous path to reach a target.

NASA's Perseverance (Fig. 1.36) rover landed on Mars on February 18, 2021, handling structure of Perseverance is also based on the rocker-bogie system (68). This type of suspension, after being used on the Sojourner rover, has since been repurposed and used on all Mars rovers. In fact, it can be found on the Spirit and Opportunity robots of the 2003 Mars Exploration Rover mission, on the Curiosity rover of the 2012 Mars Science Laboratory (MSL) mission, and on the Perseverance rover of the Mars 2020 mission. The Perseverance rover carries seven different sets of instruments, including 23 cameras: nine engineering cameras, seven scientific cameras, and an additional seven cameras for the descent and landing phase on the Martian surface. It is also equipped

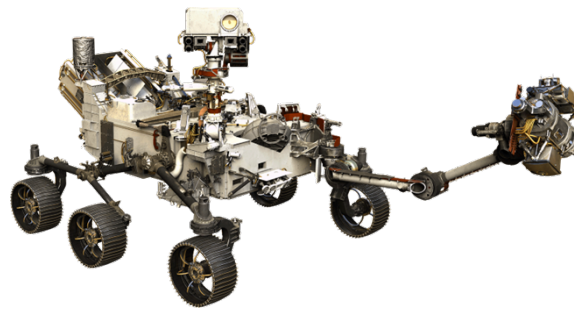


Figure 1.36: Perseverance (57)

with two microphones to record environmental sounds during the descent, landing and operation of the rover on the ground. In recent years, thanks in part to significant incentives provided by governments to encourage the development of automation in industry, numerous research organizations have been involved in the development of rovers and mobile robots for terrestrial applications in industrial, military, and security environments. Rovers have become increasingly common in commercial and industrial environments. Major applications include security, nuclear material handling, surveillance, mine clearance, hazardous area reconnaissance, and agriculture. Mobile robotic systems have been installed in many warehouses (Fig. 1.37) to efficiently move materials from storage shelves to order fulfillment areas.



Figure 1.37: Mobile Robots in Wharehouses

In recent years, the green and ecological topic has gained significant importance. This has led to the development of the first robots designed for the ecological cleaning of open spaces, such as BeBot (Fig. 1.38). BeBot is an innovative robot designed for the ecological cleaning of beaches and other open spaces. Powered by solar panels, it is fully electric and silent, ensuring zero emissions and minimal impact on the local ecosystem. Its main functions include sifting and leveling sand, collecting organic, inorganic and microplastic waste, and being able to carry and move heavy loads of up to 400 kg over sandy and steep terrain. In addition to BeBot, there are other robotic devices dedicated to cleaning beaches and open spaces. These include automated machines that collect solid waste, rake seaweed, and sift sand to remove debris and microplastics. The common goal of



Figure 1.38: BeBot (69)

these rovers is to improve the efficiency of cleanup operations, reduce environmental impact, and ensure safer and cleaner beaches for visitors. The adoption of technologies such as BeBot is an important step towards sustainable coastal management, promoting the protection of the marine environment and providing a better service to citizens and tourists.

1.7 Modeling and Simulation of Mobile Robots

The materials and methods used in the design and simulation of the multibody model (Chapter 2) of the rover will be described here. The importance of dynamic analysis of complex systems through decomposition into rigid bodies, obtained as CAD models, and constraints, performed using advanced software tools such as Matlab® Simulink and Simscape Multibody™, is illustrated. The section goes into detail on how modeling allows for understanding the interactions between components and predicting system performance under realistic operating conditions. This multiphysics approach is critical to design development and optimization, providing essential data for validation and improvement of the final design.

1.7.1 Computer-Aided Design (CAD)

Computer-Aided Design (CAD) systems were developed to provide designers with advanced tools to automate the complex design process. Over time, two main approaches have emerged: computer-aided design systems, aimed primarily at drafting technical drawings, and design-oriented CAD systems that support different stages of the product development cycle. The latter enable activities such as conceptualization, parametric modeling, and structural analysis. CAD models created during the design phase are later used in simulation, particularly in the Simscape Multibody environment, for detailed analysis of system behavior, optimizing the functional verification process and enabling accurate predictions of the structural and dynamic performance of designed components.

1.7.1.1 CAD System

Computer-aided design (CAD) systems were developed with the intention of providing designers with the necessary tools to automate the complex design process. Nevertheless, this objective proved to be more challenging than initially anticipated. Despite years of evolution, CAD systems continue to support and/or automate only a portion of the numerous intricate design tasks. A common feature of all CAD systems is their role in assisting the design phase, which is more effectively documented through technical drawings and more readily modelled with available technologies and computing resources. Other design activities are supported by only a subset of CAD systems and typically require greater computational resources.

CAD systems can be classified into two principal categories. Computer-Aided Drafting (CAD) and Computer-Aided Design (CAD) are two distinct but related fields. The former are functionally analogous to traditional drafting machines, with a primary focus on drafting tasks. Such systems enable the creation of drawings by automating repetitive steps, enhancing quality, standardizing output, and integrating information technologies for classification and technical document management. An illustrative example of this category is AutoCAD (AutoDesk).

In contrast, Computer-Aided Design (CAD) systems are capable of supporting multiple phases of the design process, including activities such as conceptualization, sketching, and preliminary sizing, as well as more advanced stages such as functional verification, detailed definition, parametric modeling, structural analysis, and production process design. Examples of systems in this category include SolidWorks (used in this thesis work), Unigraphics NX, Pro-Engineer, and Catia. CAD systems intended for two-dimensional applications are predominantly drafting tools, as the modeling of functional, structural, and physical characteristics of components typically necessitates the use of three-dimensional geometric models. However, in certain instances, such as those involving parts produced by shearing and bending sheet metal, a two-dimensional representation is sufficient.

1.7.1.2 CAD Model

A CAD model represents a real object, limited to those aspects relevant to the chosen activity that can be represented by a computer at a reasonable cost and time. Typically, these aspects are primarily geometric, with additional consideration given to functional structure and production-related information. For example, while a two-dimensional model cannot automatically provide mass, volume, or center of gravity properties, these properties are easily obtained from three-dimensional models.

The types of models used in CAD systems reflect the classification of the systems themselves: drafting-oriented systems focus on two-dimensional graphical representations, while design-oriented systems emphasize the spatial geometry of the object and treat its two-dimensional graphical representation as secondary. As a result, CAD models generally fall into two categories: Document Models and Object Models.

Document models refer to paper documents that describe a part using engineering drawing conventions. These models represent entities such as lines, arcs, cones, and dimensions, and allow extraction of two-dimensional geometry information such as areas, segment lengths, angles, and distances. Object Models include varying levels of realism, integrating both geometric, functional, and technological information into a representation of the object's solid form. Although these models can comprehensively describe an object's shape, they often contain information in a highly implicit and condensed form, making them difficult to interpret directly in CAD systems. Object models contain varying degrees of realism, integrating geometric, functional, and technological information into a representation of the solid form of the object. Although these models can comprehensively describe the shape of an object, they often contain information in a highly implicit and condensed form, making them difficult to interpret directly in CAD systems.

CAD systems provide a set of commands that differ in user interface and naming conventions from system to system, but share core functionality. Key features include commands for drawing and editing, structuring and managing recurring parts, querying drawings, automating repetitive tasks, customizing environments, and managing digital archives. Elementary graphics commands let you draw basic shapes such as lines, arcs, and circles. Advanced drawing commands make it easy to quickly create complex structures such as regular polygons, perpendicular or parallel lines, fillets, and dimensioning. Di-

menting commands are designed to be easy to use yet highly customizable, supporting standards and user preferences. Advanced dimensioning can even interactively adjust component lengths, increasing precision and adaptability to specific design requirements.

1.7.2 Multibody Dynamics

A Multibody dynamics system consists of a set of solid bodies (rigid or elastic) connected by joints that restrict their relative motions. Multibody dynamics analysis focuses on the motion of these systems under the effect of generalized forces or in identifying the forces necessary to determine a certain motion of the system. The importance of dynamics analysis stems from the need to understand how the moving parts of a system, such as rovers, cars or aircraft, interact with each other and the environment. These complex mechanical assemblies require detailed system-level dynamic analysis. Multibody modeling, carried out through numerical simulations, decomposes the system into bodies, joints, constraints and other components and represents them in a dendrogram structure using specialized software with graphical interfaces. These programs allow two types of dynamic analysis to be conducted: solving direct dynamic problems, where movements are determined from applied forces, and inverse dynamic problems, where the required forces are calculated from observed movements. For accurate results, it is crucial to take a multiphysics approach to modeling, evaluating the impact of different disciplines such as electromagnetism, thermal, fluid dynamics and chemistry. This integrated approach is vital for accurately predicting product performance in real-world scenarios. The work in this thesis uses Matlab® Simulink software and the Simscape toolbox to build a virtual simulated model. This model allows visualization of 3D animations of the system in motion and numerical analysis of the data generated by the simulations, facilitating understanding of the behaviour of the system and verification of expected performance. Numerical data analysis remains critical to validate the correct mechanical operation of the proposed design.

1.7.3 Main Physics of Multibody Software

In the thesis work, the Multibody model was implemented using Matlab® Simulink and the specific toolbox Simscape Multibody™ (formerly known as SimMechanics'). This toolbox was chosen for its capabilities to formulate and solve equations of motion of complex mechanical systems, providing a robust and integrated computational environment. Key features of Simscape Multibody that motivated its selection include:

- **Modularity:** The environment supports the application of block algebra, typical of Matlab® Simulink, facilitating multibody modeling.
- **Integration with Matlab®/Simulink:** This integration enables extensive parameterization of models, making it easy to interact with other Simulink and Matlab® functions.

- Efficient handling of complex systems: Simscape Multibody handles complex multi-body systems with both algorithmic and graphical efficiency.
- Numerical efficiency: The toolbox provides all the advanced computational tools normally found in Matlab®/Simulink, ensuring accurate and detailed analysis.

These features make Simscape Multibody a powerful and flexible tool for the simulation and analysis of dynamic mechanical systems, effectively supporting the development of the thesis project.

As mentioned, the approach is a multiphysics approach. In addition to making the mechanical components by connecting them through joints there are three main physics that have been modeled: the contact and the electric motor.

1.7.3.1 Contact

One of the most important aspects to consider is certainly how our mobile robot interacts with the ground. This is crucial, as it can significantly influence the behavior of the rover. Contact modeling is very useful in the multibody environment and is essential for simulating the interaction between the wheel and the tire. At the contact points, a tangent plane can be defined: this one is simultaneously tangent the surfaces of the two bodies. In relation to this plane, forces can be distinguished: the Normal Force, the Damping Force, which acts perpendicularly, and the Friction Force, which acts along the tangential direction, as it is possible to see in Fig. 1.39 (70). Contact forces are determined using the

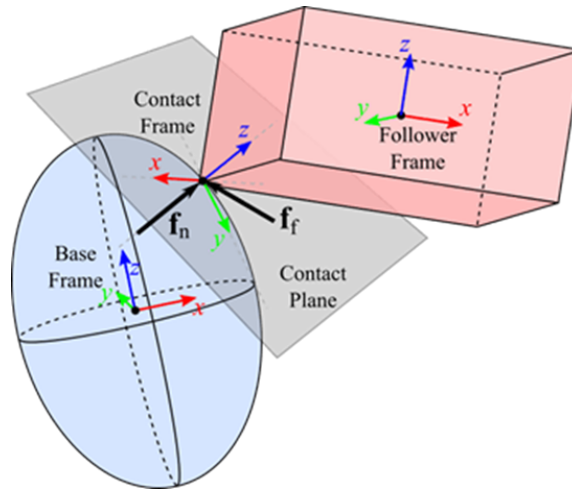


Figure 1.39: Contact Force in the Simscape Multibody Model (70)

Penalty Method, which allows solid bodies to slightly penetrate each other to calculate contact forces using the equation of a classic spring-damper system. The normal contact force, according to the Hooke's Law, is:

$$F_n = k \cdot \delta \quad (1.17)$$

where k is the stiffness coefficient of the contact and δ is the penetration depth. While the Damping force is:

$$F_d = -c \cdot \dot{\delta} \quad (1.18)$$

where c is the damping coefficient and $\dot{\delta}$ is the rate of change of penetration depth. This force provides dissipation of energy, simulating the effects of friction and material inelasticity during contact.

Once the normal force is determined, the friction component can be calculated using Coulomb model:

$$F_a = \mu(V_r)F_n \quad (1.19)$$

where F_a is the friction force, F_n is the normal force and $\mu(V_r)$ if the friction coefficient which is function of the relative tangential velocity V_r . The trend of the friction coefficient is shown in the Fig. 1.40. This friction coefficient depends on three constants: the static

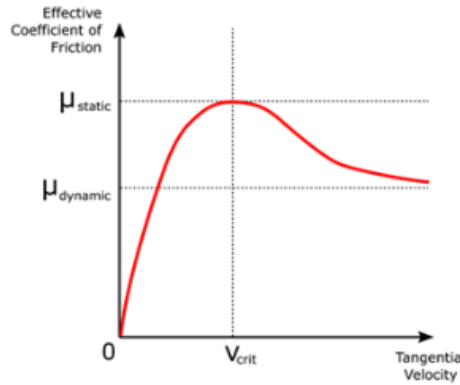


Figure 1.40: Variation of the friction coefficient with tangential velocity (70)

friction coefficient, the dynamic friction coefficient, and the critical velocity. In the Coulomb model to define the friction force is used just the normal force because friction is fundamentally a force resisting relative tangential motion due to the interaction of surfaces in normal contact. The damping force, while it affects the overall dynamics by stabilizing the system and reducing oscillations, doesn't directly contribute to the tangential resistance created by the surface interactions (71)(72)(73).

In practice, the forces generated at the point of wheel-floor contact are dependent upon a number of factors, including the design and components of the mechanical suspension system. Typically, the suspension of a vehicle is composed of multiple components that connect the wheels to the chassis. These components can be divided into three primary categories: structural, elastic, and damping. The structural component can be effectively modeled using multibody techniques. On the other hand, the elastic and damping components operate collectively as a unified system constituted by a spring and a damper, both of which are constrained to the chassis and the structural element of the suspension. This configuration results in the generation of a pair of equal and opposite forces and moments between the two bodies, which act upon the binding points. In the

most basic case, the forces generated are directed along the line joining the two constraint points in opposite directions. The intensity of these forces, F , can be expressed through the following linear equation:

$$F = (x_i - x)k + \sigma \dot{x} \quad (1.20)$$

Where x_i represents both the free extension length and the distance between the constraints, while k and σ denote the spring constant and the damping coefficient of the suspension, respectively. In Simulink, there is a specific block that implements this formulation, called *Spring and Damper Force*.

1.7.3.2 DC Motor

In order to design an effective mechatronic system, it is important to model the power electronics, with a particular focus on motors, since they are the primary element linking the multibody environment with the applied physics. Next, it will be explored the physics of the DC motors used in the rover discussed in this thesis, both the small and big prototype. This type of motor, Fig. 1.41, will be used throughout the development of the rover.

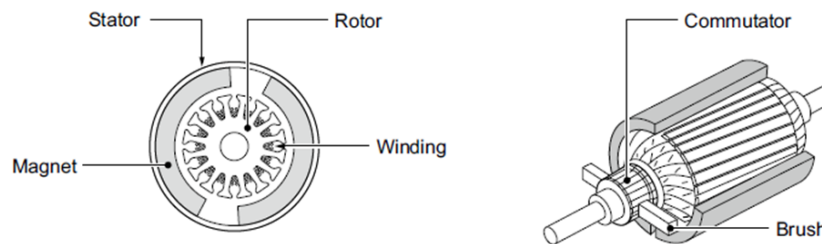


Figure 1.41: Structural components of an electric motor, including the stator, rotor, winding, commutator, and brushes(74)

In summary, the conversion of electrical power to mechanical power in a DC motor is based on the interaction between a magnetic field and an electric current (Fig. 1.42).

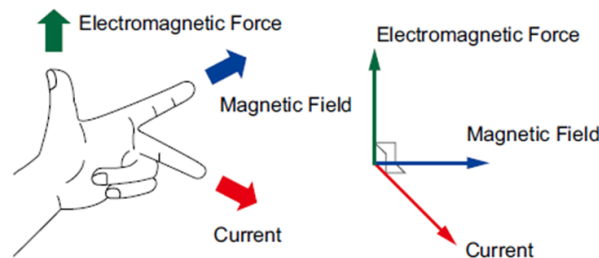


Figure 1.42: Right-hand rule showing directions of force, magnetic field, and current (74)

In the DC motor, the current passes through rotor coils that are located in a constant magnetic field created by electromagnets or permanent magnets in the stator (Fig. 1.43).

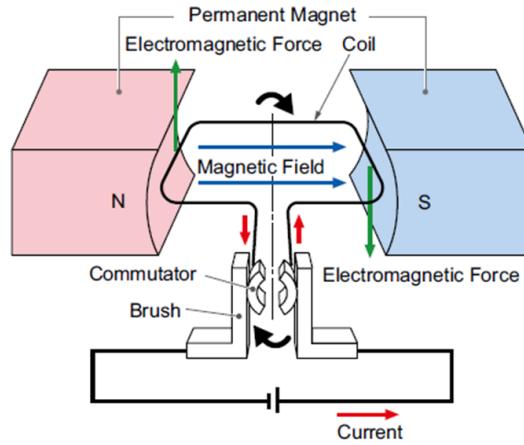


Figure 1.43: Diagram of an electric motor showing the interaction of permanent magnets, electromagnetic forces, and current flow (74)

This interaction gives rise to a force, known as the Lorentz force, which is perpendicular to both the direction of the magnetic field and the direction of current flow. This principle is fundamental to the operation of DC electric motors.

$$\vec{F} = I\vec{L} \times \vec{B} \quad (1.21)$$

Where F is the Lorentz force, I is the current flowing through the conductor, L is the length of the conductor within the magnetic field, and B is the magnetic field itself. This force acts on the rotor coils, causing the rotor to rotate and converting electrical energy into mechanical energy. The torque generated is calculated as the component of the Lorentz force perpendicular to the radius of the rotor multiplied by the distance from the center of rotation. This relationship is expressed in equation 1.22:

$$\tau = \vec{r} \times \vec{F}_{\perp} \quad (1.22)$$

The mechanical power can be calculated by multiplying the generated torque by the angular velocity of the rotor, represented by the symbol ω :

$$P = \tau\omega \quad (1.23)$$

An equivalent electrical circuit may also be used to represent the direct current (DC) motor, as illustrated in the Fig.1.44.

As illustrated in Fig. 1.44, it is evident that the system representing a DC motor is a nonlinear dynamic system. Nevertheless, it is essential to take into account certain limitations pertaining to the actual motor.

1. The magnetic circuit is assumed to be linear, although this is merely an assumption, given that the metal surface may not be perfectly smooth. If the surface were not

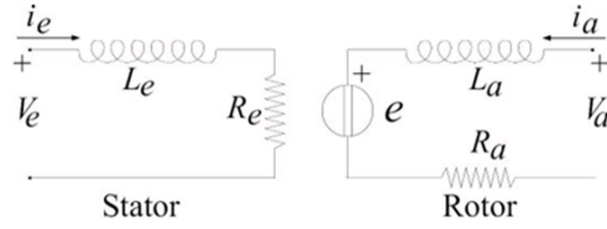


Figure 1.44: Equivalent circuit of a DC motor

perfectly smooth, magnetic flux leakage could occur inside the motor. Additionally, the magnetic flux equation is not valid for high values of the current i .

2. It is assumed that mechanical friction is uniquely linear during motor rotation, that is, there is only viscous friction. This assumption is based on the fact that friction is estimated through the Coulomb model.

It is assumed that the motor stator has a single coil, which has an inductance L_e attributable to the windings and a resistance R_e due to losses in the conductor, as illustrated in the Fig.1.44:

$$v_e(t) = L_e \frac{di_e}{dt} + R_e i_e \quad (1.24)$$

In considering the rotor, it is assumed that it has a single coil with inductance L_a and resistance R_a . Furthermore, the back electromotive force generated by the motor is also taken into account:

$$v_a(t) = L_a \frac{di_a}{dt} + R_a i_a + e \quad (1.25)$$

where the back EFM is:

$$e = K_\Phi \Phi \omega \quad (1.26)$$

and K_Φ :

$$K_\Phi = \frac{ld}{A} \quad (1.27)$$

with l represents the length of the solenoid, d denotes the length of an edge of the square, and A is the surface of flux application.

From the mechanical point of view, the Eq. 1.22 can be written as:

$$T = i l d |\vec{B}| \quad (1.28)$$

but considering that:

$$\Phi = |\vec{B}| A \cos \beta \quad (1.29)$$

it is possible to write the torque as:

$$T = K_\Phi \Phi i \quad (1.30)$$

Considering the elements described, it can be analyzed the torque exerted by the motor together with the reverse electromotive force generated during its operation:

$$\begin{aligned} T_M &= K_\Phi \Phi i_a, \\ e &= K_\Phi \Phi \omega \end{aligned} \quad (1.31)$$

Given that the magnetic flux is generated by the coils on the stator, it can be stated that the flux is proportional to the current flowing through these windings:

$$\Phi = K_0 N i \quad (1.32)$$

where N are the number of windings and:

$$K_0 := \frac{\mu A}{l} \quad (1.33)$$

with μ is the magnetic permeability of the dielectric inside the solenoid. In this way the eq. 1.31 can be rewritten as:

$$T_M = K i_e i_a e = K i_e \omega \quad (1.34)$$

with:

$$K := K_\Phi K_0 N \quad (1.35)$$

This analysis is concluded with a mechanical representation of the motor. When voltages are applied to the stator and rotor, the motor generates torque. This torque interacts with the mechanical structure of the motor, which is characterized by the rotor inertia J and the viscous friction coefficient F . It is also crucial to acknowledge that in each operational setting, a load torque is exerted on the motor. Consequently, if T_L represents the load torque, the equation of motion can be expressed as follows:

$$T_M - T_L = J \frac{d\omega}{dt} + F\omega \quad (1.36)$$

In robotics and other applications, when the load requires a significantly slower speed than the nominal speed of the motor, gears are employed to reduce the angular velocity of the load by a factor of n . The introduction of gears not only increases damping and inertia due to the addition of rotating components but also modifies the mechanical coupling between the motor and the load. It is crucial to acknowledge that the effects of damping and inertia exhibit distinct characteristics when measured at the input or output of the gear. A comprehensive examination of the motor system necessitates an emphasis on the output characteristics, particularly the internal damping (F_G) and the internal inertia (J_G) of the gear.

As the power applied by the motor is maintained at both the input and output of the gear, and with the understanding that T_M and ω represent the torque and speed at the output of the gear system, this relationship is sustained throughout the gearing system:

$$T_M \omega = T'_M \omega' \quad (1.37)$$

since, the gear ratio:

$$\omega' = \frac{\omega}{n} \quad (1.38)$$

then:

$$T'_M = nT_m \quad (1.39)$$

Substituting the eq. 1.39 into eq. 1.36 and considering the increase in damping and inertia due to the gears in the gearbox, it results that:

$$T'_M - T_L = (J_G + n^2 J) \frac{d\omega'}{dt} + (F_G + n^2 F) \omega'. \quad (1.40)$$

The comparison between eq. 1.40 and 1.36 highlights that the presence of the gear significantly increases both the inertia and the damping of the motor as perceived by the load.

The introduction of a gear into an actuator system results in *backlash* at the output, caused by the interaction between the cogwheels of the gear and the resulting non-linearities, which may lead to instability. Consequently, high-precision systems often employ direct-drive motors, which are capable of generating sufficient torque at low speeds without the necessity for gears. However, these motors are not well-suited for high-power tasks due to the physical limitations on the maximum torque they are capable of delivering(75).

In the Simulink environment, a Simscape block called *DC Motor* is available, which accurately simulates the behavior of the electric motor previously described. This block also allows the integration of a gearbox.

1.8 Communication and Signal Processing

Signal processing and inter-device communication play a very important role in robotic systems, enabling the acquisition and interpretation of data essential for control and navigation. In this context, the conversion of analogue to digital signals through sampling processes is crucial to ensure efficient and accurate processing.

One of the main tools for signal analysis is the Fast Fourier Transform (FFT), which allows a signal to be decomposed into its frequency components, facilitating the identification of dynamic phenomena. Furthermore, in robotic systems, communication between sensors and controllers takes place via protocols such as USB, I2C and CAN, each optimised for specific data transfer requirements and robustness to interference.

In order to improve the accuracy of the data acquired, to ensure efficient information transfer and to optimise the control of the robotic system, both in simulation and in experimental tests, these aspects are very important.

1.8.1 Basic Concepts in Signal Theory

In the context of this thesis work, one of the application that will be seen involve the monitoring of the wheel velocities thanks to the use of a FFT-based algorithm applied to the vibrations obtained by IMUs, both in simulation and experimental tests. The phenomenon under investigation is detected by the IMU and converted into an electrical signal that varies over time, resulting in an analog signal. Subsequently, the data is digitized through sampling, enabling storage in a computer. In a simulation, this process is purely theoretical and does not account for real-world effects such as electrical and magnetic interference, which are common in experimental testing. As a result, data obtained from simulations is typically of a higher quality than that obtained from experiments, which is an important consideration when analysing the results of simulations.

In both real-world tests and simulations, the final output is a discrete signal. This signifies that the signal, regardless of whether it is a result of actual measurements or simulations, is represented as a sequence of numerical values (samples). The samples correspond to the values of the signal at specific points in time, as defined by the selected sampling rate or time interval.

For a generic signal $S(t)$ in Fig.1.45, each sample can be represented as a value for a defined t , as can be viewed in Tab.1.1.

Given that $\Delta(t)$ represents the sampling interval and $1/\Delta(t)$ denotes the sampling frequency.

The following steps should be taken when analyzing a signal:

- Acquisition: The acquisition phase employs particular instrumentation with the objective of transforming the signal from an analog to a digital format. It is essential to recognize that this process is exclusively applicable to real-world scenarios, as in simulative contexts, the signal is inherently digital.

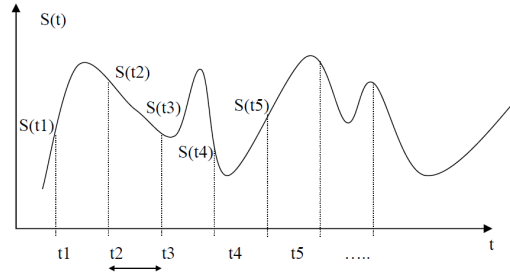


Figure 1.45: Representation of a generic signal $S(t)$ over time with discrete sampling points

Time	Signal
$t1$	$S(t1)$
$t2 = t1 + \Delta t$	$S(t2)$
$t3 = t1 + 2\Delta t$	$S(t3)$
$t4 = t1 + 3\Delta t$	$S(t4)$
$t5 = t1 + 4\Delta t$	$S(t5)$
...	...

Table 1.1: Time intervals and corresponding signal values for $S(t)$

- **Analysis:** In this phase, the acquired signal is subjected to processing in order to manipulate it and extract pertinent information. This may entail operations such as filtering, decomposition, or amplification of the signal.
- **Interpretation:** The final step is to correlate the extracted information with the observed physical phenomenon. At this juncture, an effort is made to comprehend and contextualize the data in relation to the phenomenon under study, thereby enabling the formulation of valid conclusions or hypotheses for subsequent investigation.

It is critical to understand that the three fundamental stages of signal analysis—acquisition, analysis, and interpretation—are essential in both real-world scenarios and simulations. However, the specific processes for each stage differ due to the distinct nature of the input signals. As previously stated, real-world signals are affected by a variety of noise factors, which may be absent or reduced in a simulation. This distinction suggests that, while the simulated signal can be managed in a more controlled and predictable manner, the analysis of the actual signal requires the implementation of additional strategies to effectively handle and filter out noise.

The instruments employed for signal acquisition in actual experimental procedures will be examined in detail when the experimental tests are analyzed. The process of acquisition in the simulative phase is relatively straightforward. As the model is simulated in Simulink, the utilization of specific sensory blocks is sufficient. For example, the *InertiaSensor* block facilitates the identification of inertial characteristics within the

simulation environment.

The *Inertia Sensor* block is responsible for measuring the inertial properties of bodies within the model to which it is connected. This block is capable of detecting a number of parameters that are fundamental to the analysis of body dynamics, including:

- Mass: This block is used to measure the weight of the body.
- Center of mass: This indicates the point at which the entirety of the mass of the body can be considered to be concentrated for the purposes of motion analysis.
- Inertia matrix: It provides values that describe the distribution of the mass of the body in relation to its principal axes.
- Centered inertia matrix: This is a specific form of the inertia matrix that assumes the center of mass as the origin.
- Principal inertia matrix: A streamlined variant of the inertia matrix that considers solely the principal axes of inertia.
- Orientation of the principal inertia frame: This defines the orientation of the principal inertia axes in relation to the global reference system.

The collected signals were analyzed using the MATLAB® software. Within this platform, signal-processing scripts were developed to generate the graphs presented in the following sections. The plots resulting from the analysis phase serve as fundamental tools for data interpretation. It is of the greatest importance to give each of these steps the attention they deserve in order to gain a thorough understanding of the dynamics of the rover system during testing.

1.8.1.1 FFT Theory

In order to provide a comprehensive analysis of both simulative and experimental approaches, it is essential to introduce the key theoretical concepts that are fundamental to this discussion, including the Fast Fourier Transform (FFT). The FFT represents a fundamental technique for signal analysis, allowing the decomposition of a signal into its spectral components and facilitating the acquisition of detailed information on the frequency content of the analyzed signal.

A signal is defined as a physical variable that changes over time and whose value conveys information. Signals are classified into two principal categories:

- Analog Signals: These are typical of natural phenomena and exhibit continuous variations over time.
- Digital Signals: In modern data acquisition systems, analog signals are numerically represented to facilitate processing with digital instruments.

The Fourier transform represents the fundamental mathematical tool for signal analysis, enabling a signal to be transformed from its original domain (typically time) into a frequency domain representation.

The FFT enables the rapid and efficient execution of these transformations, making it an optimal choice for analyzing complex signals in real-time or in applications where critical processing time requirements must be met. The FFT plays a central role in the simulation approach employed in this study, serving as a crucial tool for analyzing vibrations and signals generated by the interactions between the wheels of the rover and the ground.

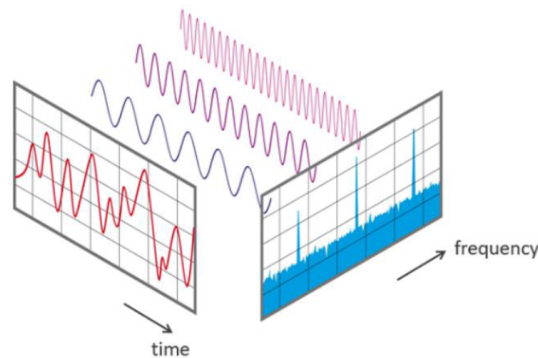


Figure 1.46: Representation of a signal in the time domain and its corresponding frequency domain

A signal is sampled over a specified time interval and analyzed to identify its frequencies and the components that comprise them. Each component is associated with a sinusoidal oscillation at a specific frequency, defined by its unique amplitude and phase. The Fig. 1.46 illustrates that, within the observed period, the signal exhibits three distinct dominant frequencies. This analysis allows for the decomposition of the original signal into simpler elements, facilitating a more detailed examination of its characteristics.

In the initial stage of the process, a segment of the signal is scanned and stored in memory for subsequent analysis. In the process of analyzing a signal through the use of the Fast Fourier Transform (FFT), two parameters of particular significance must be considered:

- The sampling frequency (denoted as f_s) is defined as follows: This parameter indicates the number of samples collected per second, expressed in samples per second. This parameter defines the frequency range that can be accurately analyzed and influences the time resolution of the sampled signal
- Block length (BL) is defined as: This term refers to the number of samples selected for each segment of the signal to be analyzed by the FFT. It is typically set as a power of two in order to facilitate the calculation of the transformation. This value also represents the points that will compose the bilateral FFT, affecting both the

frequency resolution and the ability to distinguish spectral components within the signal.

A reduction in block length results in a more rapid measurement interval, but a concomitant reduction in frequency resolution. On the other hand, an increase in block length yields a more refined frequency resolution, but at the cost of a slower measurement interval. Additional measurement parameters can be derived from the two fundamental parameters, f_s and BL .

The bandwidth, denoted as f_n , represents the Nyquist frequency. This value represents the theoretical maximum frequency that can be resolved by the FFT:

$$f_n = \frac{f_s}{2} \quad (1.41)$$

The sampling theorem, also known as the Shannon-Nyquist theorem, states that a band-limited signal $x(t)$ with a maximum frequency f_n can be fully reconstructed from its samples if the sampling frequency f_s is greater than twice f_n , that is, $f_s > 2f_n$.

In the case of non-compliance with the aforementioned criterion, resulting in an undersampling of the analog signal in the time domain, aliases will emerge in the frequency domain, thereby distorting the original signal when it is transferred from the frequency domain back to the time domain.

Passive filters fall into three main categories based on the range of frequencies they allow to pass:

- Low-pass: They pass low frequencies and block high frequencies
- High-pass: They pass high frequencies and block low frequencies
- Bandpass: They pass a certain range of intermediate frequencies and block both higher and lower frequencies

In particular, the low-pass filter is used to ensure that frequencies above the Nyquist frequency are suppressed, thus preventing aliasing during the sampling process. As for the measurement duration D , it depends on the sampling frequency f_s and the block length BL . This length determines the amount of data collected and the resolution of the measurement:

$$D = \frac{BL}{f_s} \quad (1.42)$$

while:

$$d_f = \frac{f_s}{BL} \quad (1.43)$$

where d_f is Frequency Resolution: the frequency resolution indicates the frequency spacing between two measurements.

In the implementation, it is assumed that the analyzed signal segment is periodically extended to infinity. This approach leads one to consider some important points:

-
- The FFT is effective for periodic signals: The FFT is particularly useful for analyzing signals that are inherently periodic or can be reasonably approximated as such. The analysis will be more accurate and meaningful if the signal itself supports the assumption of periodicity
 - An integer number of periods in the sampled segment: To minimize representation errors and discontinuity effects at the edges of the segment (a phenomenon known as *leakage*), it is ideal for the sampled signal segment to contain an integer number of signal periods. This helps to ensure that signal characteristics are correctly represented without artifacts introduced by arbitrary cuts in time.

These considerations are critical to the effective use of the FFT in signal analysis in many applications, from signal processing to harmonic analysis (76).

1.8.2 Communication Protocol Definition

This section provides a comprehensive examination of the USB and I²C technologies utilized in the prototype. The USB communication protocol, selected for its convenience and compatibility, permits the transmission of data and the provision of power to the device. The I²C protocol, meanwhile, facilitates efficient communication between the Arduino and external sensors. The two systems, which are distinguished by clearly delineated communication protocols, guarantee the proper functioning and management of data flow within the rover.

1.8.2.1 USB

Universal Serial Bus (USB) is a standardized technology that enables the connection of various devices to computers or to each other for the transfer of data and the provision of power. The USB was developed in the mid-1990s through a collaborative effort by major technology companies, including Intel, Microsoft, and IBM. The objective was to simplify connections between computers and peripheral devices, replacing older interfaces like serial and parallel connectors with a unified standard.

The key features of USB include:

- Hot swapping: The Universal Serial Bus (USB) enables devices to be connected or disconnected without requiring the computer to be powered down.
- Plug-and-Play: Upon connection, devices are configured automatically, facilitated by the implementation of standardized drivers and protocols.
- Backward Compatibility: Newer versions (in Fig.1.47 is possible to see different USB ports) of the Universal Serial Bus (USB) typically support older devices and cables, although this may impact the data transfer speeds.



Figure 1.47: Different types of USB ports and connectors

The USB Implementers Forum (USB-IF) is responsible for maintaining USB protocols, ensuring compatibility, implementing updates, and conducting certification. This encompasses the definition of data encoding methodologies, power management protocols, and device control frameworks.

In order for the transmission of data via USB to occur correctly, it is necessary to set the baud rate. The term "baud rate" is used to describe the rate at which the signal changes per second. This variation per second is referred to as the baud rate, which can be defined as the rate at which the highs and lows must be sampled in order to decode the signal. In the Arduino environment, the user has the option of selecting one of the following baud rates: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, or 57600. Alternatively, the user may select 115200 as the baud rate. Nevertheless, 9600 is the most commonly utilized standard baud rate.

1.8.2.2 I²C

The I²C protocol, an acronym for "Inter-Integrated Circuit," is a widely used communication standard in modern Arduino boards, as well as in other platforms and advanced robotic applications. This protocol enables devices, such as Arduino boards, to establish a connection with external sensors, such as magnetometers, thereby facilitating the collection of data without involving the main processor.

I²C operates via a two-wire bus, comprising the SDA (Serial Data Line) and SCL (Serial Clock Line) lines, along with a common voltage reference. In an I²C network, devices are classified as either masters or slaves. The master device is responsible for controlling the SCL clock line and triggering the start of data transmission, while the slaves are designed to respond to the commands issued by the master. A slave device is unable to initiate communication on the I²C bus; this function is exclusively reserved for the master device. Typically, the I²C bus is capable of supporting multiple slave devices, but only a single master device, which is responsible for managing communication with all connected devices.

In accordance with the I²C hardware protocol, two communication lines are necessary. As the network in Fig. 1.48, the SDA line is utilized for the transfer of data, whereas the SCL line provides clock pulses, distinguishing I²C as a synchronous bus due to the presence of the SCL signal. The SCL and SDA lines of the I²C bus are shared among all



Figure 1.48: I²C protocol network showing master-slave communication over SDA and SCL lines (77)

connected devices, and a third wire for ground (0 volts) completes the circuit, as reference Fig.1.49. This configuration represents a typical I²C bus configuration.

In a manner distinct from other buses, the I²C bus employs open-collector (for TTL logic devices) or open-drain (for CMOS logic devices) lines, rather than tri-state outputs. This configuration permits the sharing of communication lines by multiple devices without conflict, provided that no device attempts to drive the line to a high state while another device pulls it to ground.

In order to guarantee that the SCL and SDA lines remain in a high state when idle, pull-up resistors are a necessity. These resistors establish a connection between each line (SCL and SDA) and a positive voltage source, typically 5V. The specific values of the pull-up resistors are dependent upon the particular implementation and the length of the bus. However, values within the range of a few $k\Omega$ are commonly used. The functionality of the I²C bus is dependent upon the presence of pull-up resistors. In order to guarantee the optimal functioning of the I²C bus, it is imperative that pull-up resistors are installed on the SCL and SDA lines. In the absence of these resistors, the lines will remain at a low level (near 0 volts), which will impede the functionality of the bus. The appropriate resistor values are dependent upon a number of factors, including the supply voltage, the capacitance of the bus, and the number of connected devices. In the majority of cases, resistor values of approximately $2\text{ k}\Omega$ are sufficient for the majority of applications. Nevertheless, the utilization of lower resistor values may result in the flow of excessive currents, particularly in the context of 5 V supplies. On the other hand, the deployment of higher resistor values may give rise to transmission complications on buses characterized by elevated capacitance. The standard I²C bus voltage is 3.3 V or 5 V; however, varying voltages may be employed for different devices on the bus, necessitating the use of level shifter circuits for devices that are incompatible with the available voltage. These components guarantee that the lines revert to a high state when no device is transmitting, ensuring stable and reliable communication between master and slave devices.

In standard mode, the I²C bus clock frequency ranges from 0 to 100 kHz. The introduction of fast mode in 1992 resulted in an increase in the maximum frequency to 400 kHz. The introduction of high-speed mode in 1998 saw a further increase in the maximum

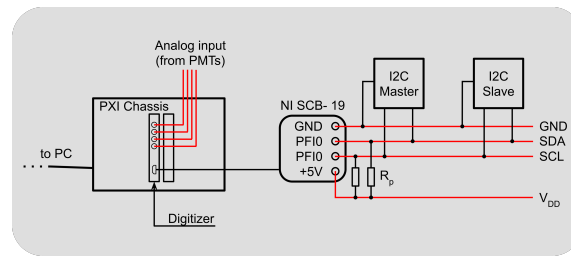


Figure 1.49: Example of the I²C wiring diagram illustrating master-slave connections (78)

frequency, which was raised to 3.4 MHz. Communication begins with the issuance of a start sequence by the master device. It should be noted that the aforementioned start and stop sequences represent the only instances in which the SDA line may transition to a different state while the SCL line is maintained at a high level. It is of great importance that the SDA line remains stable when SCL is high during the transmission of data. The start and stop sequences delineate the beginning and end of transactions with the slave device, respectively.

In a synchronous I²C data transfer, the data is organized into sequences of eight bits. The order in which bits are placed on the SDA line is specific: the most significant bit (MSB) is transmitted first, followed by the subsequent bits in descending significance. The SCL line alternates between high and low states, thereby synchronizing the transmission of each bit. Following the transfer of each 8-bit byte, the receiving device generates an acknowledgment bit (ACK), resulting in nine SCL clock pulses per data byte—eight for the data bits and one for the acknowledgment bit.

In the case of a low acknowledgment bit (ACK) being transmitted by the receiving device, this indicates that the data has been successfully received and that the receiving device is prepared to accept the next byte. Conversely, if the receiving device returns a high bit (NACK), it signals that it is unable to receive further data, which prompts the master to terminate the transfer by issuing a stop sequence. This process allows for the effective regulation of data flow between the master and slave devices, thereby ensuring the integrity of communication on the I²C bus.

1.8.2.3 CAN Protocol

The Controller Area Network (CAN) is a serial communication protocol developed in the 1980s by Robert Bosch GmbH to facilitate interconnection between different electronic control units (ECUs) in vehicles. Designed to operate effectively in high electromagnetic noise environments, CAN uses a balanced differential transmission line, often implemented with twisted pair cables, to ensure high noise immunity. This protocol was standardised as ISO 11898-1 in 1993.

CAN is a serial communication system that allows microcontrollers and devices to communicate with each other without the need for a host computer. It uses a bus topology in which all devices are connected to a single shared data cable, enabling efficient peer-to-peer communication. Data is transmitted over a pair of twisted wires called CAN High

and CAN Low, which carry differential signals to improve resistance to electromagnetic interference (79).

In terms of CAN protocol integration, the SCOUT MINI is equipped with a CAN 2.0B interface with a baud rate of 500 kbps, compliant with the Motorola message format. This interface allows secondary development through chassis motion control, allowing the user to send commands to adjust the linear and angular speed of the robot. In addition, the Scout Mini provides feedback on current motion status and other diagnostic information via the CAN bus, facilitating integration with external systems and expansion of robot functionality (80).

–2–

Integrated Design, Modelling, and Simulation of Rover Systems

CONTENTS: **2.1 Introduction.** **2.2 Rocker-Bogie Suspension Analysis.** 2.2.1 Fundamental Principles of the Rocker-Bogie Suspension – 2.2.2 Obstacle Crossing: Mechanics and Dynamics – 2.2.3 Advantages and Disadvantages of the Rocker-Bogie System – 2.2.4 Kinematics Analysis of Suspension – 2.2.5 Load Distribution Analysis – 2.2.6 Conclusions. **2.3 Small Rover Prototype.** 2.3.1 Multibody Prototype – 2.3.2 Rover Movement Analysis and Simulations – 2.3.3 Vibration Analysis in Straight-Forward and Steering Configurations – 2.3.4 Integration of Robotic Arm for Obstacle Crossing. **2.4 Full-Scale Rover Prototype.** 2.4.1 First Prototype – 2.4.2 Second Prototype Development and Analysis – 2.4.3 Conclusions.

2.1 Introduction

This chapter explores the study and development process that led to the creation of the first prototype of the rover. The path towards the realisation of this system is characterised by an integration of design, modelling and simulation. This integrated methodology not only allows the progressive refinement of the project, but also results in a final solution that reflects the initial objectives and the operational requirements imposed by the operating environment of the rover. In detail, this chapter describes how each step of the process contributed to defining and refining the characteristics of the rover, starting from a preliminary concept to the creation of a working digital twin. It will cover the different stages involved, including suspension analysis, load distribution, kinematic evaluation and simulation-driven optimisation. Attention is given to how simulations were used to progressively improve the designed rover at each stage of development. Additionally, this chapter introduces the small prototype that served as an experimental platform for testing key hypotheses and refining design principles before implementation in the full-scale rover. In fact, a digital twin of the small rover was developed over the years. Although a separate object, this smaller model played an important role in the development process, serving as a test bed to validate hypotheses about how it would work and to gain a deeper understanding of the dynamics and potential difficulties of the bigger one. The knowledge gained from the study of this compact version influenced the design of the bigger model and further helped to refine the final prototype. Each step brought new challenges and solutions, highlighting the need for adaptive strategies in the design of complex systems. By analyzing these phases, the chapter aims to provide a structured overview of the methodologies employed, linking theoretical concepts to practical implementations. By examining these cycles, it is intended to show the crucial role of adaptability and iterative testing in the field of robotics. This introduction outlines the path taken to arrive at the definition of the rover, highlighting the essential role of theory and simulation in overcoming technical challenges and achieving project goals. Through this structured approach, the chapter also offers insights into future research directions and improvements in mobile robotic system design.

2.2 Rocker-Bogie Suspension Analysis

The analysis presented in references (81) and (82) revealed that, in the context of navigating rough terrain at moderate speeds, the rocker-bogie suspension configuration proved to be the most advantageous. This section will introduce the suspension system under consideration, beginning with a general overview and subsequently presenting a kinematic analysis (83).

2.2.1 Fundamental Principles of the Rocker-Bogie Suspension

The Rocker-Bogie suspension system is an articulated, passive configuration developed by NASA to provide stability for six-wheeled rovers over rough terrain, particularly effective at low forward velocity. This system was first introduced with the Sojourner rover. Unlike traditional elastic suspension systems, which are efficient at higher speeds, the Rocker-Bogie was chosen for its simplicity of construction and adaptability to the slow movements typical of planetary-use rovers. The articulated design of the Rocker-Bogie allows all wheels to actively participate in supporting the vertical load. In a simplified example, a two-dimensional rigid bogie with three wheels on uneven ground is not statically determined; one of the wheels may not touch the ground, compromising the load support and mobility of the rover. The Rocker-Bogie suspension, on the other hand, ensures that all wheels maintain contact with the ground, and each wheel can be driven independently by six separate motors. The geometry of the suspension is designed to distribute the weight of the vehicle equally over all the wheels. The wheels are connected to the chassis through movable links, arranged symmetrically on both sides of the rover in two 'banks'. Each bank consists of three wheels, connected by two links; the main link is called the 'Rocker,' featuring two rotational joints. The first joint connects the Rocker to the rover chassis and, through a differential system, coordinates the movement of the links of the two banks. The second rocker joint, on the opposite end, is connected to the second movable link called the Bogie. On the opposite end of the rocker a wheel is mounted. On the Bogie link, meanwhile, the other two wheels are connected symmetrically, as seen in Fig. 2.1.

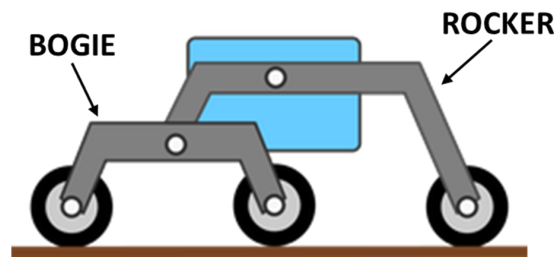


Figure 2.1: Rocker-bogie suspension system of a mobile robot

The wheels are mounted on a frame consisting of the chassis and two oscillating links, which allows the system to keep all six wheels in contact with the ground, even when driving over uneven and bumpy surfaces. The classic Rocker-Bogie design does not include springs or dampers, allowing the rover to overcome obstacles while maintaining constant ground contact between all six wheels. The two Rockers are connected to the chassis by a differential system consisting of a rocker and two brackets. This system ensures that when one rocker lifts up by rotating around its linkage joint with the chassis, the other rocker lowers down. As a result, the pitch angle of the frame corresponds to the average of the pitch angles of the two rockers (84).

For almost all applications, the Bogie is placed in the front because it is able to overcome higher obstacles than a configuration with the Rocker in the front position.

2.2.2 Obstacle Crossing: Mechanics and Dynamics

The use of an articulated suspension system such as the Rocker-Bogie allows the rover to move slowly over rough terrain and climb any obstacles. This slowness is deliberately chosen to allow the vehicle to climb over obstacles as large as the radius of its wheels, reducing the risk of rollover or damage to the suspension system. When the rover approaches an obstacle, regardless of the front configuration (Bogie or Rocker), the front wheels are pushed against the obstacle by the four rear wheels. That action causes the rotation of the front wheels to lift the front of the rover, allowing it to climb the obstacle, as showed in Fig. 2.2. As soon as the front wheels have crossed the obstacle, they allow

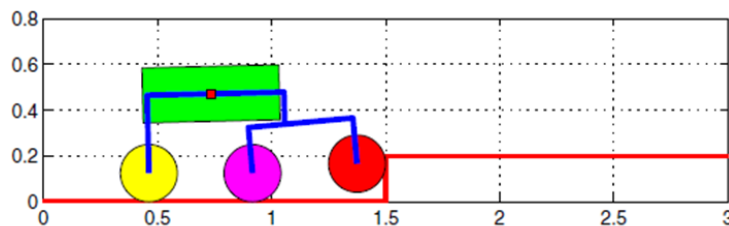


Figure 2.2: Obstacle Crossing: 1st phase (85)

the middle wheels to come into contact with it. The latter, in addition to gaining their own traction, are pushed against the obstacle by the rear wheels and pulled upward by the front wheels as seen in Fig. 2.3. This action continues until the middle wheels are also lifted up and begin to clear the obstacle. Finally, the rear wheels, once arrived against the obstacle, are pulled over it by the four front wheels, as showed in Fig.2.4.

During obstacle climbing, the movement of the rover is slowed or even stopped if the height of the obstacle is equal to or greater than the radius of the wheel. Such slowdown is appreciable only with obstacles of considerable size. For smaller obstacles, no significant slowdown is observed since no real overrun occurs. One benefit is that, at low speeds, the rover stops while the front wheel climbs over the obstacle, minimizing impact and reducing dynamic shocks. However, at higher speeds, such as 1 m/s, the shocks could

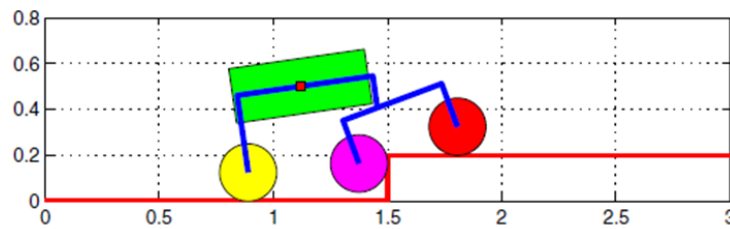


Figure 2.3: Obstacle Crossing: 2nd phase (85)

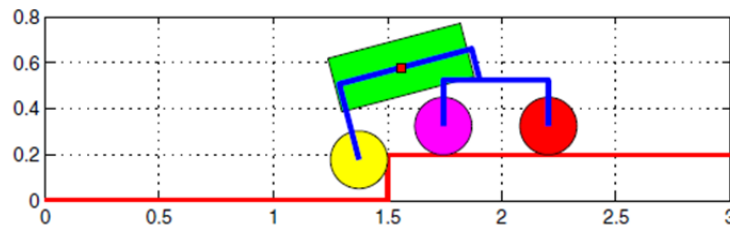


Figure 2.4: Obstacle Crossing: 3rd phase (85)

damage the load, instrumentation, or the rover itself. A solution could be to adopt control strategies that slow the rover down when an obstacle is detected by the vision and sensor systems, thus avoiding violent impacts. This approach would mitigate the risks of impact but would not solve the problem of slowing down. The idea of implementing suspension links, turning the system into a leg-wheel hybrid, could compromise the simplicity and self-adaptability of the Rocker-Bogie design to rough terrain. Therefore, rather than seeing it as a problem, this slowdown should be seen as an intrinsic feature of the suspension system, closely related to its morphology and operational logic.

2.2.3 Advantages and Disadvantages of the Rocker-Bogie System

The Rocker-Bogie passive articulated suspension system shares many advantages with other mobile robots and rovers. This configuration offers additional benefits that make it preferable in certain applications over other options, although it has some disadvantages.

The main advantages of the Rocker-Bogie, summarized in the introduction of this chapter, include:

- The ability to overcome obstacles while maintaining ground contact of all wheels, ensuring consistent traction.
- The almost identical load distribution on all wheels, if the system is sized correctly.
- The ability for the two banks, left and right, to move independently, allowing the rover to overcome different obstacles at the same time.
- The independence of the movements of the two rockers ensures that the tilt of the chassis matches the average tilt between the two, limiting pitching.

-
- The absence of spring-damper systems reduces the risk of rollover.
 - The presence of an independent motor for each wheel, and the absence of springs or drive axles, makes the system safer and simpler.
 - The design minimizes center-body vibration, since wheel oscillations are transmitted to the body as rotations through the differential that connects the two rockers, and not as translations.
 - On uneven terrain, vertical displacements of the frame are less than those experienced by the wheels, improving overall stability.

In sum, the Rocker-Bogie offers a unique combination of advantages that distinguish it from other suspension systems, making it particularly suitable for exploration on complex and uneven surfaces.

Critical issues of the Rocker-Bogie articulated suspension system include:

- Reduced speed and slowing of movement when overcoming obstacles whose height exceeds the wheel radius.
- Variable obstacle-crossing ability depending on the approach, whether initiated with the Rocker or the Bogie.
- Increased wheel wear due to slippage during skid-steering maneuvers, as terrain may offer variable resistance.
- Potential loss of traction of intermediate wheels in the initial stages of approach to vertical obstacles, due to slight weight relief that can cause slippage.

These critical issues underscore the limitations of the system in operations that require steering frequently or climbing over particularly challenging vertical obstacles. Among the disadvantages of the Rocker-Bogie configuration, some are beatable. For example, wheel slip during steering could be mitigated by implementing a steering system that makes the wheels omnidirectional, actuating them with additional electric motors, similar to what was adopted on Perseverance rover of NASA. This solution, however, complicates the system by requiring six additional actuators to enable wheel steering as showed in Fig. 2.5.

However, other disadvantages are inherently related to the morphology of the system, such as, for example, slowing down the forward motion when climbing over obstacles.

2.2.4 Kinematics Analysis of Suspension

The arrangement of the rover over rough terrain required an analysis of the mutual position of each link of the Rocker-Bogie suspension system. Using a simplified two-dimensional model, each significant component of the mechanical system was represented

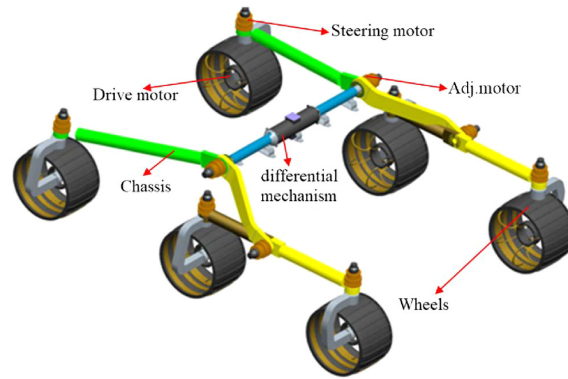


Figure 2.5: Example of Rocker-Bogie system with steerable wheels

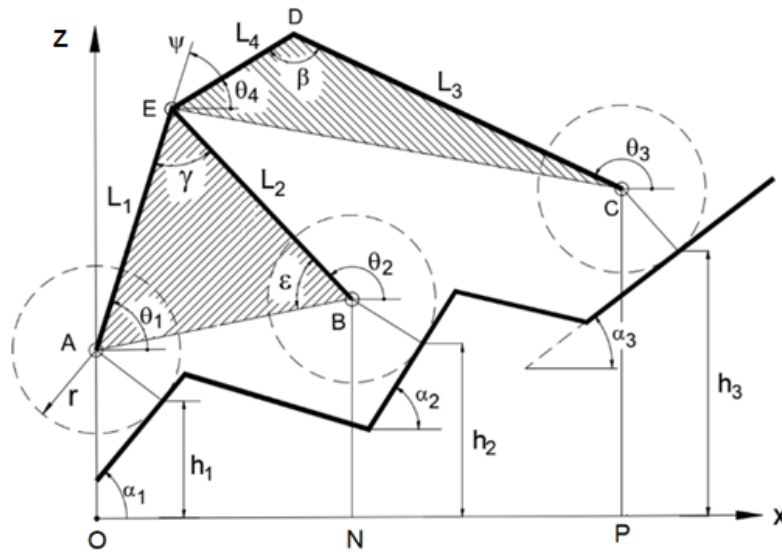


Figure 2.6: Kinematic diagram of the rocker-bogie suspension system

with connected segments, assuming these to be rigid elements. This allowed the kinematic scheme used to study the behavior of the system to be generated.

According to the scheme in Fig. 2.6, the geometry of the suspension and the soil profile are characterized by the following quantities:

- r : wheels radius;
- L_i : suspension links length;
- β e γ : fixed rocker and bogie angles;
- θ_i : angles that define the suspension configuration;
- h_i : elevation of the contact point relative to the horizontal reference x ;

- α_i : the angle between the common tangent and the x-axis at the point of contact.

The angles θ_2 and θ_3 are linear compositions of the angles β , γ , θ_1 and θ_4 , resulting in the following:

$$\begin{aligned}\theta_2 &= \theta_1 + \gamma \\ \theta_3 &= \theta_4 + \beta\end{aligned}\tag{2.1}$$

The relative rotation between Rocker and Bogie is defined in 2.2:

$$\psi = \theta_1 + \theta_4\tag{2.2}$$

By applying trigonometric relations, given the elevations h_i and angles α_i as defined above, the x and y coordinates of the axes of the wheels and joints D and E can be derived. The angles θ_i are obtained by considering the polygons OAEBN and NBEDCP

Point	x-coordinate	y-coordinate
A	$x_A = 0$	$y_A = h_1 + r \cos \alpha_1$
B	$x_B = AB + r \cos(\theta_1 - \varepsilon)$	$y_B = h_2 + r \cos \alpha_2$
E	$x_E = L_1 \cos \theta_1$	$y_E = y_A + L_1 \sin \theta_1$
C	$x_C = x_E + L_4 \cos \theta_4 + L_3 \cos(\pi - \theta_3)$	$y_C = h_3 + r \cos \alpha_3$
D	$x_D = x_E + L_4 \cos \theta_4$	$y_D = y_C + L_3 \sin \theta_3$

Table 2.1: Geometric coordinates of key points in the rocker-bogie mechanism

and performing the polygon closure equation for them, we obtain two equations that projected onto the y -axis give us the system of equations in 2.3:

$$\begin{cases} (h_1 + r \cos \alpha_1) + L_1 \sin \theta_1 = (h_2 + r \cos \alpha_2) + L_2 \sin \theta_2 \\ (h_2 + r \cos \alpha_2) + L_2 \sin \theta_2 + L_4 \sin \theta_4 = (h_3 + r \cos \alpha_3) + L_3 \sin \theta_3 \end{cases}\tag{2.3}$$

The system of equations has θ_2 and θ_4 among the unknowns. If we derive the angle θ_2 from eq. 2.1, it is possible to define the angle θ_4 from the second equation and thus obtain the relative rotation between Rocker and Bogie defined in eq. 2.2, as defined in Fig. 2.7. By substituting the two relations 2.1, using trigonometric sine addition formulas, and using parametric formulas, the eq. 2.3 can be reformulated as linear trigonometric equations, thus succeeding in calculating angles:

$$(L_1 \cos \gamma - L_2) \sin \theta_2 + (-L_1 \sin \gamma) \cos \theta_2 + [(h_2 - h_1) + r(\cos \alpha_1 - \cos \alpha_2)] = 0\tag{2.4}$$

$$(L_4 \cos \beta - L_3) \sin \theta_3 + (-L_4 \sin \beta) \cos \theta_3 + L_2 \sin \theta_2 + [(h_2 - h_3) + r(\cos \alpha_2 - \cos \alpha_3)] = 0\tag{2.5}$$

The two equations can thus be reformulated through the following coefficients:

$$A' = L_1 \cos \gamma - L_2\tag{2.6}$$

$$B' = -L_1 \sin \gamma\tag{2.7}$$

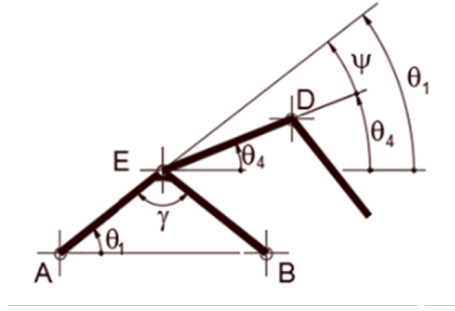


Figure 2.7: Composition of angles in the rocker-bogie mechanism

$$C' = (h_2 - h_1) + r(\cos \alpha_1 - \cos \alpha_2) \quad (2.8)$$

$$A'' = L_4 \cos \beta - L_3 \quad (2.9)$$

$$B'' = -L_4 \sin \beta \quad (2.10)$$

$$C'' = (h_2 - h_3) + r(\cos \alpha_2 - \cos \alpha_3) \quad (2.11)$$

in order to make eq. 2.4 and 2.5 into:

$$A' \sin \theta_2 + B' \cos \theta_2 + C' = 0 \quad (2.12)$$

$$A'' \sin \theta_3 + B'' \cos \theta_3 + C'' + L_2 \sin \theta_2 = 0 \quad (2.13)$$

From eq. 2.12 and 2.13, it is possible to obtain the reciprocal position of the links as a function of the assigned ground profile, given the geometric characteristics of the rover. Once the rover configuration has been determined, the ground reactions and traction capability of each wheel can be evaluated.

2.2.5 Load Distribution Analysis

Ground reactions can be estimated through a static analysis, considering the very slow speed of the rover. This analysis assumes that the calculated reactions are valid even for vehicles moving forward at a low constant speed, provided that each driving wheel exerts a driving force equal and opposite to the resisting force acting on the same wheel.

To perform the following analysis will be used the measurements in 2.8 and 2.2:

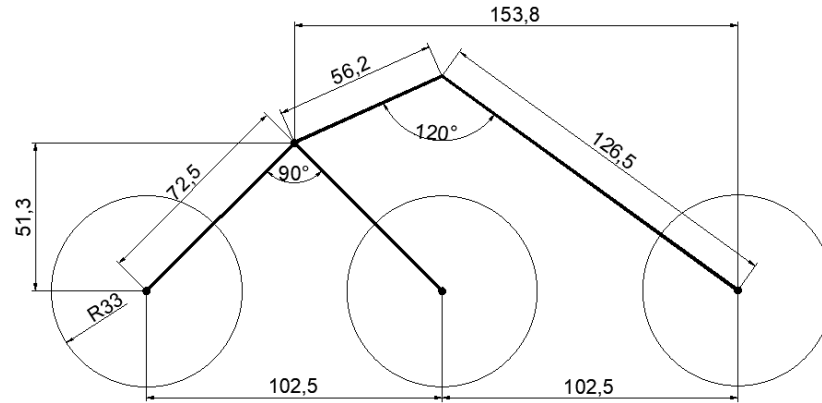


Figure 2.8: Geometric characteristics of the small prototype rover

$L_1(\text{mm})$	$L_2(\text{mm})$	$L_3(\text{mm})$	$L_4(\text{mm})$	$\psi(\text{deg})$	$\gamma(\text{deg})$	$\beta(\text{deg})$
72.5	72.5	126.46	56.23	33	90	120

Table 2.2: Geometric parameters of the small prototype rover

2.2.5.1 Load Distribution Without Obstacles

The purpose of this section is to verify the load distribution on the six wheels of the rover. Given the symmetry of the structure, only half of the rover is considered in this analysis. Furthermore, it is assumed that the mass of the rover is concentrated at its center of gravity. In Fig. 2.9, the load W , corresponding to the weight of the rover, is applied along a vertical line passing through the axis of wheel B. Additionally, it is assumed that the distance p between two consecutive wheels (the wheelbase) remains constant, provided that the wheels are positioned at the same level. Since the simplified structure in Fig. 2.10 is hyperstatic, the reactions of the wheels with respect to the ground are calculated using the method of forces. The introduction of a hinge in position E results in the division of the system into two distinct parts. The reaction of the hinge is exclusively vertical, given that the equilibrium of translation has no components in the horizontal plane, resulting in a zero component along that direction:

$$R_{Ex} = 0 \quad (2.14)$$

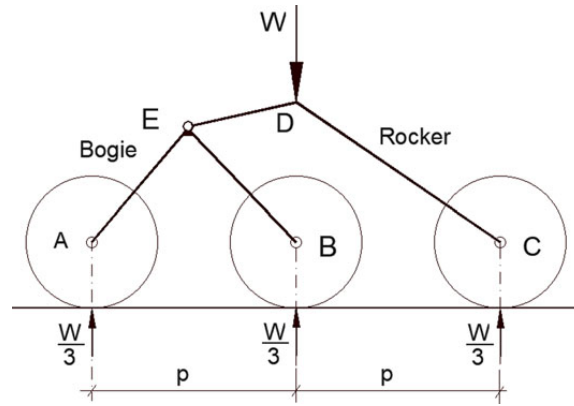


Figure 2.9: Load distribution in the rocker-bogie suspension system under uniform weight (W)

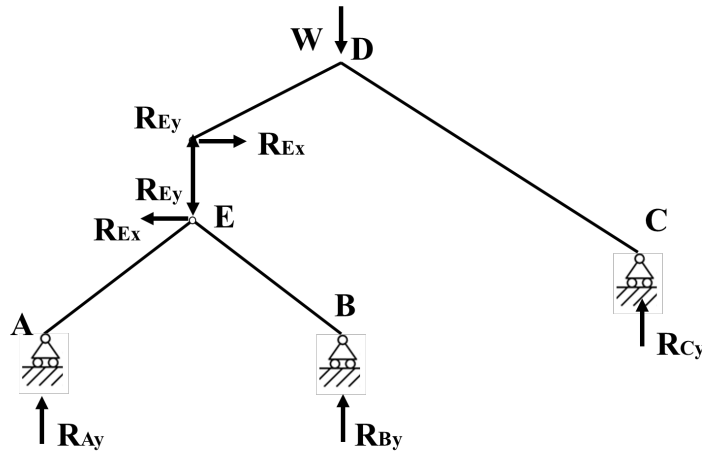


Figure 2.10: Equilibrium analysis of the rocker-bogie system in the absence of obstacles

The application of this method brings to the following result in the reaction forces:

$$R_{Ay} = \frac{W}{3} \quad R_{By} = \frac{W}{3} \quad R_{Cy} = \frac{W}{3} \quad (2.15)$$

2.2.5.2 Obstacle-Induced Load Changes

In the context of six-wheeled rovers traveling uneven terrain, a detailed analysis of the reaction forces affecting the suspension system is important. Such an analysis is significant to guarantee that the vehicle maintains optimal stability and sufficient performance when crossing obstacles such as steps.

Front wheel A approaching the obstacle

It is assumed that the weight of the rover is concentrated at point D and is evenly distributed across the wheels on flat, horizontal terrain. However, should the front wheel

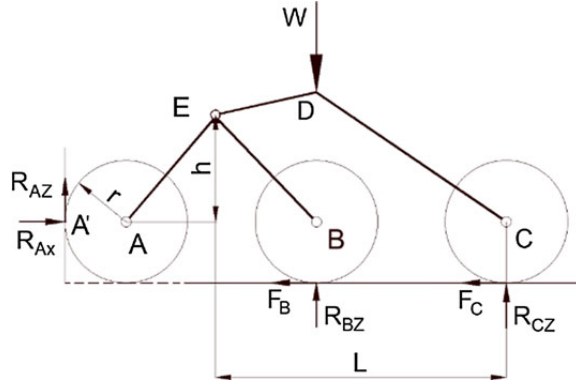


Figure 2.11: Load distribution in the rocker-bogie suspension system as Front Wheel approaches Obstacle)

come into contact with a vertical obstacle that is of a height equal to or greater than its radius (Fig. 2.11), the distribution of the vertical reactions on the ground will become uneven. The rear driving wheels (F_B and F_C) exert pushing forces, which allow us to consider the contact force between wheel A and the horizontal ground as negligible in comparison to the force exchanged with the vertical surface. Furthermore, the friction at contact point A can support the vertical load that the wheel is bearing. To prevent the front wheel from slipping, it is necessary that the vertical reaction meet a specific condition:

$$R_{Az} \leq R_{Ax} = f(F_b + F_c) \quad (2.16)$$

To prevent the rear wheels from slipping, it is necessary to ensure that the magnitude of the pushing forces remains below the corresponding grip limit:

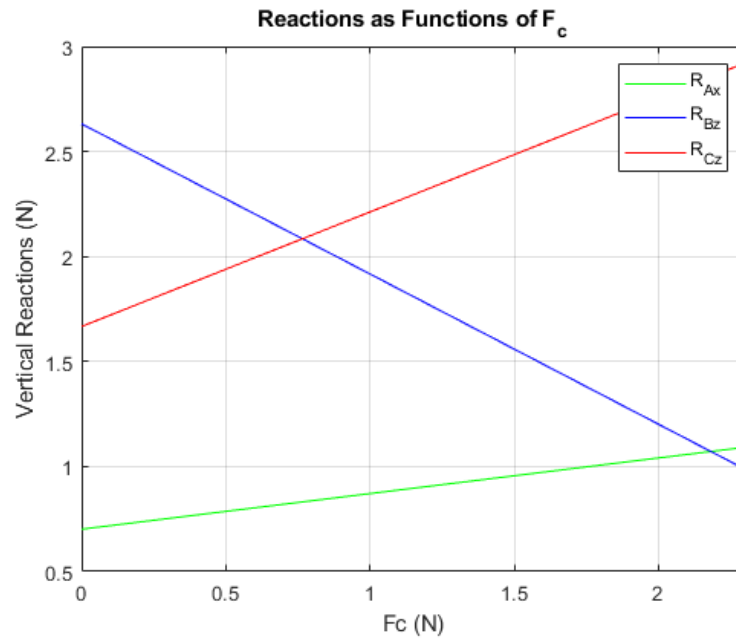
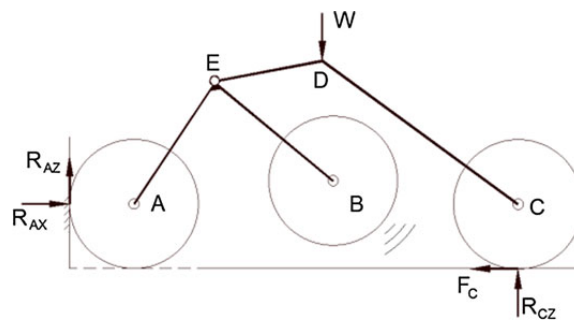
$$F_b \leq R_{bx}} \quad F_c \leq R_{cx} \quad (2.17)$$

In reference to Fig. 2.11, where h represents the bogie height and L denotes the horizontal distance between the axis of wheel C and hinge E when the three wheels are level, the external and internal reactions of the rover can be evaluated based on the condition of static equilibrium:

$$R'_{cz} = \frac{W}{3} + \frac{F_c(h+r)}{L} \quad R'_{ez} = \frac{2W}{3} - \frac{F_c(h+r)}{L} \quad R'_{ex} = F_c \quad (2.18)$$

$$R'_{bz} = \frac{R_{ez} \left(\frac{L}{3} + r \right) + F_b r - F_c h}{\frac{2}{3}L + r} \quad R'_{az} = R_{ez} - R_{bz}, \quad R'_{ax} = F_b + F_c \quad (2.19)$$

The diagram in Fig. 2.12 illustrates that as the force F_c increases, the vertical reaction on the intermediate wheel B decreases. Once F_c exceeds a critical value F_c^* (which is basically the grip limit), the vertical reaction at wheel B becomes null (Fig. 2.13).

Figure 2.12: Vertical reaction forces as a function of F_c Figure 2.13: Configuration when $F_c > F_c^*$

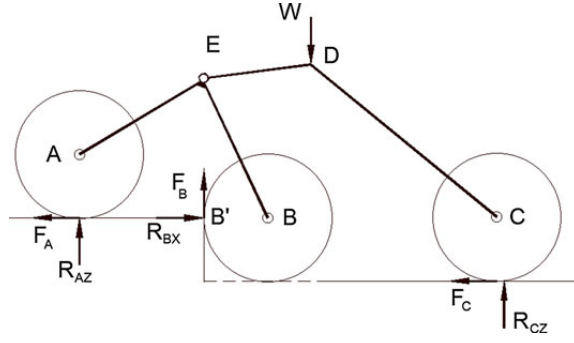


Figure 2.14: Load distribution as Wheel B approaches the obstacle ($h_1 = r$, $h_2 = h_3 = 0$)

Middle and Rear wheels approaching the obstacle

It is assumed that the front wheel A has successfully overcome the step obstacle, the height of which is equal to the radius of the wheel (Fig. 2.14). Wheel B is in contact with the vertical surface of the step obstacle as a result of the driving forces applied by wheels A and C. As in the preceding scenario, it is assumed that the contact force between wheel B and the horizontal surface is negligible and that the vertical friction force can support the wheel if specific conditions are met:

$$R_{bz} \leq f(R_{bx}) = f(F_a + F_c) \quad F_a + F_c \leq f(R_{az} + R_{cz}) \quad (2.20)$$

In order to evaluate the traction force F_c required to overcome the obstacle, it is necessary to consider wheel A as an idler. The traction capacity of the wheels is dependent upon the normal ground reactions, which can be expressed as follows:

$$R_{cz} = W \frac{(DE)_x}{(EC)_x} \quad R_{ez} = W \frac{(DC)_x}{(EC)_x} \quad R_{az} = R_{ez} \frac{(EB)_x}{(AB)_x} \quad R_{bz} = R_{ez} \frac{(AE)_x}{(AB)_x} \quad (2.21)$$

where:

$$(DC)_x = DC \cos(\pi - \theta_3) \quad (DE)_x = DE \cos \theta_4 \quad (EC)_x = (DC)_x + (DE)_x \quad (2.22)$$

Considering the reaction in eq. 2.21, the configuration in Tab. 2.3 is obtained:

$\theta_1(\text{deg})$	$\theta_2(\text{deg})$	$\theta_3(\text{deg})$	$\theta_4(\text{deg})$	$\psi(\text{deg})$	$R_a(N)$	$R_b(N)$	$R_c(N)$
26.2	116	139	18.9	7.37	1.62	2.69	2.34

Table 2.3: Rocker-bogie configuration parameters with wheel A on an obstacle step

As in (81) has been demonstrated, the trends of the soil vertical reactions relative to the force F_c indicate that F_c must be sufficiently high to ensure that R_{bz} values remain below the grip limit. However, it is notable that F_c itself is always below the grip limit. The reaction R_{az} increases with F_c . When wheel A is activated, the driving capacity of wheel C remains unchanged, while wheel A only contributes significantly at higher values

of F_c . Furthermore, the R_{bz} reaction increases as the force pressing the wheel against the obstacle intensifies.

In the final phase, which involves the climbing of wheel C (Fig.2.15), the driving action of one of the two bogie wheels is considered crucial. Specifically, wheel B is chosen due to its higher load among the bogie wheels. For this condition, the angles θ_i that define the configuration of the rover and the ground reactions are detailed in Tab. 2.4. When wheel C is pushed against the step obstacle, the external and internal constraint reactions are described as in eq. 2.23:

$$\begin{aligned}
 R_{cx} &= R_{ex} = F_a + F_b \\
 R_{cz} &= \frac{W(DE)_x - (F_a + F_b)(EC)_z}{(EC)_x - r} \\
 R_{ez} &= W - R_{cz} \\
 R_{bz} &= \frac{F_b[r + (EB)_z] + R_{ez}(AE)_x - F_a[r + (EA)_z]}{(AB)_x} \\
 R_{az} &= R_{ez} - R_{bz}
 \end{aligned} \tag{2.23}$$

with:

$$R_{cz} \leq fR_{cx} = f(F_a + F_b) \quad F_a + F_b \leq f(R_{az} + R_{bz}) \tag{2.24}$$

As the value of F_b increases, the vertical reaction on the intermediate wheel also increases, while remaining within the grip condition. Conversely, the vertical friction on wheel C decreases with F_c . The horizontal reactions R_{cx} and R_{ex} contribute to a counterclockwise rotation of the rocker, thereby facilitating the final phase of the process of overcoming the obstacle (81).

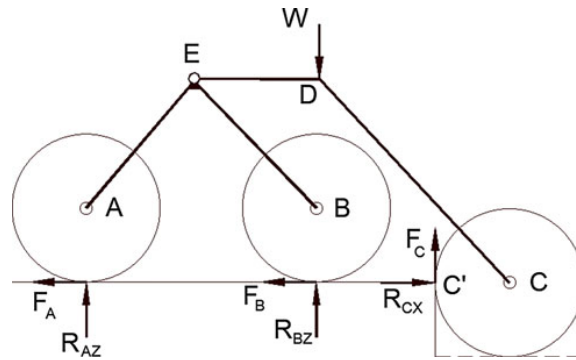


Figure 2.15: Load distribution as Wheel C approaches the obstacle

$\theta_1(\text{deg})$	$\theta_2(\text{deg})$	$\theta_3(\text{deg})$	$\theta_4(\text{deg})$	$\psi(\text{deg})$	$R_a(N)$	$R_b(N)$	$R_c(N)$
45.0	135	131	11.2	33.8	2.06	2.06	2.54

Table 2.4: Rocker-Bogie Configuration with wheel A and wheel B on obstacle step

2.2.5.3 Static Stability Analysis

Rovers are designed to perform complex missions with high levels of autonomy, as evidenced by their ability to operate effectively in a variety of environments and to complete a range of tasks without direct human supervision. A crucial attribute of these vehicles is their capacity to adapt to unfamiliar terrain. As the specific types of obstacles and slopes that rovers will face during explorations can not be predicted in advance, it is crucial to equip them with a mobility system that exhibits the necessary kinematic characteristics. The system should thus enable the rover to safely overcome obstacles of comparable size to its wheels. The conditions under which the binding reactions to the ground become zero are referred to as Rollover Boundary Conditions. For the sake of simplicity, only half of the structure is considered, and it is assumed that the mass of the Rocker and Bogie is negligible in comparison to the mass of the chassis and motor-wheel assemblies. By assuming that all the mass of the system is concentrated at point D of the rover body, rollover can occur if certain criteria are met:

- $\theta_4 > \frac{\pi}{2}$ in this configuration, the rollover occurs around point E (Fig. 2.16b).
- $\theta_3 > \frac{\pi}{2}$ in this configuration, the rollover occurs around point C (Fig. 2.16c).

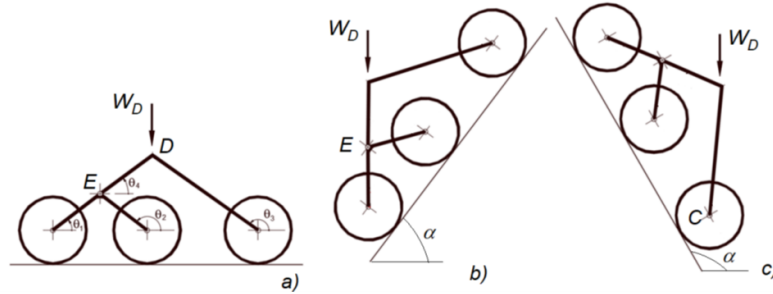


Figure 2.16: Rollover Boundary Condition: (a) on flat ground, (b) ascending an incline, (c) descending an incline

Once the simplifying assumption that neglects the mass of the links and the motor/gear-box/wheel assemblies is removed and it is instead assumed that these additional masses are concentrated at points A, B, and C corresponding to the axles of the three wheels, a stabilizing effect is observed. This results in larger values of the angles θ at which stability is maintained. In this scenario, the rollover limit is defined as the point at which the following equalities are reached:

$$W_D(DE)_x = W_C [(DC)_x - (DE)_x] \quad (2.25)$$

$$W_D(CD)_x = W_E [(ED)_x - (CD)_x] \quad (2.26)$$

With a rotation around hinge E (Fig. 2.16b), if the soil vertical reaction becomes zero, the

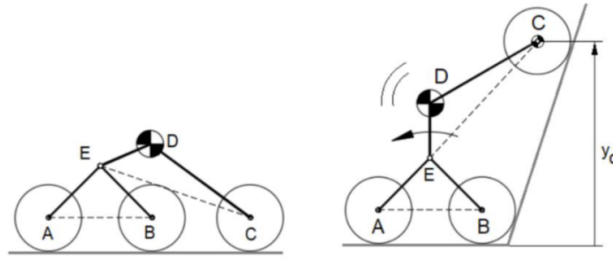


Figure 2.17: Stability analysis of the rocker-bogie mechanism: (left) on flat terrain, (right) at the tipping point during an obstacle climb

condition occurs when the horizontal projection of the distance ED becomes zero, i.e., $(ED)_x = L_4 \cos \theta_4 = 0$. This implies that $\theta_4 = 90^\circ$. This condition can still be stable if the stabilizing mass of the wheel-motor, lumped at point C, is considered. In this case, the overturning limit is defined through the balance eq. 2.25. With a rotation around the axis of wheel C (2.16c), if the soil reaction at wheel A becomes null, the condition occurs when the horizontal projection of distance DC becomes zero: $(DC)_x = -L_3 \cos \theta_3 = 0$, implying that $\theta_3 = 90^\circ$. This condition remains stable due to the presence of the wheel rotor mass of the bogie. The overturning limit in this case is defined in eq. 2.26.

It is essential to recognize that the effective angle limit θ_i must account for the longitudinal accelerations experienced during motion, which can induce inertia forces that amplify the tilting moment. When the stabilizing mass is positioned at point C and the moments of the weights at C and D are analyzed with respect to point E, it can be observed that the angle θ_4 at which overturning occurs (with $R_c = 0$) is greater than 90° (Fig. 2.17)(86).

2.2.6 Conclusions

In conclusion, the analysis of the rocker-bogie suspension system has confirmed its strengths and limitations in handling complex terrain and obstacles. The passive suspension configuration allows the rover to maintain stability on uneven surfaces, while the load distribution between the wheels helps to optimize traction and minimize structural stresses. However, overcoming obstacles of significant height requires additional precautions, as the center wheels tend to lose traction in some configurations, limiting forward capability.

While the rocker bogie suspension system effectively provides stability and adaptability on uneven terrain, its limitations in traction and obstacle overcoming highlight the need for further design optimisation. The following sections detail the development and simulation of a small-scale rover prototype, providing insights into practical implementation and performance evaluation.

2.3 Small Rover Prototype

This section describes the process of designing and modeling the multibody prototype of a small-size six-wheeled rover, using Simscape Multibody™ to simulate the structural and dynamic characteristics of the system. Through a modular approach, the model includes both the rover components and the interaction environment, providing a realistic representation of the operating conditions. Each part of the rover was meticulously modeled to ensure fidelity to the physical prototype, with imported CAD and detailed configuration of mechanical and dynamic properties. The goal of the chapter is to show how the integration of electrical and mechanical components enables accurate simulation of the performance of the rover under different operating conditions.

2.3.1 Multibody Prototype

In accordance with the block algebra of Simscape Multibody™, the constructed multibody model exhibits a modular structure, comprising multiple layers. The initial level, which will be illustrated in the Fig. 2.18, comprises the following components:

- Reference Block, Basic Configurations, and Solver: Basic elements that define the operational framework and computational approach for the rover's system simulations.
- Multibody of the Rover: Detailed configuration of the rover's structure, encompassing all interconnected components that contribute to its physical and functional properties.
- Multibody of the Environment: Description of the environment in which the rover operates, focusing on the physical and dynamic characteristics that influence rover interactions and performance.

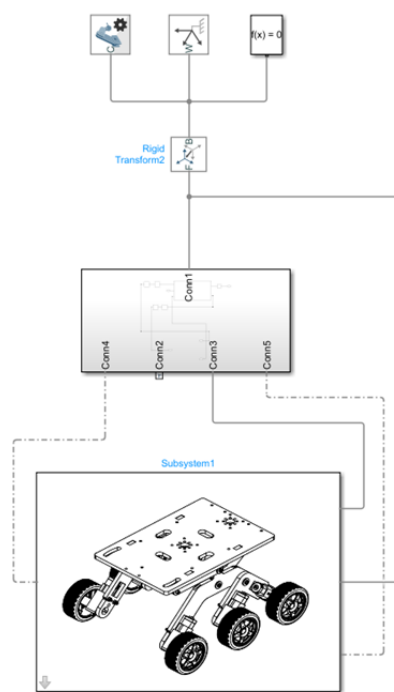


Figure 2.18: Multibody model of the rover system

2.3.1.1 Rover Macroblock Development

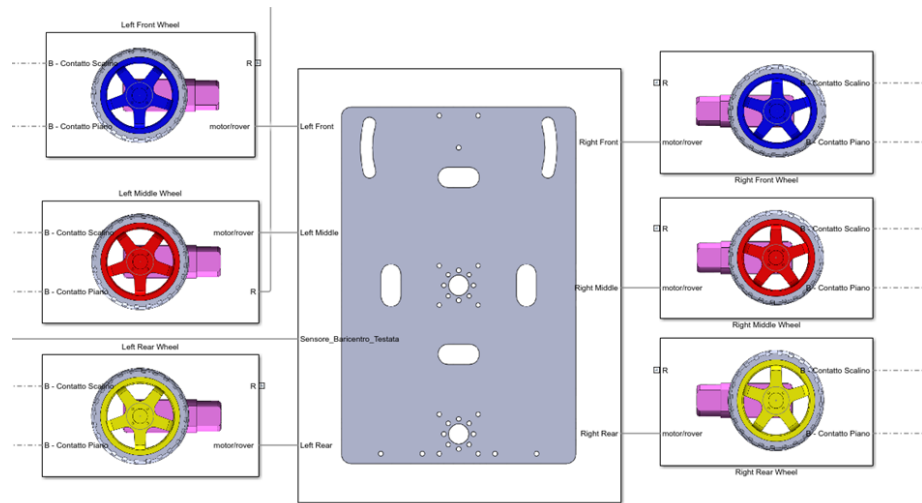


Figure 2.19: Layout of the rover

The Rover Multibody System macro block is internally organized into seven sub-macro blocks as shown in Fig. 2.19:

- One macro block is dedicated to the rover chassis multibody.
- Six macro blocks are dedicated to the drive units, each consisting of engine, transmission, and wheels.

Rover Chassis Multibody

To implement rigid bodies in the model, Simscape Multibody™ offers several options: define parameterized 3D shapes, create 2D profiles in Matlab® to extrude or rotate about an axis, or import CAD files via the File Solid block. While the first two options are ideal for models with simplified geometries, File Solid allows you to import more detailed and realistic 3D drawing, as it possible to see in Fig 2.20. The choice of modeling technique depends on the specific goals of the project. In the thesis presented here, the import of CAD files was chosen in order to obtain a model and simulation as close as possible to reality, with the aim of obtaining more accurate results. Starting from the actual prototype, a reverse engineering operation was performed to obtain the dimensions of each component, which was then modeled in CAD using SolidWorks®. The modeled structural components were implemented in the Matlab® environment to take advantage of true-to-life simulation. Individual components were assembled in SolidWorks® into two macro-assemblies to verify proper sizing and mating without geometric interference. The first macro-assembly includes all elements of the rover frame, specifically divided into header, rocker, bogie, brackets, rocker arm, and joints. The macro-assembly, which includes the Adafruit Motorshields and battery pack with four AA batteries, used STEP

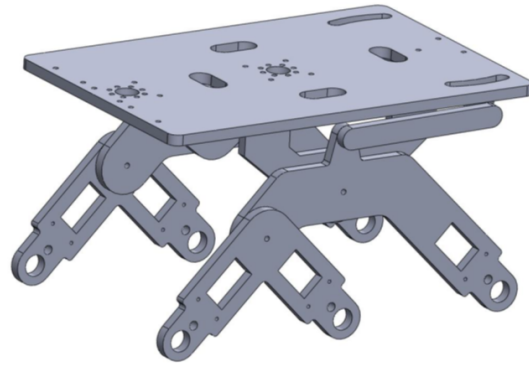


Figure 2.20: 3D model of the rover chassis

files from GrabCad, an online platform with free files. It was possible to import assemblies made in SolidWorks® into the multibody model. This would have allowed Simscape to automatically build a block model. However, this option was not pursued.

A more manual strategy was chosen to maintain control over the model. Each component was entered into the multibody model with properties like inertia and density and joints to connect parts. This approach allowed more detailed customization and finer control over the model, which is essential for accurate and reliable simulations. In the various macroblocks of the multibody model, it is essential to include elements corresponding to those present in reality, such as rotoidal and cylindrical joints. Rotoidal joints allow a rotational degree of freedom between two local reference systems, aligning their origins and z-axes, while allowing rotation of the x- and y-axes of the follower system around the common z-axis. In contrast, cylindrical joints offer two degrees of freedom, allowing both rotation and translation around or along the z-axis, which coincides between the two connected reference triads.

The properties panel of these joints allows the configuration of various aspects, including the state of the joint, the actuation method, and internal mechanical parameters such as stiffness and damping. Moreover, the panel allows the user to set the quantities to be monitored by sensors embedded in the joints, which can provide real-time data on displacement, velocity, and acceleration. The aforementioned configurations modify the blocks by incorporating output ports for the desired physical signals.

In multibody models, the type of joint in question determines the specific quantities that can be measured. In the case of rotational or spherical joints, the measurable parameters include the angle of rotation, angular velocity, and angular acceleration. These parameters reflect movement around a fixed axis. In contrast, for cylindrical-type joints, which combine rotation and translation, the measured parameters are offset distance (the translation along the axis of the joint), linear velocity, and linear acceleration. This distinction is of paramount importance for the proper design of the sensing system and the assurance that the measurements are aligned with the dynamic specifications of the joint.

Drive Units Multibody

The second macro-assembly includes the handling system of the rover, specifically the motor, gearbox, wheel, and tire. The motor-gearbox assembly was not detailed in the multibody model because the objective was not to simulate its internal dynamics. These geometries were modeled to represent the corresponding masses in the multibody (Fig. 2.21)

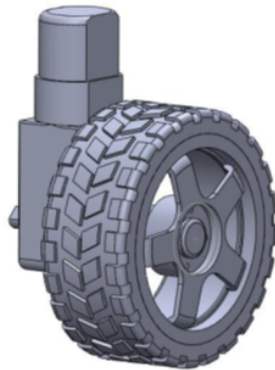


Figure 2.21: 3D model of the Motor-Wheel

In Fig. 2.22, there is one of the six handling assemblies of the system. In addition to the solid files representing the rim, tire, and motor mass, this block also includes a model of a direct-current (DC) electric motor that is intended for handling. Furthermore, the block incorporates the modeling of the physical interactions between the tire and the ground, as well as between the tire and any obstacles. This section of the model serves to illustrate the multiphysics nature of the model, integrating elements of mechanics, electromagnetism, and contact dynamics.

DC Motor Block

In the DC motor model, power electronics is represented by integrating two distinct physical domains: electrical and rotational mechanics, as it possible to see in Fig. 2.23. This multidisciplinary approach allows for the accurate simulation of both the electrical behavior of the motor, which includes voltage, current, and resistance, and its mechanical dynamics, which covers torque, inertia, and angular velocity. The interaction between these two physical domains is essential for obtaining a realistic representation of motor performance under diverse loads and operational conditions. The modeling of the DC motor in the model incorporates elements from both the electrical and mechanical domains. A DC motor model was constructed by integrating electrical and mechanical domain components. The electrical component was employed to construct the motor power circuit. In contrast, the mechanical domain was utilized to represent the gearbox and facilitate the connection between the motor and the multibody system. The flow of the model progresses from left to right through the following principal blocks:

1. An ideal voltage generator provides the requisite supply voltage for motor operation.

6. A multibody coupling serves to mechanically connect the motor and gearbox to the remainder of the multibody system.
7. An ideal angular speed source is defined as a block that imposes a specific angular speed on the system, thereby generating the requisite torque to maintain that speed.

A Simscape block was configured to represent a permanent magnet motor in order to model the DC motor. Two sources of data were employed to ensure the accurate definition of the model parameters. The first was the technical specifications provided in the motor-gearbox assembly sheet, and the second was data collected through experimental characterization of prototype motors in the laboratory. The latter were crucial for the refinement of the model, ensuring that it accurately reflected the actual performance of the components under test. The methodology employed to obtain these data will be detailed on following. At the end of this procedure, it comes out with the parameters in Fig. 2.24.

Block Parameters: REVOLUTE_R1	
Subsystem (mask)	
Motor	
no-load speed (rpm)	230*48
nominal speed (rpm)	180*48
nominal torque (Nm)	0.07845384/48
nominal voltage (V)	6
no load current(A)	0.190
motor inertia (g*cm ²)(I=1/2Mr ²)	1/2*100*0.3 ²
transmission ratio	48
Supply voltage (V)	6
nominal current (A)	0.250

Figure 2.24: Specifications of the DC Motor

The operation of a macroblock can be outlined as follows, with reference to a typical, generic moment in time:

1. **Generation of the Voltage Signal:** In the initial stage, a voltage signal is generated externally via a ramp block and subsequently limited to a specified maximum value. The signal is then transmitted as an input to the Voltage Handling.
2. **Voltage Handling:** At this point, an ideal voltage generator assumes control, allowing the earlier-mentioned voltage to pass through. This configuration guarantees that the voltage delivered to the motor remains stable and regulated.

3. Power Measurement: Subsequently, the voltage signal is directed through a power sensor, which is responsible for measuring the electrical power consumed by the system before it reaches the DC electric motor.
4. Motor Speed Computing: Concurrently, data on the instantaneous rotational velocity are obtained from the multibody coupling. Once the data has been modified by the gearbox, it is used to establish the speed of the motor.
5. Torque and Current Calculation: Once the motor speed and applied voltage are known, the armature torque and current are determined using the formulas provided in the 1.7.3.2.
6. Motor Torque Transmission: The generated torque of the motor is transmitted via the gearbox to the rotodynamic coupling, affecting the rotational speed of the system. This phase comprises a continuous feedback loop that enables adjustments to be made to the motor speed and torque in accordance with the dynamic conditions of the system.

As previously outlined, the speed of DC motors can be adjusted by varying the voltage supplied to the motor. This method allows for the modification of the operating point of the system, defined by the intersection of the motor torque curve and the resistance torque curve (Fig.2.25). By varying the voltage, the characteristic curve of the electric motor is modified, enabling the use of various combinations of speed and torque.

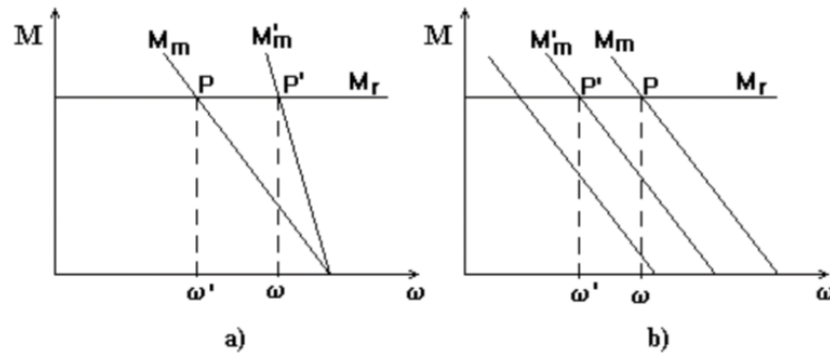


Figure 2.25: DC motor characteristic curves: (a) Slope variation indicates motor resistance changes; (b) Rightward shift indicates a voltage increase

In the prototype, the operation of the electric motors is controlled via a pulse-width modulation (PWM) signal. The PWM converts the voltage into a square wave (Fig. 2.26) by alternating active voltage intervals (denoted as T_a) with inactive intervals (denoted as T_s). By modifying the duration of these intervals, specifically the ratio of T_a to the total period ($T_a + T_s$), which is referred to as the *duty cycle*, it is possible to regulate the average voltage applied to the motor. This method allows for a wide range of average voltages, which directly influences the speed and torque of the motor as required for operation.

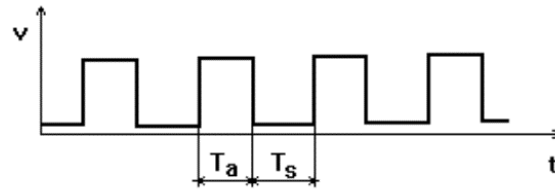


Figure 2.26: Representation of Pulse Width Modulation (PWM) signal

PWM is a control method that facilitates the regulation of the average voltage applied to DC motors. This is achieved by alteration of the duty cycle. The resulting average voltage can then be adjusted based on the desired torque characteristic curve, as illustrated in Fig. 2.25b.

2.3.1.2 Experimental Characterization of DC Motors

The manufacturer of the electric motors used in the prototype did not provide a diagrammatic representation illustrating the variation in torque with different supply voltages. As a result, an unexpected opportunity was presented for conducting experimental research to determine the characterization curves of the motors. The experimental process allowed for the collection of data directly from the behavior of the motor under different supply conditions, thus providing an empirical foundation for optimizing the performance of the motor in the context of the prototype.

A typical characteristic curve of a DC electric motor in the angular momentum-velocity ($M - \omega$) plane is represented by a straight line with a negative slope, which is characteristic of the fundamental operating principle of such motors. In order to define these curves, two crucial parameters—no-load speed and stall torque—were obtained through experimentation at different values of supply voltage:

1. No-load Speed: This represents the maximum speed that the motor is capable of reaching without a load. It is indicative of the point at which the torque is at its minimum and marks the point of intersection between the curve and the x-axis.
2. Stall Torque: The maximum torque delivered by the motor at zero speed is represented by this point, which marks the intersection of the curve with the y-axis.

In order to plot the characteristic curve for each voltage level, a range of PWM values was configured on the Arduino®. The duty cycle was varied from 0 to 255 (8-bit PWM registers). The selected values were 255, 220, 200, 170, 150, 130, 100, and 40, which correspond to duty cycles ranging from 100% to approximately 16%, respectively. The voltage applied to the motor for each PWM value was measured using a voltmeter in electrical contact with the motor shield terminals.

The characterization of the motor was preceded by an evaluation of the inertia of the masses mounted on the crankshaft, including the tire, rim, and a cardboard disk

utilized for system stabilization. Subsequently, the masses were shaped into cylinders with a radius of 32mm, thereby enabling the moment of inertia, given by the expression $I = mR^2/2$, to be rapidly calculated as 0.039204 kg·m².

At this point, the experiment was divided into two distinct phases: the first was the estimation of no-load speed, and the second was the estimation of stall torque:

1. No-load Speed Measurement: Initially, an Rs-Pro model Rm-1501 electronic tachometer was used to measure rotational velocity without direct physical contact by employing a laser beam reflected off a tape square on the rotating (Fig. 2.27). The tachometer detects this reflection to calculate and display the speed in revolutions per minute (RPM). To improve measurement accuracy, especially at low speeds, five reflectors were placed on the wheel spokes, with the recorded value then divided by five to obtain an accurate speed estimate. This method facilitated the identification



Figure 2.27: Electronic tachometer used to measure the rotational velocity of a wheel

of the intersections between the motor characteristic curves and the x-axis of the $M-\omega$ plane.

2. Stall Torque Measurement: The second phase aimed to measure the maximum torque, occurring when the motor stalls under increasing resistive load (Fig. 2.28).

PWM	Voltage [V]	No-Load Speed [RPM]
255	6.15	244
220	5.33	209.8
200	4.88	188
170	4.13	158.8
150	3.64	138
130	3.17	119.6
100	2.44	87
40	0.98	29.15134

Table 2.5: PWM values, corresponding voltages, and measured no-load speeds of the DC motor

An "inextensible" wire system, assumed to be reliable, was connected to suspended masses that simulated the load, converting the wheel into a rope-cog system. The load was increased until the motor could no longer move, thus defining the maximum torque deliverable at zero speed. This torque corresponds to the intersection between the motor characteristic curves and the y-axis of the $M-\omega$ plane. The same supply voltages used in the no-load speed measurement were employed here. The

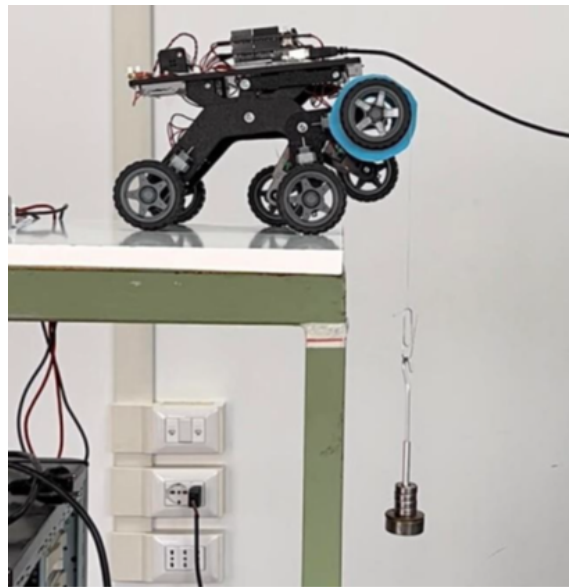


Figure 2.28: Stall torque characterization test for the motor of the rover

standardized weights attached to the wire exerted a resistive torque equal to the radius arm of the wheel. The maximum load lifted in grams for each voltage was noted, and the maximum torque was calculated using the formula $M = F_p \times r$, where M is the torque in newton-metres, F_p is the force in newtons, and r is the wheel radius, taken as 32.5 mm.

PWM	Weight [g]	Stall Torque [Nmm]
255	309.8	100.29154
200	243.8	78.92537
170	198.0	64.09854
150	186.8	60.47276
130	159.5	51.63493
100	121.3	39.26849
40	29.8	9.64715

Table 2.6: PWM values and corresponding measurements of weight and stall torque for the DC motor

As result of this 2-steps test, it is possible to plot the Characteristic DC-Motor Curve for each Voltage level (Fig. 2.29). In comparison, the nominal Characteristic Curve has been plotted according the No-load nominal speed and Stall Torque at 6V.

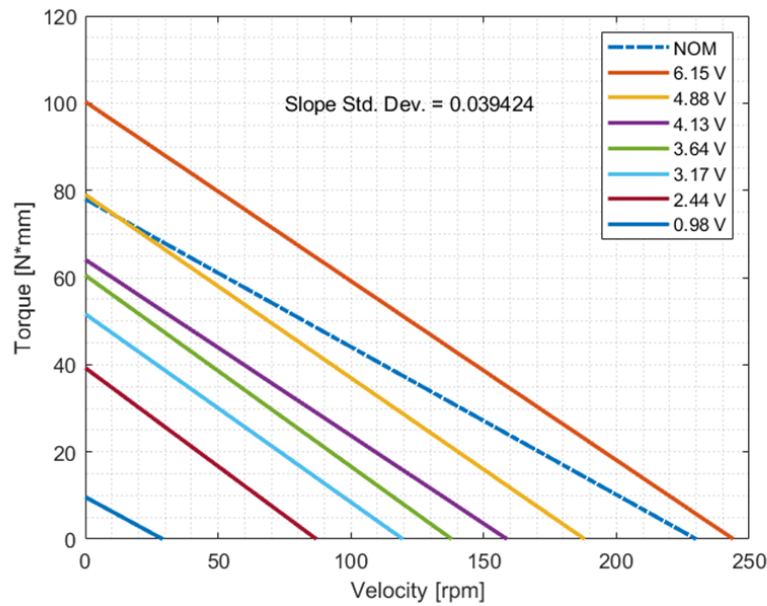


Figure 2.29: Torque-Velocity ($M-\omega$) plane for the DC motor at different voltage levels

As anticipated by the theory, it was predicted that modifying the supply voltage would not result in alterations to the mechanical characteristics of the motor. Rather, it was expected that the operating curves would shift in parallel with one another. The results validated this hypothesis, demonstrating that the curves are essentially parallel. Additional evidence in support of this observation was obtained by calculating the angular coefficient for each curve, with the standard deviation of these coefficients calculated to be 0.03942. This finding is in alignment with the macroscopic observations.

For a more comprehensive assessment, the resulting curves were compared with the nominal values provided in the datasheet. Although there is a slight discrepancy between

the nominal and actual values, this difference is considered acceptable given the quality of the motors used in the prototype.

The values obtained from these experimental tests were then inserted into the multi-physics motor model in Simscape.

Starting from this type of motor model, looking at the response in terms of angular velocity, a maximum velocity value of approximately 24 rad/s was achieved.

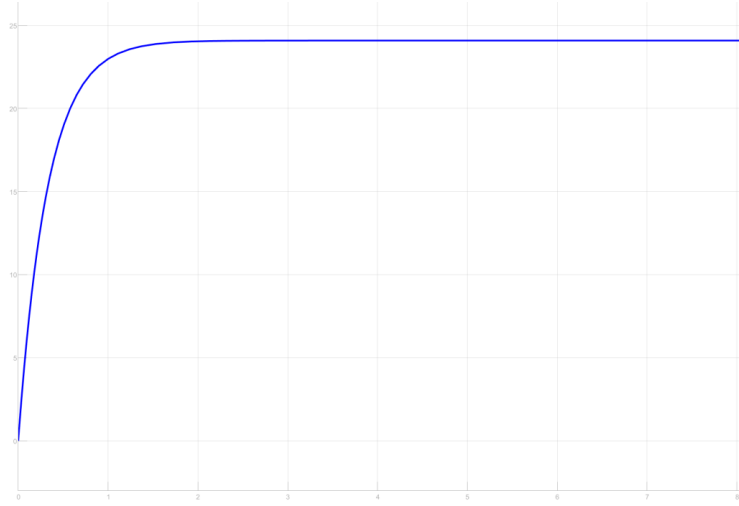


Figure 2.30: Motor-wheel velocity for the Simscape Model

Observing this trend (Fig. 2.30), and given the absence of oscillations and overshoots, it was possible to assimilate the angular velocity trend of the motor to first order dynamics. On the basis of the following speed evolution, the differential equation for the dynamics of the motor-wheel applied to an electric voltage u is written:

$$a_1 \dot{\omega} + a_2 \omega = bu \quad (2.27)$$

The following differential equation has been subjected to the Laplace transformation, resulting in the following transfer function (output/input ratio) of the motor + wheel system:

$$\frac{W(s)}{U(s)} = \frac{b}{a_1 \cdot s + a_2} \quad (2.28)$$

First, the parameters a_1 , a_2 and b were initialised to the unit value and a voltage of 6V was assigned to the input, which corresponds to the supply voltage of the Simscape model with reference to the real electric motor + wheel assembly.

From the measured velocity trend in the Simscape model (Fig. 2.31), it was possible to identify the parameters a_1 , a_2 and b using the Simulink parameter estimation toolbox. The software optimises the model parameters to obtain a simulated response y that matches the measured response $\hat{y}$. To do this, the solver minimises the cost function or estimation error, a measure of the difference between the simulated and measured responses. The

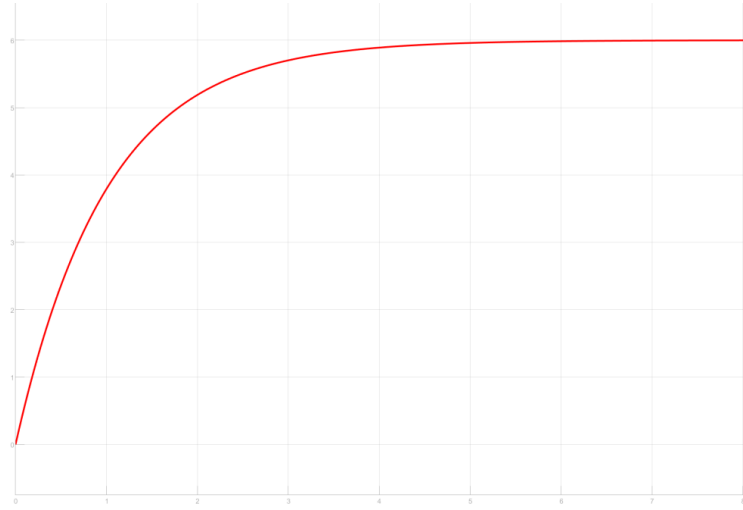


Figure 2.31: Measured Velocity Wheel $\hat{y}$

cost function $F(x)$ is the objective function of the optimisation problem and can be written as follows:

$$F(x) = \sum_{t=0}^{t_f} e(t)^2 \quad (2.29)$$

$$e(t) = y(t) - \hat{y}(t) \quad (2.30)$$

where t_f is the final time.

The aim is to minimise the objective function $F(x)$, where the vector of parameters to be optimised is defined by

$$x = \{a_1, a_2, b\} \quad (2.31)$$

The method used is least squares, which minimises the squares of the residuals (eq. 2.30) evaluated at each time point.

The identified parameters:

$$a_1 = 0.85 \quad a_2 = 2.64 \quad b = 10.6 \quad (2.32)$$

make it possible to obtain a dynamic behaviour of the motor-wheel assembly, modelled with the differential equation represented in eq. 2.27, comparable to that of the Simscape multiphysics model validated on experimental tests, giving the same velocity tending to a value of 24 rad/s (the response of the Simscape model y and that of the motor-wheel model $\hat{y}$ tend to coincide). After this initial identification of the parameters related to the electric motor model on the rover, we have the tools to correctly calibrate the two controllers and proceed with the analysis of each control system and its subsequent application on the prototype.

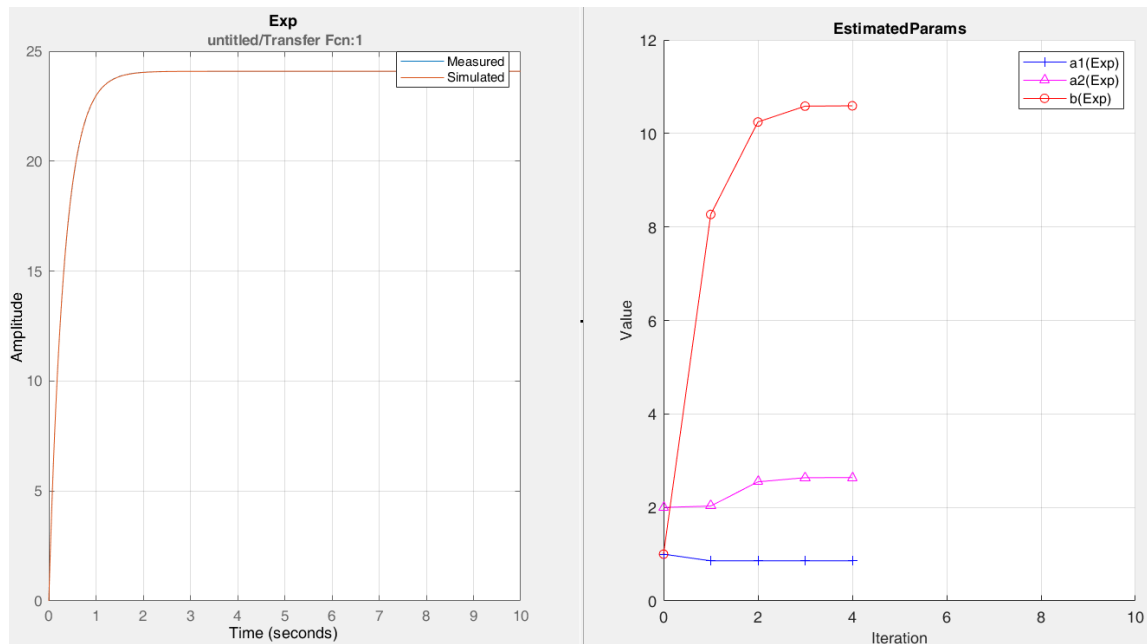


Figure 2.32: Parameters Optimization

2.3.1.3 Contact Model: Ground and Obstacle

In the model, interactions between tires and road surfaces or obstacles are simulated using the Simscape *Spatial Contact Force* block, which is based on the penalty method, described in detail in 1.7.3.1. Figure 2.33 illustrates the parameter configuration panel pertinent to the model. In this context, the symbols “mi_s” and “mi_d” represent the static and dynamic soil friction coefficients, respectively, while the symbols “mi_s_g” and “mi_d_g” indicate the friction coefficients for the step.

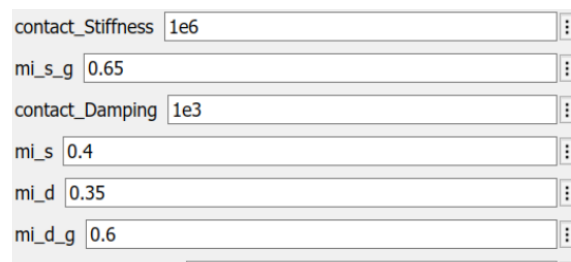


Figure 2.33: Friction Coefficients: ground and obstacle step

In order to facilitate the relative motion of the rover on the modeled ground, it is necessary to define the relative position between the multibody structure of the rover and the soil (Fig. 2.34). This is accomplished through the utilization of the Simscape *6-DOF Joint* block. The six-degree-of-freedom joint does not impose constraints on the two structures; instead, it determines their relative positions, enabling the accurate simulation of their interactions. The ground and obstacle multibody system was designed using two

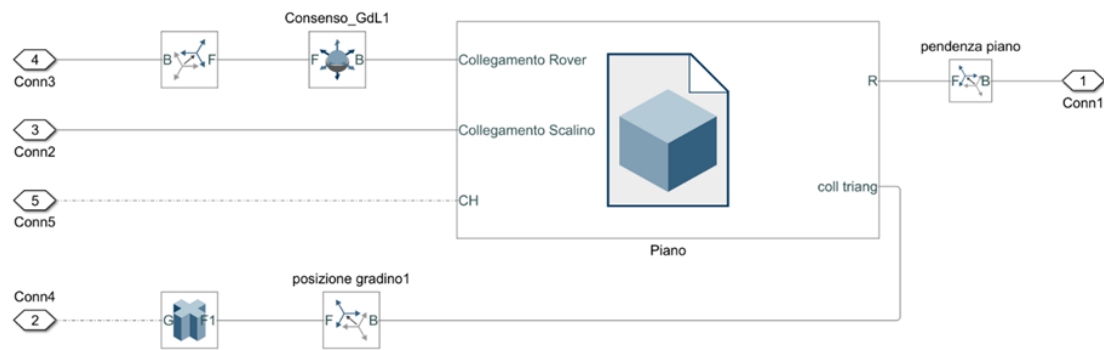


Figure 2.34: Soil Model

solid bodies that are rigidly connected and oriented in space. Modifications to the CAD model of the obstacle facilitate the creation of an array of obstacle types, thereby enabling the evaluation of the performance of the rover across a spectrum of terrains.

Wheel as Convex Hull

As it said in this section, a fundamental part is the examination of the general wheel-ground contact. The manner in which the simulator interacts with the two bodies during the contact phase is of very important. This section will provide a brief overview of the approximating the wheel by a shape known as the convex hull. In the field of geometry, the convex hull is defined as the smallest convex set that is capable of enveloping a given body. The convex hull can be conceptualized as the shape that emerges when an elastic band is stretched around a set of points and subsequently allowed to contract until it reaches its optimal degree of tautness. A set is convex if, for any two points, A and B , within the set, the line segment AB is entirely contained within the set. The convex hull of a set X can be defined in a number of ways.

1. The minimal convex set that encompasses the given set.
2. The intersection of all convex sets that include X .
3. The set of all convex combinations of the points in X .
4. The union of all convex hulls with vertices at the points in X .

In the case of bounded sets in the Euclidean plane, the edge of the convex hull represents the simplest closed curve with the minimum perimeter that encloses the set. This process may be conceptualized as extending a rubber band around the perimeter of the set and then allowing it to contract until it becomes fully taut, forming the convex hull. In three-dimensional space, the convex hull represents the smallest convex volume capable of containing the given set of objects. As illustrated in Fig. 2.35, the wheel is



Figure 2.35: Wheel Convex Hull

represented as a convex hull, which is considered to be a "smooth" wheel. However, this model does not consider the typical roughness of a wheel, such as grooves or cleats. This simplification may affect the accuracy of the results regarding the interaction between the wheel and the road, as it ignores elements that could have a significant impact on the actual behavior of the wheel.

Wheel as Point Cloud

When the approximation of the wheel becomes the central part of the analysis, the point cloud approximation is employed because it more closely aligns the simulations with real-world scenarios. This approach is particularly advantageous when contact physics is a primary focus of the simulations. However, in other cases, the convex hull model may still prove effective.

A point cloud is a data set that represents a three-dimensional object or surface through a set of points, each with specific coordinates (xyz) in a reference system. These clouds are typically generated from photogrammetric surveys or laser scans, thereby creating an accurate three-dimensional representation of the surveyed environment or object. From the point cloud, a mesh model can be generated, which is a network of polygonal surfaces constructed by connecting segments between points in the cloud.

This technique is widely employed in spatial surveying, the representation of three-dimensional structures, and metrology, offering substantial benefits such as the generation of three-dimensional CAD models, precision measurements, and quality control. Point clouds are compatible with a variety of file formats, including LAS, LAZ, PLY, BIN, XYZ, and ASCII, and can be exported to CAD or GIS environments. Some of the simulation analysis will examine a point cloud comprising all 24 studs on the outer surface of the tire, allowing for a comparison between ideal and actual cases to assess the impact of surface roughness on wheel-ground contact simulations.

In order to guarantee the highest degree of precision, the approach of commencing with the STL file was utilized for the effective incorporation of the point cloud into the Simscape model. STL (Stereo Lithography Interface Format), also known as "Standard Triangulation Language" or "Standard Tessellation Language," is a file format that is widely utilized in CAD stereolithography software, particularly in the context of rapid prototyping.

An STL file represents a solid as a surface divided into triangles, with each triangle defined by the X, Y, and Z coordinates of its three vertices, accompanied by a vector describing the orientation of the normal to the surface. The normal vector, which indicates the exterior of the solid, is determined using the right-hand rule based on the order in which the vertices of the triangle are arranged.

The adoption of the STL format permits the exact delineation of intricate surfaces, thereby facilitating their incorporation into the simulation model and, consequently, enhancing the precision of analyses conducted through Simscape.

The degree of complexity inherent in an object's surface directly influences the number of triangles that are generated within an STL file. The latest versions of CAD software are typically equipped to export models in this format, offering users a straightforward

process: simply select the option to save the file with the .stl extension. The resolution of the STL file can be modified through the utilization of specific settings, such as "Chordal Tolerance" and "Angular Control." These settings establish the number of triangles that comprise the model, increasing its complexity and the file size. A larger file, containing a greater number of triangles, can facilitate an enhanced quality of model representation. In contrast, simple geometries, which are characterised by minimal curves, result in smaller files, which are typically only a few hundred kilobytes in size. For more complex models, files ranging from 1 to 5 MB are typically sufficient to achieve high-quality parts. However, files exceeding 5 MB do not typically offer substantial enhancements in model quality and may result in longer processing times, which could potentially impact the preparation of quotations and the delivery of parts. It is advisable to avoid excessive reduction in thickness, as this can result in inadequate feature formation and the potential for quality issues during the fabrication process.

The STL format offers several notable advantages, including ease of generation and processing. However, it also has inherent limitations. One of the limitations of this format is that it provides only an approximate geometric representation, which often results in the repetition of the same vertex multiple times within the data structure.

In the process of exporting to STL format, the most critical parameter to consider is the tolerance or angular deviation, expressed in degrees. A lower tolerance value indicates a higher resolution, which results in a larger exported file. Conversely, a higher tolerance value results in a reduction in resolution and a concomitant decrease in the file size.

In the case of the tire model, the STL file was extracted from the CAD model with minimal tolerance to ensure maximum resolution. This approach resulted in the generation of a point cloud comprising 146,999 points (Fig.2.36), which indicated a high level of detail. While this level of detail is essential for accurately modeling complex features like tire cleats, it can also result in large files and extended processing times during simulations. In order to optimize the analysis while maintaining the requisite quality of the simulation and reducing the computational load, it was essential to reduce the number of points in the point cloud. This approach permitted the importation of a more compact point coordinate matrix than was originally possible, thereby facilitating faster software execution without any loss of accuracy that would have been required for the specific analysis.

The objective was achieved through the development of a Matlab code designed to remove sections of the wheel that do not engage with the ground. This entailed the elimination of regions of the object that, despite being part of the wheel, do not directly contribute to the aspects of wheel-ground interaction under study. The removal of these nonessential parts resulted in a reduction of the number of points to be processed, thereby simplifying data handling and enhancing the efficiency of the simulation process.

The Matlab code then selectively extracted data from the point cloud, focusing on the importance of various wheel sections in relation to their interaction with the ground surface. This approach guaranteed that the final analysis was both accurate and computationally manageable resulting in a point cloud as in Fig.2.37. The initial reduction resulted in a point cloud that retained a total of 51,046 points, which constituted a significant quantity that negatively impacted the computational efficiency. Accordingly, an additional

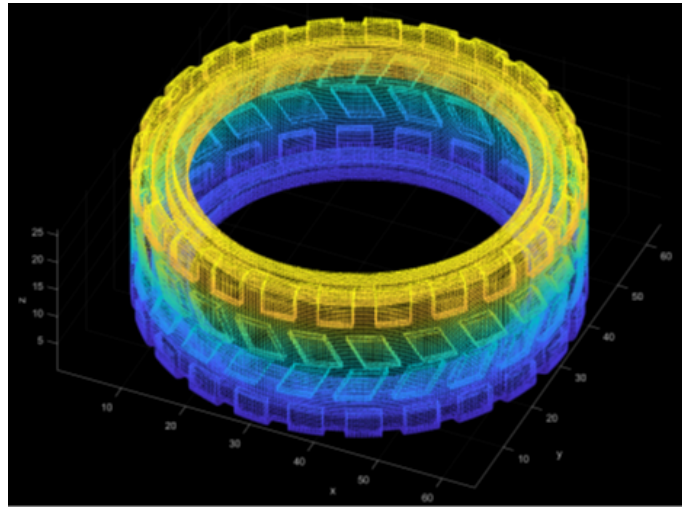


Figure 2.36: High-accuracy mesh model of the wheel

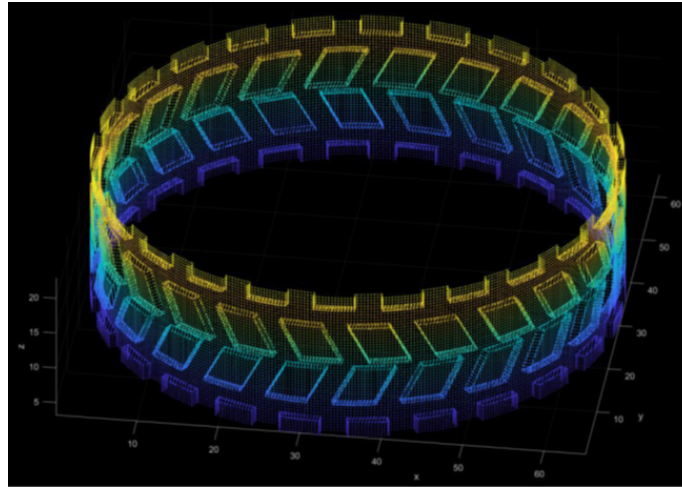


Figure 2.37: Medium-accuracy mesh model of the wheel

reduction in point density was necessary: by modifying the STL file tolerance from 0 mm to 40 mm, a point distribution with reduced density was obtained.

This increased tolerance resulted in a considerable reduction in the level of detail in the wheel geometry, but it also enabled the necessary number of points to be retained for the intended analysis. As a result, the point cloud was further reduced to 1,232 points, which resulted in a more manageable coordinate matrix. This significant reduction in points lowered the computational burden and improved simulation speed, thereby simplifying the process without compromising the analytical integrity or effectiveness of the study. The careful tolerance adjustment allowed for a balance between detail retention and computational efficiency, achieving an optimal solution for the specific context of the study. Once the last iteration of the point cloud has been generated (Fig. 2.38), the coordinate matrix can be imported into the Simulink model.

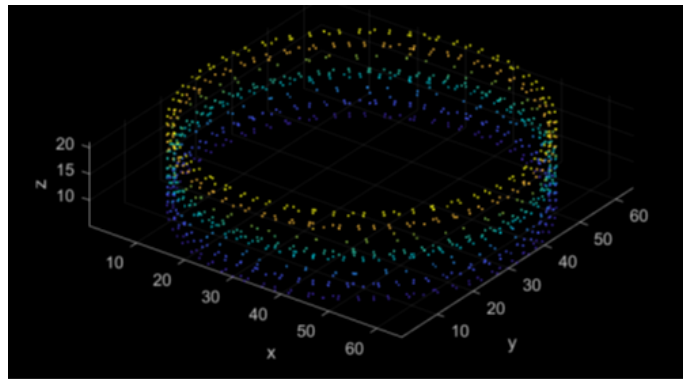


Figure 2.38: Low-accuracy mesh model of the wheel

In the Simulink model, the wheel is represented as illustrated in Fig. 2.39, where orange dots indicate the locations of points in the previously reduced and optimized point cloud. This visual representation provides insight into the manner in which the wheel interacts with the ground, thus enabling an assessment of the effectiveness of the simulation in regard to the grip and dynamic behavior of the wheel. The configuration of the wheel within Simulink, enhanced by the point cloud data, facilitates a more realistic and detailed simulation, enabling precise observation of the influence of variations in wheel geometry on its motion and interaction with the ground.

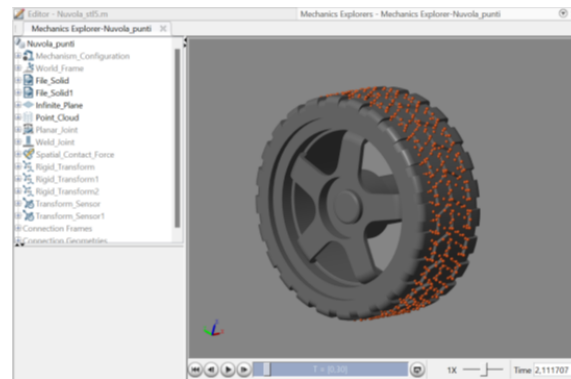


Figure 2.39: Point cloud representation of a wheel in Simulink

2.3.1.4 Conclusions

The multibody model developed for the rover represents a modular structure that combines physical and functional components, enabling advanced simulations in a virtual environment. Through the use of CAD and the precise definition of mechanical properties and joints, it was possible to obtain a realistic model that accurately simulates the dynamics and interactions of the system in diverse environmental contexts. The modeling of the DC motor, which integrates electrical and mechanical domains, allows for accurate analysis of motor performance under different loads, simulating realistic usage scenarios. This section provides the necessary foundation for advanced simulations, facilitating the future implementation of rover control strategies and enhancing the understanding of its operational dynamics.

2.3.2 Rover Movement Analysis and Simulations

This section presents a series of simulations of a prototype rover traveling on a variety of terrain types, with the goal of analyzing its dynamic behavior. By employing a multibody model developed within Simulink Simscape, it was feasible to virtually replicate the physical attributes of the rover, including the rocker-bogie passive suspension system. The simulations include straightforward tests and the crossing of obstacles, with the objective of collecting data on the dynamic behavior and stability of the rover. The outcomes of the simulations will then be compared with the results of the experimental tests on the actual prototype, as presented in Chapter 3, in order to verify the accuracy of the simulation model in predicting the expected behaviour under real operating conditions. Before of proceeding with the analysis of the simulations and their results, it is useful to provide some background on the concept of a signal and the appropriate methodology for its analysis.

2.3.2.1 Rectilinear Motion Simulations

The initial simulative tests were conducted with the aim of evaluating the forward rectilinear motion of the rover on a flat, uniform surface. Two distinct configurations were employed: one in which the "Bogie" was positioned at the front, and another in which the "Rocker" was positioned in the same location. In both configurations, the wheel model used is the Convex Hull for this part of the thesis.

In these tests, the masses of the rover were arranged on the top plate in accordance with the configuration of the physical prototype, and an identical rotational speed was applied to all wheels. The configuration resulted in the observation of straight-line motion for both the Bogie and Rocker configurations, with the aforementioned components positioned at the front. Hereafter, it provides a brief description of a scenario in which the multibody model, with the Bogie configuration at the front, was powered by motors at 6 V (255 PWM) (Fig. 2.40).



Figure 2.40: Multibody simulation of the rover in straight-line motion

As it possible to see in Fig. 2.41, the rover was observed to travel a distance of approximately three meters along the forward x-axis. An analysis of the translation plot along the y-axis revealed a transverse deviation of approximately 3 millimeters over the 3-meter distance. This deviation was considered to be of minimal consequence, allowing the motion to be regarded as essentially rectilinear. Among the potential causes of this

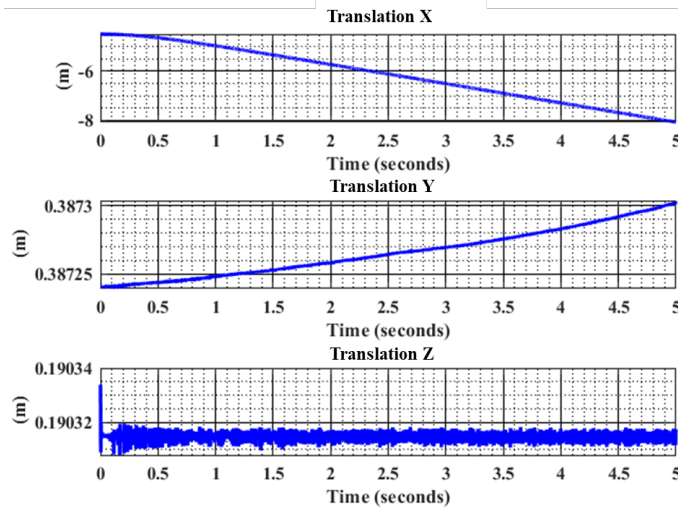


Figure 2.41: Center of gravity motion in the multibody model along X, Y, and Z axes over time.

minor discrepancy between the intended and actual trajectories, an imbalance is presumed, primarily due to the masses carried on the rover frame and the wiring masses, which appear significantly asymmetrical in the model. Nevertheless, the effect of this imbalance seems less significant in the model than in experimental tests, which will be discussed in greater detail in subsequent chapters.

2.3.2.2 Obstacle Crossing Simulations

Simulations conducted on the model indicated that, with a supply voltage for the electric motors set to 125 PWM (2.5 V), the rover successfully crossed a triangular obstacle with 7-centimeter sides when configured with the bogie at the front (Fig.2.42). In contrast, when the rocker is positioned at the front, the rover is unable to cross the step. In this configuration, the central wheels lose traction and begin to slip, resulting in an increase in their rotational speed. When the rover, configured with the bogie at the front, approaches the step (Fig. 2.42), it crosses it with minimal difficulty. This behavior is clearly observable in the plots (Fig. 2.43) generated from the multibody model, which show the displacement of the center of gravity of the top plate of the rover. The data illustrate a notable displacement along the x-axis (the forward axis), with an approximate magnitude of one meter, while variations along the y-axis remain minimal. Furthermore, the impact of the three wheels on the position of the top plate along the z-axis is also evident.

An analysis of the rotational speeds of the six wheels demonstrated that no increase could be attributed to slippage or subsequent loss of traction, as it is evident in Fig. 2.44.

In comparison, when the rover is configured with the Rocker at the front (Fig. 2.45) and approaches the step, both the simulated model and the real prototype—as will be discussed in the Chapter 3—finds difficulties in crossing the obstacle. This consistency in results suggests that, in contrast to the bogie configuration, the rocker configuration

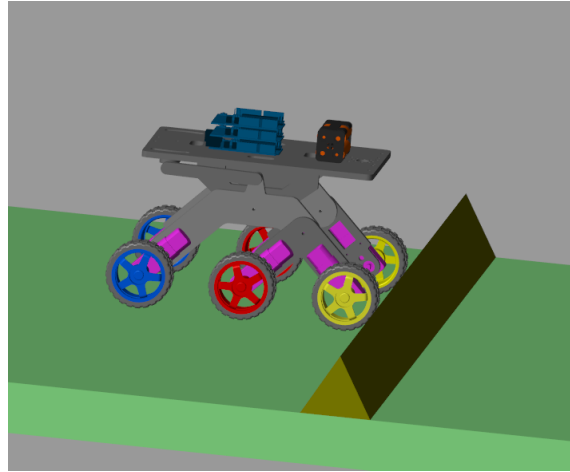


Figure 2.42: Multibody simulation of the rover crossing a triangular obstacle with Bogie Front

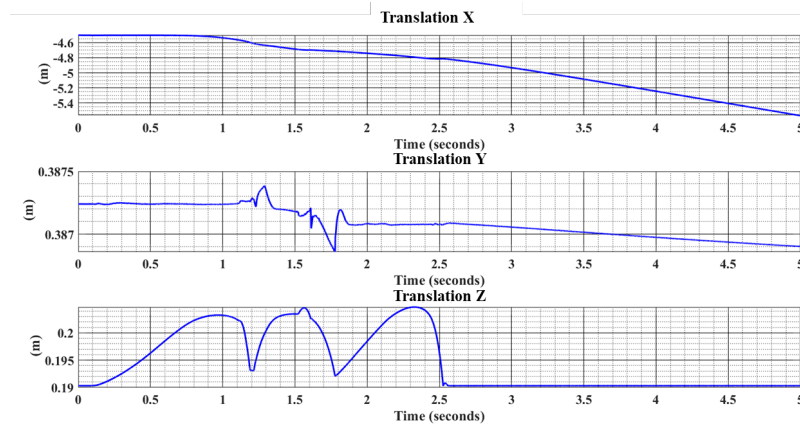


Figure 2.43: Translation of the center of gravity in the case of the Bogie Front

is not an effective approach for overcoming obstacles of this nature. The results of the multibody analysis, which demonstrate the rotational velocities of the wheels, reveal that the wheels situated on the central axle rotate at a considerably higher speed (Fig.2.46). This observation indicates the presence of slippage in the simulated model, which ultimately results in a loss of traction and prevents the rover from crossing the step. By increasing the voltage supplied to the motors of the model to a 3V, equivalent to 150 PWM, the multibody rover successfully passes the step in the Rocker at the front configuration. This outcome aligns precisely with what will be observed in the prototype, thus further confirming the accuracy and reliability of the multibody model.

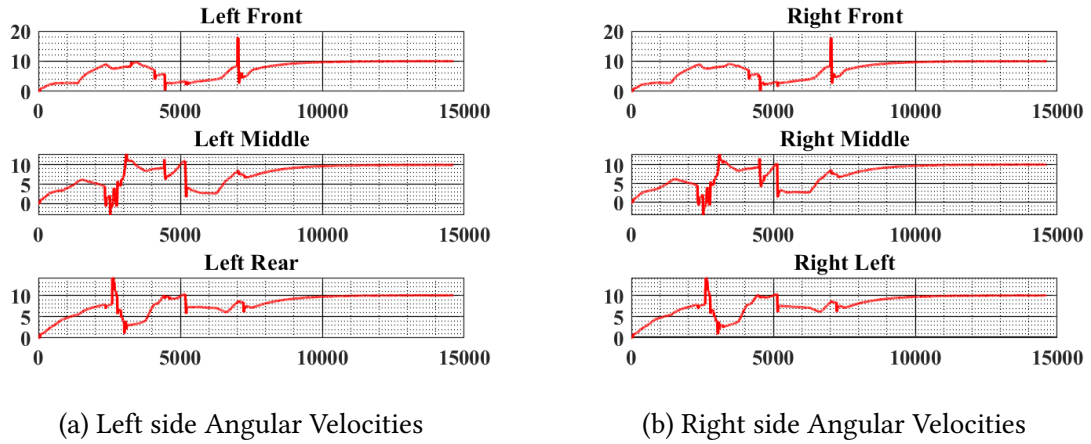


Figure 2.44: Angular Velocities - Bogie Front

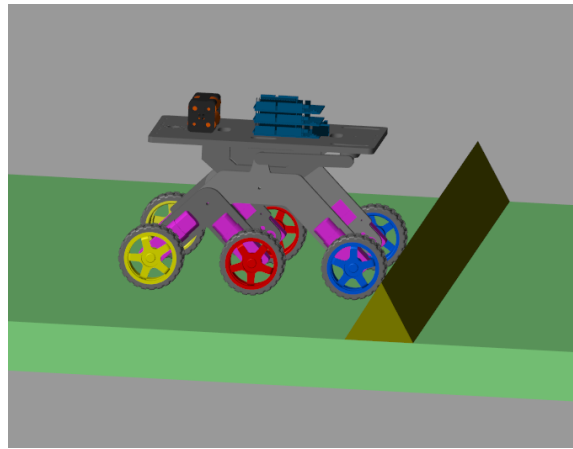


Figure 2.45: Multibody simulation of the rover crossing a triangular obstacle with Rocker Front

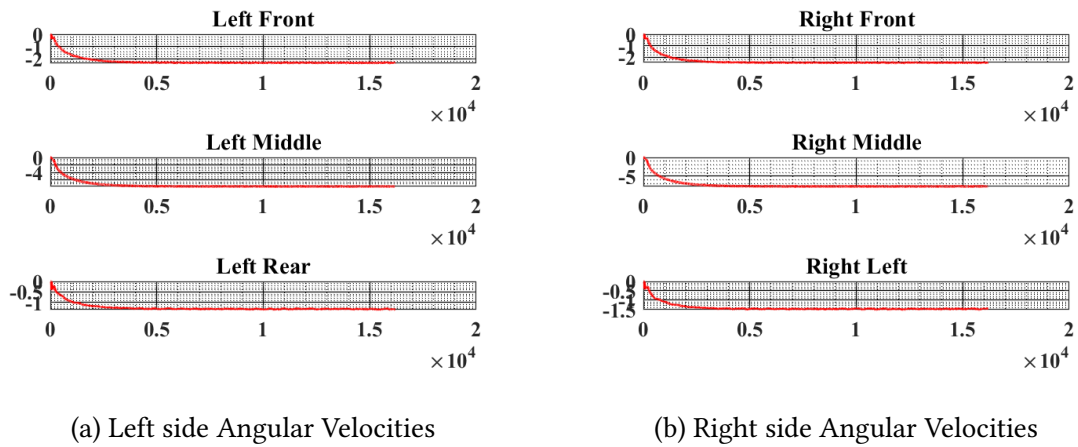


Figure 2.46: Angular Velocities - Rocker Front

2.3.2.3 Conclusions

The section explored the analysis of simulating the motion of a rover in different configurations, paying particular attention to straight-forward and obstacle-crossing configurations. Using the multibody model in Simulink Simscape, it was possible to accurately reproduce the dynamics of the system, including suspension behavior and wheel-ground interactions. The results show that the Bogie configuration enables more effective obstacle crossing than the Rocker configuration, which encounters more difficulties. These results support the accuracy of the simulative model and confirm its reliability for analysis and control of the real prototype, facilitating the development of future navigation and stability strategies on uneven terrain.

2.3.3 Vibration Analysis in Straight-Forward and Steering Configurations

This chapter presents a spectral vibration analysis of the small-size six-wheel rover. The analysis employs advanced multibody modeling and Fourier transform (FFT) techniques. The initial stage of the analysis focuses on a single wheel model, before extending to the entire rover in order to evaluate the wheel-ground interaction and the resulting acceleration signals. The influence of wheel speeds on the frequency distribution is investigated through the exploration of different advancing and steering configurations. The aim is to demonstrate how the obtained signals can be used to estimate wheel speeds, with implications for real-time control of the rover during the experimental phase. This could facilitate autonomous navigation and advanced control strategies.

A body exhibits oscillatory motion when it is in a state of vibration, with parameters such as amplitude and mean value that may vary over time. Vibration can be analyzed through the measurement of displacement, acceleration, and other related metrics. The present study is concerned with the analysis of acceleration, through the FFT, and the examination of its behaviour in response to varying conditions.

2.3.3.1 Single-Wheel Analysis

The tests were conducted on an ideal ground surface modeled as a parallelepiped and imported into the multibody modeling system as a CAD file. Initial analyses focused on a wheel modeled using the convex hull method, which neglects the presence of the 24 tire studs and instead considers a smooth cylinder in direct contact with the ground.

In the first simulation, the wheel was assigned a constant speed of 3 rad/s to observe the dynamics of its interaction with the ground and to analyze the resulting vibrations.

The spectrum in Fig.2.47 is obtained: From the zoom in Fig.2.48, the dominant

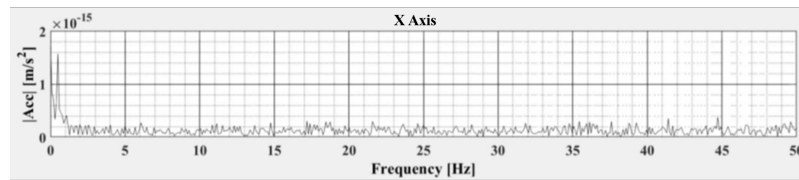


Figure 2.47: Frequency spectrum of a ConvexHull-Wheel at a rotational speed of 3 rad/s

frequency in the spectrum shows a peak at 0.488 Hz, which is in close alignment with the rolling frequency derived from the assigned speed of the wheel, which was determined to be 3 rad/s. Upon conversion to revolutions per second, this speed results in a frequency of 0.477 Hz, which is in close alignment with the observed value in the multibody simulation, confirming theoretical expectations.

Following, a point cloud was distributed over the entire pneumatic surface, with the objective of highlighting the 24 studs. This approach allows for the simulation of each contact point between the stud and the road surface, creating a more detailed and realistic model of the wheel-ground interaction.

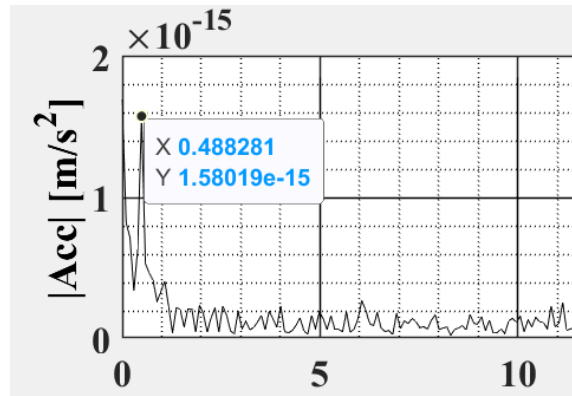


Figure 2.48: Zoom of Spectrum of a ConvexHull-Wheel at 3 rad/s

A series of simulations were conducted to identify the minimum number of points required in the point cloud to accurately capture the frequencies associated with the studs in the FFT (Fig. 2.49). It was observed that an excessive reduction in point density resulted in an inaccurate frequency spectrum, failing to align with the expected outcome. After several tests, it was determined that a minimum of 1232 points in the point cloud was necessary to ensure accurate analysis of the frequency spectrum. The analysis of the

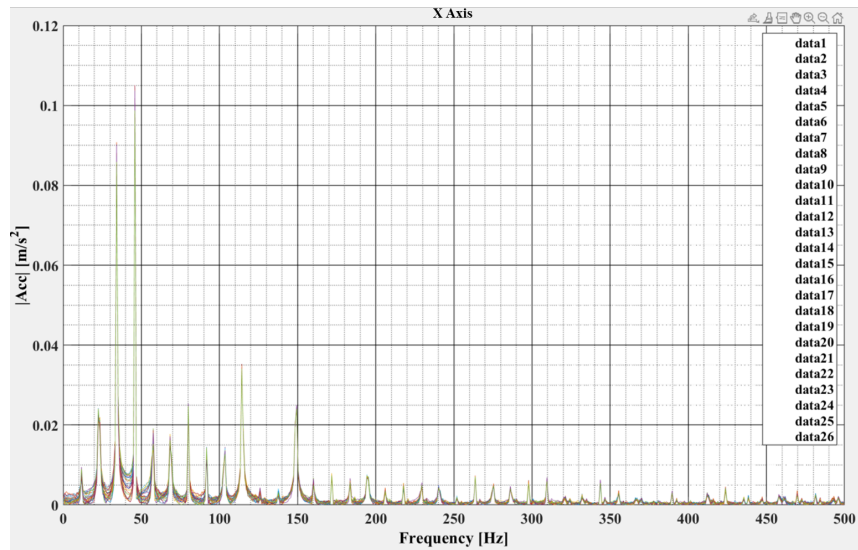


Figure 2.49: Frequency spectrum generated from point clouds in each simulation test

point cloud revealed the absence of a single peak corresponding to the rolling speed of the wheel. Instead, multiple peaks were observed. This phenomenon arises due to the fact that the use of the point cloud serves to accentuate the presence of each stud on the wheel, resulting in the generation of multiple peaks in the FFT that represent the 24 studs.

The relationship between the base rolling frequency of 0.48 Hz and the number of studs can be demonstrated by multiplying this frequency by the number of studs, which

corresponds to potential harmonics in the frequency response of the wheel. This approach offers insight into the contribution of each stud to the observed frequency spectrum:

$$0.48\text{Hz} \cdot 24 = 11.52\text{ Hz} \quad (2.33)$$

The calculation, which multiplies the rolling frequency of 0.48 Hz by the 24 studs, yields a frequency of 11.52 Hz. An examination of the FFT reveals that the first peak occurs at approximately 11.72 Hz, which is in close alignment with the theoretically calculated frequency. This alignment serves to confirm the direct influence of the studs on the observed frequencies. As anticipated, the following peaks are multiples of the base frequency. As shown in Fig.2.50, the second peak is observed at 22.46 Hz, which provides further evidence for the direct correlation between the physical structure of the wheel and the frequency spectrum observed in the FFT.

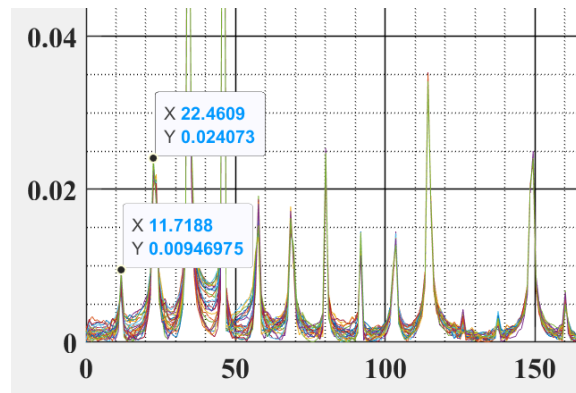


Figure 2.50: Zoom of Spectrum of a Point Cloud for each simulation test

In the two previous analyzed cases, the analysis was performed only on the single wheel. What should be analyzed now is the FFT of the whole rover, consisting of the 6 wheels.

2.3.3.2 Straight-Forward Configuration

It is essential that all six wheels of the rover rotate at the same speed in order for it to maintain straight-line motion. A series of simulations were performed within the multibody model, testing a range of wheel speeds to analyse their impact on the frequency spectrum. The wheels were modelled using the point cloud, which resulted in the introduction of specific peaks in the frequency spectrum. The aim of this study is to examine how variations in wheel speed influence the frequency spectra, with a particular focus on the different cases that have been analysed.

Angular Velocity: 10 rad/s

The first case analyzed is that of assigning a speed of 10 rad/s. In Fig.2.51, the frequency spectrum is showed.

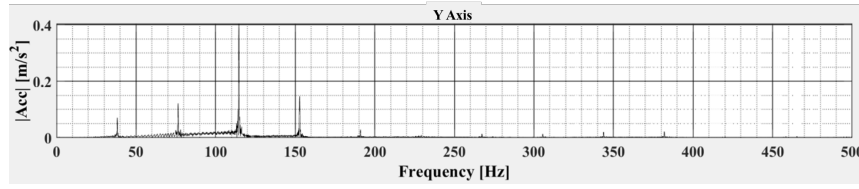


Figure 2.51: FFT at 10 rad/s along Y Axis

Given a speed of 10 rad/s, the initial peak in the frequency spectrum should align with the rolling frequency of the wheels at this velocity.

$$10 \text{ rad/s} = \frac{10 \text{ rev}}{2\pi} \text{ s} = 1.59 \text{ Hz} \quad (2.34)$$

The use of a convex hull to model the wheels of the rover would have resulted in the first peak in the frequency spectrum occurring at approximately 1.59 Hz. This indicates that the wheel motion exhibited a simpler and less detailed response due to the absence of studs in the model. However, as previously stated, a point cloud-based wheel model was selected in this study to more accurately represent the interaction between the wheel studs and the ground. This approach results in the generation of a more complex and informative frequency spectrum, facilitating a deeper understanding of the dynamic behaviour observed in the rover across a range of ground types.

Therefore, the first peak is given by:

$$1.59 \text{ Hz} \cdot 24 = 38.21 \text{ Hz} \quad (2.35)$$

where 24 is the number of studs on the outer tread of the wheel. Zooming in on the FFT shows the frequency relative to the first peak, in Fig.2.52. The first peak thus returns a

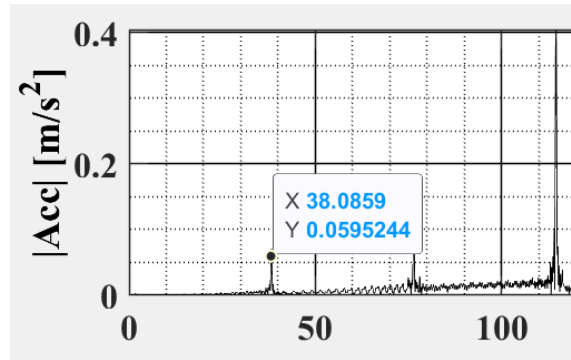


Figure 2.52: Zoom FFT at 10 rad/s

frequency of 38.08 Hz, and a value very close to the calculated one.

Angular Velocity: 20 rad/s

The following step is to set the wheels in motion at a rate of 20 rad/s.

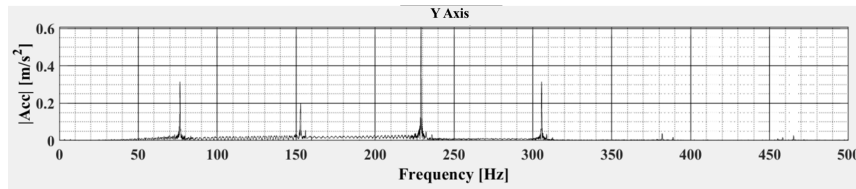


Figure 2.53: FFT at 20 rad/s

The corresponding frequencies are calculated:

$$20 \text{ rad/s} = \frac{20 \text{ rep}}{2\pi} \text{ s} = 3.18 \text{ Hz} \quad (2.36)$$

that:

$$3.18 \text{ Hz} \cdot 24 = 76.43 \text{ Hz} \quad (2.37)$$

By comparing this value with the corresponding value obtained from the FFT, it is possible to confirm, as it has seen in Fig. 2.54 that a very close value is obtained.

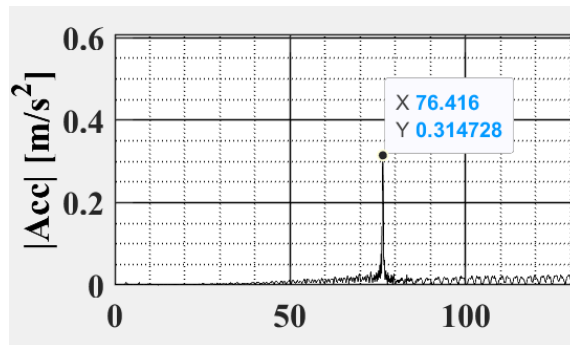


Figure 2.54: Zoom FFT at 20 rad/s

Angular Velocity: 30 rad/s

The final analysis is to assign a velocity of 30 rad/s.

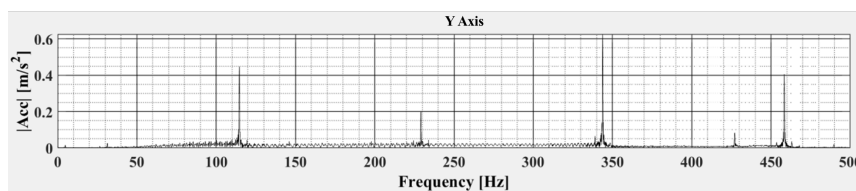


Figure 2.55: FFT at 30 rad/s

The corresponding frequencies are calculated:

$$30 \text{ rad/s} = \frac{30 \text{ rep}}{2\pi} \text{ s} = 4.77 \text{ Hz} \quad (2.38)$$

and then:

$$4.77 \text{ Hz} \cdot 24 = 114.64 \text{ Hz} \quad (2.39)$$

A comparison of the assigned speed with the corresponding value obtained from the FFT once again confirms the observation of two very similar values, validating the consistency of the simulation results with theoretical expectations.

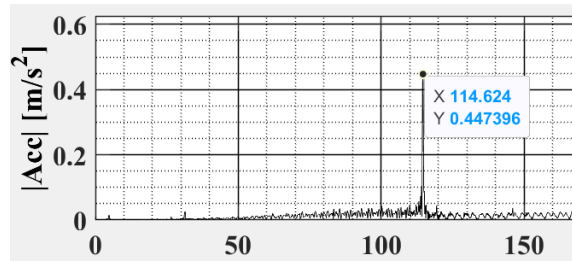


Figure 2.56: Zoom FFT at 30 rad/s

2.3.3.3 Steering Configuration

In order to steer the rover in a leftward or rightward direction, it is necessary to apply a differential control to the wheel speeds on the two banks. In order to effect a rightward turn, the speed of the wheels on the left bank must be increased; to turn left, the speed of the wheels on the right bank must be increased. This approach makes use of the differential drive principle to generate the requisite steering torque for changing direction.

Angular Velocities: 15 and 10 rad/s

In this case, the left bank was assigned a speed of 15 rad/s and the right bank was assigned a speed of 10 rad/s. During this analysis of the turning rover, the FFT shows a

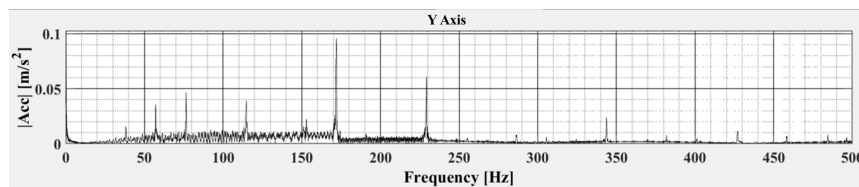


Figure 2.57: FFT at 15 and 10 rad/s

greater number of peaks compared to the uniform velocity configuration (Fig. 2.57). This increase is due to the fact that each of the two wheel banks now has a different velocity, rather than a single constant velocity for all of them. In the FFT, the peaks corresponding to the different wheel speeds are expected to be clearly visible. Analyzing the first two peaks, one should correspond to a frequency derived from a rolling speed of 15 rad/s and the other to a frequency derived from a rolling speed of 10 rad/s. These frequencies

directly represent the different speeds set for the wheel banks, reflecting the steering intent of the rover:

$$\begin{aligned} 10 \text{ rad/s} &= \frac{10 \text{ giri}}{2\pi} \text{ s} = 1.59 \text{ Hz} \\ 15 \text{ rad/s} &= \frac{15 \text{ giri}}{2\pi} \text{ s} = 2.38 \text{ Hz} \end{aligned} \quad (2.40)$$

then:

$$\begin{aligned} 1.59 \text{ Hz} \cdot 24 &= 38.21 \text{ Hz} \\ 2.38 \text{ Hz} \cdot 24 &= 57.32 \text{ Hz} \end{aligned} \quad (2.41)$$

A more detailed examination (Fig.2.58) of the initial two frequencies in the analysed spectrum demonstrates that these values are in close alignment with the anticipated frequencies, confirming the accuracy of the frequency response predicted by the model.

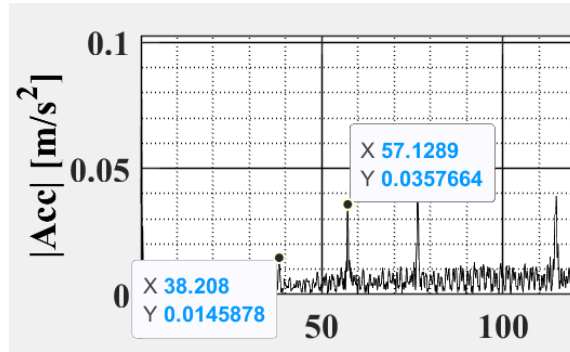


Figure 2.58: Zoom in FFT at 15 and 10 rad/s

Angular Velocities: 15 and 5 rad/s

In the second analysis, the rover was configured for a tighter turn, with significantly different speed assignments. The left wheel bank was assigned a velocity of 15 rad/s, while the right wheel bank was assigned a rate of 5 rad/s. This notable discrepancy in velocity serves to enhance the steering effect.

The resulting FFT plot for this configuration should display two primary sets of peaks, each corresponding to the rolling frequencies derived from the 15 rad/s and 5 rad/s speeds, respectively. In addition to these fundamental peaks, multiples of these frequencies will also be present. These peaks and their harmonics reflect the rotational differences between the two wheel banks, providing a detailed depiction of how the differential velocities influence the trajectory of the rover.

Mathematically it can be verified as:

$$\begin{aligned} 15 \text{ rad/s} &= \frac{15 \text{ giri}}{2\pi} \text{ s} = 2.38 \text{ Hz} \\ 5 \text{ rad/s} &= \frac{5 \text{ giri}}{2\pi} \text{ s} = 0.79 \text{ Hz} \end{aligned} \quad (2.42)$$

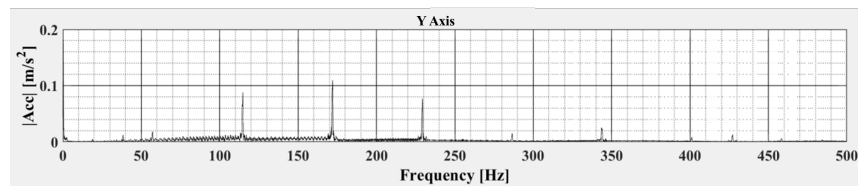


Figure 2.59: FFT at 15 and 5 rad/s

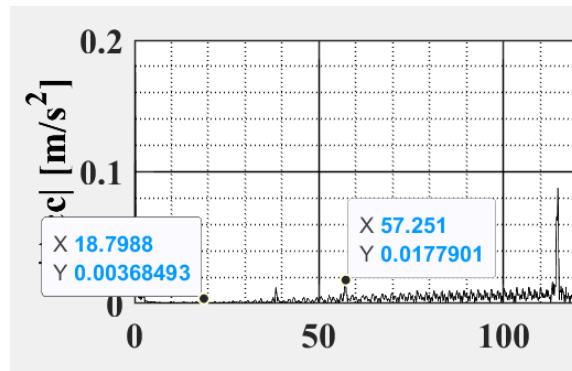


Figure 2.60: Zoom in of FFT at 15 and 5 rad/s

then:

$$\begin{aligned} 2.38 \text{ Hz} \cdot 24 &= 57.32 \text{ Hz} \\ 0.79 \text{ Hz} \cdot 24 &= 19.11 \text{ Hz} \end{aligned} \quad (2.43)$$

The values in question are approximately equivalent to those observed in the Fig. 2.59 and better in Fig.2.60.

2.3.3.4 Conclusions

This section explored the spectral analysis of the six-wheel rover vibration in the straight forward and steering configurations, employing a multibody model and the Fourier transform (FFT). After an initial analysis on a single wheel, the investigation was extended to the entire rover, using point cloud models to accurately simulate wheel-ground interaction and monitor vibration frequencies arising from different wheel speeds. The results demonstrate a strong correlation between the physical structures of the wheels and the observed frequency peaks, confirming the accuracy of the model in reproducing the physical dynamics of the system. The relevance of straight-forward and steering configurations suggests the opportunity to use these data for real-time control of the rover, leading the way for implementing advanced autonomous navigation strategies and improving the ability of the rover to adapt to different terrain surfaces.

2.3.4 Integration of Robotic Arm for Obstacle Crossing

In general, mobile robots tend to avoid obstacles when they are encountered. An analysis of their ability to overcome these challenges is likely to prove of interest. One of the primary systems embedded in the rover is the robotic arm, which is essential for operations such as waste collection and weed cutting. This section examines the potential of employing the robotic arm to cross obstacles, with an analysis of the advantages and disadvantages of this approach. In contrast to some rovers described in the literature, which modify their configuration or design to overcome obstacles, this model employs a robotic arm to face these challenges. This approach is entirely distinct and does not affect the design of the rover, including the rocker-bogie suspension system. As a result though, some of the main advantages of the original design are lost.

It is proposed an analysis conducted on the rover system that is equipped with a robotic arm. The aim of the study is to cross an obstacle whose height is at least equal to the radius of the wheel of the rover. This specification is not trivial, as previous research has demonstrated that the rocker-bogie suspension system has not encountered difficulties when crossing obstacles shorter than the wheel radius. In this section of the study, the hypothesis is that the bogie is the first component to come into contact with the obstacle. The objective of the study is to generate a dynamic imbalance in the rover in order to overcome the obstacle. This is achieved by shifting the position of the center of gravity from its initial "rest" condition, which is located at the center of the upper plate of the rover, to a configuration that allows one of the wheels of the bogie to have no ground reaction, thus facilitating the climb up (87).

2.3.4.1 Analysis of Integration Feasibility

The schematic presented in Fig.2.61 considers the presence of an imbalance, which, as previously mentioned, causes a shift in the center of gravity. It is assumed that all the mass is concentrated at the center of gravity: therefore, it no longer coincides with point G but is at a distance indicated as e . To simplify the diagram, the weight force is positioned at point G , and the transport moment We is added as required.

No Obstacle analysis

The schematic in Fig. 2.62 does not account for the presence of an obstacle. The objective is to assess how the reactive forces vary with changes in the eccentricity (e), defined as the distance at which the center of gravity of the rover is positioned to facilitate the imbalance.

By solving the structure, for balance in horizontal translation:

$$R_{Ex} = 0 \quad (2.44)$$

By solving the equilibrium of rotation at pole C and the equilibrium of vertical translation, as referenced in the EDC system, the structure can be solved. The resulting equation is as

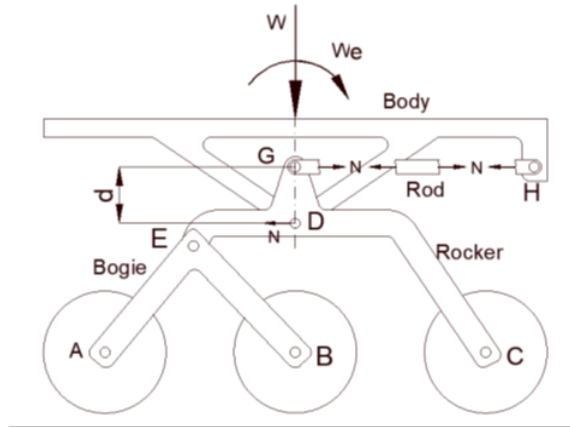


Figure 2.61: Schematic diagram of the rover showing the imbalance caused by the robotic arm

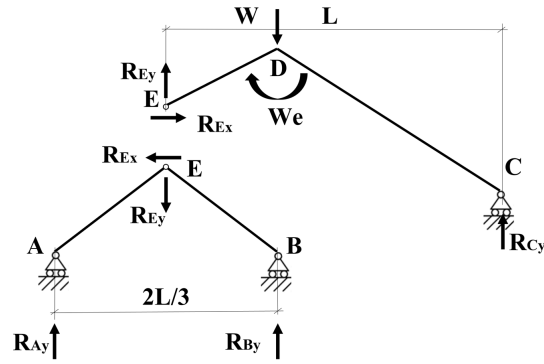


Figure 2.62: Rocker-bogie reaction forces in the absence of obstacles

follows:

$$\begin{cases} -R_{Ey}L + W\frac{2}{3}L - We = 0 \\ W - R_{Ey} - R_{Cy} = 0 \end{cases} \quad (2.45)$$

and:

$$\begin{cases} R_{Ey} = W\left(\frac{2}{3} - \frac{e}{L}\right) \\ R_{Cy} = W\left(\frac{1}{3} + \frac{e}{L}\right) \end{cases} \quad (2.46)$$

Upon returning to the AEB system and recalling that the hinge reaction for equilibrium changes sign from Fig. 2.62, it is possible to rewrite the equilibrium at rotation in pole B and the equilibrium at vertical translation:

$$\begin{cases} -\frac{2}{3}R_{Ay}L + \frac{1}{3}R_{Ey}L = 0 \\ R_{Ey} - R_{By} - R_{Ay} = 0 \end{cases} \quad (2.47)$$

Substituting R_{Ey} calculated in 2.46 gives:

$$\begin{cases} R_{Ay} = W \left(\frac{1}{3} - \frac{e}{2L} \right) \\ R_{By} = W \left(\frac{1}{3} - \frac{e}{2L} \right) \end{cases} \quad (2.48)$$

In conclusion, it can be stated that:

$$\begin{cases} R_{Ay} = \frac{W}{3} \left(1 - \frac{e}{p} \right) \\ R_{By} = \frac{W}{3} \left(1 - \frac{e}{p} \right) \\ R_{Cy} = \frac{W}{3} \left(1 + \frac{2e}{p} \right) \end{cases} \quad (2.49)$$

The Fig.2.63 shows how the reactions, calculated using equation 2.49, change in response of changing in e . It can be observed that R_{Ay} and R_{By} are equal, resulting in the two reactions overlapping on the graph.

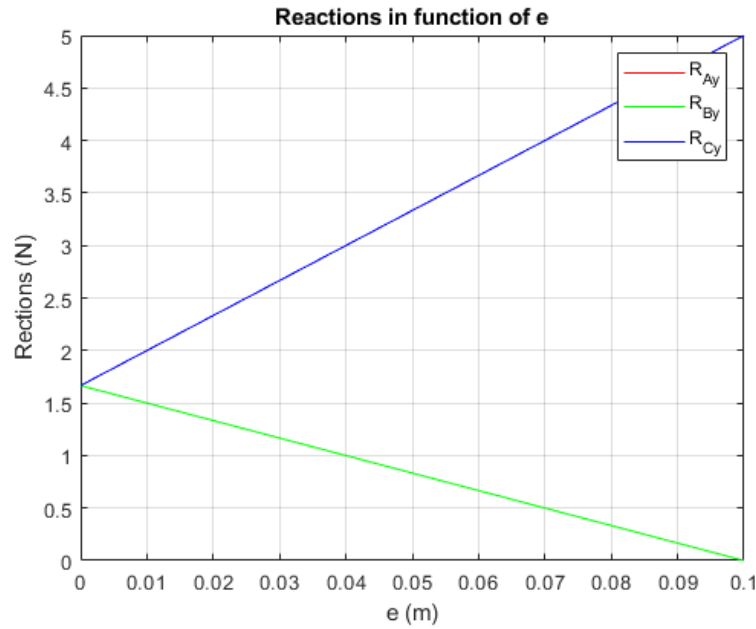


Figure 2.63: Reaction forces as a function of eccentricity (e)

Obstacle analysis

In this phase of the analysis, it is proposed that the ground reactions be studied in the presence of an obstacle with a height at least equal to the wheel radius, as said before. It is observed that another fundamental factor influencing the obstacle crossing by the rover is the friction force. In this study, a static friction coefficient of 0.4 was assumed.

In the static analysis that follows, the torque of the wheels has also been considered. It is represented as a force applied to a lever arm, specifically the radius of the wheels, as it undoubtedly plays a significant role in the system under study.

Wheel A in contact with the obstacle

This analysis assumes that the rover approaches the step obstacle with the bogie front. As it approaches, the four rear wheels drive the front wheels against the step. This action makes the front wheels rotate upwards, lifting the front of the rover and enabling it to climb the obstacle (Fig. 2.64). The diagram shows the structure of the rover, the forces acting on it and the main geometric parameters.

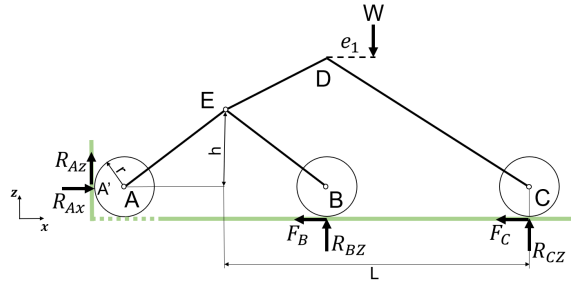


Figure 2.64: Configuration of the rover as it approaches the obstacle with Wheel A

It can be concluded that the thrust forces (F_B and F_C) generated by the center and rear drive wheels help to push the front wheel against the obstacle. In addition, the frictional force at contact point A' acts against the vertical load on wheel A, preventing it from slipping downwards.

$$R_{Az} \leq f R_{Ax} = f(F_B + F_C) \quad (2.50)$$

At the same time, to prevent the middle and rear wheels from slipping, it is necessary to ensure that the thrust forces remain below the grip limit:

$$F_B + F_C \leq f(R_{Bz} + R_{Cz}) \quad (2.51)$$

By solving the scheme in Fig.2.64, the following reaction forces can be obtained:

$$\begin{aligned}
 R_{Az} &= \frac{\frac{F_c}{3}(2h - r) - F_B r + W \left(\frac{2L}{9} - \frac{e_1}{3} \right)}{\frac{2}{3}L + r} \\
 R_{Bz} &= \frac{R_{Az}r + \frac{L}{3} \left[W \left(\frac{2}{3} - \frac{e_1}{L} \right) - \frac{F_c(r+h)}{L} \right] - F_c h + F_B r}{\frac{2}{3}L} \\
 R_{Cz} &= \frac{F_c(r+h)}{L} + W \left(\frac{1}{3} + \frac{e_1}{L} \right)
 \end{aligned} \tag{2.52}$$

By replacing R_{az} in 2.50 the eccentricity value necessary for the front wheel to lift and for the first pair of wheels to begin climbing can be determined.

$$e_1 = \frac{-9F_b r + 6F_c h - 3F_c r + 2LW - 6Lf(F_b + F_c) - 9rf(F_b + F_c)}{-3W} \tag{2.53}$$

After calculating the reactions, it is essential to evaluate their evolution as a function of the eccentricity e_1 in order to study their effect on the obstacle crossing.

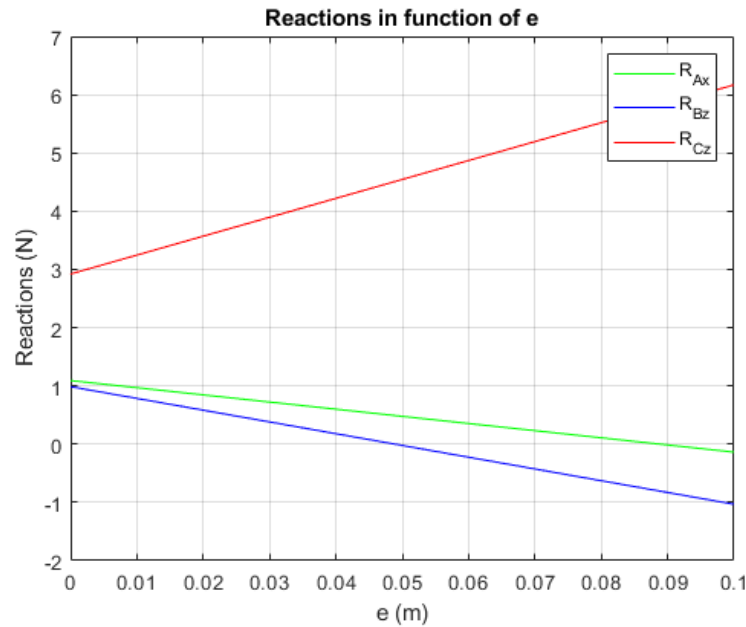


Figure 2.65: Reaction forces as a function of eccentricity (e) while Wheel A approaches the obstacle

From the plot in Fig.2.65 the first reaction to go to zero as the distance e increases is R_{Bz} . Usually, when the rover crosses an obstacle that is shorter than the radius of its wheels, the first wheel in contact with the obstacle will be the first to lift to pass over it; however according to the reaction forces in Fig. 2.65, a non-linear motion of the unbalanced rover

leads to the unloading of the central wheel, as shown in Fig. 2.66: this happens because as e_1 increases, the center of mass shifts and the reaction R_{Bz} becomes zero. At the same time, the reaction R_{Cz} increases, which increases the grip of the rear wheel. The higher traction allows the rear wheel to provide more thrust, allowing wheel A to successfully cross the obstacle.

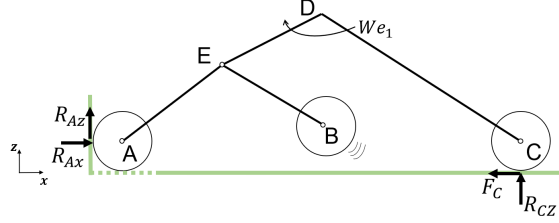


Figure 2.66: Nonlinear motion causing the unloading of the central wheel

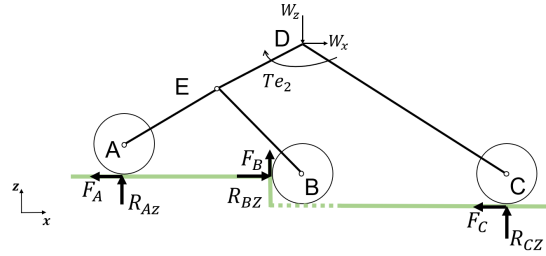


Figure 2.67: Configuration of the rover as it approaches the obstacle with Wheel B

Wheel B in contact with the obstacle

When wheel A has cleared the obstacle, wheel B approaches and makes contact with the step. This continues until the central wheels are also lifted and begin to clear the obstacle. In the configuration shown in Fig.2.67, wheel B is in static contact with the obstacle. In order for wheel B to overcome the obstacle, the condition in (2.54) must be satisfied:

$$F_B \geq f R_{Bx} \quad (2.54)$$

For the evaluation of R_{Bx} it is necessary to solve the equilibrium in Fig.2.67. As a result of the configuration, the kinematics have changed. As a result, the kinematics formula calculated in the Kinematics analysis using the Fig. 2.6 must be applied, resulting in the Tab. 2.7:

In the Fig. 2.68, the geometric distances of the rover are parameterized in order to write the equilibrium equations and are the following:

- b : the vertical distance from the point D, where the load is typically applied, to the wheel center A

$\theta_1(\text{deg})$	$\theta_2(\text{deg})$	$\theta_3(\text{deg})$	$\theta_4(\text{deg})$	$\psi(\text{deg})$
26.2	116	139	18.9	7.37

Table 2.7: Suspension angle configuration with the robotic arm as Wheel B approaches the obstacle

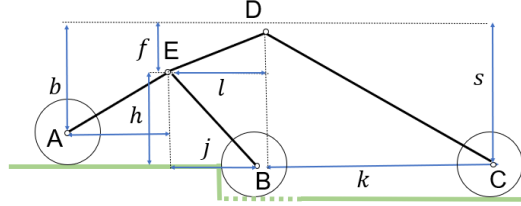


Figure 2.68: Parameters of the rover configuration as it approaches the obstacle with Wheel B

- h : the vertical distance from the hinge E to the wheel center B
- a : the horizontal distance from the hinge E to the wheel center A
- j : the horizontal distance from hinge E to the wheel center B
- l : the horizontal distance from hinge E to the point D
- k : the horizontal distance from the wheel center C to the point D
- s : the vertical distance from the wheel center C to the point D
- f : the vertical distance from the point D to the hinge E

Before computing R_{Bx} , it is essential to consider an aspect of Fig. 2.67. It will be seen that the load is no longer simply the vertical load due to the weight W , but is divided into two components. This is due to the fact that in this configuration the plate is tilted, typically by a very small angle of pitch, giving rise to the horizontal and vertical components:

$$W_x = W \sin \alpha \quad (2.55)$$

$$W_z = W \cos \alpha \quad (2.56)$$

$$T = W_x \tan \alpha + W_z \quad (2.57)$$

In this context, α represents the pitch angle of the rover's plate.

For the evaluation of R_{Bx} , Fig.2.68 shows a scheme enriched with dimensions relative to the wheel B approaching the obstacle, necessary to solve Fig.2.67. This is achieved by starting from the structure and evaluating R_{Ez} using the rotational equilibrium around pole A.

$$\begin{cases} R_{Ez} = \frac{W_x s + W_z k - T e - F_A r - R_{Ex} h}{k + l} \\ R_{Ex} = W_x + F_A \end{cases} \quad (2.58)$$

while the diagram AEB is solved using the rotational equilibrium about the pole C:

$$R_{Bx} = \frac{-F_B(j + a - r) - R_{Ex}b + R_{Ez}a + F_c}{r} \quad (2.59)$$

Substituting eq. 2.59 into eq. 2.54, the value of e_2 that satisfies the above equation can be obtained.

$$\begin{aligned} e_2 = \frac{1}{afT} \bigg[& -F_B[r(k+l) + af(k+l+j-r)] \\ & + af[W_x(s-h) + W_zk - F_A(h+r)] \\ & + f[F_C(k+l) - W_xb(k+l) - F_Ab(k+l)] \bigg] \end{aligned} \quad (2.60)$$

Wheel C in contact with the obstacle

When the rear wheels have reached the obstacle, they are pulled over it by the four front wheels (Fig. 2.69).

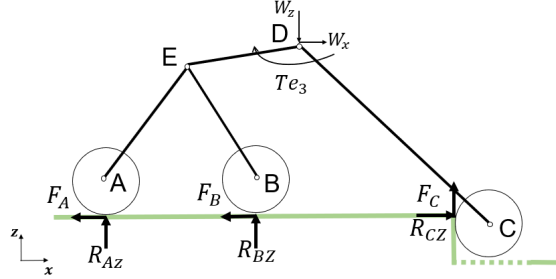


Figure 2.69: Configuration of the rover as it approaches the obstacle with Wheel C

The purpose is to find the value of eccentricity required to overcome the obstacle. As before, the condition for climbing is as follows in eq. 2.61.

$$F_C \geq f R_{Cx} \quad (2.61)$$

Solving the eq. 2.61, it is essential to have a clear understanding of the kinematics of the system under consideration. For the scheme in Fig. 2.69, the Tab. 2.8:

$\theta_1(\text{deg})$	$\theta_2(\text{deg})$	$\theta_3(\text{deg})$	$\theta_4(\text{deg})$	$\psi(\text{deg})$
45.0	135	131	11.2	33.8

Table 2.8: Suspension angle configuration with the robotic arm as Wheel C approaches the obstacle

When determining the eccentricity value, it is first required to calculate R_{Cx} . This is achieved by solving the EDC scheme (part of the scheme in Fig. 2.69), applying the conditions of rotational equilibrium around pole E:

$$R_{Cx} = \frac{-F_c(k + l - r) + T e_3 + W_z l - W_x(s - h)}{h} \quad (2.62)$$

in this way, R_{Cx} is function of e .

Replacing eq. 2.62 in eq. 2.61, the value of eccentricity can be defined for which this condition is satisfied.

$$e_3 = \frac{F_C(h + f k + f l - f r) + f(W_x(s - h) - W_z l)}{T f} \quad (2.63)$$

In obstacle climbing, the rover will slow down or even stop if the height of the obstacle is equal to or greater than the radius of the wheel.

Tilting equilibrium analysis

This section deals with a particular condition that can occur during the crossing sequence, which could potentially lead to the rover overturning. The purpose of this section is to identify the value of e_{lim} that will keep the rover in balance. As can be seen in the figure.2.70, it is assumed that the wheels of the bogie are at a height greater than the radius of the wheels, at most equal to the diameter of the wheels.

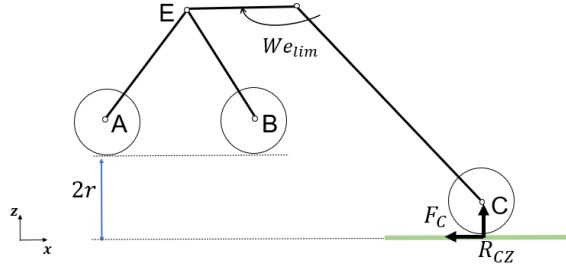


Figure 2.70: Configuration of the rover as it approaches the obstacle with Wheel C, avoiding tipping

$$\begin{cases} h_1 = 66 \text{ mm} \\ h_2 = 66 \text{ mm} \\ h_3 = 0 \text{ mm} \end{cases} \quad (2.64)$$

According the Kinematic Scheme in Fig. 2.6, the Tab. 2.9 can be achieved:

$\theta_1(\text{deg})$	$\theta_2(\text{deg})$	$\theta_3(\text{deg})$	$\theta_4(\text{deg})$	$\psi(\text{deg})$
45.0	135	118.36	-1.64	46.64

Table 2.9: Suspension angle configuration with the robotic arm in a tipping condition

When analysing this critical situation, the triangle AEB can be said to be unloaded, as evidenced by the moments preceding the overturning. In this context, after studying the EDC structure and applying the conditions of equilibrium to the vertical translation and rotation around pole E, the system of equations is formulated as follows:

$$\begin{cases} R_{Cz} = W \\ R_{Cz}(k+l) - F_c(s-f+r) - W e_{lim} - W l = 0 \end{cases} \quad (2.65)$$

where e_{lim} is obtained as follow:

$$e_{lim} = k - \frac{F_c(s-f+r)}{W} \quad (2.66)$$

The analysis is important in order to define the configuration of the robotic arm in the detailed model simulation that will be presented later, in order to avoid the overturning of the entire mobile robot system in the crossing.

2.3.4.2 Simulation Processes

This study examines the effects of adding a robotic arm, proportionate to the rover, on the overall stability of the system (the combined rover and arm system), thereby facilitating obstacle crossing. The purpose of this section is to investigate the logical and functional structure of the robotic arm mounted on the rover, which has been specifically designed to optimize dynamic imbalance and ensure effective obstacle crossing. Both the robotic arm and the rover were simulated using Matlab® Simulink Simscape Multibody.

Robotic Arm Structure

The structure of the multibody model of the robotic arm is analyzed in the Fig. 2.71. The model of the robotic arm, which will be employed during the prototyping phase,

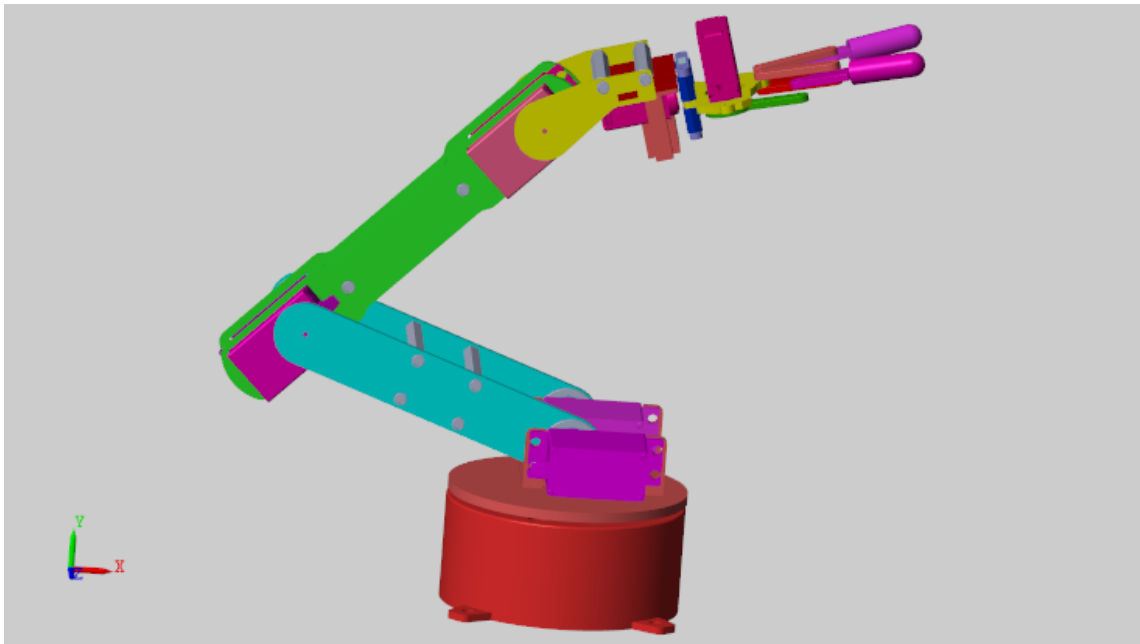


Figure 2.71: Multibody model of the robotic arm in Simscape

was initially designed using Solidworks and later imported into Simscape for further simulations. This model represents the Lynx 6, a product by LynxMotion, known for its applications in small robotic manipulators. The choice of this model is based on its reliability and operational flexibility, which are essential aspects for testing the functionalities of the robotic arm during the several steps of the later prototyping of the entire rover-arm system.

The technical specifications of the robotic arm will be examined in detail in the experimental section (Chapter 3). For the time being, however, the analysis of the multibody model will proceed. This model is a crucial element for future simulations and for validating the performance of the system.

The model discussed represents only a portion of the comprehensive model used in simulations, which includes an analysis of the combined rover-robotic arm structure. The analysis is particularly focused on the position assumed by the robot to facilitate unbalancing. However, it is essential to note that this model constitutes a single logical block within a larger and more complex structure.

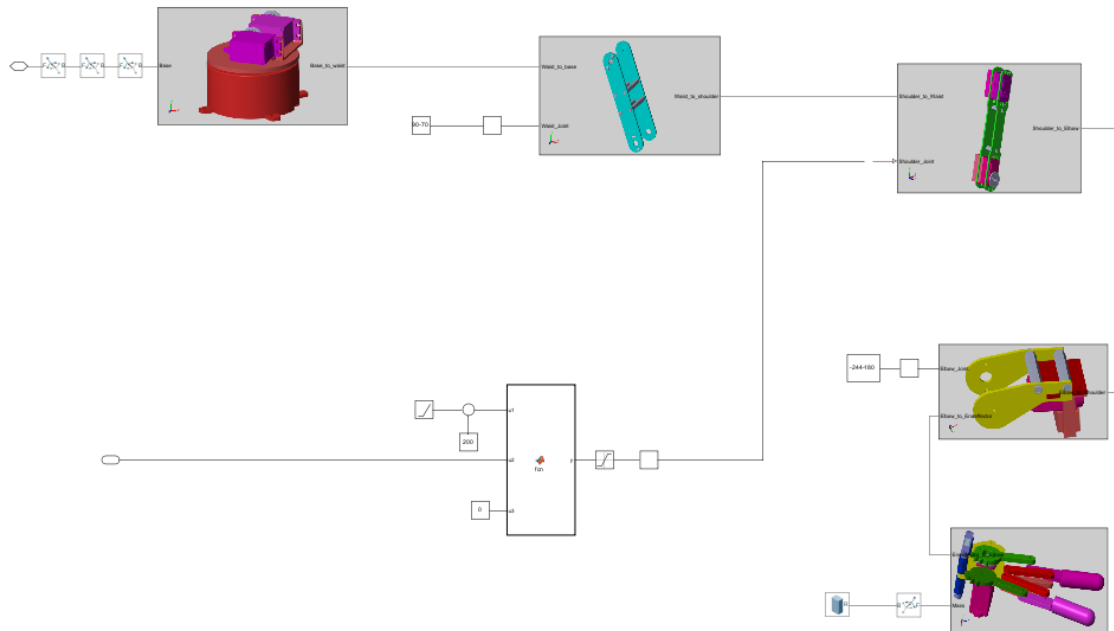


Figure 2.72: Simscape model of a robotic arm

The robotic arm Simscape model in Fig.2.72 has been divided by five parts to logically reflect the robotic arm in reality:

- *Base*: The base of a robotic arm is the stationary foundation that supports the arm and allows it to rotate. It typically includes a motor and gear assembly that provides the rotational motion necessary for the entire arm to move around a fixed point.
- *Waist*: The waist of a robotic arm is a joint that connects the base to the arm, allowing the arm to pivot or rotate horizontally. This movement can provide the arm with the ability to reach objects in a wide arc without changing its location.
- *Shoulder*: The shoulder is a joint that attaches the main upper arm to the waist and enables the arm to move in an upward and downward arc. This joint typically allows for a wide range of motion, similar to a human shoulder.
- *Elbow*: The elbow joint connects the upper arm (above the elbow) to the forearm (below the elbow) of the robotic arm. This joint allows the arm to extend and retract, similar to the bending and straightening of a human arm's elbow.

- *Endeffector*: The endeffector is the tool attached to the end of the robotic arm. It can be a gripper, a welding torch, a camera, or any other device designed for specific tasks. The endeffector is crucial as it interacts directly with the environment, performing tasks based on the arm's design.

It is crucial to recognize that not all joints will be employed to modify the overall mechanical configuration of the system. This decision is strategic for the dynamics of the system. In particular, the ability to modify the overall configuration to overcome an obstacle by adjusting only the "shoulder" joint allows for a considerable reduction in the complexity of the manipulative process itself, as well as a significant decrease in the complexity of the analysis. Modifying the configuration through a single joint allows for the study of the effects of dynamics on minor structural changes, demonstrating how slight variations can have a significant impact on the balance and functionality of the system. The purpose of this thesis is not to provide a comprehensive analysis of each block in the Simscape model. However, it is crucial to highlight that by varying the position of the specified joint, the configuration of the system can be modified to effectively overcome obstacles.

2.3.4.3 Prototype Simulations

The simulation within the Simscape environment is examined initially through a preliminary empirical analysis to confirm whether the rover is capable of overcoming an obstacle through dynamic imbalance. Subsequently, the analysis is refined using a more analytical approach. In accordance with the rocker-bogie suspension concept, it is important to note that this simulation, like all simulations conducted in this thesis, is performed at moderate speeds. This approach, combined with the modest mass of the involved bodies (for the small prototype of the rover), ensures that the inertias are small and therefore quite negligible.

Simplified Simulation Model

The simulation process starts with a preliminary simplified multi-body model to evaluate the ability of the rover to cross an obstacle through an imbalance. A detailed model is then developed. In the simplified model, the robotic arm is modelled as two cylindrical rods connected by a prismatic joint and a sphere at the end of the free end of the entire link. The base of the robotic arm is connected to the rover by a revolute joint. For simplicity, the rod has been placed at the edge of the plate of the rover (Fig. 2.73). The arm has a revolute joint at the base, shown in red, and a prismatic joint at the joint between the two links that make up the arm. The whole simulation was carried out in quasi-static behaviour. Due to the small amount of moving mass in the whole system, the effect of inertia can be considered as minimal. In fact, it is crucial to emphasise that in this particular simulation, the moving mass (rod-ball) represents less than 20 % of the total mass of the system.

The rover, equipped with this robotic arm, appears as shown in Fig. 2.73.

The results of the simulations are presented in Fig. 2.74. 2.73:

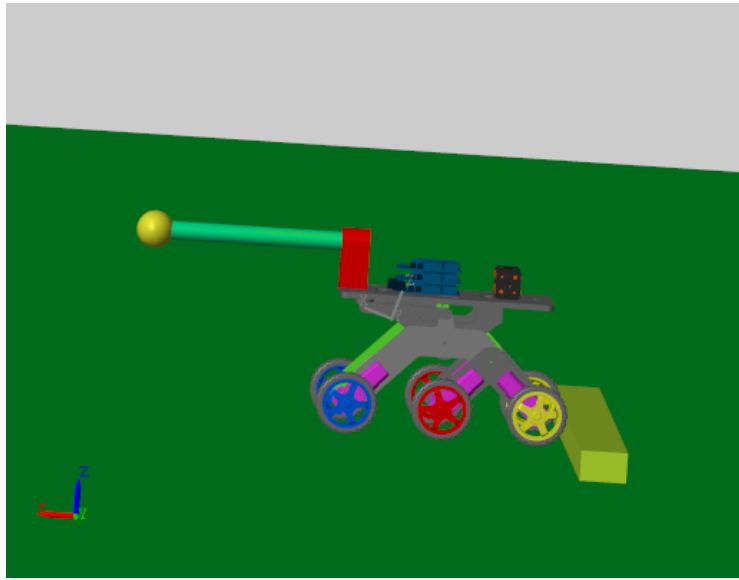


Figure 2.73: Rover model with an empirical robotic arm

- Frame A:** In the initial frame, the rover makes contact with the obstacle. The wheels are in motion, and with the assistance of the robotic arm, the center of gravity shifts from its resting position at the center of the front plate, which is the position considered when only the rover is taken into account. For the sake of simplicity, the initial configuration selected was that of the robot arm in a parallel position relative to the front plate of the rover.
- Frame B:** As previously outlined in the section 2.3.4.1, the imbalance and pushing action of the driving wheels result in the lifting of the central wheel off the ground, thereby initiating the climbing phase.
- Frame C:** Thereafter, the bogie wheels overcome the obstacle, allowing for the continuation of the forward movement. During this phase, the risk of overturning is minimal due to the friction in the rocker-plate joint, which prevents excessive and rapid rotation.
- Frame D:** In Frame D, the central wheels make contact with the obstacle. Concurrently, the robotic arm undergoes a configuration change, shifting the center of gravity towards the center of the platform on which the rover is mounted. As a result, the wheels that have made contact with the obstacle begin to proceed in an upward direction.
- Frame E:** In contrast to the central wheel of the rover, the rear wheels exhibit difficulty in overcoming the obstacle when they impact it in the configuration shown in frame E. This is due to a number of factors, including the center of gravity remaining too far back and the current configuration not allowing sufficient wheel torque to be produced for the system to successfully cross the obstacle.

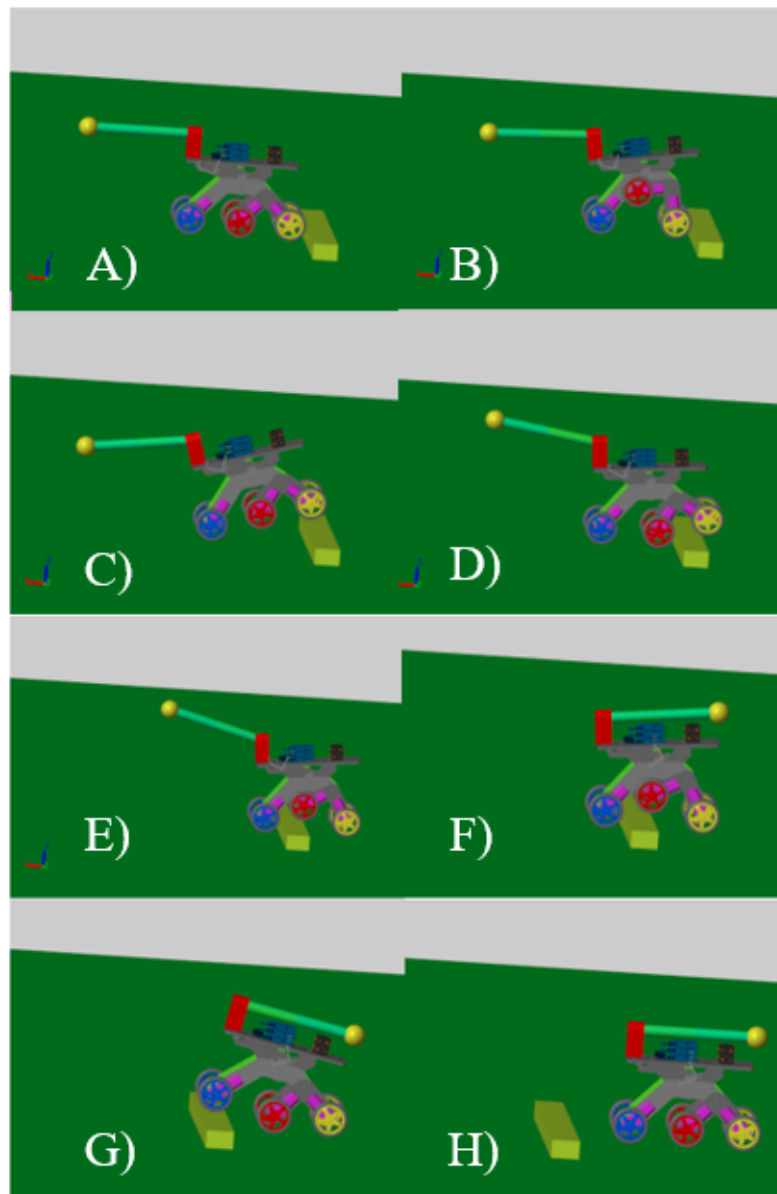


Figure 2.74: Simulation frames of the rover model with an "empirical" robotic arm

Frame F: Therefore, the robotic arm must begin to move towards the crossing configuration. Once the arm has reached the configuration shown in frame F, the link is able to extend itself to the right. This causes the center of gravity to move closer to the center of the plate, thereby enabling the rover to overcome the obstacle.

Frame G-H: The rover then successfully overcomes the obstacle.

Detailed Simulation Model

This section outlines the analytical approach employed in the simulation. It was initially

established that the rover was capable of overcoming the obstacle through the use of dynamic unbalancing. Subsequently, a model comprising both the rover and the robotic arm was developed using the Simscape environment.

The simulation analysis evaluated two distinct limit configurations of the robotic arm. The first configuration was designed to ensure the ability of the rover to climb the obstacle, while the second configuration was developed to prevent the rover from overturning.

As in previous instances, the center of gravity was adjusted to prevent overturning. It is necessary that the system demonstrate sufficient responsiveness to changes in attitude in order to avoid overturning. Furthermore, attaining this outcome through the actuation of a single joint provides a high degree of precision in the analysis of the rover-arm system and significantly simplifies the control policy of the entire robotic arm system.

The simulation frames are presented in Fig. 2.75.

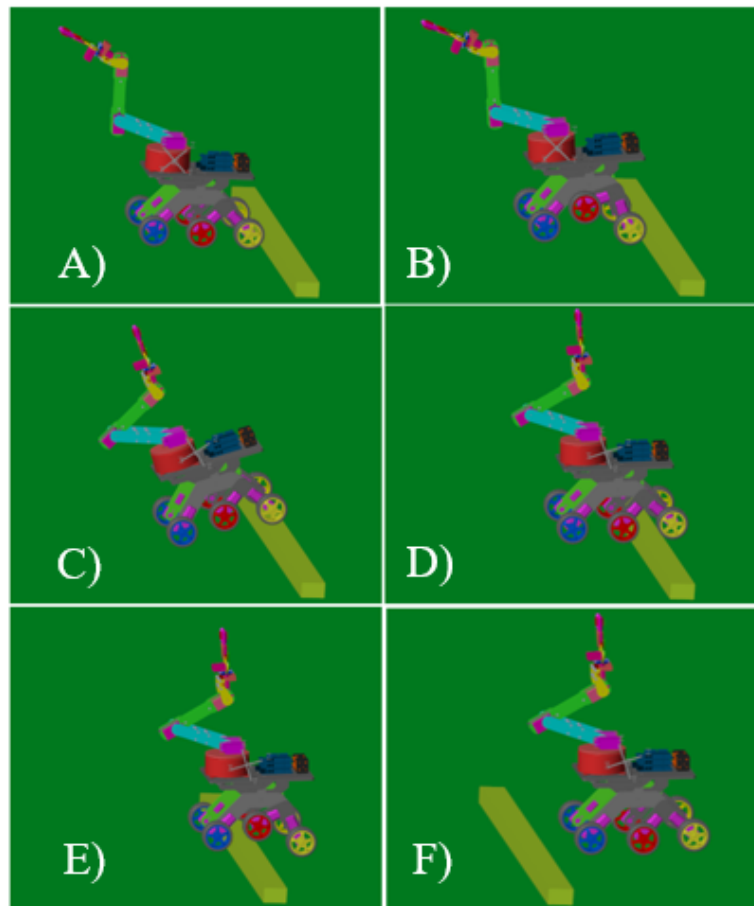


Figure 2.75: Simulation frames of the rover model with an "analytical" robotic arm

Robotic-Arm configurations analysis

It is important to define the correct configurations of the robotic arm, capable of creating

an imbalance that can improve the ability of the rover to cross obstacles. For this reason, the mechanism of the robotic arm will be analysed in order to achieve two necessary configurations: one that allows the prototype to cross obstacles and the other that ensures stability during the last stages of the crossing, avoiding the overturning of the prototype.

Due to the small size of the prototype used in both the simulation and the experimental test phases, it was observed that the first three eccentricities analysed in section 2.3.4.1 had very similar values, with a progressive increase from the most significant (e_1) to the least significant (e_3).

$$e_1 > e_2 > e_3 \quad (2.67)$$

$$e_1 \approx e_2 \approx e_3 \quad (2.68)$$

This condition made it possible to select a single robot arm configuration, designed on the basis of e_1 , which is valid for all three conditions analysed. In this configuration, the first two lines (front and centre) can cross the obstacle and the last line can start the crossing phase. However, in order to avoid overturning of the prototype, a second configuration was defined based on the eccentricity e_{lim} , corresponding to the overturning limit, in order to guarantee the stability of the prototype during the last phase of the crossing experiment.

The first step in defining such configurations is to determine the position of the center of mass of the robotic arm. When analysing the weight distribution of the rover (without the robotic arm embedded), the weight can be expressed as a vector aligned with the axis passing through the centre of the rover, which is used as the reference axis for the analysis. Using a 1D lumped mass model to represent the mechanical system, the mass distribution of the robotic arm can be analysed to determine the center of mass of the entire prototype.

The 1D system shown in Fig.2.76 has multiple components, such as the mass of the robotic arm m_{arm} , which is at a distance g from the reference axis z , while the mass of the rover m_{rov} is positioned on the reference axis. Additionally, the controller with a mass of m_c is placed at a distance d_1 and the batteries with a mass of m_b are positioned at a distance d_2 . The main mass for the analysis is the blue circle which is the centre of mass of the prototype at a distance e from the reference axis.

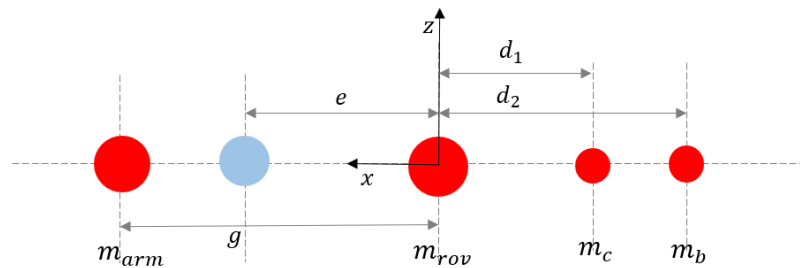


Figure 2.76: Mass distribution of the robotic arm

The eccentricity e represents the shift from the balanced state of the prototype to an unbalanced state. In particular, e gives the distance between the center of mass of the prototype, considering the robotic arm and the additional masses, and the ideal center of mass of the rover without these components. So the distance g must be calculated using the definition of the centre of mass for e :

$$e = \frac{\sum m_i x_i}{\sum m_i} = \frac{m_{\text{arm}}g - m_c d_1 - m_b d_2}{m_{\text{arm}} + m_{\text{rov}} + m_c + m_b} \quad (2.69)$$

$$g = \frac{(m_{\text{arm}} + m_{\text{rov}} + m_c + m_b)e + m_c d_1 + m_b d_2}{m_{\text{arm}}} \quad (2.70)$$

In the 2.3.4.1 section, the values of e were defined as those for which the prototype can cross the obstacle and avoid overturning. This means that the two configurations described in this section refer to the configurations of the robotic arm that ensure that the center of mass of the prototype is at a value of e that allows it first to lift the first, second and third pairs of wheels (e_1) and then to avoid tipping over (e_{lim}). The first configuration is applicable for the crossing ability of all the pair of wheels, because the two values of e are close to each other for this prototype. Once the eccentricity e has been defined, the distance g can be computed as shown in the equation 2.70. It is not enough to define the distance g from the reference axis; it is necessary to assess the angles of the robotic arm configurations. The distances of the centres of mass of the two configurations from the reference axis are given as g_1 and g_2 .

The robotic arm is constituted by four links having the mass given in Tab. 2.10.

m_1	m_2	m_3	m_4
0.179 kg	0.027 kg	0.081 kg	0.205 kg

Table 2.10: Masses of the link.

The moving mass now represents approximately 22 % of the total mass of the robotic arm. Furthermore, considering the reduced accelerations within the arm system, it can be said that the inertia involved is low compared to the overall system.

As for the lengths of the links, they are given as l_i , where $i=1,2,3,4$. Since the position of the center of mass of each member is known, the distances p_{ij} with $ij=1,2$ have been defined as the distances from one end of the link to the center of mass and from the center of mass to the other end of the link. This method of measurement allows accurate calculations and analysis of the dynamics and stability of the arm.

Climbing-up configuration

The necessary parameters to define the configuration of the robot are the angles shown in red in Fig. 2.77: α' , β' , and γ' . This challenge is commonly known as the inverse kinematics problem. The solution of the inverse kinematics problem can be complex due to the non-linear nature of the equations and the possibility of multiple solutions

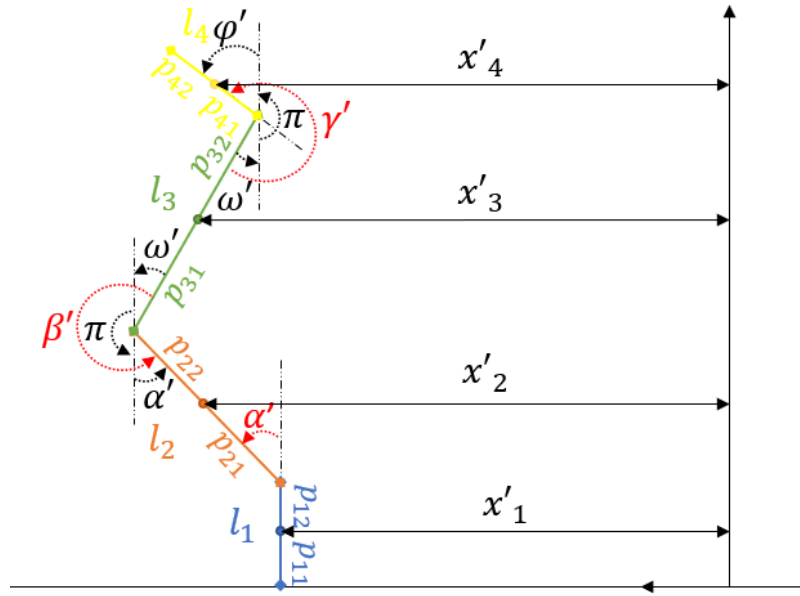


Figure 2.77: Configuration of the robotic arm during a climbing-up maneuver

depending on the configuration and physical constraints of the robot. The problem is simplified by fixing the angles α' and β' in a position that prevents singular configurations and allows the calculation of γ' . This approach reduces the complexity of the problem while ensuring that the robot arm operates reliably within its working constraints.

Additional angles, such as ω' and φ' , can be introduced and associated with the primary angles through specific geometric relationships. These relations are defined by the configuration and constraints of the robot arm: defining such relations simplifies the inverse kinematics problem by ensuring that the angles complement each other while keeping the structural integrity and functionality of the robotic arm. This approach not only simplifies the computational process, but also improves the accuracy and feasibility of the movements of the robotic arm.

$$\omega' = \beta' - \pi - \alpha' \quad (2.71)$$

$$\varphi' = \gamma' - \pi - \omega' \quad (2.72)$$

The distance between the center of mass of each individual link and the reference axis (the same as in Fig. 2.76) is defined in Tab. 2.11.

The distance x'_4 can also be expressed as a function of φ' and then rearranged to solve for φ' .

$$x'_4 = x'_1 + l_2 \sin \alpha' + l_3 \sin \omega' + p_{14} \sin \varphi' \quad (2.73)$$

$$\varphi' = \text{asin} \left[\frac{(x'_4 - l_3 \sin \omega' - l_2 \sin \alpha' - x'_1)}{p_{14}} \right] \quad (2.74)$$

Substituting the eq. 2.74 into eq. 2.72, γ' can be obtained.

$$\gamma' = \text{asin} \left[\frac{x'_4 - l_3 \sin \omega' - l_2 \sin \alpha' - x'_1}{p_{14}} \right] + \pi + \omega' \quad (2.75)$$

x-axis	Value
x'_1	65 mm
x'_2	$x'_1 + p_{12} \sin \alpha'$
x'_3	$x'_1 + l_2 \sin \alpha' + p_{31} \sin \omega'$
x'_4	$\frac{m_{\text{rob}}g_1 - m_1x'_1 - m_2x'_2 - m_3x'_3}{m_4}$

Table 2.11: Distances of the centers of mass relative to the reference axis.

On the basis of the considerations just made, this approach made it possible to define the angles that describe the configuration of the robotic arm.

$$\begin{cases} \alpha' = 70^\circ \\ \beta' = 250^\circ \\ \gamma' = 245^\circ \end{cases} \quad (2.76)$$

The important thing to note is that this configuration results from choosing one of the possible pairs of α' and β' . It should be said, however, that there are many other combinations: in our case, this particular pair of angles was chosen on the basis of some considerations regarding α' . In fact, the choice of angle α' is very important, mainly because of experimental considerations. The simulation does not include the waist joint, which is influenced by the weight of the entire robot arm. This means that as the angle α' approaches zero, the motor has to apply more torque to maintain the position; this increased torque requirement results in higher current consumption and therefore higher power consumption.

Anti-overturning configuration

The second configuration was designed to avoid overturning, as explained in Section 2.3.4.1. As the bogie wheels climb the obstacle step, the rocker wheels in touch with the obstacle act as a hinge, increasing the probability of the rover overturning. In the second configuration (Fig. 2.78), a procedure similar to the previous one is used, but the fixed angles are those defined in 2.77.

$$\begin{aligned} \alpha'' &= \alpha' \\ \gamma'' &= \gamma' \end{aligned} \quad (2.77)$$

It was decided in this case to modify the angle β'' , using an approach similar to that used previously. This makes it easier to control the arm during this phase. The angles are defined as shown in 2.78 and 2.79.

$$\omega'' = \beta'' - \pi - \alpha'' \quad (2.78)$$

$$\varphi'' = \gamma'' - \pi - \omega'' \quad (2.79)$$

The unknowns in the above are ω'' and φ'' , since they depend on β'' .

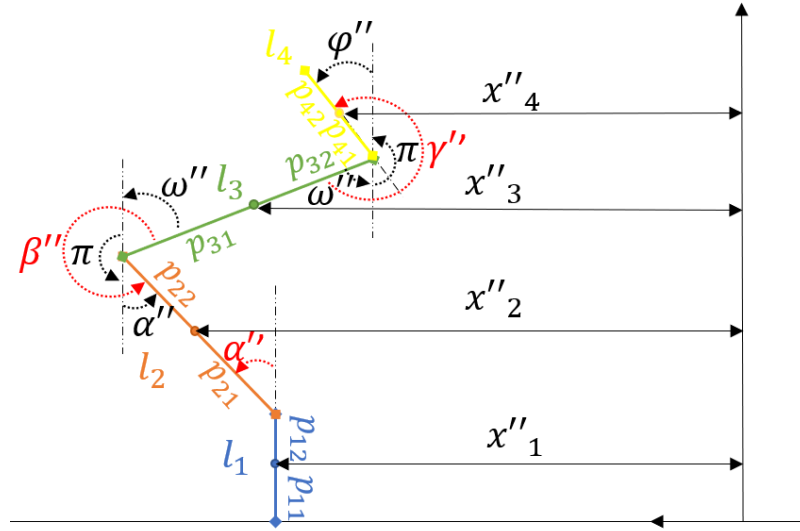


Figure 2.78: Robotic-Arm anti-overturning configuration

x-axis	Value
x''_1	x'_1
x''_2	$x''_1 + p_{12} \sin \alpha''$
x''_3	$x''_1 + l_2 \sin \alpha'' + p_{31} \sin \omega''$
x''_4	$\frac{m_{\text{rob}} g_2 - m_1 x''_1 - m_2 x''_2 - m_3 x''_3}{m_4}$

Table 2.12: Distances of the centers of mass relative to the reference axis for anti-overturning configuration.

The distances between the center of mass of each link and the reference axis are defined in Tab. 2.12.

The distance x''_4 can also be expressed as a function of φ'' and then rearranged to isolate $\sin \varphi''$.

$$x''_4 = x''_1 - l_3 \sin \omega'' + l_2 \sin \alpha'' + p_{14} \sin \varphi'' \quad (2.80)$$

$$\sin \varphi'' = \frac{x''_4 - l_3 \sin \omega'' + l_2 \sin \alpha'' + x''_1}{p_{14}} \quad (2.81)$$

Starting from 2.79, the eq. 2.82 can be defined.

$$\sin \varphi'' = \sin(\gamma'' - \omega'' - \pi) = -\sin(\gamma'' - \omega'') = \sin(\omega'' - \gamma'') \quad (2.82)$$

in order to arrange 2.81 as in 2.83

$$\sin \omega'' \cos \gamma'' - \cos \omega'' \sin \gamma'' = \frac{x''_4 - l_3 \sin \omega'' + l_2 \sin \alpha'' + x''_1}{p_{14}} \quad (2.83)$$

From 2.83, ω'' can be evaluated. However, since 2.83 is a transcendental equation and ω'' is on both sides of the equation, a numerical method is required to solve it. In this case, the Newton-Raphson method was chosen to solve it.

Once the value of ω'' has been found, the angle β'' can be computed using the expression in 2.84.

$$\beta'' = \omega'' + \pi + \alpha'' \quad (2.84)$$

Using this method, a configuration was identified that simplified the definition of the centre of gravity and ensured the stability of the prototype during the final phase of the crossing.

$$\begin{cases} \alpha'' = 70^\circ \\ \beta'' = 313^\circ \\ \gamma'' = 245^\circ \end{cases} \quad (2.85)$$

2.3.4.4 Conclusions

The aim of the study is to demonstrate how the addition of a robotic arm improves the ability of the rover to cross several obstacles. This addition is necessary because the rocker-bogie suspension system alone is not enough to overcome specific types of obstacles. The integration of the robotic arm gives the rover the ability to cross obstacles that would otherwise be uncrossable.

Both static and dynamic analyses have shown that strategic manipulation of the center of gravity of the rover using the robotic arm significantly assists in overcoming obstacles while maintaining stability. This stability is provided by adaptive logic control strategies that respond to the nature of the obstacles faced. Simulations using Simscape and the multi-body model have provided critical data confirming the feasibility and effectiveness of this method. These results suggest that the approach is not only theoretically sound, but also ready for experimental testing on actual rover prototypes, as will be discussed in the Chapter 3.

Additionally, the integration of the robotic arm significantly enhances the operational capabilities of the rover, allowing navigation through more complex terrain while improving both energy and time efficiency. Directly overcoming obstacles, while challenging, typically requires less time and distance than avoiding them, resulting in potential energy savings.

Particularly noteworthy is the enhanced ability of the robotic arm to cross obstacles higher than the radius of its wheels. It preserves critical functionality without compromising the structural integrity of the system. However, it is important to recognise that this modification adds complexity to the rover. As a result, further improvements may be required to optimise the system, focusing on a more sophisticated control design that balances efficiency with the new capabilities provided by the robotic arm.

2.4 Full-Scale Rover Prototype

The design and realization of the full-scale prototype represents the conclusion of a sophisticated engineering process, which is defined by the incorporation of advanced modeling, simulation, and optimization tools. This section presents an illustration of the development of the first and second rover prototypes, with a detailed account of the structural components, multibody models, simulations for overcoming obstacles and performance analysis on uneven surfaces. By undertaking an exhaustive investigation into energy consumption and exploring alternative configurations, the work exemplifies a meticulous methodology aimed at optimising the efficiency and resilience of the rover for operations in challenging environments. The objective of this chapter is to provide a comprehensive overview of the development process and the milestones achieved, emphasising the significance of design iteration and experimental validation in attaining the desired outcomes.

2.4.1 First Prototype

In partnership with Green Tech Solution, the initial concept for the first prototype was developed based on a set of specific requirements. The goal of this collaboration was to utilize the expertise of Green Tech Solution to create a prototype that closely followed predetermined specifications. These specifications were carefully defined to ensure that the prototype would meet both functional and environmental sustainability criteria, providing a solid foundation for subsequent phases of testing and refinement.

2.4.1.1 CAD Model of the Rover

The main specifications for the first prototype of rover have been defined as follows:

- Maximum speed: 5 km/h
- Maximum slope: 20°
- Required autonomy: 40 minutes

The rover will be equipped with a 5-axis robotic arm with the following specifications:

- Maximum extension: about 500 mm
- Maximum weight capacity: approximately 20 kg

The robotic arm is not included in this paper as it is still in the initial design stage. The rover presented is a first prototype and does not include the 5-axis robotic arm. The dimensions of the prototype are *1220 mm x 1206 mm x 418 mm*, as shown in the Fig. 2.79, 2.80, 2.81.

The most important subsystems of the rover, which allow the correct behaviour of the system, are:

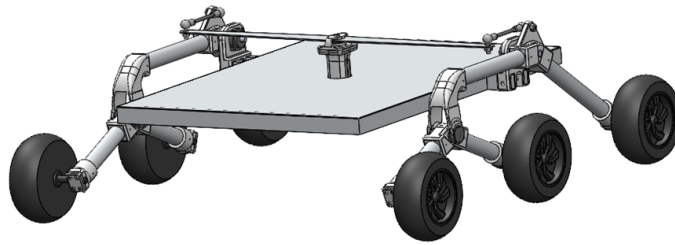


Figure 2.79: CAD model of the initial rover design

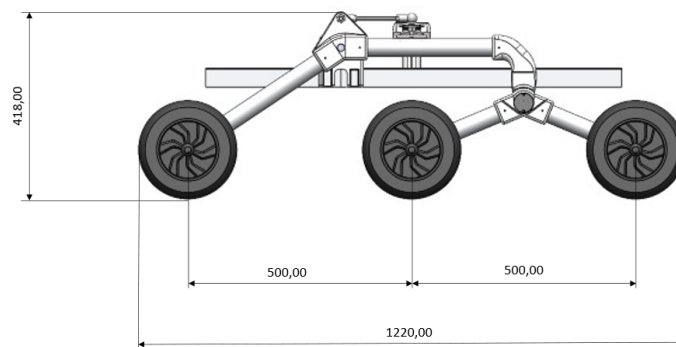


Figure 2.80: Side view of the initial rover

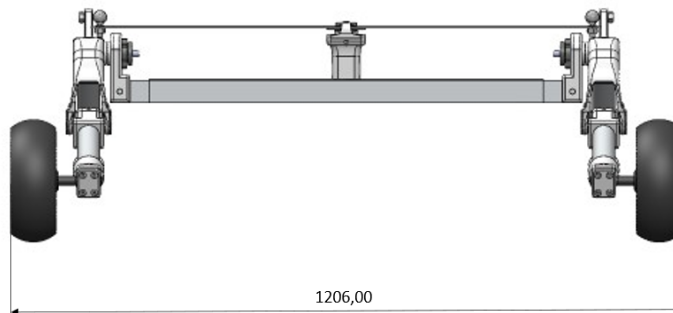


Figure 2.81: Front view of the initial rover

- The differential system
- The two rocker-bogie suspensions
- The spherical joints subsystem

Rocker-Bogie Suspension

The rover is equipped with a rocker-bogie assembly on each side, which is designed

to increase the mobility and stability of the rover over uneven terrain. The rocker, the larger link in the assembly, is connected to the main body of the rover (the chassis) at a central pivot point known as the rocker pivot. This component has a wheel at one end and is connected to the bogie at the opposite end. The bogie, the smaller link, connects to the rocker at the bogie pivot and has wheels mounted at both ends to provide additional stability (see Fig. 2.82). Each of the six wheels in the assembly is independently powered by its own motor, allowing for greater control and adaptability to different terrains. In

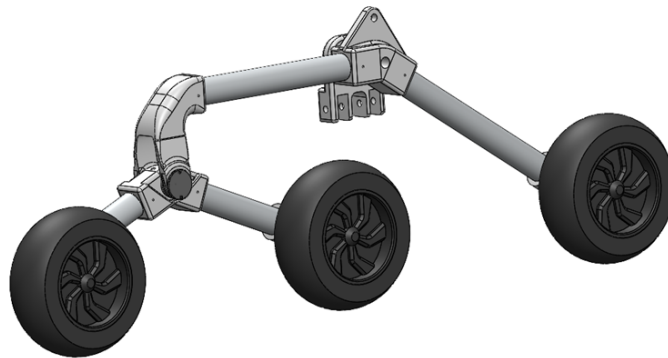


Figure 2.82: CAD model of a single Rocker-Bogie suspension

accordance with the definition, the rocker and bogie components of the rocker-bogie suspension system are shown in Figure 2.83, 2.84, 2.85 and 2.86, respectively.

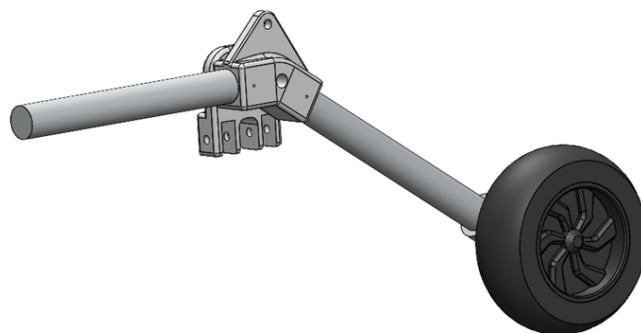


Figure 2.83: CAD model of the Rocker subsystem

The center section of each bogie is designed to rotate around the pivot located on the left side of the rocker. Importantly, the two bogies are not connected: this allows them to rotate in either the same or opposite directions, allowing the rover to dynamically adapt to irregular terrain. This adaptability is illustrated in Figures 2.87, 2.88, and 2.89 within the Simscape environment.

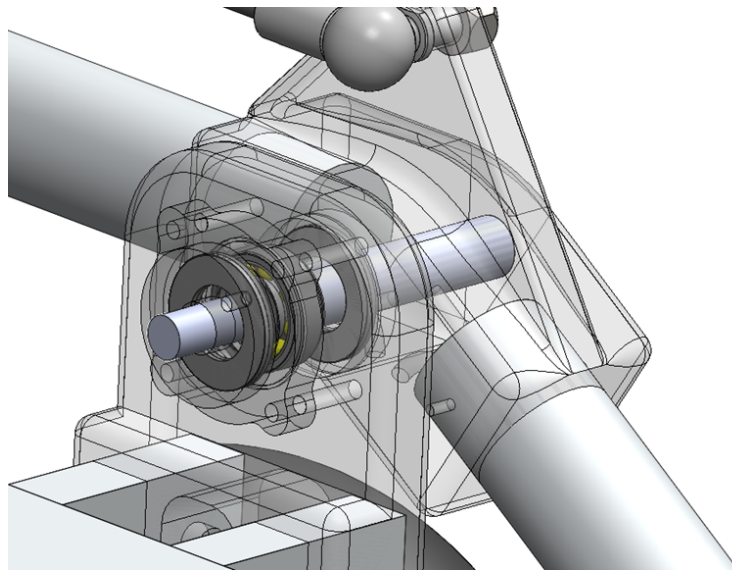


Figure 2.84: Components of the Rocker subsystem

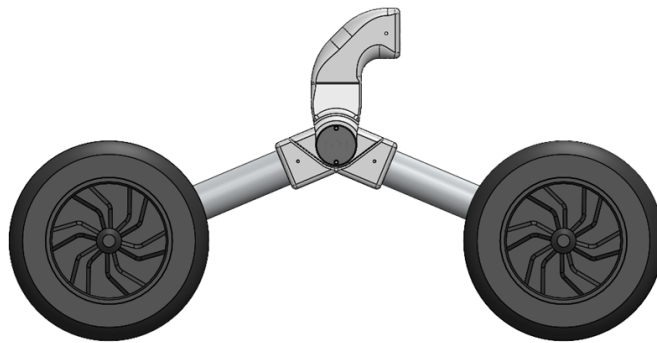


Figure 2.85: CAD model of the Bogie subsystem

Differential System

Understanding how the rocker-bogie mechanism maintains a level rover body can be challenging. If you build a model of the rover and attach the rockers to the body with an axle or two pivot pins, the body will tend to tilt forward or backward until it touches the ground. To prevent this tilting around the rocker pivots, real rovers use a differential mechanism (see Fig. 2.90) that connects the two rockers and the body. This differential is essential for stabilizing the body and keeping it level as the rover travels uneven terrain.

As one rocker moves up relative to the body, the other moves down. As a result, the rover body maintains an angle relative to the ground that is halfway between the angles of the two rockers. The differential mechanism provides this balance: the center of the differential rod (Fig. 2.91) is connected to the rover body by a pivot (Fig. 2.92), while each end is connected to a rocker by short links. To illustrate how this mechanism

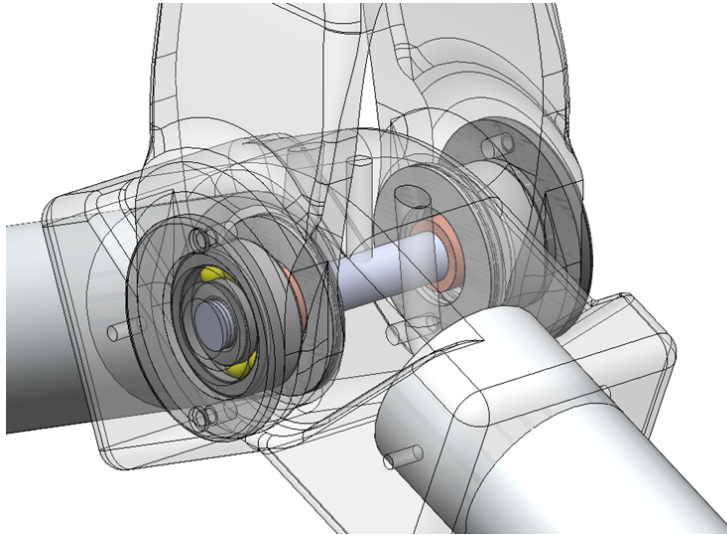


Figure 2.86: Components of the Bogie subsystem

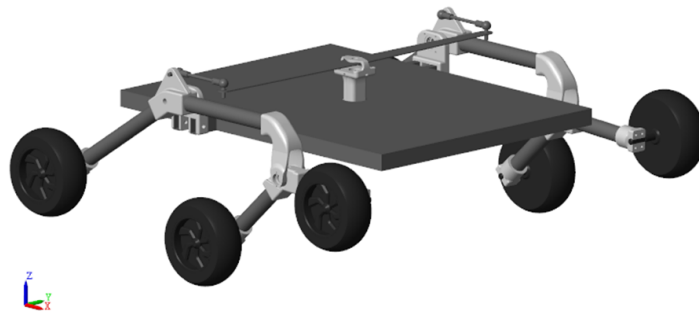


Figure 2.87: Uniform rotation of the Bogies

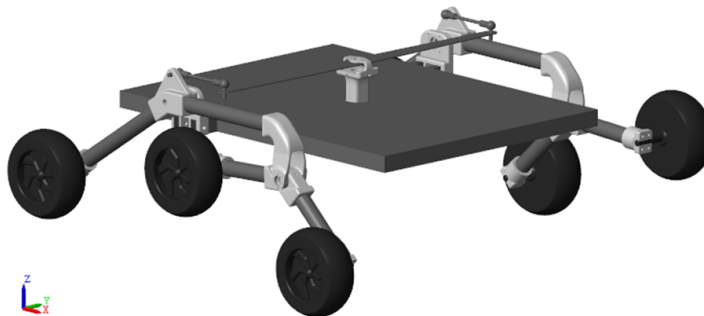


Figure 2.88: Opposite rotation of the Bogies

works, images from the Simscape simulation environment have been provided where the two rocker joints have been rotated 20 degrees in opposite directions. Fig. 2.93 and

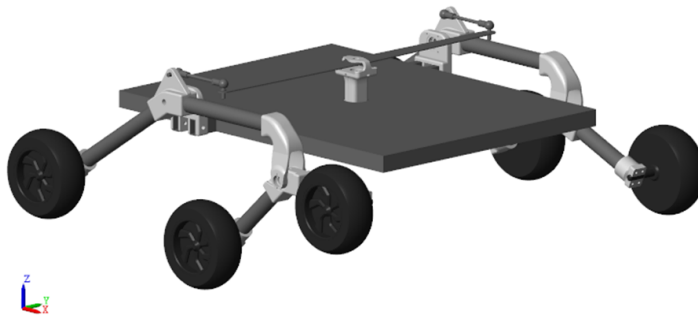


Figure 2.89: Independent rotation of the Bogies

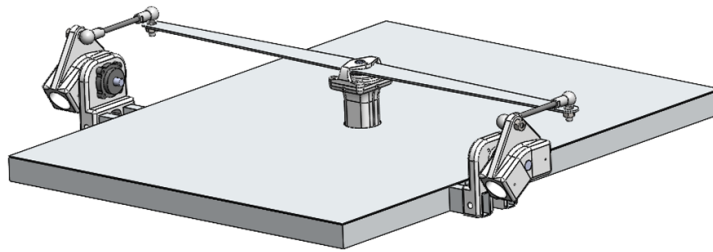


Figure 2.90: CAD model of the differential system

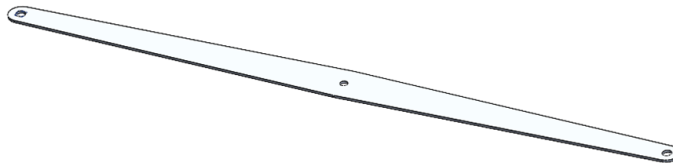


Figure 2.91: CAD model of the differential rod

2.94 show the rotation of the differential rod, which allows the rocker-bogie systems to independently adapt to uneven terrain and maintain the stability of the rover.

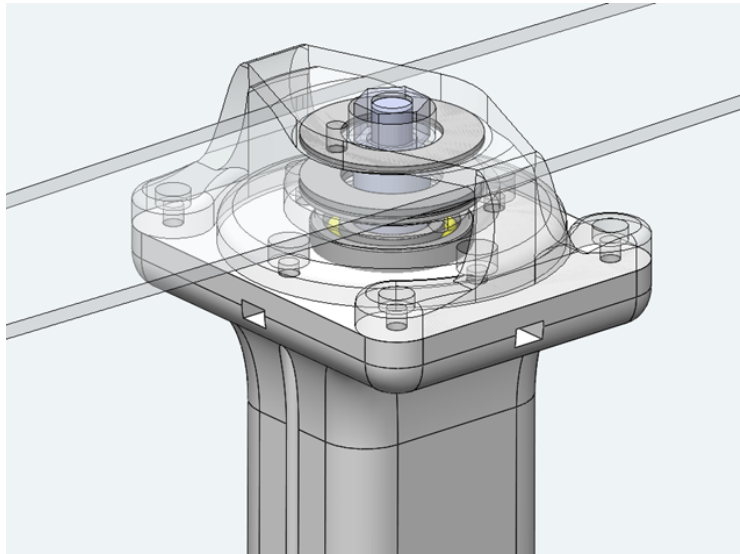


Figure 2.92: Detailed view of the pivot in the differential system

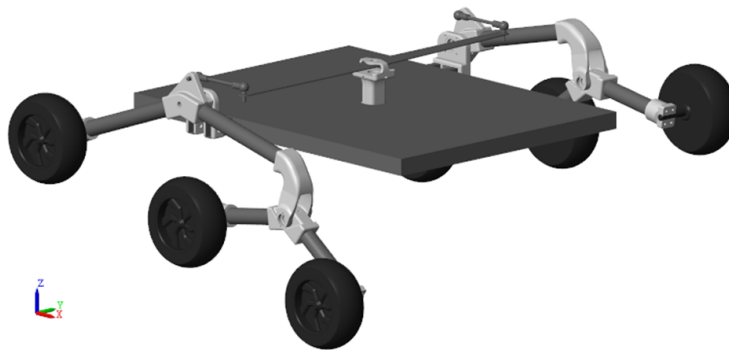


Figure 2.93: Differential Mechanism in Simscape environment: Isometric view

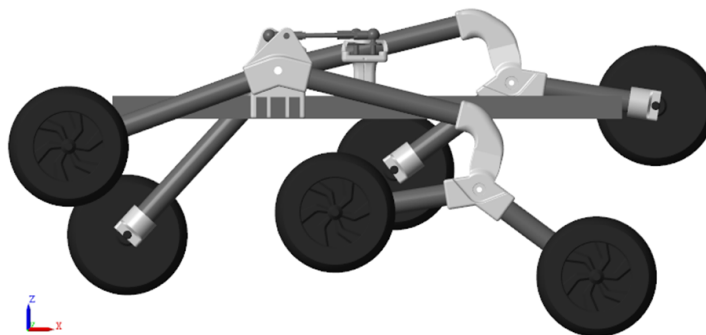


Figure 2.94: Differential Mechanism in Simscape environment: Side view

Spherical Joints

The link between the differential bar and the rocker is represented by the chain shown in Fig. 2.95, which is made up by:

- 2 x Spherical Joint WG35
- 1 x Spherical Joint Link
- 2 x Bolts

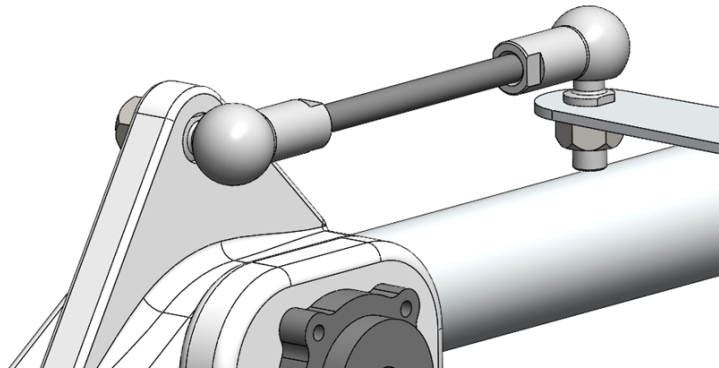


Figure 2.95: Connection using spherical joints in the linkage system

This component of the rover plays a critical role by allowing the differential rod to rotate around its pivot in response to the rocker's rotation around its own pivot. This synchronized motion ensures that the body of the rover remains balanced as the rocker-bogie assembly adjusts to changes in terrain, as shown in Fig. 2.96. Fig. 2.96 shows how

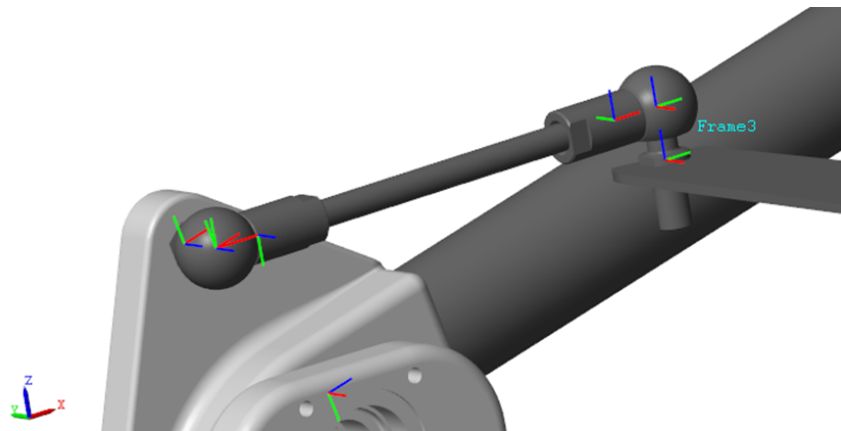


Figure 2.96: Visualization of spherical joint rotation with reference frames

the frames of the two components making up the spherical joint rotate relative to each other while maintaining a common origin. This design allows for multi-axis rotation without displacement, allowing for smooth and flexible movement within the joint.

Wheel-Motor Subsystem

It was not possible to create separate models for the wheel and motor because the wheel-motor subsystem was taken from a hoverboard. Separating these components was avoided to avoid the risk of introducing dust or dirt into the motor and to ensure that the system could be reassembled correctly without compromising its functionality. To



Figure 2.97: Wheel-Motor subsystem: Isometric view

connect the wheel-motor subsystem with the tubular bar, a support has been created, as shown in Fig. 2.99.

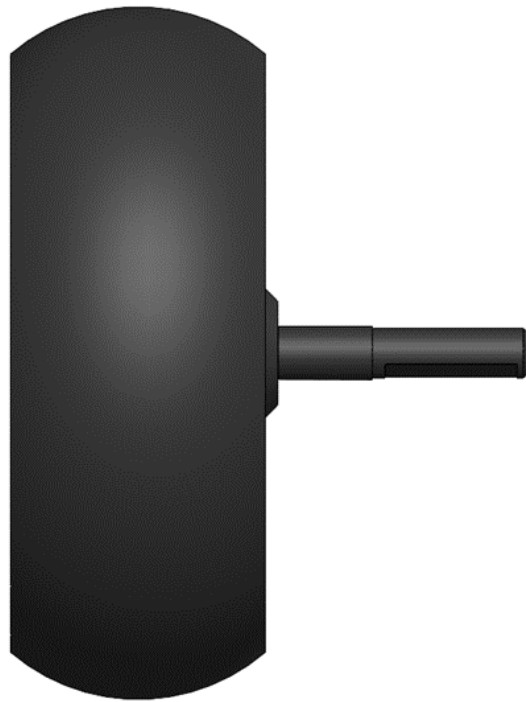


Figure 2.98: Wheel-Motor subsystem: Side view

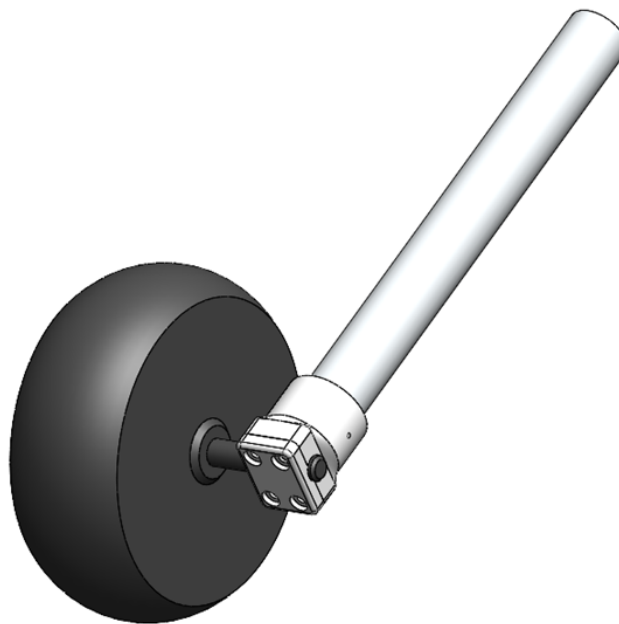


Figure 2.99: Wheel-Motor Subsystem with tubular bar attachment for structural support

2.4.1.2 Multibody Model: Development and Optimization

The multibody, multi-physics model of the rover was developed in MATLAB Simulink using multi-domain blocks within the Simscape environment. As done previously for the small-size prototype, this setup allows the simulation of interactions across different physical domains, increasing the accuracy and functionality of the model.

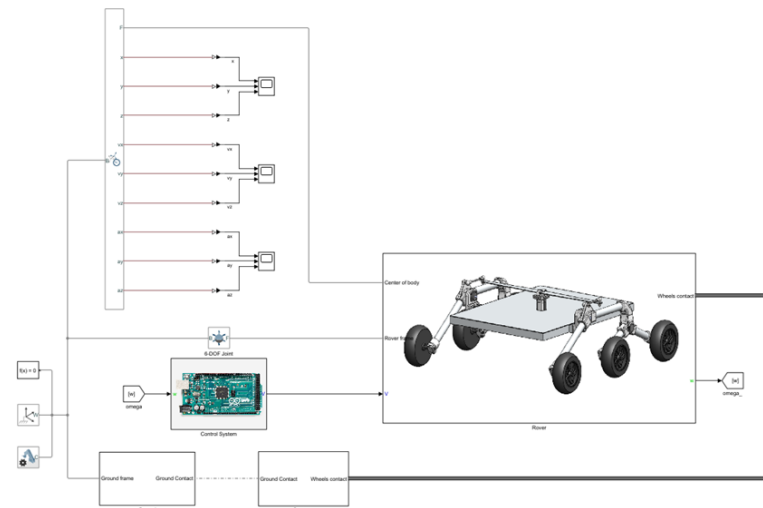


Figure 2.100: Macro-blocks structure of the rover model illustrating the modular configuration of the system

The model in Fig. 2.100 is made up of several macro-blocks:

1. Multibody model of the rover
2. Control System
3. Contact
4. Ground
5. IMU
6. Solver Configuration, World Frame and Mechanism Configuration blocks
7. 6-DOF joint

Certain macro blocks will not be analyzed in detail here because they are identical to those used in the small-size prototype. These components have already been thoroughly tested and validated in the previous prototype, and their functionality remains unchanged in the current model. Consequently, a detailed analysis of these blocks would be redundant and is therefore omitted from this discussion.

Multibody Model of The Rover

The multi-body model of the rover was created using the File Solid block, which allows CAD parts to be imported into the Simscape environment. For parts with relative motion, various types of joints were introduced, while for parts without relative motion, simply connecting the frames correctly was sufficient.

Not all components (such as bearings, screws, bolts, etc.) were imported to create the multi-body model, as their inclusion would have increased the computational load and consequently the simulation time.

The following CAD files were imported into Simulink:

- Body Plate.
- Body Chassis.
- Lower Differential Joint Link.
- Upper Differential Joint Link.
- Differential Bar.
- 4 x Spherical Joint WG35.
- 2 x Spherical Joint Link M10.
- 2 x Inner Rocker.
- 2 x External Rocker.
- 2 x Horizontal Bar.
- 2 x Upper Bogie Joint.
- 2 x Lower Bogie triangular Joint.
- 4 x Bogie Bar.
- 2 x Rear Bar.
- 6 x Upper Wheel attachment.
- 6 x Wheel Spindle.
- 6 x Wheel-Motor system.

The multibody structure of the rover can be seen in Fig. 2.101.

The multibody structure of the rover is made up of three parts:

1. Central Body
2. Left Rocker-Bogie system

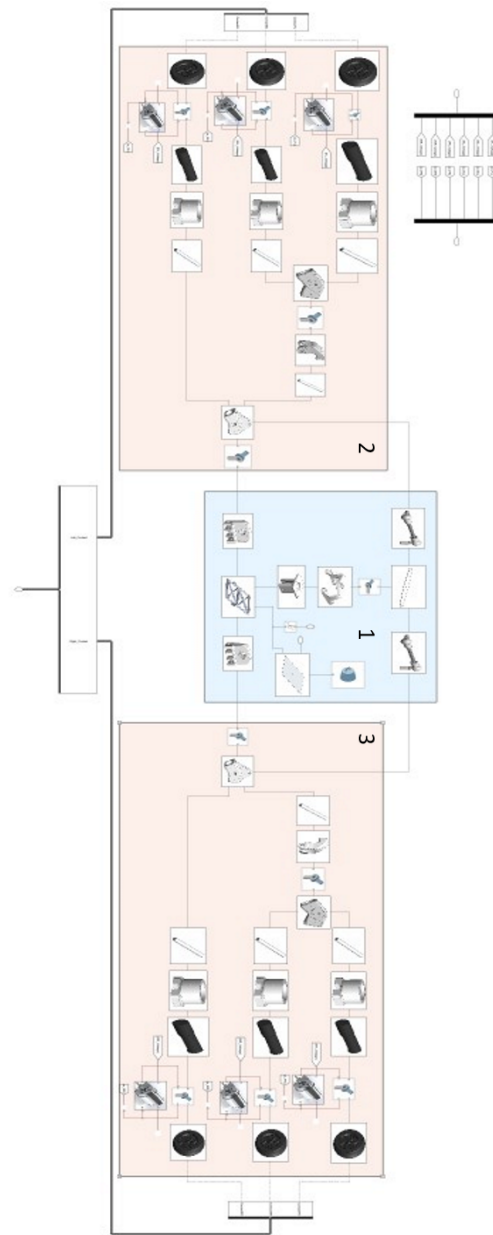


Figure 2.101: Multibody structure of the rover highlighting its modular and interconnected components

3. Right Rocker-Bogie system

The central part is made up by the parts showed in fig. 2.102. An additional inertia has been included to simulate the weight of the non-imported components and possibly the weight of the robot arm.

An additional inertia has been added to account for the weight of non-imported components and, if necessary, the weight of the robotic arm.

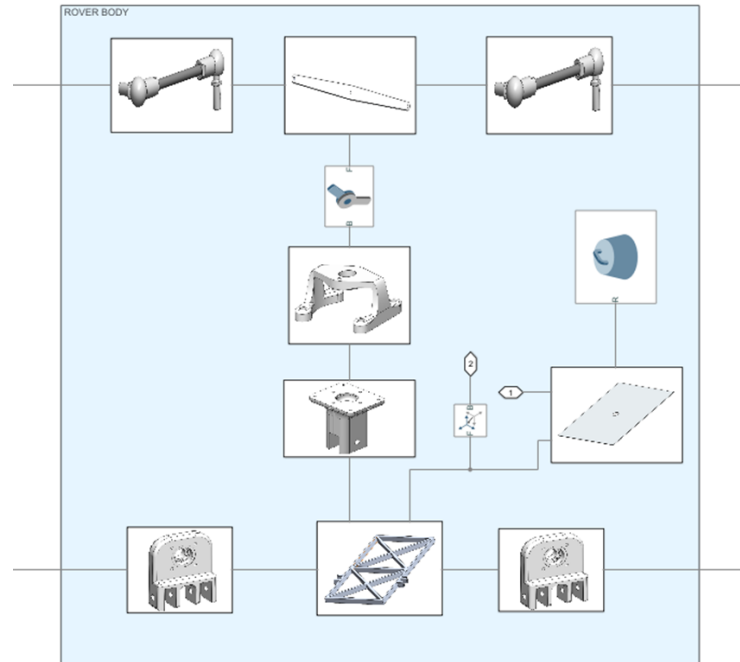


Figure 2.102: Multibody structure of the Central Body, illustrating the key components and their interconnections

As shown in figure 2.102, a revolute joint is positioned between the differential rod and the upper GUR. This joint provides a rotational degree of freedom through a revolute primitive. It forces the origins of the two frames to coincide, aligning the z-axes of the base and follower frames, while allowing the x-axis and y-axis of the follower to rotate about the z-axis. Fig. 2.103 and 2.104 illustrate this setup, showing the rotation about the z-axis in the frames linked by this joint.

The subsystem of the spherical joints has been modelled as in Fig.2.105. A spherical joint was required for this subsystem. This type of joint, also known as a ball-and-socket joint, provides three degrees of rotational freedom through a single spherical primitive. It constrains the origins of the two frames to coincide and allows the follower frame to rotate freely around this common origin as a pivot.

Figure 2.106 shows the model of one of the two bogie subsystems. Since the two subsystems are symmetrical, only one of them (the left subsystem) will be described in more detail. Between the central body and the rocker-bogie suspension is a rotating joint. Starting from the left, the top and middle rows in the model represent the "bogie" portion of the suspension, while the last row represents the rocker portion connected to the third wheel. In addition, it is important to note the other rotary joints that allow the z-axis of the wheels to rotate around the z-axis of their respective spindles.

To complete the description of the multi-physics model of the rover, two additional elements are key:

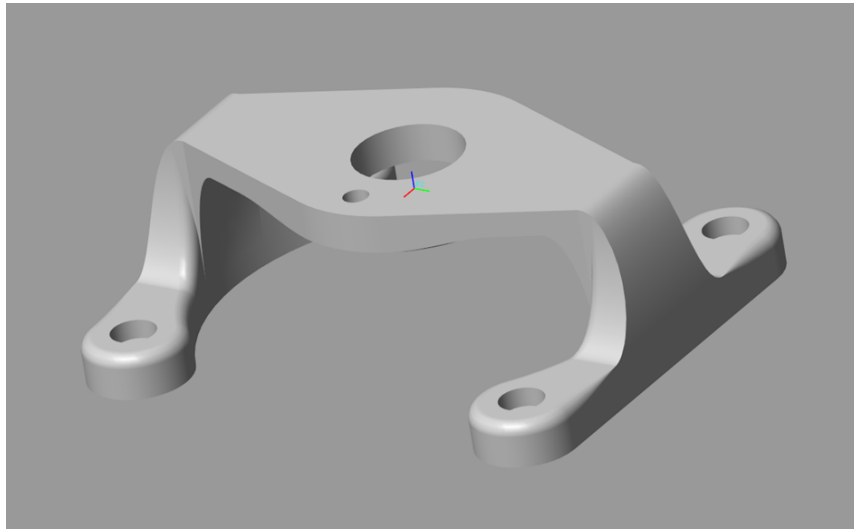


Figure 2.103: Reference frame for the upper cap of the differential joint

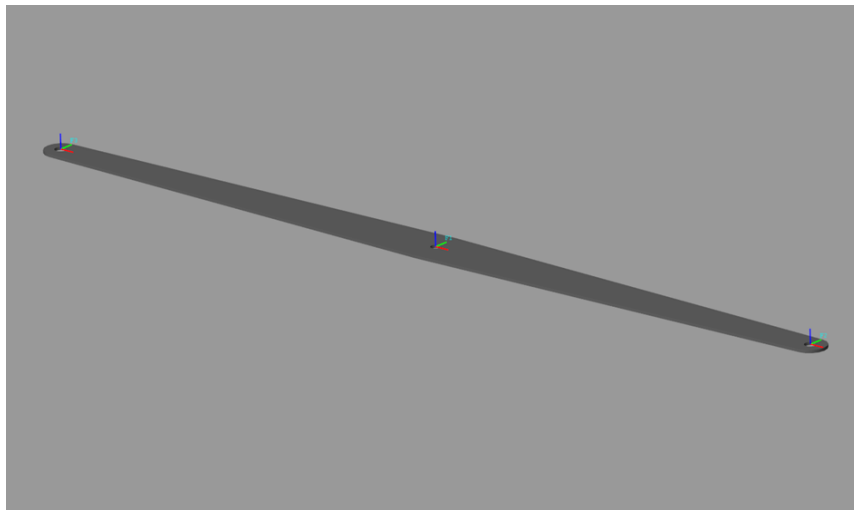


Figure 2.104: Reference frames along the Differential Bar

1. The DC motor block, which is used to model the motors for each wheel
2. The Wheel block to illustrate how the contact was defined for the simulations

DC Motor

In Fig. 2.107, the model of the DC motor is shown. The DC motor block interacts with the rotary joint by exchanging speed and torque information: it provides torque as an input to the joint and receives speed as an input from the joint.

Modeling the multi-domain motor required the integration of two primary domains: the electrical domain and the mechanical rotational domain. The electrical domain was used to simulate the power supply circuit of the motor, while the mechanical rotational

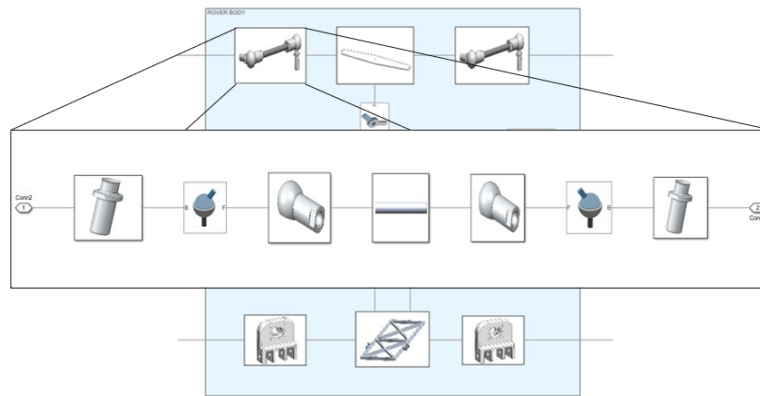


Figure 2.105: Multibody model representation of the spherical joint link, illustrating the structural and dynamic connections

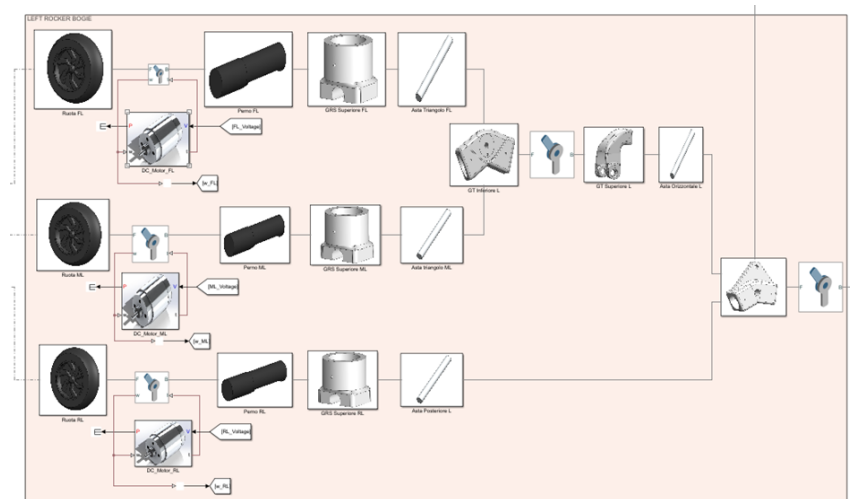


Figure 2.106: Multibody model of the rocker-bogie subsystem, showing its components

domain was used to model the gearbox. Starting from the left, the main blocks used in this setup include:

- Controlled Voltage Source.
- Power Sensor.
- Current Limiter.
- DC Motor.
- Gear Box.
- Rotational Multibody Interface.

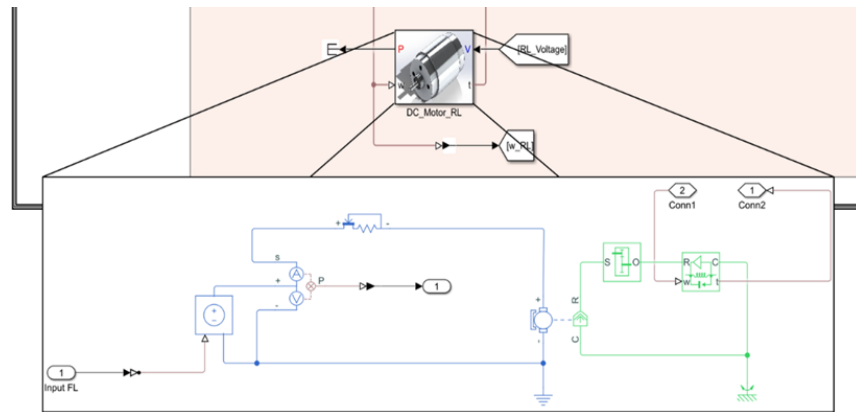


Figure 2.107: Multi-domain model of the DC motor, integrating mechanical and electrical systems

A current limiter was introduced because the drivers selected by the company are less powerful than the six DC motors. As a result, the motors can not operate at their maximum power and a current limit must be imposed.

The Rotational Multibody Interface block acts as an interface between a Simscape Multibody joint and a Simscape Mechanical Rotational network, facilitating the interaction between these two domains.

Wheel Subsystem

As explained in the previous sections, the wheel has been designed using the two different methods: pointcloud and the convex hull.

In the first case the subsystem is represented in Fig. 2.108. The File Solid block is used

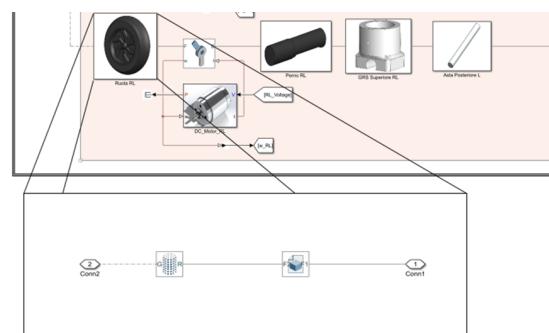


Figure 2.108: Multibody model of the wheel (Case 1) illustrating the subsystem design with a File Solid block for CAD model import and a point cloud representation

to import the CAD model of the wheel into Simscape. In addition, the Point Cloud block represents the geometry as a point cloud, where each point has a fixed offset relative to the reference frame of the block. Within this block, an $n \times 3$ matrix can be defined, where

n is the number of points and "pcm" is the coordinates matrix generated by a MATLAB script.

The points in the point cloud can also be visualized graphically, as shown in Fig. 2.109 and Fig. 2.110. In the second case, within the subsystem shown in figure 2.111,

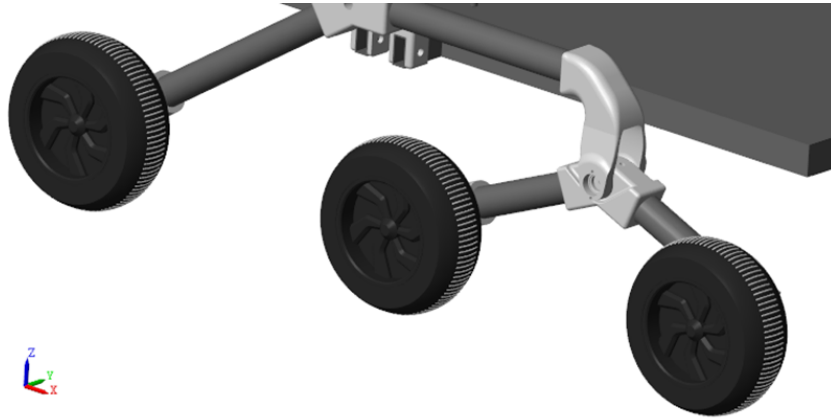


Figure 2.109: Graphical representation of the wheel using a point cloud with 100 circumferential points

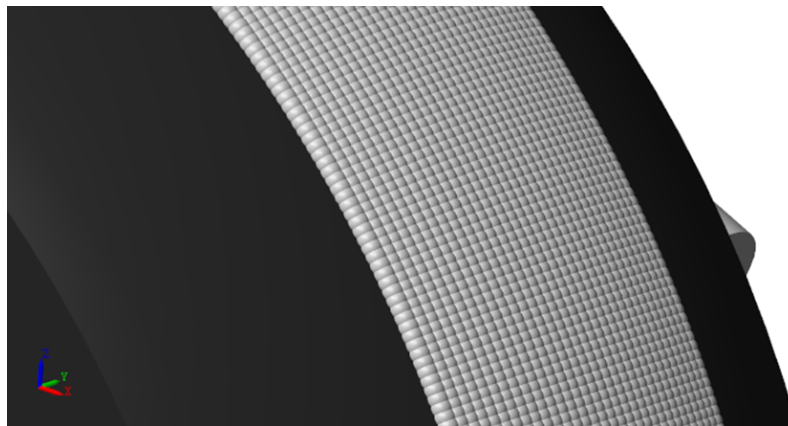


Figure 2.110: Zoomed view of the graphical representation of the point cloud on the wheel surface

the File Solid block is used to import the CAD model of the wheel into Simscapen as previously but here, the block allows the export of the convex hull of the solid, allowing a simplified geometry representation of the imported CAD file within the Simscape Multibody environment.

This convex hull geometry can be used to model spatial contact forces, providing an efficient approximation of the true geometry, as shown in Figure 2.112.

Control System

The Control System subsystem is designed to set a target angular velocity for each

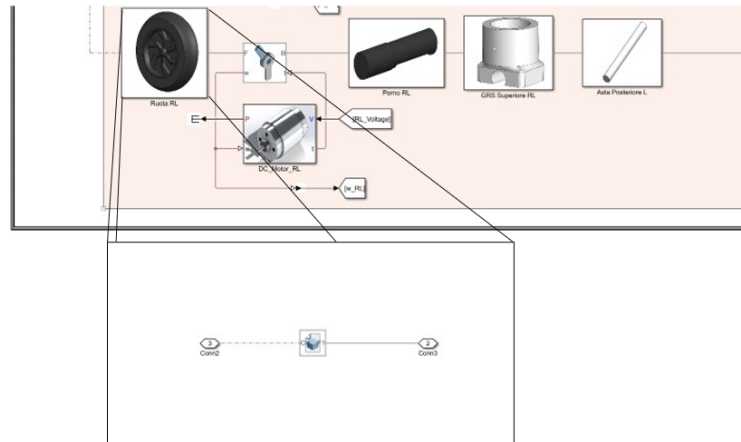


Figure 2.111: Multibody model of the wheel (Case 2) illustrating the subsystem design with a File Solid block for CAD model import as Convex Hull

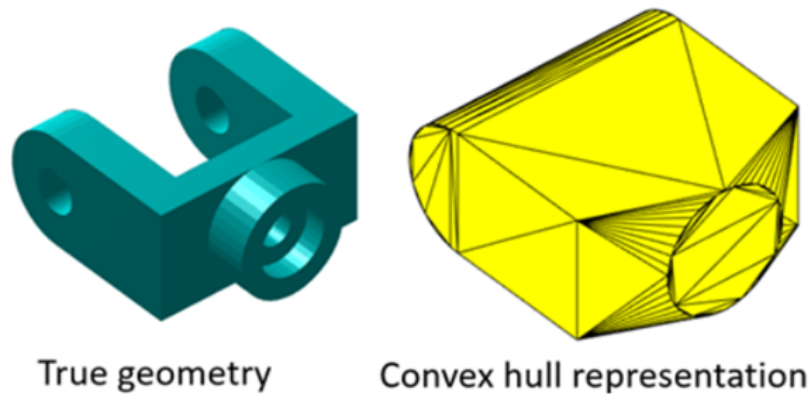


Figure 2.112: Comparison between the true geometry and its convex hull approximation

wheel. A feedback control loop is implemented for each wheel using encoders provided by each motor. The encoder measures the angular velocity of the motor ω , which is then compared to a reference velocity. The six output voltages, V , from this subsystem serve as inputs to the six DC motor blocks. The logic of the control system is shown in Fig. 2.113.

Starting from the left, the blocks used include:

- Repeating Sequence Block: Outputs a repeating sequence based on specified time-value pairs.
- Sum Block: Adds or subtracts inputs. Here, it subtracts the actual speed from the reference speed and generates an error signal.
- PID Block: Implements continuous and discrete-time PID control algorithms.
- Saturation Block: Limits the input signal to defined upper and lower limits.

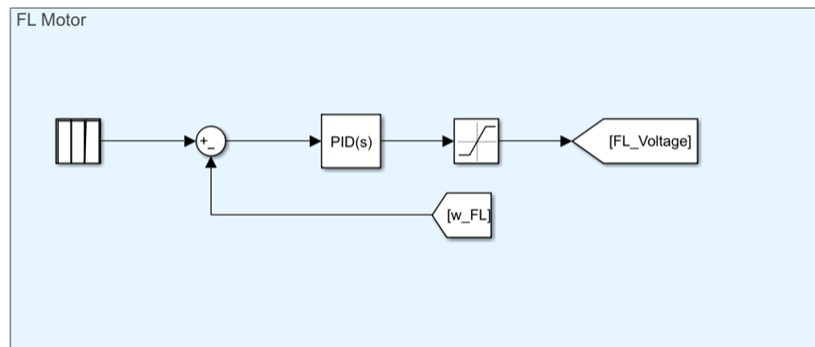


Figure 2.113: Control system logic diagram for each wheel, illustrating the PID controller and voltage regulation

- From" block: Receives signals from a corresponding "Goto" block with the same tag, in this case retrieving angular velocity data from the DC Motor block.
- Goto Block: Sends signals to "From" blocks with matching tags; in this case, it sends the calculated voltage as an input to the DC Motor block.

A PID controller was selected for this system due to its effectiveness in managing the dynamic response of each motor.

Contact Subsystem

As with the small rover prototype, the contact model here has been designed according to the criteria described in 1.7.3.1. Each wheel has its own Spatial Contact Force block, for a total of six blocks. This block applies a contact force between the two connected geometries, preventing penetration by generating forces that push the geometries apart near the point of contact. These forces are equal and opposite, aligned along a common line of action.

The contact parameters that can be adjusted include the following, as shown in Figure 2.114:

- Stiffness
- Damping
- Transition width
- Static friction coefficient
- Dynamic friction coefficient
- Critical Speed

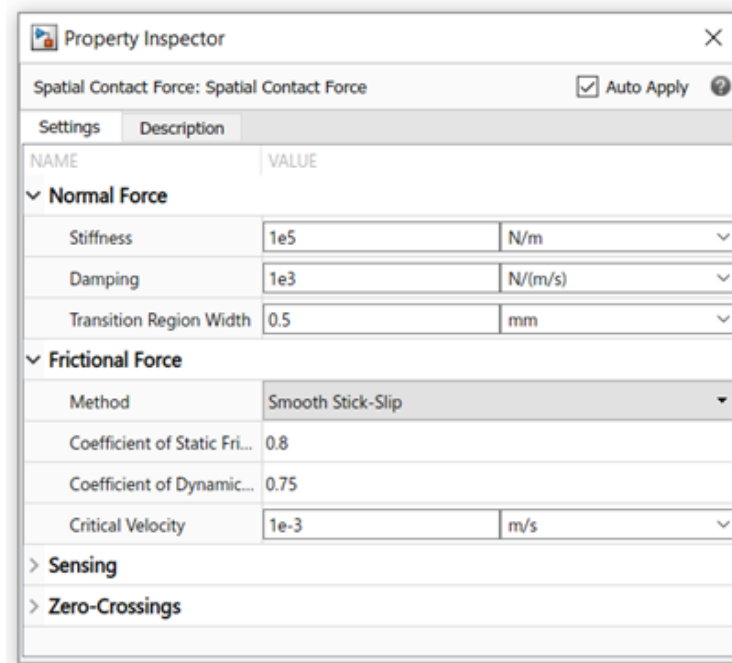


Figure 2.114: Parameter settings for the "Spatial Contact Force" block, including normal force and frictional force configurations

Ground Subsystem

For the ground subsystem, similar to the wheel subsystem, two cases are distinguished:

In the first case, when using the Point Cloud Wheel, the ground is modeled by the Grid Surface block. This block represents a grid surface defined by a rectangular grid in the X-Y plane with a Z-height assigned to each grid point. The grid follows the MATLAB NDGRID format, where the X and Y grid vectors correspond to the rows and columns of the Z matrix, respectively. Figure 2.115 illustrates how the surface is created in this setup.

For the second case, when the convex hull of the wheel is used, the ground is modeled using a "Brick Solid" block, as shown in Figure 2.116.

The difference between the two cases is the ease of modeling non-uniform surfaces. In the first case, the use of a mesh surface makes it easier to represent a non-uniform surface. In the second case, however, multiple brick solids are required to model a non-regular surface because the convex hull format still applies.

The advantage of the brick solid approach is that it allows the extraction of the normal force signal from the spatial contact force block, which is not possible with the point cloud grid surface blocks. This capability makes the Brick Solid approach advantageous when detailed contact force data is required.

IMU

The function of the IMU is discussed in detail in A.1.3.1. For now, it is important to note that the IMU is an electronic device that measures and reports a specific force, angular rate,

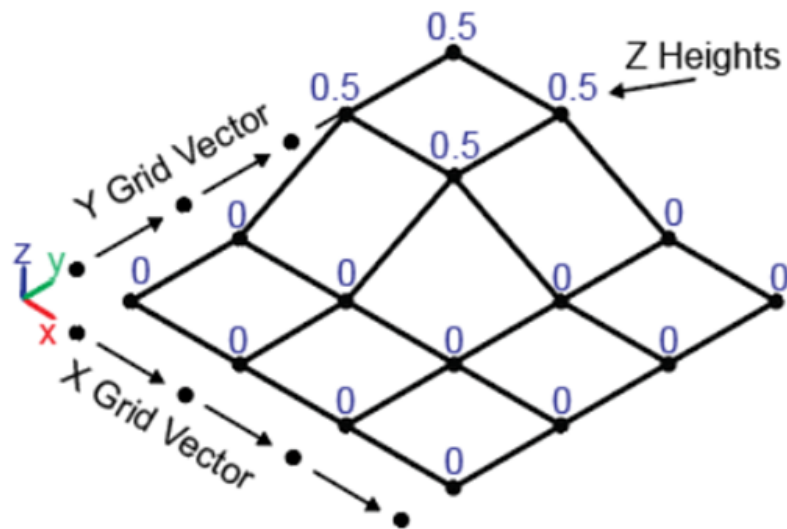


Figure 2.115: Grid surface representation illustrating X, Y grid vectors and Z height values

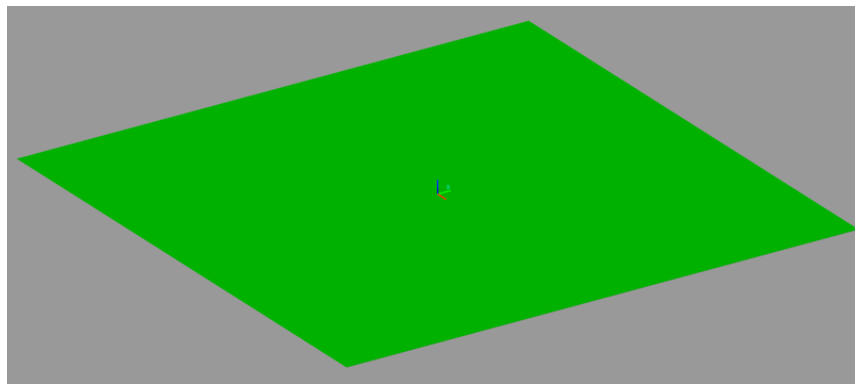


Figure 2.116: Representation of a surface using a Brick Solid block in the simulation environment

and sometimes orientation of a body using a combination of accelerometers, gyroscopes, and sometimes magnetometers.

In Simscape, the IMU is simulated using a Transform Sensor Block. This sensor block measures the relative spatial relationship between any two frames within a Simscape™ Multibody™ model. The measurements, resolved in the measurement frame of the block, include relative translations, rotations, and their first and second derivatives over time.

The frames chosen for this simulation are the world frame for the Base and the center of the rover for the Follower, as shown in Fig. 2.117.

Six-DOF joint

The block shown in Figure 2.118 represents a 6-DOF (degrees of freedom) joint between two frames. This joint has three translational and three rotational degrees of freedom,

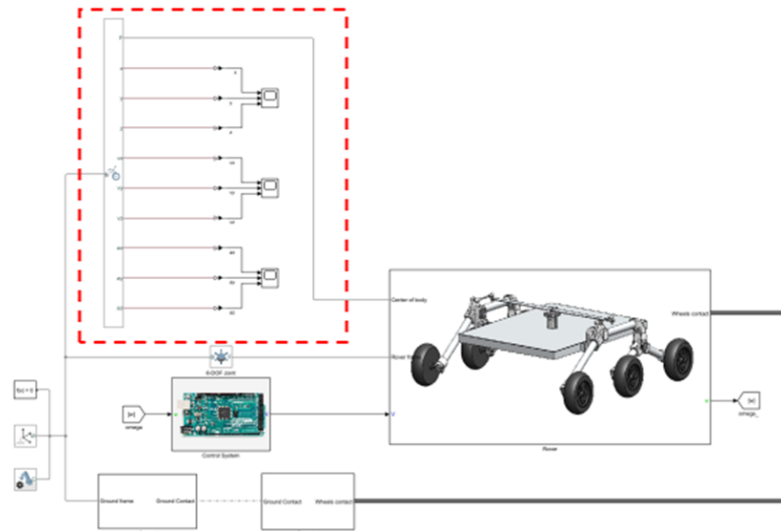


Figure 2.117: IMU Sensor sub-block integrated within the multibody model

represented by three prismatic primitives along a set of mutually orthogonal axes, and one spherical primitive. It allows unconstrained 3-D translation and rotation. Specifically, the follower origin translates relative to the base frame, after which the follower frame can rotate freely around the follower origin as a fulcrum.

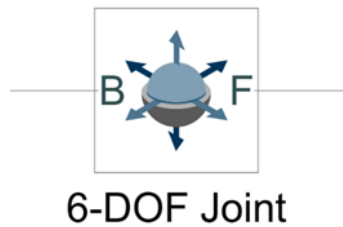


Figure 2.118: Six-DOF joint block enabling unconstrained 3-D translation and rotation

Optimization of the Computational Time

In order to optimize the computational time required to complete the simulation, various simulations were performed by adjusting certain parameters and recording the time required to simulate a distance of 200 meters. The parameters that remained constant during these tests included those of the DC motor and the PID controller, which were set as follows:

- DC Motor Parameters:
 - Supply voltage: 36 V

- Current limit: 8.33 A
- Gear ratio: 10
- No-load speed: 5000 rpm
- Stall torque: 0.8 N·m
- PID Controller Parameters:
 - Proportional Gain (K_p): 20
 - Derivative gain (K_d): 0
 - Integral gain (K_i): 10

These motor parameters were selected from the data sheet for 350W DC motors with a stall torque of 8 Nm and a no-load speed of 500 rpm. The current limit was determined from the DBLS - 01S driver data sheet, which operates at a 24V supply with a current limit of 8.33A.

For each simulation, the computational time required to complete a 200-meter simulation was recorded to evaluate the efficiency of different configurations.

As mentioned in the previous chapter, it is important to distinguish between the Convex Hull and the Point Cloud blocks. Neither is inherently better; the choice depends on the focus of the study.

The Convex Hull block is useful if the study emphasizes contact forces. Only with the Convex Hull model can the Spatial Contact Force block capture normal and frictional forces. However, this approach requires more computation and approximates the geometry, limiting its effectiveness in motion studies for complex-shaped systems.

In contrast, the Point Cloud block allows accurate modeling of the real geometry of a system. This can be achieved either by generating it through MATLAB code or by importing an STL file that can be easily created in any CAD system. An example of this setup is shown in Fig. 2.119.

Thus, the point cloud method is suitable for studies where accurate geometric representation is critical, while the convex hull method is more suitable for analyses that focus on contact force measurements.

With these considerations in mind, different simulations were performed by adjusting parameters in the coordinate matrix of the Point Cloud block, which defines the points of the geometry. The results were compared with the computational time required to complete the simulation using the convex hull. Parameters varied included:

- Number of circumferential points
- Number of circumferential lines
- Angular spacing θ

The first simulation using the convex hull took 04:05.15 [min:sec] to complete. This served as a baseline to assess the computational efficiency of the point cloud configuration by comparing the resulting times as each parameter was adjusted.

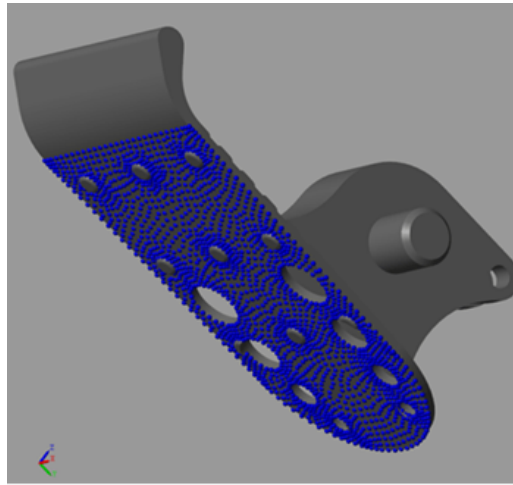


Figure 2.119: Point cloud representation highlighting the geometry and surface details of the component

Number of Circumferential Points

Using a single circumferential line, several simulations have been carried out with different number of points:

1. 200 points.
2. 400 points.
3. 800 points.
4. 1600 points.
5. 3200 points.
6. 6400 points.
7. 12800 points.

In the Tab. 2.13, the time needed to complete the simulation has been reported for each simulation. In Fig. 2.120 and Fig. 2.121 are represented the first and last case of the simulation study. Using the Curve fitting toolbox, a “number of circumferential points – time” curve has been created, using the data acquired with the simulation, as shown in Fig. 2.122.

The initial simulations required a great deal of computational time and effort. This was due to the limited number of points defining the wheel, which resulted in irregular system motion and large vibrations transmitted throughout the structure, making the simulation difficult to complete. These vibrations made it difficult to solve the differential equations governing the simulation.

Number of points	Time needed (min:sec)
200	09:00.24
400	02:14.68
800	00:06.44
1600	00:10.95
3200	00:14.07
6400	00:22.21
12800	00:41.31

Table 2.13: Simulation time required for varying numbers of circumferential points, illustrating the computational cost as the number of points increases

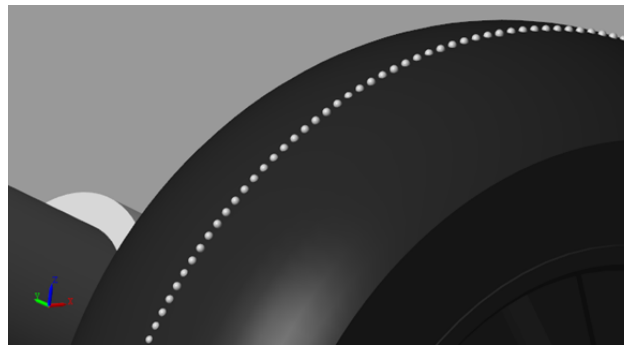


Figure 2.120: Simulation 1: 200 points

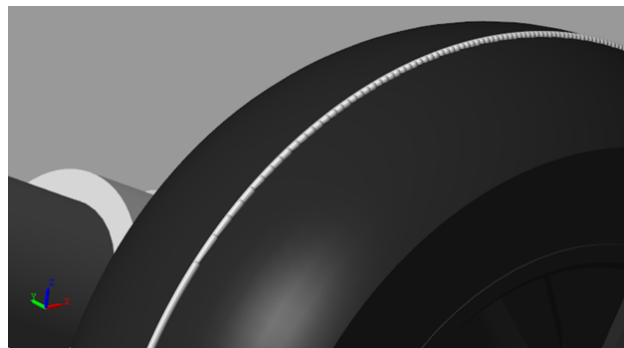


Figure 2.121: Simulation 7: 12800 points

However, once the number of points reached 800, the computational time, effort, and simulation results stabilized and showed minimal difference even when the number of points was increased to 12,800. This indicates that 800 points provided a sufficient level of detail to achieve accurate results without unnecessary computational effort.

Number of Circumferential Lines

A similar comparison has been made varying the number of circumferential lines. The simulations were carried out with the following differences:

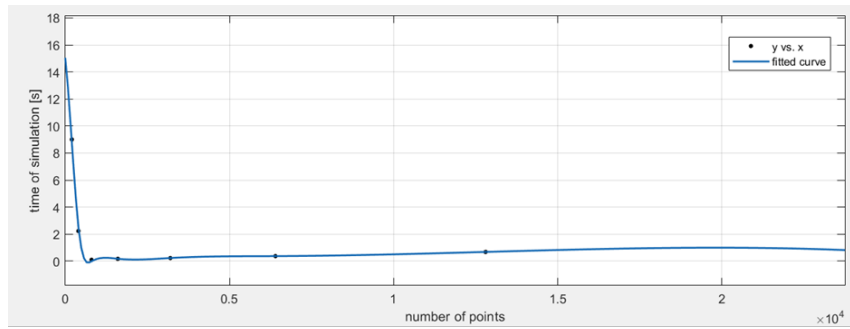


Figure 2.122: Relationship between the number of circumferential points and simulation time, including a fitted curve for better trend analysis

1. 31 circumferential lines.
2. 15 circumferential lines.
3. 7 circumferential lines.
4. 3 circumferential lines.
5. 1 circumferential line.

For this comparison, as shown in Fig. 2.123, the angular separation θ between each circumferential line was set to 1° . This configuration aimed to achieve a balance between detail and computational efficiency in the point cloud model. Each circumferential line

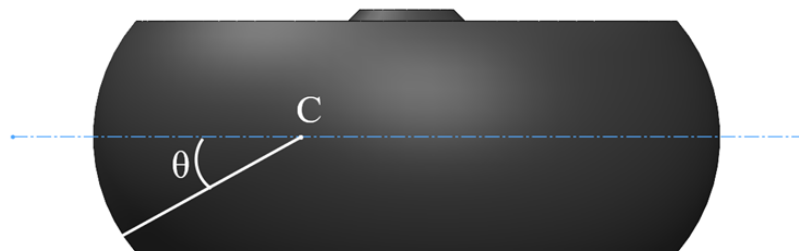


Figure 2.123: Angular distance θ

has 800 points. In the Tab 2.14 the time needed to complete the simulation has been reported for each simulation.

In Fig. 2.124 and Fig. 2.125, two different representations in simulations.

Using the Curve fitting toolbox, a “number of circumferential lines – time” curve has been created, using the data acquired with the simulation, as shown in Fig. 2.126. There is a direct proportional relationship between the simulation run time and the number of circumferential lines. As more circumferential lines are modeled, the time required to

Number of circumferential lines	Time needed (min:sec)
31	01:53.53
15	00:53.17
7	00:26.41
3	00:12.23
1	00:06.44

Table 2.14: Simulation time required for different numbers of circumferential lines

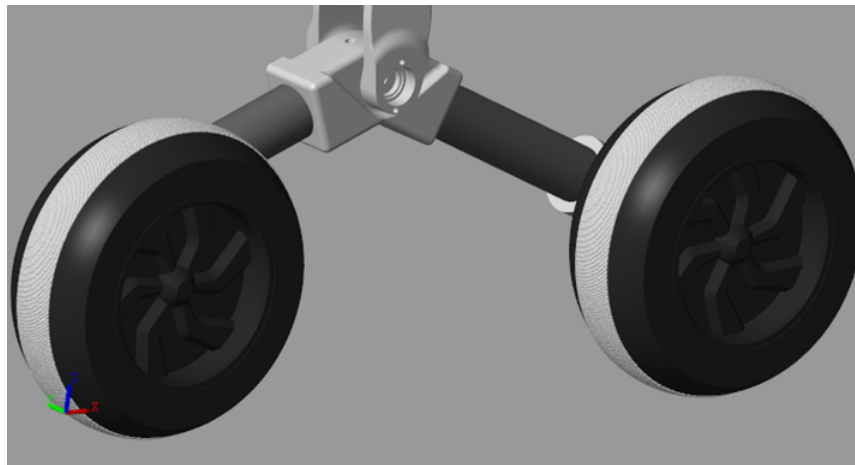


Figure 2.124: Simulation 1: 31 circumferential lines

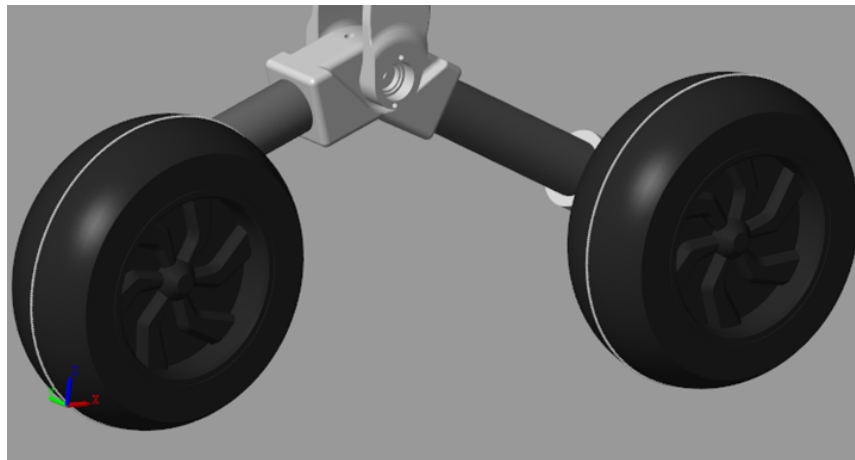


Figure 2.125: Simulation 5: 1 circumferential line

complete the simulation increases almost proportionally. This indicates that increasing the level of detail by adding circumferential lines increases the computational load, which affects the overall efficiency of the simulation.

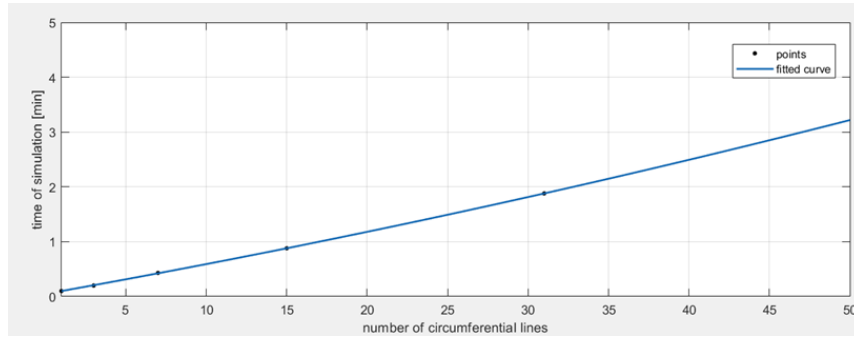


Figure 2.126: Fitted curve showing the relationship between the number of circumferential lines and the simulation time

Angular Distances θ

One more comparison could be done using different values of θ . Considering 31 circumferential lines, each one made of 1600 points, different simulations have been carried out:

1. $\theta = 1^\circ$.
2. $\theta = 0.5^\circ$.
3. $\theta = 0.1^\circ$.
4. $\theta = 0.05^\circ$.
5. $\theta = 0.01^\circ$.

In the Tab. 2.15, the time needed to complete the simulation has been reported for each simulation.

Angular distance θ	Time needed (min:sec)
1°	01:53.53
0.5°	02:42.40
0.1°	02:46.22
0.05°	02:50.93
0.01°	02:48.25

Table 2.15: Simulation time required for different angular distances θ

In the Fig. 2.127 and Fig. 2.128 is possible two different theta configuration with in particular in Fig. 2.129 a zoom in Simulation 5.

Using the Curve fitting toolbox, a “ θ -time” curve has been created, using the data acquired with the simulation, as shown in Fig. 2.130. As the value of the angle θ increases, the simulation completion time begins to decrease. This is due to the reduction in the number of circumferential lines involved in the contact, which reduces the computational load and speeds up the simulation process.

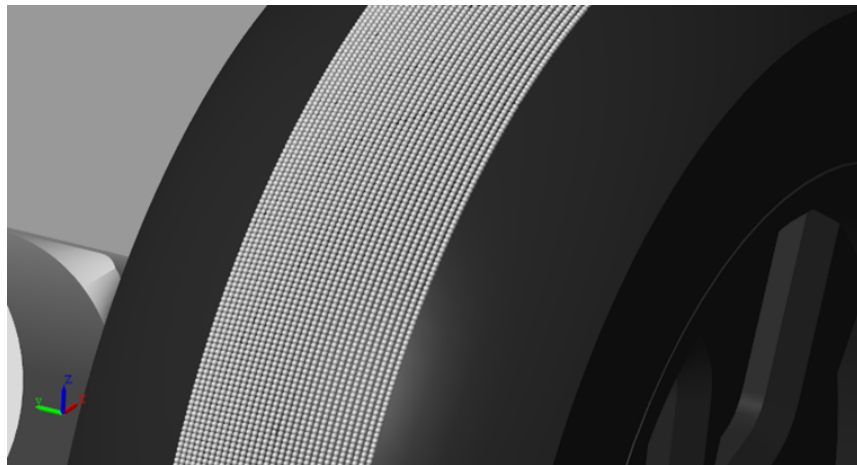


Figure 2.127: Simulation 3: $\theta = 1^\circ$



Figure 2.128: Simulation 5: $\theta = 0.01^\circ$



Figure 2.129: Simulation 5: $\theta = 0.01^\circ$, zoom

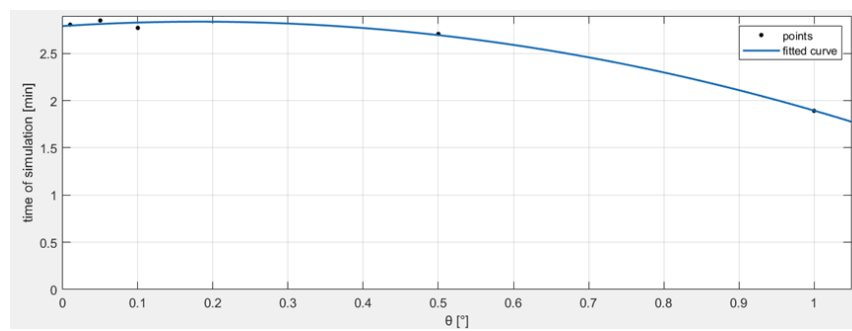


Figure 2.130: Fitted curve showing the relationship between the angular distance θ and the simulation time

2.4.1.3 Simulation Results for Obstacle Crossing

In this section, simulations were performed to analyze the behavior of the rover in overcoming obstacles, its performance on irregular surfaces, and its energy consumption on both flat and sloped surfaces. These simulations used the "Point Cloud" block described in the previous section, configured with three circumferential lines, each consisting of 1600 points, and an angular separation of 0.5° between the lines. This setup provided a detailed representation of the interaction of the rover with the terrain, allowing an accurate study of its dynamics and energy efficiency.

In the first sections of this chapter, the behavior of the rover in overcoming an obstacle is analyzed for both configurations: bogie at the front and rocker at the front. The goal is to determine the most effective advancement configuration by comparing performance in each scenario.

Obstacle Crossing with the Bogie

In overcoming the obstacles, the speed reference has been set equal to 1 rad/s for each wheel.

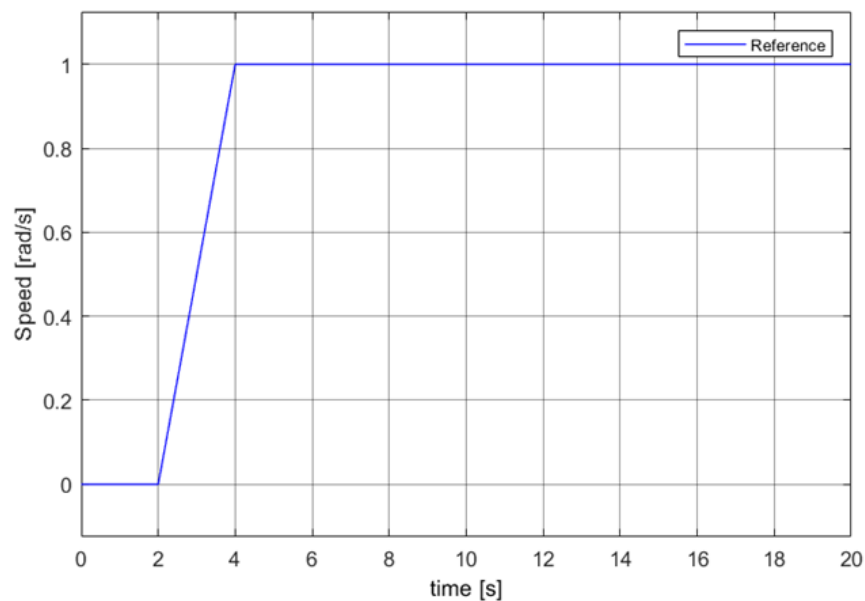


Figure 2.131: Torque profile of the front motors during obstacle crossing, with a reference speed of 1 rad/s for each wheel

In the first simulation the obstacle has been modelled through the Grid surface block. Its height has been set to 100 mm, as shown in Fig. 2.132. From Fig. 2.133 it can be seen that the rover successfully passes the obstacle, as none of the motor torques exceed the imposed current limit of 5.55 N-m. A first observation is that the torque of each motor is affected not only by the pass of its corresponding wheel, but also by the motion of other wheels. This highlights the interdependence of the dynamics of the rover on the

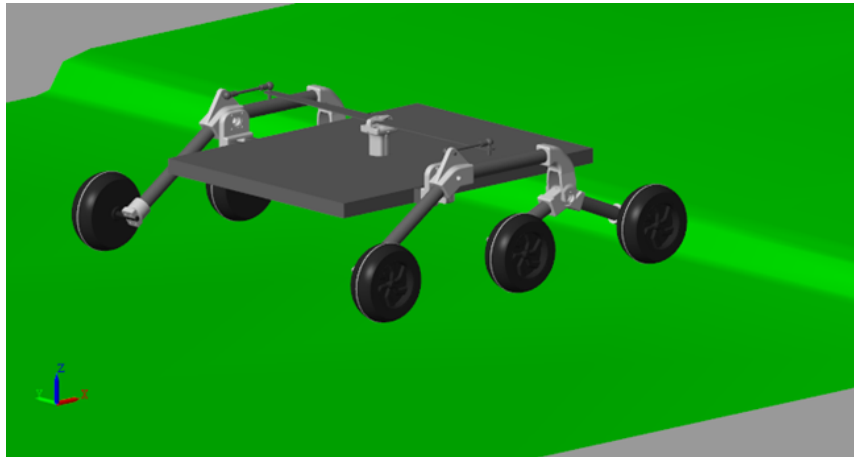


Figure 2.132: Obstacle crossing, Bogie Front: Case 1

individual motors. In addition, the increase in torque is not uniform as each wheel passes. For example, the front motor experiences its largest torque increase not when the front wheels cross the obstacle, but when the center wheels pass the obstacle. Similarly, for the rear motor, the largest torque increase occurs when the front wheels cross the obstacle, due to the change in the setup of the rover that increases the load on the rear wheels. In contrast, the center motor shows consistent torque increases for each wheel crossing the obstacle, indicating a more balanced load distribution for this motor. These observations are critical to understanding the mechanical interactions within the rover and optimizing its performance during obstacle traversal.

In the second simulation, the obstacle was modeled with a height of 100 mm, as shown in Figure 2.134. This configuration was used to analyze the performance and motor response of the rover when encountering a more challenging terrain feature. In this case, overcoming the obstacle is more challenging than in the previous scenario. Because the height of the obstacle exceeds the wheel radius, the rover requires higher torque on all wheels to successfully cross the obstacle (see Fig. 2.135). The rear wheels get more torque than the others when the front and middle wheels are in contact with the obstacle. However, during the final phase of overcoming the obstacle, the rover is lifted primarily by the front and center wheels, which take on a greater portion of the load. This dynamic shows the increased demands placed on the drive train and suspension system as the rover negotiates larger obstacles.

Obstacle Crossing with the Rocker

In the first case, the obstacle has a height of 100 mm and the simulation appears like in Fig. 2.136. In Fig. 2.137 it can be seen that, similar to the first case with the bogie front, the rover successfully overcomes the obstacle. However, compared to the bogie front configuration, the torques in this scenario are significantly higher, exceeding 5 Nm. On average, the required torques are higher in this configuration, and the conditions under which each motor experiences maximum torque are different. For example, in the case

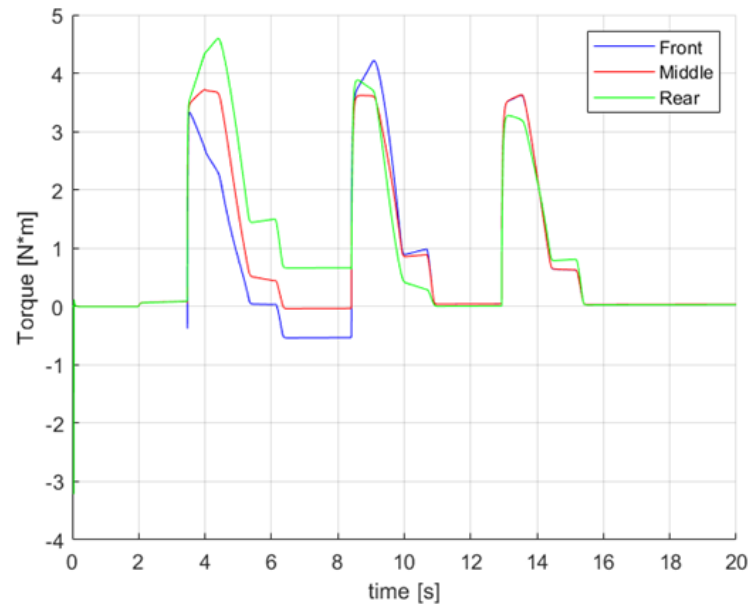


Figure 2.133: Torque profiles of the front, middle, and rear motors during obstacle crossing with Bogie Front: Case 1

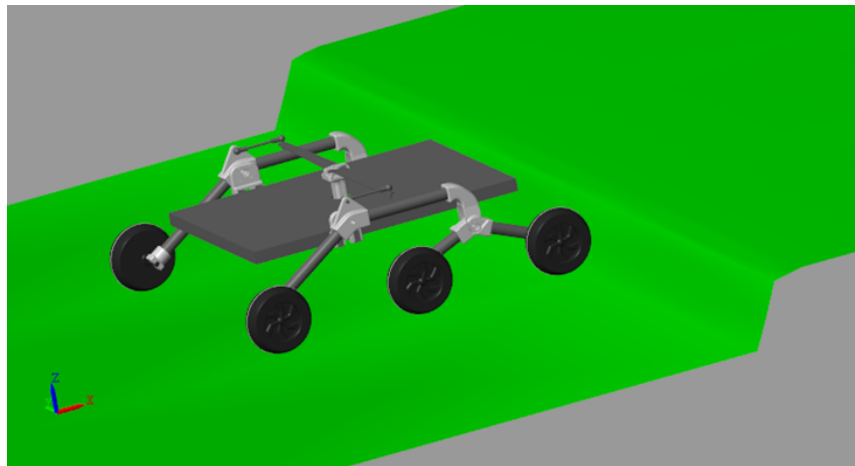


Figure 2.134: Obstacle crossing, Bogie Front: Case 2

of the front motors, the maximum torque occurs when the front wheels are overcoming the obstacle. This difference is due to the different load distribution between the two configurations, which affects the dynamics and torque requirements during the maneuver.

In the second simulation case, the obstacle height remains the same as in the corresponding bogie-front case study: 330 mm. The simulation shown in Fig. 2.138 provides a comparative analysis of the behavior and torque requirements of the rover under similar conditions, but with the rocker-front configuration.

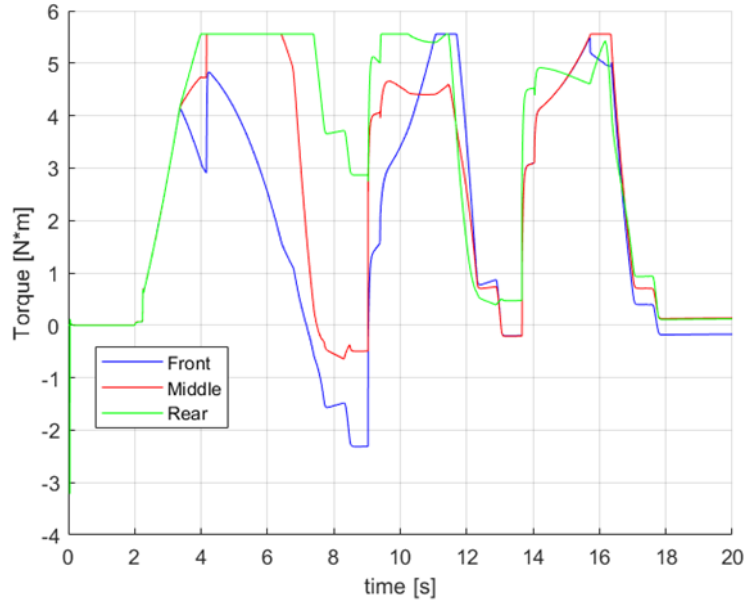


Figure 2.135: Torque profiles of the front, middle, and rear motors during obstacle crossing with Bogie Front: Case 2

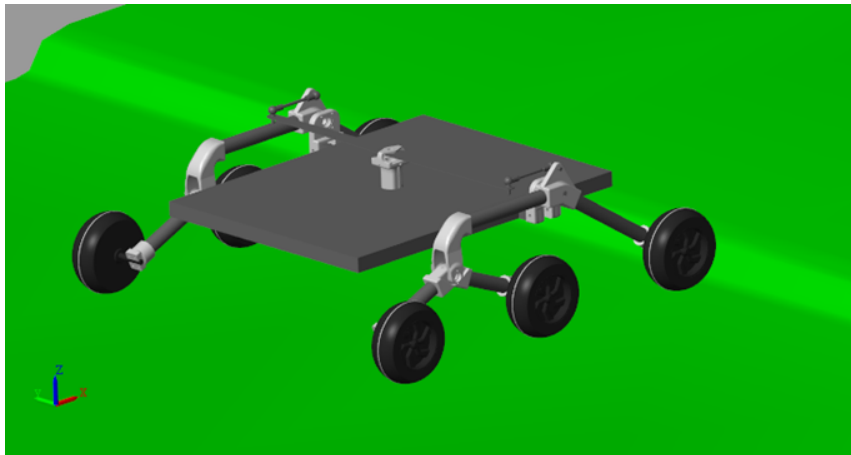


Figure 2.136: Obstacle crossing, Rocker Front: Case 1

In this last case, as it possible to see in Fig. 2.139, the rover is unable to overcome the obstacle. Both the front and rear wheels provide their maximum torque, but this is not enough to get the rover over the obstacle. This result shows that the best advance configuration is with the bogie at the front, as it can overcome higher obstacles while requiring less torque, demonstrating superior efficiency and capability in challenging terrain conditions.

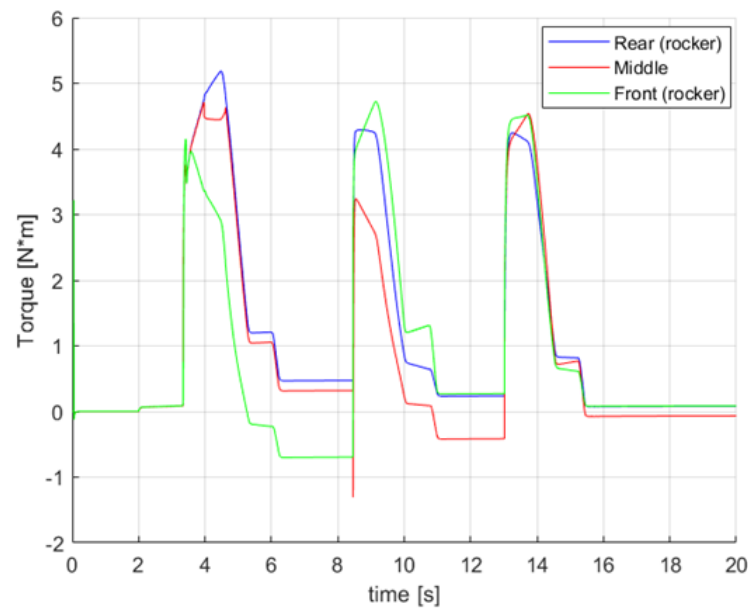


Figure 2.137: Torque profiles of the front, middle, and rear motors during obstacle crossing with Rocker Front: Case 1

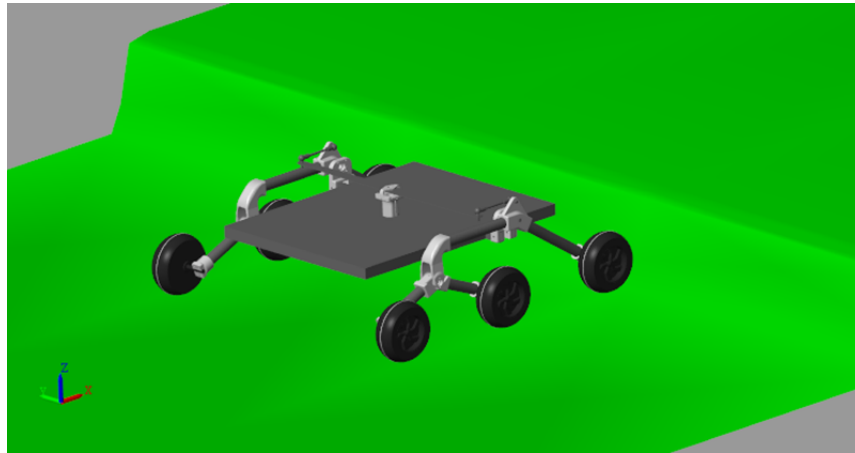


Figure 2.138: Obstacle crossing, Rocker Front: Case 2

2.4.1.4 Irregular Surface Simulations

The introduction of the Grid Surface and Point Cloud blocks made it possible to simulate the behavior of the rover on highly irregular surfaces in a very short time. Since the input to the Grid Surface block consists of two vectors and a matrix, a script was developed to generate the irregular surface by simply selecting parameters such as the height of peaks and valleys. This approach allows for efficient and customizable terrain modeling, providing a practical method for analyzing the performance of the rover under various conditions. By defining maximum peak heights and minimum valley depths, and ran-

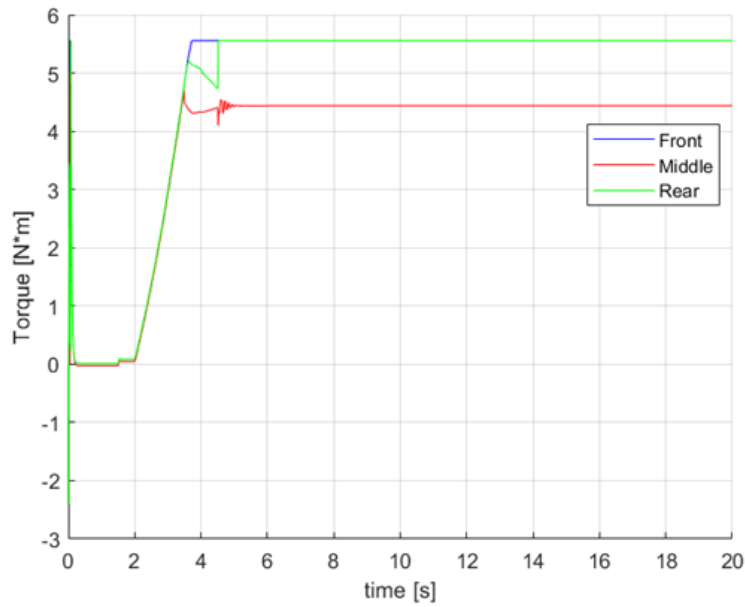


Figure 2.139: Torque profiles of the front, middle, and rear motors during obstacle crossing with Rocker Front: Case 2

domizing hill heights to stay within these ranges, a surface like the one shown in Figure 2.140 can be generated. The parameters were chosen to effectively simulate the irregular terrain of an agricultural field, providing a realistic environment for testing and analyzing the performance of the rover.

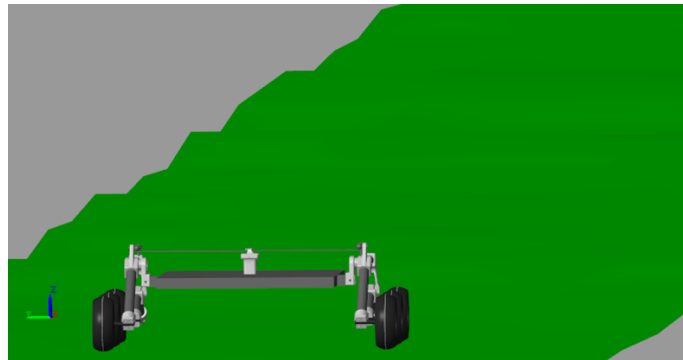


Figure 2.140: Representation of an irregular surface using a grid-based model

Another example of an irregular surface is shown below. This simulation was run to analyze the behavior of the rover on an extremely irregular terrain that would be unrealistic for agricultural fields. The Δh (height difference) between peaks and valleys was set to 24 cm, creating a challenging terrain to evaluate the performance of the rover under extreme surface conditions, as in Fig. 2.141. On very uneven surfaces, it is quite challenging for the rover to maintain a straight forward trajectory. The y_c , which

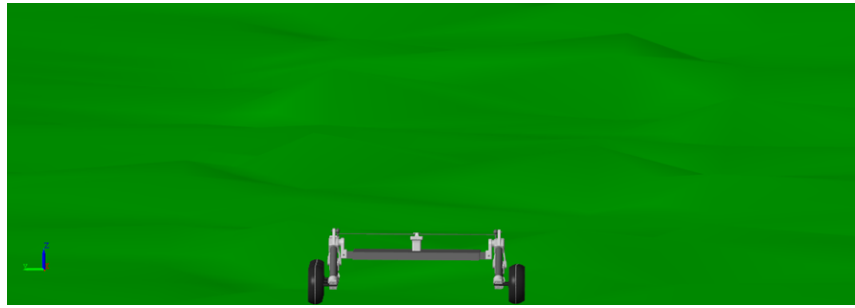


Figure 2.141: Representation of an irregular surface using a grid-based model

represents the lateral deviation, is not always zero; its value varies depending on the obstacles the rover encounters and their positions on the ground (see Fig. 2.142 and Fig.2.143).

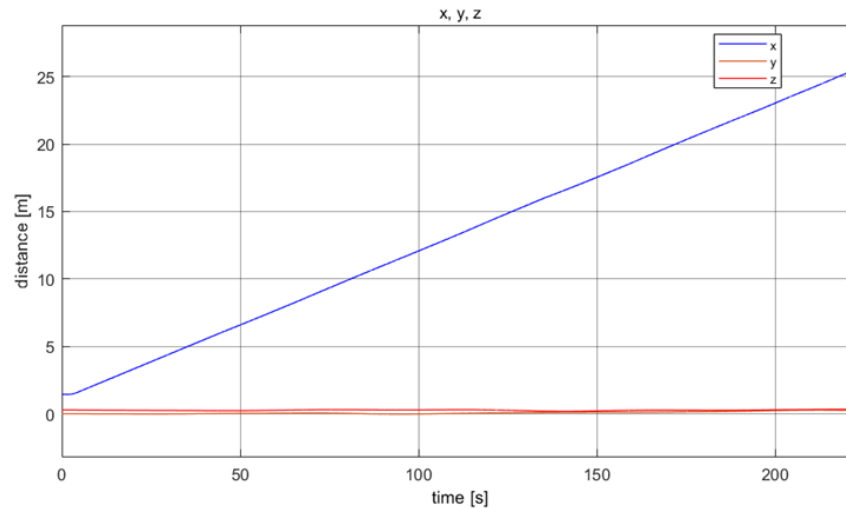


Figure 2.142: Position of the center of the rover over time along the x, y, and z axes during traversal of uneven terrain

In the presence of such uneven terrain, observing only the voltage supplied to the six motors shows (see Fig.2.144) that the wheel speeds differ significantly. This observation highlights the need to use sensors, such as an IMU, to monitor and adjust the trajectory. Sensors that do not rely on the evaluation of individual wheel speeds are essential to ensure accurate trajectory tracking and maintain the desired trajectory under such conditions.

2.4.1.5 Energy Consumption Estimation

Additional simulations were performed to estimate the energy consumption of the rover. The model used for these simulations is similar to the one used in the previous studies, but requires significantly more computation. The main difference is in the implementation of the battery subsystem. This subsystem takes as input the currents absorbed by the six

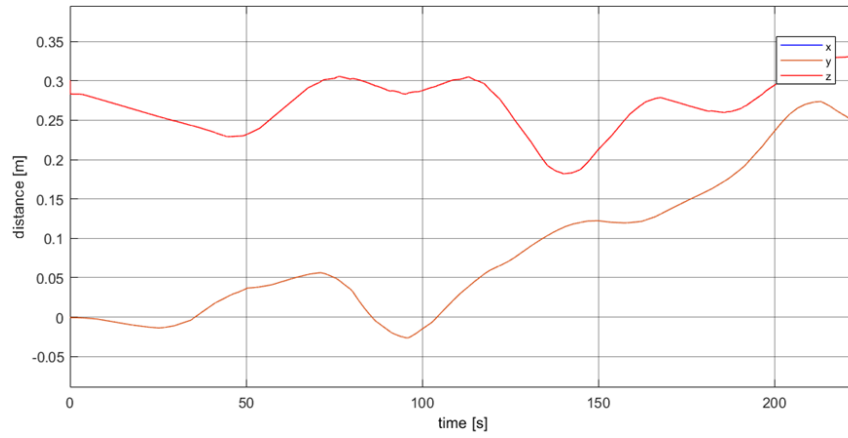


Figure 2.143: Zoomed view of the position of the rover along the z_c and y_c axes over time

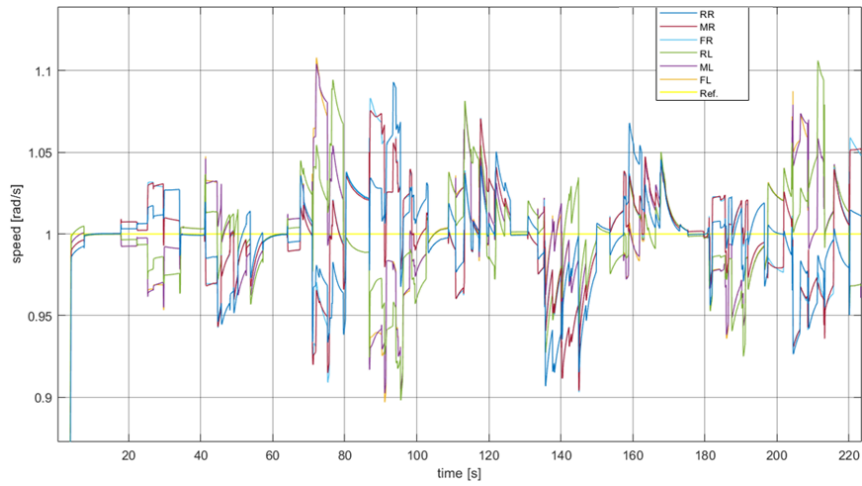


Figure 2.144: Speed profiles of the six wheels during traversal of uneven terrain

motors and outputs the voltage limits for the motors. This extension allows a detailed analysis of the energy consumption and performance dynamics of the rover under different operating conditions. In Fig. 2.145, the battery parameters used were selected based on the specifications of a commercial lithium-ion battery for hoverboard. These parameters ensure realistic simulation results and reflect the performance and energy characteristics typically associated with such batteries. For this prototype, three batteries were selected for each wheel axis to provide sufficient power for the six motors while allowing for efficient energy distribution and management. This configuration was designed to meet the energy requirements of the rover during operation in a variety of conditions.

Flat Surface

The energy consumption of the rover has been first tested on a flat surface. The speed of the rover has been set to 5 km/h. Without considering the energy consumed by sensors or additional actuators and assuming a 10Kg robotic arm and a linear decreasing trend of

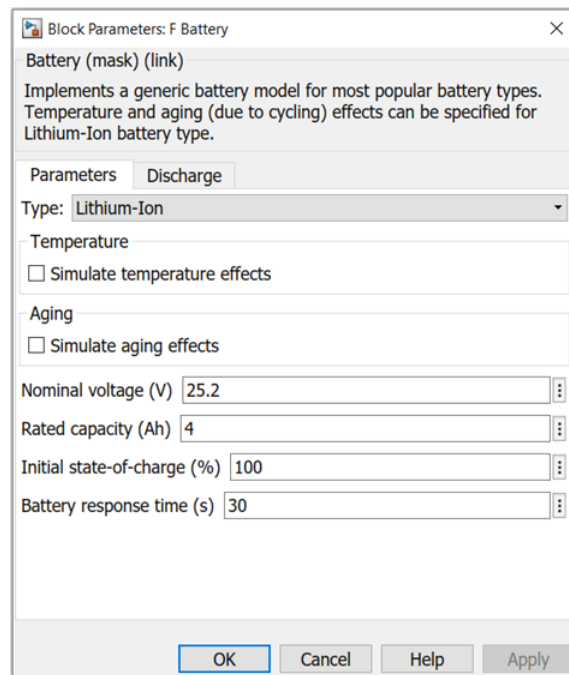


Figure 2.145: Battery parameters configuration

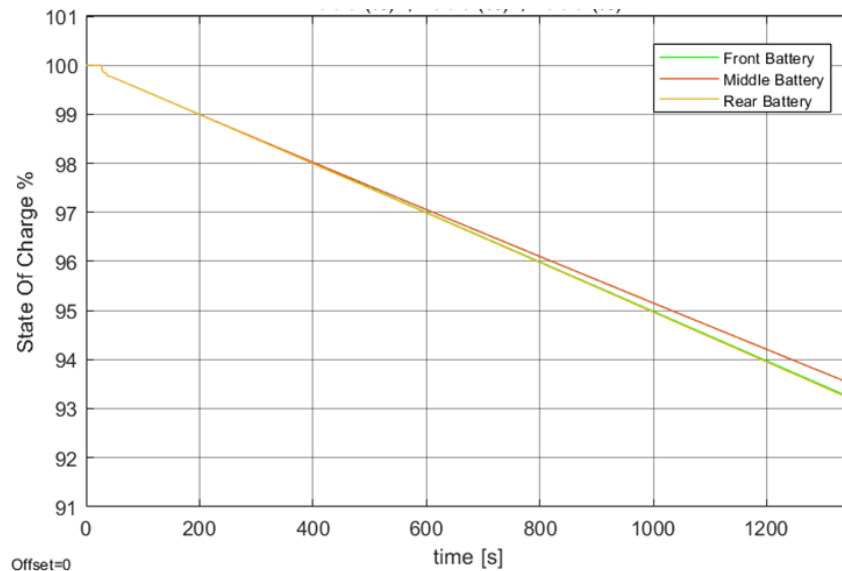


Figure 2.146: Energy consumption in Case 1, showing the state of charge for the front, middle, and rear batteries over time

the State of Charge (Fig. 2.146), the rover can operate on a flat surface for about 22,500 seconds (about 6 hours), according to the simulations. In the case of the rover, the total weight of the structure is significantly less than 75 kg, and unlike the hoverboard with two drive wheels, the rover is equipped with six drive wheels which distributes the load

more effectively than a for example. Under these conditions, the rover can travel about 30 km without recharging, making it highly efficient for extended operations.

Inclined Surface

A surface with a slope of approximately 20° was created using the Grid Surface block to test the behavior of the rover under challenging conditions (Fig. 2.147). Due to the steep slope, the speed of the rover was set to 0.5 km/h to ensure stability and control during the simulation. This setup allowed the evaluation of the performance, power consumption, and traction capabilities of the rover on a significantly sloped surface.

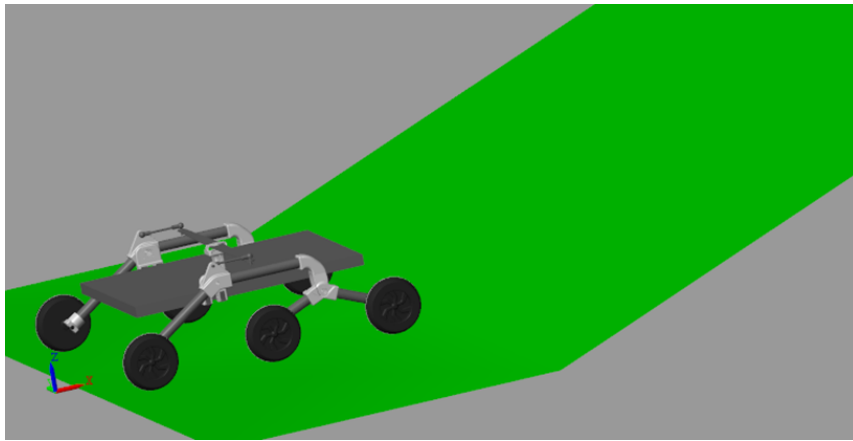


Figure 2.147: Rover navigating an inclined surface with a slope of approximately 20° , used to evaluate performance and energy consumption under challenging conditions

In this scenario, the torques required to overcome the steep slope were much higher, resulting in much greater energy consumption than in the previous flat surface case. Since the weight of the rover is predominantly distributed over the center and rear wheels, the battery powering the rear wheels is the first to be discharged. Under these conditions, the operational autonomy of the rover is about 1,100 seconds (about 20 minutes), during which it can cover a distance of about 150 meters. This illustrates the impact of steep slopes on energy efficiency and battery performance (Fig. 2.148).

As soon as the state of charge of the Middle and of the Rear batteries gets zero, the front motors cannot explicate the needed torque to let the rover continue its linear motion, so it goes back to the start, as shown in Fig. 2.149.

Obviously, neither case represents a completely realistic situation, as the actual application context is likely to be somewhere in between. This scenario will include not only the movement of the rover, but also the use of sensors and actuators required to perform operations in the environment. These additional elements will have a significant impact on both power consumption and operating range, making the theoretical calculations a baseline to be adapted to actual conditions.

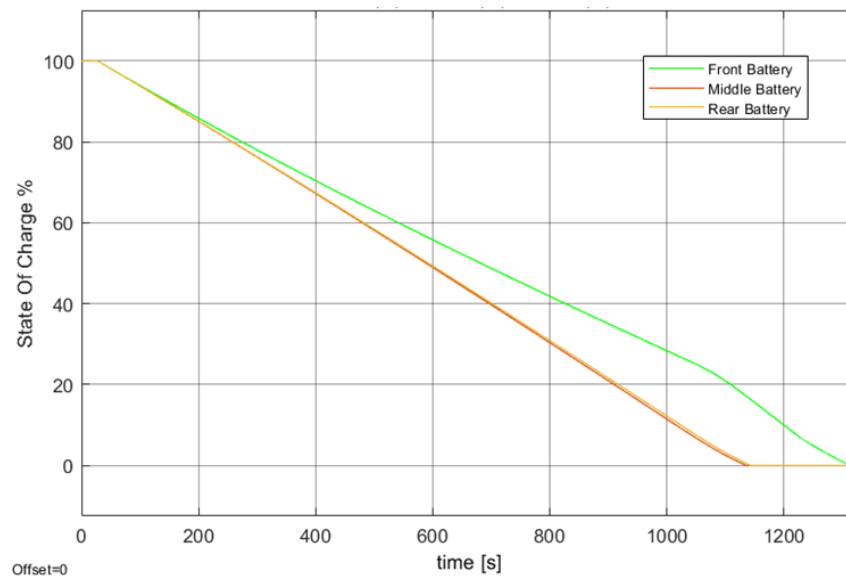


Figure 2.148: Energy consumption in Case 2, showing the state of charge for the front, middle, and rear batteries over time

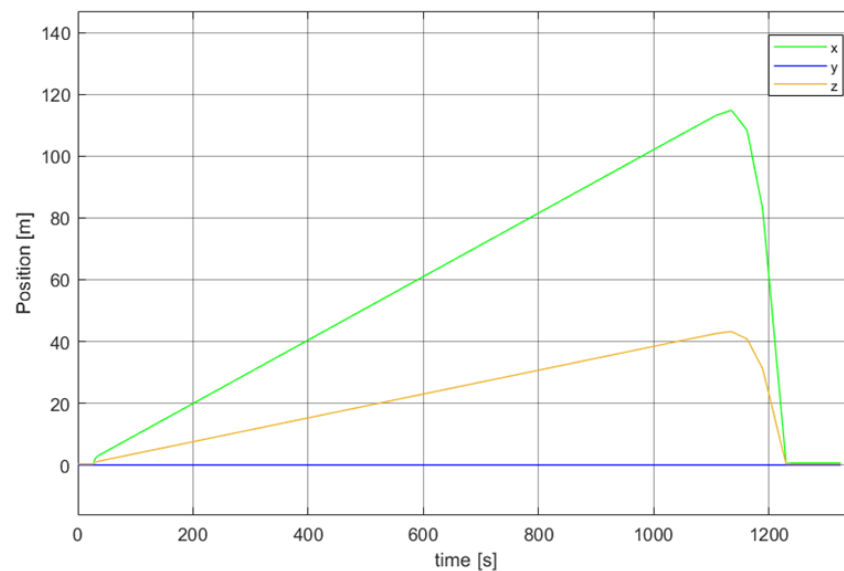


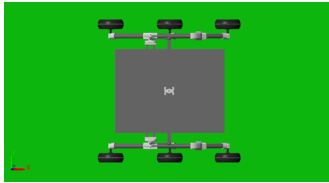
Figure 2.149: Rover position over time along the x , y , and z axes during traversal of an inclined surface

2.4.1.6 Study of Alternative Configurations

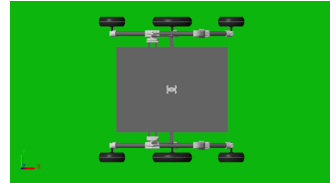
The purpose of this study is to use multibody modeling to identify design solutions that improve the maneuverability of the rover. Both rectilinear straight-line motion and curvilinear motion have been analyzed, with the torques of the six motors reported and compared across four different rover configurations.

The first configuration corresponds to the structure described in the previous paragraph, while the other three configurations differ from the first in the following aspects:

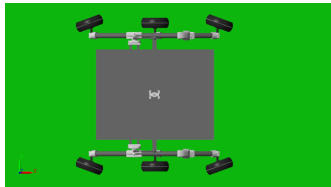
- One has bigger middle wheels for both sides (Fig. 2.150b).
- One has both front and rear steered wheels (Fig. 2.150c).
- One has middle wheels more distant from the center of the rover than the other four (Fig. 2.150d).



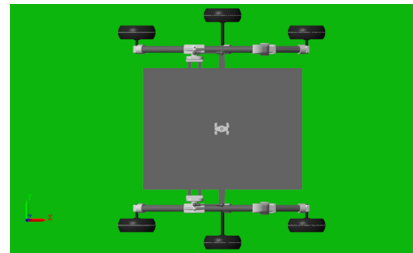
(a) First configuration (standard)



(b) Second configuration



(c) Third configuration



(d) Fourth configuration

Figure 2.150: First Prototype Configurations

Straight-Line Motion

For the first type of motion, forward rectilinear movement, it was necessary to command each motor to operate at the same speed. Across all four configurations, the trend is shown in Fig. 2.151. The only variation in reference speed is found in the second configuration,

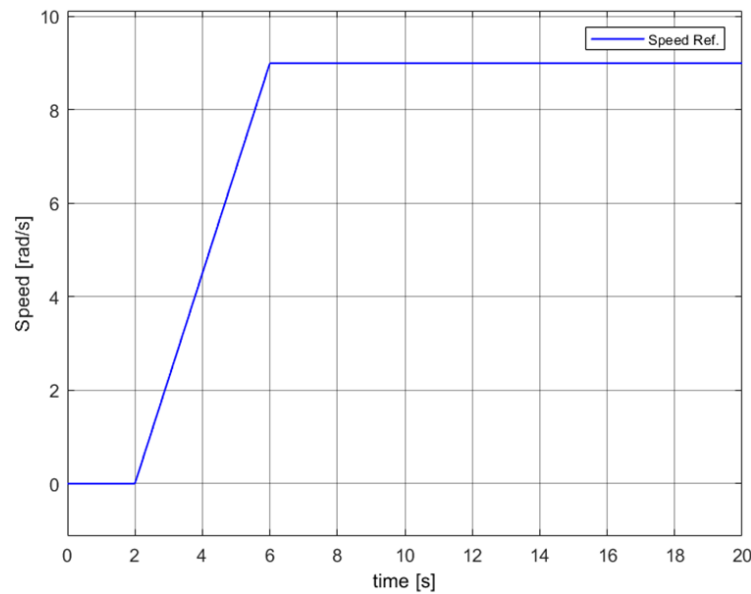


Figure 2.151: Speed reference profile over time, illustrating the target rotational speed in rad/s

where the center wheels are controlled with a reference speed equal to $\frac{2}{3}$ of the standard reference speed: this is because of the radius of the center wheels is $\frac{3}{2}$ that of the standard wheels as in Fig.2.152. For the third configuration, instead, the transversal components

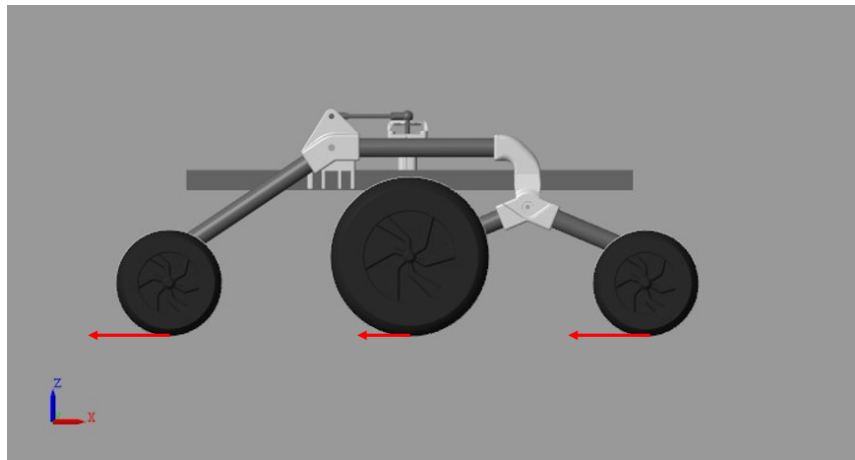


Figure 2.152: Visualization of speed differences across the wheels for the second configuration

delete each other, as shown in Fig. 2.153.

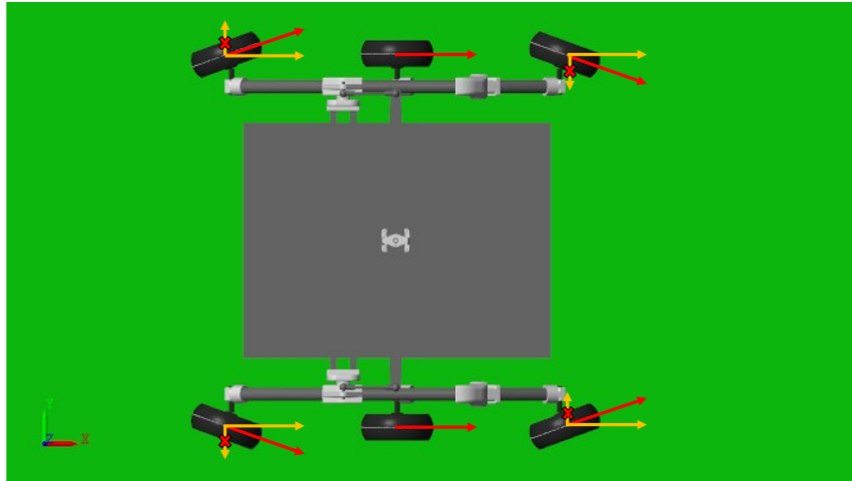
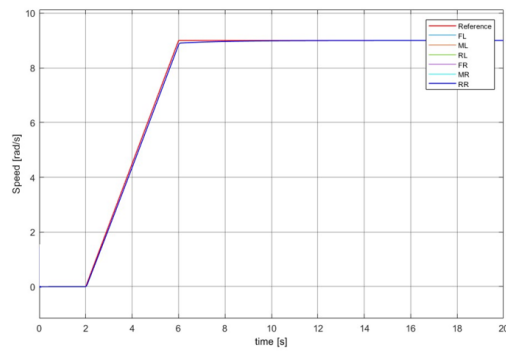


Figure 2.153: Visualization of speed components for the third configuration, highlighting individual wheel velocities

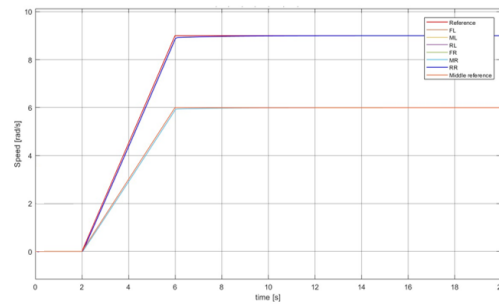
In Fig. 2.154 is showed the trend of the speed of each motor for each configuration.

For each motor and for all the four configurations, the torque has been reported in Fig. 2.155,2.156 and 2.157.

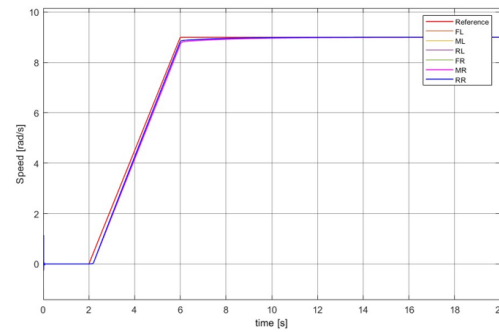
The torques required by the motors are nearly identical for the first, second, and fourth configurations. However, in the third configuration, the required torque is significantly higher. This is because the front and rear wheels are not oriented in the direction of travel, resulting in a much greater load on the two middle wheels. This uneven load distribution increases the torque demands on the center motors, highlighting the inefficiency of this configuration in facilitating smooth motion.



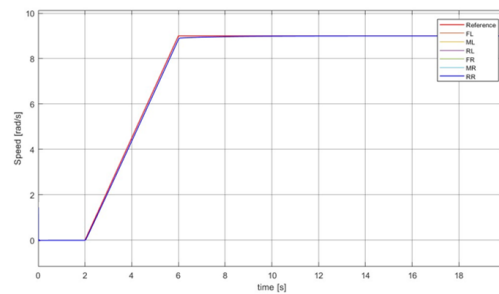
(a) Speed for First configuration (standard)



(b) Speed for Second configuration



(c) Speed for Third configuration



(d) Speed for Fourth configuration

Figure 2.154: Speed for First Prototype Configurations

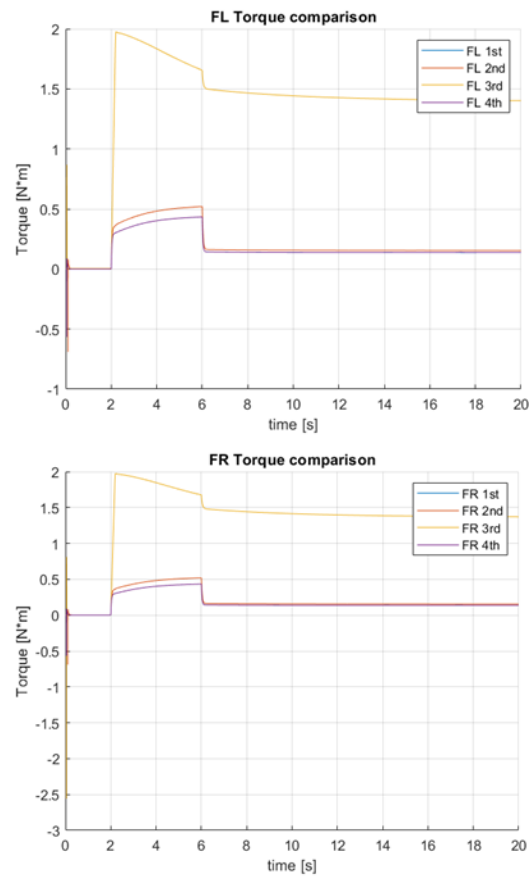


Figure 2.155: Front Motors Torque for the four configurations

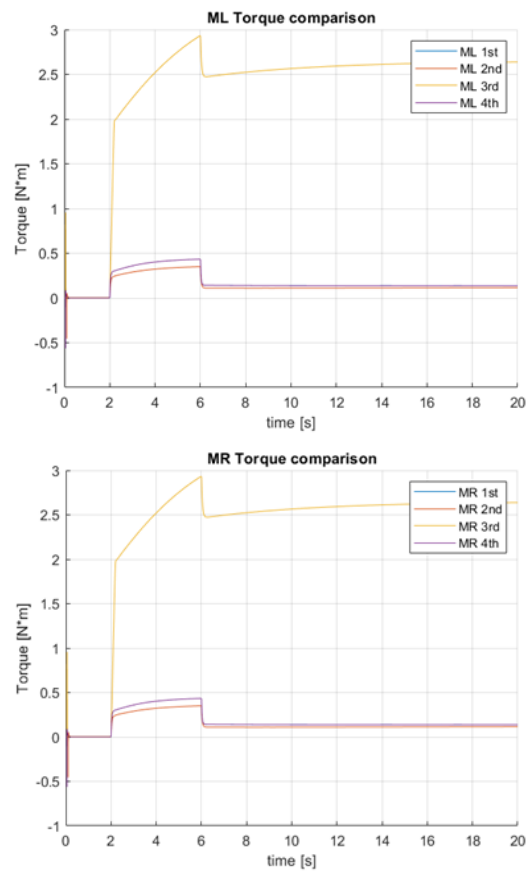


Figure 2.156: Middle Motors Torque for the four configurations

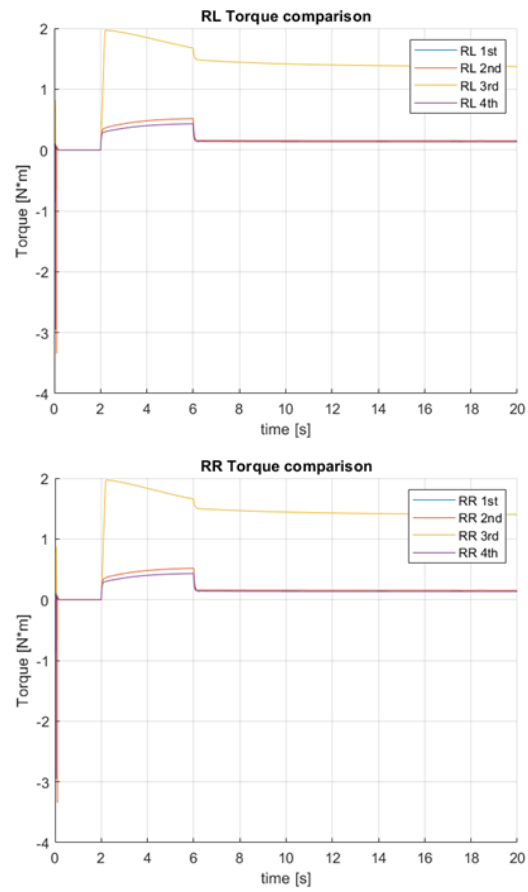


Figure 2.157: Rear Motors Torque for the four configurations

Curvilinear Motion

As discussed in previous chapters, the primary steering mode selected for the rover is skid steering, which treats the rover as a mobile robot with differential drive. This allows to have few components and constructive simplicity, allows very fast change directions, but needs higher torque values and increases tire wear.

Assigning a right speed of 0 rad/s and a left speed of 9 rad/s the scheme of the two-wheeled robot can be represented as in Fig.2.158.

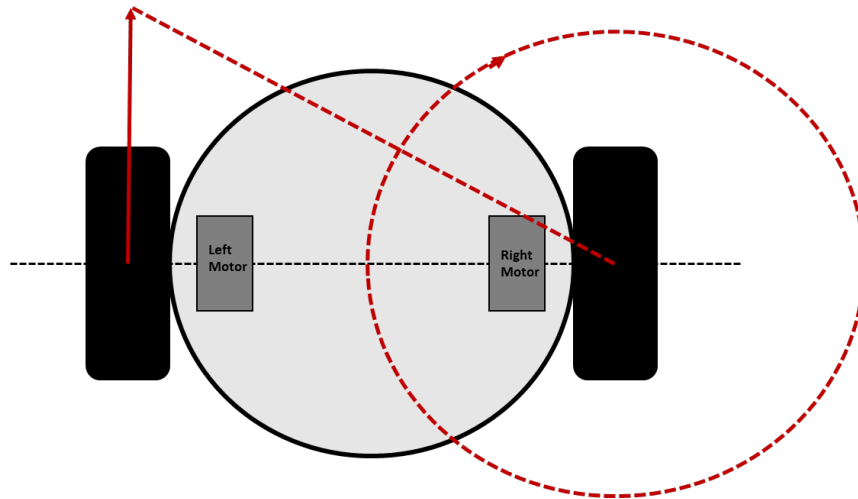


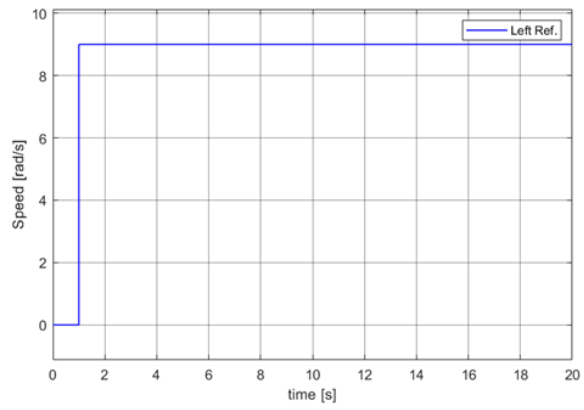
Figure 2.158: Curve scheme for the two-wheeled robot, illustrating the skid-steering behavior with differential drive

The test was conducted using a speed reference (Fig. 2.159) input for the left wheels that matched the previously defined speed reference, while the right wheels were set to a zero speed reference. This configuration allowed the behavior of the rover to be compared to that of a two-wheeled robot, effectively simulating a sharp turn.

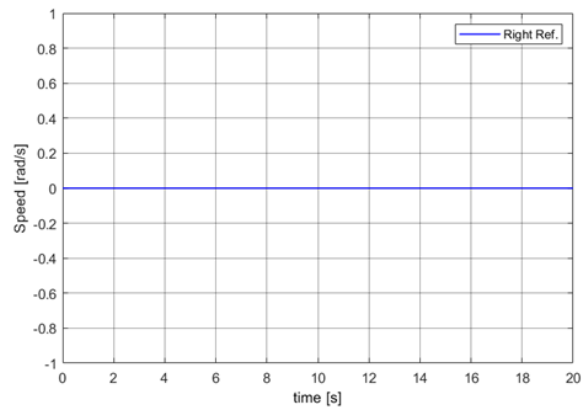
In the Fig. 2.160, 2.161, 2.162 and 2.163 are reported the trend of the speed of each motor and the x-y trajectory for each configuration.

For each motor and for all the four configurations the torque has been reported in Fig. 2.164, 2.165 and 2.166.

The torques required by the motors are nearly identical for the first, second, and fourth configurations. However, in the third configuration, the torques are significantly lower, especially for the front and rear wheels. This reduction is due to the alignment of the wheels in this configuration: the left front and left rear wheels are aligned to support the direction of the curve, while the right wheels require minimal torque because they are aligned almost opposite the curve. This orientation causes the right wheels to skid more easily due to the traction generated by the other wheels, reducing their torque requirements.



(a) Left Speed reference



(b) Right Speed reference

Figure 2.159: Speed Reference

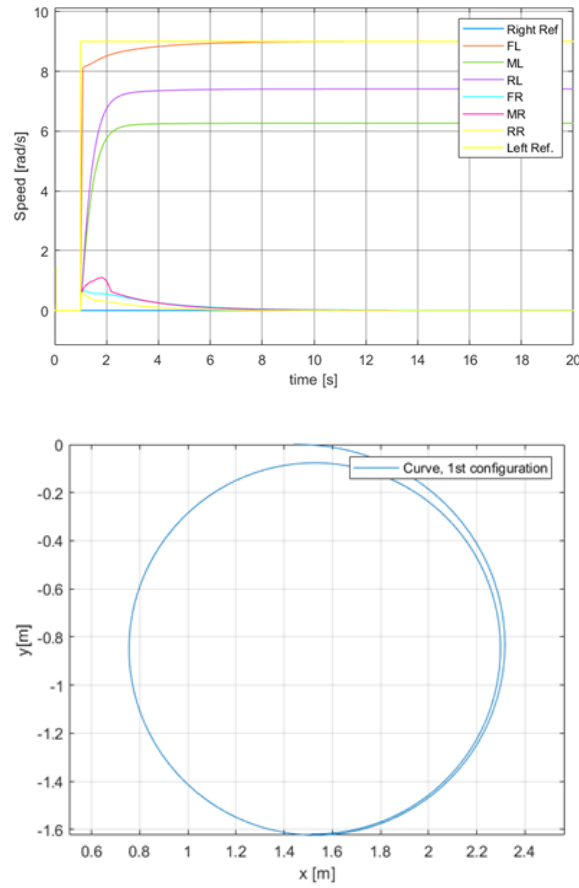


Figure 2.160: (a) Speed trend for the six motors and (b) trajectory curve for the first configuration

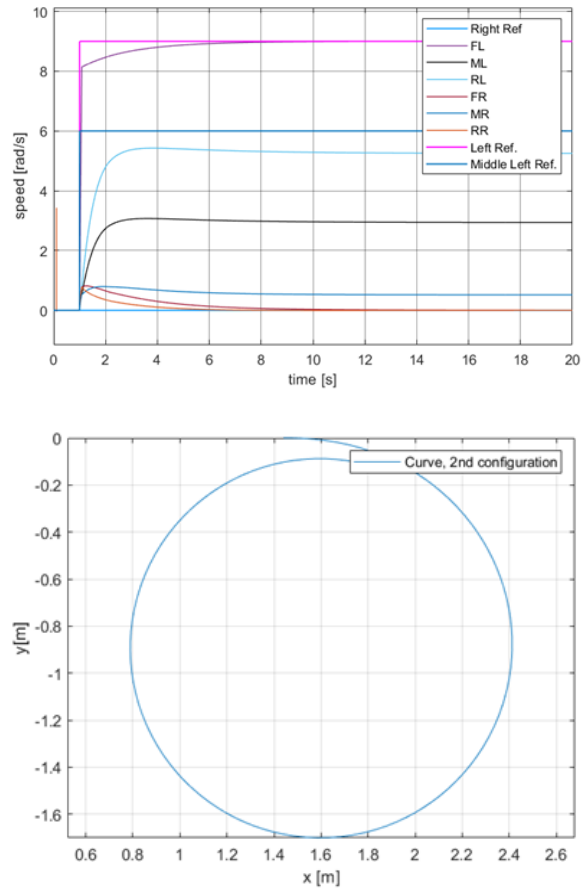


Figure 2.161: (a) Speed trend for the six motors and (b) trajectory curve for the second configuration

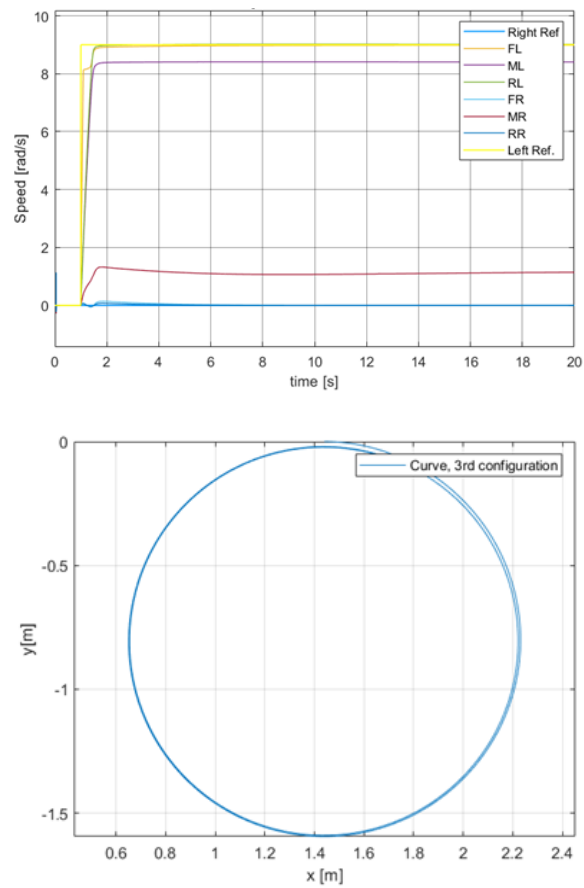


Figure 2.162: (a) Speed trend for the six motors and (b) trajectory curve for the third configuration

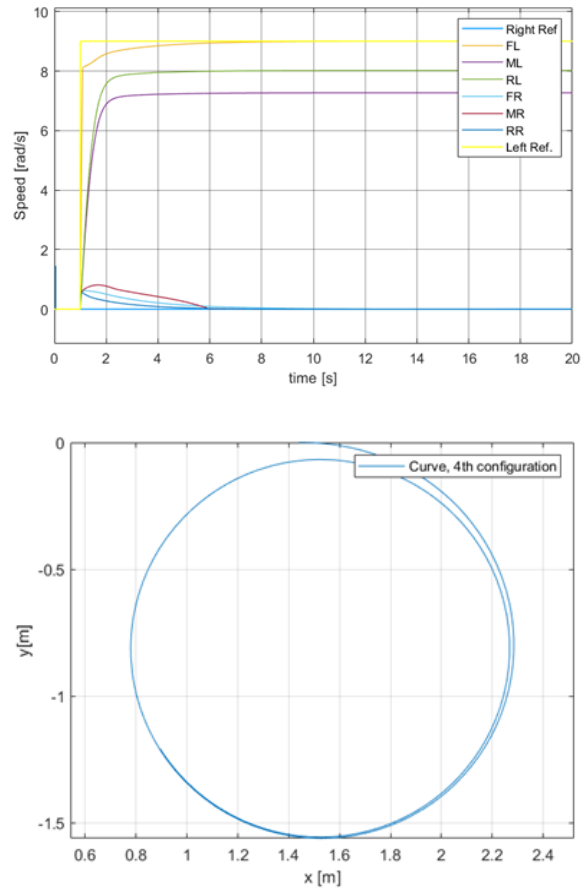


Figure 2.163: (a) Speed trend for the six motors and (b) trajectory curve for the fourth configuration

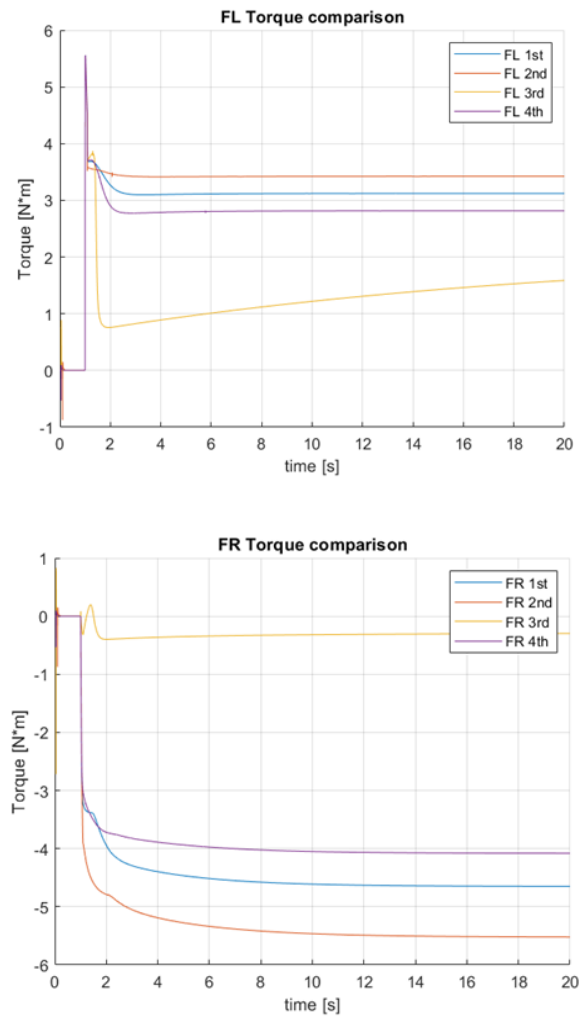


Figure 2.164: Torque comparison of the front left (FL) and front right (FR) motors for the four configurations

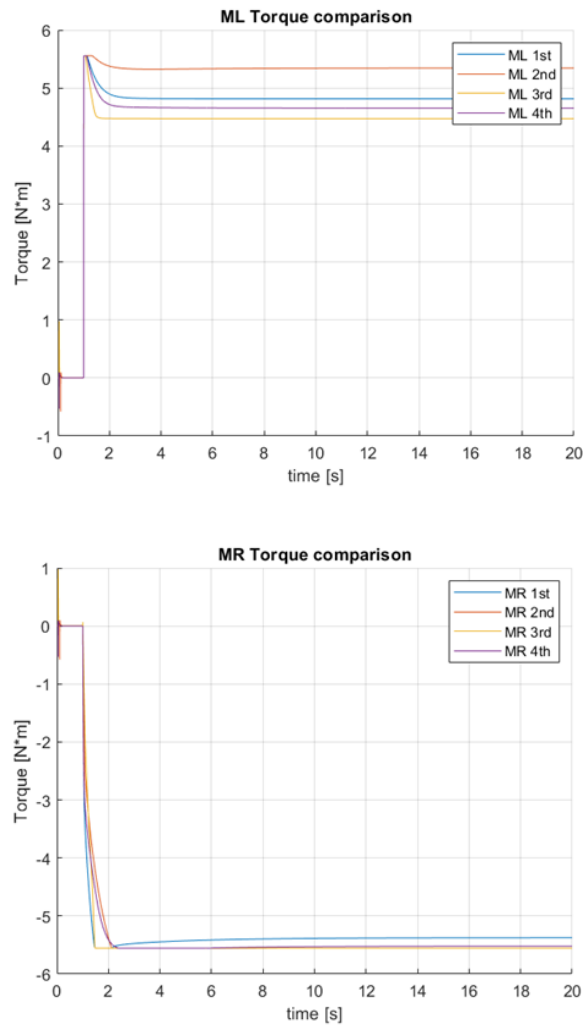


Figure 2.165: Torque comparison of the middle left (ML) and middle right (MR) motors for the four configurations

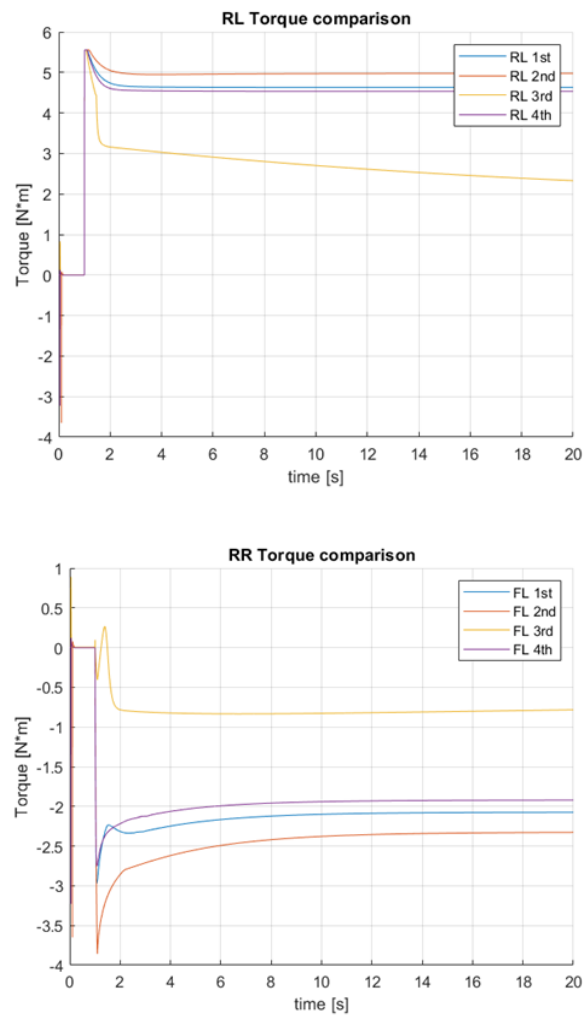


Figure 2.166: Torque comparison of the rear left (RL) and rear right (RR) motors for the four configurations

Hybrid Configuration

The analysis of the previous simulations led to the proposal of an additional configuration whose behavior was studied for the same two cases already described. This configuration introduces four additional degrees of freedom, allowing the front and rear wheels to rotate around an axis perpendicular to the ground surface within a range of $\pm 20^\circ$.

This configuration combines the benefits of the first configuration, such as lower torque requirements during straight motion, with the benefits of the third configuration, including improved trajectory control during curvilinear motion. This hybrid approach provides a balance between efficiency and maneuverability, making it a promising option for optimizing rover performance in different operational scenarios.

The challenge with this configuration is its unpredictable behavior when navigating irregular terrain or approaching angled obstacles, as shown in Figure 2.168. The additional degrees of freedom, while beneficial for certain scenarios, can complicate the stability and trajectory control of the rover under these conditions. This unpredictability results from the dynamic interactions between the wheel orientations and the uneven terrain, which can lead to unexpected torque distributions and potential difficulties in following the desired path.

An interesting area of study for this configuration would be to provide the system with actuators capable of locking the rotation of the wheels during straight motion. This approach could provide the benefits of improved maneuverability during curvilinear motion while maintaining stability and predictability during straight-line travel.

By selectively engaging or disengaging wheel rotation based on terrain or motion requirements, the system could optimize torque distribution and minimize energy consumption. Such an adaptive mechanism would also improve the ability of the rover to handle different operational scenarios, such as navigating irregular terrain or encountering rotated obstacles, while keeping control and efficiency.

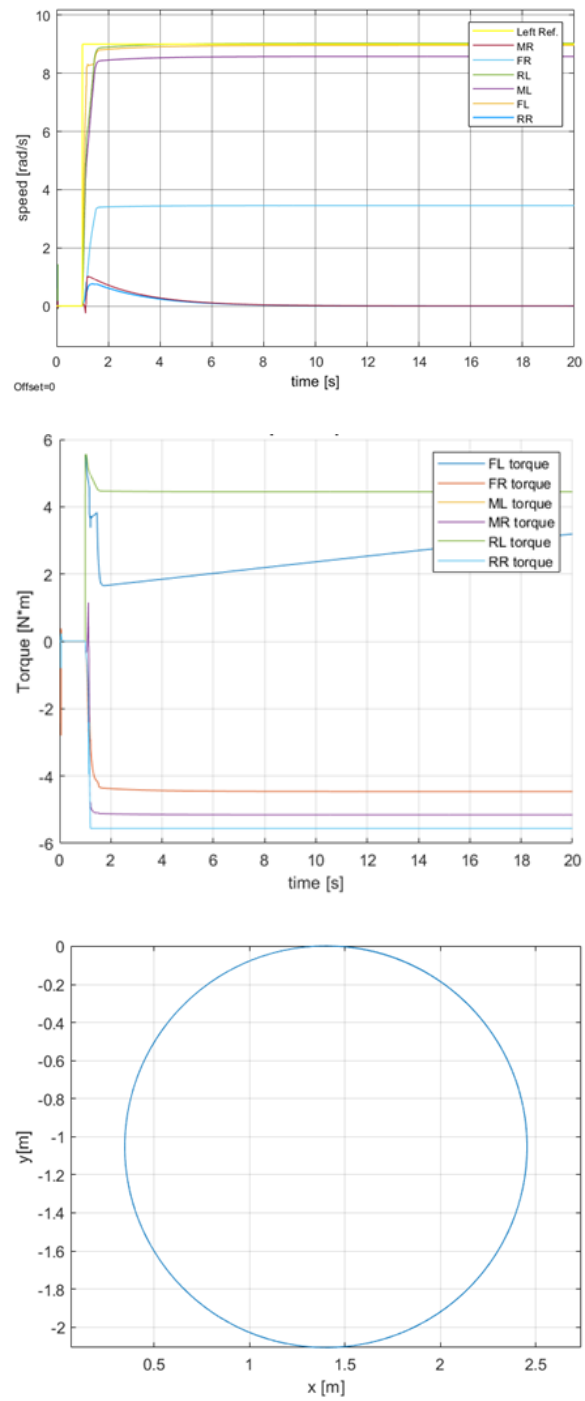


Figure 2.167: (a) Speed and (b) torque trends for the six motors; (c) trajectory curve for the hybrid configuration

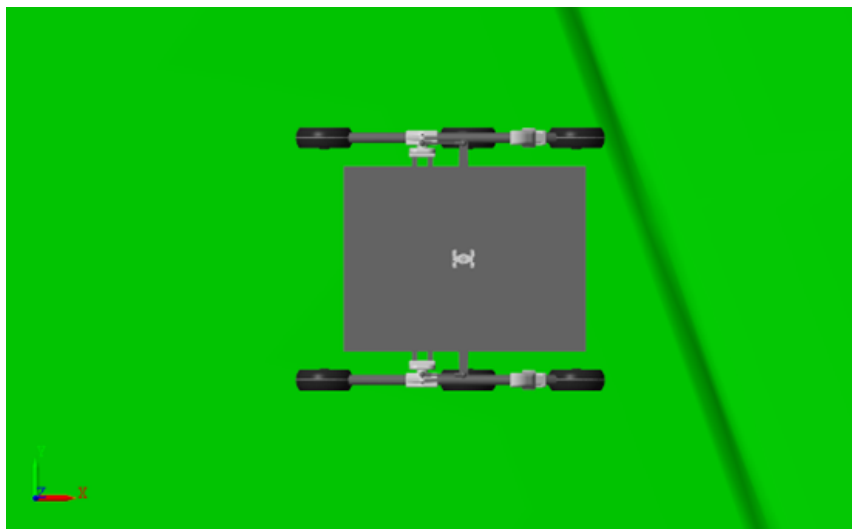


Figure 2.168: Hybrid configuration traveling a turned obstacle

2.4.2 Second Prototype Development and Analysis

2.4.2.1 Evolution of Rover Design

The previous chapter described the first model of a rover, the result of the collaboration between Green Tech Solution and the Department of Industrial Engineering of the Federico II University of Naples. Based on the developed model, a prototype was built in order to make some observations on the stability and functionality of the rover. Simulations were carried out using the Simscape/Simulink environment to identify and highlight some critical issues. These simulations provided fundamental information on the limitations of the initial model, allowing further investigation of aspects related to the dynamics and behaviour of the system under simulated operating conditions. This chapter will focus on highlighting the problems found and presenting the modifications studied and made to the rover, with the aim of obtaining a new optimized model.

Three macro-areas of intervention are identified: changes to the overall dimensions of the rover, changes to the core body (defined as the assembly of the plate and differential system), and changes made to the suspension system and wheel support.

Changing the overall size of the rover

The rover project, developed in collaboration between the Department of Industrial Engineering (DII) and Green Tech Solution, consists of four main topics: robotic arm, rover, optical sensing systems and autopilot. The first model has been developed using design parameters provided by Green Tech Solution, which include:

- Maximum rover speed: $\approx 1.38\text{m/s}$
- Accessible distance between working lines: $\approx 1\text{m}$
- Maximum slope: $30^\circ - 40^\circ$
- Required autonomy: 40 min

This section focused on the development of the rover, including an assessment of the impact of the robotic arm on the overall design. In the developed model, the arm was represented as a parallelepiped positioned in the center of the plate. The assigned dimensions (0.2 m x 0.2 m x 0.8 m) and mass (equal to 23 kg) match those of the JAKA Zu 5s model currently available from the department (Fig. 2.169).

With overall dimensions of 1220 mm x 1206 mm x 418 mm, the initial model developed to meet the design requirements consisted of: a welded aluminum plate, six drive wheels, a rocker-bogie suspension system, and a differential system designed to ensure stable plate position during operation. As mentioned earlier, experimental tests were conducted on the first prototype using the Simulink simulation environment. The results showed that the rover had difficulty crossing obstacles of considerable height. In particular, it was observed that due to the heavy weight of the wheels, the center of gravity of the structure was very low (Fig. 2.170). Although this aspect contributed to the improved stability of the system, it also led to significant limitations in overcoming bumps.



Figure 2.169: JAKA Zu 5s (88)

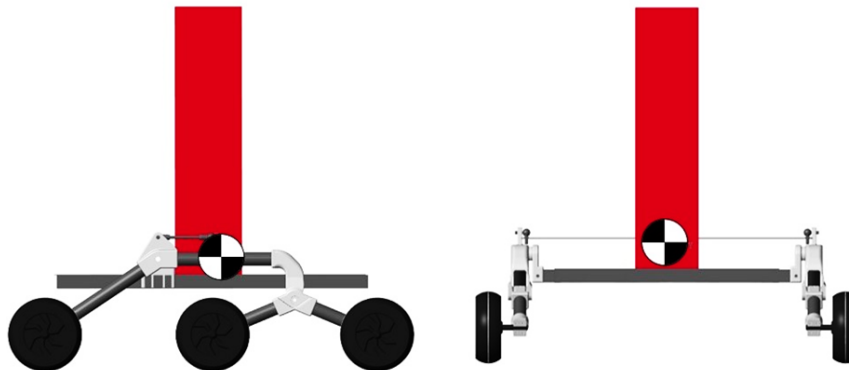


Figure 2.170: Center of gravity of the first model rover shown from side and front views

In the context of self-driving mobile robots, such as those used in field robotics, obstacle avoidance is a critical issue, as these systems typically operate on rough and uneven terrain. In order to overcome obstacles, the use of more powerful motors had to be considered, a choice that would have made the management and control of the rover much more difficult. For this reason, in rover design, it has been crucial to act on the center of gravity to improve performance. This can be achieved by acting on several aspects of the design. A first modification, which does not require a complete redesign, is to adjust the length of the profiles and change the angles of the rocker-bogie suspension joints. However, it is important to note that simply increasing the length of the profiles could result in an excessively large footprint and wheelbase, compromising the compactness

of the rover. For this reason, a combined solution was adopted: in addition to using longer Bosch profiles, the angles of the rocker-bogie suspension joints were also modified. Specifically, the new model and its dimensions were defined by fixing the length of the suspension profiles and establishing a 90° angle between the rocker and the bogie.

In the sketch shown in Figure 2.171, you can see the parameters set and the changes made, highlighting the approach taken to optimize stability and obstacle clearance capabilities.

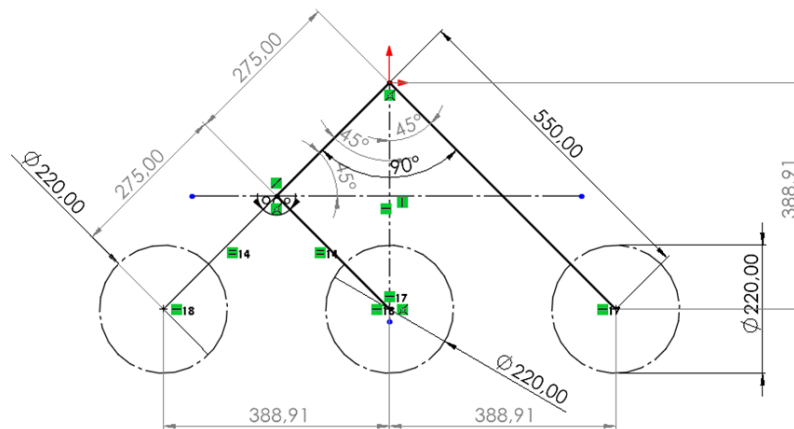


Figure 2.171: Sketch illustrating the new dimensions of the rover design

At the conclusion of the new dimensional study, the overall dimensions obtained for the model were 1000 mm x 892 mm x 474 mm, as shown in Fig. 2.172 and Fig. 2.173.

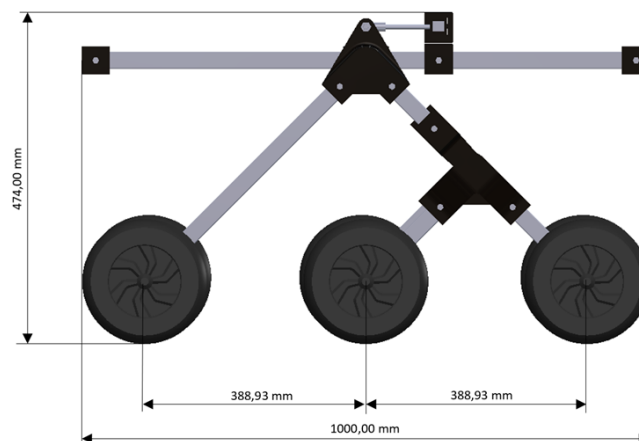


Figure 2.172: Side view of the new rover model with updated dimensions

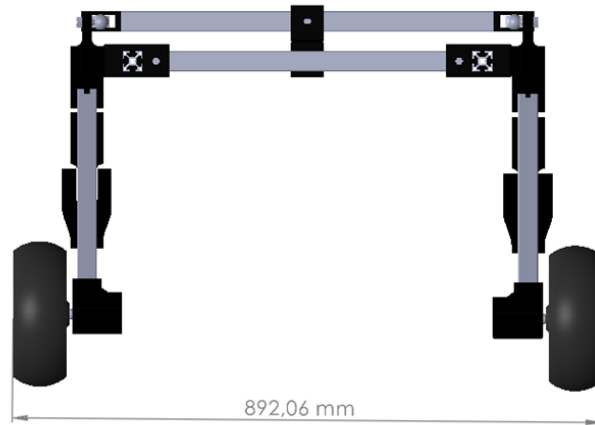


Figure 2.173: Front view of the new rover model with updated dimensions

The modifications resulted in a rover that was smaller in width and length than the original design, but larger in overall height.

The new structure, including the arm, will have a center of gravity positioned as shown in Fig. 2.174.

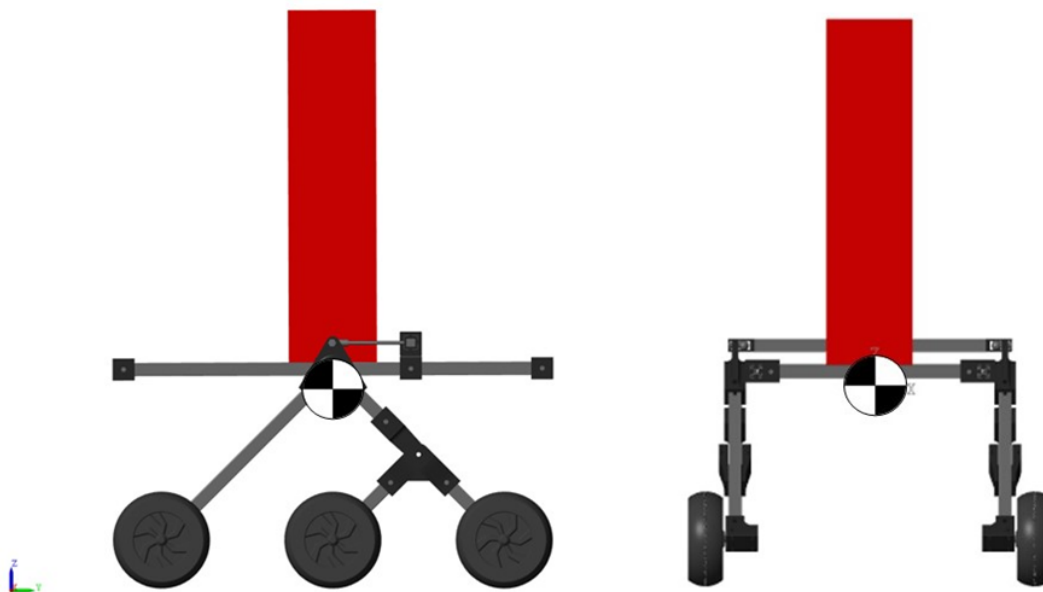


Figure 2.174: Center of gravity of the new model rover shown from side and front views

A comparison of the current model of the rover with its predecessor revealed an increase in the center of gravity of 0.12 m (Fig. 2.175). Given the limited dimensions of the

prototype, this outcome is notable, particularly in view of the minimal changes required to implement the requisite modifications.

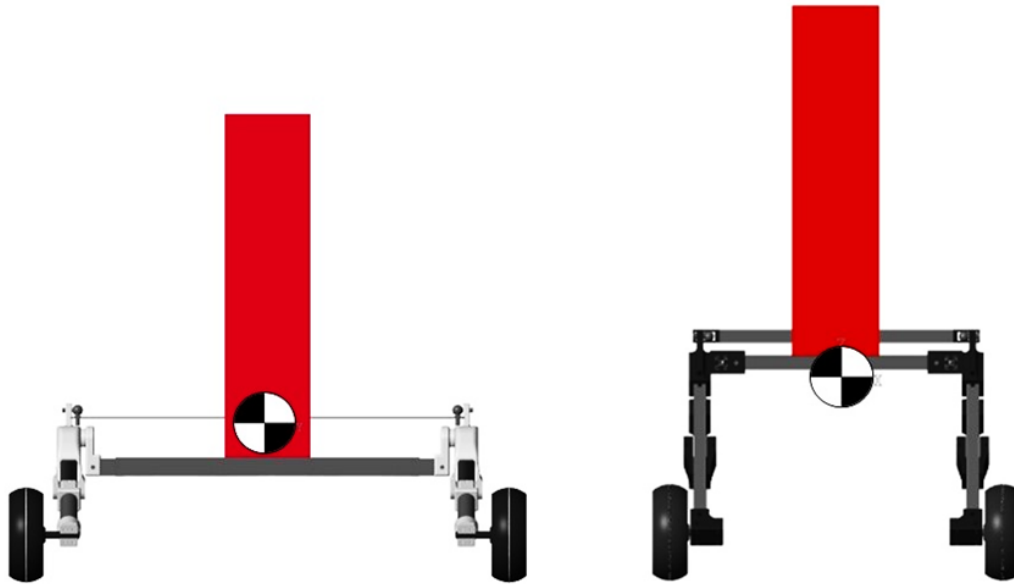


Figure 2.175: Comparison of the center of gravity positions between the two rover models

Changing the central body of the rover

In the initial phase of the project, the central body of the rover was designed as a fully welded aluminum plate (Fig. 2.176), with the purpose of enabling connection to the suspension system.

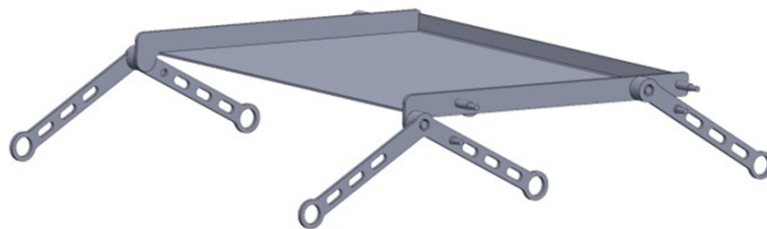


Figure 2.176: First model of the rover frame design

Accordingly, the frame was conceptualized as a singular plate, to which aluminum supports were to be welded for attachment to the rocker-bogie system. However, welding aluminum presents a number of significant complications. In particular, aluminum has a relatively low melting point of 660°C , but its exposure to oxygen causes the formation of an oxide layer (corundum), which has a much higher melting point of 2054°C . This

characteristic makes the welding process inherently complex, as the underlying metal tends to melt above a temperature range of 700-800 °C, forming a portion of molten aluminum covered by a film of corundum. The failure of the oxide layer can result in its mixing with the molten aluminum, leading to the formation of inclusions within the weld bead. Such inclusions can, in turn, give rise to cracks, thereby compromising the quality and strength of the weld joint. To circumvent these complications, the Friction Stir Welding (FSW) process is the preferred methodology, as it avoids the issues associated with the presence of the oxide and results in high-quality joints. While traditional aluminum welding is technically feasible under certain conditions, FSW represents a more reliable and less problematic solution for this particular application.

In consideration of these factors, the central body was redesigned with a different shape (Fig. 2.177) to facilitate an alternative connection with the suspension system. Specifically, additional appendages were constructed for the purpose of attaching the GAR (Joint to Rocker) joint, which was manufactured using screws and bolts. Moreover, a platform was constructed to cover the plate, designed to provide support for the mounting of the robotic arm (Fig. 2.177). This solution not only simplified the assembly process but also enhanced structural versatility, facilitating more effective integration of the functional components of the rover.

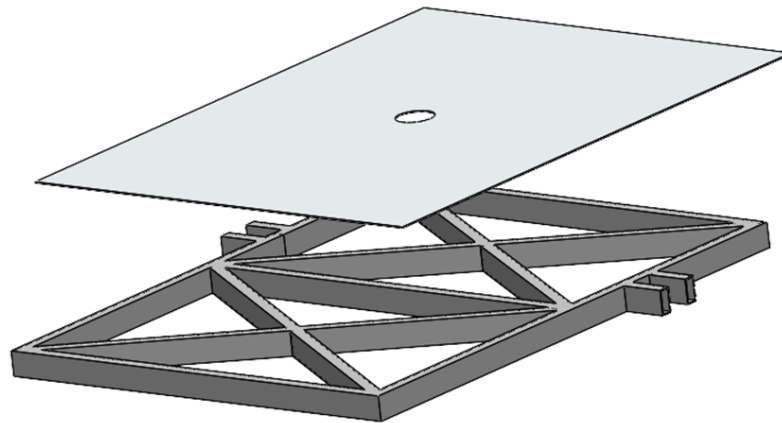


Figure 2.177: Plate constructed using triangular aluminum corner pieces to enhance stability and structural integrity

Due to the difficulties of welding aluminium, it was decided to use mechanical joints for the central body. A solution based on the use of Bosch profiles (Fig. 2.178), which offer greater modularity and ease of assembly, was also adopted for the frame structure.

Bosch profiles have many advantages: their hollow shape helps to streamline and lighten the structure, they are easy to assemble and simplify the prototyping process. A concrete example of their contribution is the reduction in the overall weight of the rover. The previous model, excluding the robotic arm, had a total weight of 87.2kg, while the new model weighs 64.7kg. This reduction represents a significant improvement in

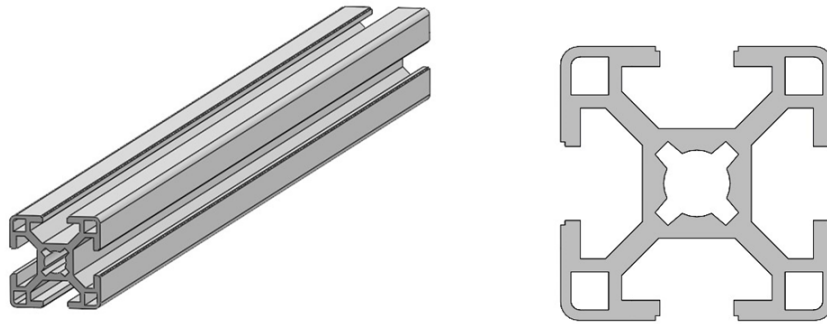


Figure 2.178: Isometric and side view of the Bosch

structural efficiency and ease of handling. In addition, the profiles make the central body modular, allowing it to be composed of units that can be added, removed or modified without affecting the rest of the system. For example, the modularity offered by Bosch profiles allows the frame to be made narrower or longer, depending on the specific needs of the experiments to be performed, thus providing greater design flexibility. The first solution adopted for connecting the profiles was the use of triangular aluminium corner pieces (Fig.2.179). These elements, characterized by their triangular shape, are commonly used in the design of frames and structures to improve stability and strength. However, the use of angles can present some critical issues, such as alignment problems and uneven load distribution, which could compromise the overall effectiveness of the structure.

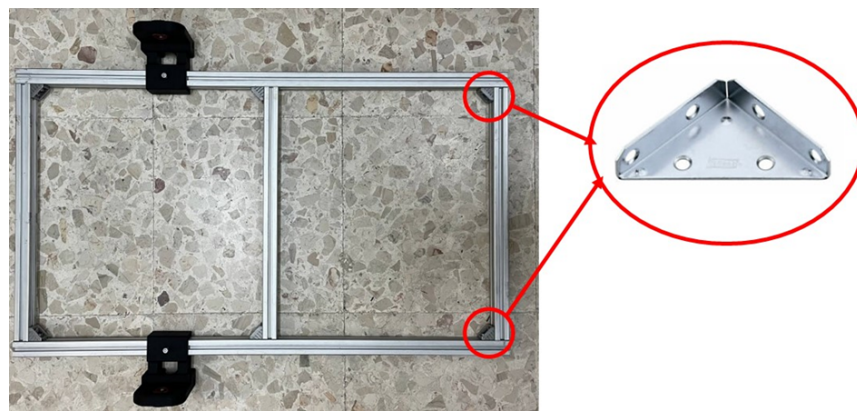


Figure 2.179: Plate made with the use of triangular angles

Some play was noted between the profiles during assembly. However, this movement was considered excessive for the model in question, especially in view of applications on rough terrain, where a particularly stable and strong structure is required. An alternative solution was developed to overcome this problem. The new strategy involves the use of angle brackets, also known as collars, specially designed and manufactured using 3D printing (Fig. 2.180). This solution makes it possible to significantly reduce the gaps between the profiles, ensuring a stronger assembly and greater structural reliability of the

frame. The collar is designed with a continuous cavity that mimics the shape of the Bosch

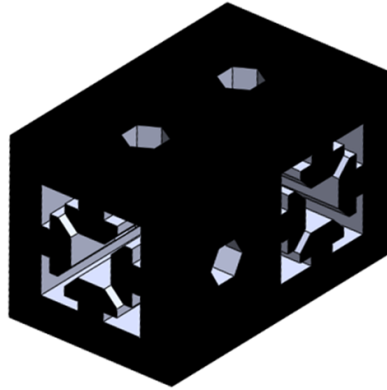


Figure 2.180: Connector angle enabling the assembly of Bosch profiles for the central body plate

profile and has additional holes for tightening. The profile is inserted into the cavity of the collar and secured later. Nuts and screws are used for tightening: the nuts fit into the Bosch profile rails, while the collar acts as a nut. During tightening, the screws are screwed into the collar, pulling the nut against the profile wings that form the main base of the trapezoidal rail. This process creates friction as soon as contact is made, preventing any relative sliding between the components. An additional advantage of the angle design is its versatility: it can be used for both "L" and "T" connections (Fig. 2.181), making it particularly adaptable to different structural configurations.

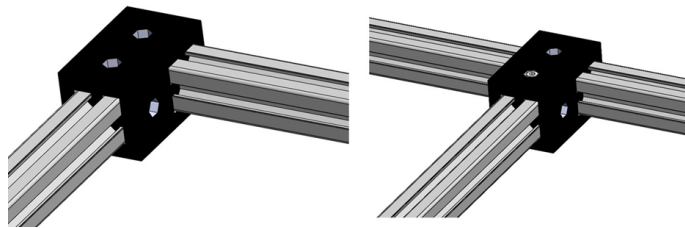


Figure 2.181: Angles used for "L" and "T" connections in the assembly of Bosch profiles

The central differential mechanism has also been adapted to incorporate this change: the differential bar has been replaced by a Bosch profile. This consists of two crossbars: one fixed, anchored to the central body, and one mobile, connected to the rocker system inherited from the previous design. The relative movement between the two crossbars is provided by a rotating joint. Essentially, this modification leaves the operating principle of the differential mechanism unchanged, while optimizing its integration into the new structure. Taking into account all the changes made, the central plate of the rover has been completely redesigned, becoming lighter and more flexible. The following figures (Fig. 2.182 and Fig. 2.183) show the original version, characterized by a welded plate, and

the current version, consisting of Bosch profiles joined by ABS (acrylonitrile butadiene styrene) corner pieces.

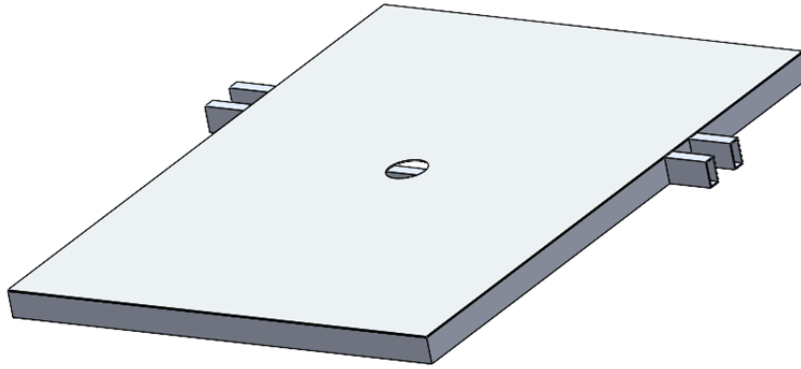


Figure 2.182: Plate-suspension system connection using L-joints

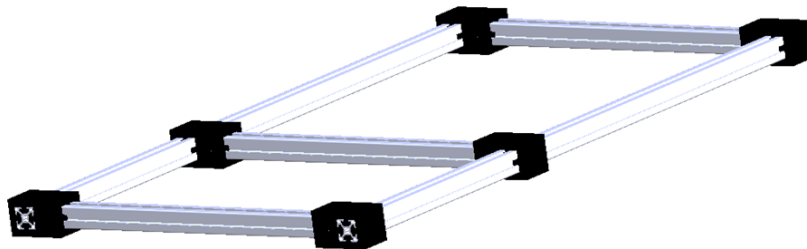


Figure 2.183: Connection with L-joints rotated 180 degrees

Changing the joints connecting the central body to the suspension system

The first rover model, described in the previous chapter, had a hinge that connected the central plate to the suspension system via a system of bolts. Initially, these joints, with their characteristic 'L' shape (Fig. 2.184), were mounted so that the revolute joint, responsible for the rotation between the plate and the suspension, was above the plane of the plate.

At a later stage in the design, the joint was rotated 180 degrees (Fig. 2.185), positioning the revolute joint below the plane of the plate. This configuration was chosen with the aim of raising the centre of gravity and consequently increasing the overall height of the rover.

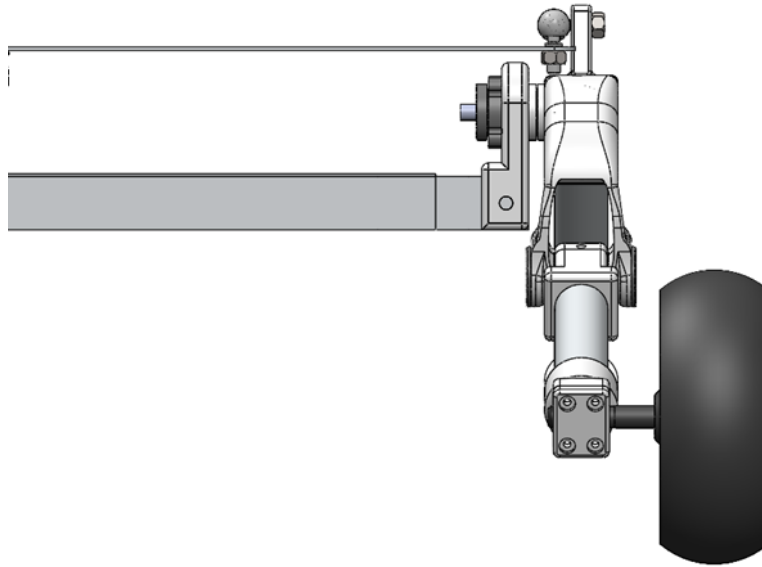


Figure 2.184: Plate-suspension system connection via L-joints

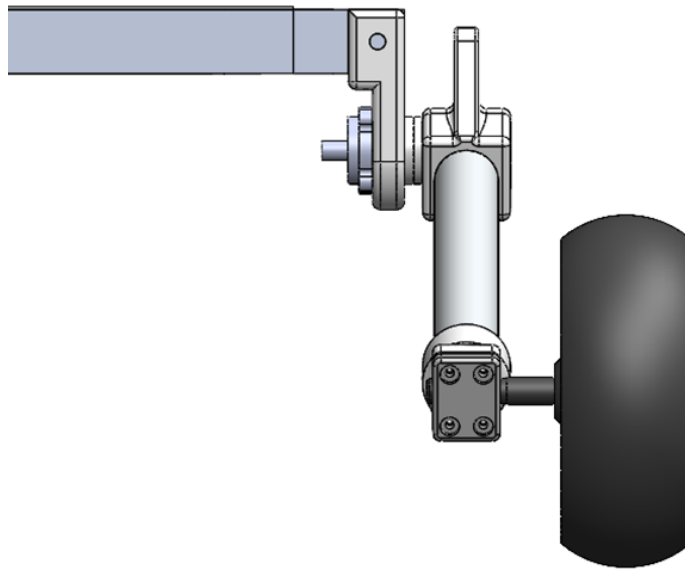


Figure 2.185: Connection with 'L' joints rotated 180

However, these modifications were not sufficient to raise the centre of gravity enough. In addition, the height difference between the rotation joint and the plate caused problems with uncontrolled rotation under certain operating conditions. In order to overcome these problems, a solution was developed to position the pivot at the same height as the plate. To meet this requirement and adapt the joint for use with Bosch profiles, the 'L' joint was replaced with a 3D printed collar (Fig. 2.186). This component has similar characteristics to the collars used to clamp the Bosch profiles that make up the central body of the rover, ensuring uniformity and structural compatibility. As with the angle brackets, the collar

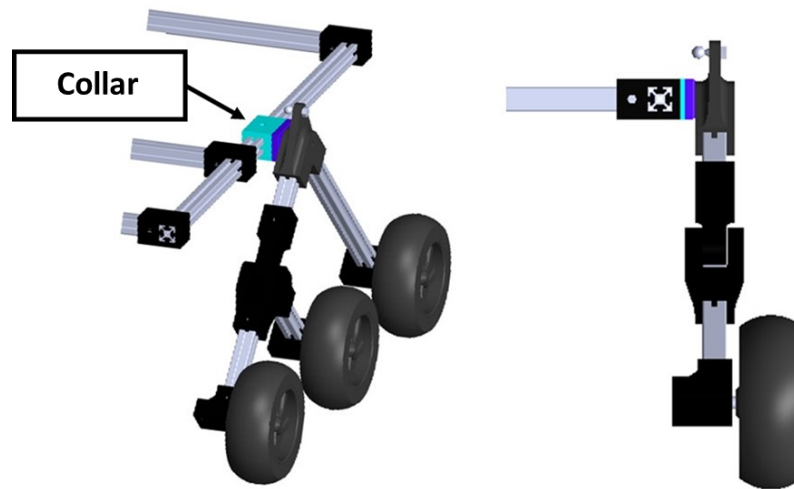


Figure 2.186: Plate-suspension system connection using a collar

is screwed and bolted to the Bosch profile. The collar is then fitted with a cap, which is also screwed and bolted to the structure (Fig. 2.187). The collar is also connected to the GAR joint via a bearing system which allows the rotational movement required for the mechanism to function properly. Fig. 2.188 shows how the connection between the plate

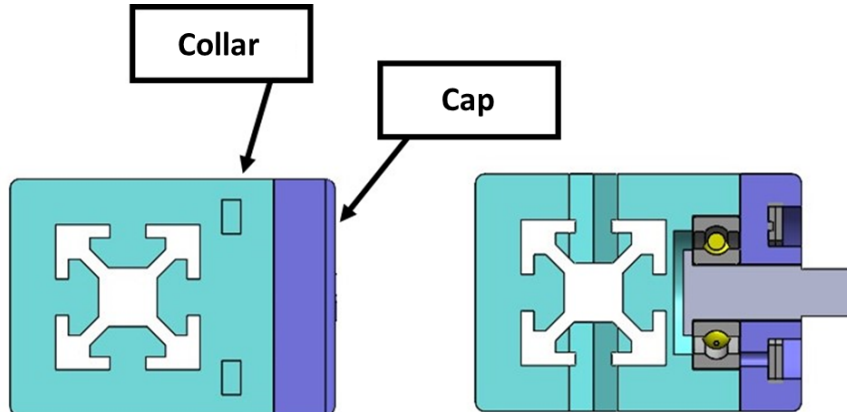


Figure 2.187: Collar and cap used for connecting the panel to the suspension system

and the suspension system has changed.

Changing the shaft between the GAR and the central body collar

In the first rover model, the shaft that connected the GAR joint to the 'L' joint, which connected the suspension to the plate, had a circular cross section (Fig. 2.189). This variation in cross section allowed for the locking of the bearings, a fundamental element in allowing the system to rotate. In the latest rover model, the modification of the GAR

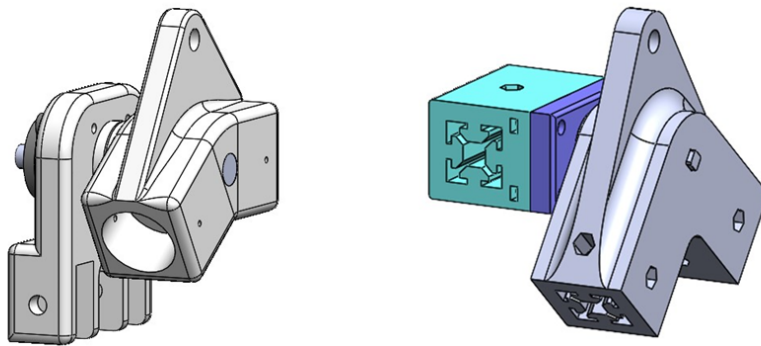


Figure 2.188: Comparison of the old and new connections between the central body plate and the suspension system

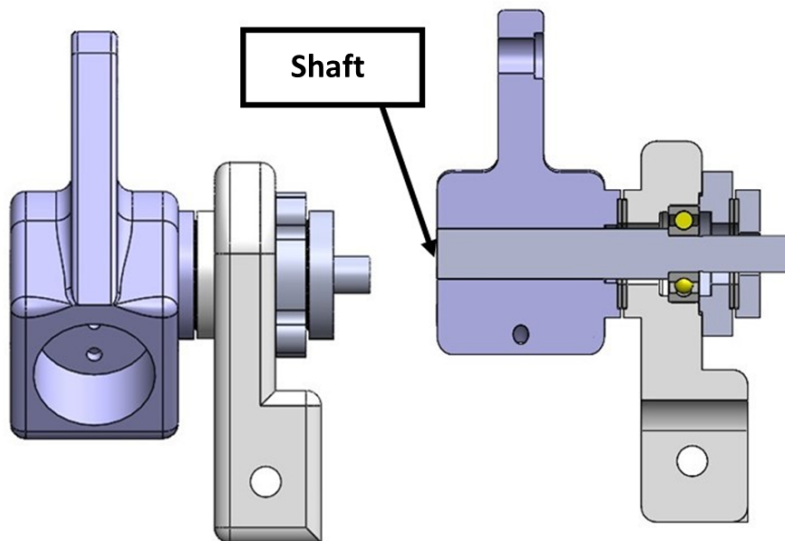


Figure 2.189: L-shaped GAR-joint with a sectional view

joint and the replacement of the 'L' joint by the collar has introduced a different shaft (Fig. 2.190). This new shaft retains a variable cross section to ensure correct assembly, but has one significant modification: the end section has been machined to a square cross section. This machining avoids any relative movement between the GAR and the shaft. In fact, since the GAR is made of plastic, the use of a shaft with a circular section coupled by interference would, after a few rotations, have led to deformation of the plastic and consequent failure of the system, compromising the solidity between the shaft and the

GAR. To avoid this problem, the GAR has been designed with an internal square recess that is compatible with the milled shaft end. This coupling ensures an interference fit, eliminating any relative movement between the two components and improving the robustness and reliability of the system. Despite this modification, movement of the

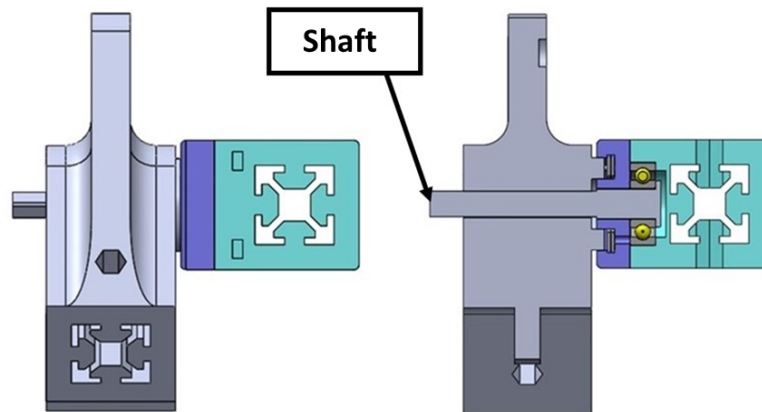


Figure 2.190: GAR-Collar joint with sectional view

bearing in the plastic housing in which it is mounted has been observed. This movement, increased by the weight of the wheels, can cause the suspension system to tilt (Fig. 2.191), with the risk of compromising structural stability and leading to breakage of the ABS elements. To prevent unwanted movement, it was decided to improve the connection

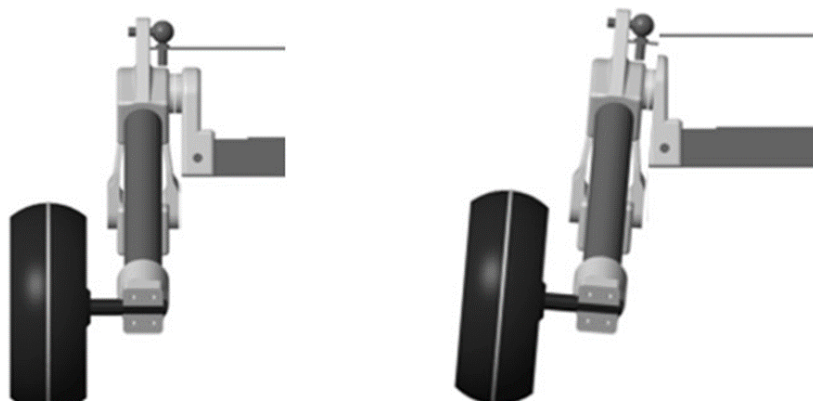


Figure 2.191: Tilting of the suspension system induced by shaft movement

between the collar cap and the GAR joint. The main problem was the movement of the bearing inside the collar caused by the shaft. It was therefore necessary to reduce the movement of the shaft. To this end, a thicker plug was designed with a circular crown recess (Fig. 2.192), commonly referred to as a 'cup', to accommodate the bearing washers. When the shaft bends, the washers come into contact with the socket coupling, which

resists the deformation and helps to limit the movement. However, the hollow cylinder alone proved insufficient to completely eliminate the problem. Therefore, an external solid cylinder was added to the GAR joint (Fig. 2.192)), designed to allow an interference fit with the plug, thus ensuring a solid connection between the collar, the plate and the GAR joint. This solution effectively prevents tilting of the suspension system due to its connection to the central body of the rover, thereby improving the overall stability and robustness of the system.

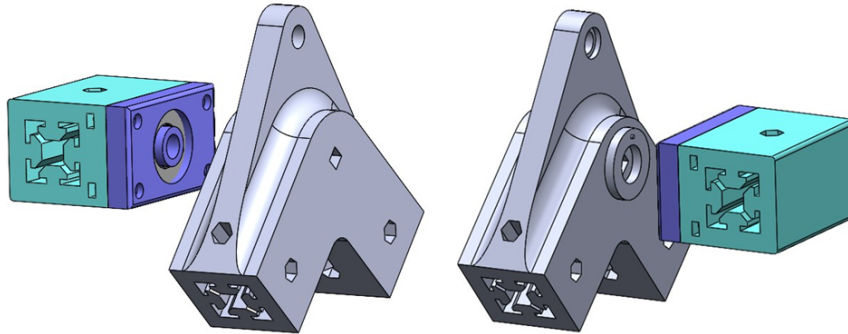


Figure 2.192: GAR-collar cap linkage

Spherical Joint substitution

The modified component consists of two ball joints at the ends connected by a threaded rod (Fig. 2.193) and is the link between the GAR and the differential rod.

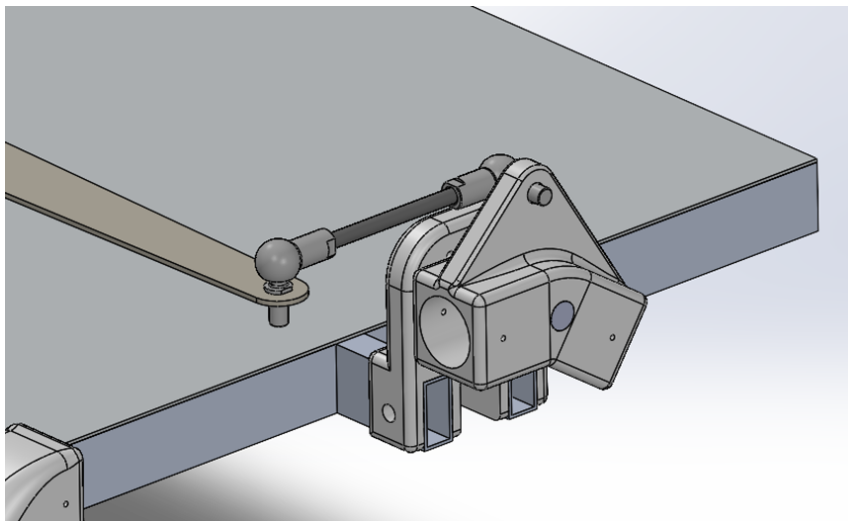


Figure 2.193: Old spherical joint used in the suspension system connection

The purpose of the differential system is to ensure that the central body plate remains balanced when the rover is stationary, and moves solidly with the system as it moves,

without ever coming into contact with the ground. However, the introduction of this system creates an imbalance in the weight of the model, a problem that is increased by the installation of the robotic arm. The differential bar takes this into account, considering that the rover will be operating on rough and not perfectly smooth terrain. One of the key components of the differential mechanism is the ball joint, which has three degrees of freedom corresponding to the rotations around the three main axes. These joints connect the differential rod to the two banks, allowing movement with a total of six degrees of freedom. However, a significant problem appeared during the assembly of the differential system: although the components are rigid, the connection between them generates relative movements. As a result of these movements, the first rover model experienced uncontrolled rotation of the central plate, which eventually fell to the ground. In addition, the turret on which the differential rod is mounted has a rotation of 1-2 degrees relative to the crossbar on which it is mounted. Although this is not visible in the simulation environment, it became apparent during the construction of the first prototype. In order to better understand this situation, an additional simulation was carried out in the Simulink environment, introducing a revolute joint between the turret and the crossbar. Assigning the joint a rotation of 1-2 degrees, the occurrence of uncontrolled rotation of the centre plate was observed, even with the rover completely stationary. This problem highlights the need to confront reality: even if the components are rigid, there is inevitably play in the connections, which cannot be completely eliminated but must be managed through specific solutions.

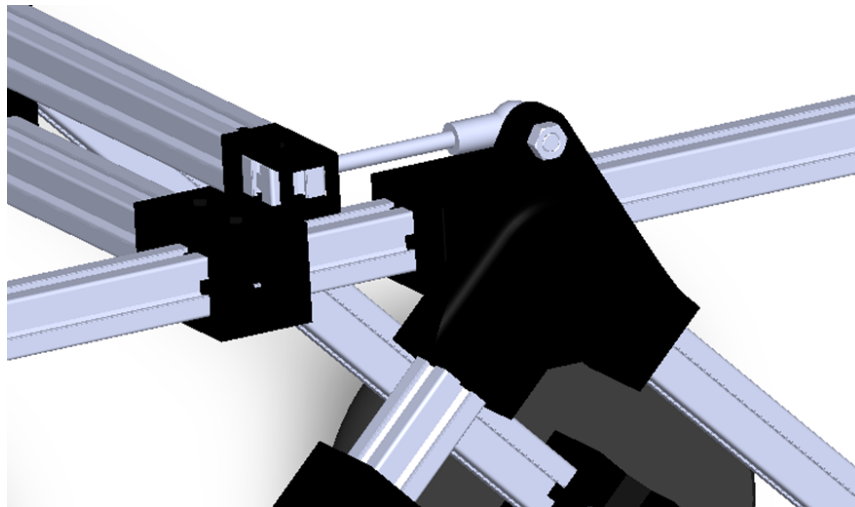


Figure 2.194: New joint used in the suspension system connection

To avoid this problem, it is possible to reduce the degrees of freedom of the system. By acting on the ball joint, it could be considered to eliminate two rotations and replace it with a rotating joint (Fig. 2.194). This modification would limit the relative movements between the components, improving the stability and control of the central plate of the rover. The revolute joint allows a single rotation around the vertical axis, perpendicular to the ground. Even in the most unfavourable conditions, such as when overcoming an

obstacle that causes the wheel to almost touch the plate, the rotation of the rod can reach a maximum of three degrees in both directions. This shows that the use of a revolute joint is an optimal solution. From a conceptual point of view, it would have been possible to replace both ball joints with revolute joints and still maintain the correct operation of the rover. However, it was decided to retain one ball joint to simplify assembly. The ball joint, with its ability to allow three rotations, offers more flexibility during assembly and facilitates the connection between the rocker and the threaded rod.

Turret height reduction

The term 'turret' (Fig. 2.195) is used to describe the component that connects the rod to the plate, enabling the movement of the latter via a revolute joint. It is clear that,

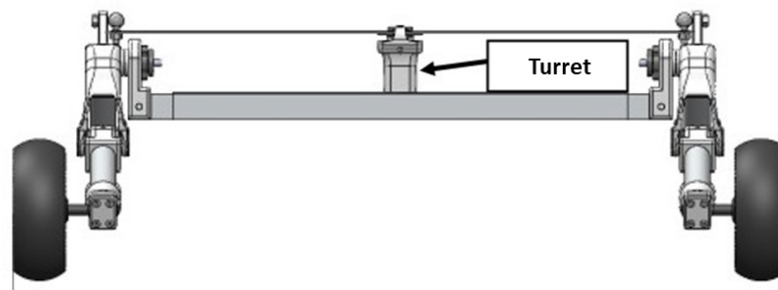


Figure 2.195: Frontal View of the first prototype with the 'turret' highlighted

in consideration of the modifications made to the plate, the turret was also modified, given that the latest model features a differential bar crafted with a Bosch profile. In addition to a system of pins and bearings that ensure the movement of the bar, a number of three-dimensionally printed components have been employed in its fabrication. The configuration resulting from this configuration is illustrated in Fig. 2.196.

In a static configuration, the plate would be subject to a tendency to rotate. However, this is counterbalanced by the threaded rods, which act as tie rods and maintain the plate in equilibrium. As a consequence of the imbalance of the plate, the action of gravity and the necessity to guarantee a state of equilibrium, forces are generated. The resulting forces are multiplied by the distance between the balance axis and the crossbar axis (or, alternatively, by the height of the turret), which results in a bending moment. This is dependent on the length of the arm and increases in proportion to this length, therefore the objective is to minimize this distance. In order to reduce the height of the turret, the initial proposal was to eliminate the turret entirely by incorporating it into the crossbar. The feasibility of this concept was established through the implementation of a crossbar incision. This solution presented a challenge: the crossbeam itself was prone to bending as a consequence of the incision. It was therefore necessary to accept the presence of a moment and to concentrate on minimising its effect. It is important to note that the assembly of the mechanism (which includes shaft, bearings, washers, etc.) requires a certain amount of space. It was thus decided that the minimum distance between the

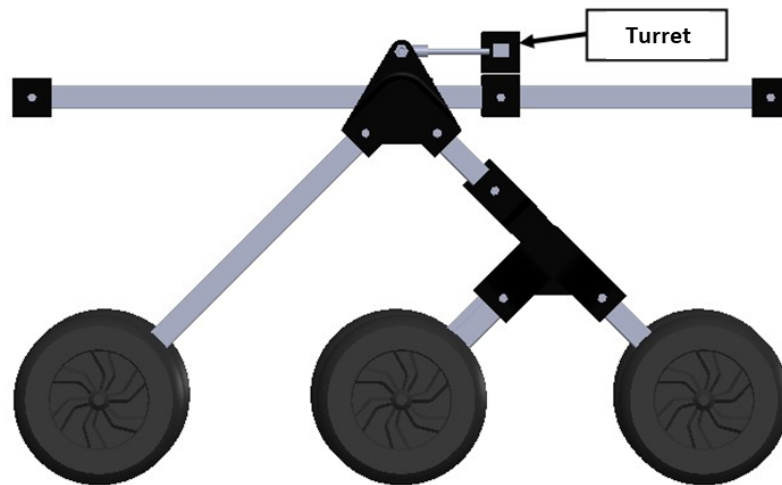


Figure 2.196: Side view of the latest rover prototype with the ‘turret’ highlighted

crossbar and the rod should be 3 cm, which corresponds precisely to the size of a Bosch profile.

So the balance collar and its joint are modelled in such a way that the distance between the crossbar and the balance is just the size of a Bosch profile. The use of this solution and the choice of this dimension made it possible to make the structure modular.

Comparison of the central body

To conclude this discussion on the evolution of the central body of the rover, the two solutions are presented for comparison. In the first rover model, the central body (Fig. 2.197) consisted of a welded plate, weighed down by the presence of a platform. Furthermore, the connection system between the central body and the suspension system was complex and not very functional. A further problem was the differential system, which made use of ball joints. The latter caused uncontrolled plate rotation, compromising the stability and proper functioning of the rover.

As a result of the modifications described, the central body (Fig. 2.198) is lighter. The use of Bosch profiles gives the rover greater flexibility and adaptability, making it easy to modify to suit different sizes and configurations. The differential system, thanks to the introduction of a revolute joint to replace the spherical joint, ensures that the rover operates without problems of uncontrolled plate rotation. Finally, the suspension linkage system, now consisting of a collar and GAR joint, is considerably more rigid, preventing unwanted tilting and improving the stability of the rover.

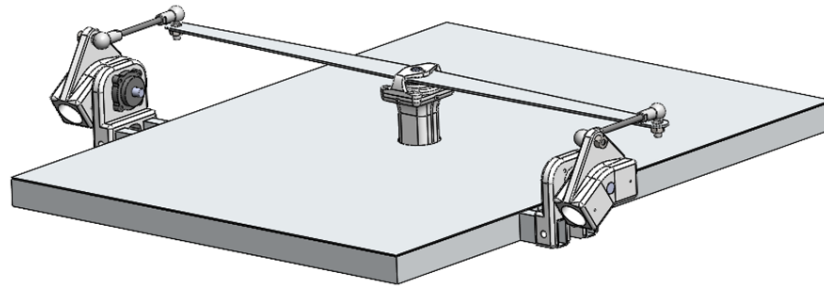


Figure 2.197: Central body of the first rover

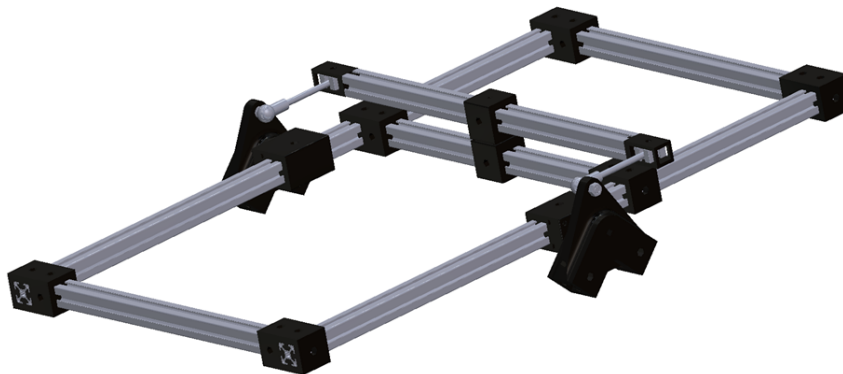


Figure 2.198: Central body of the latest rover

Rocker-Bogie modifications

One of the most significant changes between the new and old models was the replacement of the circular hollow sections (Fig. 2.199) used for the rigid connections between the rocker and bogie joints and the wheels with Bosch Rexroth sections (Fig. 2.199). The circular tubes presented some difficulties during assembly, as they required holes to be drilled to ensure alignment between the wheel joints and the GAR joint of the suspension system. These holes, combined with the use of a spring pin, ensured the correct connection between the tube and the wheel support, keeping the latter parallel to the ground. However, the spring pin did not guarantee perfect alignment and was subject to excessive stress, which could lead to failure over time. To overcome these problems, it was decided to use Bosch profiles. These eliminate the problems associated with assembly, thanks to their flat surfaces that simplify installation and ensure that the wheels are correctly aligned with the ground, while improving the robustness and reliability of the system. The GAR joints (Fig. 2.200) have also been adapted to this change: they have been

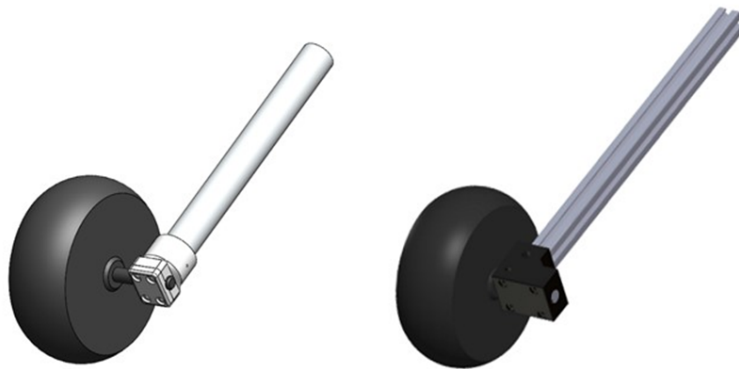


Figure 2.199: Comparison between the old and new links in the rocker-bogie suspension system

redesigned to fit the specific Bosch profile, making the whole assembly more compact and functional. A key feature of Bosch profiles is their ability to make the structure modular, i.e. composed of units that can be added, removed or modified without affecting the rest of the system. This modularity makes it easy to adapt the frame to the needs of the experiments to be carried out, making the whole structure more flexible and versatile.

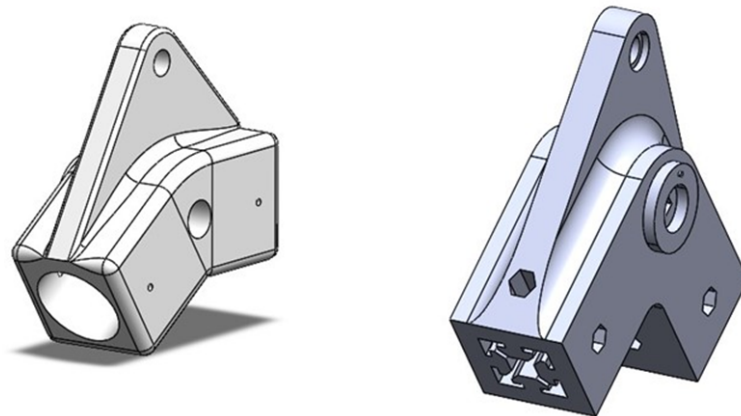


Figure 2.200: Comparison between the old and new GAR joint designs

Rocker angle modification

As well as being taller than its predecessors, the latest rover has undergone changes to some of its components. In particular, it was decided to modify the joint between the rocker and the plate. It was designed so that the GAR pivot was perfectly positioned on the same axis as the central wheel (Fig. 2.201). As shown in the Fig.2.201, the structure

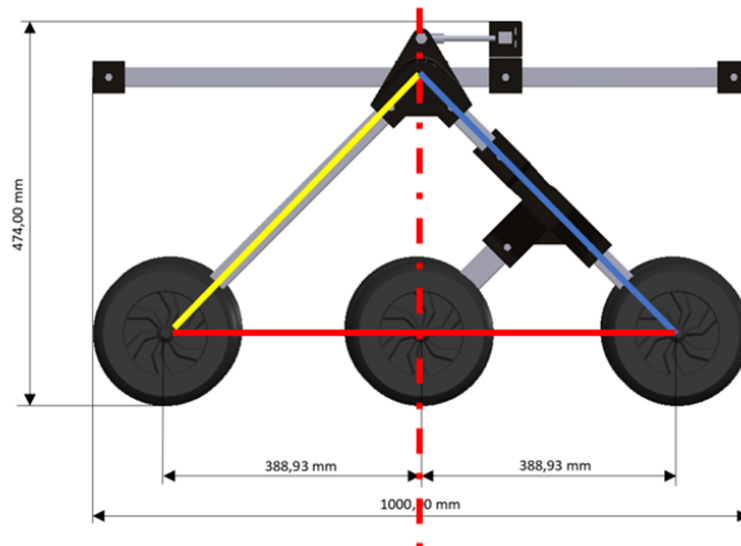


Figure 2.201: Alignment between the center wheel axis and the GAR joint axis

is designed so that the distance between the rear wheel and the centre of the GAR joint (shown in yellow in the diagram) is equal to the corresponding distance from the front wheel (shown in blue) and the angle between these elements is correct. The result is a triangular structure. By positioning the central wheel exactly on the centre line between the front and rear wheel axles, i.e. halfway along the wheelbase, the axis of the GAR revolute joint coincides with the axis of the central wheel, corresponding to the height of the hypothetical triangle described. This configuration ensures that the weight is evenly distributed over the three wheels of each wheelbase, and therefore over all six wheels of the rover. Uniform weight distribution ensures that the rover has greater ground contact, improved traction and a reduced risk of tipping, particularly when crossing obstacles or operating on uneven terrain.

Rocker-Bogie angle variation

The side view of the first rover model (Fig. 2.202) shows that the tube connecting the rocker to the bogie was horizontal to the ground. However, this configuration presented a significant bending problem. The resulting forces propagated along the horizontal axis of the rocker and were then transferred to the bogie joint. At this point, the force was distributed along the two directions of the tubes that formed the bogie. The critical aspect was the bogie joint, which was made of plastic and subjected to stresses from several directions. This situation could cause the joint to fail, compromising the stability and functionality of the system.

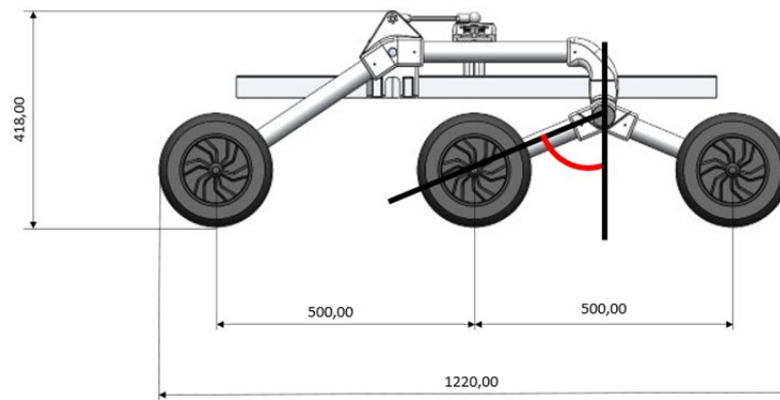


Figure 2.202: Rocker-Bogie angle in the first model

To avoid these problems, it was decided to align the tube connecting the rocker to the bogie with one of the tubes of the bogie itself (Fig. 2.203). This configuration ensures that the generated force component is collinear with the bogie arm, avoiding shearing and consequently the formation of a bending component.

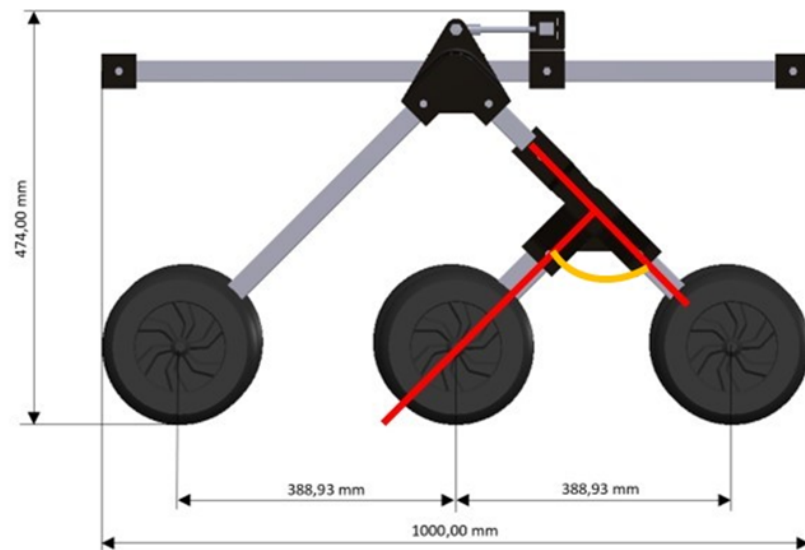


Figure 2.203: Rocker-Bogie angle in the new model

Wheel position evaluation

The first rover model was used to study the moments generated at the GAR joints of the two banks. Quite large bending moments were generated even when the rover was stationary (Fig. 2.204).

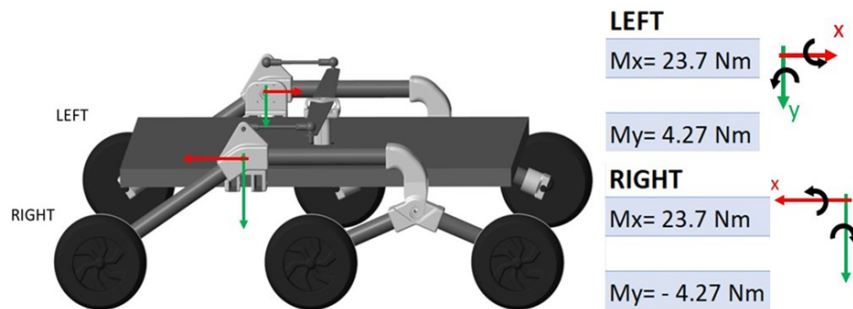


Figure 2.204: Torques at GAR of the two banks: external wheels version

This causes the rover to tilt (Fig. 2.205) as well as the possible failure of the components that make up the connection between the centre plate and the suspension.

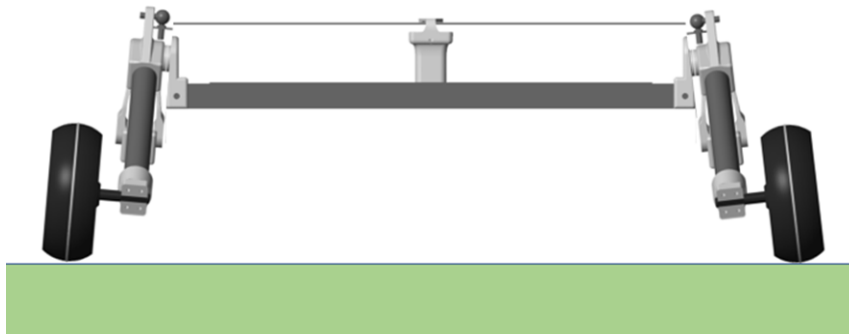


Figure 2.205: High torques experienced at the GAR during operation for the first rover

As a first solution, the position of the rocker wheels was reversed (Fig. 2.206): this modification showed that the moments were significantly reduced by more than half. The optimal configuration was, however, in which the centre wheels of the two banks were outside and the other wheels all inside (Fig. 2.207): the moment component was almost zero and the problems encountered previously no longer occurred. In the latest rover model, the torques on the joints were also analysed as the wheel position changed. The analysis showed that, with the correct design and modifications, a reduction in torque was achieved (Fig. 2.208). These improvements represent a significant advance over the previous design and will help to optimise the operation of the rover. In particular, the best solution in terms of torque is the version with the outer wheels on the inside and the middle wheels on the outside (Fig. 2.209).

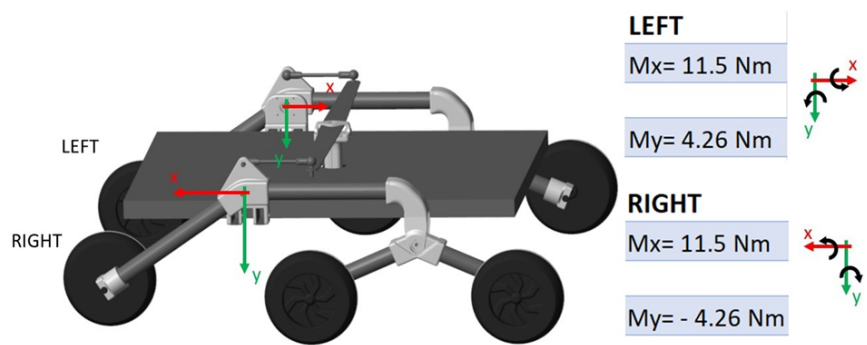


Figure 2.206: Torques at GAR of the two banks: rocker wheels internal version

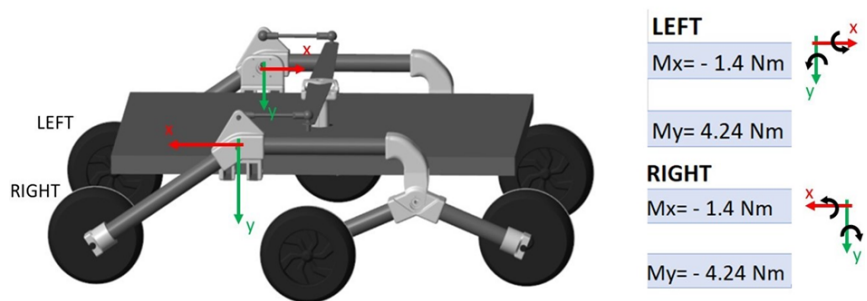


Figure 2.207: Torques at of the two banks: central wheels external version

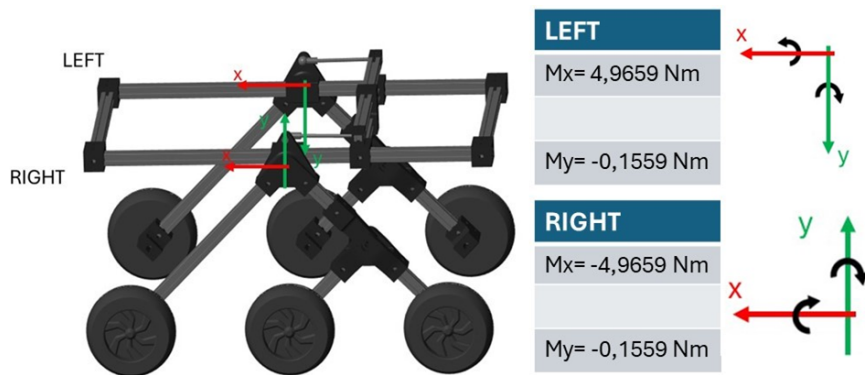


Figure 2.208: Torques at of the two banks: new rover model

The torques were also analyzed considering the presence of the robotic arm. The evaluations showed that, in the configuration in which all three wheels of the banks are positioned externally (Fig. 2.210), the torques are significantly higher than in the

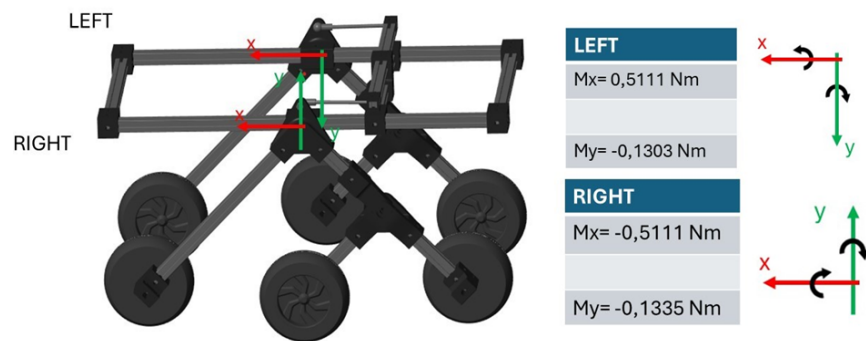


Figure 2.209: Torques at of the two banks: new rover model with external central wheels

configuration with only the central wheel in an external position (Fig. 2.211).

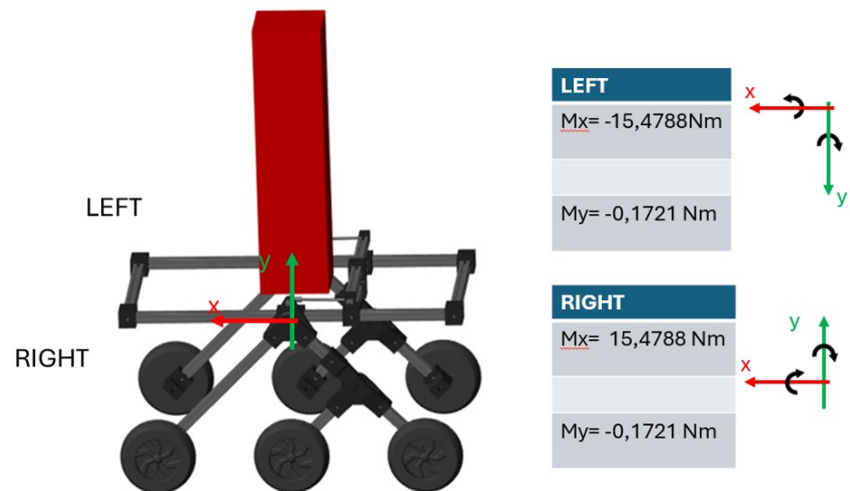


Figure 2.210: Torques at of the two banks: new rover model with robotic arm

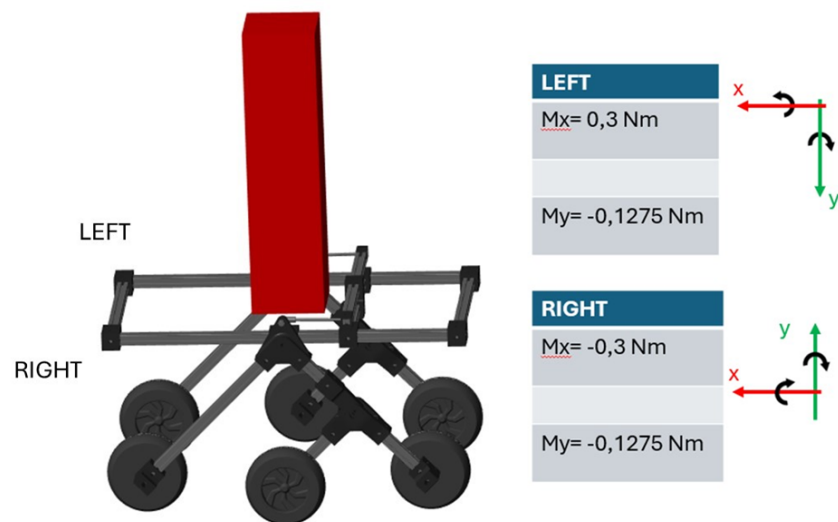


Figure 2.211: Torques at of the two banks: new rover model with robotic arm and central external wheels

Changing in the support connecting the wheels to the suspension

The motor-wheel subsystem is connected to the suspension system by means of a bracket called the GRS (wheel-suspension coupling), the shape of which has been modified over time to improve performance. The wheel shaft is fitted with a flange (Fig. 2.212) designed to ensure a secure connection to the GRS. This support holds the shaft firmly in place, ensuring that the wheel rotates while the shaft remains fixed. In the first rover

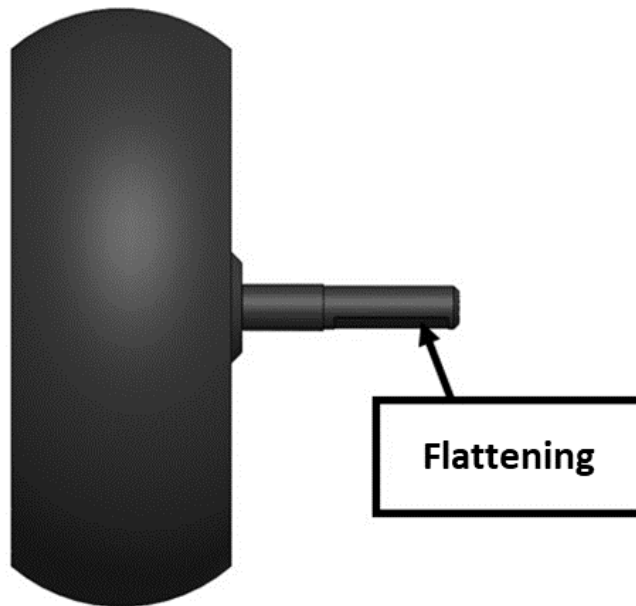


Figure 2.212: Motor-wheel

model, the upper GRS was inserted onto the shaft and the GRS cap, whose dimensions corresponded to those of the shaft flattening, was screwed onto it (Fig. 2.213). In the

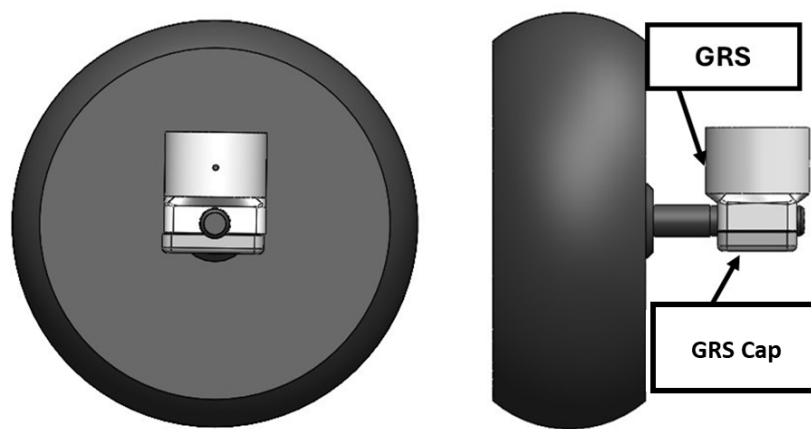


Figure 2.213: Front View and side view of the old support connecting the wheels to the suspension

latest rover model, the shape of the GRS has been modified (Fig. 2.214) to bring the wheel closer to the Bosch profile of the suspension. The aim of this modification was to reduce the bending moment generated during operation. By bringing the wheel closer to the beam, the arm, i.e. the distance between the beam and the wheel, is reduced and consequently the bending moment is minimized. This improves the stability and durability of the suspension system. A slot (Fig. 2.215) has been added to the lower GRS to fit and

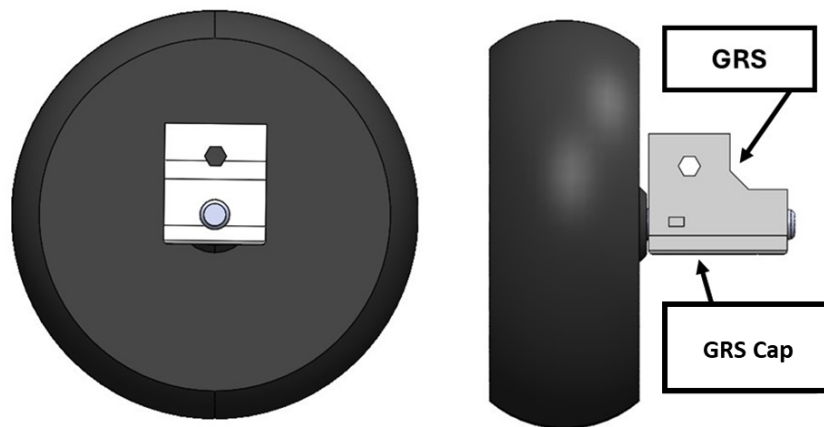


Figure 2.214: Front View and side view of the new support connecting the wheels to the suspension

clamp securely onto the wheel shaft. This modification ensures the proper functioning of the motor-wheel assembly and provides a stable and reliable connection during rover operation.

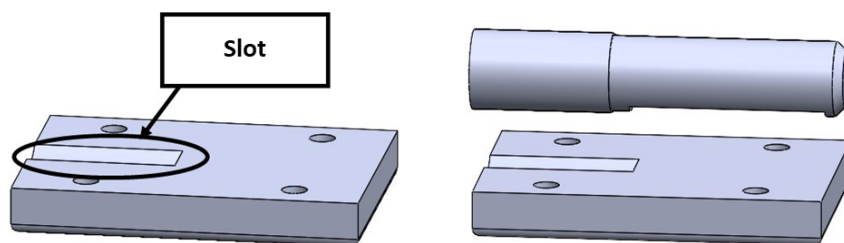


Figure 2.215: GRS Cap

2.4.2.2 Simulation Results

As with the first version of the rover, a multi-body model was developed for this new version, which is essential for analysing the dynamic behaviour of the system and evaluating its performance. Following the approach adopted for the first model, the second model was structured around four main elements: the structure of the rover, the wheel-ground interaction, the terrain model and the references. The structure includes the chassis, suspension system, Bosch profiles and motor-wheel system, while the wheel-ground interaction and terrain model allow different operating conditions to be simulated, including friction and obstacles. This section analyzes how the new version of the rover performs in some of the tests performed on the first version, highlighting the differences in results and the improvements made by the structural modifications. A direct comparison between the two models will highlight the progress made, focusing on stability, manoeuvrability and overall system efficiency.

Obstacle Crossing

In order to compare the two rover models, a special simulation was carried out in which an obstacle with a height equal to the diameter of the wheels was crossed. This condition represents a particularly severe scenario for the overall structure of the rover, testing both the suspension system and the overall balance. The choice of this simulation is motivated by the operational area of the rover, which involves operation on rough and uneven terrain. This test allows important observations to be made that will be useful in assessing the effectiveness of structural modifications and identifying any critical issues that need to be addressed.

Simulation Conditions

It was decided to compare the rover models in terms of their ability to negotiate an obstacle, specifically a step with a height equal to the diameter of each wheel. The obstacle was modelled using the "Grid Surface" block, with the surface set to a height of 220mm (Figure 2.216). This configuration allows an extreme condition to be simulated that is representative of the challenges the rover may face when operating in rough terrain. Using these specifications in simulation, the obtained ground is as in Fig. 2.217.

A speed law (Fig. 2.218) with a maximum value of 1 rad/s was imposed on each motor. The rover is positioned so that the wheels are parallel to the step (Fig. 2.219) and can therefore face the obstacle at the same time. In addition, the rover is brought as close to the step as possible to shorten the simulations and reduce the computational load. The aim of the simulations is to evaluate the torques of the six engines to understand how the changes made to the rover affect them. An improvement in torque is expected, with lower values for the same operating conditions, indicating an increase in overall system efficiency. As mentioned above, the weight of the rover has been significantly reduced from 87 kg to 64 kg, mainly due to the removal of the plate present in the first model, which alone contributed about 25 kg to the total weight. In order to adequately analyze the dynamic behaviour and assess the effect of the centre of gravity shift on the engine

Block Parameters: Scalino 220m		
Grid Surface		
Settings	Description	
NAME	VALUE	
X Grid Vector	[-5 -0.6 0.6 5]	m
Y Grid Vector	[-20 0 2 2.05 2.1 2.15 2.2 6]	m
Z Heights	[0 0 0 0.1 0.20 0.21 0.22 0.22; 0 0 0 0.1 0.20 0.21 0.22 0.22...	m
Graphic		
Type	From Geometry	
> Visual Properties	Simple	

Figure 2.216: "Grid Surface" specifications block

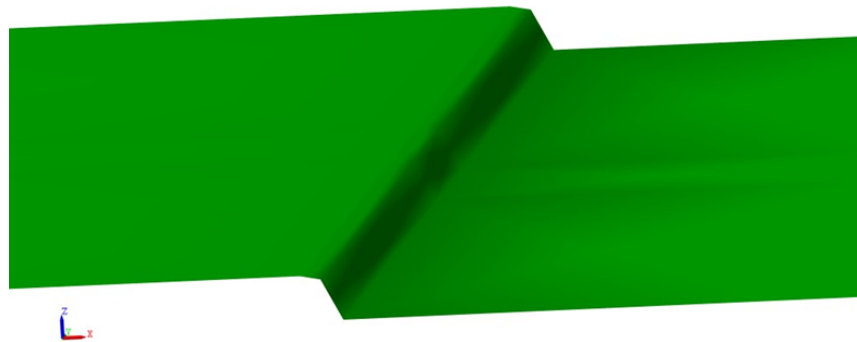


Figure 2.217: Ground in Simulation

torques, it was decided to run simulations while keeping the weight of the rover constant. To achieve this, the plate of the first model was modified and its weight reduced so that both rovers weighed approximately 64 kg. In this way, torque differences associated with significant weight variations can be eliminated, ensuring that the results reflect only the effects of the design changes made.

Torque trend of the six motors

Torque trends for the front, centre and rear wheels were plotted in Fig.2.220, showing that they behave identically in pairs. This observation immediately confirms that the rover is correctly aligned to the step and that the structure is symmetrical as intended by the design. The pair plots for each rover model are shown first. A direct comparison is then made between the two models. To avoid visual overload in the graphs during the comparison, only two curves are shown, one for each pair of wheels, to make the data clearer and more readable. A first aspect checked during the analysis is that the input to the motors is provided exactly at the time $t=2s$, in accordance with the set speed law. Although this observation is not repeated for all pairs of wheels, it is considered to be generally valid for the whole system. Next, the front, centre and rear wheel torque trends are shown.

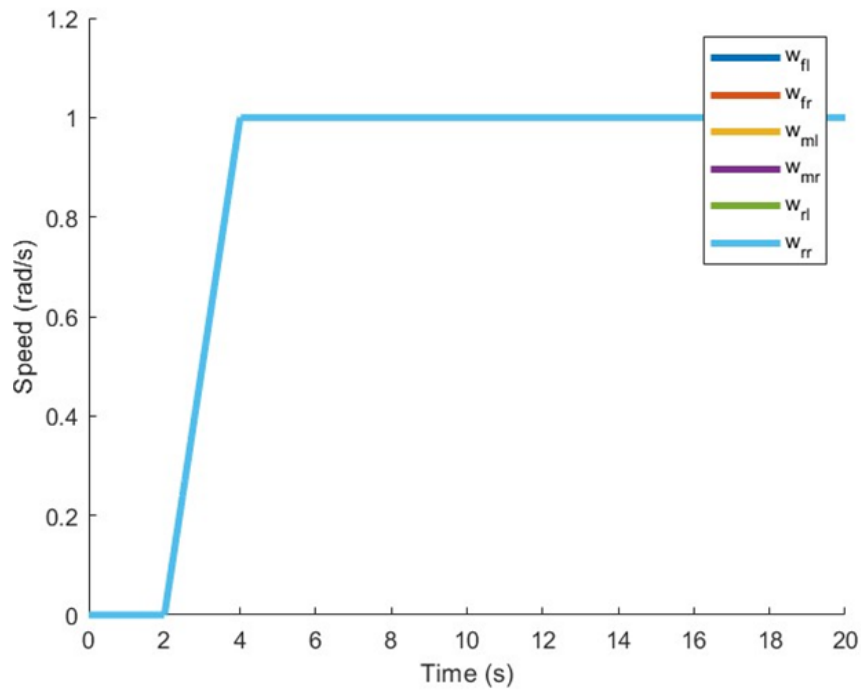


Figure 2.218: Imposed speed Law for each motor

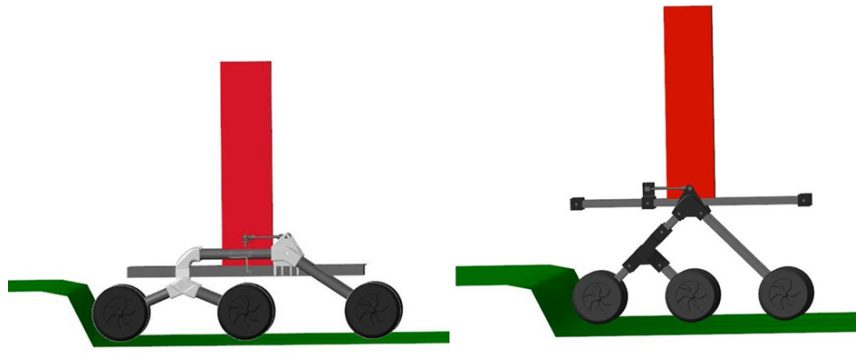


Figure 2.219: Starting position of the rovers

Front Wheels

In Fig.2.221, there is the front wheels torque for the first rover prototype. It is possible to observe a peak in the torques at time $t=2.4s$, caused by the contact of the front wheels with the kerb (Fig. 2.222). During the phase of crossing the obstacle, the torques show a decreasing trend. Then, at time $t=5.8s$, there is a maximum torque value of $6Nm$, just when the front wheels reach the edge (the highest part) of the step. After this moment, the trend becomes almost constant at times until time $t=8.8s$, when another peak occurs due to the contact of the central wheels with the step. Slip is then observed, associated

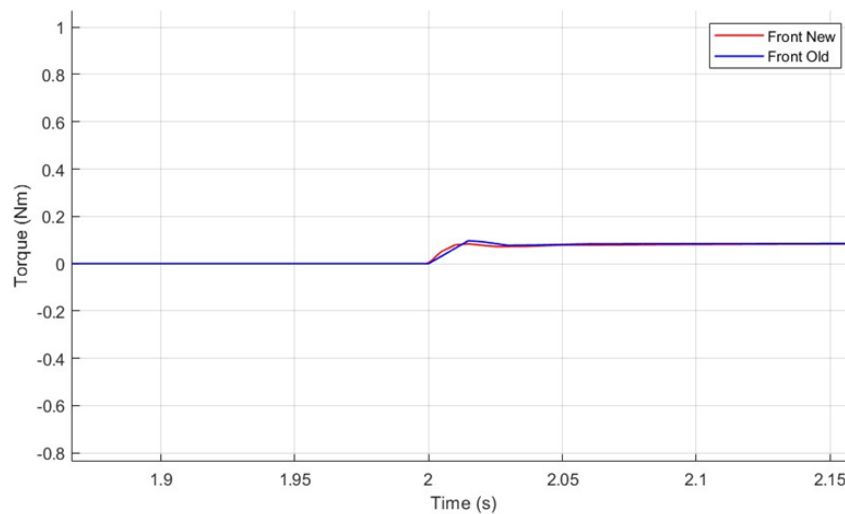


Figure 2.220: Motor impulse response at $t = 2$ s comparing the new and old front motors

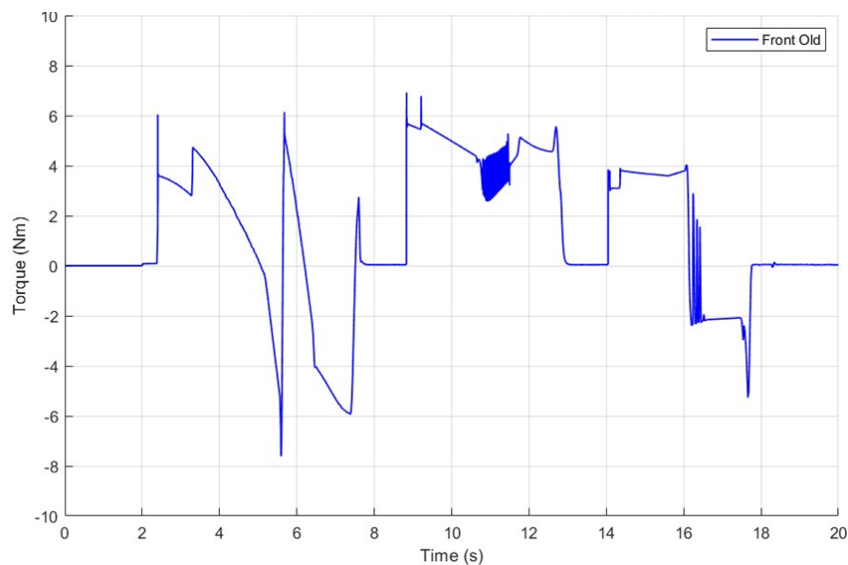


Figure 2.221: Torque profile of the front wheels in the first model of the rover

with the passage of the central wheels over the edge of the step. This is followed by an interval in which the torques remain constant and close to zero, during which the front wheels travel along a flat section and no other wheels are involved in overcoming the obstacle. Another peak occurs at $t=14$ s when the rear wheel makes contact with the obstacle. During this phase, the torque trend is constant until slip occurs due to the rear wheel passing over the edge of the step. Once the rear wheel has passed the obstacle, the contribution of the front wheels to the manoeuvre becomes minimal. The obstacle is passed at $t=16$ s, at which point the torque returns to zero.

A comparison of the torques for both models (Fig. 2.223) reveals a notable discrepancy

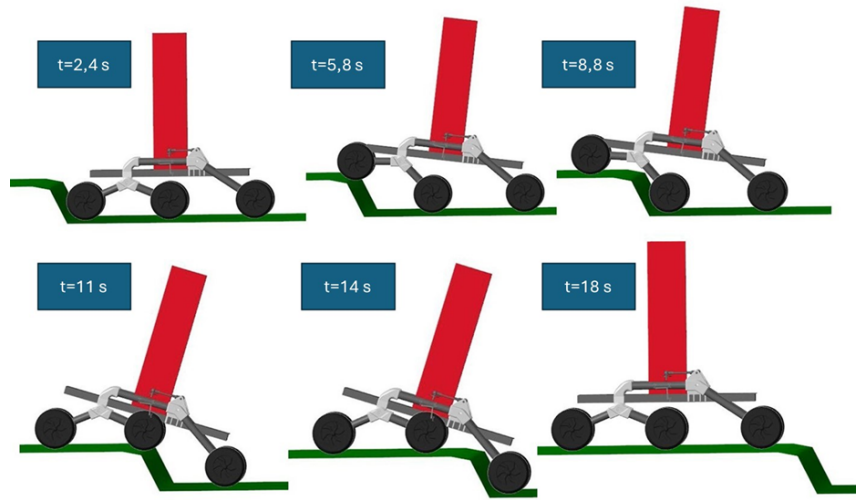


Figure 2.222: Obstacle crossing phases for the first model of the rover at different time intervals

in the time required to overcome the obstacle. The new model of rover is observed to complete this task in two seconds less than the old one. It is noteworthy that the front wheels of the new model make contact with the step slightly later than those of the old model ($t=2.4s$ versus $t=2.55s$). Nevertheless, the new model displays superior efficiency during the simulation, completing the obstacle crossing in a shorter overall time. One aspect common to both models is the torque peak, which occurs when one of the wheels makes contact with the obstacle. However, one particularly relevant element is the maximum torque values achieved. The old model exhibited a torque value of 6 Nm, while the new model demonstrated a significantly lower value of 3 Nm. A comparison of the curves reveals that the new model (in red) consistently exhibits lower torque values than the old model (in blue). This indicates that the new model requires less torque to overcome the obstacle, which is a significant advantage from an energy perspective. Lower torques translate into less energy consumed by the motors, improving the overall efficiency of the rover and reducing energy consumption during operation.

Middle Wheels

The considerations for the central wheels are analogous to those made for the front wheels. The torque peaks occur at the instant of contact between the wheels and the step, which occurs at $t=2.4s$ for the older model and at $t=2.55s$ for the newer model. In both instances, the maximum peak is recorded at the moment the central wheel makes contact with the step. It was observed that the skidding phenomenon persisted when the rover was situated on the edge of the step, with both the central and rear wheels in use. Three intervals can be identified, corresponding to the passage of each wheel over the step (see Fig. 2.224). In each of these instances, the trend of the torques demonstrates consistent behaviour at defined intervals. In particular, the torque remains constant and

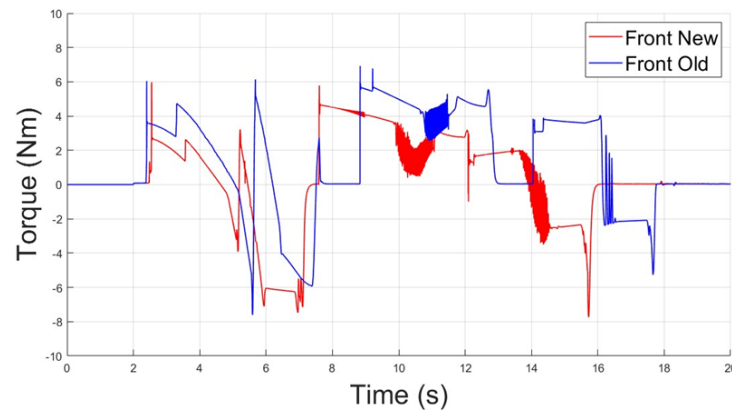


Figure 2.223: Comparative analysis of the torque generated by the front wheel motors in the new and old models

positive until the front wheel has completely passed the obstacle ($t=5.2s$ for the new model and $t=5.6s$ for the old model). Subsequently, a braking torque is observed, due to the fact that the rear wheels begin to contribute to maintaining the set speed. The skidding phenomenon is observed to recur when the rover is positioned on the edge of the step, with both the middle and rear wheels in use. Ultimately, even when the third rear wheel

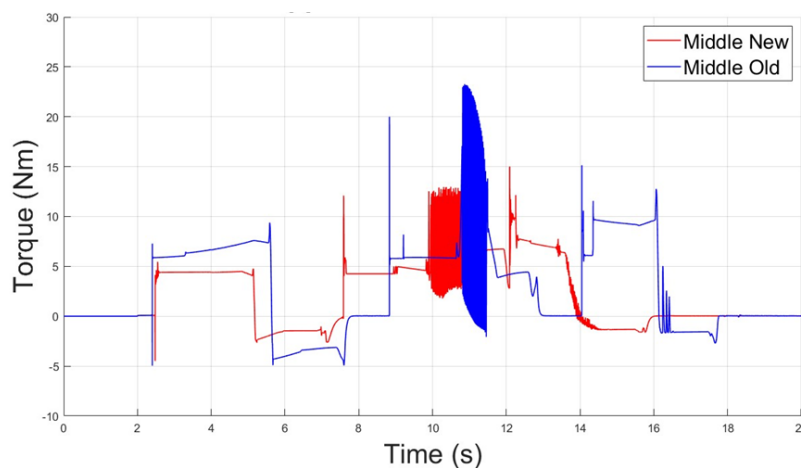


Figure 2.224: Comparative analysis of the torque generated by the middle wheel motors in the new and old models

has crossed the step, the pattern of central wheel torques remains consistent at specific intervals: positive until the rear wheel has fully crossed the step and nearly zero when all wheels have completed their respective maneuvers. A comparison of the two curves (Fig. 2.224) also demonstrates that the new rover requires less time to cross the step than the previous model. Moreover, it is evident that the red curve, which correlates with the new rover, is persistently positioned below the blue curve, which corresponds to the old model. This verifies the previously demonstrated result for the front wheels, namely that

the torque values required by the new model are lower. Consequently, from an energy perspective, the stepping operation is less energy-intensive, which enhances the overall efficiency of the system.

Rear Wheels

With regard to the rear wheels, the distinction between the two models is less discernible (see Fig. 2.225). This leads to the conclusion that these wheels are less involved in overcoming the obstacle and that the displacement of the centre of gravity has a lesser influence on their behaviour. This is likely attributable to the fact that the obstacle was approached with the bogie. If the obstacle had been overcome with the rocker, the involvement of the rear wheels would have been greater, resulting in a more pronounced difference in torque between the two models.

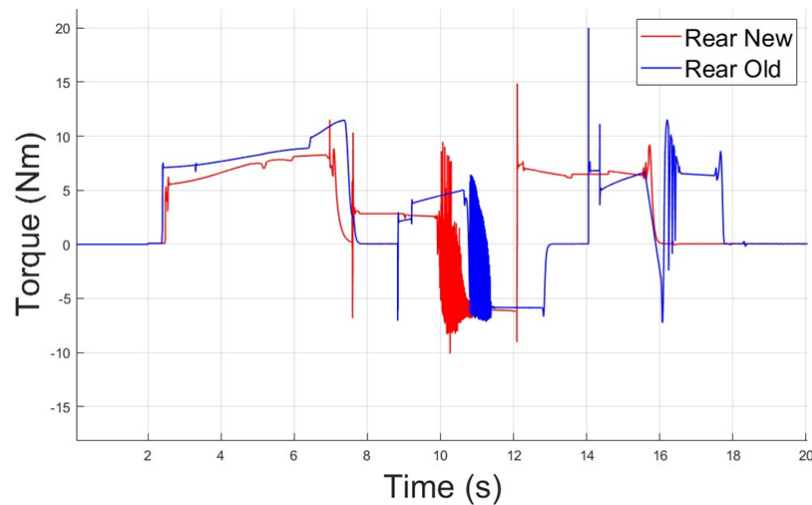


Figure 2.225: Comparative analysis of the torque generated by the rear wheel motors in the new and old models

2.4.3 Conclusions

The process of developing the full-scale prototype constituted a crucial phase in the validation of the theoretical model and the conversion of simulation results into practical solutions. The structural and dynamic analysis of the rover enabled the identification and resolution of a series of design issues related to the navigation of complex terrain, stability on uneven surfaces and the optimisation of energy efficiency. The simulations demonstrated that the rover is capable of maintaining an optimal balance between structural strength and operational flexibility, while ensuring a load distribution that maximises traction and reduces stress on mechanical components. It is notable that the system demonstrated the capacity to adapt to variable operating conditions, confirming the validity of the modular design and configurations that were explored. Simulations have demonstrated that alternative configurations can offer significant advantages in terms of specific performance, providing a foundation for further optimisation for future applications. The analysis of energy consumption, conducted under a variety of operational conditions, has provided valuable insights that can be leveraged to enhance the autonomy of the rover, a critical aspect for long-term missions in challenging environments. Concurrently, the comparative analysis of mechanical and kinematic configurations has enabled the identification of innovative design solutions that combine constructive simplicity and high performance. These solutions demonstrate the potential of the rover not only for exploratory applications but also for industrial use and urban environments. In conclusion, the enhancements made to the second prototype represent a significant advancement in comparison to the initial version, demonstrating the value of an iterative approach in design. The integration of advanced components and the refinement of simulation methodologies have enabled a more accurate representation of the potential of the system, confirming the importance of a synergy between theoretical research, technological development and modelling validation. This work provides a robust foundation for future project developments and the design of future rovers and advanced robotic systems capable of meeting the technical and operational challenges of the most demanding contexts.

–3–

Development and Experimental Validation of Rover Prototypes

CONTENTS: **3.1 Introduction.** **3.2 Small Rover Prototype.** 3.2.1 Movement Testing of the Rover – 3.2.2 Vibration Spectral Analysis – 3.2.3 Robotic Arm Integration for Enhanced Obstacle Overcoming. **3.3 Full-Scale Rover Prototype.** 3.3.1 First Prototyping – 3.3.2 Second Prototyping – 3.3.3 Conclusions.

3.1 Introduction

The development of a robotic rover requires an iterative process that integrates modelling, simulation and experimental validation. This chapter presents the key steps taken to move from conceptual design to the realisation of physical prototypes, highlighting the methods used to test and refine the system at different sizes.

The chapter is divided into two main sections: firstly, the small rover prototype, which was developed as a preliminary test platform, and secondly, the full-scale prototype, which represents the final design version. The small rover prototype played a fundamental role in validating early design hypotheses and logics. Through a series of experimental tests, including motion evaluation, obstacle crossing, vibration analysis and robotic arm integration, its performance was evaluated under controlled conditions, allowing refinement of mechanical and software components.

The second part presents the full-scale prototype, which represents the final stage of the design process. The construction of this prototype required advanced manufacturing techniques such as 3D printing and precision assembly to ensure structural robustness and functional accuracy.

The details regarding the hardware used in these prototypes are provided in the appendix: the small rover prototype in Appendix A.1, the robotic arm in Appendix A.2, and the full-scale rover in Appendix A.3.

Through the analysis of these different experimental phases, this chapter aims to provide insights into the challenges encountered during prototype development and the solutions adopted to overcome them. It also provides a basis for future extensions and optimisations, ensuring that the methods presented can be extended to more complex robotic applications.

3.2 Small Rover Prototype

This section describes the development and experimental validation of a small scale rover prototype (Fig. 3.1). The primary objective of this prototype was to evaluate the locomotion performance, obstacle crossing and stability of the system under various operating conditions. The rover was subjected to a series of tests to assess its mobility, response to terrain variations and vibration behaviour.

Controlled experiments were used to test different motion strategies, providing insight into the effectiveness of the suspension system and control algorithms employed. This iterative approach allowed the identification of key design improvements that could be transferred to the full-scale model, ensuring a smoother transition from conceptual development to real-world implementation.

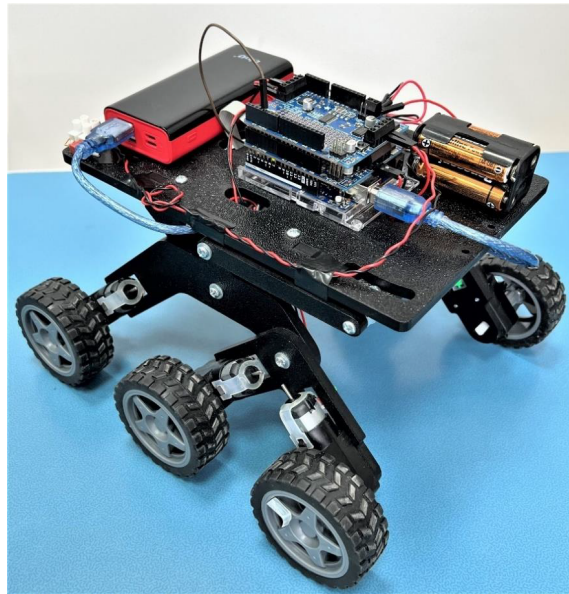


Figure 3.1: Configuration of the rover with two different battery packs

The diagram in Fig. 3.2 represents the functional diagram of the prototype shown in Fig. 3.1. The system is powered by two independent power supplies: a 6V, 2600mAh source powering the MCU (Motor Control Unit) and a 5V, 2A source supplying the Controller.

In this configuration, the controller acts as the main processing unit, managing the overall operations and decision-making of the system. It communicates with the MCU, which is responsible for sending control signals to the motors.

3.2.1 Movement Testing of the Rover

This section presents the experimental validation of the motion capabilities of the rover, focusing on straight-line motion, steering performance and obstacle crossing. These

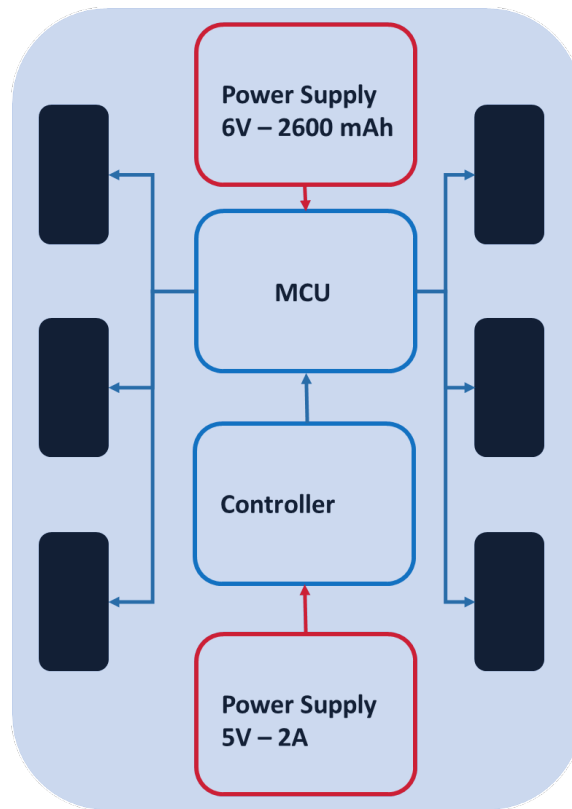


Figure 3.2: Configuration diagram of the prototype

tests assess the effectiveness of the locomotion system and provide insights for further optimisation. The data collected will also support improvements to the full-scale prototype.

3.2.1.1 Remote Management and Communication Logic

In the process of experimentation, two distinct methodologies were employed for the remote control of the mobile robot. The first was based on the Transmission Control Protocol (TCP), while the second was based on the User Datagram Protocol (UDP). In accordance with the TCP protocol, the Arduino was configured to operate as a web server within the WiFi network. A simple HTML web page was developed and implemented on the server to allow users to select and send commands to the motor control system of the rover. Figure 3.3 illustrates an early iteration of the Web-Server Dashboard. The web page that was implemented utilized the Transmission Control Protocol (TCP), which is known for its secure data transmission capabilities through the implementation of connection checks and packet integrity verification. However, this communication method was found to be too slow and resource-intensive for the Arduino board, which was required to manage the simultaneous creation of a WiFi network, the hosting of a webpage, and the execution of continuous checks on connectivity and data integrity within each code cycle.

In view of the aforementioned limitations, an alternative control logic based on the

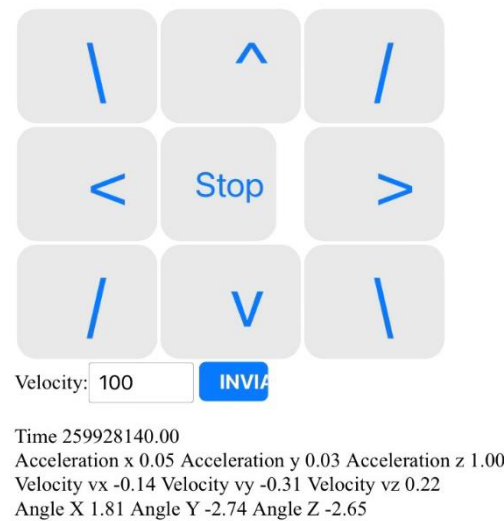


Figure 3.3: Dashboard of the developed web server for remote control of the mobile robot

UDP protocol was adopted. Typically utilized for streaming and broadcast services, UDP enables immediate data transmission from sender to receiver following connection initiation, without necessitating packet integrity checks or continuous connection maintenance.

The "Osoyoo® WiFi UDP Robot Car Controller" application (see Fig.3.4), available on Google Play™ and the Apple Store™, was employed for the transmission of data to the Arduino device. The application in consideration facilitates the operation of a mobile robot through the transmission of alphanumeric commands. Upon receipt of these characters, Arduino initiates the execution of specific code sections through the use of switch-case logic, thereby enabling the implementation of diverse motor operations in accordance with the command. Arduino is then configured to establish a WiFi network, after which it awaits the transmission of data packets from the connected smartphone application.

3.2.1.2 Straight-Line Motion Test

This chapter presents the results of experimental tests conducted on a prototype moving along variable paths with the aim of gathering data on its dynamics using an IMU sensor. The primary objective is to analyze the dynamics of the system by processing and interpreting the acquired data, allowing for an assessment of the performance of the prototype and the identification of potential areas for improvement.

The initial experiments concentrated on the advanced straight-line motion of the rover on a flat, uniform surface. This was tested in two configurations: Bogie-Front and Rocker-Front.

The tests were conducted at varying forward speeds, which were adjusted through DC motors supplied with different voltages. This was achieved by modulating PWM (pulse width modulation) values via an Arduino® controller board with 8-bit registers.

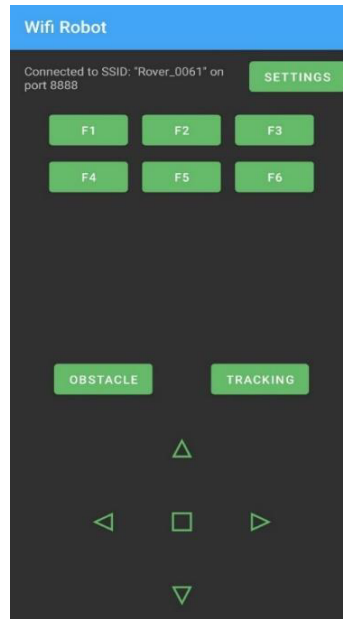


Figure 3.4: Osoyo App for remote control of the mobile robot

The PWM values utilized to establish the requisite voltages were as follows: 255, 225, 200, 175, 150, 125, and 100.

Despite the initial configuration of the onboard sensor to sample at 416 Hz, the constraints imposed by the Arduino®'s dynamic memory and data acquisition logic resulted in an actual sampling rate of approximately 90-100 Hz.

In the data processing stage, before the signal interpretation phase, the graphs were modified by establishing limits on the x-axis (time axis, expressed in milliseconds) through the utilisation of the 'xlim' command. This adjustment excluded an interval of approximately 10 seconds from the graphs, corresponding to the initial latency between the commencement of the acquisition process and the actual commencement of the motion of the rover. This delay was attributed to the acquisition software.

The results of the accelerations measured during the test, with the bogie at the front and a PWM value of 100, are shown in Fig. 3.5. The x-axis represents time in milliseconds, showing the time-varying nature of the acceleration data throughout the test.

A preliminary observation concerns the unit of measurement and the initial values of the accelerations recorded by the IMU. The accelerometer measures accelerations in units of [g], which is the acceleration due to gravity. Consequently, while the accelerations along the horizontal axes (a_x and a_y) are initialized to zero, the vertical axis (a_z) registers a value of one, reflecting the influence of gravity. The positive value of a_z indicates that the axis is oriented downward.

In Fig. 3.5, where the x-axis represents time in milliseconds, clearly distinguishes periods of rover motion from periods of relative inactivity, and helps to identify travel times within the test section. As hypothesized, the longitudinal accelerations (a_x) show significant fluctuations aligned with the forward motion of the rover, providing crucial

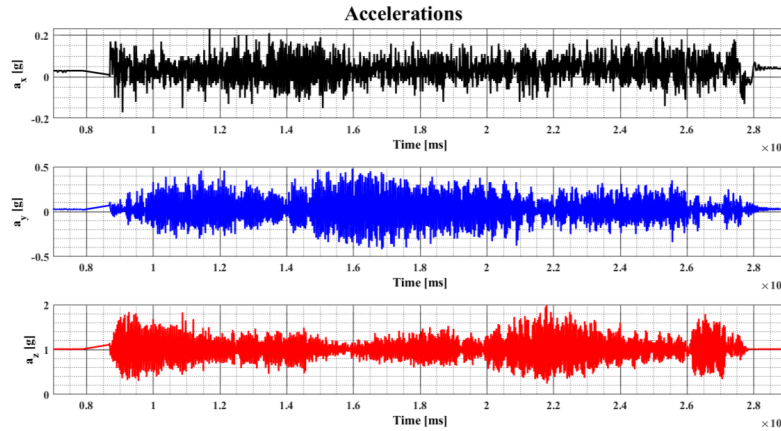


Figure 3.5: Acceleration plot at 100 PWM in the straight-line test, with Bogie Front

insight into the dynamic variations that occur during motion. These oscillations provide valuable information for analyzing the dynamic performance and motion stability of the rover.

Significant variations in lateral accelerations (a_y) can be attributed to several factors, including play between suspension components, their deformability, and a pronounced imbalance of the suspended masses on the rover chassis. These variations add complexity to the data analysis process.

As a result, data manipulation is critical to gain additional insight from the highly oscillatory acceleration plots. Plots of accelerations obtained with PWM values set to 100, 150, 200, and 255 for both directions of motion are shown in Fig. 3.6. Analysis of the figure 3.6 shows that the accelerations for both directions of motion oscillate within a range of similar maximum values. This consistency suggests that the dynamic behavior of the rover remains comparable across different speeds as controlled by the varying PWM inputs.

For the purpose of focusing the analysis, it is examined a single case: the Bogie front configuration with a PWM value of 100 (see Fig. 3.7). This will allow a detailed evaluation of the dynamic behavior of the rover under this specific setting, providing insight into acceleration patterns and any significant oscillations or variations.

Comparing this plot in Fig. 3.7 with the one obtained with the forward rocker configuration (see Fig. 3.8), it can be seen that the maximum values for the angular velocities remain similar for both pitch (ω_y), roll (ω_x), and yaw (ω_z). This suggests a consistency in the angular dynamics of the rover, regardless of the forward configuration adopted.

The test conducted with the bogie at the front and a PWM value of 100 revealed that the attitude angle plots demonstrated a trajectory of the rover that was not perfectly straight, despite the motors being set to turn at the same speed. In particular, the Bogie front resulted in a slight rightward steering tendency, while the Rocker front exhibited a leftward deflection. These observations are in contradiction with the initial hypothesis that the prototype would move in a straight line, indicating a deviation from the planned

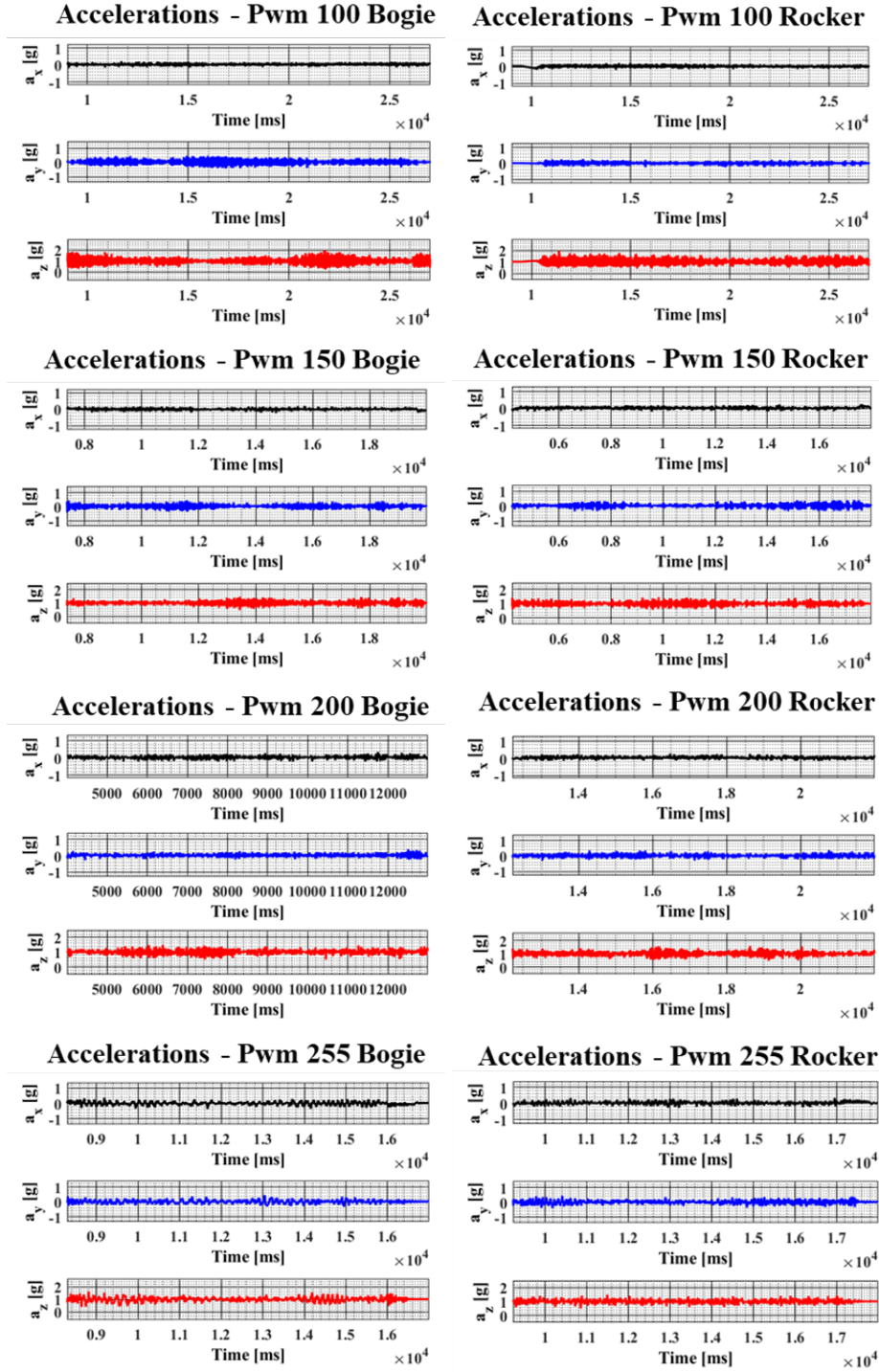


Figure 3.6: Accelerations Straight-Line PWM: 100, 150, 200, 255

trajectory. To identify the potential causes of this discrepancy, the hypothesis that an imbalance was the cause was evaluated but not considered, as the simulations, with the

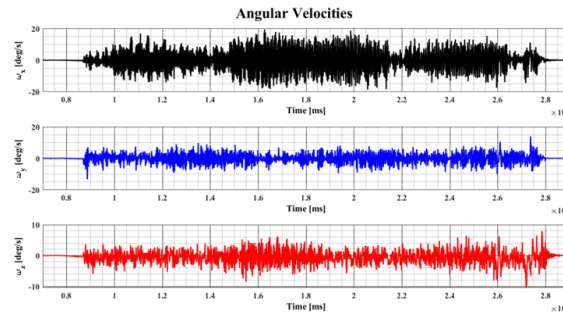


Figure 3.7: Angular velocities at 100 PWM input for the Bogie front configuration (Serial Port Acquisition)

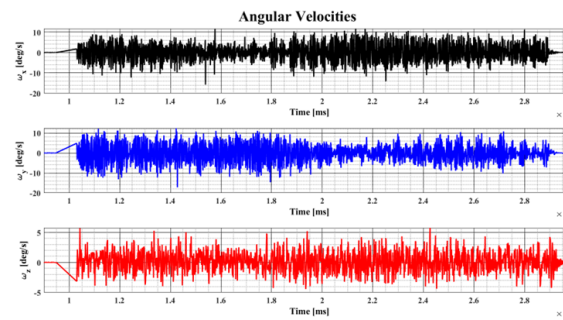


Figure 3.8: Angular velocities at 100 PWM input for the Rocker front configuration (Serial Port Acquisition)

same imbalance, predicted a rectilinear motion. It was hypothesized that an imbalance could result from the asymmetrical arrangement of the wiring masses on the prototype: however, this factor was found to be insufficient to explain the observed deviation at the front of the wiring masses.

Consequently, additional possible explanations were explored, such as differential clearance between the joints or variations in the torque of the threaded connections coupling the joints. These mechanical variables could significantly affect the dynamics of the rover, altering the trajectory of motion during testing. An additional hypothesis that has been put forth to explain the observed divergence from the expected trajectory is a discrepancy in the speed of the two motor banks. However, this can be excluded on the assumption that both motor shields are powered by the same voltage, which should ensure uniformity in voltage distribution. In the absence of any hardware defects in the shields, the discrepancy in the trajectory of the rover must therefore be attributed to other causes.

The inclination of the rover to curve to the right with the bogie at the front is clearly evident from Fig. 3.9. The roll (x-axis) and pitch (y-axis) angles illustrate a certain degree of stability, although some degree of wobble is discernible. However, the yaw angle (z-axis) plot is particularly enlightening, as it illustrates a consistent decrease over time. In fact, the yaw angle begins at a value close to zero at the outset of the test and reaches

-10° at the conclusion of the run. This reduction of approximately 10° in the yaw angle, which represents a rotation around the z-axis, verifies that the rover exhibits a tendency to drift to the right during its movement.

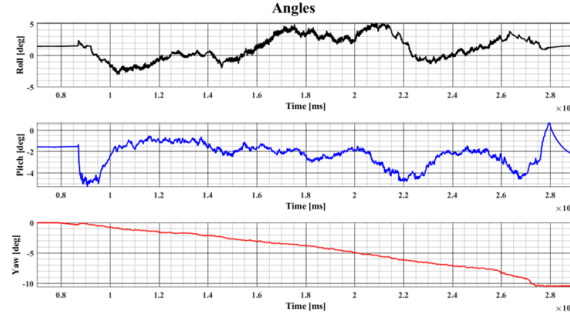


Figure 3.9: Prototype angles trajectory at Bogie front configuration as 100 PWM

During straight motion tests with the Rocker front, the rover exhibited a tendency to curve slightly to the left (see Fig. 3.10). A progressive increase in yaw angle was observed. This increase, indicative of a rotation in the opposite direction compared to the tests with the Bogie front, confirms the tendency of the rover to deviate to the left when configured with the Rocker front.

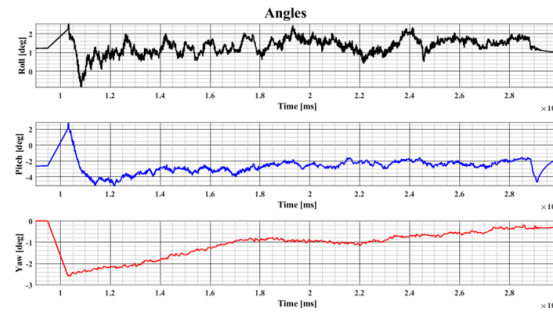


Figure 3.10: Prototype angles trajectory at Rocker front configuration as 100 PWM

As the rotational speed of the engines increases, two principal effects on the plots of attitude angles are observed: a reduction in the travel time for a given path length and an increase in the trajectory error. These results suggest that the observed deviations may be less related to an imbalance of masses and more to a difference in rotational speeds between the two banks. Such a discrepancy could result in wheel slip, which in turn could lead to skid steering. For instance, an examination of the plot in Fig. 3.11 of the attitude angles in the configuration with the bogie at the front and a PWM value of 225 reveals that the yaw angle is approximately twice that recorded with a lower PWM value. This considerable increase in yaw angle implies that, at elevated speeds, discrepancies in motor rotation between the two banks become more pronounced, influencing the trajectory of the rover to a greater effect. This phenomenon underscores the necessity for precise

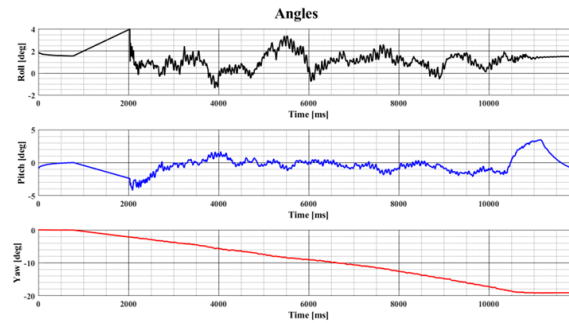


Figure 3.11: Prototype angles trajectory at Rocker front configuration as 255 PWM

motor speed control to minimize trajectory errors and optimize path stability during high-speed operations.

The data regarding the attitude angles plots for all PWM speeds (150, 200, and 255) obtained in both bogie and rocker front are presented in Fig. 3.12 for the sake of completeness.

Although the curvatures performed in the two configurations were comparable during the tests, it was observed that the yaw angle computed for the configuration with the rocker at the front was lower than that actually observed. This discrepancy can be attributed to the methodology employed in the construction of the algorithm utilized to calculate the yaw angle. This algorithm is based on an integral calculation performed from the angular velocity data (ω_z). One potential solution to this issue is the integration of a magnetometer into the inertial measurement unit (IMU), which would enhance the precision of the yaw angle measurement.

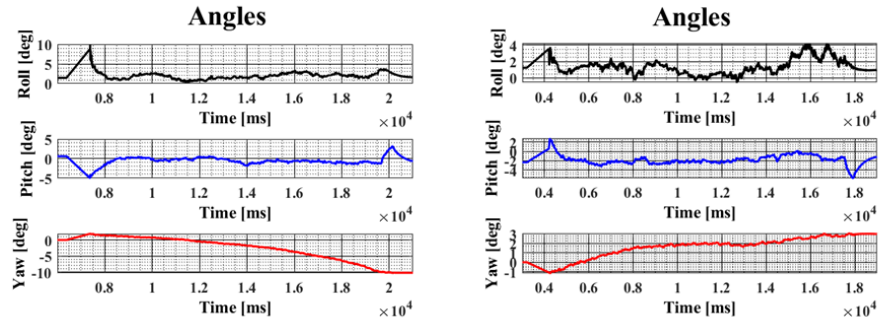
3.2.1.3 Steering Performance Test

Experimental steering tests on flat terrain showed that the front-bogie configuration was the most optimal. The rover demonstrated its ability to execute both left and right turns, demonstrating the principles of motion of a six-wheeled, unsteered rover. In such a vehicle, turns are achieved by modulating the rotational speeds of the wheels on each bank, rather than using traditional steering mechanisms.

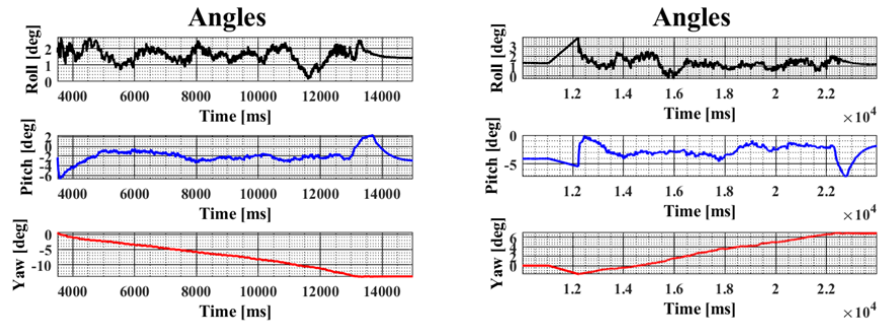
Rather than using an alternative technique in which the banks rotate in opposite directions - which could lead to uncontrolled skidding - the banks were made to rotate in the same direction but at different speeds, with one bank turning at twice the speed of the other. This strategy minimizes wheel slip, resulting in smoother steering and more precise control as the bogie moves forward.

The right turn test was performed by keeping a constant 1:2 speed ratio between the motor banks. The following PWM values were used in Arduino® to achieve the required speeds: 150 for the left bank and 75 for the right bank. This ratio allowed for precise regulation of the wheel speed, providing smooth and controlled steering.

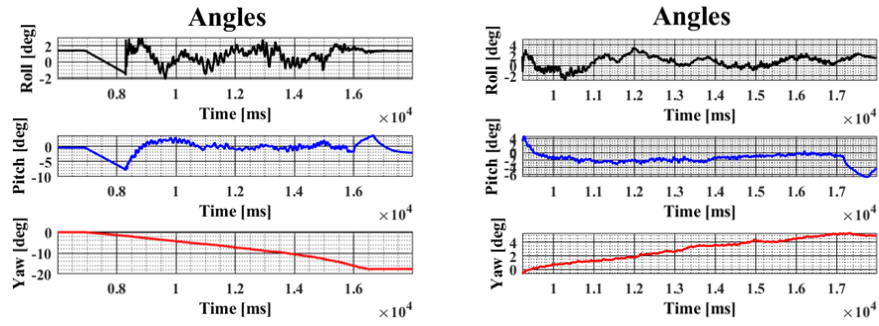
The plot in Fig. 3.13 of the angular accelerations and attitude angles obtained during this test shows an increase in the yaw angle in the negative direction, confirming the



(a) Angle at Bogie Front as input 150 PWM (b) Angle at Rocker Front as input 150 PWM



(c) Angle at Bogie Front as input 200 PWM (d) Angle at Rocker Front as input 200 PWM



(e) Angle at Bogie Front as input 255 PWM (f) Angle at Rocker Front as input 255 PWM

Figure 3.12: Angles Straight-Line PWM: 150, 200, 255

successful execution of the right turn maneuver. The significant increase in the yaw angle, compared to that observed during the straight line motion, indicates precise and effective steering.

The roll angle oscillates around a positive non-zero value. Since roll is measured around the x-axis, this result is consistent with expectations for a right turn. During such maneuvers, vehicle dynamics and lateral force distribution affect the attitude of the rover,

causing a change in the roll angle. This effect is typical because centrifugal forces act laterally during steering and affect the roll.

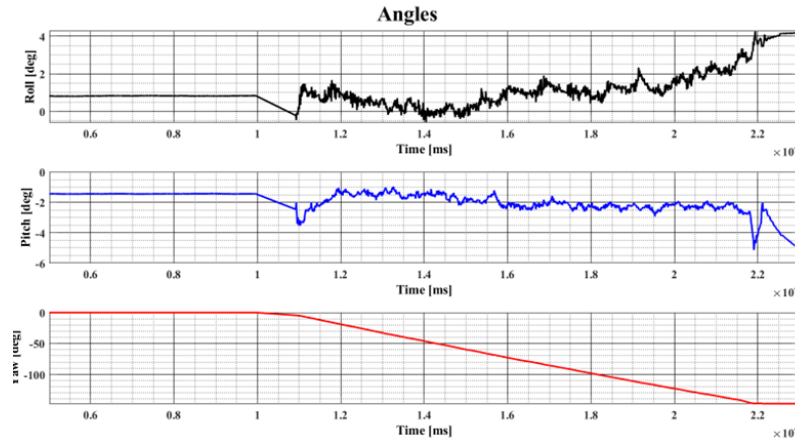


Figure 3.13: Angles plot for steering maneuver (150 left 75 right) with bogie front configuration

The effect of rover steering is also evident in the plot of angular velocities in Fig. 3.14. While the ω_x (roll) and ω_y (pitch) components tend to oscillate around zero, indicating relative stability in these axes during steering, the behavior of ω_z (yaw) is notably different. At the beginning of the steering maneuver, ω_z undergoes a modular increase and then oscillates around a non-zero value.

This shift in angular velocity, particularly along the z-axis, highlights a significant change in the motion behavior of the rover, confirming the steering dynamics hypothesis. The increase and subsequent oscillation in ω_z indicate that the rover has begun a change of direction, with rotation about the vertical axis reflected in the increased yaw angular velocity.

These observations are critical for verifying the effectiveness of the motion control settings and for further optimizing the navigation and steering strategies of the rover in real-world scenarios.

3.2.1.4 Obstacle Crossing Test

Following the flat terrain tests, the prototype was evaluated for its obstacle crossing capability, specifically with an equilateral triangular obstacle with 7 cm sides (see Fig. 3.15). Tests were conducted in two configurations: with the bogie forward and with the rocker forward to evaluate the differences in obstacle-crossing performance between these configurations.

In the first attempt, the obstacle was approached after traveling a straight section, using the gained inertia to aid and accelerate the maneuver. However, this strategy resulted in a slight rebound of the rover upon impact with the step due to the inherent stiffness of the passive articulated suspension system. In addition, the attitude angle plots

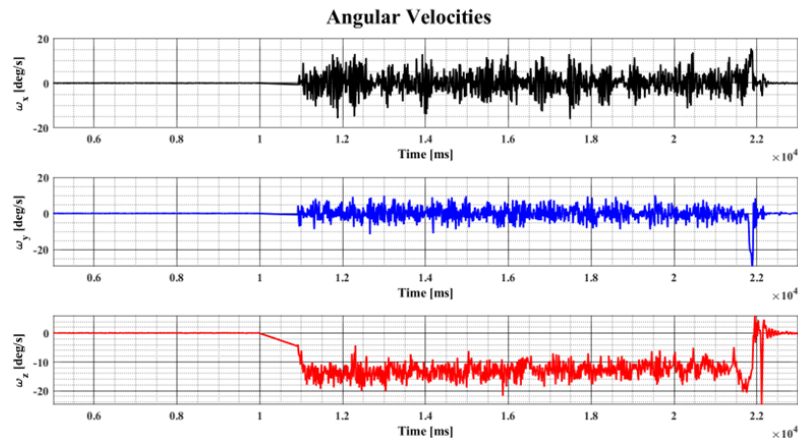


Figure 3.14: Angular velocities Plot for steering maneuver (150 left 75 right) with bogie front configuration

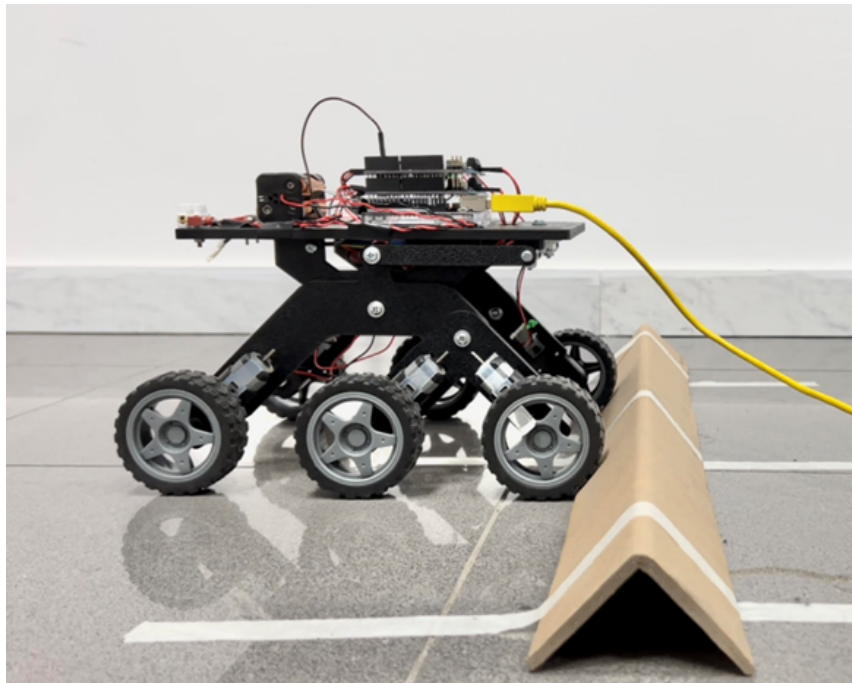


Figure 3.15: Experimental setup illustrating the robot crossing an obstacle

showed significant roll as the vehicle hit the obstacle with one set of wheels first, followed by the other.

The following tests were performed from a stationary position with the front wheels in full contact with the obstacle. The experiments showed that a minimum PWM output of 150 was required for the rover to successfully traverse the obstacle with the rocker configuration at the front. In contrast, with the bogie configuration at the front, the rover crossed the obstacle with lower PWM values, indicating less demand for higher speeds or

force.

These results support the hypothesis that forward progress with the bogie is consistently more advantageous than with the rocker. When the rover approaches an obstacle with the rocker front, the load distribution causes the two center wheels to lift, resulting in skidding and rapid spinning without sufficient grip, leading to loss of traction. On the other hand, the bogie front configuration allows the rover to face obstacles more smoothly because the lifting of the front wheel pushes the middle wheel down, increasing its tractive force.

The plots in Figure 3.16, obtained from the IMU data recorded onboard the prototype during the obstacle crossing test with the motor power set to 150 PWM, are now analyzed. These plots provide insight into the dynamic behavior of the rover, including its accelerations, angular velocities, and attitude angles as it overcomes the obstacle. From the

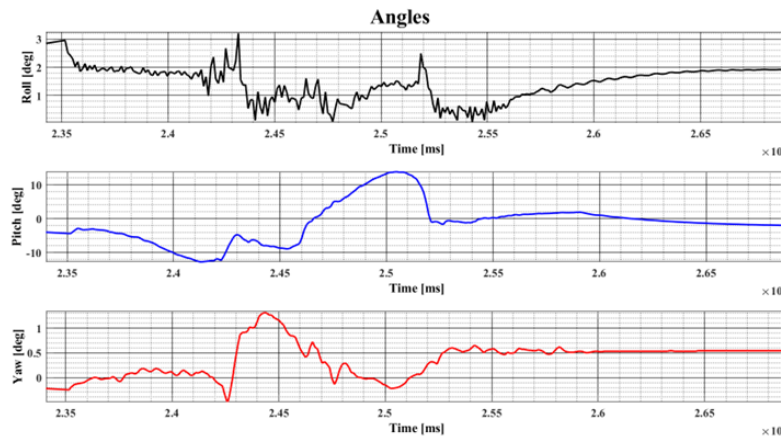


Figure 3.16: Angle variations (roll, pitch, and yaw) during obstacle crossing at 150 PWM

pitch plot analysis (see Fig. 3.16), three distinct peaks are observed. The first two peaks, both negative, indicate a lowering of the frame corresponding to the lift caused by the bogie wheels. The third peak, positive and the highest, results from the rocker causing the frame to tilt more in the opposite direction.

This analysis was extended to different engine speeds - 200, 225 and 255 PWM. Despite the variation in speed, the shape of the peaks in the pitch plot remains consistent, as it is primarily determined by the step geometry. For brevity, other plots are not shown as they are essentially similar. However, it is noteworthy that the plots become more compressed at higher wheel speeds, since the steps occur over a shorter duration when the time scale on the x-axis is kept uniform for comparison.

3.2.1.5 Conclusions

Finally, tests on the motion of the rover highlighted the dynamic performance of the prototype in variable trajectory configurations. Using an IMU sensor, data on accelerations and angular velocities were collected and analyzed to identify changes in stability and

oscillations in performance. Bogie-Front" and "Rocker-Front" configurations provided insight into dynamic differences and highlighted the need for specific calibrations to optimize stability and trajectory control.

3.2.2 Vibration Spectral Analysis

Consistent with the multibody simulations performed in section 2.3.3, it is now reasonable to proceed with testing of the simulated prototype.

The efficiency of the IMU sensors was first verified by creating a controlled imbalance on the rover. This was done by attaching a DC motor to the upper body of the rover and running it through an electric power source. The goal was to confirm that the frequency identified in the FFT plots, analyzed using the Arduino code associated with the sensors, matched the rotational frequency of the motor. This frequency was then measured using a tachometer, specifically the Rs-Pro Rm-1501 electronic model (see Fig. 3.17). This procedure ensured the correct functionality of the sensors before they were used for more complex analysis on the rover.



Figure 3.17: Tachometer Rs-Pro Rm-1501

A tachometer is an instrument that measures the rotational speed of an element by projecting a laser beam onto a reflective point. In this case, a small square of reflective tape is attached to the rotating element. The tachometer detects the reflected beam of light and displays the instantaneous speed on its screen. This non-invasive and accurate measurement method is particularly valuable for calibrating and verifying the performance of devices such as motors, as in the case of the rover.

The procedure was used to verify the accuracy of the LSM6DS3TR and MPU6050 IMU sensors in detecting the rotational frequencies of the DC motor mounted on the rover (see figure 3.18). By applying variable voltages to the motor, controlled by a potentiometer, the frequency trends were monitored. The frequency peaks identified by the sensors and analyzed by the Arduino code matched those recorded by the tachometer. This result validated the calibration of the sensors and confirmed their effectiveness under different operating conditions.

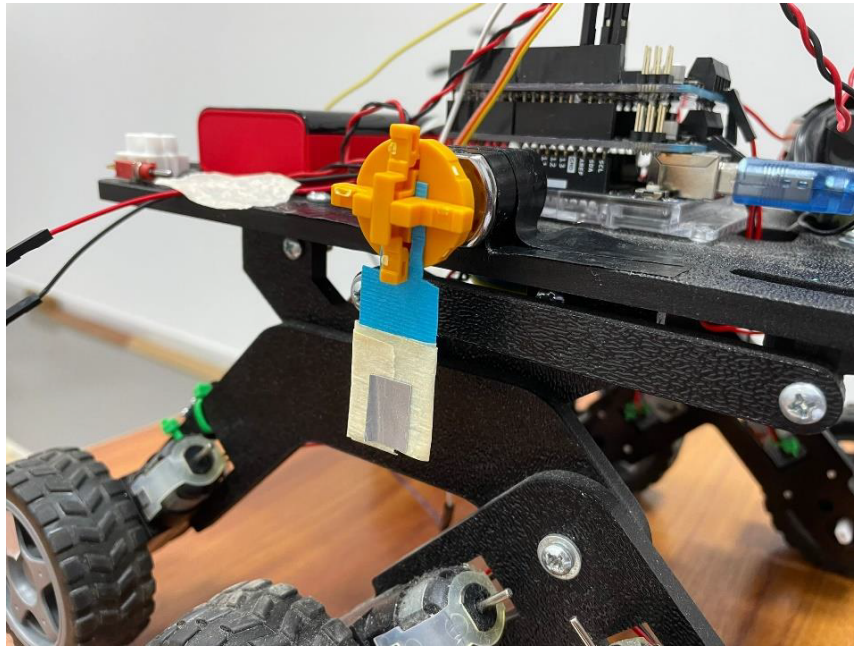


Figure 3.18: DC motor setup for sensor testing and calibration

In Fig. 3.19, the red line in the frequency plot indicates the maximum frequency achieved, highlighting the peak speed of the motor at the highest voltage set by the potentiometer. This peak represents the upper limit of the performance of the motor during the test.

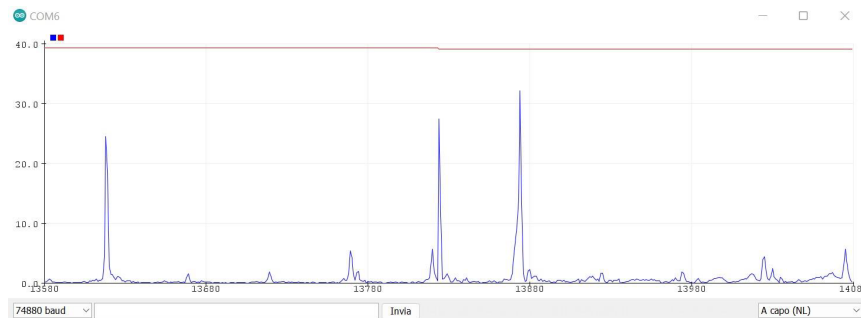


Figure 3.19: FFT Analysis with major peak representation

After confirming the functionality of the sensors through verification with the tachometer, extensive testing was performed on the rover. First, the LSM6DS3TR sensor integrated into the Arduino control board was used to collect acceleration data in three directions. Fast Fourier Transform (FFT) analysis of this sensor revealed peaks corresponding to rolling speeds.

Next, MPU6050 sensors were used, which had the advantage of not being mounted on the top plate, but on the left and right rockers. This setup allowed more precise differentiation of motion between the two motor banks. Each motor bank was equipped

with an MPU6050 sensor, allowing tests to be performed that produced FFT plots showing peaks corresponding to the specific rolling speeds of each motor bank.

The MPU6050 sensors were connected to the Arduino via the SDA and SCL pins. Upon assembly, it was observed that the Z-axis of the sensors was aligned with the Y-axis representing the lateral motion of the rover. Therefore, the accelerations of greatest interest during steering maneuvers were those detected along the Z-axis.

Real-time FFT analysis of the prototype allowed a detailed study of the dynamic behavior of the two wheel motor banks during rover navigation. This in-depth real-time analysis provided significant insights into the motion dynamics of the rover, which helped to optimize its stability and efficiency.

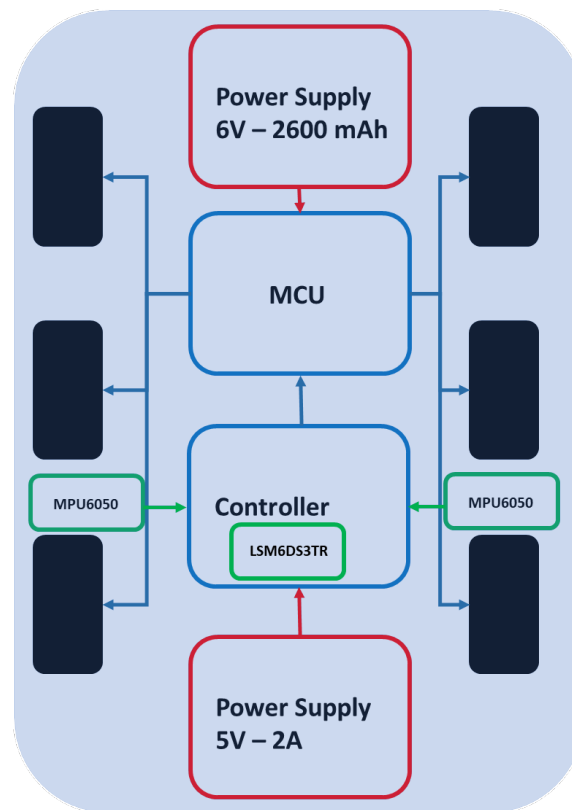


Figure 3.20: Configuration diagram of the prototype for the tests

In Fig. 3.20 is shown the configuration of the prototype for the following tests.

3.2.2.1 Straight-Line Vibration Analysis

The initial experimental tests focused on evaluating the straight-line motion of the rover on smooth, level terrain. To maintain straight-line motion, it is essential that all wheels travel at the same speed. A series of tests were conducted to ensure that the rover followed a straight path at a nearly constant speed, with different forward speeds in each test.

Arduino code was then developed to perform real-time FFT analysis on the rover to monitor its trajectory and verify the forward speed of each motor bank. Two different control methods were used for this purpose: proportional-integral-derivative (PID) control and sliding mode control. These approaches were designed to provide precise speed control and maintain the desired trajectory.

PID results

To achieve effective velocity control using PID control, it is critical to accurately define the controller gains- K_p , K_i , and K_d -which correspond to the proportional, integral, and derivative contributions, respectively. In this case, only the proportional (K_p) and integral (K_i) contributions were used. The parameters for these contributions were defined based on the model simulations described in the Chapter 2:

$$K_p = 0.13$$

$$K_i = 0.28$$

Through a series of tests, it was experimentally verified that differentiating the gains between the two motor banks produced better results. Specifically, the K_i associated with the left bank was adjusted to correct for small deviations in the steering of the rover, improving its trajectory control and ensuring more precise straight-line motion:

$$K_{p1} = 0.13$$

$$K_{i1} = 0.20$$

$$K_{p2} = 0.13$$

$$K_{i2} = 0.28$$

where subscript 1 is relative to the left bank and subscript 2 is relative to the right bank.

Once the gain parameters were set, the ability of the rover to maintain a straight trajectory was verified. A target velocity of 10 rad/s was set, with the PID controller tasked with keeping the velocities of both motor banks constant around this optimal value.

The test was conducted on smooth, level ground over a 90-second interval. The velocity values from the test were displayed on the serial monitor and captured for further analysis in MATLAB. Once imported into MATLAB, graphs were generated to analyze the velocity trends of both motor banks, providing insight into the performance and stability of the PID control system.

In Fig. 3.21, it confirms the correct operation of the PID controller. The angular velocity of the two banks, starting from a zero value corresponding to the stationary rover, quickly reaches the steady state condition in about one second and then maintains an almost constant velocity close to the desired value of 10 rad/s.

When the error, defined as the difference between the target speed and the actual forward speed, is plotted, it is expected that the values will oscillate as close to zero as possible. This behavior is illustrated in Fig. 3.22, where the error starts from a maximum value that initially matches the target speed. After about a second, the error stabilizes near zero and oscillates around a constant, near-zero value for the remainder of the simulation.

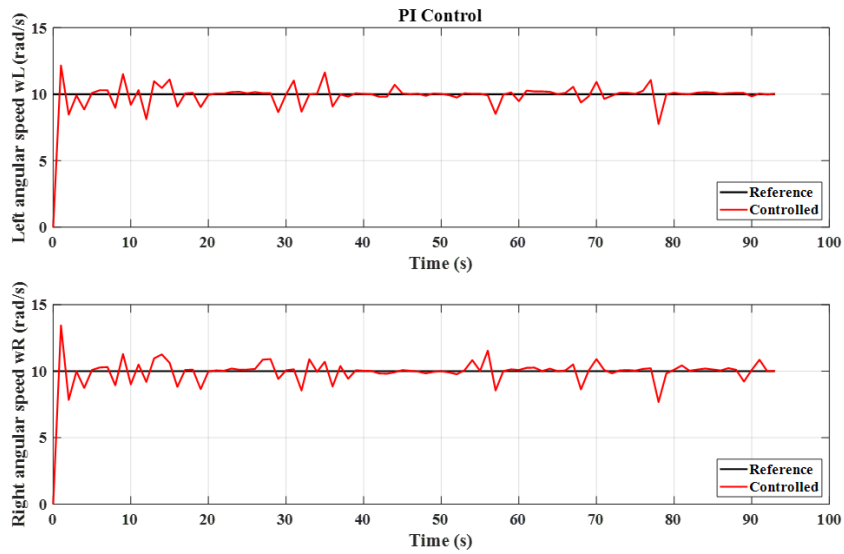


Figure 3.21: Angular velocities of the left and right wheels under PI controller, comparing reference and controlled outputs

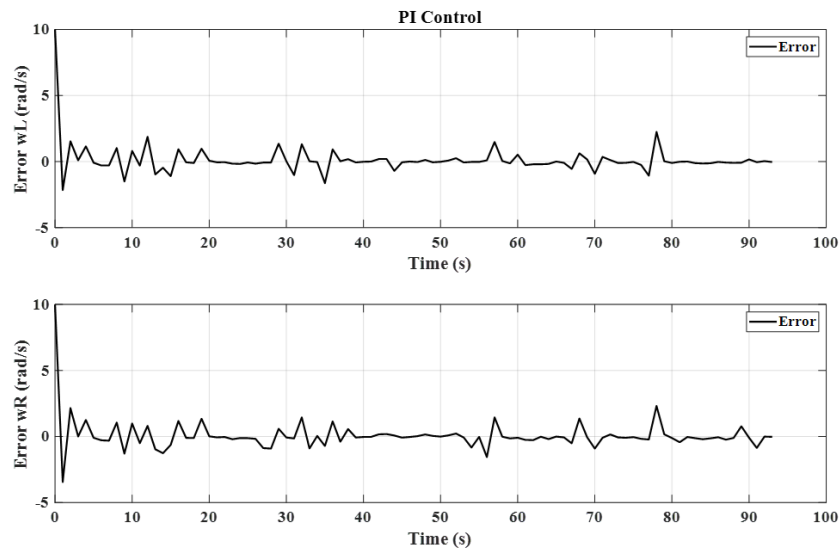


Figure 3.22: Error trends of the left and right wheels under PI control

This indicates effective control and minimal deviation from the desired speed. while in Fig.3.23 is shown the voltage sent to the dc motors.

After confirming the trajectory of the rover along a straight path and maintaining a constant forward speed, a comparison was made between the peak frequency (Tab. 3.1) parameters from the multibody simulation described in chapter 2 and those obtained from the real-time simulation on the prototype. The purpose of this comparison was to validate

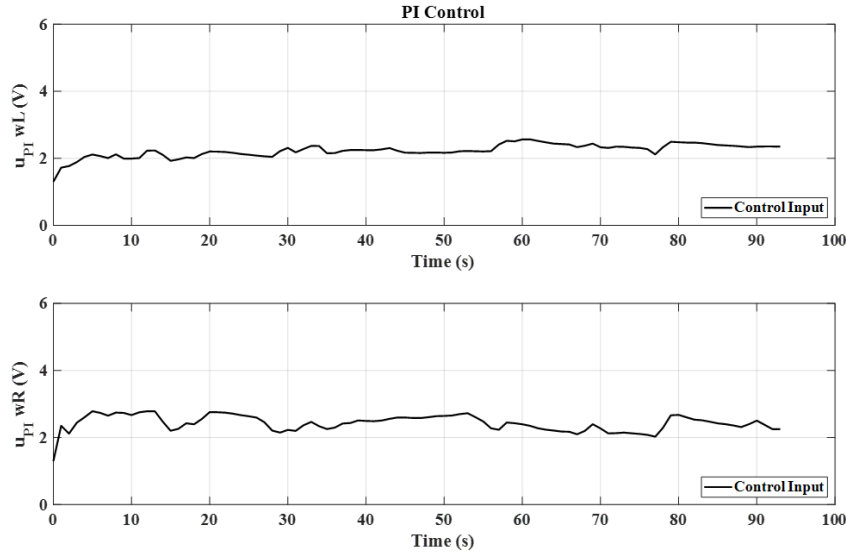


Figure 3.23: Voltage input trends for the left and right wheels under PI control

the consistency and accuracy of the behavior of the prototype relative to the simulated model.

Left Bank	Right Bank
0.00	0.00
40.62	38.05
38.68	34.46
38.30	39.66
39.94	40.06
40.95	39.95
34.62	35.43
38.14	38.41
37.88	37.98
40.90	38.19
39.85	39.25

Table 3.1: Comparison of values between the left and right banks under PI control

The obtained values oscillate around the value of 38.21 Hz obtained earlier from the calculation:

$$10 \text{ rad/s} = \frac{10}{2\pi} \frac{\text{giri}}{\text{s}} = 1.59 \text{ Hz} \cdot 24 = 38.21 \text{ Hz} \quad (3.1)$$

SMC results

The Switching control is defined as:

$$u_{sw} = K \operatorname{sgn}(s) = K \frac{s}{|s| + \epsilon} \quad (3.2)$$

where $K > 0$ is chosen to be sufficiently large. The larger the value of K , the faster the trajectory converges to the sliding surface. The sign function is typically replaced by a smoother version to mitigate the chattering phenomenon (high frequency vibrations that can damage the actuators), where ϵ is a scalar greater than zero but very small.

In order to define the Equivalent Control, it is essential that the derivative of the sliding surface is zero:

$$s = C \cdot e \rightarrow \dot{s} = C \cdot \dot{e} = C \cdot (\dot{\omega}_{ref} - \dot{\omega}) = 0 \quad (3.3)$$

In this context, $e = \omega_{ref} - \omega$ represents the angular velocity error of a single electric motor or wheel group. Once the parameters $a_1 = 0.85$, $a_2 = 2.64$, and $b = 10.6$ have been identified, as described in Section 2.3.1.2, the equivalent control term can be defined.

To derive the equivalent control, start by setting eq.3.3 where s is the sliding surface and C is a constant. From the equation

$$a_1 \cdot \dot{\omega} + a_2 \cdot \omega = b \cdot u \quad (3.4)$$

where u is the control input, we can solve for $\dot{\omega}$ to determine the equivalent control u_{eq} :

$$\dot{\omega} = \frac{b \cdot u - a_2 \cdot \omega}{a_1} \quad (3.5)$$

Setting $\dot{s} = 0$ implies steady-state conditions on the sliding surface, leading to the definition of the equivalent control as:

$$u_{eq} = \frac{a_1 \cdot \dot{\omega}_{ref} + a_2 \cdot \omega}{b} \quad (3.6)$$

This equation provides the necessary control input to maintain the system state along the sliding surface, ensuring the desired dynamic behavior of the motor or wheel group.

When combined with the Switching Control, results in the Sliding Mode control law, expressed as follows:

$$u_{SMC} = \frac{a_1}{b} \dot{\omega} + \frac{a_2}{b} \omega + K \frac{s}{|s| + \epsilon} \quad (3.7)$$

This is the control law that was incorporated into the Arduino code to carry out the required experimental trials. The SMC code was implemented to verify the correct velocity of the rover. The goal was to ensure that all six wheels maintained straight-line motion with a target angular velocity of 10 rad/s. The controller was designed to keep the angular velocity of the wheels close to this target, adjusting as needed to correct deviations and maintain stability. This approach ensured consistent performance even under varying conditions or disturbances.

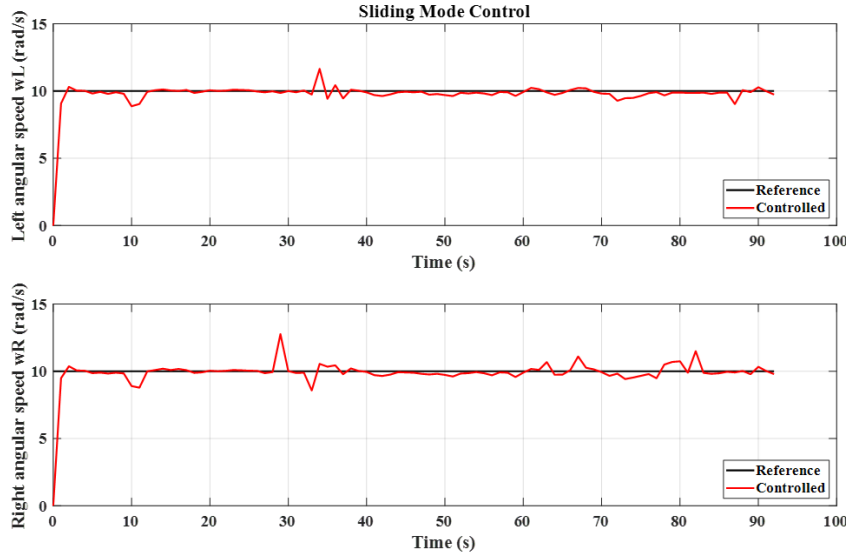


Figure 3.24: Angular velocities of the left and right wheels under Sliding Mode Control, comparing reference and controlled outputs

The result in Fig. 3.24 are obtained:

From Fig. 3.24, following initial transition period, the velocity stabilizes at approximately 10 rad/s. A comparative analysis of the graphs for the two control systems - Sliding Mode Control (SMC) and Proportional-Integral (PI) control - shows that SMC provides a more stable angular velocity response. The SMC curve is more linear and closer to the 10 rad/s setpoint, indicating a higher degree of robustness. This stability highlights the effectiveness of SMC in maintaining consistent performance under varying conditions compared to PI control.

Additionally, the error trend in the SMC system demonstrates a stronger tendency to converge to zero (see Fig. 3.25)

In addition, analysis of Fig. 3.26 showing the evolution of the supply voltage for the sliding mode control also confirms an improvement in performance compared with the PI control, showing more effective and precise management of the control variables.

In addition, a comparative analysis (Tab. 3.2) of the peak frequencies derived from the multibody simulation (discussed in Chapter 2) with those obtained from the real-time simulation on the prototype was performed for the SMC control to further validate the effectiveness of the implemented controller. Looking at the table above, it was again possible to verify that the values obtained fluctuated around the value of 38.21 Hz obtained from the Multibody simulation.

3.2.2.2 Obstacle Crossing Vibration Analysis

During the straight-line rover test, an obstacle was introduced to affect only the wheels on the left bank (see Fig. 3.27). A target speed of 15 rad/s was set for all six wheels to evaluate

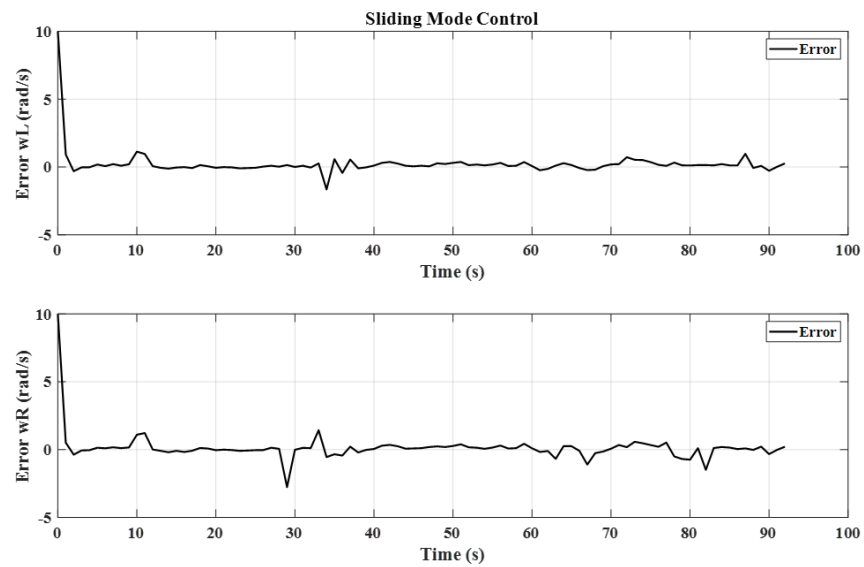


Figure 3.25: Error trends of the left and right wheels under Sliding Mode Control

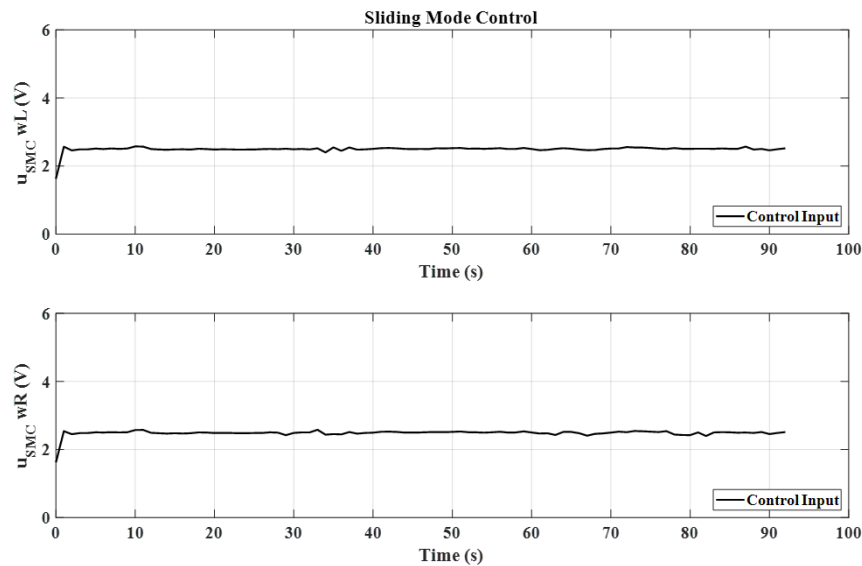


Figure 3.26: Voltage input trends for the left and right wheels under Sliding Mode Control

the ability of the control system to handle terrain irregularities while maintaining the desired motion. This setup allowed an assessment of how effectively the control system could compensate for terrain irregularities and ensure that the rover remained stable and close to the target speed despite the obstacle.

The graph in Fig. 3.28 shows that at about 7 seconds - when the left side of the rover approaches the obstacle - there is a sudden decrease in the speed of the left wheels and a corresponding increase in the speed of the right wheels. This phenomenon is caused

Left bank	Right bank
0.00	0.00
20.35	37.93
37.57	37.28
41.81	39.67
38.30	36.79
40.15	40.87
36.20	35.94
39.82	39.96
38.98	38.73

Table 3.2: Comparison of values between the left and right banks under Sliding Mode Control

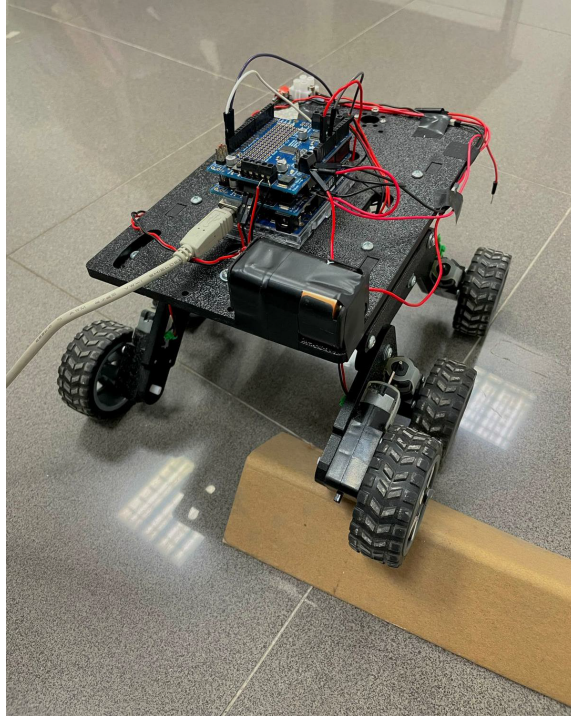


Figure 3.27: Robot demonstrating obstacle crossing on the left bank

by a constraint torque that induces a rotation of the rover around its center of gravity (specifically around the z-axis) upon impact with the obstacle. This dynamic pushes the left wheels backward and the right wheels forward.

This interpretation is further supported by the vibration frequency analysis, which shows a decreasing velocity peak for the left bank and an increasing velocity peak for the right bank, confirming the response of the control system to the induced torque and compensating adjustments in wheel speeds.

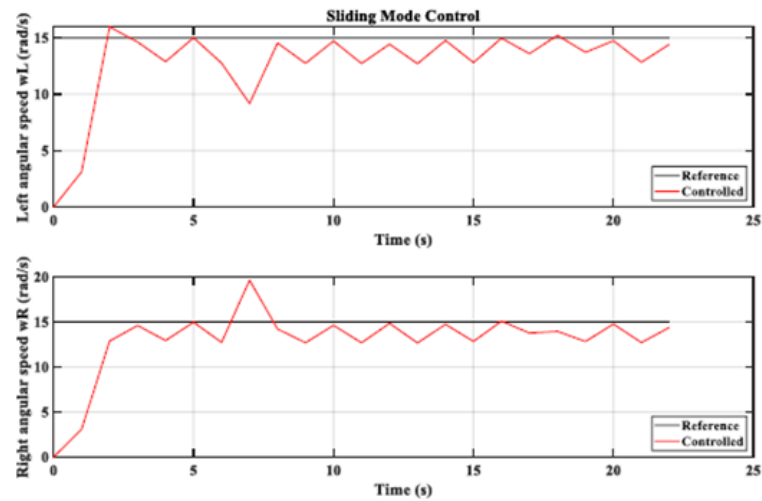


Figure 3.28: Angular velocity trends during obstacle crossing on the left bank under Sliding Mode Control

Analysis of the error and voltage in the presence of an obstacle reveals another dynamic of the control system of the rover. As shown in Fig. 3.29, there is a noticeable increase in error that peaks at the moment the rover encounters the obstacle. This spike in error reflects the challenge the control system faces in maintaining a constant velocity under changing external conditions.

At the same time, the supply voltage trend for the left bank of the rover, as shown in Fig. 3.30, shows a significant drop at the same moment. This drop in voltage occurs because the left wheels, obstructed by the obstacle, require less energy to move. In contrast, the voltage for the right bank remains constant, supporting the increased speed of these wheels. This response partially compensates for the loss of speed on the left side, allowing the rover to continue its forward progress despite the obstacle.

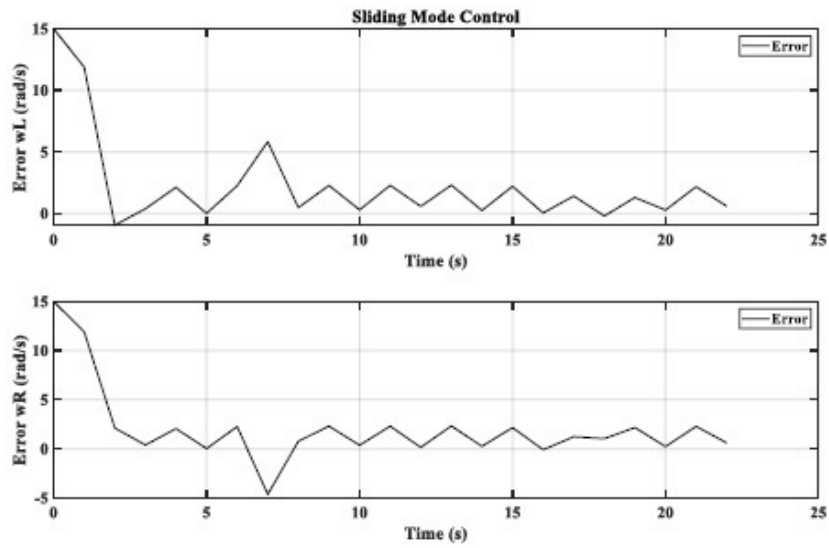


Figure 3.29: Error trends during obstacle crossing on the left bank under Sliding Mode Control

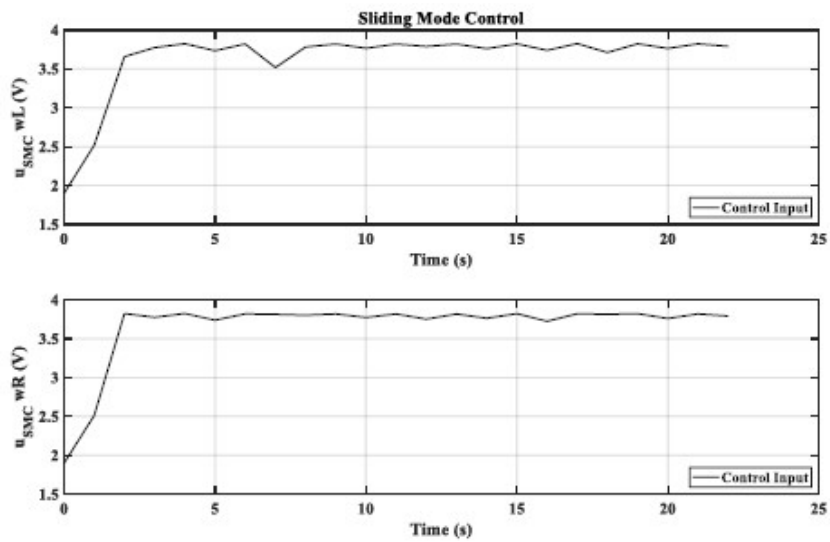


Figure 3.30: Voltage input trends during obstacle crossing on the left bank under Sliding Mode Control

3.2.2.3 Steering Configuration Vibration Analysis

After the straight-line tests, attention shifted to the steering tests. In this six-wheel, unsteered rover model, the turning maneuver is performed using the skid-steer technique, which involves applying different speeds to the wheels on each bank.

During a skid-steer maneuver, the frequencies recorded in the Fourier Transform (FFT) by sensors on the two banks are expected to differ. A higher frequency value is expected for the bank on the side opposite the direction of steering. This means that if the rover is turning right, the left bank - which is moving faster - will have a higher frequency peak on the sensor mounted on the left rocker. Similarly, if the rover turns left, the right bank will have a higher speed and the sensor on the right rocker will detect the higher frequency peak.

These results are expected because the speed difference between the banks induces different levels of stress and vibration, which show up in the FFT analysis as different frequency peaks corresponding to the relative speed of each bank.

The steering maneuver of the rover was performed using Arduino code specifically designed for SMC to manage the speeds of the two wheel banks. During testing, each bank was assigned a different target speed, allowing the system to perform a skid-steer maneuver. This approach effectively simulated steering and allowed the unique dynamics of the vehicle to be studied as the wheel banks operated at different speeds, highlighting the specific behavior of the rover under skid-steer conditions.

setpoint left = 15 rad/s

setpoint right = 5 rad/s

After setting a higher speed for the left bank, the rover naturally turns to the right. To monitor and analyze the performance during this maneuver, Arduino code was developed to acquire the vibration frequency data from both banks.

This data was then transferred to the MATLAB environment for FFT post-processing with the goal of generating FFT plots that clearly illustrate the frequency differences between the two banks-the "fast bank" (left) and the "slow bank" (right). These plots allow visualization of the frequency variations associated with different wheel speeds, providing a detailed analysis of the dynamics of the rover during steering maneuvers. In the diagram shown in Fig.3.31, several frequency peaks are observed. Three significant peaks have been identified for the slow bank. The highest peak, at 36.2205 Hz, actually corresponds to the fast bank, reflecting its higher velocity. However, a common mistake is made when interpreting the second highest peak at 33.0709 Hz; while it may initially appear to belong to the slow bank, its proximity to the primary peak suggests that it is likely to be from the fast bank.

Although the rover approximates a differential vehicle model, each bank consists of multiple wheels. During a maneuver, it is not guaranteed that all wheels in a bank will maintain a uniform speed. Due to slip, some wheels may deviate from their nominal speed, either slowing or speeding up. This effect is particularly evident in the slower bank, where the dominant frequency, being of lower energy, is more influenced by the

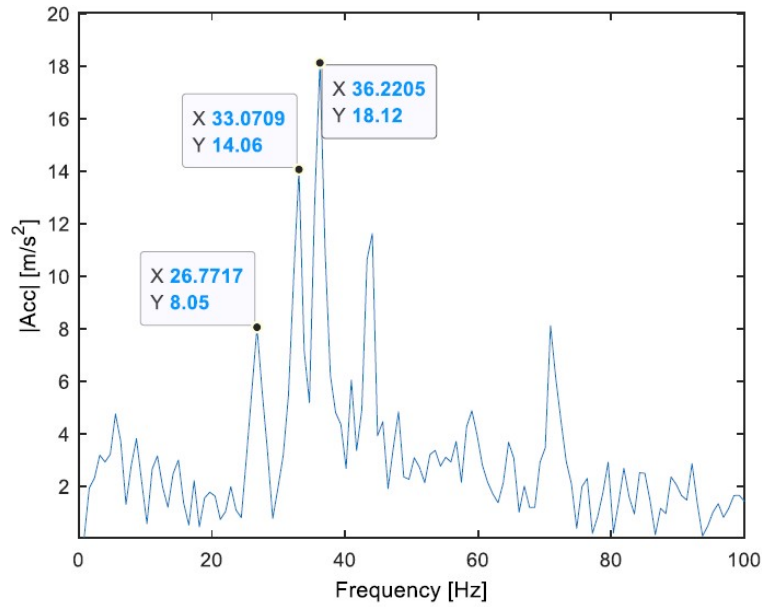


Figure 3.31: FFT analysis of the slow bank

dynamics of the faster bank, resulting in an additional significant frequency spike. This interaction highlights the influence of the dynamics between the banks on the observed frequencies.

An additional condition was added to the Arduino code to reduce confusion regarding the frequency peaks associated with each bank in the FFT plot. This condition identifies a known frequency interval to subtract from the maximum observed frequency. For example, if an interval of about 7 Hz is identified, then the relevant frequency peak for the slow bank is no longer the one immediately below the maximum peak, but rather the peak at least 7 Hz lower—such as at 26.7717 Hz.

This change in the code allows for a clearer distinction of frequency peaks between banks, helping to avoid misinterpretations, especially when peaks are close together. This approach proved beneficial not only in this specific case, but also in other scenarios where the dominant peak corresponded to the slow bank being analyzed. By correctly assigning frequency data to the appropriate bank, this method increases the accuracy of performance analysis during rover steering movements.

When analyzing the FFT plot for the fast bank (see Fig.3.32), a single dominant frequency peak corresponding to the angular velocity of that bank is evident. In this case, the frequency peaks associated with the slow bank are negligible compared to the dominant peak of the fast bank due to the greater kinetic energy and vibration generated by the fast bank during the steering maneuver.

The angular velocity, error, and motor supply voltage data from both banks were then transferred to MATLAB. Further analysis in this environment allowed a detailed comparison of the performance of the two banks. This step allowed quantification of the control effectiveness of the rover and identification of potential areas for improvement,

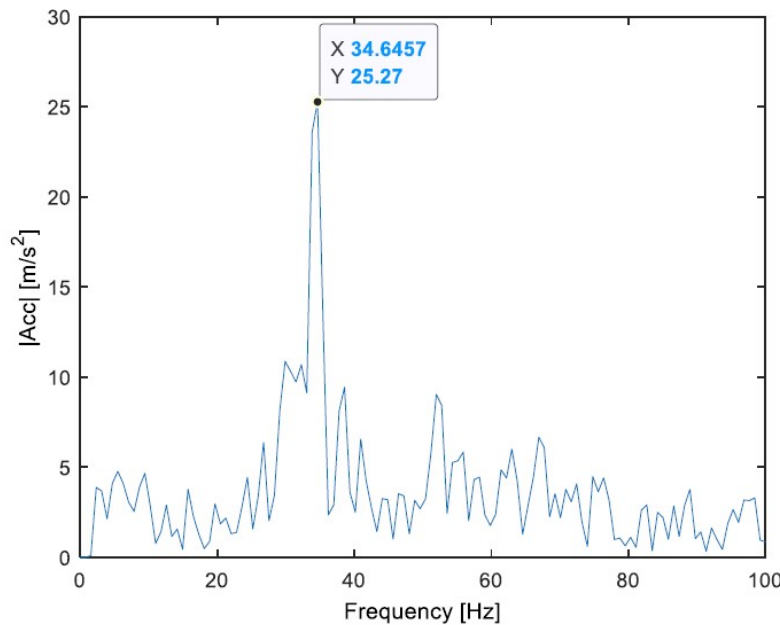


Figure 3.32: FFT analysis of the fast bank

deepening our understanding of the steering dynamics of the vehicle.

The plot in Fig.3.33 shows that neither the fast nor the slow banks reach their respective target velocities of 15 rad/s for the left bank and 5 rad/s for the right bank. This result is consistent with the steering dynamics of the rover, as these target velocities would be achievable under straight-line, constant velocity conditions. However, during steering, drag torques occur that prevent the theoretical speeds from being reached, highlighting a discrepancy between the actual system behavior and the unloaded theoretical model.

During a right turn, the fast bank (left) is held back by the resistance of the slow bank (right), which limits its speed and forces the rover into a curve, making it impossible to reach the preset speed target. As shown in the angular velocity plot, the fast bank stabilizes below the setpoint and oscillates around a nearly constant value of about 12 rad/s.

At the same time, the slow bank exceeds its target speed and stabilizes around 7 rad/s. This dynamic reflects the physical interaction between the banks during the steering maneuver, causing the velocities to deviate from the theoretically expected values due to the influence of interbank forces during the turn.

In Fig. 3.34, the concept of error can be analyzed using the same principles: the error will not cancel out because the wheel speeds do not exactly reach the setpoint values for the reasons previously discussed. The error, defined as $\text{error} = \text{setpoint} - \omega$, will tend to oscillate around a non-zero value.

For the left bank of the rover, the error will remain close to but greater than zero because the actual speed achieved is less than the setpoint, resulting in a positive error. This positive error indicates that the left bank cannot reach the desired speed due to the interaction dynamics with the right bank.

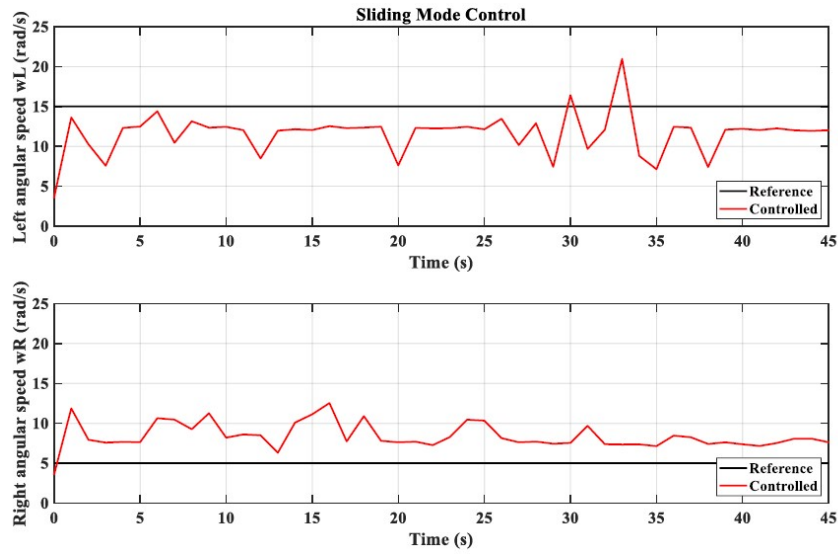


Figure 3.33: Angular velocities of the left and right wheels during the steering test under Sliding Mode Control

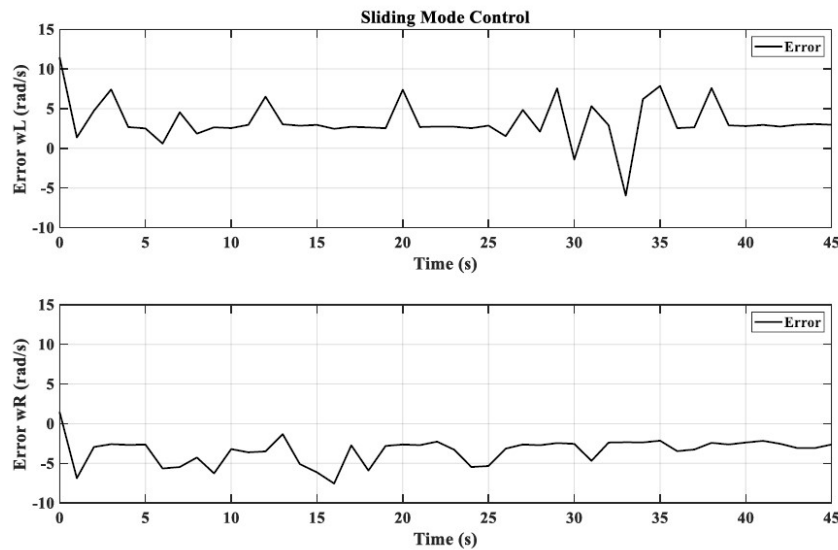


Figure 3.34: Error trends of the left and right wheels during the steering test under Sliding Mode Control

Conversely, the error for the right bank will be negative because the wheels on that bank are moving faster than the set speed, creating a negative difference between the two values. This phenomenon is consistent with the observed behavior, where the right bank compensates for the lower speed of the left bank during the steering maneuver, influencing the error values and reflecting the dynamic response of the vehicle.

To complete the analysis of the behavior of the rover during steering, it is essential to

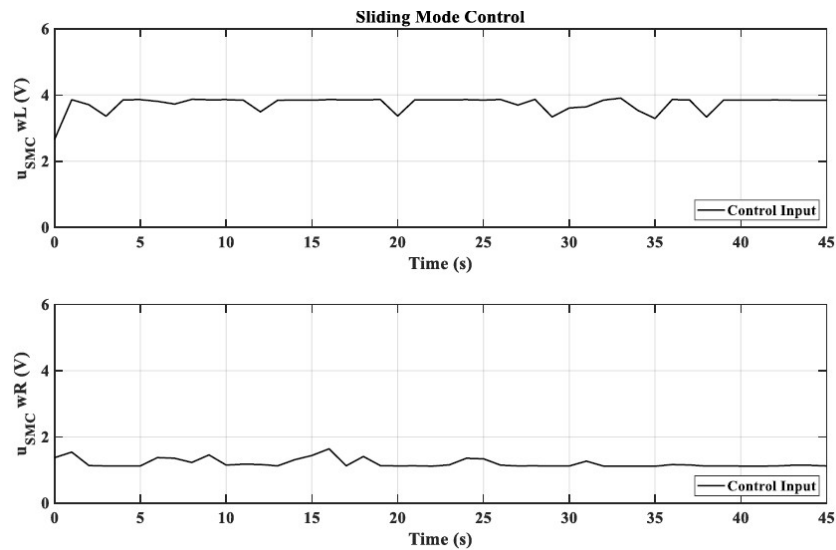


Figure 3.35: Voltage inputs for the left and right wheels during the steering test under Sliding Mode Control

examine the supply voltages of the motors in each bank (see Fig. 3.35). The motors in the fast bank, which need to reach a higher speed, naturally receive a higher input voltage. This increased voltage helps the wheels accelerate to the desired setpoint, which supports the rightward steering action.

In contrast, the right bank, which maintains a lower speed, receives a lower voltage. This difference in voltage reflects the different power requirements of the two banks, with the right bank requiring less power to reach or maintain its assigned speed, which is lower than that of the left bank.

These observations underscore the importance of managing the supply voltages to effectively control the speeds of each bank, allowing for efficient steering and maintaining balance between the two sides of the rover during maneuvers.

3.2.2.4 Conclusions

Following the simulations discussed in chapter 2.3.3, experimental tests were performed using the same principles to verify the accuracy of the modeling results.

For this investigation, the prototype was instrumented with two IMU MPU-6050 sensors mounted on the rocker arms of each side of the rover. These sensors enabled real-time acquisition of accelerometer data, which was used to assess instantaneous wheel speed through frequency analysis. The accelerometer data served as input to an Arduino controller, supported by algorithms developed to precisely adjust the speed of the rover and ensure that a set forward speed was maintained.

Initial experimental tests focused on straight-line motion, followed by steering and obstacle crossing tests. During this study, two types of controllers were implemented and evaluated: a PID controller and a Sliding Mode Control (SMC) controller. A careful characterization of the prototype motors allowed the identification of the parameters needed to calibrate each controller.

Experimental results indicated superior performance of the SMC controller, which was more robust than the PID controller and more effective at keeping the rover on its intended trajectory and at the desired forward speed. These tests provided extensive angular velocity, error, and power voltage data for each bank of the rover, confirming the effectiveness of the chosen methodology.

The use of more advanced hardware with higher processing frequencies represents a potential improvement to these techniques, promising even greater performance in rover control.

3.2.3 Robotic Arm Integration for Enhanced Obstacle Overcoming

The mobile robot of this study is a small rover prototype, Fig. 3.36, composed by six drive wheels, equipped with an actuated robotic arm and mounting a rocker bogie suspension system. It was decided to equip the rover with a passive rocker-bogie suspension since it was not necessary to work at high speed. This type of suspension is suitable for moving at low speed, allowing all wheels to keep contact with the ground on rough terrain or when facing obstacles.



Figure 3.36: Rover prototype equipped with a robotic arm for enhanced obstacle overcoming capabilities

The system architecture, shown in the block diagram in Fig. 3.37, shows a controller at the center of the architecture, in charge of coordinating the actions of the manipulator and the mobile platform. A dedicated MCU (Motor Control Unit) interfaces the motor-wheels with the controller and provides trajectory actuation and execution. The power supply of the prototype is provided by two separate 6V-3500mAh battery packs, which separate the power circuits for the MCU and the motor-wheels connected to it, and the control circuit for the microcontroller and the servomotors connected to it. The separation of the power supplies is designed to reduce electrical interference.

The commercial robotic arm originally had five degrees of freedom (DoFs), as shown in Fig. 3.38, where the blue circles are the actuated DoFs and the red circles are the constrained DoFs. The robotic arm was reconfigured for the specific purposes of the experiment to operate with three DoFs: shoulder α , elbow β and wrist γ . This decision was made in order to significantly simplify the operation of the arm during obstacle crossing tasks.

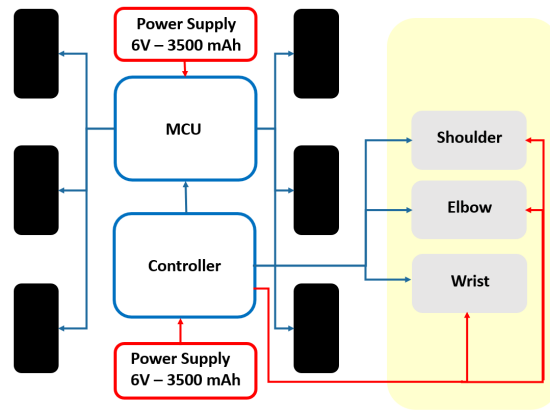


Figure 3.37: Configuration diagram of the prototype for crossing tests

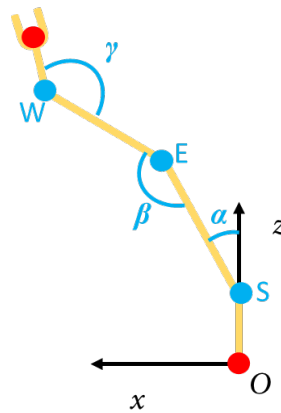


Figure 3.38: Robotic Arm Scheme

3.2.3.1 Command Flow Optimization

The experimental tests follow the same logic as the simulation. The algorithm is composed of several steps as shown in Fig. 3.39.



Figure 3.39: Command flow

The algorithm starts with filtering and sensor fusion, processing data from the 6-axis IMU using a Kalman filter, allowing the three fundamental angles to be obtained: roll,

pitch and yaw. In the experiment, the focus is on the pitch angle as it is important for switching between robotic arm configurations. In addition to low-level control, wheel motors move smoothly from a resting state to a steady state, following a trapezoidal motion profile. There is a delay when power is applied to the motor wheels to ensure stability, and an auto-stop mechanism is triggered when the obstacle is passed. The robotic arm servomotors are controlled using logic similar to that used in the simulation phase. Transition between the two configurations depends on the angle of pitch of the prototype: the first to start climbing and the second to avoid overturning. Initially, the robotic arm is configured to maintain stability, and when a specific pitch threshold is reached, it switches to the second configuration to adapt to the new stability requirement. Finally, data acquisition is carried out using two methods: SD card and serial acquisition. The SD card requires no external cabling, while serial acquisition, which is faster, is preferred during the test.

3.2.3.2 Experimental Tests and Results

This section presents the experimental validation of the small rover prototype, focusing on locomotion, stability, and obstacle crossing. Through a series of structured experiments, data was collected to evaluate the dynamic behavior of the rover under various conditions, including straight-line motion, steering performance, and obstacle crossing.

Kalman Filter

A critical insight arises from an examination of the Kalman filter structure employed in this context. It becomes clear that, in this particular scenario, the filter is not utilized for position estimation. Instead, it is exclusively deployed for signal filtering. This distinction is significant because, in general, the Kalman filter is employed as a state observer, designed to estimate the overall state of a system rather than merely reduce noise in the data. In this instance, however, the objective is solely to filter the data, without the necessity of observing or estimating additional state dynamics.

The Kalman filter is comprised of two principal phases: prediction and state update. In the prediction phase, the filter employs past information and a dynamic model to anticipate the future state of the system. In the update phase, new measurements are incorporated to correct and refine the state estimate, thereby improving prediction accuracy. This iterative process allows the Kalman filter to deliver accurate estimates even when the data are noisy.

In the specific context of this project, however, the filter is used exclusively for signal filtering, without employing a complex dynamic model. In other words, a unified dynamic model is adopted, meaning the prediction step is bypassed, and the signal is corrected solely based on updated measurements. This simplified approach is well-suited to scenarios where the primary goal is noise reduction, without the need to infer future states or engage in complex system dynamics.

The prediction phase is defined as follows:

$$\hat{x}_{k|k-1} = A_k \hat{x}_{k-1|k-1} \quad (3.8)$$

$$P_{k|k-1} = A_k P_{k-1|k-1} A_k^T + Q_k \quad (3.9)$$

In the prediction stage of the Kalman filter, $\hat{x}_{k|k-1}$ represents the estimate of the state at time k based on the information available up to time $k-1$. $P_{k|k-1}$ denotes the covariance of the prediction error, which measures the uncertainty associated with the state estimate. A_k is the state transition matrix, i.e., the process model, while Q_k represents the process noise covariance. In this specific case, since the signal is one-dimensional, no matrices but one-dimensional single values are used. Furthermore, since the objective is not to observe the state but simply to filter the signal, the process matrix A_k takes a unit value. The analysis of the update phase is as follow:

$$K_k = P_{k|k-1} H_k^T \left(H_k P_{k|k-1} H_k^T + R_k \right)^{-1} \quad (3.10)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - H_k \hat{x}_{k|k-1}) \quad (3.11)$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \quad (3.12)$$

In the update phase, K_k represents the Kalman gain, which adjusts the intensity of the state update based on the new observation z_k . The state update is indicated by $\hat{x}_{k|k}$. H_k is the observation matrix, which in this context takes on a unit value, reflecting the use of the Kalman filter primarily for signal filtering rather than state observation. Finally, R_k is the measurement noise covariance, which quantifies the uncertainty associated with new measurements. It is important to note that the values of the covariances, both that of prediction error and that of measurement noise, are generally defined by analyzing experimental data or consulting the technical documentation of the sensor used, in this case an inertial measurement unit (IMU). In this section the experimental results obtained after the assembly of the prototype are presented. Images from the different experiments are shown in parallel with the evolution of the pitch angle. There is a clear distinction between the angle measured directly (i.e. the raw angle) and the angle obtained after applying the Kalman filter. The obstacles used in the experimental tests are made of wood, affecting the coefficient of friction between the wheel rubber and the obstacle, resulting in a higher value than that assumed in the simulations. In fact, the simulations were designed to reflect the most unfavourable conditions. In this test, the pitch angle is defined using raw data from a six-axis IMU integrated into an Arduino board. The reference system adopted for this IMU is defined as follows: the x-axis corresponds to the forward direction, the z-axis is normal to the plane and the y-axis completes the positive orthonormal reference system. Once the three components of acceleration and angular velocity have been obtained, the pitch angle is calculated using a trigonometric relationship based on the accelerations. It is necessary to note that this trigonometric relation is specific to the reference system adopted:

$$\text{Pitch} = \arctan \left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}} \right) \quad (3.13)$$

When the pitch angle has been calculated, it is important to integrate it with the Pitch computed by using the angular velocity. It can be done using a Kalman Filter. This is based on the measurement of the angular velocity, called ω_y , which measures the rotation around the y-axis. The numerical integration of this angular velocity provides an accurate estimate of angular position over time:

$$\text{Pitch}(t) = \text{Pitch}(t - 1) + \omega_y \Delta t \quad (3.14)$$

In this context, the symbol Δt represents the time interval between consecutive measurements. This method of integration requires accurate calibration and precise initial conditions, which in this experiment are consistently close to zero. Subsequently, the calculated angle must be filtered to guarantee a more uniform and realistic variation, eliminating anomalies primarily caused by fluctuations in accelerations. To this end, a Kalman filter was employed, as previously described.

It is important to highlight that the configurations studied in the analytical simulation, as outlined in Chapter 2, are based on a reference system attached to the plate. This should be taken into account before a detailed analysis of the experiments is conducted. In the initial configuration, no significant issues are observed since the plate and floor reference systems can be considered parallel. This assumption is of critical importance, as the floor reference system is used to calculate the required eccentricity to generate the necessary imbalance to overcome the obstacle, as outlined in the kinematic diagrams for the various stages of obstacle traversal in Chapter 2.

In the second configuration, however, the parallel assumption is no longer valid due to the significant inclination of the plate relative to the floor, as observed in both the simulation and the experiment. Accordingly, to ensure accurate alignment of the reference systems, the filtered pitch angle measured by the IMU must be subtracted from each joint angle.

Test 1: Failure crossing test with robotic arm in inert configuration

In order to test the initial hypothesis that the rover is unable to overcome obstacles that are at least as high as the radius of its wheels, an initial experiment was conducted. In this experiment, the wheels were situated in close proximity to a step in order to minimize the effects of inertia forces, as illustrated in Fig. 3.40. In this test, the robotic arm was positioned in a configuration designed to maintain the center of gravity of the rover at essentially unchanged levels. This configuration of the robotic arm is regarded as inert during the obstacle-crossing phases. This allowed us to demonstrate the core of the study. In these conditions, it was possible to quantify the pitch angle of the rover in an equilibrium state (Fig. 3.41), where the filtered angle approaches zero. However, this configuration does not allow the rover to overcome the obstacle, as the center of gravity remains in a position that distributes the load approximately equally across the three axes. This suggests that, with the obstacle in place, the current center of gravity is inadequate for successfully crossing the obstacle.

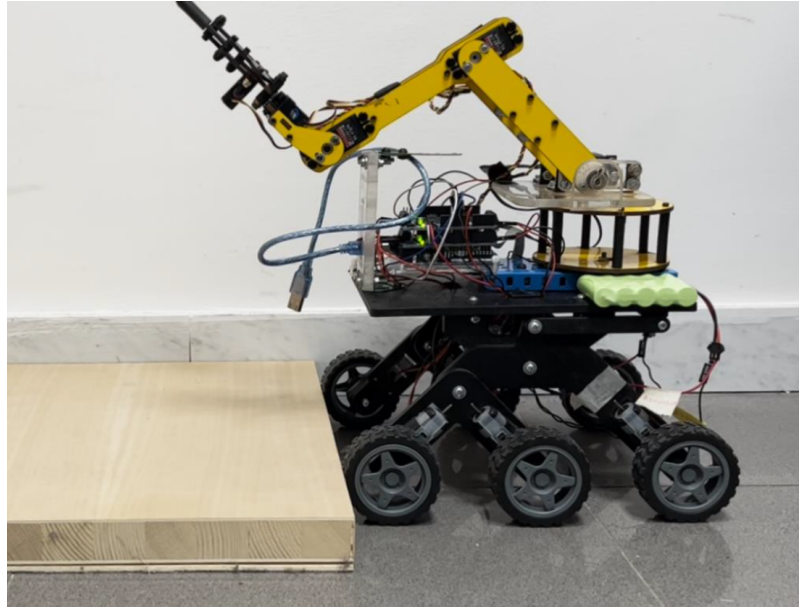


Figure 3.40: Test 1: Rover prototype positioned at the initial stage of the obstacle

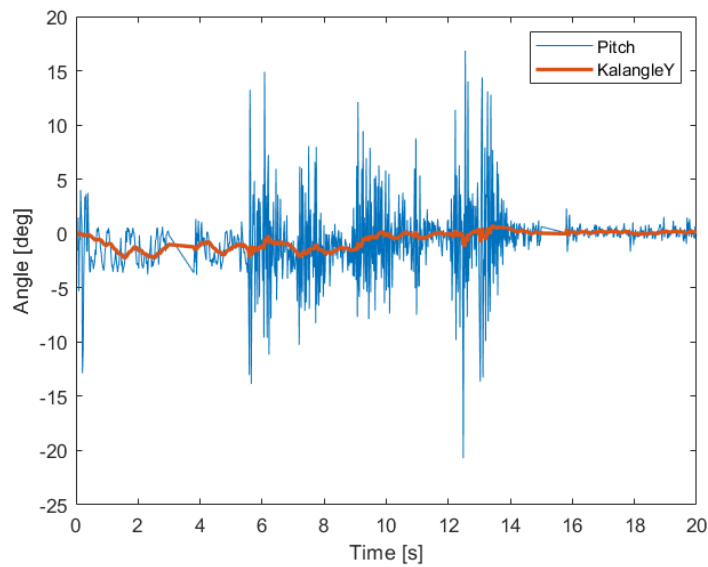


Figure 3.41: Measured pitch angle during the failure crossing test: raw signal (blue line) and Kalman-filtered signal (red line)

Test 2: Obstacle crossing with robotic arm in active configuration, wide obstacle height 3.3 cm

To facilitate the evaluation of the mobility of the rover, a 3 cm obstacle was introduced into the testing environment. This approach simplifies the test conditions, allowing the focus to be entirely on the ability of the rover to cross the obstacle. The performance of

the rover in climbing the obstacle can thus be evaluated with greater accuracy. The stages of the crossing process are illustrated in Fig. 3.42.

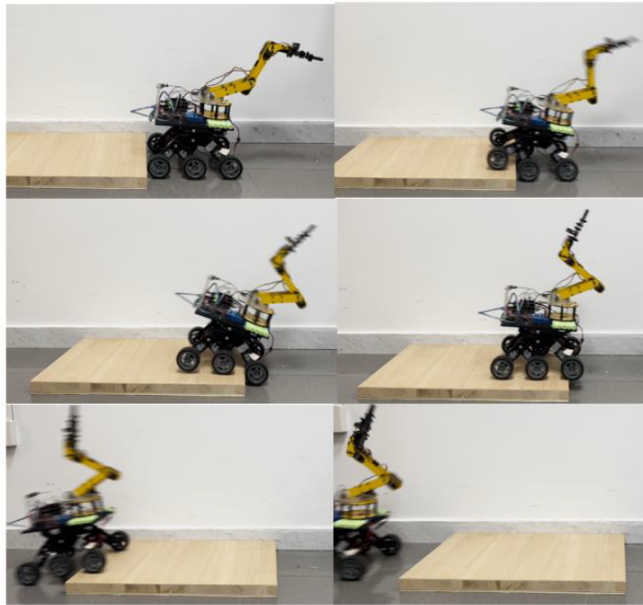


Figure 3.42: Obstacle crossing with a robotic arm in active configuration, wide obstacle: six consecutive frames illustrating the crossing process

This study analyzes the variation in pitch angles, both with and without the Kalman filter. As anticipated, the angles processed with the Kalman filter exhibit a more uniform and less irregular trajectory in comparison to those directly measured by the IMU board. It is important to note that the initial angle at the beginning of the test is not precisely zero, which can be attributed to the tilt caused by the weight of the robotic arm.

This initial tilt, in conjunction with the associated load redistribution, plays a significant role in enabling the obstacle crossing. In the absence of this adjustment, the situation would revert to the previous condition, wherein the obstacle could not be overcome.

By analyzing the graphs of the accelerations on the x- and z-axes, which correspond to the forward axis and the axis normal to the floor, respectively, it becomes evident that the IMU exhibits remarkable sensitivity. The IMU, composed of a series of accelerometers, exhibits two key characteristics, which are highlighted in Fig. 3.44:

- in the graph for acceleration along the x-axis (Fig. 3.44 top), a significant peak is observed at the sixth second. This coincides with the moment when the rover wheel descends from the step. During this phase, although the direction of rotation of the engine remains constant, the contact and impact with the ground cause a marked deceleration of the rover.
- in the acceleration plot along the z-axis (Fig. 3.44 bottom), the ascent and descent phases of the rover are distinctly observed. These phases can be identified by the

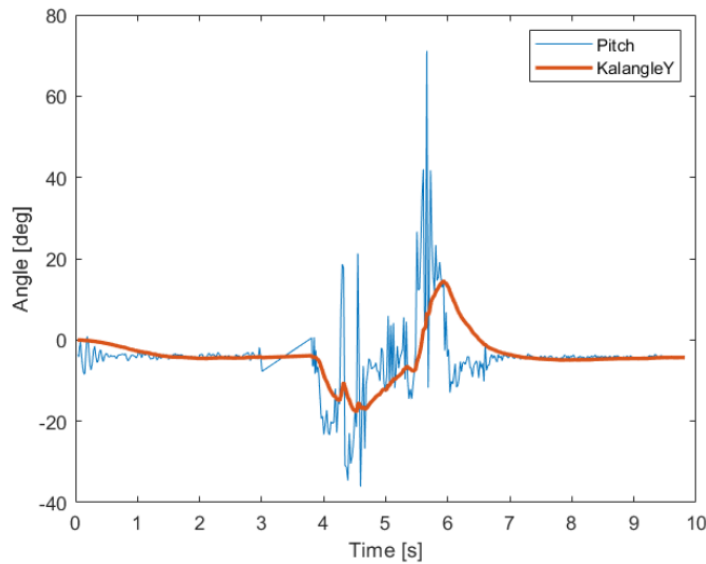


Figure 3.43: Measured pitch angle during obstacle crossing with a robotic arm in active configuration: raw signal (blue line) and Kalman-filtered signal (red line)

time between two consecutive peaks in opposite directions, a phenomenon due to the rover bouncing during these moments.

Such phenomena are commonly observed because the sensor fusion processes used to establish fundamental angles rely on the accelerations measured in the planes where the rotations occur.

Test 3: Obstacle crossing with robotic arm in active configuration, short obstacle height 3.3 cm

Following confirmation that the rover, with the assistance of the robotic arm, is capable of crossing the step employed in the previous test, a smaller obstacle was introduced. This allows for the analysis of not only the ascent phase but also the descent phase of the rover. Frames related to this test are presented below in Fig. 3.45 By analyzing the graph in Fig. 3.46 and focusing only on the filtered pitch angle, several conclusions can be drawn. Firstly, the pitch angle remains close to zero for the greater part of the time, indicating that the rover-arm system maintains relative stability, at least when at rest. A notable shift in pitch angle is observed between 3 and 7 seconds, reaching a maximum of approximately 60 degrees and subsequently declining to approximately -20 degrees. This marked shift, signifying an instantaneous upward tilt followed by a swift descent, correlates with the instance when the system crosses the obstacle, exhibiting a motion analogous to a "jump." Following the overcoming of the obstacle, the pitch angle rapidly reverts to its baseline state, stabilizing at a value close to zero. This rapid return to a stable equilibrium state after the dynamic event is of great significance, as it serves to

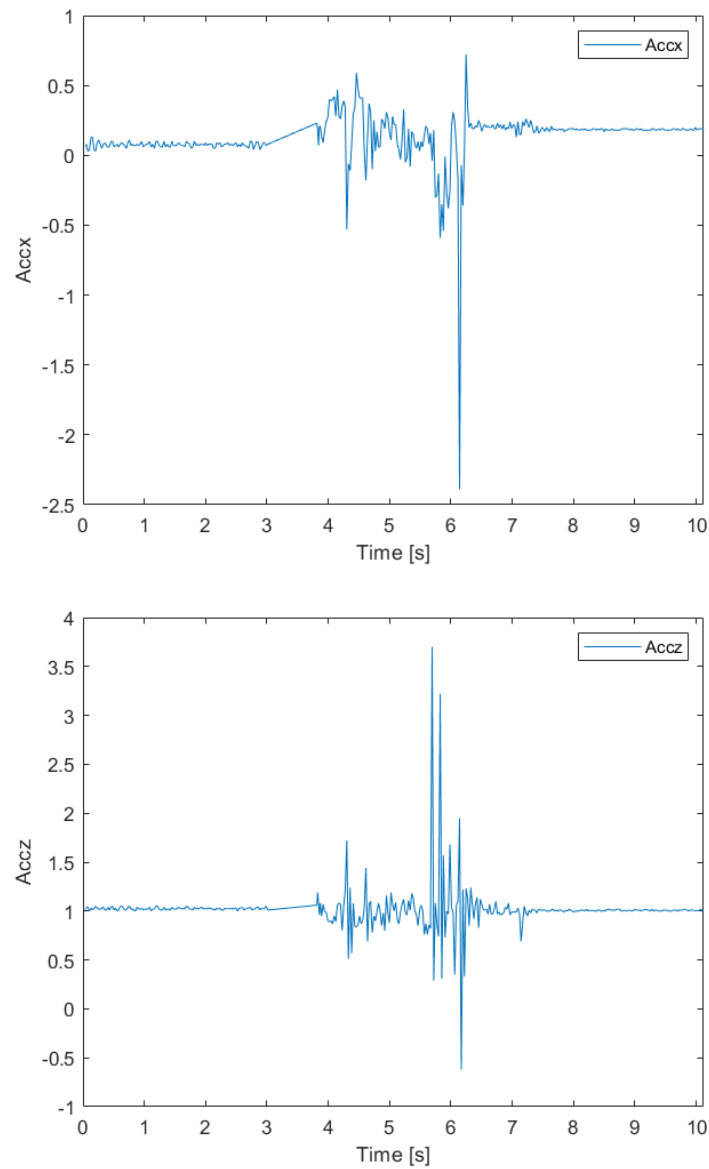


Figure 3.44: Measured accelerations during obstacle crossing along (a) x-axis and (b) z-axis

confirm the efficacy of the system in maintaining overall stability even in the wake of considerable movement.

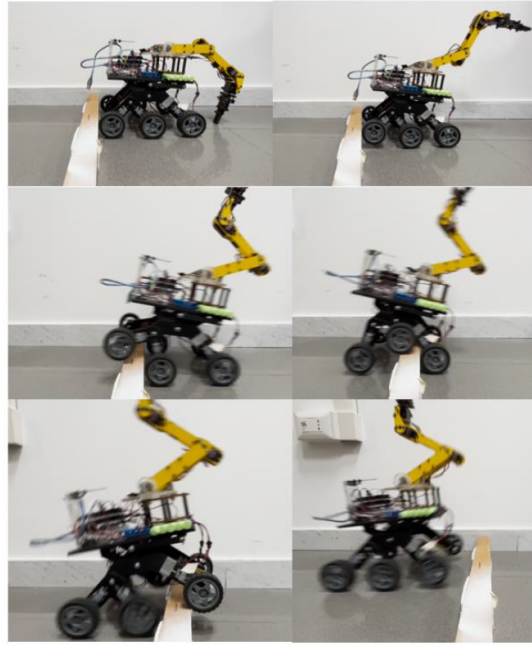


Figure 3.45: Obstacle crossing with a robotic arm in active configuration, short obstacle: six consecutive frames illustrating the crossing process

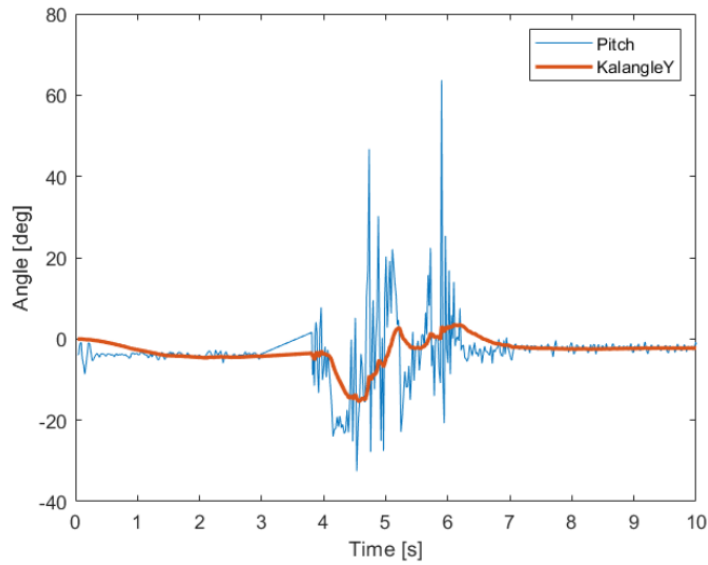


Figure 3.46: Measured pitch angle during obstacle crossing with a robotic arm in active configuration, short obstacle: raw signal (blue line) and Kalman-filtered signal (red line)

Test 4: Rover tilting during crossing stages

In order to illustrate the significance of the Kalman filter in these experiments, it was essential to evaluate the discrepancies in the behavior of the rover when the algorithm employs unfiltered versus filtered angle values. The relevant frames from this test are presented in Fig. 3.47. During the test, it was observed that the raw pitch angle values were

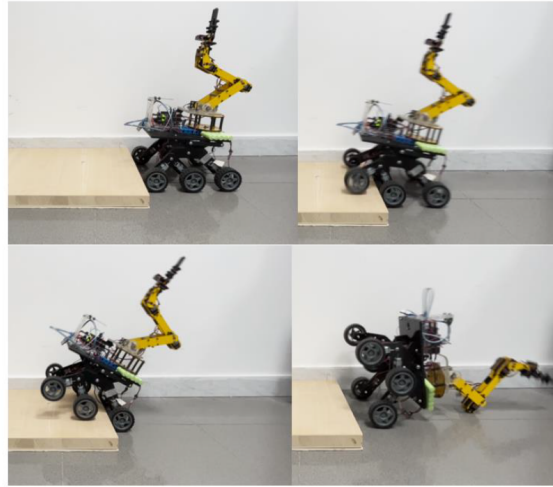


Figure 3.47: Rover tilting during crossing stages: four consecutive frames.

significantly affected by noise, resulting in values that exceeded the threshold required for transitioning to the second configuration of the robotic arm. Consequently, the arm was already in this configuration at the beginning of the test, preventing the rover from successfully crossing the obstacle and even causing it to roll over. This situation highlights the crucial role of data filtering techniques, such as the Kalman filter, in mitigating the impact of data noise.

3.2.3.3 Conclusions

In conclusion, the testing of the rover system equipped with the Lynx 6 robotic arm resulted in the generation of significant data. The tests demonstrated the effectiveness of the arm in overcoming various obstacles, thereby underscoring the importance of the selected mechanical and electronic components, as well as the precision of the algorithms employed. Moreover, the system demonstrated remarkable proficiency in managing considerable variations in pitch angle, thereby confirming its stability and control during obstacle traversal. The rapid return to horizontal stability after overcoming obstacles is indicative of the efficacy of the dynamic design and the management strategy implemented.

The simulations conducted in the previous chapters provided a robust basis for the subsequent experimental stages, validating the proposed theories and refining the system management processes. The integration of a Kalman filter-based signal filter into the sensor fusion process resulted in a significant enhancement in the accuracy and efficiency of the system. In conclusion, this study advanced the technical understanding of robotic systems in complex scenarios and established a foundation for further research and development, thereby opening new possibilities for future applications.

3.3 Full-Scale Rover Prototype

The design and realisation of the full-scale prototype is a crucial step in the development of an advanced robotic system. This process allows design ideas and theoretical simulations to be translated into a concrete physical structure, and represents the intermediate step necessary to validate and refine the design before experimental testing. The main objective was to create a complete system that faithfully reflects the technical requirements defined in the design phase, with particular attention paid to construction efficiency and operational functionality.

The work was carried out using rapid prototyping techniques, such as 3D printing, which made it possible to produce components with complex and highly customised geometries. These techniques, combined with the choice of materials such as PLA and ABS, made it possible to combine lightness and mechanical strength, essential factors to ensure the mobility and robustness of the system. The structure has been designed to integrate perfectly with a rocking bogie suspension, a solution chosen for its proven ability to adapt to uneven terrain while maintaining constant contact with the ground.

In parallel, the integration of motors, lithium-ion batteries and an advanced control system required careful configuration to ensure the right balance between performance, autonomy and reliability. Each component was selected with the intention of supporting the rover in complex operational scenarios, with a particular focus on modularity and system scalability for future upgrades.

3.3.1 First Prototyping

The following section describes the initial prototyping phase of the full-scale rover, focusing on the transition from digital design to physical realisation. The first prototype was developed using 3D printed components and rapid prototyping techniques to allow early testing and design validation before finalising the structure.

3.3.1.1 3D Printed Components

The majority of the components that comprise the rover were fabricated using the material extrusion process. This entailed the printing of all components using Polylactic Acid (PLA), which is one of the most popular materials for desktop 3D printing. PLA is a material that is widely preferred due to its ease of use, low cost, and versatility. The printing process occurs at relatively low temperatures and does not require the use of a heated bed, which makes it the default filament for the majority of extrusion-based 3D printers.

In addition to its practical applications, PLA is also one of the most environmentally friendly filaments currently available. Derived from renewable resources such as corn and sugarcane, PLA is biodegradable and emits a pleasant, sweet aroma during printing, making it a sustainable and user-friendly choice.

For the fabrication of larger components, a hexagonal infill pattern was employed. This pattern is renowned for its optimal strength-to-weight ratio, a consequence of its tessellation of hexagons, rendering it an optimal choice for robust yet lightweight components. However, it is also one of the most time-consuming infill types to print, due to the frequent changes in direction of the printhead.

The smaller components were printed with a grid infill pattern, which comprises two-dimensional intersecting lines on each layer. This pattern offers an adequate level of strength for smaller components, requires a moderate amount of material, and results in a shorter printing time than the hexagonal pattern.

The selected infill density was dependent upon the dimensions and designated function of the components. For smaller components, a density of 50–70% was selected, as alterations in density did not markedly affect printing time. For larger elements, a density of 20% was chosen to minimize printing time, as incremental increases in density would result in significantly longer print times.

The Fig. 3.48, 3.49 and 3.50 illustrate a range of infill patterns with varying densities for a single component, as previewed using the IdeaMaker software.

In Fig.3.51, 3.52 and 3.53, instead, some of the printed components are shown. Some of them have already been equipped with bearings, threaded bushing etc...

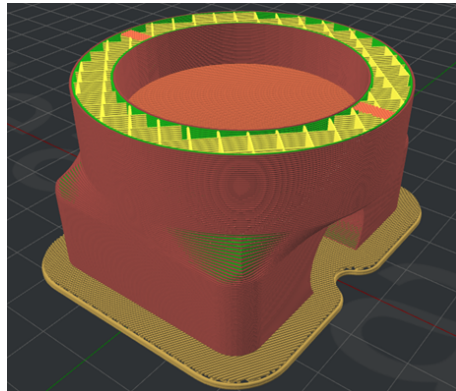


Figure 3.48: 3D printed model with a grid infill pattern at 20% density

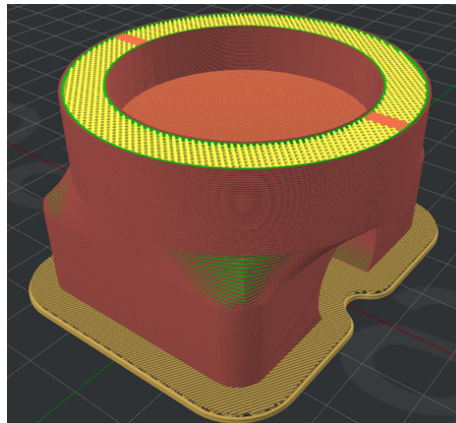


Figure 3.49: 3D printed model with a grid infill pattern at 50% density

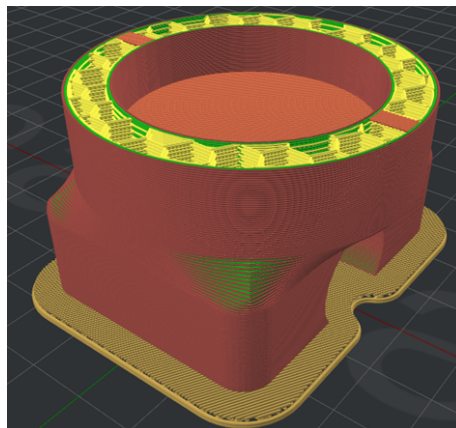


Figure 3.50: 3D printed model with a hexagonal infill pattern at 20% density

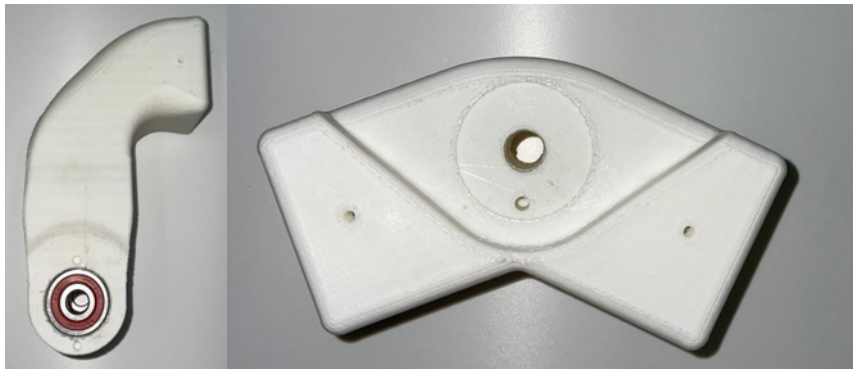


Figure 3.51: (left) Upper bogie joint and (right) lower bogie joint used in the suspension system



Figure 3.52: Body components – Rocker bogie link (left) and central joint (right)

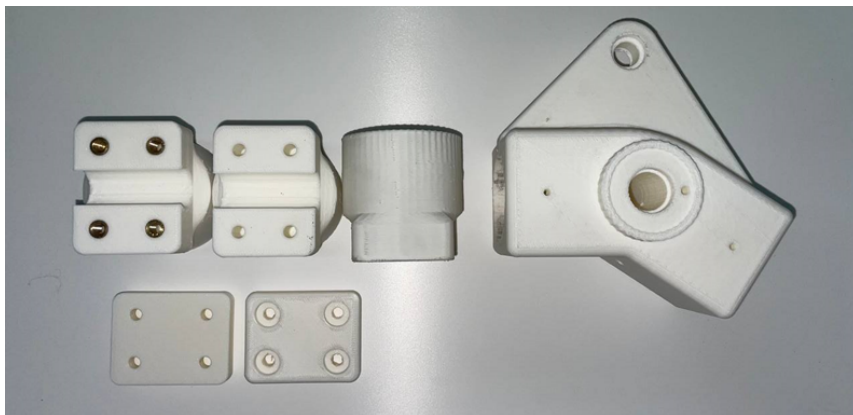


Figure 3.53: Wheel attachments and rocker joint

3.3.1.2 Characteristics of PLA Material

Poly(lactic acid) (PLA) is a thermoplastic polymer belonging to the aliphatic polyester family. It is derived from renewable resources, including maize starch, sugar cane, and beet. The chemical structure of PLA comprises repeating units of lactic acid, a chiral monomer that exists in two enantiomeric forms: L-lactic and D-lactic. The polymerization of these monomers results in the formation of polymers with disparate physical and mechanical properties, contingent on the enantiomeric composition. The main characteristics of the PLA that makes it suitable for 3D printing are:

- One of the principal characteristics of PLA is its biodegradability. In industrial composting conditions, PLA is capable of undergoing degradation into carbon dioxide, water, and biomass within a timeframe of approximately six months to two years. Nevertheless, in natural environments, such as soil or seawater, the degradation process may take considerably longer.
- PLA exhibits comparable tensile strength and stiffness to polystyrene. Nevertheless, its intrinsic lack of ductility results in a low elongation at break, which in turn restricts the potential applications of PLA in situations where high ductility is required.
- The glass transition temperature (T_g) of PLA is approximately 60°C, while its melting temperature varies between 130°C and 180°C, contingent on the degree of crystallinity and enantiomeric composition. The aforementioned thermal properties restrict the utilisation of PLA in applications that necessitate high-temperature resistance.
- PLA can be readily processed using standard thermoplastic processing techniques, including extrusion, injection moulding and thermoforming. The low melting point of the material permits processing at temperatures below those required for other polymers, reducing the energy consumption associated with the manufacturing process.

In the field of three-dimensional printing, PLA is one of the most commonly utilised materials, particularly in the context of fused deposition modelling technologies. The popularity of PLA can be attributed to a number of factors: easy to print, bonding of the printing plan, several colour and finish and low odour emission (89; 90)

3.3.1.3 Assembly Procedures

Once all the components have been printed, the phase of assembly started. In Fig. 3.54, 3.55, 3.56, 3.57 and 3.58 some subsystems and the final structure are shown.

The prototyping phase was successfully completed using 3D printing technology, specifically the Raise 3D Pro 2 Plus printer and the IdeaMaker slicing software. The combination of hardware and software provided the necessary precision and flexibility to



Figure 3.54: Bogie Joint subsystem

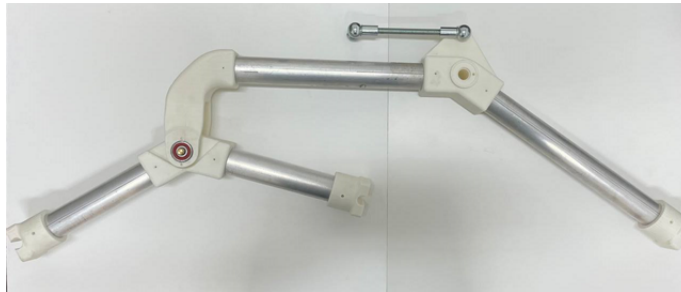


Figure 3.55: Rocker-Bogie subsystem

create the complex components of the rover, while enabling iterative design improvements throughout the development process.

One of the principal benefits of employing 3D printing in this phase was the simplicity with which the parameters of the system could be modified. The flexibility of the modelling and printing setup allowed for the rapid modification of the design, ensuring that adjustments could be made in a smooth and coherent manner in response to evolving requirements or test results. This adaptability allowed for the refinement of specific features and the enhancement of certain structural or functional aspects of the rover.

Another significant advantage of this methodology was the capacity to reprint and replace specific components when necessary. By simply updating the design and reprinting, it became possible to modify the structure in a straightforward and efficient manner, thus facilitating the straightforward and efficient modification of the structure. This capability not only reduced the time and cost typically associated with prototyping but also facilitated the exploration of multiple design iterations without the need for complex

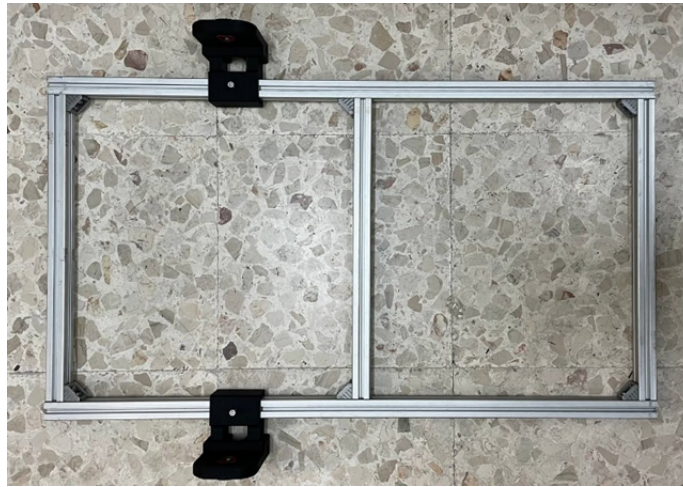


Figure 3.56: Body structure



Figure 3.57: Rover structure: isometric view



Figure 3.58: Rover structure: side view

or resource-intensive manufacturing processes.

The combination of the Raise 3D Pro 2 Plus printer and IdeaMaker software proved

instrumental in achieving a high degree of precision in the printed parts, while the flexibility of the process ensured that the rover's prototype could evolve dynamically during development.

3.3.2 Second Prototyping

In the previous section, the process of prototyping the initial rover model was outlined. This section will provide a comprehensive overview of the techniques employed in the construction of the second prototype, with particular attention paid to the components that comprise the prototype, the assembly process, and the modifications introduced in comparison to the initial model. One of the principal elements of this phase is the incorporation of design alterations. As discussed in Chapter 2, these changes resulted in significant enhancements at the structural level, conferring upon the rover enhanced robustness and stability. The redesign of the joints and optimisation of mass distribution were among the most significant changes, resulting in a more efficient and versatile model than its predecessor. A further crucial modification was the selection of materials. In the initial prototype, the joints were constructed from PLA, a material that is well-suited to 3D printing due to its low cost and ease of use, but which has inherent limitations in terms of mechanical strength and durability. In the second prototype, the joints were manufactured using ABS. ABS offers greater impact resistance and a superior ability to withstand prolonged stresses in comparison to PLA, rendering it an optimal selection for the critical components of the rover.

In addition to the modification of the materials used, the techniques employed in 3D printing were also subject to significant refinement, particularly in relation to the manner of filling the printed structure. An optimal infill pattern and density were selected for each component in order to achieve an equilibrium between mechanical strength, weight and printing time. These enhancements led to the creation of more robust components without compromising the overall lightness of the rover. In summary, the modifications implemented at this stage not only enhanced the structural and functional capabilities of the rover, but also simplified the assembly process, increasing the robustness and reliability of the new prototype for the intended applications.

3.3.2.1 Advanced Prototyping Techniques

Following the implementation of the proposed design modifications and the subsequent verification of their efficacy through comprehensive simulations, the prototype realisation phase commenced. This phase represents a crucial step in the development process, as it allows the theoretical improvements and design solutions to be translated into a tangible physical model. The construction of the prototype necessitated a methodical approach, commencing with the selection of the most appropriate materials and manufacturing techniques for each component. Design modifications were implemented during the 3D printing and assembly phase, including the redesign of joints, optimisation of the frame and the adoption of new materials, such as ABS. Indeed, all joints have been manufactured through the process of additive manufacturing, a modern manufacturing method that employs a diverse range of materials to create a variety of products with considerable flexibility, while reducing the reliance on carbon, a characteristic of traditional foundry work. As previously stated, additive technologies facilitate the production of artefacts through the adduction of material in successive layers, thereby enabling the creation of

free-form and, when necessary, highly complex objects. The Raise3D Pro 2 Plus printer was employed to fabricate the rover components, while the slicing software utilized was IdeaMaker, as previously described. The selected printing material was ABS, a thermoplastic polymer commonly utilized in the manufacturing industry due to its resistance to impact, thermal stability, and capacity to withstand prolonged stress, rendering it an optimal choice for components subjected to mechanical stress. The components were produced using optimised parameters, including fill density and infill pattern, in order to achieve an optimal balance between structural strength and lightness. The most critical components, such as joints and supports, were subjected to rigorous quality control procedures to confirm their compliance with the specified design requirements. The assembly of the prototype demanded particular care and attention to ensure the correct integration of the various subsystems, including the motor-wheel system, the frame structure and the suspension system. The documentation of each stage was undertaken in order to ensure the traceability of changes and to facilitate any adjustments that might be required during the course of construction.

This phase enabled the feasibility of the proposed design solutions to be verified, while also facilitating the identification of any critical issues or potential areas for improvement, which could then be addressed in subsequent iteration cycles.

3.3.2.2 ABS Material Characteristics

Acrylonitrile butadiene styrene (ABS) is a thermoplastic polymer that is widely used in a variety of industrial contexts due to its excellent mechanical and thermal properties. It is a copolymer, obtained from the combination of three monomers: acrylonitrile, butadiene and styrene. Each of these monomers contributes to the specific properties of the material.

The structure of ABS is amorphous, comprising a continuous phase of acrylonitrile-styrene (SAN) and a dispersed phase of butadiene rubber. The latter can impart a number of properties to the material, including:

- ABS is renowned for its high impact resistance and resilience, which are maintained even at low temperatures as low as -40°C .
- The material exhibits good heat resistance, with a glass transition temperature (T_g) of approximately 105°C , rendering it suitable for applications that necessitate thermal stability.
- ABS exhibits robust resistance to a multitude of acids, provided they are not of a particularly strong nature. It also demonstrates considerable resilience to a wide range of inorganic salts, alcohols and hydrocarbons. Nevertheless, it shows poor resistance to aldehydes, ketones, esters and certain chlorinated hydrocarbons.
- The material is readily processed by a range of techniques, including injection moulding, extrusion and thermoforming. Furthermore, it can be affixed by means of adhesives, welding, and screw fasteners with self-tapping capabilities.

ABS is a versatile and durable material that is employed in a multitude of industries and in 3D printing. The excellent mechanical and thermal properties of the material make it an ideal choice for applications that require durability and stability. Nevertheless, its non-biodegradable nature gives rise to environmental concerns pertaining to its disposal and recycling (91; 92)

3.3.2.3 3D Printing Techniques for ABS

Available in filament form with a diameter of 1.75 mm or 2.85 mm and in a wide range of colours, ABS exhibits characteristics that render it an ideal material for use in a multitude of industries that employ 3D printing for the fabrication of functional components or prototypes. With a melt temperature of approximately 200°C, ABS necessitates an extrusion temperature of between 230°C and 260°C to guarantee optimal print quality. Furthermore, it is imperative to utilise a heated platen, with temperatures between 80°C and 130°C, as ABS exhibits a pronounced tendency to shrink rapidly upon contact with air. This phenomenon can result in warping, which manifests as the detachment of the part from the print bed (Fig. 3.59). In order to mitigate these issues and enhance the quality of the printed output, it is advised that a 3D printer with a closed chamber be utilized. This not only assists in maintaining a consistent temperature throughout the process, but it also minimizes the exposure of the user to particles emitted by the ABS, which could potentially be harmful. It is important to note that temperature control is a crucial

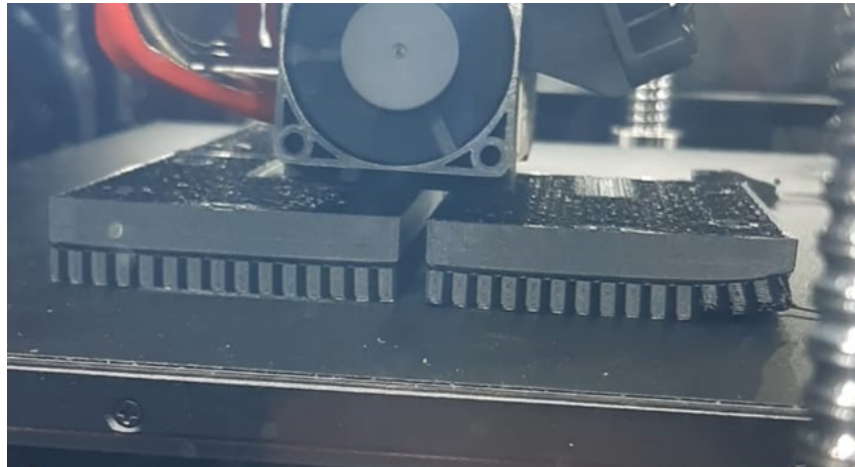


Figure 3.59: Warping phenomenon observed during 3D printing

aspect of the printing process, as fluctuations in temperature can lead to errors in the printed material. This is particularly relevant in the context of the chamber, the printing table, and the room in general. Furthermore, it is important to note that ABS is a material that must be kept in a dry environment, as it absorbs moisture from the air, which can significantly impair the printing process. The thermoplastic polymer in question lends itself to printing with minimal post-processing, which is a significant advantage in the

production of complex parts. The material can be painted and exhibits durability, with minimal deformation observed.

3.3.2.4 Prototyping of Joints and Connections

Once the CAD of the components had been completed, the next stage was to proceed with 3D printing. Specifically, the process was configured such that the workpiece table remained stationary, while the nozzle was programmed to move and deposit the layers. Additionally, the temperatures of the print table were set to 105°, while the nozzle temperature was set to 255°. From the control panel (Fig. 3.60) of the printer, the workpiece can be selected and the printing conditions set. The printer requires a brief period of

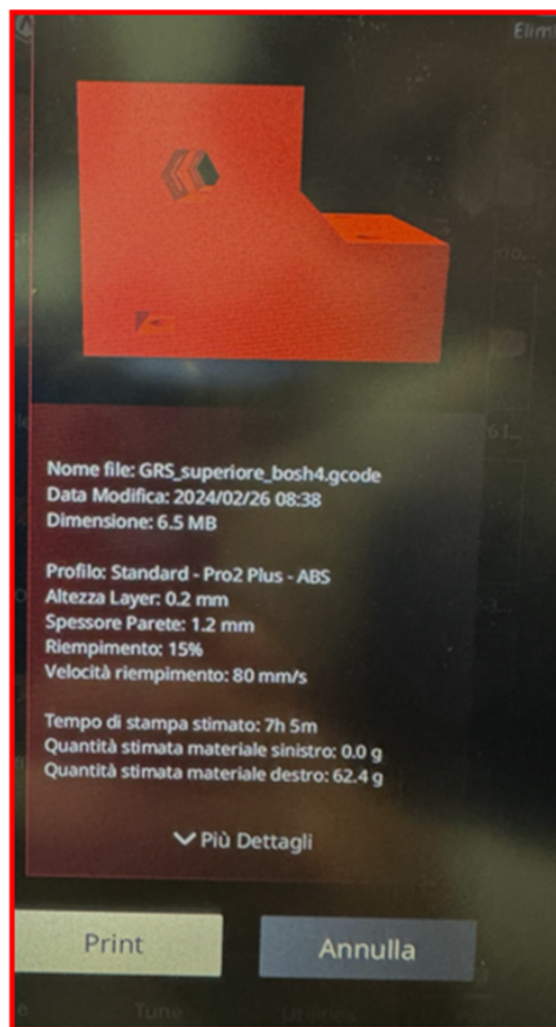


Figure 3.60: Control Panel of the 3D printer

time, during which it reaches the pre-set temperature conditions, before the deposition of material layers can commence. This period is crucial for ensuring the correct adhesion

of the material to the print bed and for preventing any issues during the initial stages of processing. The preliminary step of creating a support structure before commencing the printing of the selected joint fulfils two key functions. Firstly, it ensures the removal of the component from the printing table can be achieved without compromising its shape or causing damage. Then, the system allows for controlled cooling, which prevents the occurrence of deformations that could compromise the integrity of the primary component. During the printing process, the progress and remaining time can be monitored directly from the printer's control panel (Fig. 3.61). This real-time control enables the user to verify the smooth operation of the process and to intervene in case of any issues. Although the

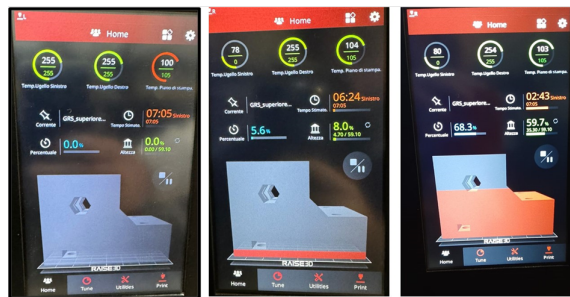


Figure 3.61: Printing progress

printer offers the possibility of printing several parts at the same time, this mode was only used for small parts. For larger parts, single printing was preferred. This choice was motivated by the need to ensure greater control over the process and to achieve higher precision, avoiding any deformations or problems that might occur when printing larger objects simultaneously.

3.3.2.5 Final Assembly Steps

Once all the requisite components had been obtained, the rover was assembled in the following sequence: first, the suspension systems (Fig. 3.62) and then the central body plate (Fig. 3.63). Subsequently, the entire rover was assembled (see Fig. 3.64, 3.65, 3.66). In particular, the wheels were initially excluded from the assembly process in order to assess the proper functioning of the differential system.

Subsequently, the wheels were equipped with the needed supports, as illustrated in Fig. 3.67.



Figure 3.62: Rocker-Bogie Suspension for the new prototype

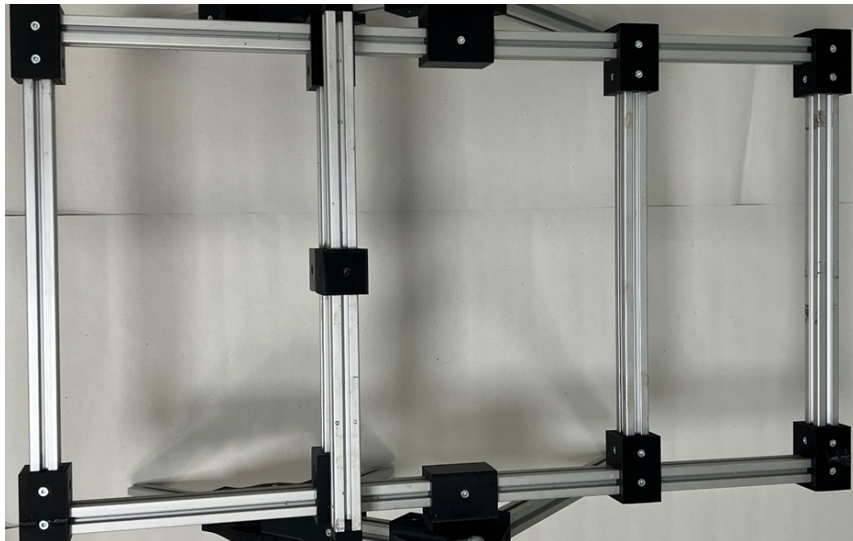


Figure 3.63: Central body plate built from Bosch profiles and ABS corners



Figure 3.64: Isometric view of the Rover



Figure 3.65: Side view of the Rover

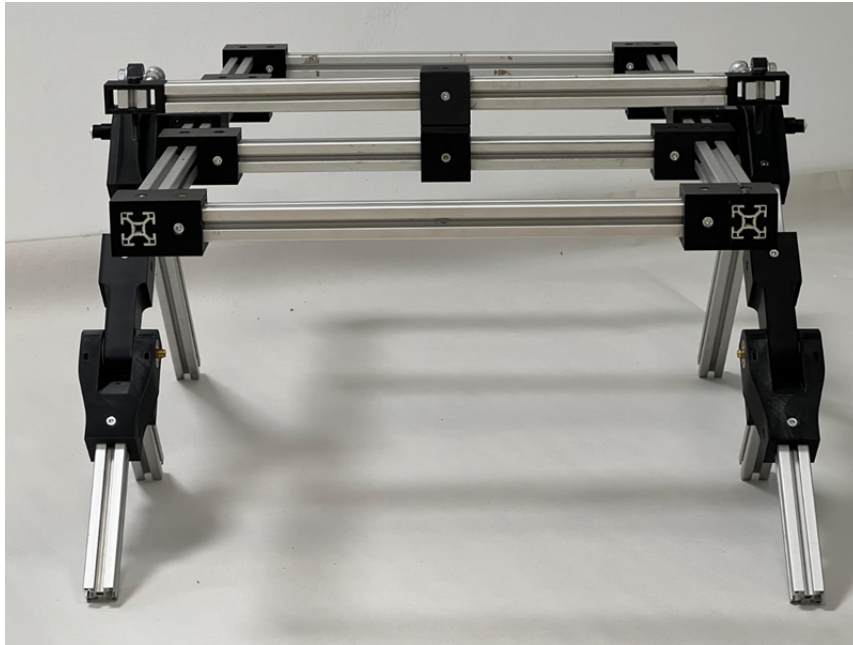


Figure 3.66: Frontal view of the Rover



Figure 3.67: Last rover prototype

3.3.3 Conclusions

The realization of the full-scale rover prototype represents an important step towards the validation of the project and provides a tangible platform for the subsequent phases of analysis and experimentation. Although no operational tests have yet been carried out, the work carried out so far has successfully addressed the technical complexities involved in building an advanced mobile system and has confirmed the feasibility of the initial design choices. Prototyping has enabled potential critical issues relating to component compatibility and structural robustness to be identified and resolved. The system design has been optimized to facilitate future testing and improvements, while maintaining a modular and adaptable configuration. This approach provides plenty of opportunity to integrate further functionality or make modifications based on the data collected during the experimental validation phase. The foundations laid by this work make it possible to move towards experimentation with a system that, in its preliminary stages, already meets the defined requirements. The prototype provides an operational model that can be further refined, opening up promising prospects for future applications.

–4–

Calibration Technique for Vision-Based Systems on the Rover

CONTENTS: **4.1 Introduction.** **4.2 Hand-EYE Calibration.** **4.3 Experimental Setup.** 4.3.1 Experimental Configuration – 4.3.2 Robot Configurations for Testing. **4.4 Proposed Calibration Algorithm.** 4.4.1 Mathematical Framework – 4.4.2 Data Acquisition Strategies – 4.4.3 Image Processing and Marker Detection – 4.4.4 3D Point Reconstruction Techniques – 4.4.5 Plane Fitting for Calibration Accuracy – 4.4.6 Projection and Registration Techniques – 4.4.7 Optimization and Matrix Computation – 4.4.8 Validation and Error Analysis. **4.5 Conclusions.**

4.1 Introduction

Computer vision is a field of study that uses a variety of techniques to automatically extract information from digital images. This information can be as simple as the average value of a colour channel, or as complex as the recognition, position and orientation of objects in a scene. The enormous potential of computer vision has led to its increasing use in many advanced applications, including automatic inspection, industrial process control, diagnostics, autonomous driving and, most notably, robotics.

In mobile robotics, and particularly in mobile manipulator robots, the integration between the vision system and the robotic arm mounted on the rover is a crucial aspect. Effective collaboration between these two components requires accurate calibration between the camera mounted on the mobile robot and the manipulator. Inadequate calibration would affect the accuracy of the operations performed by the arm, making it difficult to grasp and manipulate objects and reducing the reliability of the system as a whole. For this reason, the calibration technique discussed in this chapter play a crucial role in optimising the performance of the rover. The hardware components used in this calibration process are detailed in Appendix A.4.

Although the proposed calibration algorithm was initially developed and tested on a test bench, its implementation on the mobile robot is straightforward and an essential step in ensuring an accurate match between visual perception and manipulation capabilities. This transition from the test bench to the real system is a key element of the thesis, as it allows the experimental work to be used to directly improve the performance of the rover in the field applications.

4.2 Hand-EYE Calibration

It is a common practice to describe the movement of a robot towards an object. However, in reality, the robot is not moving towards the object itself, but rather towards a pose in which it expects the object to be located. This distinction is nevertheless significant. In the event that the object is not situated at the anticipated pose, the robot will be unable to grasp it. Similarly, the robot may be unable to complete the task if imperfections in its mechanism or controller prevent the end-effector from achieving the specified pose. The conventional approach is contingent upon the ability to address two significant challenges: the determination of the desired object pose and the assurance of the requisite accuracy in reaching that pose.

The initial challenge of determining the pose of the object is frequently circumvented in manufacturing by ensuring that the object is consistently positioned with precision through the use of mechanical jigs and fixtures. However, these solutions are costly and require the design and setup of a new solution for each part the robot interacts with, which limits the flexibility of robotic automation. The second issue, namely the assurance that the robot attains the intended pose, is equally intricate. This necessitates the utilisation of meticulous kinematic models, exacting joint sensors and high-performance joint controllers. It is imperative that the links of the robot are of exact length and that its axes are perfectly parallel or orthogonal. Furthermore, the links must be rigid in order to withstand deformation from gravity or dynamic loads, and the controllers must surmount any errors caused by friction or external forces.

An alternative methodology is to facilitate the ability of the robot to perceive both the object and its end-effector, utilising this data to direct the end-effector towards the object. This process, similar to the coordination between the human hand and eye, is known as vision-based control or visual servoing, specifically Position-Based Visual Servoing (PBVS). PBVS is a robotic control strategy that uses visual data to estimate 3D pose of a target object - including both position and orientation - relative to the camera. This estimated pose is then used to compute the control commands necessary to move the end-effector of the robot to the desired position and orientation relative to the target. PBVS typically requires a geometric model (eg. Kinematics of a robotic arm) of the object and calibrated camera parameters (Intrinsic and Extrinsic) to achieve accurate pose estimation and effective control (93).

This approach entails the utilisation of one or more cameras to facilitate real-time feedback, thereby guiding the robot to accomplish the desired task (Fig. 4.1).

With visual servoing, the position of the target does not need to be known *a priori*. Instead, the robot moves towards the observed target wherever it is in the workspace. This approach has many advantages:

- It reduces the need for precise part placement.
- It can handle moving parts seamlessly.
- Errors in the accuracy of the robot are compensated for in real time.

Visual servoing involves controlling the position of the end-effector of the robot relative to the target object using visual features extracted from an image of the object. This method significantly increases the flexibility and robustness of robotic systems, making them more adaptable to dynamic environments (18).

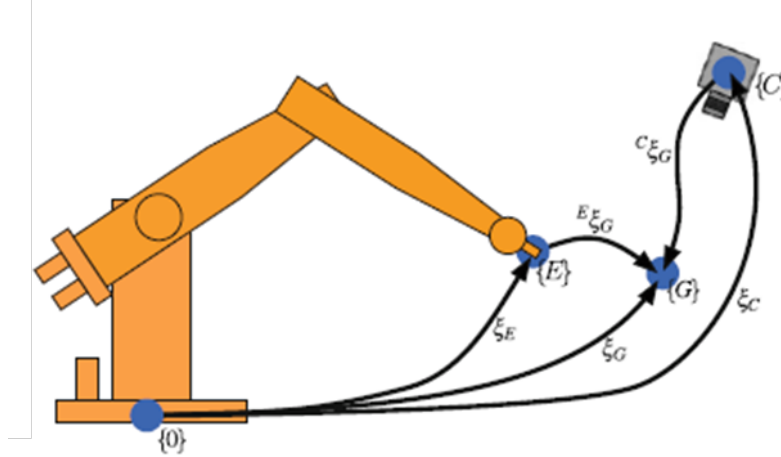


Figure 4.1: Visual servo configurations as a pose graph. The coordinate frames are: world 0, end effector E, camera C, and goal G for the end-point open-loop configuration

To ensure proper coordination between the robot arm and the camera, it is essential to perform a precise hand-eye calibration. This calibration enables the robot to correctly interpret the visual information captured by the camera and translate it into precise arm movements, allowing it to perform each operation optimally. Hand-eye calibration aims to establish a precise relationship between the reference system of the camera and that of the robot arm. In practice, it allows the robot to 'understand' where an object is in space in relation to its current configuration and how it needs to move to reach or manipulate it. This is critical in scenarios where the robot needs to perform complex tasks, such as grasping moving objects, interacting with objects placed in imprecise locations, or performing tasks that require high precision. Inaccurate calibration could lead to significant errors in arm movements, reducing the efficiency and reliability of the system.

Hand-eye calibration methodologies, originally developed for stationary robotic arms operating in controlled environments, have subsequently been extended to include mobile robotics, including rovers designed for field operations. This transition from stationary setups to dynamic, unstructured terrains presents challenges that necessitate the development of robust and adaptable calibration approaches.

Hand-eye calibration, particularly for RGB-D cameras, is a crucial aspect of operational capabilities of the rover. This process ensures accurate spatial alignment between the coordinate frame of the camera and the reference frame of the robot, thus facilitating real-time environmental perception and informed decision-making. As demonstrated by (94), the use of invariant calibrators and iterative algorithms enhances calibration precision, reducing errors to levels suitable for high-accuracy industrial applications.

Such developments are of critical importance when applied to rovers, given the additional complexities introduced by environmental variability and mobility.

Moreover, it is crucial to conduct a hand-eye calibration in order to employ visual servoing, a technique that relies on visual feedback to regulate the motion of a robot. The efficacy of visual servoing is contingent upon accurate calibration, which is necessary for the interpretation of visual data and the subsequent adjustment of movements. In the context of mobile robotics, the use of uncalibrated or poorly calibrated systems can result in significant errors in navigation and manipulation tasks. For example, an uncalibrated hand-eye visual servoing approach may result in the inability of the system to track moving objects or maintain desired features (95).

In the field of robotics, the ability of a rover to navigate and interact with its environment is contingent upon its sensor-actuator coordination. Accurate hand-eye calibration exerts a direct influence upon:

- It is essential to ensure that the depth camera is properly calibrated in order to guarantee the reliability of the data it provides for terrain mapping, obstacle avoidance and navigation. In the absence of accurate alignment, the rover is susceptible to misinterpreting the spatial layout, which could result in navigation errors or potential damage to the system.
- For rovers equipped with robotic arms, hand-eye calibration is a crucial aspect to consider, as it enables the completion of tasks such as sample collection, manipulation of objects, or deployment of tools with the desired precision. Errors in calibration can potentially lead to a compromise in the accuracy and reliability of these activities.
- The integration of data from multiple sensors, including cameras, LiDAR, and IMUs, is dependent upon the implementation of consistent calibration. The presence of misalignments between sensor frames has the potential to result in discrepancies that may lead to a deterioration in system performance.

This chapter presents a hand-eye calibration algorithm designed to determine the homogeneous transformation between the coordinate system of the robotic arm (eq. 4.1) and that of the fixed camera mounted on the rover, as in Fig. 4.2. This calibration is critical to ensure the accuracy of robotic manipulation operations when the arm interacts with the external environment. As described in (18), the correct determination of the transformation matrix between sensors and actuators is essential for the fusion of sensory data and the accurate execution of robotic tasks. where the in Trasformation terms is:

$${}^r\xi_C = {}^r\xi_M \oplus^M \xi_C \implies {}^rP = {}^rT_M {}^MT_C {}^CP = {}^rT_C {}^CP \quad (4.1)$$

It is easy to recall that r represents the robotic base reference frame, C denotes the camera reference frame, and M corresponds to the invariant marker reference frame.

The theoretical principles behind the algorithm are examined, followed by a detailed analysis of the implementation process, focusing on the challenges encountered during

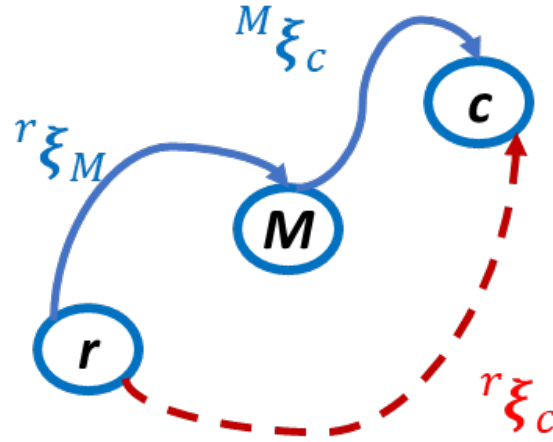


Figure 4.2: Pose Graph for case study application

practical application. The results of the tests carried out under static conditions show that this methodology is easily adaptable to complex scenarios. Furthermore, the proposed method proves to be suitable not only in the specific context of this study, but also in a wide range of industrial applications where calibration is an essential element for automation and operational efficiency.

This chapter is dedicated to an exhaustive examination of the technical aspects of the algorithm, encompassing both mathematical analysis and implementation specifications. This is done with the objective of providing a lucid and comprehensive account of the process. Subsequent to this, the implications of the methodology are discussed, with particular attention paid to the limitations and potentialities that emerged from the tests conducted. This approach allows for the delineation of not only the capabilities of the algorithm, but also possible avenues for further development and improvement.

4.3 Experimental Setup

The experimental setup represents a crucial element of the hand-eye calibration process, which is designed to assess the efficacy of the proposed algorithm in computing the transformation matrix between a robotic arm reference frame and a vision system reference frame (as in Fig. 4.2 according to eq. 4.1). The combination of precision hardware, an invariant calibrator and a systematic workspace configuration ensures the reproducibility and accuracy of the calibration results. The utilization of state-of-the-art components and meticulously devised configurations enables the proposed methodology to achieve high precision, satisfying the demands of contemporary industrial robotics applications.

A red marker (Fig. 4.3b) is fixed to the end effector of the robot, providing a constant reference point throughout the calibration process. The red marker is selected for its high contrast against typical backgrounds, facilitating the efficient segmentation of images during the subsequent processing stage. The distinctive colouration facilitates the accurate detection of the position and orientation of the marker in the captured images. By maintaining a fixed position on the end-effector, the marker provides a stable reference across a range of robot poses, ensuring that the calibration algorithm receives uniform and reliable data. This consistency is of great importance for the accurate determination of the transformation matrix between the coordinate frame of the robot and that of the camera, which in turn allows for precise hand-eye coordination.

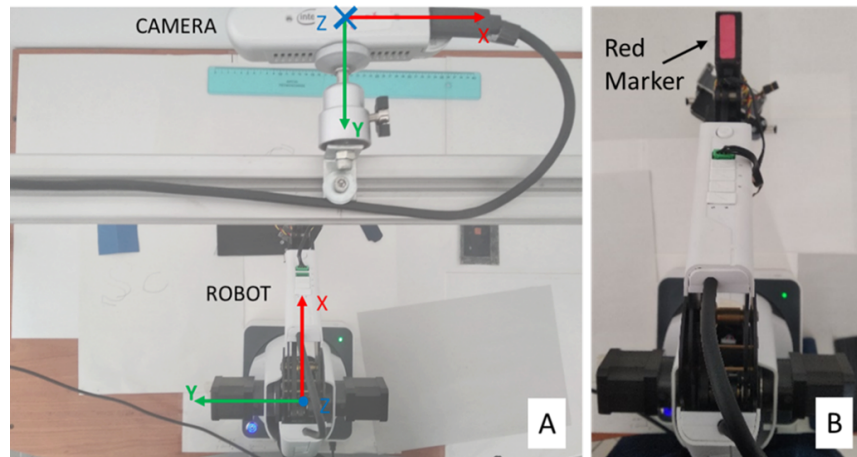


Figure 4.3: Experimental bench setup: (A) robot and camera reference frames, and (B) red color marker located on the robotic arm

4.3.1 Experimental Configuration

4.3.1.1 Reference Frame Definitions

Two primary reference frames are established as showed in Fig. 4.3a:

- Robot Base Frame: Defines the coordinate system of the robotic arm, serving as the origin for all robot movements and positions.
- Camera Frame: Represents the coordinate system of the vision system, with its origin at the optical center of the camera.

The purpose of the calibration process is to identify the transformation matrix that accurately maps coordinates from the camera frame to the robot base frame.

4.3.1.2 Workspace Design and Setup

The workspace(Fig.4.4) is designed with precision to guarantee optimal conditions for the calibration process. A primary consideration is to ensure that the calibration marker remains within the field of view (FoV) of the camera across all planned robot poses. Such visibility is of the highest importance for the acquisition of consistent data and the achievement of accurate calibration results. To avoid any potential collisions between the robot and the camera setup, the necessary clearance is maintained at all times, thus ensuring the safety of the operation. Both the robot and the camera are mounted securely to prevent any movement during the calibration process. This approach minimises potential sources of error and enhances the reliability of the collected data.

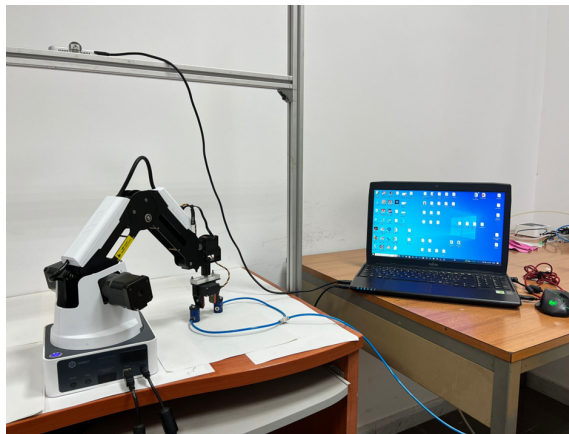


Figure 4.4: Workbench setup for the case study

4.3.2 Robot Configurations for Testing

In order to achieve a fully comprehensive and accurate calibration, the robot is positioned in nine distinct configurations (Fig. 4.5). These poses are strategically selected to encompass a wide range of the operational workspace of the robot, thereby ensuring that the calibration process accounts for a variety of positions and orientations. By selecting non-collinear end-effector positions, the robustness of the calibration process is enhanced, as this diversity enables the accurate determination of the transformation matrix between the camera and robot frames. Each pose is designed to be repeatable with high accuracy, facilitating consistent data collection and contributing to the reliability of the calibration results.

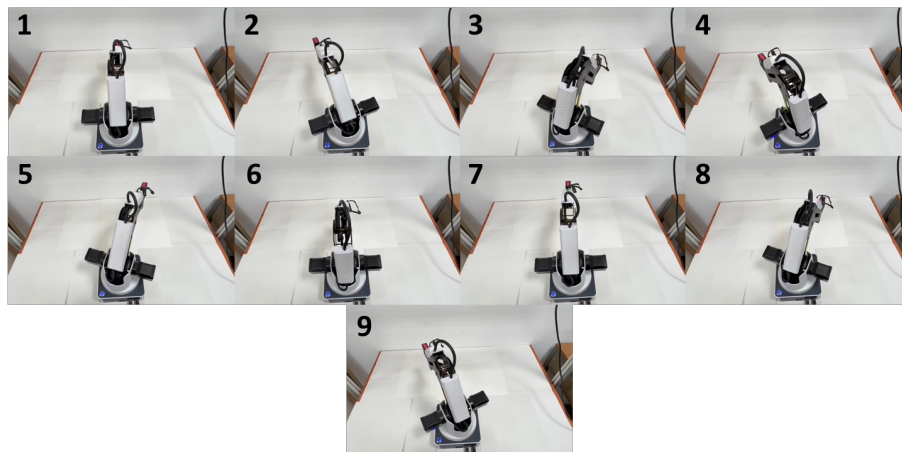


Figure 4.5: Sequence of images illustrating the movement of the robotic arm in the workspace

4.4 Proposed Calibration Algorithm

This part presents the calibration algorithm, which has been developed with the purpose of establishing a precise transformation between the coordinate frame of the robot and that of the camera. The algorithm has been designed with the intention of ensuring accurate hand-eye coordination, which is of critical importance for tasks that necessitate precise interaction between the robotic manipulator and its surrounding environment.

Hand-eye calibration represents a fundamental process in the field of robotics, with the aim of determining the spatial relationship between a base robotic reference frame and an attached vision system. This relationship is typically represented by a homogeneous transformation matrix, which encapsulates both the rotational and translational components. It is of great importance that calibration is conducted accurately, as this is a fundamental process for applications such as object manipulation, assembly and inspection. In these applications, the robot relies on visual feedback in order to perform tasks with high precision.

4.4.1 Mathematical Framework

In the context of hand-eye calibration, the aim is to establish the transformation between the camera frame and the base frame of the robot. This relationship is of significant importance for applications that necessitate precise coordination between visual inputs and robotic movements. The calibration process entails the resolution of the matrix equation:

$${}^rP = \begin{pmatrix} {}^rR_c & {}^rt_c \\ 0_{1 \times 3} & 1 \end{pmatrix} {}^cP = {}^rT_c {}^cP \quad (4.2)$$

where the unknown in this problem is the rT_c . As previously stated, the test bench setup comprises two fixed reference frames: the robot base reference frame, denoted as $O_rX_rY_rZ_r$, and the vision system reference frame, denoted as $O_cX_cY_cZ_c$. The aforementioned frames exhibit differences in both origin position and orientation. It is of significant importance to accurately determine the transformation between these frames in order to ensure precise coordination between the robot and the vision system. This process, referred to as hand-eye calibration, involves the computation of the rotation and translation parameters that align the coordinate system of the robot with that of the camera. The matrix rT_c , which is unknown, describes the rotation of the camera reference frame with respect to the robotic base reference frame and the origin of the camera reference frame with respect to the robotic base reference frame.

4.4.2 Data Acquisition Strategies

The acquisition of accurate data is of crucial importance for the effective calibration of the hand-eye coordination. This process requires the capture of synchronized data from both the robot and the vision system across a range of configurations. The quality and diversity of the data points collected significantly influence the precision of the calibration process.

Some of the best ways to enhance accuracy are: ensure diverse robot poses, maintain synchronization, minimize measurement noise and optimize data selection. As previously mentioned, the robot is moved to nine distinct poses within its operational workspace, thereby ensuring a diverse set of orientations and positions. For each pose, the camera captures both RGB(Fig.4.6) and depth(Fig.4.7) images of the scene, with a red calibration marker (Fig. 4.8) attached to the end-effector of the robot. The corresponding end-effector positions and captured images are recorded for each configuration, forming the basis for subsequent analysis. This dataset provides the foundation for the calibration algorithm, supplying the requisite information for computing the transformation matrix.

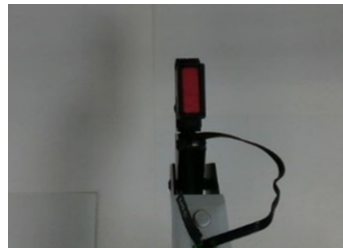


Figure 4.6: RGB Image

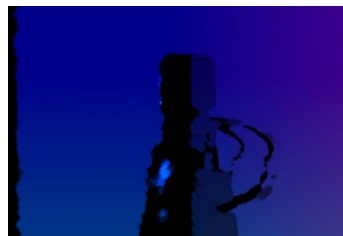


Figure 4.7: Depth Image



Figure 4.8: Invariant Calibrator

4.4.3 Image Processing and Marker Detection

The accurate detection of the calibration marker in the captured images represents a critical step in the calibration process. The marker's distinctive red hue facilitates its discernment against the backdrop. To isolate the marker, the red-*RGB* channel is subtracted from the grayscale image, thereby facilitating more efficacious colour-based segmentation. By applying specific threshold values to the resulting image, regions corresponding to the red marker are extracted (see Fig. 4.9).

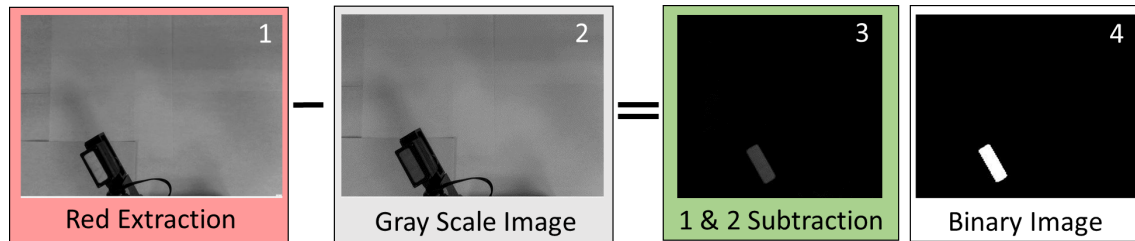


Figure 4.9: Color segmentation algorithm: (1) Red extraction, (2) grayscale image, (3) subtraction of steps 1 and 2, (4) binary image result

Subsequent to the implementation of the segmentation process, a series of morphological operations, including erosion and dilation, are employed to enhance the precision of the identified regions. Erosion removes minor artefacts of noise by eliminating pixels situated along the boundaries of the object, whereas dilation restores the eroded areas, thereby ensuring the accurate preservation of the morphological characteristics of the marker. The aforementioned operations enhance the quality of the segmented image, rendering the region occupied by the marker more distinct. The employment of these image processing techniques enables the system to achieve reliable and accurate detection of the calibration marker, which is essential for the establishment of the spatial relationship between the camera and the coordinate frames of the robot.

4.4.4 3D Point Reconstruction Techniques

Once the calibration mark has been correctly identified in the two-dimensional image plane, the subsequent step is to determine its corresponding three-dimensional coordinates within the frame of reference of the camera. This process, referred to as 3D point reconstruction, is fundamental to establishing the spatial relationship between the camera and the coordinate systems of the robot.

The reconstruction process begins with the extraction of depth information at the specified pixel coordinates from the depth image captured by the Intel RealSense D415 camera. This depth value represents the distance between the camera and the marker along the line of sight corresponding to the pixel location. The depth value obtained is used to transform the 2D pixel coordinates (u, v) and the depth (z) into 3D coordinates (X, Y, Z) in the camera frame. This transformation makes use of the intrinsic parameters of the camera, which include the focal lengths (f_x, f_y) and the principal point coordinates (c_x, c_y) . The equations that define this transformation are as follows:

$$X = \frac{(u - c_x) \times z}{f_x} \quad (4.3)$$

$$Y = \frac{(v - c_y) \times z}{f_y} \quad (4.4)$$

$$Z = z \quad (4.5)$$

The application of these equations enables the conversion of the 2D image coordinates and the associated depth information into 3D spatial coordinates relative to the camera. This process is repeated for each robot configuration, resulting in a set of three-dimensional points that correspond to the position of the marker in a manner that reflects the various poses assumed by the marker. The accurate reconstruction of these 3D points is of critical importance for the subsequent calibration stages, as it provides the points that will be used to define the center and the fitted plane in the next step.

4.4.5 Plane Fitting for Calibration Accuracy

Once the colour data has been segmented and 3D points are obtained, the subsequent step is to fit these points to a plane (Fig. 4.10). This procedure is of critical importance for the purpose of mitigating the effects of noise and camera limitations, which can otherwise lead to misalignment of the points in question. By computing a best-fit plane,

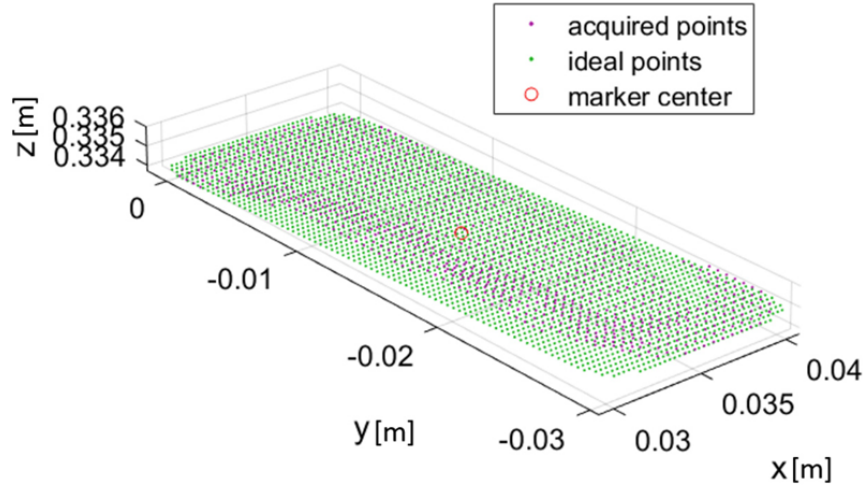


Figure 4.10: Fitting of the plane to the 3D data points: acquired points, ideal points, and marker center

the discrepancies are corrected, ensuring that the data points align with the expected geometric configuration. This process enhances the overall accuracy of the model and provides greater reliability to the calibration process in the context of varying operational conditions.

There are a number of methods that can be employed in order to derive the plane that most closely approximates the given point cloud. These include, but are not limited to, Least Squares Fitting, Robust Estimation Techniques, Weighted Total Least Squares and Improved Cook's Distance. In this instance, the Least Squares Fitting method was selected.

Once the best-fit plane has been determined, the next step is to project all points onto the plane.

4.4.5.1 Least Squares Plane Fitting Method

The objective of Least Squares Fitting of a plane to a set of 3D points is to find the plane equation that represents the data set in the least squares sense. A plane in three-dimensional space can be described by the following equation:

$$ax + by + cz + d = 0 \quad (4.6)$$

where (a, b, c) is the normal vector to the plane, and d is the distance from the origin to the plane along the normal vector. Having a set of M points (x_i, y_i, z_i) which belong to

the 3D point cloud of the marker, the objective is minimize the distance of each of these points to a plane, where the distance is defined as *dist*:

$$\text{dist} = \frac{|a \cdot x_i + b \cdot y_i + c \cdot z_i + d|}{\sqrt{a^2 + b^2 + c^2}} \quad (4.7)$$

The aim is to find the parameters (a, b, c, d) of the plane that minimize the sum of distances from all the points (x_i, y_i, z_i) to the plane.

$$\text{Objective: } \min_{a,b,c,d} \sum_{i=1}^M \frac{|a \cdot x_i + b \cdot y_i + c \cdot z_i + d|}{\sqrt{a^2 + b^2 + c^2}} \quad (4.8)$$

To optimize it, the MATLAB function *fminunc* use the *gradient descent method* starting from a default initial guess that represents a plane uniformly oriented in space:

$$a = b = c = \frac{1}{\sqrt{3}}, \quad d = 0 \quad (4.9)$$

It is important that at the end of the optimization process, the parameters (a, b, c) are normalized to make the normal vector a unit vector which becomes a constrain for the minimization:

$$\sqrt{a^2 + b^2 + c^2} = 1 \quad (4.10)$$

4.4.6 Projection and Registration Techniques

Once the points have been fitted to a plane, it is important to ensure that the resulting plane is parallel to the reference plane of the robotic system. However, due to noise, camera imperfections and limitations in the previous optimization, this is often not done automatically. It is therefore essential to introduce a constraint into the algorithm: during calibration, the robot must move while keeping the marker parallel to the working plane. In this way, the rotation between the obtained plane and an artificial plane constructed to be parallel to the working plane (Fig. 4.11) can be computed and used in the final transformation matrix computation. To compute this rotation it is used the Iterative Closest Point (ICP) method where it is registered the projection of the points fitted to the previous plane to the artificial one. This step is critical for refining the alignment, ensuring that each point in our dataset is positioned accurately within the 3D space.

This approach utilizes a priori knowledge of the system geometry, enhancing the precision of the calibration process. By ensuring that the marker remains parallel to the workplane throughout the course of robot motion, the likelihood of unwanted orientation changes is reduced, thus facilitating a more accurate estimation of the transformation that is required to align the vision system with the robot base reference frame.

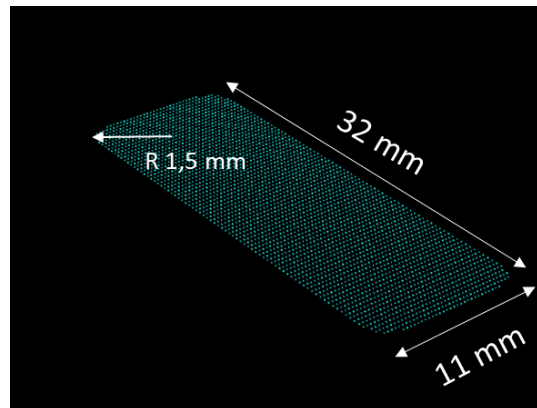


Figure 4.11: Size of the artificial point cloud

By employing a set of parallel planes with a common normal, as illustrated in Figure 4.12, it was possible to establish the spatial orientation of the marker plane. Subsequently, the centre point of the marker was identified by exploiting the uniformity of the artificial marker point cloud. The location of this central point is of great consequence in the solution of the equation that defines the transformation matrix necessary for calibration.

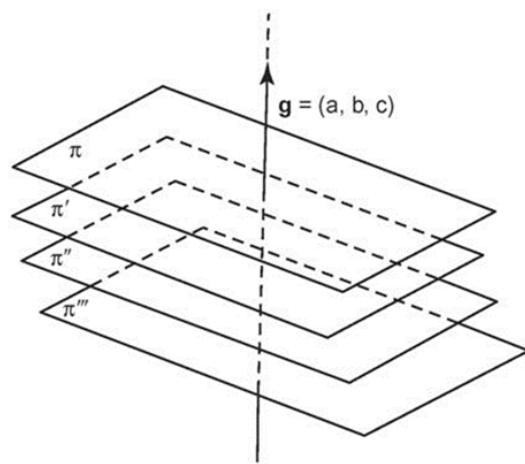


Figure 4.12: Sheaf of parallel planes

4.4.7 Optimization and Matrix Computation

Following the projection and registration steps, the process advances to the optimization phase. In this stage, a Nonlinear Least-Squares solver is employed to refine the transformation matrix, which is essential for converting the perspective of the camera into coordinates that can be used by the robot. The objective is to minimize the calibration error by adjusting the transformation parameters. The function to be minimised, the Transformation Matrix, is formulated in such a way as to incorporate all coordinate data points, enhancing the precision of the calibration. This approach ensures that the transformation matrix accurately represents the spatial relationship between the camera and the robot.

$$\min_{\{P\}} \|F(\{P\}, \{P\}^c) - \{P\}^r\|_2^2 = \min_{\{P\}} \sum_i (F(\{P\}, \{P\}^c)_i - \{P\}^r_i)^2 \quad (4.11)$$

In the context of eq. 4.11, P^c represents the point obtained from the proposed calibration procedure, while P^r denotes the point derived from the forward kinematics of the robotic system. The objective of the nonlinear optimization is to minimize the discrepancy between the two points, refining the transformation matrix that aligns the coordinate system of the camera with that of the robot. This alignment is crucial for ensuring accurate correspondence between the visual data captured by the camera and the operational coordinates of the robot, which is essential for precise manipulation and interaction with the environment.

4.4.8 Validation and Error Analysis

The trend observed in the Least Squares (LSQ) analysis indicates that utilizing nine distinct arm positions leads to a plateau in error reduction, signifying the convergence of the calibration solution (see Fig.4.13). This outcome suggests that employing nine calibration points effectively balances accuracy and efficiency in the calibration process. Such a strategy ensures sufficient data diversity to accurately determine the transformation parameters while maintaining a manageable number of calibration steps.

$$\text{average error} = \frac{\sqrt{\text{SNR}}}{\text{points}} = \frac{\sqrt{336,4527}}{9} = 2,0381 \text{ mm} \quad (4.12)$$

The proposed calibration methodology has been shown to result in an enhancement in precision, with an average error margin of approximately 2 millimetres.

In the bullseye diagrams presented in Figure 4.14, the distribution of calibration errors is visualized, with each point representing the deviation of a measurement from the target center (the point effectively reached according to the robot software). The concentric circles correspond to specific error thresholds. This visualization allows for an intuitive assessment of the magnitude. A concentration of points near the center indicates high precision and accuracy in the calibration process, while points dispersed towards the outer rings suggest greater variability and potential areas for improvement.

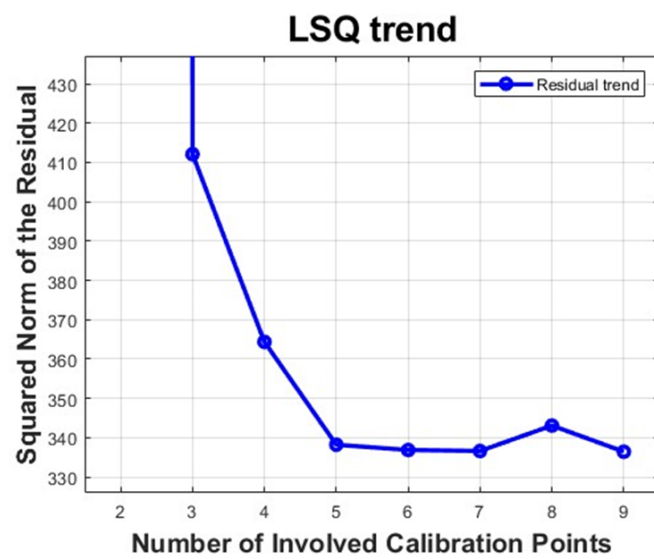


Figure 4.13: LSQ Trend

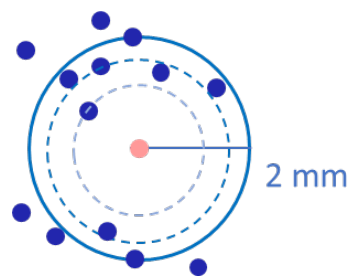


Figure 4.14: Bull-eye diagram

4.5 Conclusions

The research presented in this chapter has successfully demonstrated a robust and precise methodology for calibrating the hand-eye system, which integrates a depth camera and a marker affixed to a robotic arm end-effector. This calibration technique is fundamental to the computation of the transformation matrix, which establishes the geometric relationship between the reference frames of the camera and the robot's base. The procedure was meticulously designed to address the inherent challenges of industrial and robotic applications, particularly the integration of the robotic arm on a rover for field operations. It prioritises accuracy, repeatability, and adaptability within Industry 4.0 paradigms.

A significant advantage of the proposed calibration system is its cost-effectiveness, which is a notable benefit in comparison to other available options. The utilization of readily available and affordable components enables the system to offer a practical solution without compromising precision. This cost-effective methodology facilitates the implementation of sophisticated calibration techniques across a wider spectrum of applications, particularly in small and medium-sized enterprises where financial limitations are a significant concern.

The proposed methodology demonstrated an average calibration error margin of approximately 2 mm, which is competitive when compared to the current state of the art. For example, in reference to (96), the authors reported submillimetre accuracy, with translational errors below 0.28 mm. Although their results outperform the 2 mm error margin achieved in this work, their approach relies on markerless calibration using high-precision robotic flange features, which may require more expensive and specialized hardware. In contrast, the methodology proposed in this chapter offers a practical and cost-effective alternative that is sufficiently accurate for a wide range of industrial applications.

Similarly, (97) employs machine learning models and sophisticated three-dimensional vision techniques, resulting in translational deviations of 0.93 millimetres and rotational deviations of 0.265 degrees. While the learning-based approach enhances accuracy, it introduces supplementary computational complexity and necessitates substantial training data, which may not be attainable in all industrial contexts. In comparison, the approach developed in this thesis is more straightforward to implement, computationally efficient, and does not depend on pre-trained models, making it a more adaptable solution for real-world scenarios.

The proposed calibration methodology is also integrated seamlessly with the framework of Industry 4.0, offering a scalable and adaptable solution for complex industrial applications. By utilising the uniformity of the point cloud data generated by the marker, the final central point of the marker was calculated with great precision, forming the basis for solving the transformation matrix equation. This comprehensive approach ensures the applicability of the calibration system across a wide range of robotic tasks, including the integration of the robotic arm on a rover for field operations, from assembly operations to automated inspections.

The introduction of a constraint, namely the requirement that the marker remain parallel to the reference plane during calibration, led to a notable improvement in the

accuracy of the transformation matrix computation. This resulted in a reduction in the inconsistencies that had been observed in unconstrained systems. Furthermore, the implementation of a streamlined invariant marker design contributed to the reliability and ease of implementation of the method. The combination of these strategies facilitated robust calibration, enabling accurate robotic manipulation and interaction with the external environment.

In conclusion, the research presented in this thesis offers a reliable, efficient, and precise calibration framework that strikes an optimal balance between cost, simplicity, and accuracy. Although it may not attain the same degree of submillimetre precision as some more sophisticated techniques, its accessible design and adaptability make it a suitable option for a wide range of industrial applications. The demonstrated effectiveness, validated through rigorous experimental analysis, provides a robust foundation for future advancements in robotic vision systems, with significant implications for industrial automation and beyond. These findings emphasise the necessity of achieving an equilibrium between precision and practicality, thereby facilitating the extensive adoption of hand-eye calibration technologies.

–5–

Applications of Robotic Navigation

CONTENTS: **5.1 Introduction.** **5.2 Vision-Based Algorithms for Motion Perception.** 5.2.1 Techniques and Methodologies – 5.2.2 Experimental Results – 5.2.3 Conclusions. **5.3 LiDAR-Based SLAM for Single and Multi-Agent Systems.** 5.3.1 System Architecture – 5.3.2 Algorithm Workflow – 5.3.3 Validation Through Experiments – 5.3.4 Configurations for Multi-Agent Collaboration – 5.3.5 Conclusions. **5.4 Exploration with Multi-Agent and Collaborative Robotic Systems.** 5.4.1 Turtlebot 4 Setups and Simulation Environment – 5.4.2 Test and Discussion – 5.4.3 Conclusions. **5.5 Robust Control Systems for Accurate Trajectory Tracking.** 5.5.1 Hardware Overview of Robotic Systems – 5.5.2 Principles of H_∞ Control Theory – 5.5.3 Methodologies for Controller Design – 5.5.4 H_∞ Controller Design – 5.5.5 Validation of H_∞ Control Design – 5.5.6 Conclusions.

5.1 Introduction

Autonomous navigation is a fundamental element for mobile robotic systems, determining their ability to move in complex environments, adapt to changing conditions and interact intelligently with the operating environment. In the project of this thesis, which focuses on the development of a rover for environmental applications, navigation is an essential element as it allows the robot to operate without human intervention, optimising movements and interaction strategies with the terrain and obstacles.

The techniques tested in this chapter are directly related to the final implementation of the rover system and provide the basis for its autonomous exploration capability. The integration of LiDAR SLAM algorithms allowed for accurate mapping of the environment, while the use of Adaptive Monte Carlo Localisation (AMCL) improved the accuracy of localisation and ensured greater stability over uneven terrain. In addition, the implementation of multi-agent navigation strategies is perfectly aligned with the potential use of the rover in collaborative robot teams, enabling more efficient operations and advanced coordination between multiple autonomous robot units.

All the experiments carried out in this chapter were not developed as stand-alone exercises, but with the specific aim of enhancing the capabilities of the rover designed in the thesis by making the navigation modules directly transferable to the real system. The ROS-based architecture, map fusion strategies and motion control optimisation have been designed to ensure that the rover can navigate safely and efficiently in the target operational environment, maximising the impact of the work carried out and its applicability to the final project. The hardware components used for navigation are detailed in Appendix A.5.

5.2 Vision-Based Algorithms for Motion Perception

This section focuses on the development and analysis of computer vision algorithms aimed at perceiving the motion of the small size Six-Wheeled Rover, analyzed in the previous chapters. Specifically, two different algorithms have been designed and implemented to analyze the motion of the rover: a light-based segmentation algorithm from a camera-fixed position and a visual Simultaneous Localisation and Mapping (SLAM) system. The first approach is to use a camera outside the rover to track its movement. This method uses segmentation techniques of the captured images to identify and track the rover in its environment. The second approach is based on the use of an on-board camera. By analysing the movement of pixels associated with key features detected in the acquired frames, this method enables both the mapping of the environment and the tracking of the rover itself. The two methods have different purposes in industry and applications, meeting specific needs depending on the operational context. The results of the two approaches will be compared, with a focus on analyzing the differences in performance, accuracy and applicability. This comparison provides a critical assessment of the potential and limitations of each method, contributing to the understanding of optimal solutions for motion perception in robotic contexts.

5.2.1 Techniques and Methodologies

This section introduces the two different vision-based motion perception algorithms used:

- Light segmentation algorithm for tracking the mobile robot from an external fixed position where the testbench in Fig. 5.1.
- Visual Simultaneous Localisation and Mapping (vSLAM) for motion perception using an on-board camera.

Both algorithms process video frames captured by a camera to obtain information about the motion and trajectory of the robot. The main difference between the two is in the configuration of the camera used: the light segmentation algorithm uses a camera external to the robot, while the vSLAM uses a camera mounted on the robot. The light segmentation algorithm, which uses an external camera to frame the robot, aims to determine its position by processing the captured images. This is done using techniques that rely on the properties of certain objects on the robot (e.g. size, shape or colour) to reconstruct its trajectory. Visual-SLAM, on the other hand, uses a camera on board the robot and aims to estimate its pose by analysing the environment. This approach is based on feature extraction methods and the estimation of the position of the robot in relation to objects identified in the environment.

The algorithm was in the MATLAB environment with the help of the Computer vision Toolbox and the Image processing Toolbox.

For the two algorithms developed, before acquiring the images, it is necessary to define the resolution of the images to be acquired, choosing from those supported by



Figure 5.1: Testbench for light detection

the camera, and the number of frames per second. It is also essential to calibrate the intrinsic parameters of the camera, defining the focal length expressed in pixels and the coordinates of the main point (optical centre) on the image plane. These parameters influence the accuracy of the image reconstruction and the accurate estimation of the motion of the robot.

5.2.1.1 Light-Based Segmentation for Mobile Robot Tracking

The main task of the algorithm is to detect the movement of an object by analyzing a series of frames taken by a fixed camera outside the robot. The tracking of the motion of the object is achieved by in a working environment as Fig. 5.1:

- Detection of the moving object in each frame.
- Temporal association of the detections.

Both RGB images and depth maps provided by the RGB-D camera are used to track the object. In particular, the RGB images(see Fig.5.2) are used to localise the object within the images, while the depth maps help to define its spatial position. After the initial acquisition, the resulting images undergo a segmentation process, which consists of dividing the image into several regions based on criteria such as shape, colour, threshold, or size. This process simplifies the image and provides a more meaningful and compact representation for object extraction. In this case, thresholding segmentation is used to separate the object from the background by setting a threshold. This value is used to evaluate each pixel (x, y) in the image and check whether the intensity of the pixel $I(x, y)$

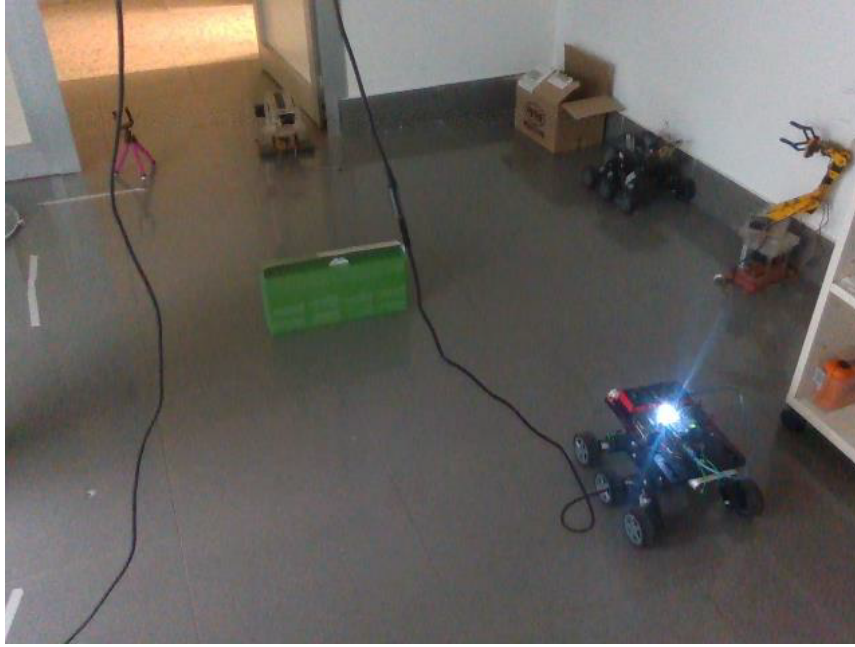


Figure 5.2: Acquired RGB frame from the external camera

exceeds this threshold or not. If the pixel intensity is above the threshold, the point belongs to the object, otherwise it belongs to the background:

$$I(x, y) > s \quad (5.1)$$

To apply segmentation by thresholding, a white LED placed on the Arduino board of the robot was used as a marker. The white colour reflects most of the visible light, making the LED particularly bright and facilitating the implementation of the algorithm. The result of the segmentation is an image composed of white areas on a black background, known as 'Blobs', representing the pixels whose values fall within the specified limits. It is important to note that due to measurement noise or inappropriate thresholds, pixels that do not actually correspond to the LED may appear white. To perform image segmentation, the colour image is first converted to a greyscale image (Fig.5.3), so as to have a scalar as the threshold intensity value, and then subjected to a binarization operation, specifying the threshold value, to obtain a binary image (Fig. 5.4). To obtain the binary image, the pixel is assigned a value of 1 (white) or 0 (black) depending on whether the intensity value of the sampled pixel $I(x, y)$ is below or above the threshold:

$$I(x, y) = \begin{cases} 1 & (x, y) \in \Omega \\ 0 & (x, y) \notin \Omega \end{cases} \quad (5.2)$$

The functions used in the following steps of the algorithm define pixels with value 1 as belonging to the led region Ω and pixels with value 0 as background. It may happen that the segmentation blob is not perfectly defined, but contains gaps within it. This

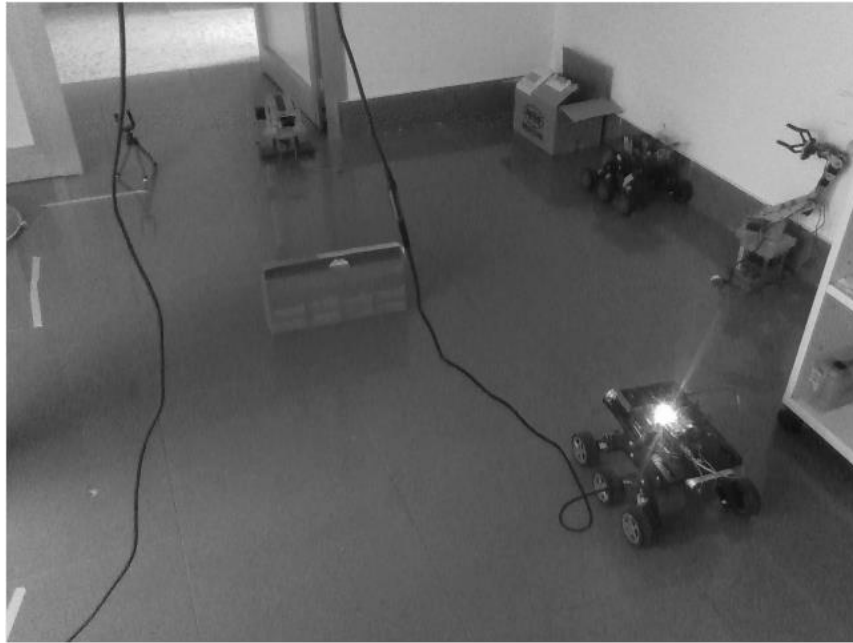


Figure 5.3: Gray-scale image

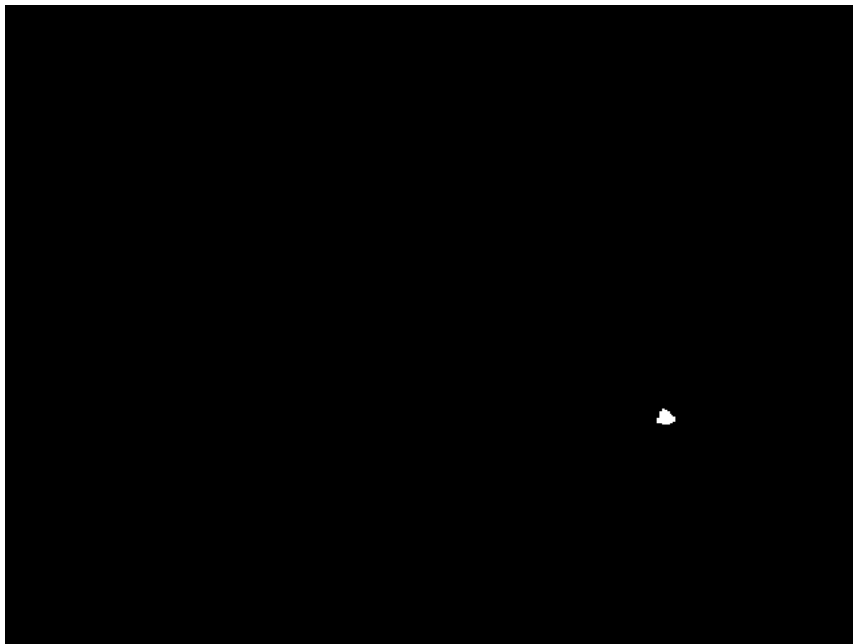


Figure 5.4: Binary Image obtained from the segmentation process

happens when the boundary of the object to be detected has been well extracted, but some points within it have a similar intensity to the background. The image then undergoes an operation to fill in the background pixels that belong to the object, filling in the gaps from the object boundary. Then, to ensure that the target corresponds to the largest blob, the

extensions of all white regions are calculated to define the pixel size P of the largest white region. All regions with smaller extensions are then removed from the image, producing a new binary image characterised only by the blob corresponding to the target. Once the blob corresponding to the target has been identified, the binary image obtained is analyzed and, in particular, the points in the matrix where the pixels take on the value of 1 are searched for in order to obtain the row and column indices of the pixels corresponding to the target. These indices are then used to extrapolate from the depth map of the entire scene only those relative to the light source, thus identifying the X , Y , Z coordinates of the target. Finally, since the objective is to find a single point corresponding to the instantaneous position of the LED, an operation is performed to average the values of the coordinates obtained in order to obtain the position of the centre of the LED. In this way, the position of the robot is obtained. Once the object has been detected by the light, by repeating the process of acquisition and processing of the acquired images, it is possible to follow its movement and obtain its trajectory in the plane by analysing the positions taken over time. The Fig. 5.5 shows a schematic of the pixels associated with the LED light in the binary image for four successive frames, in which the movement of the LED can be seen associated with the movement of the rover over time. A final clarification

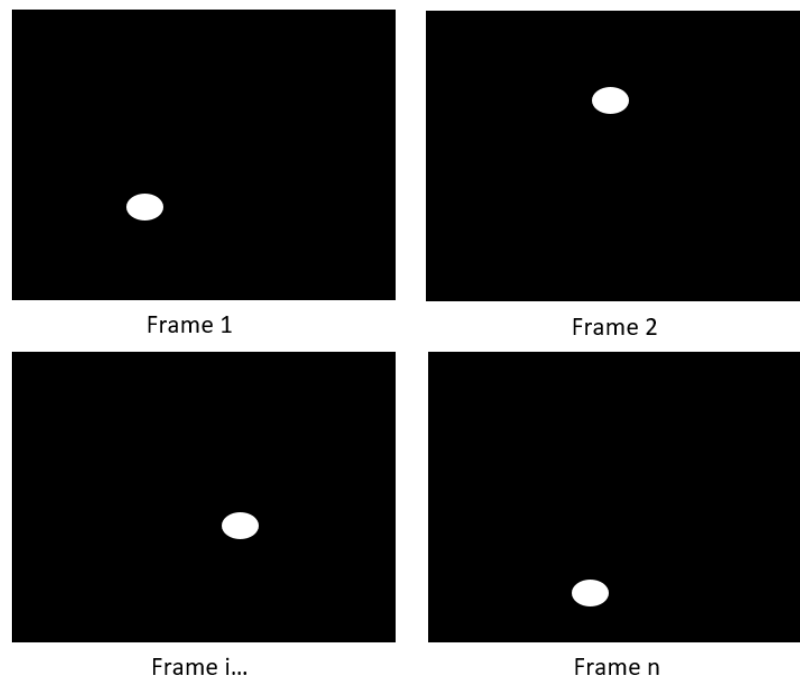


Figure 5.5: Example of following binary image for motion perception

should be made regarding the two parameters used in the algorithm, the threshold and the pixel size P value. These parameters must be properly calibrated, taking into account various factors that could make it difficult to detect the LED object and that could affect the tracking of the path taken by the robot:

-
- Outliers detection
 - Camera distance
 - Light in the environment

Reducing the threshold and pixel size could lead to detection of measurement noise, i.e. points that do not belong to the blob (outliers), causing problems in tracking the motion of the rover. On the other hand, increasing the threshold and pixel size could cause problems in detecting the blob if the LED moves too far away from the camera. Brightness also has a significant effect. In particular, if the room brightness is too high, lowering the threshold would make it difficult to detect the blob. For this reason, the threshold and pixel size are set using a trade-off between the different factors.

5.2.1.2 Visual Simultaneous Localization and Mapping (vSLAM)

With the rapid development of artificial intelligence, sensors and other hardware and software technologies, the development of intelligent robots has gradually become a hot topic in both industry and academia in recent years. More and more robotic systems are appearing in our sight, bringing great convenience to human life. To achieve intelligent functions, robots often need to use their on-board sensors to explore their environment, determine their position, implement route planning schemes, and build maps of the environment in order to reach their destination and perform the specified tasks. The Simultaneous Localisation and Mapping (SLAM) approach is one of the most widely used, which can help intelligent robots to perform these tasks in an unfamiliar environment without any previous knowledge of the environment, and is considered a major advance in robotics, with great theoretical significance and application value (98). In recent years, with the continuous development of computer vision technology, visual Simultaneous Localization and Mapping (vSLAM), which uses the visual input from a camera mounted on the robot to estimate the pose of the robot, has become increasingly popular, achieving good results in terms of accuracy and real-time performance. More precisely, the aim of vSLAM is to estimate the pose of the robot by studying the pose of a camera positioned on board the robot (Fig. 5.6), exploiting the detected movements of pixels corresponding to homologous points in the sequence of images captured by the camera, and simultaneously constructing maps of the environment. In robotics, the 'pose' of the camera is referred to as the combination of the position and orientation of the camera, and can be defined by analysing a single image, a stereo pair of images, or a sequence of images in which the camera is generally moving at a known speed. The vSLAM can be performed using a simple monocular camera, a binocular camera or an RGB-D camera. The first two require a triangulation calculation to obtain depth information, which is difficult to compute accurately, resulting in a poorly estimated trajectory that shifts over time. RGB-D cameras, on the other hand, obtain depth information directly through active physical measurement. The vSLAM algorithms are classified into two categories, depending on how they use the information from the acquired images to estimate the motion of the camera. The

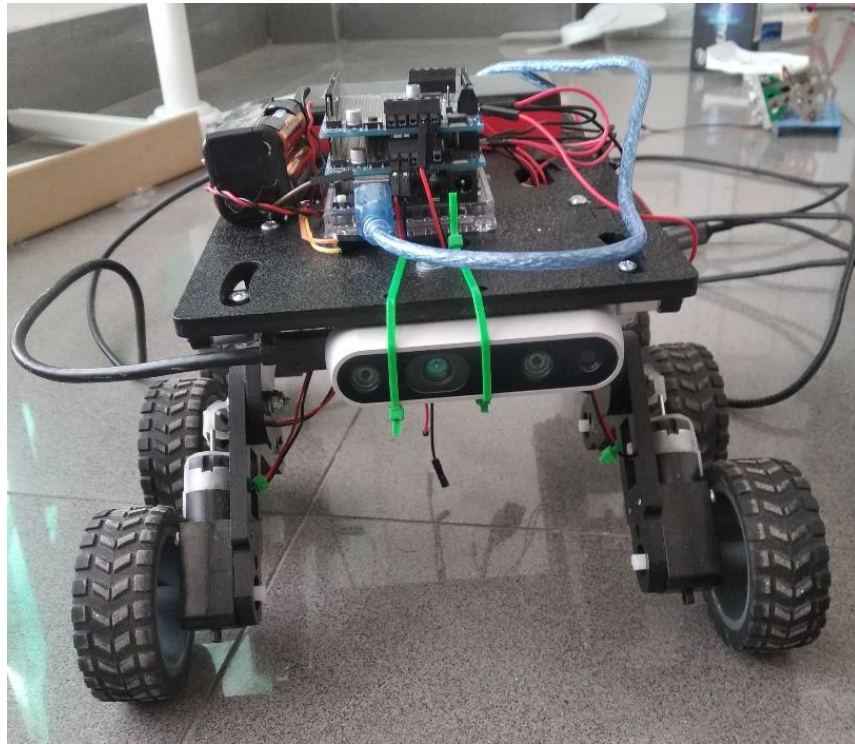


Figure 5.6: Setup for the vSLAM test

vSLAM algorithms can be divided into two categories, depending on how they use the information from the captured images to estimate the motion of the camera:

- The direct method (direct vSLAM) uses all pixels of the captured images to map the environment and estimate the pose.
- The indirect method (feature-based vSLAM) uses only a few pixels of the captured images, corresponding to characteristic points, and relies on these points to estimate position and map the environment.

The direct method provides more detailed information about the environment, allowing a more detailed map to be produced, but requires a lot of computing power and processing time. The indirect method, on the other hand, is computationally easier to implement, but has limitations and can lead to errors. It relies on the extracted points, ignoring the other information contained in the image. This can be problematic when feature matching alone cannot provide a robust reconstruction of the environment, for example in environments with too many or very few salient features or repetitive surfaces. A V-Slam algorithm has been developed based on the extraction of local features from images. A feature is a distinct pattern or structure found in an image, i.e. a point, edge, corner, blob or small patch of the image that defines a separation from a background. They are usually associated with an area of the image that differs from its surroundings in texture, colour or intensity. There are several methods for local feature extraction, some based on

'blob' detection (e.g. SIFT, SURF, KAZE and MSER), others based on 'corner' detection (e.g. FAST and ORB). Among all the methods, the Oriented FAST and rotated BRIEF (ORB) was chosen, which is based on corner extraction and is more accurate in feature extraction because it is computationally efficient, rotation invariant and noise resistant. The ORB method combines two methods: the Features from Accelerated Segments Test (FAST) method for the identification of local feature keypoints and the Binary Robust Independent Elementary Feature (BRIEF) method for the identification of 'descriptors'. The FAST method primarily detects the significant change in the grey level of local pixels. The task of the FAST method is to look at each pixel in the image and compare its grey value with that of the points around it. If the grey value of this pixel differs from most of the surrounding pixels, it is considered a key point. The FAST method primarily detects the significant change in the grey level of local pixels. The task of the FAST method is to look at each pixel in the image and compare its grey value with that of the points around it. If the grey value of this pixel differs from most of the surrounding pixels, it is considered a key point. More specifically, in the FAST method, the pixel point p (Fig.

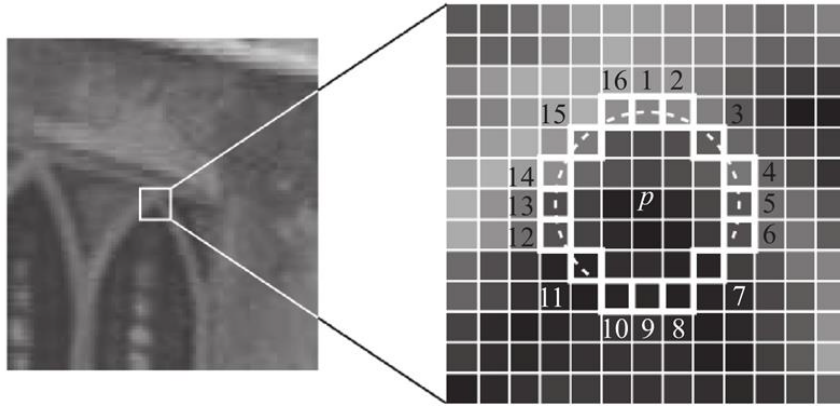


Figure 5.7: FAST method for keypoint extraction (99)

5.7) in the image is compared with m pixels arranged circumferentially around it (16 in this case). Each pixel in the circle is labelled with an integer number from 1 to 16 in a clockwise direction. Pixel p is detected as a corner if there is a set of N contiguous pixels in the circle, all of which are lighter or darker than pixel p . The two conditions can be written as:

$$\forall x \in N, I_x > I_p + t \quad \text{or} \quad \forall x \in N, I_x < I_p - t \quad (5.3)$$

With:

- I_p the brightness value of pixel p
- t the threshold
- I_x the brightness value of the contiguous points belonging to the circle.

If one of the two written conditions is fulfilled in eq. 5.3, the point p can be classified as a corner. The identified keypoints are then used by the BRIEF method to extract descriptors. The descriptor is a binary feature vector that numerically codes the pixels around each keypoint so that together they can represent the local feature. If the key point is too close to the end of the image and the descriptor cannot be identified, it is dropped and does not form part of the final local feature. For image processing and local feature extraction, the image is decomposed into a series of layers (Fig. 5.8) according to a predefined scale factor (100; 101; 102). With an initial image size of $[M \times N]$, the image size at each decomposition

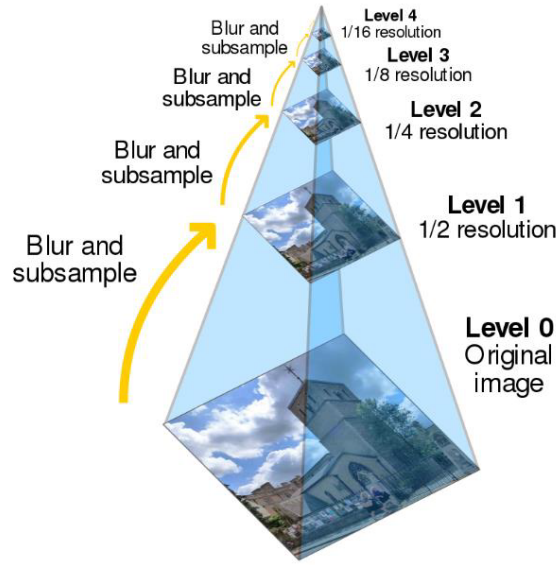


Figure 5.8: Image Decomposition in layers (100)

level is equal to:

$$\frac{M}{\text{ScaleFactor}^{(\text{level}-1)}} \times \frac{N}{\text{ScaleFactor}^{(\text{level}-1)}} \quad (5.4)$$

with $\text{level} \in [0, \text{Numlevels} - 1]$. Where:

- $\text{ScaleFactor}^{(\text{level} - 1)}$ represents the Scale Factor for each level of decomposition
- Numlevels is the number of decomposition levels

However, the number of decomposition levels is limited by the size of the image at that level. The maximum decomposition level is calculated as follows:

$$\text{level}_{\max} = \text{floor} \left(\frac{\log(\min(M, N)) - \log(\text{minSize})}{\log(\text{ScaleFactor})} \right) + 1 \quad (5.5)$$

where minSize is the minimum size at which information can be extracted from the image. Having introduced the local feature extraction method used by the developed vSLAM algorithm, we now turn our attention to it. The algorithm is based on capturing a

number of frames per second and, in order to avoid keeping irrelevant data in memory, it selects from these a set of keyframes, a subset of the most important video frames that contain significant information for robot localization and tracking (according to some threshold parameters, e.g. distance travelled or angular range). Two consecutive keyframes usually indicate a large visual change caused by camera movement. By exploiting the keyframes, and in particular the local features associated with the objects in the keyframes, the vSLAM is able to reconstruct a map of the three-dimensional environment. For a better understanding of the vSLAM algorithm, a more detailed explanation of the different steps that compose the vSLAM (100) is given below (Fig. 5.9):

- Map Initialization
- Tracking
- Local Mapping

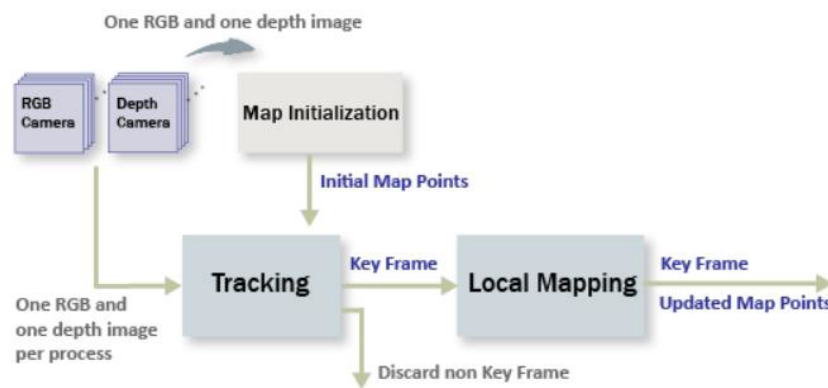


Figure 5.9: vSLAM Steps (100)

In the map initialization phase, the first image is analysed, which is automatically defined as a keyframe, and from which local features are extracted using the previously introduced ORB method. Fig. 5.10 shows a representative image of a first acquired keyframe, while in Fig. 5.11 it is possible to observe the keypoints extracted from this image, i.e. the edges marked with green crosses.

Once the points corresponding to local features have been identified in the first colour keyframe, the corresponding points in the depth image (Fig. 5.12) can be traced using the X and Y indices. In this way it is possible to obtain the X, Y, Z coordinates of the characteristic points belonging to the keyframe, which will then form part of the map points. The map points are a list of points in the 3D world that correspond to the local features of the keyframes and are used to obtain the 3D map of the environment in which the robot is working. By using the map points detected by the analysis of the first keyframe, an initialization of the map of the environment is obtained, which is then updated with the points of the subsequent keyframes. The first keyframe is used to identify



Figure 5.10: First keyframe

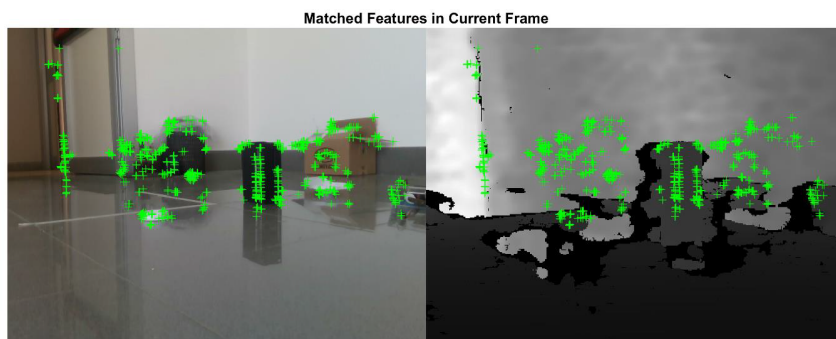


Figure 5.11: Keypoints in the Keyframe

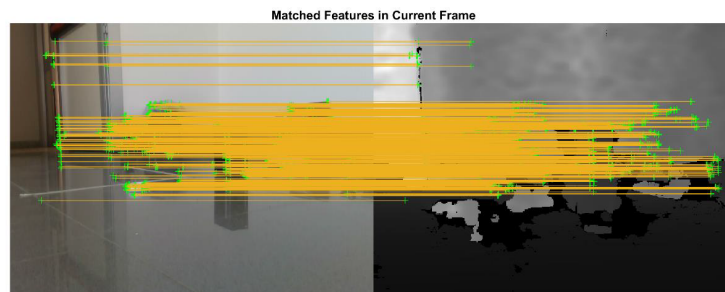


Figure 5.12: Matching the keypoint from the RGB to the Depth

the first camera pose, which later will be identified by an extrinsic parameter calibration process presented in the next section. All subsequent camera poses, which are useful for tracking the path taken by the robot, are then computed according to the latter. After initializing the map with the first pair of colour and depth images, the first keyframes and corresponding map points, useful for the subsequent stages of the algorithm, are stored. In the next tracking phase, each acquired image is analysed to determine whether it is a new keyframe. In particular, each acquired colour image is analysed, from which, as

before, local features are extracted and matched with the features in the last keyframe. By matching the local features and then evaluating rotations and displacements of the corresponding points, the new camera pose is estimated. Then, by projecting the map points observed from the last keyframe onto the current frame, more feature matches can be found, allowing the camera pose to be refined. Finally, it is decided whether the current frame should be a new keyframe. The conditions under which a keyframe can be judged as such can be as follows:

- At least a specified number of frames have elapsed since the last keyframe.
- The current frame is tracking less than a specified number of map points.
- The number of map points tracked by the current frame is less than the number of points tracked by the reference keyframe.
- The Distance Travelled or the Angular Range is less than predefined values.

As the number of keyframes decreases, the computational load decreases, but the distance to the previous keyframe increases. The resulting 'jump' can introduce a greater error in pose estimation. A larger error in pose estimation at defined t step will certainly result in an even larger error at the next time step, and so on. Fig.5.13 shows the resulting tracking plot, pose by pose. After the tracking phase, each keyframe undergoes another phase

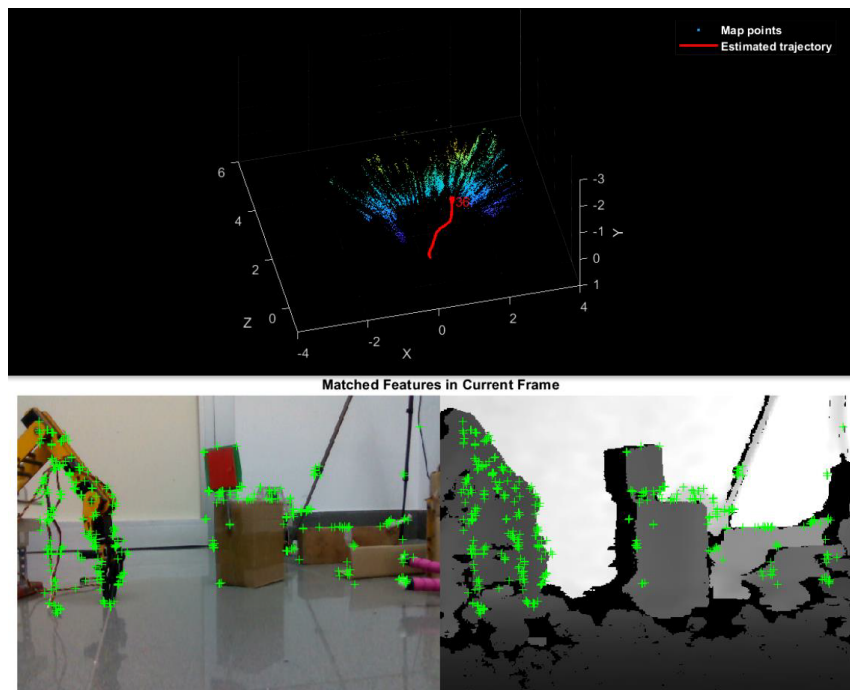


Figure 5.13: Tracking Phase

called local mapping. In this phase, each newly identified keyframe and its corresponding

map points are added to the previous ones. All the keyframes form a 'Graph of covisibility', shown in Figure 5.14, in which the nodes represent keyframes, connected by a curve segment where they share map points. Finally, the ORB points in the current and linked

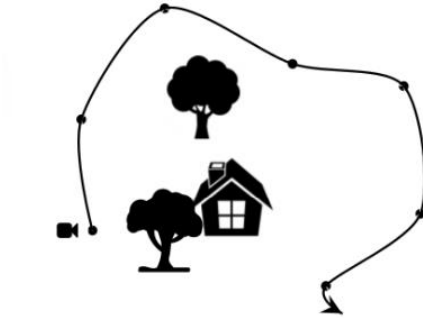


Figure 5.14: Graph of covisibility

keyframes are triangulated to obtain the map of the environment. Fig. 5.15 shows the three-dimensional reconstruction of the robot's environment.

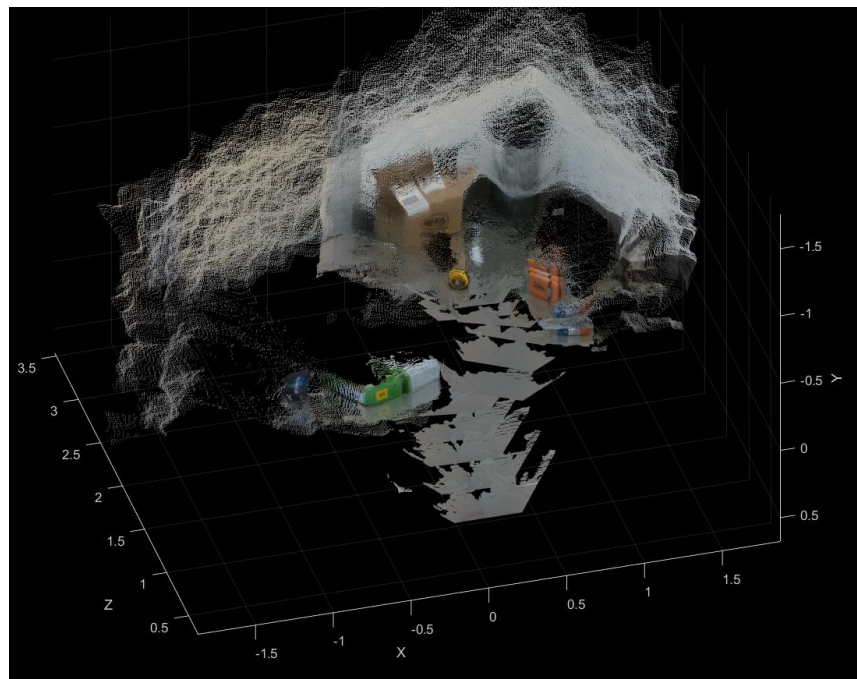


Figure 5.15: Environment Map

5.2.1.3 Camera Extrinsic Calibration

After carrying out the tests on the prototype, a phase of kinematic calibration of the two cameras was necessary in order to compare the trajectories obtained by the two algorithms. The trajectories obtained by the two algorithms are reported in different reference systems, since they are identified from different positions of the two cameras (on-board the rover and from outside), so a kinematic calibration procedure of the two cameras was necessary to report the results in the same world-defined reference system. Kinematic Calibration refers to the set of operations to define the value of those unknown parameters in a transformation matrix 0T_k between coordinate systems:

$$\{P\}_0 = [{}^0T_k]\{P\}_k \quad (5.6)$$

With:

- $\{P\}_0$ position vector of P in the 0 reference frame in homogeneous coordinate.
- $\{P\}_k$ position vector of the same point P in the k reference frame in homogeneous coordinate.
- 0T_k homogeneous transformation matrix 4x4 to bring P from the k reference frame to 0 reference frame.

In this case, in order to define the transformation of coordinates between the local reference systems of the two cameras (fixed and mobile) and a global reference system common to both (Fig. 5.16), it is necessary to know the parameters that rule this transformation.

The direct measurement of these parameters is a difficult operation; the most correct way to identify them is through a procedure that uses the camera itself. Therefore, in order to obtain the calibration of the two cameras, a global left-handed reference system was built, equipped with LEDs of different colours, in order to transform all the results to the new system. In particular, as shown in Fig. 5.17, the red LED corresponds to the x-axis, the yellow LED corresponds to the y-axis, the green LED corresponds to the z-axis and the blue LED corresponds to the origin. From the point clouds acquired by the two cameras, the coordinates of the central points of the coloured LEDs with respect to the blue, red and green points, and consequently the versors of these axes, were identified by the segmentation algorithm. Using the vector products described below, it is possible to obtain the directional cosines of the orthonormal matrix, which represent the coordinates of the rotation matrix:

$$\begin{cases} \overrightarrow{OP_z} \wedge \overrightarrow{OP_x} = normvec \\ \overrightarrow{OP_x} \wedge normvec = \overrightarrow{OP_z}norm \end{cases} \quad (5.7)$$

Where $\overrightarrow{OP_x}$ is the version identified by the blue-red LED, $\overrightarrow{OP_z}$ is the blue-green and $normvec$ is the blue-yellow (perpendicular of the blue-green-red plane). To obtain the

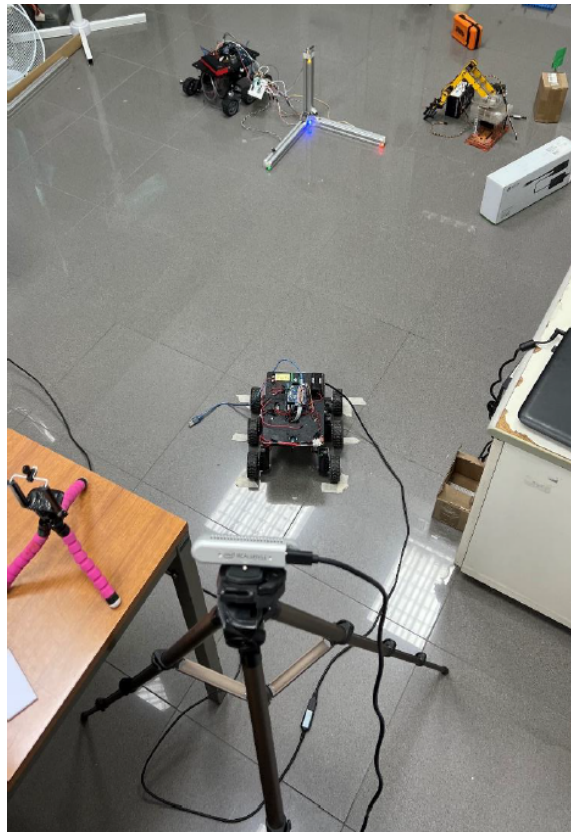


Figure 5.16: Mobile, fixed, and world reference frames represented in the experimental setup

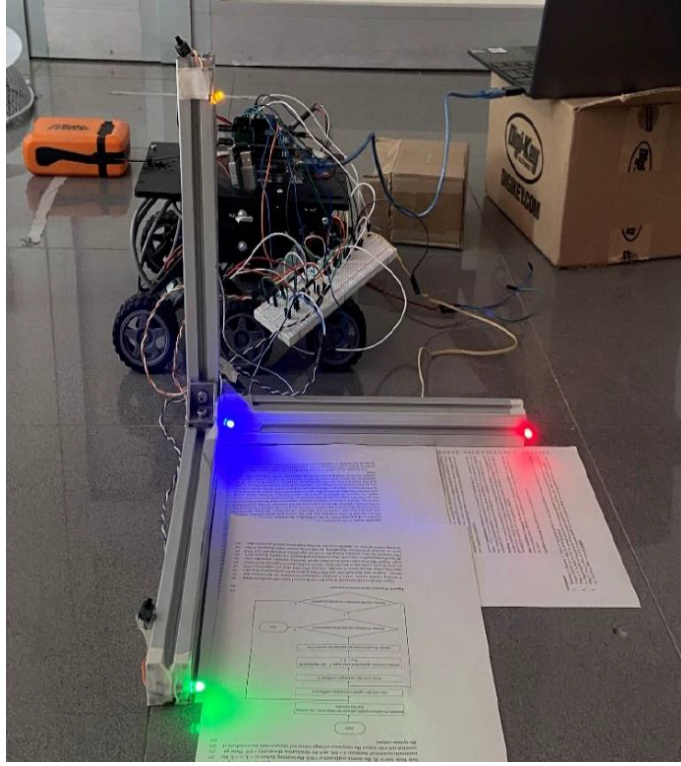


Figure 5.17: Common world reference system with visual markers for spatial alignment

translated points, the coordinates of the origin, indicated by the blue LED, are subtracted from the coordinates of the trajectory points:

$$\{P\}_{translated} = \{P\} - \{0\} \quad (5.8)$$

Finally, by multiplying the translated points by the rotation matrix obtained previously, we obtain the coordinates of the transformed end points:

$$\{P\}_{finals} = [R]\{P\}_{translated} \quad (5.9)$$

Where $\{P\}_{finals}$ are the the translated and rotated points.

5.2.2 Experimental Results

The two RealSense cameras were used to develop the algorithms. Specifically, the D415 camera was installed to observe the working area of the rover, while the D435 camera was positioned on board the rover. The LED-equipped rover was remotely controlled.

5.2.2.1 External Perspective Results in Trajectory Tracking

The results of a series of tests on the application of the light segmentation algorithm are reported in this chapter.

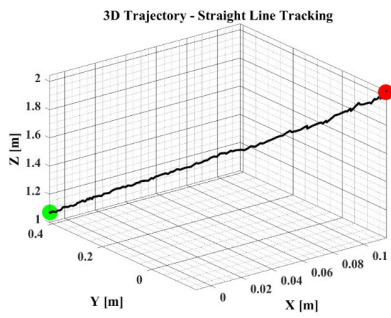
As can be seen from the smoothly plots (Fig. 5.18) in the following tests, the sampling rate for this type of data is very close to the acquisition rate of the camera, which is around 30 fps. This shows a fundamental aspect: the data segmentation and acquisition algorithm is not only efficient, as shown by the homogeneity of the trajectory (which shows no distortion due to filtering), but also computationally light. This efficiency ensures that such trajectories can be processed in real time, an essential requirement for robotic applications operating in dynamic environments.

The segmentation algorithm employed at this stage proves to be even simpler than the one described in 4. In particular, the segmentation process begins directly with the conversion of the image from RGB to greyscale format, reducing the complexity of the initial operations. Despite this simplification, the rest of the algorithmic flow remains unchanged, ensuring reliable and consistent performance.

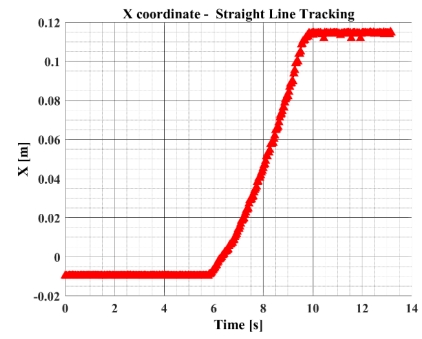
It is important to stress that this type of motion perception, based on the use of an external sensor - in this case an RGB-D camera - falls directly within the field of robot Localization. Localization is one of the most delicate and critical phases of indoor robot navigation, as it is the starting point for planning and controlling the movements of the robot in complex, unstructured environments. The contribution of an RGB-D camera is particularly relevant in this context, as it can allow the acquisition of information on both the relative position and the three-dimensional characteristics of the environment, improving the overall robustness of the indoor navigation system.

As shown in Fig. 5.18, the straightness of the motion is not only evident from the overall 3D trajectory, but also from the trajectory of the x coordinate. The latter, in the reference system of the camera, represents the component that would most easily show a deviation from linearity. It is important to stress that the analysis carried out is based on a qualitative evaluation limited to the camera reference system. This approach is useful for directly observing and interpreting the motion dynamics in the context of the acquired data.

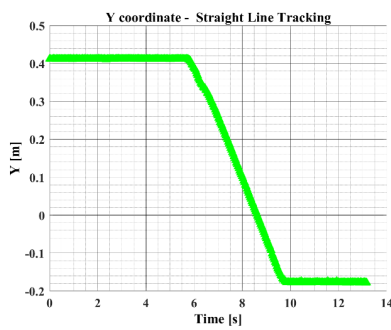
On the contrary, in the case of the turn test, as clearly shown by the graphs in Fig. 5.19, the coordinate x shows a behaviour that accurately reflects the curvature of the path. This distinction between straight and curved trajectories allows us to further validate the effectiveness of the tracking system, demonstrating its ability to accurately represent the geometric characteristics of the motion in different indoor scenarios.



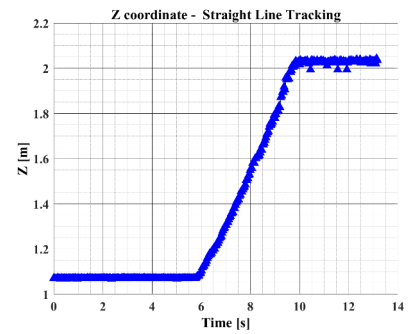
(a) 3D Trajectory - Straight Line Tracking



(b) X coordinate - Straight Line Tracking



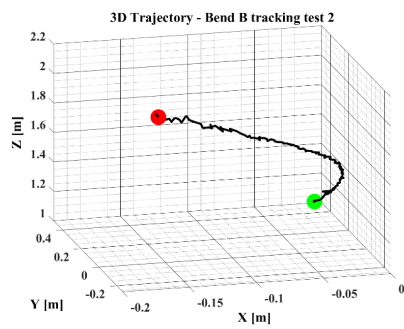
(c) Y coordinate - Straight Line Tracking



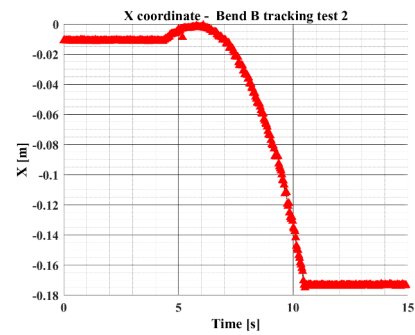
(d) Z coordinate - Straight Line Tracking

Figure 5.18: Tracking performance for a straight path

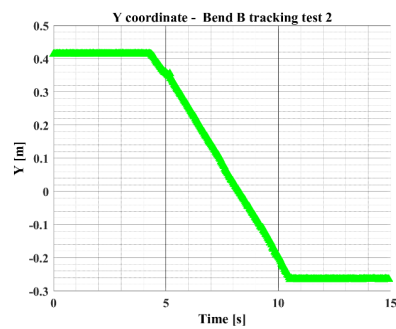
Straight Path



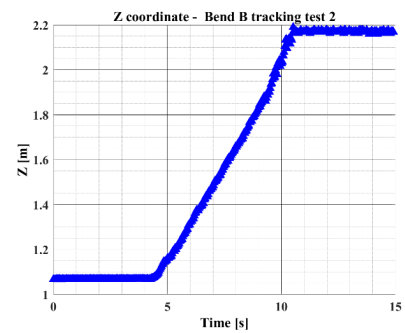
(a) 3D Trajectory - Turn Line Tracking



(b) X coordinate - Turn Line Tracking



(c) Y coordinate - Turn Line Tracking



(d) Z coordinate - Turn Line Tracking

Figure 5.19: Tracking performance for a turn path

Turn

5.2.2.2 Internal Perspective Results in Trajectory Tracking

This section analyzes trajectories obtained using a vSLAM algorithm with feature extraction based on the ORB method. In comparison to the previous case, in this case the data will be acquired using an onboard RGB-D camera and then processed using a vSLAM algorithm known for its computational intensity. Consequently, it is expected that the 'pose' information (in this case limited to the position vector in space) is sampled at a significantly lower frequency than in the method discussed in the previous section.

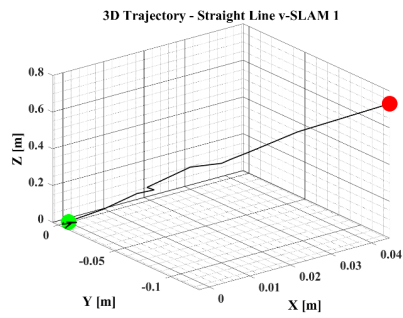
This reduction in sampling rate is not only due to the computational load of the code, but is also influenced by intrinsic constraints, both temporal and geometric, imposed by the algorithm itself. For this reason, the plots shown in Fig. 5.20 and Fig. 5.21 are expected to have a lower point density than those previously analysed.

From a qualitative point of view, it can be observed that the results obtained using the vSLAM algorithm are less accurate, in Localization, than the localization method described above. This difference is mainly due to the nature of SLAM algorithms and vSLAM in particular. Typically, these algorithms do not operate in isolation, but are integrated into more complex pipelines where the resulting pose is the product of sensor fusion processes that combine data from other sensors, such as IMUs or encoders commonly used in Unmanned Guided Vehicles (UGVs).

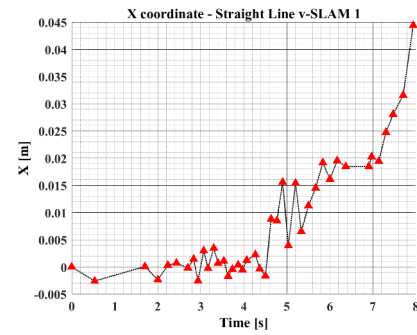
It is important to highlight that in this case the vSLAM algorithm is used exploiting just the RGB-D camera, without any integration with other external sensors such as IMUs or encoders. This choice allows the performance of the vSLAM to be evaluated in isolated conditions, providing a clear picture of its capabilities and inherent limitations.

If the camera is not fixed properly or the information computed from the algorithm is noisy, the evaluation is subject to drift over time, i.e. a progressive accumulation of errors that degrade its accuracy over time. This phenomenon is a well-known limitation of SLAM algorithms, and of vSLAM in particular, when used in the absence of compensation systems.

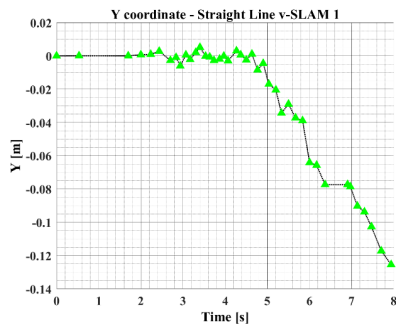
In order to improve the Localization phase of the system, it could be possible integrate in the vSLAM solution an external Localization system as the system proposed in the previous section. These complementary systems make it possible to provide the robot with a global or absolute reference, correcting accumulated drifts and ensuring more accurate and robust navigation, especially in complex or long-duration scenarios.



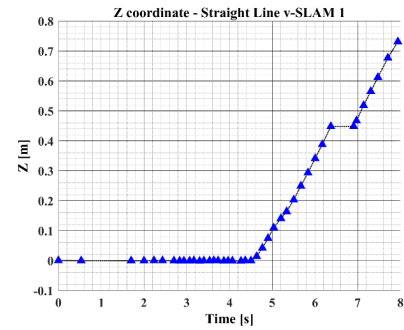
(a) 3D Trajectory - Straight Line vS-LAM



(b) X coordinate - Straight Line vS-LAM



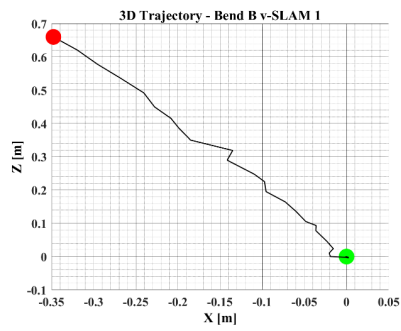
(c) Y coordinate - Straight Line vS-LAM



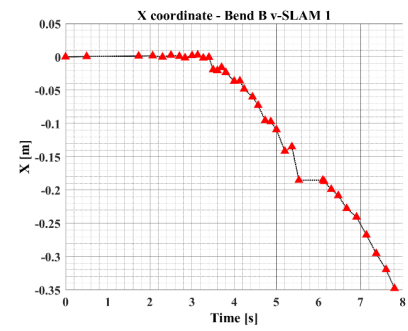
(d) Z coordinate - Straight Line vS-LAM

Figure 5.20: vSLAM performance for a straight path

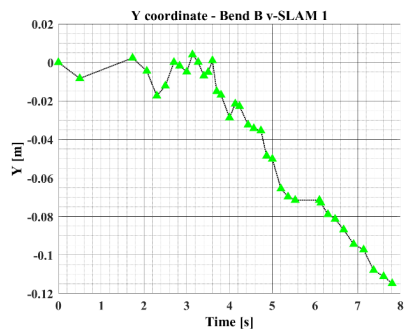
Straight Path



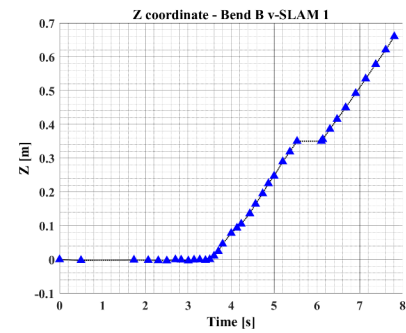
(a) 3D Trajectory - Turn vSLAM



(b) X coordinate - Turn vSLAM



(c) Y coordinate - Turn vSLAM



(d) Z coordinate - Turn vSLAM

Figure 5.21: vSLAM performance for a straight path

Turn

5.2.3 Conclusions

This section of the thesis presents two applications in the context of indoor robot navigation, both based on the use of information contained in images acquired by depth cameras. Specifically, both applications are aimed at perceiving the motion of the robot: the first defines the position of the robot in 3D environment from an external perspective, which falls within the scope of localization; the second, on the other hand, uses an on-board camera and a vSLAM algorithm to acquire the pose of the robot and define its odometry and localization.

The tracking algorithm, which uses light segmentation to track the motion of the robot from the outside, demonstrated high accuracy in trajectory detection. This approach makes it possible to capture continuous, linear motion over time, allowing even complex trajectories to be recorded. However, the algorithm has limitations related to lighting conditions, which can negatively affect the trajectory tracking of the rover. Another limitation is that the fixed camera has a limited Field of View, so it is not possible to obtain results outside this FoV.

On the other hand, the vSLAM algorithm, tested in stand-alone camera mode, has limitations in terms of the odometric accuracy of the robot. Unlike the tracking algorithm, in the case of vSLAM for this application, localization errors (localisation because the position is calculated in the world reference system using the transformation obtained in the extrinsic calibration phase) accumulate over time, leading to significant deviations from the true values. However, this algorithm is not subject to excessive limitations regarding the field of view, since the camera is mounted on the rover and moves with it.

In order to improve the tracking algorithm, an artificial intelligence system could be implemented to automatically calibrate the necessary parameters, such as pixel size and threshold, making the calibration completely autonomous. It is important to stress that the integration of artificial intelligence algorithms for automatic parameter calibration in visual tracking algorithms is an area of active research, with potential practical applications in various fields.

The data generated by the vSLAM, such as the mapping of the environment and the location of the rover, can be used to allow the rover to move autonomously in its environment without the supervision of an operator. However, to improve the accuracy and robustness of the vSLAM, it is common to integrate data from additional sensors, such as IMU, using sensor fusion techniques.

In order to achieve a higher level of autonomy for the rover, it is essential to develop a control system supported by appropriate sensors. In this context, a possible future development of the project involves the introduction of new sensors such as additional cameras, IMU sensors and encoders. The integration of these sensors can improve the trajectory and mapping obtained, as the fusion of data from different sensor sources makes it possible to compensate for the limitations of each sensor and obtain a more accurate representation of the environment. Only in this way can the data produced by the SLAM be effectively used by the robot to execute trajectories in complete autonomy.

5.3 LiDAR-Based SLAM for Single and Multi-Agent Systems

In the context of technological and scientific innovation, it is crucial for academic and research institutions to develop collaborations and knowledge exchanges that promote scientific progress and knowledge. In this sense, a research period focused on the implementation and testing of algorithms for autonomous navigation was conducted in October 2023 at the Faculty of Engineering and Architecture of the University of Udine.

The ultimate goal of this chapter is to lay the groundwork for the realization of a multi-agent autonomous navigation system designed to enable effective cooperation and coordination between multiple robotic units.

The long-term goal of this approach is to extend the system to more units in order to apply swarm autonomous navigation in different scenarios and applications, both in industrial and research contexts.

This chapter presents in a detailed and systematic way the activities carried out during the research period at the Department of Engineering and Architecture of the University of Udine, the challenges faced, the solutions implemented and the results obtained during this research period. The collaboration between the University of Udine and the University of Naples "Federico II" has provided a valuable opportunity to exchange knowledge and expertise, laying the foundations for further developments in the field of autonomous navigation and swarm robotics. This report aims to provide an analysis of the work carried out, which can be developed in future collaborative projects.

The main goal of the project was to implement and test algorithms for autonomous navigation based on Li-SLAM (LiDAR - Simultaneous Localization and Mapping) technology. These algorithms were tested on AGILEX mobile robots equipped with a LiDAR sensor, using Nvidia Jetson Xavier as control board for real-time computation and data processing, exploiting the potential of the Robot Operating System (ROS) as a development framework.

In the preliminary tests, different localization and mapping techniques within the work area were tested on a single robot, while in the final tests conducted, the focus was on the interaction and synergy of two robots, in order to evaluate the feasibility and effectiveness of the system designed in a structured environment.

5.3.1 System Architecture

5.3.1.1 Integration with Robotic Operating System (ROS)

The Robot Operating System (ROS) is an open source framework designed to facilitate the development of robotic applications. Despite its name, ROS is not a traditional operating system, but a collection of libraries and tools that provide services typical of an operating system on a heterogeneous cluster of computers. These services include hardware abstraction, device control via drivers, inter-process communication, and software package management.

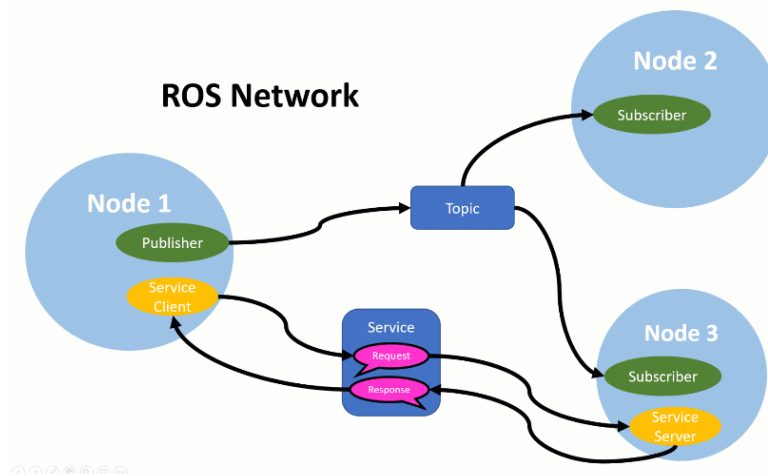


Figure 5.22: Illustrative representation of the ROS network architecture, showing how nodes interact through topics and services to facilitate data exchange and communication (103)

ROS is designed to promote collaboration in robotics by enabling developers to efficiently share and reuse code. Its modular architecture allows the creation of independent software components, facilitating integration and updates without affecting the entire system. In addition, ROS supports multiple programming languages, including C++ and Python.

ROS and its evolution, ROS 2, use a modular architecture based on nodes, topics, services, actions, and parameters to enable efficient and flexible communication between the various components of a robotic system. In ROS, a node is an executable process that performs a specific function within the robotic system. This division into nodes allows for a modular design in which each node can be developed and tested independently. Nodes communicate with each other through mechanisms such as topics, services, and actions. Topics are communication channels used for asynchronous message exchange between nodes (Fig. 5.22). A node can post messages to a particular topic, and other nodes can subscribe to that topic to receive the messages. This communication model is particularly

useful for transmitting continuous streams of data, such as sensor readings. Services implement a synchronous communication model based on request and response. A client node sends a request to a server node, which processes the request and returns a response. This approach is appropriate for operations that require immediate and deterministic interaction, such as accessing specific parameters or executing point commands. Actions extend the concept of services to handle operations that take a long time to complete and can provide feedback during execution. A client node sends a goal to a server node, which executes the action and provides progress updates, allowing even early termination if necessary. Parameters are configurable values that affect the behavior of nodes. They are managed by a centralized parameter server that allows nodes to dynamically read and write parameters. This mechanism is useful for setting configurations that can change without having to change the source code of the nodes.

However, ROS has some limitations, such as lack of native support for real-time control and multi-robot systems, and limited compatibility with platforms other than Linux. To address these challenges, ROS 2 was developed. ROS 2 introduces significant improvements, including: Real-time control support, Multi-platform compatibility, Modular and scalable architecture and, Advanced security.

Together, these components form the computational graph of ROS, which defines the interactions and communications within the robotic system. This modular and flexible architecture makes it easy to develop, maintain, and extend complex robotic applications, promoting code reusability and collaboration among development teams (104).

5.3.2 Algorithm Workflow

For the sake of clarity, a representative diagram of a single robotic unit is shown in Fig. 5.23. As can be seen, the robot is equipped with all the necessary elements to ensure the proper functioning of an autonomous mobile system. The diagram shows the high-level architecture, the core element of which is the NVIDIA Jetson AGX Xavier, which acts as the main computing unit. This component is indispensable as it manages the data processing and execution of the complex algorithms required for the system to operate.

Several devices are connected to the control unit, including a network switch, which is responsible for generating the internal network (or, alternatively, connecting to an external network), and the LiDAR sensor. In addition, the control unit is interfaced with the entire low-level architecture, which is required for the actuation and control of the electric motors. Since the high-level computational unit is not designed to operate directly with the CAN protocol, a USB-CAN interface was integrated. This interface, properly configured at both hardware and software level, allows the signals from the low-level sensors to be included in the ROS network and ensures efficient communication between the control unit and the low-level components, including the motorshields and motors themselves.

The USB-CAN interface plays a crucial role in the integration of low-level control and high-level system management. Thanks to this connection, the AgileX Scout Mini can be connected to a wide range of devices and systems, greatly extending its operational



Figure 5.23: Single agent scheme for the Scout Mini

capabilities. At the software level, the entire system is managed by a package called *ugv_sdk*, which facilitates communication between the different components.

In order to test the interaction between the low-level controller and the high-level management system, a basic ROS package called *teleop* was used to manage movements from the keyboard. This package, included in the standard installation of ROS, simplified the testing process by allowing the robot to be controlled by keyboard commands and simplified the verification of the functionality of the system.

An important aspect of the system is that the connection to the robot controller is not via direct physical connections, but via an SSH connection. This type of connection allows the robot to be managed dynamically and flexibly from the outside, making it particularly useful for making changes and optimizations to the software quickly and efficiently. This approach has accelerated the development and debugging process and significantly improved the reactivity in handling required changes to the system.

Before proceeding with the detailed analysis of all the components of the algorithm, an illustrative diagram of the overall functioning of the system is shown in Fig. 5.24. This diagram is intended to provide a clear and intuitive overview, highlighting the interconnections between the different parts of the system and the flow of information, in order to better contextualize the descriptions that follow.

For this algorithm, the target pose will be defined manually from the outside. For this navigation algorithm purposes, the main sensor used will be LiDAR, which will play a central role in several functions. First, the LiDAR will generate a map of the environment, which can be constructed in real time or created in advance, stored, and used later. In addition to mapping, the sensor will provide the scan for localization algorithms such as Adaptive Monte Carlo Localization (AMCL) and odometry. In addition, LiDAR will be

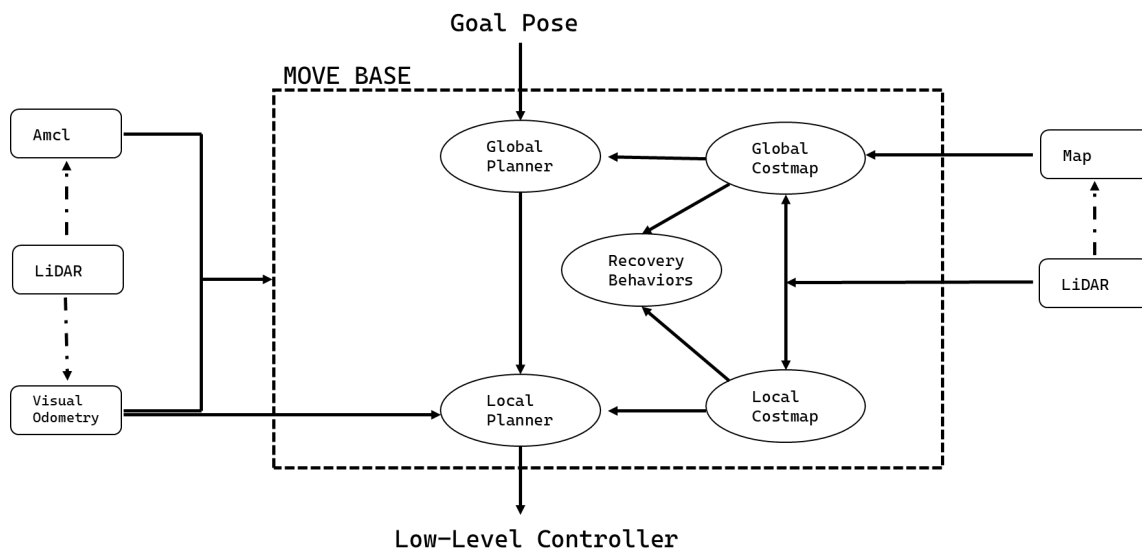


Figure 5.24: Diagram illustrating the overall functioning of the agent in the environment

responsible for generating both global and local cost maps required for path and trajectory planning. Once the Move Base nodelet has received the navigation target and processed the odometry and localization information, it will be able to compute an optimal path to the target through a pathfinding algorithm. The output of this process will be a series of speed commands, which will be sent to the low-level control model. This will manage the motors of the robot, enabling it to move towards the destination.

5.3.2.1 Point Cloud Mapping with LiDAR

A key component of the robotic system is the driver package for the LiDAR, the Velodyne LiDAR Puck VLP-16. The drivers required to interface and manage the sensor are provided through the proprietary *velodyne* package, which allows the LiDAR to be integrated into the ROS framework. This package provides tools and resources to configure, calibrate, and acquire sensor data in real time, facilitating the processing and interpretation of environmental information by the navigation algorithm.

It is critical to configure the communication network to assign static IP addresses to the devices involved. This configuration ensures that each device always has the same IP address, which allows for unique identification and reliable communication between the system components. The static IPs assigned to the various devices are shown in Fig. 5.23 to avoid network conflicts and to ensure that the nodes of the ROS framework can correctly connect to their respective sensors, actuators, and controllers. A very important aspect is the correct configuration of the LiDAR IP address, as well as the configuration of the HOST IP and GATEWAY IP addresses. The latter must be set as the IP address of the controller or the router to which the controller is connected. This configuration is critical to ensure stable and efficient communication between the devices so that the LiDAR can

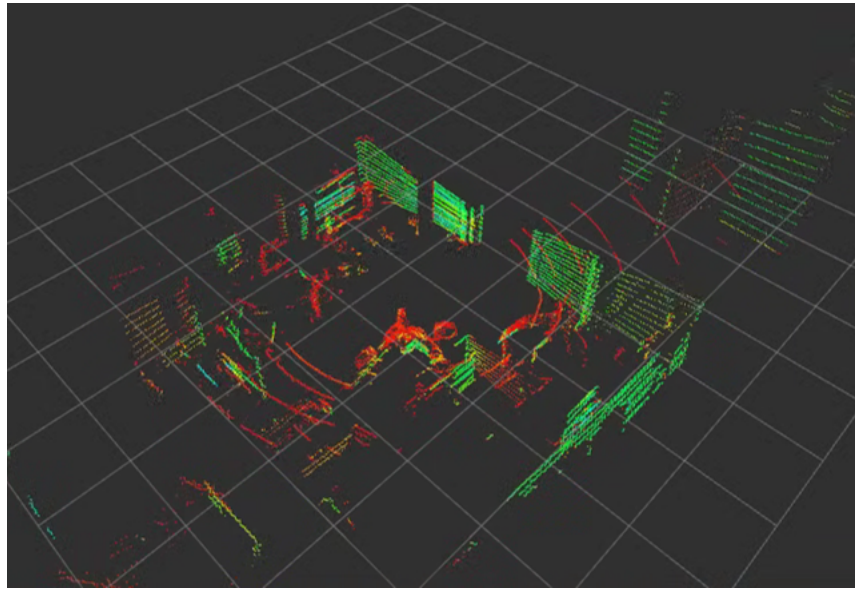


Figure 5.25: Raw point cloud data captured by the LiDAR, representing the scanned environment in three dimensions

reliably transmit the collected data to the system. Different types of data are received from LiDAR. The most relevant of these is a 3D point cloud, which represents a detailed, spatial description of the surrounding environment. As shown in Fig. 5.25, the generated PointCloud was subsequently processed and converted into a LaserScan format using a special software package. This lighter and more specific format was then used in the subsequent processing steps for navigation and mapping.

To obtain the LaserScan Message used in the mapping phase, the ROS package *pointcloud to lasercan* was used. This package allows the conversion of the raw data generated by LiDAR, the pointcloud, into a more compact, specific and 2D format for map processing, the scan. This conversion is essential to make the data compatible with subsequent navigation and localization algorithms. In parallel, the raw data (point cloud) from the LiDAR was processed to extract odometric information. This was possible thanks to the algorithm *DLO* (105) (Direct Lidar Odometry) (of which there is a ROS package), which allows to compute the robot pose directly from the LiDAR data. The use of *DLO* allowed the odometry measurements to be effectively integrated, improving the overall accuracy of the pose of the robot in the operational context.

5.3.2.2 Direct LiDAR Odometry Technique

This algorithm is designed for high computational efficiency, allowing complex point clouds to be processed with minimal pre-processing. Despite this simplification (as the voxelize process), it is able to provide accurate position estimates in real time, making it particularly suitable for applications requiring rapid operational response.

The algorithm uses a keyframe-based organisational structure to handle the data

acquired from LiDAR. Through this structure, the data is exploited to estimate the spatial transformation between two distinct reference systems: the `/base_link` frame and the `/odom` frame. The `/base_link` frame represents the current reference system associated with the robot, following its movements in real time. On the other hand, the `/odom` frame is a fixed reference system with respect to the map (`/map` frame), providing a global context for localization. This distinction between the two frames is crucial to ensure accurate tracking of the pose of the robot with respect to the environment, combining both local and global information.

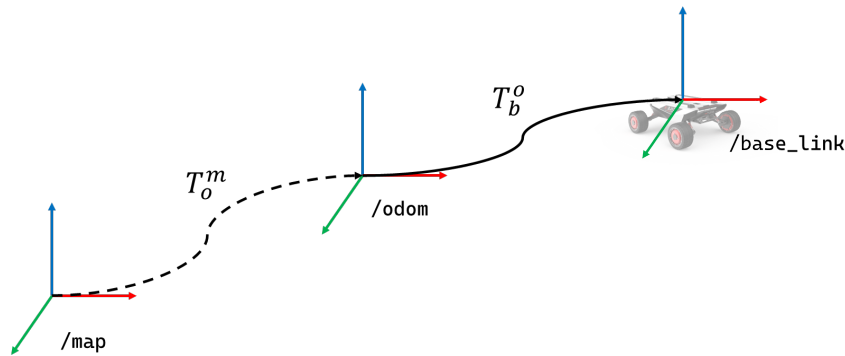


Figure 5.26: Navigation transformation frames chain

As shown in Fig., 5.26, the kinematic chain shows that there are three basic transformations for the navigation phase. When observing the map (Fig. 5.27) generated by the DLO (Direct Lidar Odometry) algorithm in the output, a modest drift was observed, which was particularly noticeable during long duration tests. This phenomenon can be corrected by using a particle filter such as the Adaptive Monte Carlo Localizer (AMCL). The AMCL filter allows the system to be fixed to a global reference and progressively corrects errors that accumulate over time. Moreover, the accuracy of the DLO-generated odometry can be improved by integrating data from an IMU. In particular, gyroscopic measurements can be used to improve the estimation of relative motion between the current and previous LiDAR scan. This data integration provides improved drift robustness by combining the benefits of LiDAR information with the dynamic properties measured by the IMU.

5.3.2.3 GMapping for Environmental Representation

In all robotics navigation, it is essential to have a map of the environment, often represented as an occupancy grid (as shown in Fig. 5.28).

GMapping is an algorithm for creating a map of the environment of the robot using a highly efficient Rao-Blackwellized particle filter (106; 107). This approach makes it possible to learn detailed grid maps from laser sensor data, combining localization and mapping into one integrated process. One of the key features of GMapping is its ability to allow the robot to "locate" itself within the map while it is being constructed or updated. This is particularly useful in dynamic environments where obstacles and spatial configurations

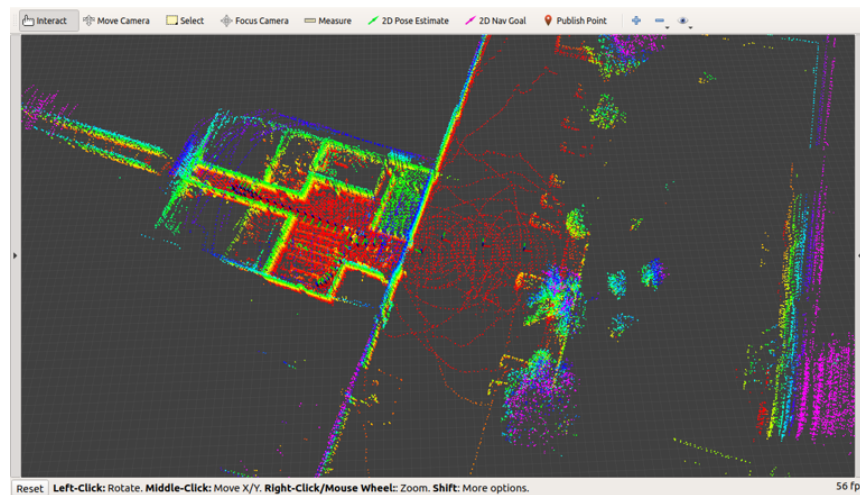


Figure 5.27: Direct LiDAR Odometry (DLO) map displaying the environment with time-variant pose estimation

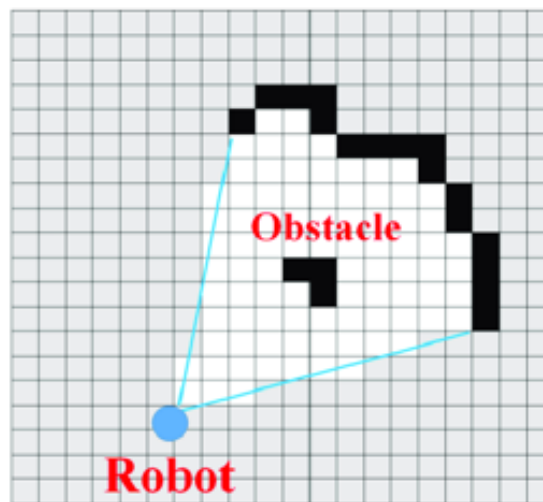


Figure 5.28: Occupancy grid map representing the position of the robot and detected obstacles within the environment

may change over time, or in previously unmapped scenarios where there are no predefined references. Questo pacchetto richiede come input: la trasformazione tra i frame robot/odom e robot/base link (trasformazione risultante dal pacchetto DLO), e i dati provenienti dalla scansione dell'ambiente. Il risultato prodotto è una mappa.

The GMapping package requires specific inputs to generate an accurate map. This includes the transformation between the `/robot/odom` and `/robot/base_link` frames, a transformation provided by the DLO algorithm. This transformation represents the spatial relationship between the odometric reference system of the robot and its base frame, providing a reference for the mapping process. Together with the scan provided by the

pointcloud to lasercan package, it generates a two-dimensional map (Occupancy Grid).

5.3.2.4 Adaptive Monte Carlo Localization (AMCL)

This section describes the method used to localize within the map created in the previous step. It is important to underline that this is an "internal" localization approach. Unlike "external" methods, such as GPS (used for outdoor applications) or UWB (Ultra Wideband) based localization, internal methods, such as AMCL (Adaptive Monte Carlo Localization), use the embedded sensors, in this case LiDAR, to correct the odometry pose generated by DLO (Direct LiDAR Odometry) and allow localization within the map.

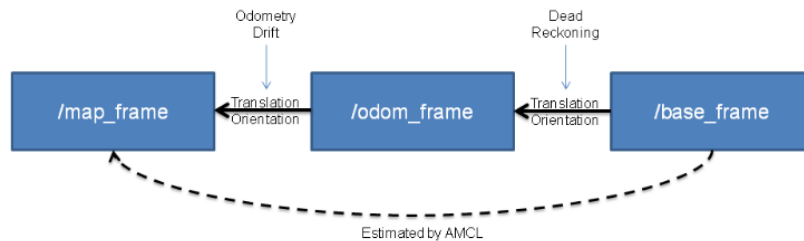


Figure 5.29: Purpose of AMCL (108)

As mentioned in the previous paragraphs, the AMCL technique is a probabilistic method used to estimate the position of a robot in an environment. In particular, this technique allows to compute the transformation between the robot frame (`/base_link`) and the map frame (`/map_frame`), correcting the accumulated odometric drift between the odometric frame (`/odom_frame`) and the map frame (`/map_frame`), as shown in Fig.5.29.

This technique is called "particle-based" because it uses a set of particles to represent different assumptions about the pose of the robot. Each particle simulates a possible combination of the position and orientation of the robot in space. The accuracy of the estimate provided by the AMCL depends on the number of particles used: a higher number of particles tends to improve the accuracy of the localization, but at the same time increases the computational load, with a possible negative impact on the execution speed.

The operation of AMCL can be described by the following steps:

1. Particle Motion: Each particle moves within the map according to the motion data provided, following the odometry and kinematics models of the robot.
2. Update weights: As data is received from sensors (as in our case, from LiDAR and then DLO), this information is used to update the weight or probability associated with each particle.
3. Update weights: As data is received from sensors (as in our case, from LiDAR and then DLO), this information is used to update the weight or probability associated with each particle.

4. Consistency evaluation: Particles that are consistent with sensor measurements will have their weight increased, while those that are inconsistent will have their weight decreased.
5. Convergence: As time passes and measurements accumulate, the particles tend to converge on the actual position of the robot, providing an accurate estimate of its pose.
6. Resampling: Periodically, a resampling process is performed that removes low weight particles while reproducing high weight particles, maintaining particle density in the most likely areas.
7. Initial guess: It is possible to provide an initial guess about the position of the robot. This can speed up the convergence of the AMCL, especially if the guess is close to the actual position of the robot in the environment.

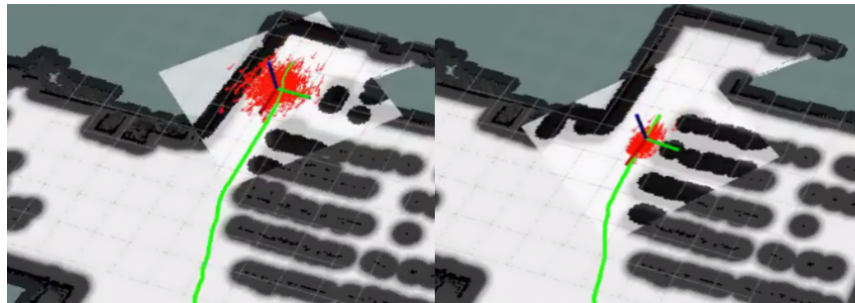


Figure 5.30: Example of AMCL convergence: (L) Low convergence and (R) High convergence

An example of the visualization of the AMCL in Gazebo is shown in Fig.5.30, comparing a low-convergence case with a high-convergence case. This plot shows how the particles are distributed in the environment and how they concentrate around the estimated position of the robot as convergence increases.

5.3.2.5 Move Base for Path Planning

The `move_base` package provides an implementation of an action that, given a goal pose in the reference system `/map`, allows the mobile robot to reach it by optimizing its trajectory. The `move_base` node integrates a global planner and a local planner, providing the ability to plan a trajectory on a global scale and dynamically adapt it locally. This flexibility allows the robot to react to unexpected obstacles not previously identified on the map, ensuring more robust and safer navigation.

The waypoints were set in different ways:

1. RViz is a robot visualization and interaction tool that can be installed on Linux systems and integrated into ROS. In this context, waypoints can be set in real time using an approach commonly referred to as "click and go". By selecting a point on the map through the RViz interface, the robot calculates a trajectory and moves to the indicated location, making the navigation process intuitive and immediate.
2. Using a pre-recorded trajectory: In this mode, the robot is remotely controlled to record a specific trajectory divided into discrete waypoints. The resulting trajectory is stored and can then be executed by the robot in autonomous mode, faithfully replicating the original path.

A block diagram of the navigation stack used, customized for the case under consideration, representing the set of elements used for navigation, and including the `move_base` node, is shown in Fig.5.24. This diagram provides an overview of the operation and interactions between the components of the navigation system.

5.3.3 Validation Through Experiments

5.3.3.1 First Experiment

In the first implemented algorithm, a two-phase approach was adopted for the operation of the robot. In the first phase, the robot was teleoperated to explore the environment and generate a detailed map, which was then stored for future use. In the second phase, this pre-recorded map was used to allow the robot to autonomously navigate the environment, using the information gathered in the previous phase to plan and optimize its movements.

5.3.3.2 Second Experiment

In the second algorithm adopted, the map is generated in real time using the GMapping package. In this case, the position of the robot is defined directly by the DLO, without any correction provided by the AMCL filter. It is important to stress that the main role of AMCL is to compute the transformation between the /base_link and /map frames, correcting the drift (i.e. the progressive deviation from the expected trajectory) that can accumulate over time. In this approach, it is decided to skip the correction offered by AMCL and rely on the transformation between /base_link and /odom computed by DLO. Consequently, the /odom frame is assumed to coincide with /map. This assumption is considered valid in this specific context, since no significant drifts were observed during the experiments.

5.3.4 Configurations for Multi-Agent Collaboration

In a system consisting of two robots configured for multi-agent navigation, a specific configuration of the routers on board each robot was required. In this configuration, the router on the 'master' robot was set as the main router. Conversely, the router on the 'slave' robot was configured to connect to the network created by the master router, acting as a simple switch for the devices connected to the slave. This network configuration allows the two robots to be part of the same Local Area Network (LAN), sharing the same subnet mask. As a result, all devices connected to this network have compatible IP addresses, allowing smooth and continuous communication between them. In real-world applications, this ensures that the robots and their peripherals can effectively interact and exchange data, a necessary condition for successful navigation in multi-agent missions.

An IP table of the above network is shown in Fig.5.31. The configuration network is shown in Fig. 5.32. In this multi-agent navigation algorithm, the role of the "master" robot is crucial. This robot has the task of generating a map of the environment in which it is moving. The map can be acquired in two ways: through a previous teleoperation session during which the master has explored the environment, or it can be generated in real time as it moves. Once the map has been created or acquired, the master starts collecting waypoints, i.e. spatial reference points within the map. These waypoints are then transmitted to the 'slave' robot to provide guidance for its path. During this process, the pose of the master is computed using the DLO algorithm, which provides an accurate

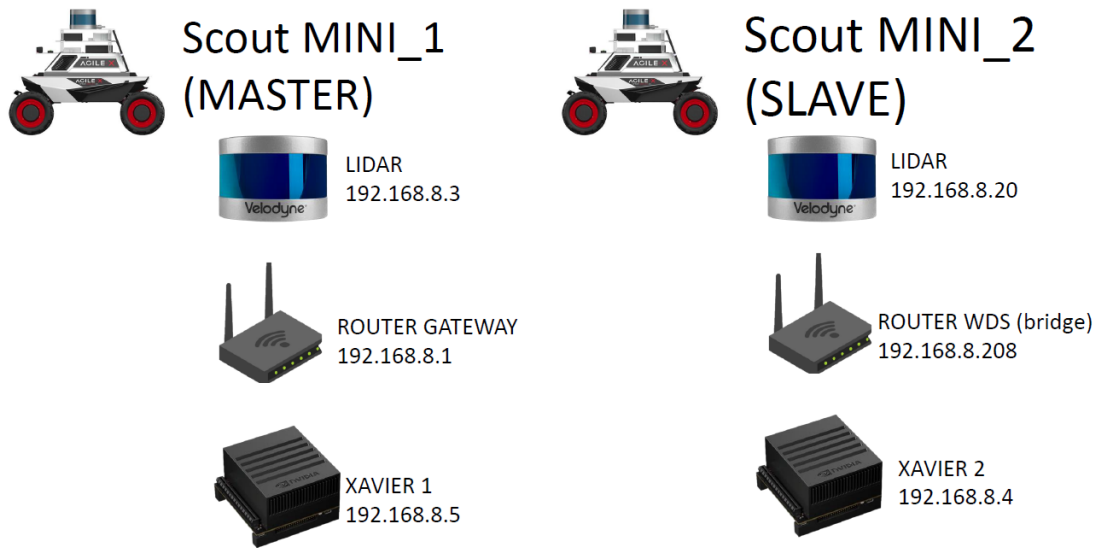


Figure 5.31: IP table of the multi-agent network

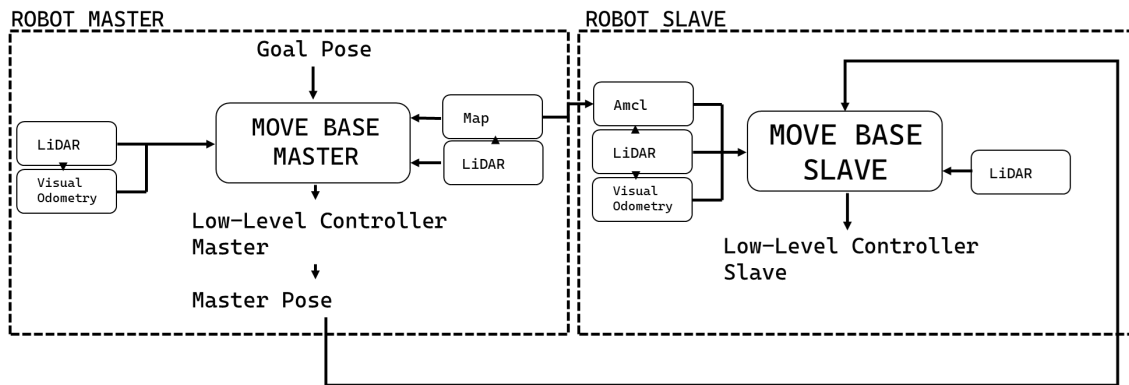


Figure 5.32: Diagram illustrating the overall functioning of the multi-agent in the environment

pose for the master in the map. In parallel, the "slave" robot uses the map created by the master to locate itself. It uses the AMCL filter, which uses a probabilistic approach to estimate the pose of the slave on the map. After receiving waypoints from the master, the slave plans its own path to reach them. While the pose of the master is calculated using DLO, the pose of the slave is estimated using the AMCL algorithm based on the map shared by the master.

The tasks of the two mobile robots are summarized as following:

1. The Master:
 - Map Creation

- Waypoints Collection
- Autonomous Navigation using DLO as Odometric Algorithm

2. The Slave:

- AMCL Localization in the Master-Shared Map
- Path Planning according to received Waypoints
- Autonomous Navigation using DLO and AMCL

5.3.5 Conclusions

This chapter presented applications related to navigation in single-agent and multi-agent scenarios.

In the single-agent case, two possible applications of a Li-SLAM algorithm were presented. In the first case, the localization phase was integrated using the AMCL algorithm, while in the second case, this phase was skipped and relied exclusively on pose reconstruction using DLO. The experimental data showed that in both scenarios, the two custom algorithms are robust in reconstructing the final robot pose. However, when AMCL is not implemented, a progressive drift is observed, especially during long operation sessions or in the presence of fast robot movements. In the future, to improve performance in the absence of AMCL, it may be useful to integrate the IMU data to support DLO and further reduce drift.

In the case of multi-agent, experimental tests showed that the localization of the "slave" robot using the AMCL algorithm, based on the map generated by the "master" robot, is less robust during sudden steering maneuvers.

To face and mitigate this problem, some solutions have been identified to be considered for future developments:

- Cross-Calibration of Maps: This solution could provide accurate calibration between the maps generated by different robots to reduce alignment errors.
- Using the ROS Multirobot Map Merge package: this ROS package enables the fusion of the maps created by the "slave" robots with that of the "master" robot. The heuristic algorithm implemented by the package performs a kind of calibration between the maps, aligning and integrating them. This process ensures that the information is consistent and that the individual maps of the slaves are properly combined into the map of the master.

In general, the algorithms tested in both applications proved to be robust and effective in indoor environments such as a hangar, ensuring reliable and satisfactory performance even in complex scenarios.

5.4 Exploration with Multi-Agent and Collaborative Robotic Systems

5.4.1 Turtlebot 4 Setups and Simulation Environment

This chapter discusses the initial configuration of TurtleBot 4. It also introduces the Rviz2 graphical user interface and the Gazebo simulator used in the thesis. Finally, the way in which the OpenCv library with Aruco was used for the tests performed and the control system adopted will be described.

5.4.1.1 TurtleBot 4 Setup and Implementation

The first step is to set up your PC and robot to establish basic communication. Both devices need to be connected to the same network. To connect the Raspberry Pi to the network, use the SSH protocol, which provides a secure way to access a remote device. In our case, SSH allows you to access the Raspberry Pi to set up the Wi-Fi connection. Once the SSH connection is established, you can access the Raspberry Pi directly from your PC and enter the credentials for the Wi-Fi network. Once the settings have been applied, the Raspberry Pi will reboot and connect to the network. At this point, DHCP (Dynamic Host Configuration Protocol), an essential protocol for dynamically managing IP addresses on a network, will assign a new IP address to the device. On the TurtleBot 4 (Fig. 5.33), the assigned IP address will appear at the top of the display, making it easy to identify and connect to the device.



Figure 5.33: Turtlebot Display

5.4.1.2 Network Configuration

The TurtleBot 4 consists of two main computing units: the Create® 3 and the Raspberry Pi. Although the two units are physically connected to each other, the configuration chosen

allows them to be connected directly to the shared network. This choice allows the units to access the network autonomously, taking full advantage of the features offered by the ROS 2 network and its DDS (Data Distribution Service) communication protocol.

DDS

ROS 2 uses the DDS as the standard for managing internal communications. DDS is a protocol designed to manage real-time data exchange within distributed systems, making it particularly relevant in this context. DDS, and more generally ROS2, facilitates the exchange of data between distributed nodes using a publish-subscribe architecture. Nodes can publish data on specific topics and subscribe to those topics to receive updates. This architecture removes the need for a centralised model and allows the sensors and actuators of the TurtleBot 4 to share information in real time. For example, sensors can continuously publish data about the environment, while control modules can subscribe to this data to make real-time decisions. The DDS also provides advanced Quality of Service (QoS) management, allowing key parameters such as message priority, data persistence and acceptable latency to be defined. This feature is particularly useful in robotic systems where communication requirements can change depending on the modules involved. The ability to configure QoS ensures an optimized flow of information, adapting communication to the specific requirements of each system component.

Simple Discovery

Simple Discovery Protocol (SDP), in Fig. 5.34 is the default protocol defined in the DDS standard. SDP uses multicasting to allow a host to send data packets to multiple recipients on the network simultaneously. In ROS 2, this feature allows devices to automatically discover nodes within a network, facilitating communication between distributed components. In the context of this thesis, the TurtleBot 4 was configured using SDP. This multicast mode requires both the Raspberry Pi 4 and the Create® 3 to be connected to the same Wi-Fi network, ensuring efficient and automatic communication between the devices.

The low-level sensors are all connected to the Create® 3, which, thanks to the configuration adopted, automatically publishes all the topics, services and actions related to the actuators and sensors connected to it. Meanwhile, the Raspberry Pi 4 takes care of managing the high-level sensors, such as the camera and LiDAR, as well as the management nodes, including the custom ones. All this information is published to the ROS 2 distributed network.

This configuration simplifies and automates the process of communication on the network and greatly simplifies the interaction between the components. However, one of the main disadvantages is the rapid consumption of network bandwidth, which can easily lead to congestion situations.

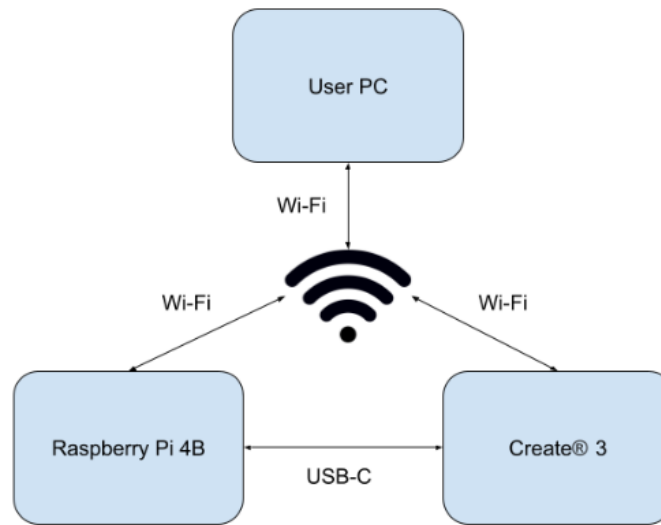


Figure 5.34: Simple Discovery configuration network (109)

5.4.1.3 Collaborative Framework for Multiple Robots

The main goal of this work is to have two Turtlebot4 working in a multi-agent configuration, i.e. simultaneously connected to the same network. By default, each TurtleBot uses the same names for topics, actions and services. Connecting two TurtleBots to the same network without further configuration would cause their topics to conflict, resulting in unwanted behavior.

To solve this problem, each TurtleBot is assigned a namespace. Namespacing consists of adding a unique prefix to the topic names, making them unique for each robot and allowing them to operate correctly within the same network.

In this work, the namespaces used were "tes_robolab" and "doc_robolab", thus ensuring an independent and interference-free operation between the two TurtleBots.

Once the two TurtleBots have been configured and started, it is possible to access a list of all active topics in the system, resulting in the list shown in Fig. 5.35.

Fig. 5.35 shows a list of topics used in the system. Not all topics will be examined, as this is beyond the scope of this work. However, some of them will be analyzed in detail to highlight their crucial role in the navigation and movement of mobile robots within the ROS 2 framework:

- `/cmd_vel` (Fig.5.36): The topic used to send command velocity information to the TurtleBot 4 transmits data in the form of linear velocity and angular velocity. In the case of a differential robot like the TurtleBot 4, only the velocity along the x-axis (linear motion) and the angular velocity around the z-axis (rotation) are relevant.

```

pasq-stl@pasq-stl:~$ ros2 topic list
/clock
/doc_robotlab/battery_state
/doc_robotlab/bumper_contact
/doc_robotlab/cmd_audio
/doc_robotlab/cmd_lightning
/doc_robotlab/cmd_vel
/doc_robotlab/diffdrive_controller/cmd_vel_unstamped
/doc_robotlab/diffdrive_controller/transition_event
/doc_robotlab/dock_status
/doc_robotlab/dynamic_joint_states
/doc_robotlab/function_calls
/doc_robotlab/hazard_detection
/doc_robotlab/hmi/buttons
/doc_robotlab/hmi/display
/doc_robotlab/hmi/display/message
/doc_robotlab/hmi/led
/doc_robotlab/interface_buttons
/doc_robotlab/ip
/doc_robotlab/ir_intensity
/doc_robotlab/ir_opcode
/doc_robotlab/joint_state_broadcaster/transition_event
/doc_robotlab/joint_states
/doc_robotlab/joy
/doc_robotlab/kidnap_status
/doc_robotlab/mouse
/doc_robotlab/oakd/rgb/preview/camera_info
/doc_robotlab/oakd/rgb/preview/depth
/doc_robotlab/oakd/rgb/preview/depth/points
/doc_robotlab/oakd/rgb/preview/image_raw
/doc_robotlab/odom
/doc_robotlab/robot_description
/doc_robotlab/scan
/doc_robotlab/sin_ground_truth_dock_pose
/doc_robotlab/sin_ground_truth_pose
/doc_robotlab/slip_status
/doc_robotlab/standard_dock_description
/doc_robotlab/stop_status
/doc_robotlab/tf
/doc_robotlab/tf_static
/doc_robotlab/wheel_status
/doc_robotlab/wheel_ticks
/doc_robotlab/wheel_vels
/parameter_events
/rosout
/tes_robotlab/battery_state
/tes_robotlab/bumper_contact
/tes_robotlab/cmd_audio
/tes_robotlab/cmd_lightning
/tes_robotlab/cmd_vel
/tes_robotlab/diffdrive_controller/cmd_vel_unstamped
/tes_robotlab/diffdrive_controller/transition_event
/tes_robotlab/dock_status
/tes_robotlab/dynamic_joint_states
/tes_robotlab/function_calls
/tes_robotlab/hazard_detection
/tes_robotlab/hmi/buttons
/tes_robotlab/hmi/display
/tes_robotlab/hmi/display/message
/tes_robotlab/hmi/led
/tes_robotlab/interface_buttons
/tes_robotlab/ip
/tes_robotlab/ir_intensity
/tes_robotlab/ir_opcode
/tes_robotlab/joint_state_broadcaster/transition_event
/tes_robotlab/joint_states
/tes_robotlab/joy
/tes_robotlab/kidnap_status
/tes_robotlab/mouse
/tes_robotlab/oakd/rgb/preview/camera_info
/tes_robotlab/oakd/rgb/preview/depth
/tes_robotlab/oakd/rgb/preview/depth/points
/tes_robotlab/oakd/rgb/preview/image_raw
/tes_robotlab/odom
/tes_robotlab/robot_description
/tes_robotlab/scan
/tes_robotlab/sin_ground_truth_dock_pose
/tes_robotlab/sin_ground_truth_pose
/tes_robotlab/slip_status
/tes_robotlab/standard_dock_description
/tes_robotlab/stop_status
/tes_robotlab/tf
/tes_robotlab/tf_static
/tes_robotlab/wheel_status
/tes_robotlab/wheel_ticks
/tes_robotlab/wheel_vels

```

Figure 5.35: ROS 2 topic list for multiple robots

```

pasq-stl@pasq-stl:~$ ros2 topic echo /doc_robotlab/cmd_vel
linear:
  x: 1.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
linear:
  x: 1.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: -0.5

```

Figure 5.36: Output of the /cmd_vel topic in ROS 2, displaying linear and angular velocity commands for controlling the movement of the robot

- /scan (Fig. 5.37): The topic provides the data collected by the LiDAR sensor, representing the distances of objects from the sensor. The accuracy and sensitivity of the measurements are intrinsic parameters contained in the topic message and are

available for further processing.

```

pasq-sti@pasq-sti:~$ ros2 topic echo /doc_robotlab/scan
header:
  stamp:
    sec: 869
    nanosec: 58000000
  frame_id: doc_robotlab/turtlebot4/rplidar_link/rplidar
angle_min: -3.1415927410125732
angle_max: 3.1415927410125732
angle_increment: 0.009832840412855148
time_increment: 0.0
scan_time: 0.0
range_min: 0.164000004529953
range_max: 12.0
ranges:
- 7.552528381347656
- 7.540840148925781
- 7.530879974365234
- 7.523434638977051
- 7.519085884094238
- 7.518528938293457

```

Figure 5.37: Output of the /scan topic in ROS2

- /odom (Fig. 5.38): The topic publishes the pose of the robot in a reference frame called /odom_frame, which should be fixed. The pose is represented as a vector containing the position of the robot and a quaternion describing its orientation in space.

```

pasq-sti@pasq-sti:~$ ros2 topic echo /tes_robotlab/odom
header:
  stamp:
    sec: 264
    nanosec: 783000000
  frame_id: /tes_robotlab/odom
child_frame_id: /tes_robotlab/base_link
pose:
  position:
    x: -0.3527712944927644
    y: -3.199247556345313e-11
    z: 0.0
  orientation:
    x: -0.0
    y: 0.0
    z: 0.9952558368063814
    w: -0.09729244216705353

```

Figure 5.38: Output of the /odom topic in ROS2

- /tf e /tf_static: Topics publish both dynamic and static coordinate transformations to represent and update the spatial relationships between the different components of the robot and the transformations of the reference system of the robot and the /odom_frame.

-
- /D455_1/color/camera_info: Provides color camera calibration information, including intrinsic matrix, distortion, and other parameters needed for geometric correction of color images.
 - /D455_1/color/image_raw: Publishes raw color images captured by the RealSense D455 RGB camera. These images are unprocessed and contain direct visual information from the camera.
 - /D455_1/depth/camera_info: Provides depth sensor calibration information, including intrinsic parameters and distortion, required for geometric correction of depth images.
 - /D455_1/depth/image_rect_raw: Publishes corrected depth images captured by the RealSense D455. These images show the distance of each pixel from the camera.

5.4.1.4 Using Ignition Gazebo and Rviz2 for Visualization

Ignition Gazebo is an open-source simulator designed to provide realistic robot simulations in 3D environments. It is part of the Ignition open-source robotics project developed by Open Robotics (Fig.5.39). Gazebo is used to develop and test control algorithms, sensors,



Figure 5.39: Graphical interface of Gazebo

and robot behavior. One of the key components of Gazebo is its physics engine which simulates the physical behavior of objects in the virtual world. The physics engine computes the motion and interaction of objects, taking into account their physical properties such as mass, inertia and friction, to ensure a realistic simulation.

While Gazebo is a simulator, Rviz (in its updated version Rviz2 for ROS 2) is a 3D visualization tool used to analyze and visualize the state of the robot and its perception of the environment.

Through the graphical user interface of Rviz2 (Fig.5.40), it is possible to select and customize the information to be displayed, such as the state of the robot or the measurements provided by the sensors, thus facilitating real-time monitoring and analysis. Rviz2 can be used both when the robot is working in a simulated environment in Gazebo and in a real context. The combined use of Gazebo and Rviz2 allows to obtain a complete view of the operation of the application on the robot. This approach makes it possible to quickly compare new features and changes, and to analyze the behavior of the robot in different working conditions.

This integration is particularly useful for localization, obstacle avoidance and path planning applications. For example, as shown in Fig.5.40, by subscribing to the /scan topic, it is possible to reconstruct a map of the obstacles surrounding the robot. Similarly, by subscribing to the camera topic, it is possible to view in real time the image captured by the camera itself.

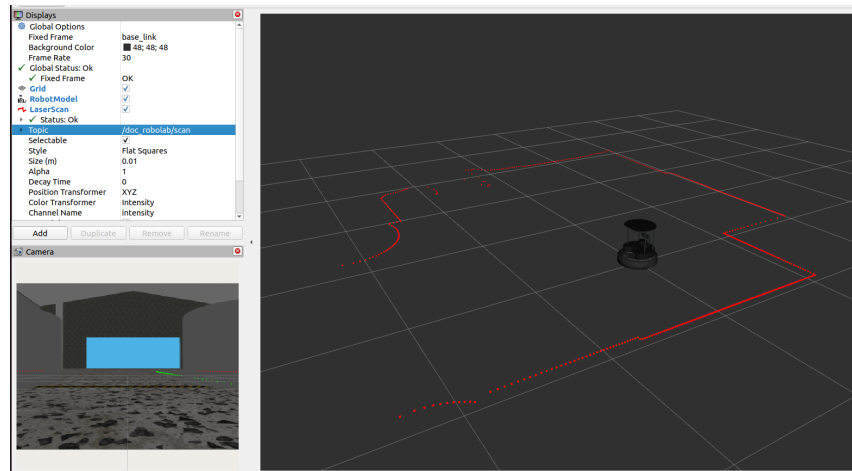


Figure 5.40: Rviz2

5.4.1.5 Teleoperation for a Single Turtlebot4

Teleoperation is the sending of commands from an input device, such as a joystick or keyboard, to the TurtleBot 4. A human operator sends the commands, which are then translated into actions that the robot can perform, allowing for direct and immediate control.

Through the keyboard, the robot can be controlled using the ROS package *teleop twist keyboard* (Fig. 5.41) designed to control any mobile robot. Although this package is designed for even more complex systems, in the case of TurtleBot 4, a differential robot, its physics simplify the analysis of the motion. In fact, given a velocity and direction command, the resulting motion can be easily understood by inserting these commands into the kinematic model of the differential robot. As already mentioned, behind the speed command there is the concept of *twist vector*. Although formally different, this concept is widely used in ROS 2 to define the motion of the robot. The twist vector represents the composition of the motion by two components: the linear velocity and the angular velocity. In "twist" messages, the linear component defines the velocities along the x , y and z axes, while the angular component specifies the rotation velocities around the same axes.

In this context, the keyboard becomes a direct input device for publishing motion commands through the "cmd_vel" topic, allowing an intuitive and interactive control of TurtleBot 4 during the experimental phases. Starting the teleoperation node with the keyboard leads to the schematization of the ROS 2 network, as shown in Fig. 5.42. The mobile robot can also be controlled using a Bluetooth joystick (Fig. 5.43).

5.4.1.6 Leader-Follower Configurations in Multi-Agent Systems

In the context of this work, one of the two TurtleBots was designated as the leader (with namespace "doc_robolab"), while the other, with namespace "tes_robolab", was

```

This node takes keypresses from the keyboard and publishes them
as Twist messages. It works best with a US keyboard layout.
-----
Moving around:
  u    l    o
  j    k    l
  m    ,    .

For Holonomic mode (strafing), hold down the shift key:
-----
  U    I    O
  J    K    L
  M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5      turn 1.0

```

Figure 5.41: Keyboard teleoperation interface

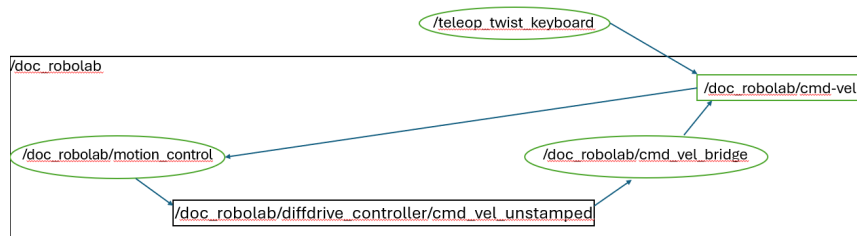


Figure 5.42: Rqt graph



Figure 5.43: Bluetooth joystick utilized for teleoperation

designated as the follower (Fig. 5.44). The main goal was to allow the TurtleBot follower to dynamically follow the TurtleBot leader.

To achieve this behavior, a visual marker-based position tracking system was implemented. In particular, an ArUco marker was used as a reference point to estimate the position and orientation of the leader. Thanks to this approach, the TurtleBot follower is able to adapt to the position variations of the leader while maintaining a stable and coherent tracking trajectory.



Figure 5.44: Leader and follower setup with Turtlebot4 robots

5.4.1.7 Aruco Marker-Based Localization and Navigation

To perform the experiments, "sight" navigation was basically used, i.e. by exploiting a peculiarity of the leader robot, the follower robot adapts its movement to that of the leader robot by exploiting its vision skills. To achieve this, an artifice was used: an ARTag (Aruco Marker) was attached in order to track the pose of the aruco and thus check the distance of the camera on the follower compared to the aruco on the leader. An aruco marker is composed of a black border that defines its perimeter and facilitates detection, and an internal binary matrix that determines the identifier (id) of the marker. The markers are grouped into sets, called dictionaries, based on their size, which also determines the size of the internal matrix (110). In this work, for example, the dictionary composed of 250 6x6 markers (DICT_6X6_250) was used, and in particular the aruco with the id 23 (Fig. 5.45). With the help of OpenCV and its functions, it is possible to easily obtain the pose of the traced ArUco from any camera. In general, the marker detection process consists of two main steps (110):

1. **Marker Candidate Detection:** in this step, the image is analyzed to find candidate square shapes to be markers. It starts with adaptive thresholding to segment the markers; then it extracts edges from the thresholded image and discard those that are not convex or do not approximate a square shape.

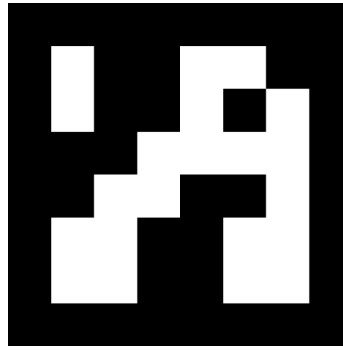


Figure 5.45: ArUco, id = 23, DICT_6X6_250 (110)

2. After identifying the candidates, it is necessary to establish whether they are actually markers by analyzing their internal encoding. This step starts by extracting the bits of each marker. To do this, a perspective transform is first applied to obtain the marker in its canonical form, as shown in Fig. 5.46 Then, the canonical image is



Figure 5.46: Prospective removal (110)

divided into a grid with a number of cells equal to the number of bits in the marker. In each cell, the number of black and white pixels is counted to determine the bit value assigned to the cell (Fig. 5.47).

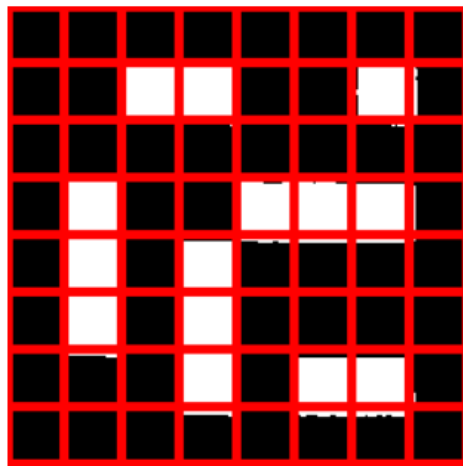


Figure 5.47: Marker Cells (110)

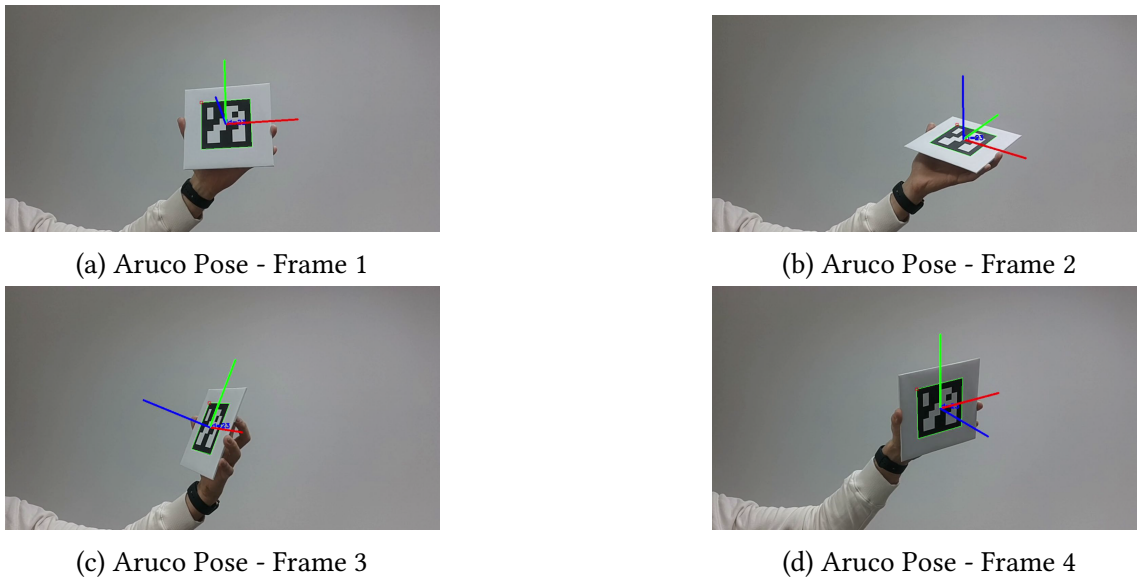


Figure 5.48: Example of Aruco Poses

5.4.1.8 Pose Estimation Techniques for Robotic Systems

After detecting the markers, the next step was to obtain the camera pose. Pose estimation is of great importance in many computer vision applications, such as robot navigation, augmented reality, and many others.

To estimate the camera pose, it is necessary to have calibration parameters, including the camera matrix and distortion coefficients. Relative pose is a 3D transformation that connects the marker coordinate system with the camera coordinate system. This transformation is described by two components: translation, which is the distance between the origins of the two systems, and rotation, which is represented by the angles between the corresponding axes of the coordinate systems. This information together defines the pose change (Fig.5.48).

5.4.1.9 Camera Frame-to-Base Link Calibration

Once the pose of the ArUco is computed in the camera reference system. However, speed commands are given in the base_link reference system, so it is necessary to transform the pose from the camera reference system to the base_link reference system (Fig.5.49). The /tf_static topic is used in ROS to publish static transformations between frames of different reference systems within the robotic environment. This topic is generally used to post transformations that remain constant over time, i.e., static transformations with respect to the execution of the ROS node. Transformations represent the position and orientation of sensors, cameras, or other fixed elements in a robotic system. These transformations are defined in a YAML configuration file and published in the /tf_static topic, making them available to other ROS nodes in the system. Among these transformations is the static transformation between the camera reference system and the base_link reference

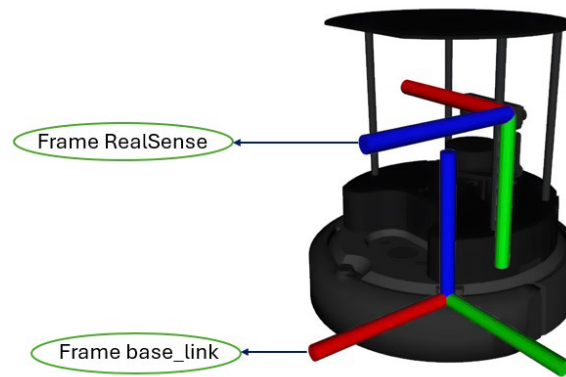


Figure 5.49: Representation of the transformation from the RealSense frame to the base link frame in the robotic system

system. It is important to note that there are several different camera reference systems; the correct one is the one in which the images are computed. By definition, this system is easy to identify: the z-axis is always perpendicular to the camera plane. By analyzing the transformation in `/tf_static`, it gets the representation shown in Fig.5.50.

```
^Cpasq-stl@pasq-stl:~/ros2_ws$ ros2 run tf2_ros tf2_echo base_link
[INFO] [1708345559.647967251] [tf2_echo]: Waiting for transform b
t
At time 0.0
- Translation: [-0.060, 0.000, 0.244]
- Rotation: in Quaternion [-0.500, 0.500, -0.500, 0.500]
- Rotation: in RPY (radian) [-1.571, -0.000, -1.571]
- Rotation: in RPY (degree) [-90.000, -0.000, -90.000]
- Matrix:
  0.000  0.000  1.000 -0.060
 -1.000  0.000  0.000  0.000
  0.000 -1.000  0.000  0.244
  0.000  0.000  0.000  1.000
```

Figure 5.50: Transformation matrix detailing the translation and rotation between frames

5.4.1.10 Collaborative Logic Development in Multi-Agent Systems

Knowing the translation vector between the camera and the ArUco marker, it is possible to calculate both the distance between the two TurtleBots and the angle formed between the line connecting the two robots and the forward direction of the TurtleBot follower (Fig. 5.51). The distance can be easily computed by this:

$$d = \sqrt{X_B^2 + Y_B^2} \quad (5.10)$$

The angle α between the conjunction of the two robots and the forward direction of the TurtleBot follower was calculated using the 'atan2' function. This function is an extension

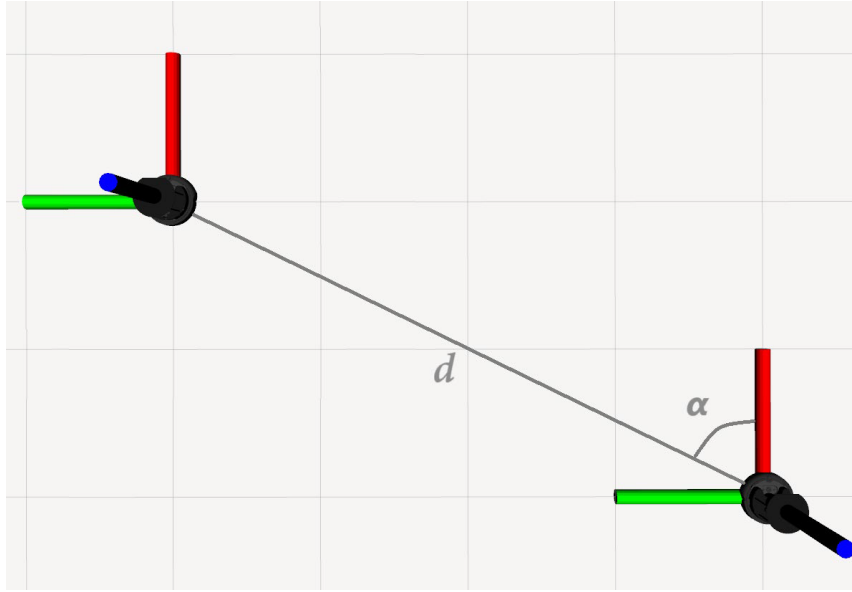


Figure 5.51: Controlled Parameters: d and α

of the 'arctan' function, since it is able to distinguish between diametrically opposed angles, taking into account both the ratio of the arguments and their sign.

$$\alpha = \text{atan2}(Y_B, X_B) \quad (5.11)$$

Within the control, the variables ' d ' and ' α ' represent the parameters to be adjusted. The goal is to keep the distance ' d ' equal to a target value and to reduce the angle ' α ' to zero.

Two different approaches in the field of automatic control have been adopted in the thesis: PID control and Sliding Mode Control.

5.4.1.11 Control Logic for Multi-Robot Coordination

The TurtleBot system can be structured as a result of combining two subsystems in a cascade, as shown in Fig. 5.52. The control system has two levels: a high-level control (shown in blue) and a low-level control (shown in red). In the high-level control, the error is calculated as the difference between the target distance and the feedback distance and is provided as input to the C_1 controller. The output of C_1 , representing a reference velocity (v_{ref}), is the input for low-level control.

Low level control is so defined because it acts directly on the drive, providing the electrical voltage needed to drive the motors. The difference between the reference value v_{ref} and the feedback rate v_x is the input for a second controller, C_2 . This controller outputs the voltage that drives the motors of the TurtleBot to reach the desired speed.

It is important to note that the low-level control is a speed-based control. Consequently, it does not take into account any slippage of the wheels of the TurtleBot. This control represents the standard control in the firmware of the robot.

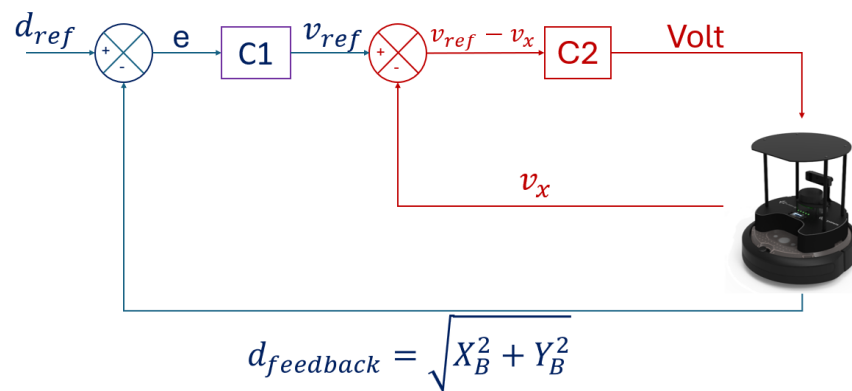


Figure 5.52: Control loop diagram showcasing the calculation of feedback distance $d_{feedback}$ and the voltage adjustment for maintaining desired robot distance

In this work, however, two different control laws have been assigned to the C_1 controller: the PID control and the sliding mode control.

5.4.2 Test and Discussion

Three main applications are presented and discussed in this chapter:

1. **Chasing Test:** keep a target distance between mobile robots.
2. **Law of Motion Test:** Apply a law of motion to the TurtleBot leader and analyze the behavior of the TurtleBot follower.
3. **Obstacle Avoidance Test:** Evaluate the ability of the system to avoid obstacles.

For each of these applications, the system response will be analyzed in both simulation and experimentation, highlighting the differences between the two approaches.

5.4.2.1 Chasing Test

In the first test, the leader TurtleBot was controlled by teleoperation, while the follower TurtleBot was required to follow it.

During the experiment, a target value for the PID of the TurtleBot follower was configured to be 0.8 meters, representing the distance to be maintained from the leader. The TurtleBot follower was programmed to keep a constant distance of 0.8 meters during the movement of the leader in this way.

The test was performed in both simulation and real experimentation. The results include the trajectories followed by the two TurtleBots, with error and percentage error analysis between the target and feedback distance. Finally, the results obtained in simulation and in the real environment will be compared to highlight differences and similarities between the two execution modes.

Simulation in Gazebo

The test was first conducted in the Gazebo simulation environment. Fig.5.53 shows the storyboard of the simulation conducted, illustrating the different phases and movements of the two TurtleBots during the simulation. Fig.5.53 shows the trajectories of TurtleBot Leader and TurtleBot Follower, saved locally in ROS 2 and then analyzed using Matlab.

The plot of the trajectories of the two TurtleBots shows that the TurtleBot follower succeeds in following the telepiloted TurtleBot leader, but tends to "anticipate" the trajectory of the leader. This behavior is normal, since the control is based solely on the distance between the two robots and not on their trajectory. Fig. 5.55 shows the error between the target distance assigned to the PID of the TurtleBot follower and the measured feedback distance. The plot shows that the error between the target distance and the measured distance is relatively small, indicating that the control system is effective in maintaining the desired distance from the leader TurtleBot. This positive result shows a good tuning of the PID controller and an adequate response of the TurtleBot follower to the commands received.

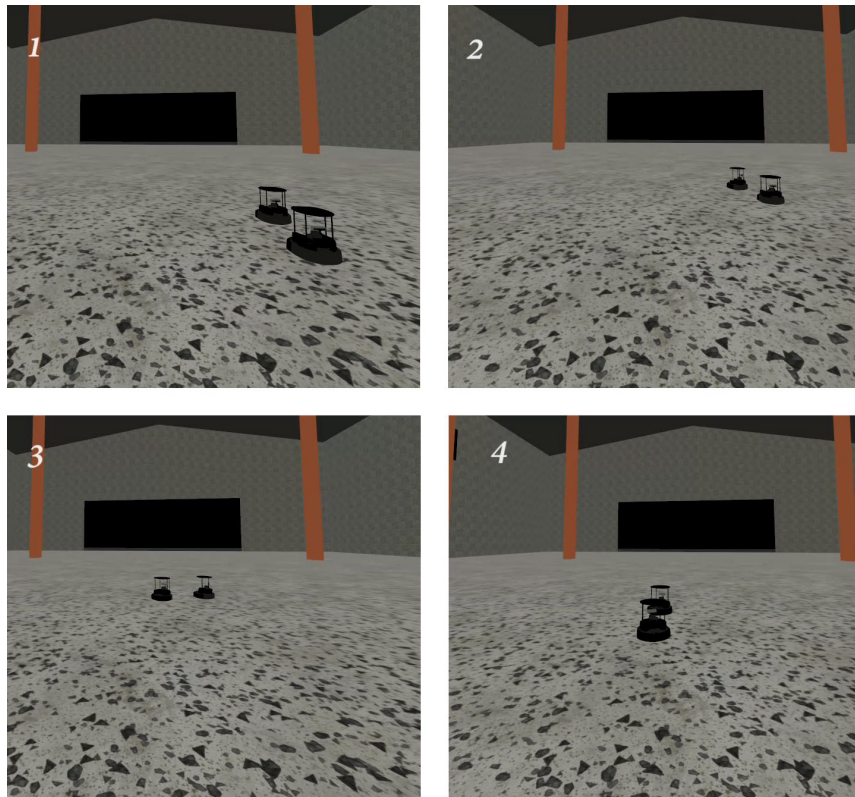


Figure 5.53: Chasing Test: Gazebo Simulation

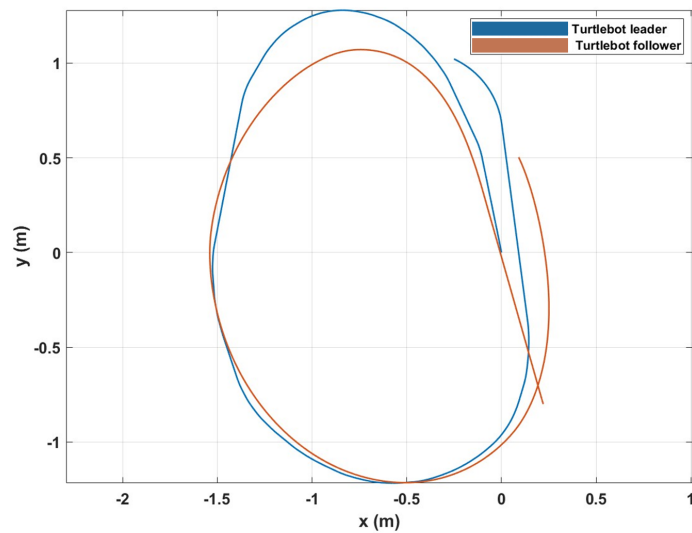


Figure 5.54: Chasing Test: Simulation Trajectories

Test

After running the test in the Gazebo simulation environment, it was replicated in the

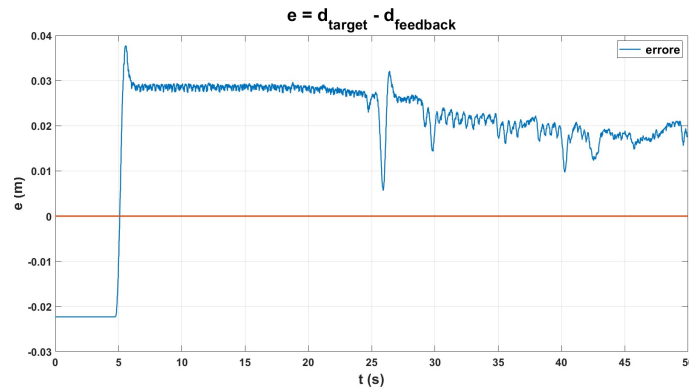


Figure 5.55: Chasing Test: Error

real experiment to determine any differences between the two modes. Figure 5.56 shows the storyboard of the test performed in the experiment. The joystick connected to the

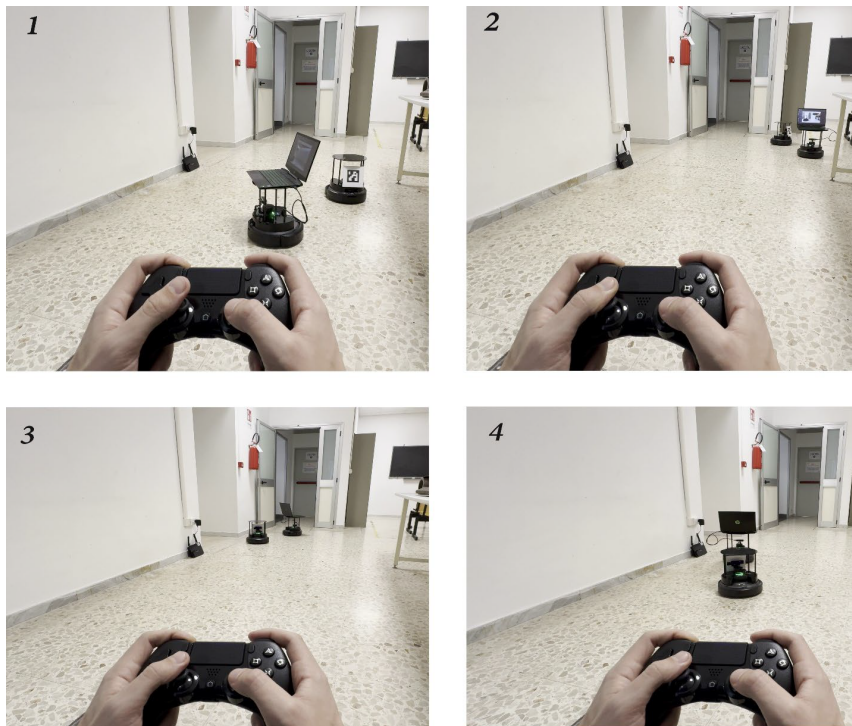


Figure 5.56: Chasing Test: Test sequence

leader TurtleBot via Bluetooth was used for remote control. A plot of the trajectories followed by the TurtleBot leader and TurtleBot follower during the test is shown in Fig. 5.57. In the experimental setting as well as in the simulation it is observed that the TurtleBot follower succeeds in following the TurtleBot leader. However, the trajectory of the follower differs more from that of the leader in the experimental setting than in the simulation. Possible reasons for these discrepancies include factors such as ground

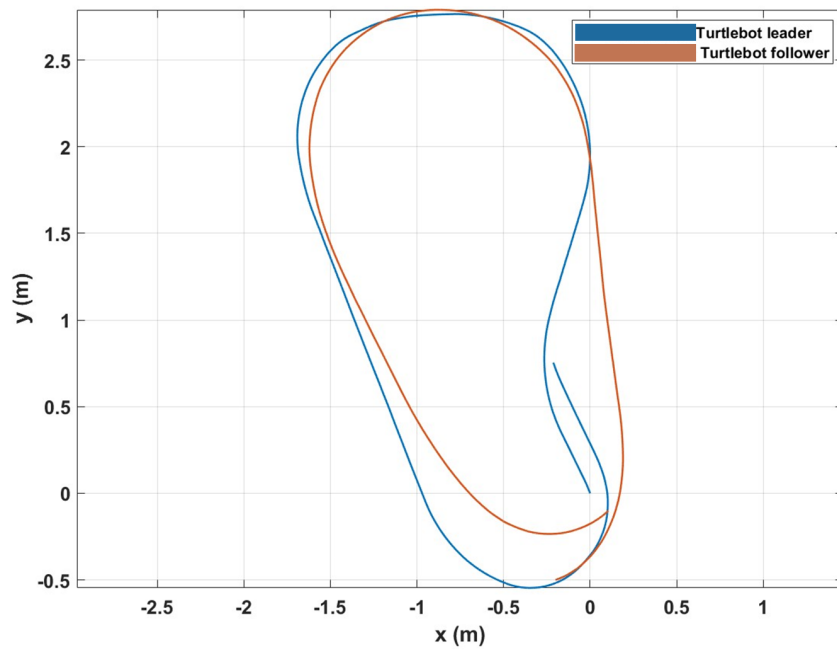


Figure 5.57: Chasing Test: Trajectories in Test

conditions or the presence of additional mass, and thus greater inertia, on the follower robot. Fig. 5.58 shows the error between the target distance assigned to the PID and the

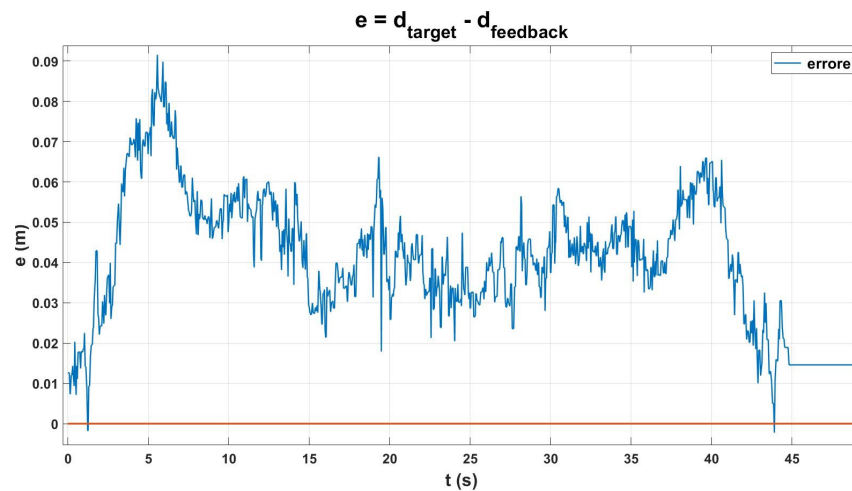


Figure 5.58: Chasing Test: Error in Test

measured feedback distance.

Even in the experiment, the error between the target distance and the measured distance is relatively small. However, as shown in the trajectory analysis, this error was

higher than the error calculated during the simulation in the Gazebo environment, as previously mentioned.

Simulation vs Test

To compare the results obtained in the Gazebo simulation environment and in the experiment, it is considered the percentage error calculated as follows:

$$e_{\%} = \frac{|d_{\text{feedback}} - d_{\text{target}}|}{d_{\text{target}}} \cdot 100 \quad (5.12)$$

Fig. 5.59 shows the percentage error found in the Gazebo simulation environment. While the experimental phase in Fig. 5.60.

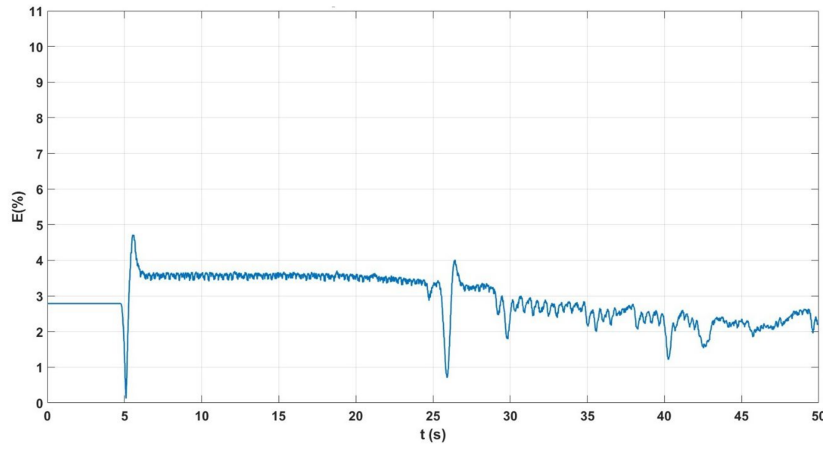


Figure 5.59: Chasing Test: Percentage Error Gazebo

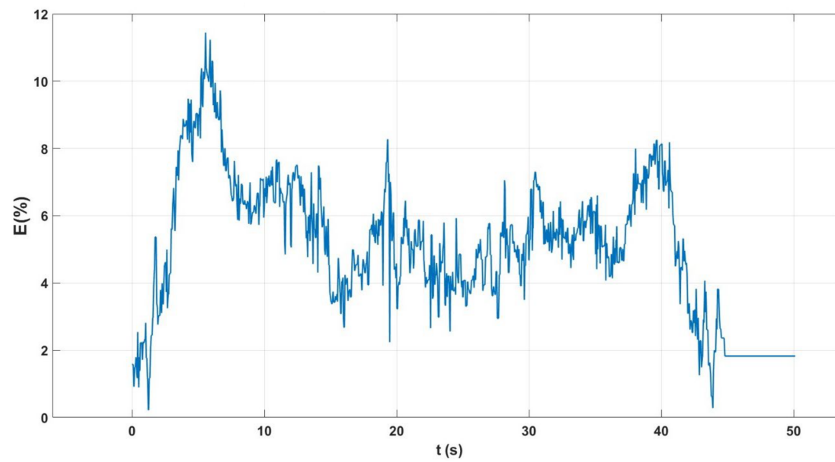


Figure 5.60: Chasing Test: Percentage Error in Test

Some performance indices are used to compare the data:

- MAE (Mean Absolute Error): is the average of the absolute differences between the target and feedback values.

$$\text{MAE} = \frac{\sum_{i=1}^n |d_{\text{feedback}} - d_{\text{target}}|}{n} \quad (5.13)$$

- MAPE (Mean Absolute Percentage Error): measures the average percentage of absolute differences between the target and feedback values from the target values.

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^n \left(\frac{|d_{\text{feedback}} - d_{\text{target}}|}{d_{\text{target}}} \right) \cdot 100 \quad (5.14)$$

- Standard Deviation: measures dispersion that indicates how far the data in a set deviate on average from their mean.

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (d_{\text{feedback}} - \bar{x})^2}{n}} \quad (5.15)$$

- RMSE (Root Mean Square Error): is the square root of the mean of the squared errors between the feedback and the target.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^n (d_{\text{feedback}} - d_{\text{target}})^2}{n}} \quad (5.16)$$

In Tab. 5.1, these performance indices were calculated in relation to the error between the target distance assigned to the PID and the measured feedback distance.

Performance Index	Gazebo Simulation	Experimentation
MAE [m]	0.0237	0.0394
MAPE [%]	2.9645	4.9212
Standard Deviation [m]	0.0107	0.0159
RMSE [m]	0.0242	0.0425

Table 5.1: Chasing Test: Gazebo vs Experimentation

In the Gazebo environment, the results show an overall higher accuracy than in the experiment. All metrics such as MAE , MAPE , Standard Deviation and RMSE are lower in Gazebo, indicating a smaller discrepancy between the target and actual distance of the two TurtleBots.

In the real world, the presence of complex and dynamic variables presents challenges that are difficult to model accurately. Factors such as friction, terrain irregularities, variations in environmental conditions, and other unpredictable elements can contribute

to the observed discrepancies. In addition, differences in the hardware used, such as less accurate or less than ideal sensors, can amplify these discrepancies.

Real-world measurements are often subject to noise, calibration errors, and other uncertainties, while the simulator tends to generate more ideal or less noisy data. Finally, network problems or latency can also affect experimental results, further contributing to the discrepancies found compared to simulation.

5.4.2.2 Law of Motion Test

In the second test, a trapezoidal profile velocity law was assigned to the leading TurtleBot (Fig.5.61) in order to analyze the behavior and the law of motion adopted by the follower TurtleBot. The law of motion consists of three main phases:

- In the first 4 seconds, the motion is accelerated uniformly, with an increase in speed up to 0.2 m/s.
- In the next 4 seconds, the motion becomes uniformly rectilinear, with a constant velocity of 0.2 m/s.
- In the last 4 seconds, the motion is uniformly decelerated, bringing the robot back to zero speed.

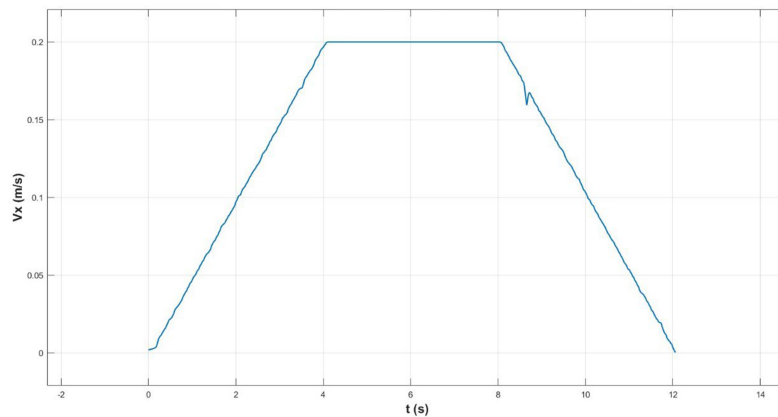


Figure 5.61: Law of motion for Turtlebot leader

Simulation in Gazebo

First, a purely proportional control was considered, with a value of $K_p=7$. With this configuration, the measured error between the feedback distance and the target distance during the simulation is shown in Fig. 5.62. To evaluate the effectiveness of the control system, it is important to consider not only the absolute error, but also the percentage error (Fig. 5.63). The latter provides a relative measure of the inaccuracy of the system,

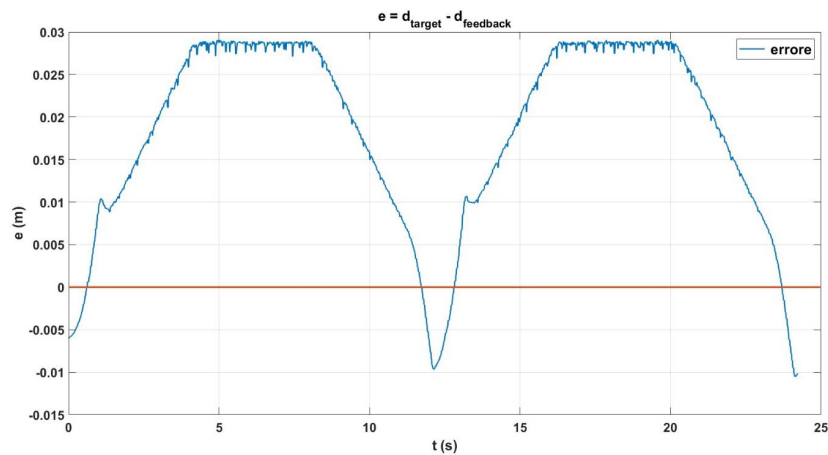


Figure 5.62: Law of Motion Test: Error in Simulation

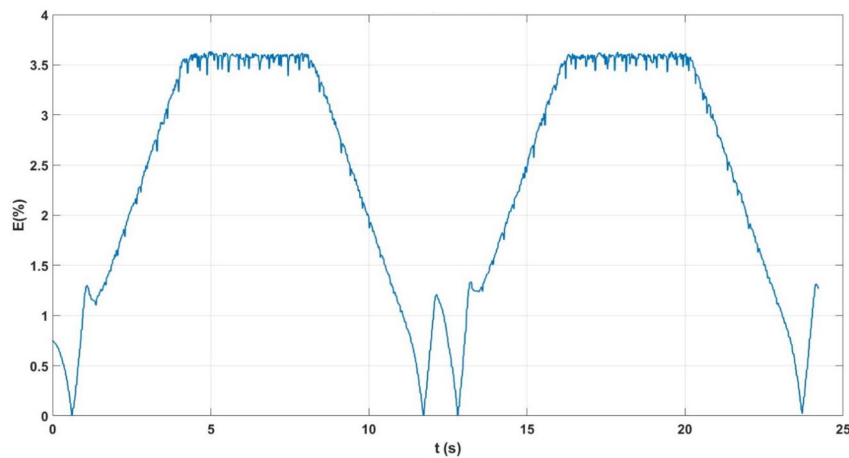


Figure 5.63: Law of Motion Test: Percentage Error in Simulation

taking into account the size of the time step, providing a more complete evaluation of performance. MAPE of 2.51%, which is relatively low and can be considered acceptable in the scope of the application. This result indicates good accuracy in controlling the position of the two mobile robots with respect to the desired target distance of 0.8 meters.

Fig. 5.64 shows the linear velocity trend V_x of the two TurtleBots.

It can be seen that the TurtleBot follower succeeds in acceptably emulating the law of motion of the TurtleBot leader. However, if one considers a purely proportional control, one can see in Fig. 5.65 oscillations in the speed trend of the TurtleBot follower.

Instead of introducing the derivative component to correct for system oscillations, with the risk of increasing instability, it was decided to add an integrative component to mitigate this phenomenon. The goal is to reduce the overshoot by taking advantage of the integrative part, trying to mitigate the drop from the maximum overshoot and thus

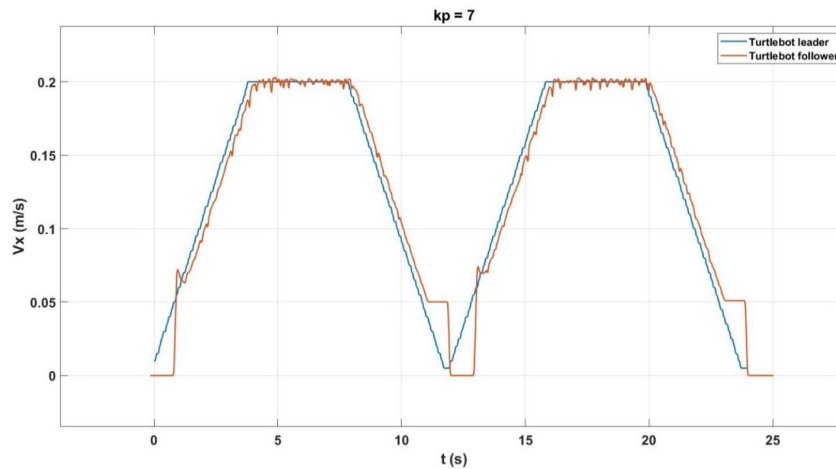


Figure 5.64: Law of Motion Test: Velocity along x robot axis



Figure 5.65: Law of Motion Test: Zoom in for velocity along x robot axis V_x trend

reduce the oscillations between phases.

Specifically, the PID of the TurtleBot follower was assigned a proportional gain of 6 and an integrative gain of 1. With this configuration, the velocity profile of the two mobile robots is shown in Fig. 5.66. The error between the actual and target distances of the two mobile robots, recorded with proportional-integral control, is shown in Fig. ??, while the percentage error is shown in Fig. 5.68. In this case, the MAPE of 1.29% turns out to be even lower than the one calculated with proportional control.

There is one really important point to note: TurtleBot4 was not allowed to move in the negative direction of linear velocity in any of the experiments. So the control is done on the distance, but only on the forward direction of the robot. For this reason, as shown in the error graph, the follower always has to wait until the leader exceeds the target distance in order to move.

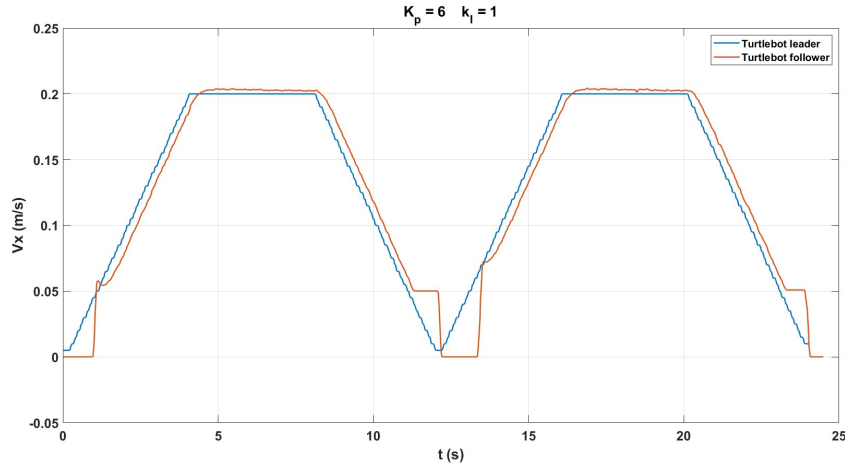
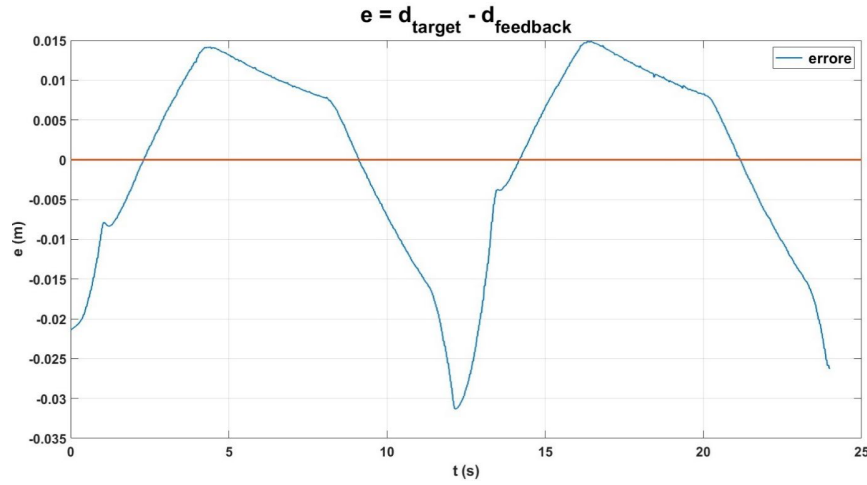
Figure 5.66: Law of Motion: V_x under PI Control

Figure 5.67: Law of Motion: Error under PI Control

Test with PID

In this experimental phase, a PID was used to control the TurtleBot follower. Specifically, the gains assigned to the controller were $K_p = 4$ and $K_i = 1.5$. Analyzing the velocity profile of the two TurtleBots in Fig. 5.70, it is observed that the TurtleBot follower manages to acceptably emulate the law of motion of the TurtleBot leader even during the experiment. However, as expected, the oscillations are larger than those observed in the simulator. In Fig.5.71, the error between the target and feedback distances is higher than that calculated during the simulation in Gazebo, with a maximum distance of 0.0754 m. In Fig.5.72, the percentage error is shown.

Fig.5.72 shows the percentage error. Note that the MAPE during the experiment is 2.160%, a higher value than the 1.29% calculated in the simulation.

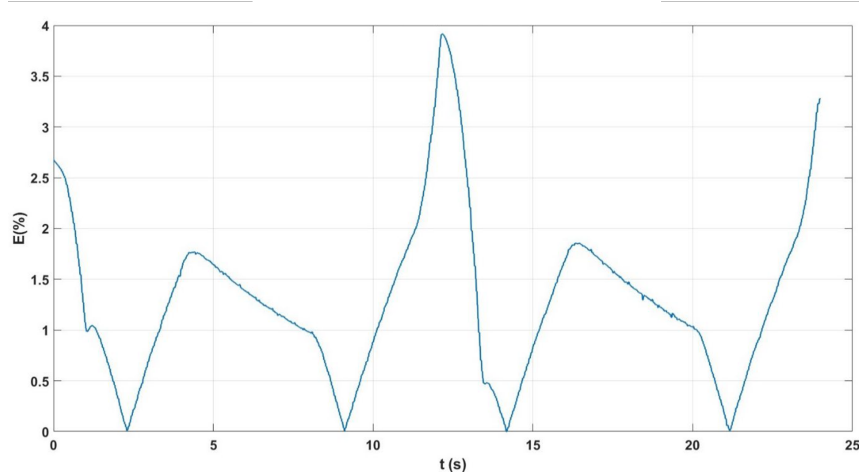


Figure 5.68: Law of Motion: Percentage Error under PI Control



Figure 5.69: Law of Motion: Test under PI

Simulation vs Test with PID

To compare the results of the test performed in simulation and experiment, it is possible to analyze the speed difference between TurtleBot leader and TurtleBot follower for each time point.

For the simulation, the difference is shown in fig. 5.73, while for the experimental test the result is shown in Fig. 5.74. The maximum difference between the speeds of the two TurtleBots in the simulation phase was 0.06 m/s, while in the test phase it was about 0.1 m/s.

In the Tab. 5.2, the performance indexes are shown.

In the simulation case, the MAE of 0.0121 indicates that the speed of the follower differs on average by about 0.0121 m/s from the speed of the leader trend. This is a relatively small error. In addition, the standard deviation and RMSE, both around 0.0161, show good consistency on the part of the TurtleBot follower in following the velocity of

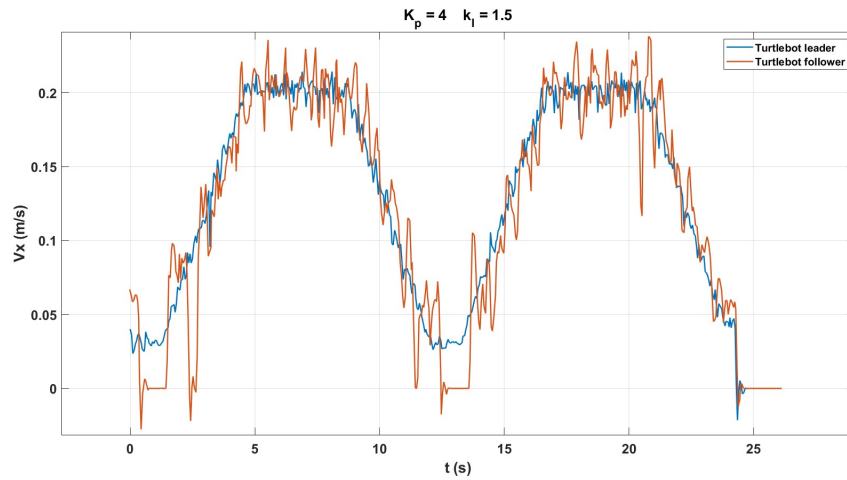
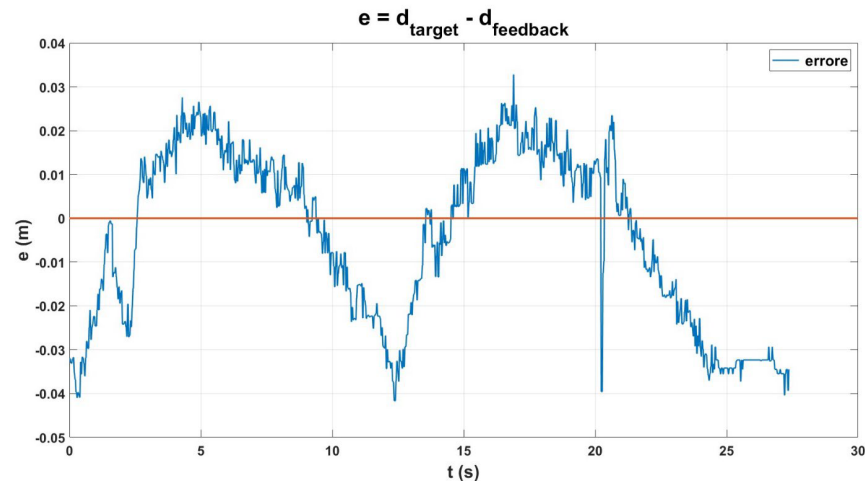
Figure 5.70: Law of Motion: V_x under PI test

Figure 5.71: Law of Motion: Error in PI Test

Performance Index	Gazebo Simulation	Experimentation
MAE [m]	0.0121	0.0198
MAPE [%]	6.0354	9.9039
Standard Deviation [m]	0.0161	0.0254
RMSE [m]	0.0161	0.0256

Table 5.2: Law of Motion Test: Gazebo vs Experimentation

the leader.

In the experiment, all indicators show slightly higher values, as observed in the previous case. This suggests a greater variability or difficulty following the speed trend of

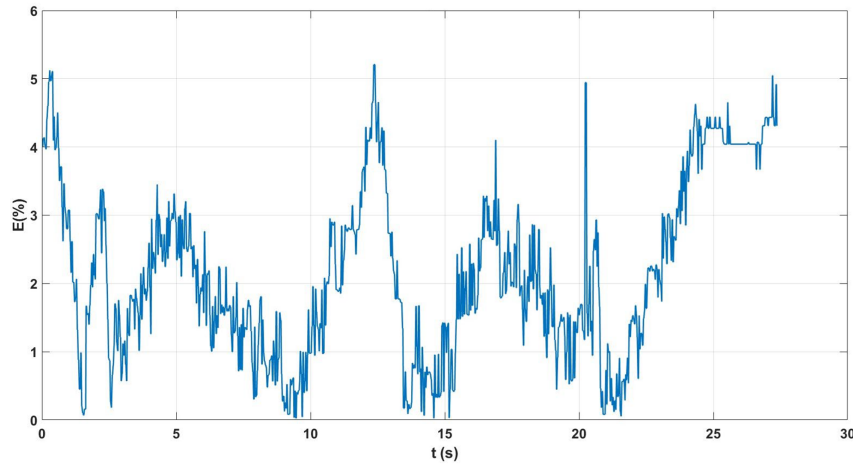


Figure 5.72: Law of Motion: Percentage Error in Test

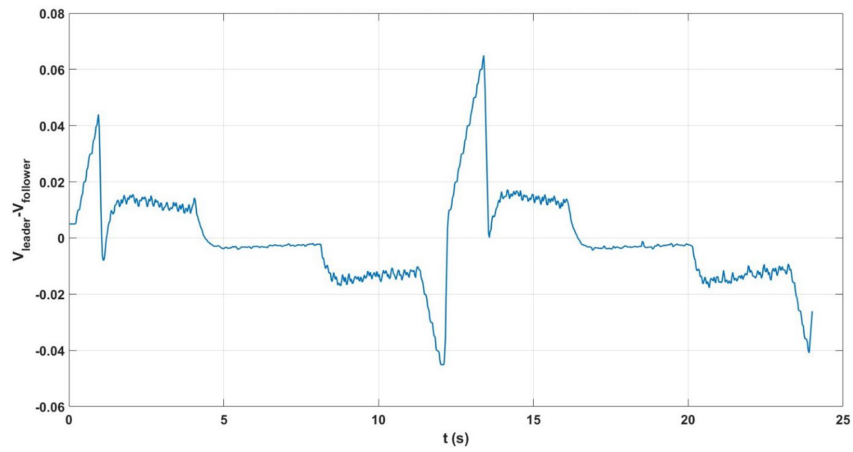


Figure 5.73: Law of Motion: Difference of the velocities in Simulation

the leader under real conditions.

Test with SMC

In order to improve the results obtained in the tests carried out, an alternative approach to PID control was implemented: Sliding Mode Control. Although the PID control gave satisfactory results when trying to emulate the speed profile of the TurtleBot leader by the TurtleBot follower, it did not reach the desired levels in terms of performance and stability.

As a result, Sliding Mode Control was adopted to address the challenges encountered and to further improve system performance. Going back to the explanation in Chapter 1.6.1.2, some specific implementations had to be made to adapt the SMC control theory to this experiment.

In the control described in chapter 1.6.1.2, the switching control (u_{sw}) was defined

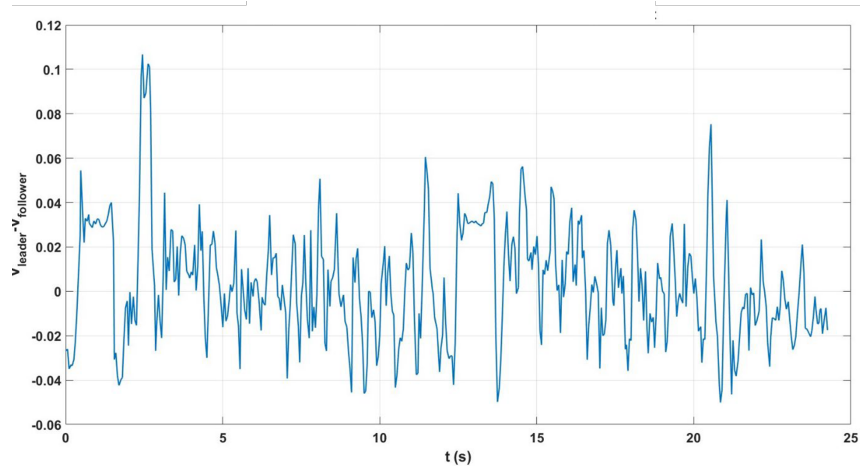


Figure 5.74: Law of Motion: Difference of the velocities in Test

by Eq.3.2. However, this switching action is based on a sign function: this function, and therefore the corresponding control law, has a first order discontinuity at the point $x = 0$. Repeated discontinuities in the control effort produce the phenomenon known as "chattering", a zigzag behavior of the controller output that negatively affects both the state of the system and the lifetime of the hardware components. There are many ways to reduce chattering. One is to replace the sign function with the $\tanh(x)$ function. The hyperbolic tangent is continuous at $x = 0$ and therefore allows the switching control to be rewritten as:

$$u_{sw} = K \cdot \tanh(m \cdot s) \quad (5.17)$$

where m is simply an additional multiplicative factor for the sliding surface around $x = 0$. In this experiment, if we denote by d_{feedback} the instantaneous distance between the two TurtleBots and by d_{target} the distance to be maintained between the two robots, we have that the error is worth its difference. As a sliding surface we use eq.3.3.

Then consider a first-order dynamics that connects the distance between the two TurtleBots:

$$a_1 \cdot \dot{d}_{\text{feedback}} + a_2 \cdot d_{\text{feedback}} = b \cdot u_{\text{eq}} \quad (5.18)$$

Where on the sliding surface will be:

$$u_{\text{eq}} = \frac{a_2}{b} \cdot d_{\text{feedback}} = K_1 \cdot d_{\text{feedback}} \quad (5.19)$$

Because on the sliding surface is:

$$s = 0 \rightarrow \dot{s} = 0 \rightarrow C \cdot \dot{e} = 0 \rightarrow \dot{e} = 0 \rightarrow \dot{d}_{\text{feedback}} - \dot{d}_{\text{target}} = 0 \quad (5.20)$$

$$\dot{d}_{\text{feedback}} = 0 \quad (5.21)$$

In this way the final control signal will be:

$$u_{\text{SMC}}(t) = K_1 \cdot d_{\text{feedback}} + K \cdot \tanh(m \cdot s) \quad (5.22)$$

After testing the control, the following parameters were obtained:

- $m = 500$
- $K_1 = 0.01$
- $K = 0.25$
- $C = 0.05$

In Fig. 5.75 it can be stated that the follower is able to follow the speed profile of the leader with greater delay.

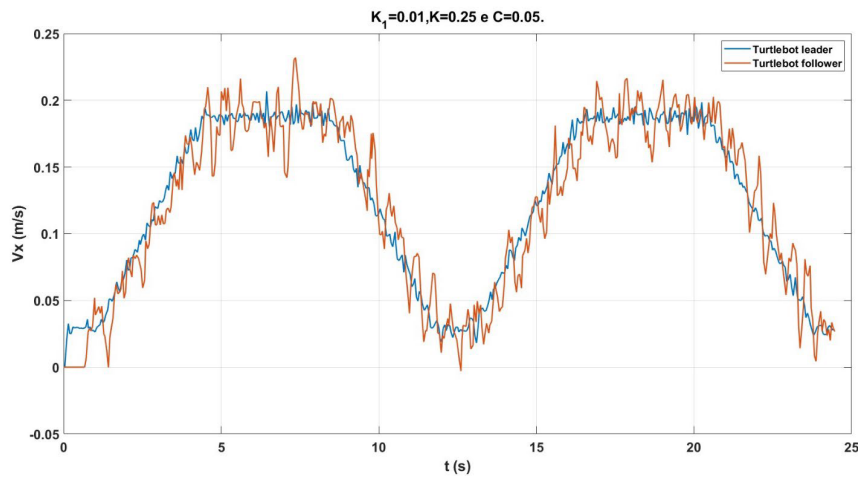


Figure 5.75: Law of Motion: Test with SMC

As shown in Fig. 5.76, there was an error between the target distance and the feedback distance which has a maximum excursion of 0.0576 metres. This is less than the maximum excursion that occurs in the case of PID control (0.0745 metres).

Test: PI vs SMC

To compare the results of the test carried out with the PI control and the SMC, in Fig. 5.77 and in Fig. 5.78, the speed difference between the TurtleBot leader and the TurtleBot follower at each time point for both controls is shown for each control.

In Tab. 5.3, the performance indexes are shown. Analyzing the performance indices of the tests carried out with the PID control and the SMC control, it is clear that the SMC control showed greater accuracy and stability in following the speed than the PID control. This is evidenced by the lower MAE, MAPE, standard deviation and RMSE values obtained with the SMC control.

In particular, the lower MAPE value for the SMC control indicates a better ability to adapt to the velocity of the leading TurtleBot, confirming the effectiveness of this approach in improving system performance.

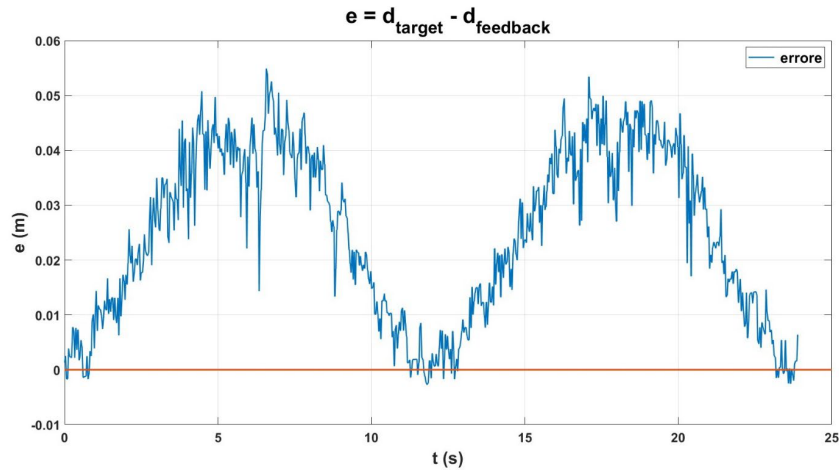


Figure 5.76: Law of Motion: Error in SMC test

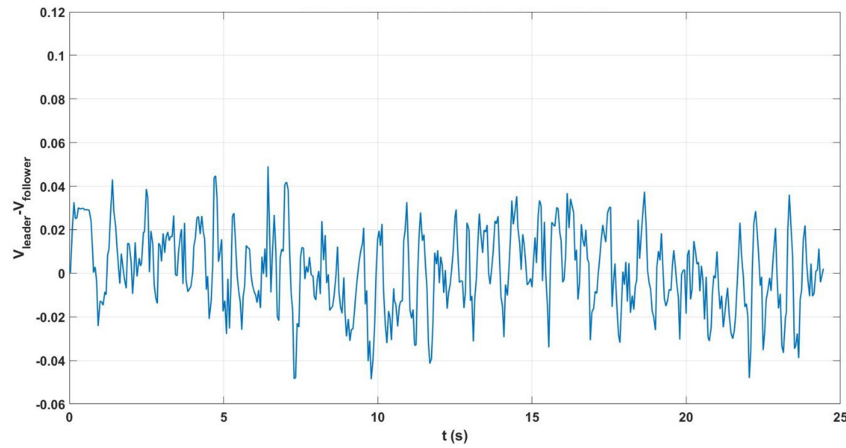


Figure 5.77: Law of motion: Difference of speeds under PI

Performance Index	PID	SMC
MAE [m]	0.0198	0.0151
MAPE [%]	9.9039	7.5404
Standard Deviation [m]	0.0254	0.0186
RMSE [m]	0.0256	0.0186

Table 5.3: Law of Motion Test: PID vs SMC

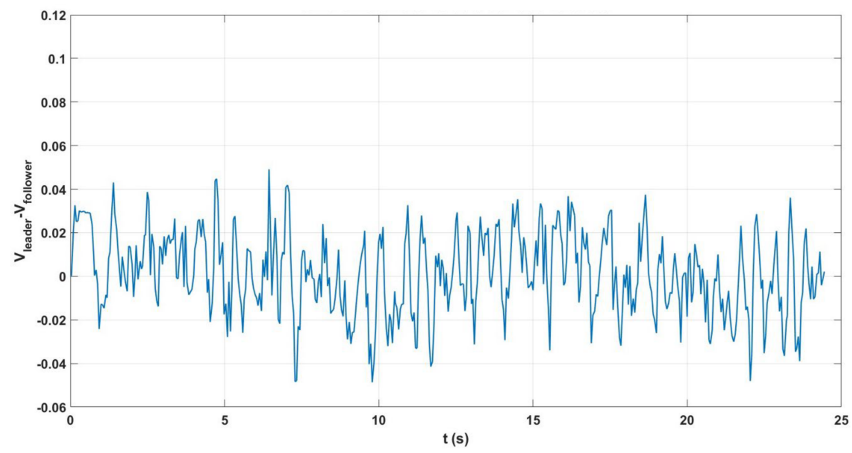


Figure 5.78: Law of motion: Difference of speeds under SMC

5.4.2.3 Obstacle Avoidance Test

The storyboard of the Obstacle Avoidance test is shown in Fig. 5.79

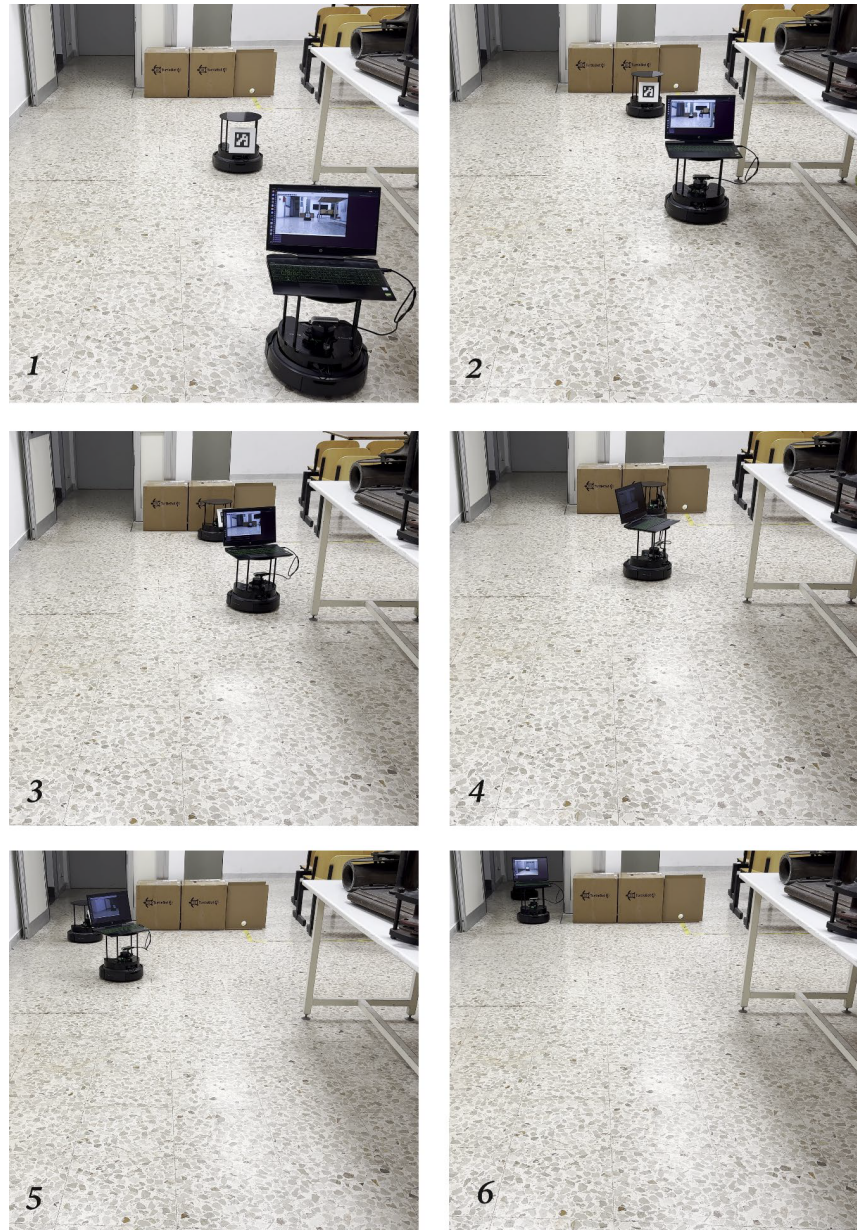


Figure 5.79: Obstacle Avoidance: Test sequence

This test represents one of the best reasons for the use of multi-agent systems. The TurtleBot leader communicates the presence of an obstacle to the TurtleBot follower. In response, the TurtleBot follower will change its trajectory in advance to avoid the obstacle by changing its forward direction. It will then return to its original direction only when the TurtleBot leader signals the end of the obstacle (Fig. 5.80). This approach

minimizes both the distance traveled and the time taken by the TurtleBot follower to pass the obstacle compared to the TurtleBot leader.

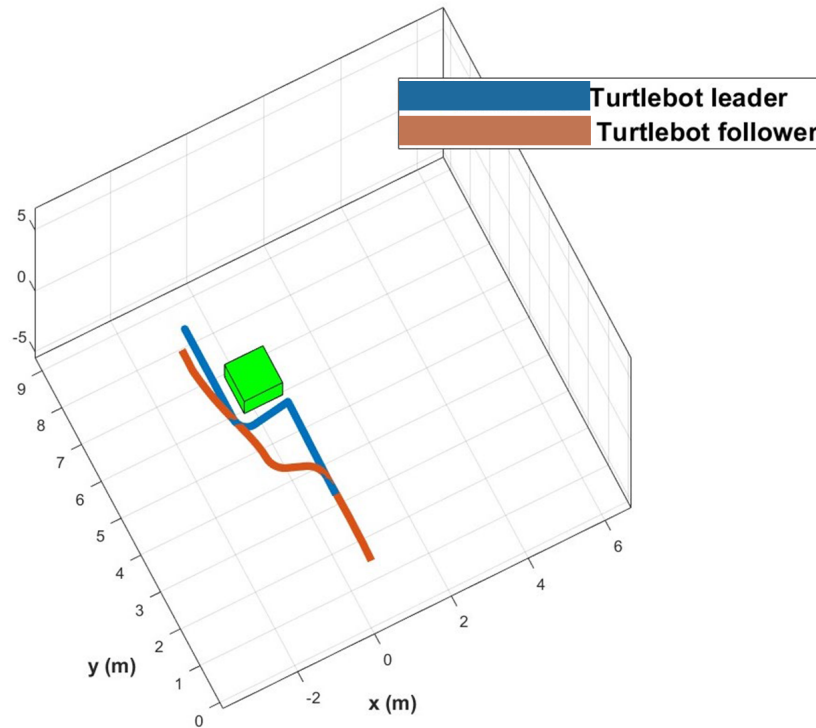


Figure 5.80: Obstacle Avoidance Test: Trajectories

Specifically, as the TurtleBot leader moves forward, it detects the presence of an obstacle using LiDAR. When it is within 50 cm of the obstacle, it stops and sends the information about the presence of the obstacle to the TurtleBot follower.

The TurtleBot follower, without stopping, starts to change its trajectory according to its own policy. Meanwhile, the TurtleBot leader turns 90° and moves forward in a direction parallel to the obstacle as long as the LiDAR signals the presence of the obstacle.

When the TurtleBot leader has completely passed the obstacle, it sends information to the TurtleBot follower that the obstacle has been passed. It then rotates 90° in the opposite direction to its previous rotation and resumes its forward progress in the starting direction.

Throughout the process, the TurtleBot follower, which has never stopped, finishes the test at a shorter distance from the TurtleBot leader than it started. This happens despite the fact that the follower has maintained a lower speed than the TurtleBot leader. For $7.1 < t < 12.6$, the leader stops and starts its 90° rotation. This behavior occurs because the robot has detected an obstacle on its path and reacts to the situation by stopping and changing direction.

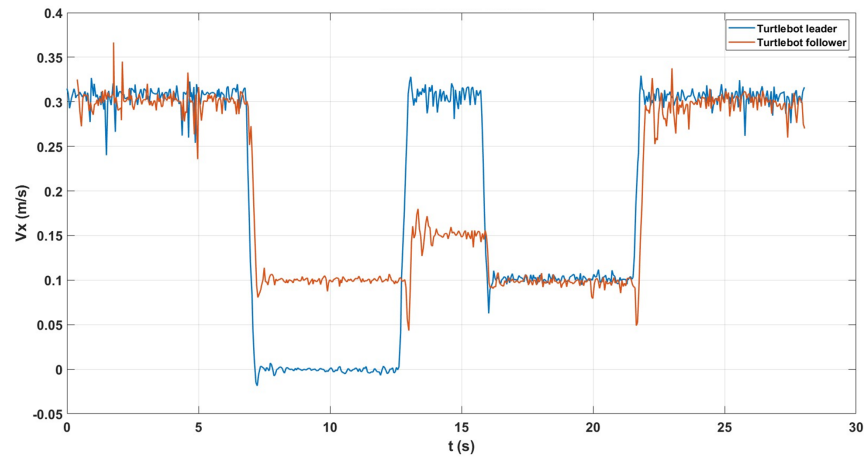


Figure 5.81: Obstacle Avoidance Test: Linear Velocities

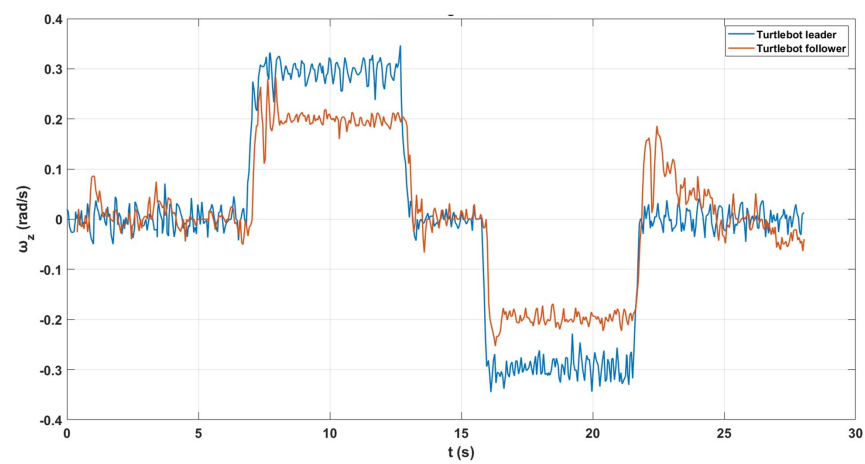


Figure 5.82: Obstacle Avoidance Test: Angular Velocities

5.4.3 Conclusions

In this work, several tests were carried out using two TurtleBot 4 mobile robots, called Leader and Follower, both equipped with a Raspberry Pi 4 board and the ROS 2 framework.

The main objective was to study and implement a control system that would allow the TurtleBot follower to follow the TurtleBot leader. For this purpose, an ArUco marker and a camera were used to provide the entire system with the necessary information for tracking and following.

The first test evaluated the effectiveness of the tracking system and confirmed that the TurtleBot follower was able to successfully follow the TurtleBot leader. This was achieved using a computer vision system and algorithms developed using the OpenCV library in Python.

The test was carried out both in a simulation environment with Gazebo and in real experiments, allowing a comparison between simulation and test.

The second test analyzed the behavior of the TurtleBot follower in response to a trapezoidal velocity profile assigned to the TurtleBot leader. By implementing two different control algorithms, PID and SMC, it was observed that the TurtleBot follower was able to effectively follow the speed trend of the leader while maintaining a predefined target distance.

The third test examined the cooperation between the two robots in the presence of obstacles along the path. Thanks to the communication between the robots, the TurtleBot leader reported the presence of obstacles to the TurtleBot follower, allowing the latter to adopt an alternative navigation strategy to successfully avoid them. The result was a difference in the Follower trajectory, reducing both the distance traveled and the time taken to avoid obstacles.

Among the various possible future developments, it could be useful and interesting to try to optimize the control algorithm to further improve the performance of the Follower. In addition, further research could be carried out into robot cooperation, for example by exploring more complex coordination strategies in which the TurtleBot follower also manages to help the TurtleBot leader. The trajectory followed by the follower could also be optimized as a function of time, distance and energy consumed. Such approaches would maximize the efficiency and coordination between the two (or more) robots when navigating in complex environments.

5.5 Robust Control Systems for Accurate Trajectory Tracking

Kinematic control of mobile robots is a fundamental component in the development of advanced robotic solutions, with applications spanning from logistics to autonomous exploration. However, the complexity of operating environments, often characterized by external disturbances, measurement noise, and unmodeled dynamics, represents a significant challenge to ensuring reliability and accuracy. Traditionally, simple strategies such as proportional control have been used successfully in well-defined contexts where system parameters and environmental conditions are relatively stable (111). While such approaches are intuitive and efficient, they become insufficient in the presence of variable dynamics and significant uncertainties.

To overcome these limitations, control H_∞ has gained attention as one of the most robust and versatile techniques. This approach, based on the modulation of the frequency response of the closed system, allows it to deal with external disturbances and structured uncertainties, ensuring reliable performance even under difficult operating conditions (112). Due to its relation to the small-gain theorem, the control H_∞ has emerged as a foundation for robust control methodologies, such as μ -synthesis, and finds application in numerous fields, from mobile robotics to collaborative robotics (113).

In the context of controlling a single robotic agent, the H_∞ controller has been shown to be particularly effective in ensuring stability and accuracy in trajectory tracking. For example, an optimal H_∞ controller based on linear matrix inequalities (LMI) has been proposed for trajectory control of omnidirectional three-wheeled mobile robots, demonstrating significant mitigation of the effects of noise and measurement noise (114). Similarly, recent studies have proposed advanced adaptive approaches to further improve the performance of H_∞ . For example, in (115), it fixed the H_∞ control with deep neural networks to handle both parametric uncertainties and external disturbances in nonholonomic mobile robots. This innovative approach demonstrates how the combination of robust control techniques and adaptive models can extend the applicability of mobile robots in real-world environments (115).

This chapter focuses on the application of a control synthesized by means of the H_∞ theory for a Turtlebot3 differential mobile robot and will discuss tests of this methodology carried out at the Universitat Politècnica de València - Department of Systems Engineering and Automation. The control obtained will fall in the field of robust control; the methods and results obtained by this technique will be analyzed.

5.5.1 Hardware Overview of Robotic Systems

This work explains how to control the TurtleBot3 Waffle Pi robot (116) to implement kinematic control for trajectory tracking, an essential task in a wide variety of mobile robotics applications ranging from autonomous exploration to indoor mapping. The TurtleBot3 Waffle Pi is equipped with an LDS-02 2D LIDAR, which provides accurate

measurements of surrounding distances critical for navigation and obstacle avoidance. XL430-W210 motors provide accurate control of wheel rotation speed, while integrated optical encoders provide essential odometry data for estimating the position and orientation of robot. In addition, a 6-axis IMU measures accelerations and angular velocities, for further improving the accuracy of robot state estimation.

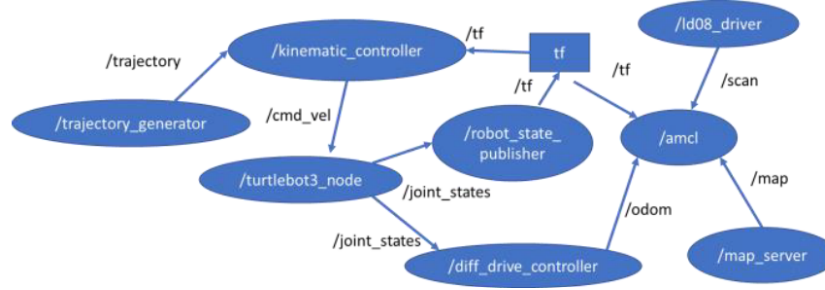


Figure 5.83: ROS2 architecture of the system

The robot is based on a Raspberry Pi 4 running Ubuntu and ROS2. ROS2 uses the DDS protocol for communication, ensuring reliable and scalable data transfer, which is especially beneficial in multi-robot scenarios. The architecture (Fig. 5.83) includes multiple ROS2 nodes:

- The `/turtlebot3_node` subscribes to the `/cmd_vel` topic to receive motion commands in the form of Twist messages.
- The `/diff_drive_controller` node calculates wheel speeds and communicates with the OpenCR1.0 board, which is responsible for sending control signals to the motors and reading the wheel position, as posted in the `/odom` topic.
- The `/ld08_driver` node manages the LIDAR and provides telemetry through the `/scan` topic (LaserScan Message type).

For localization, the AMCL algorithm is used, which requires a map provided by the `/map_server` node. The map is previously constructed using SLAM algorithms, such as the `turtlebot3_cartographer` package. The `/kinematic_controller` node receives the position of the robot via the transformations published in the `/tf` topic, correlates the position of the robot with respect to the odometry and the map, and receives the trajectory to follow via specific messages containing position and velocity information.

5.5.2 Principles of H_∞ Control Theory

This section provides some introductory remarks on H_∞ control theory.

The theory H_∞ is an advanced methodology in control theory used to design controllers that ensure optimal stability and performance even in the presence of uncertainties and disturbances in the system.

The norm H_∞ of a linear time-invariant (LTI) system represented by the transfer function $G(s)$ is defined as the maximum gain the system can have over all frequencies. Mathematically, it is expressed as

$$\|G\|_\infty = \sup_{\omega \in \mathbb{R}} \sigma_{\max}(G(j\omega)) \quad (5.23)$$

where $\sigma_{\max}(G(j\omega))$ is the maximum singular value of $G(j\omega)$ and $j\omega$ is the complex variable on the imaginary circumference of the plane s .

The norm H_∞ quantifies the worst-case response of the system in terms of the gain of the input signal due to a disturbance or uncertainty. If the norm H_∞ of a system is small, it means that the system can effectively suppress large disturbances at all frequencies.

In the context of robust control, minimizing the norm H_∞ of a closed-loop system is a common goal. This is because minimizing this norm means minimizing the worst possible amplification of the disturbance on the output of the system, ensuring that the system is robust to a wide range of uncertainties and disturbances.

Controller synthesis H_∞ is based on solving an optimization problem that tries to find a controller that minimizes the norm H_∞ of the closed loop system. This problem can be transformed into a convex optimization problem using techniques such as Linear Matrix Inequalities (LMI).

The control problem is to design a controller K that stabilizes a system and minimizes the norm H_∞ of the closed-loop system below a certain gamma level γ . This means finding a controller K such that

$$\|T_{zw}(s)\|_\infty < \gamma \quad (5.24)$$

where $T_{zw}(s)$ is the transfer function from a noise signal w to the output of the system z .

In summary, control theory H_∞ provides a powerful tool for designing robust control systems capable of optimal performance even in the presence of uncertainties and disturbances. By minimizing the norm H_∞ , this methodology allows the effects of disturbances on the system to be mitigated as much as possible, improving overall stability and reliability.

5.5.3 Methodologies for Controller Design

The aim of this section is to detail the proposed design methodology for the control of non-holonomic wheeled mobile robots aimed at trajectory tracking.

The problem is divided into two main components: the feedforward signal and the feedback signal. The first signal generates the control actions to be applied in the absence of disturbances and measurement noise in order to follow a given trajectory. The second signal, on the other hand, is concerned with correcting deviations from the trajectory to be followed caused by disturbances and measurement noise.

Fig.5.84 shows the proposed structure for feedforward and feedback, and next will detail all the elements.

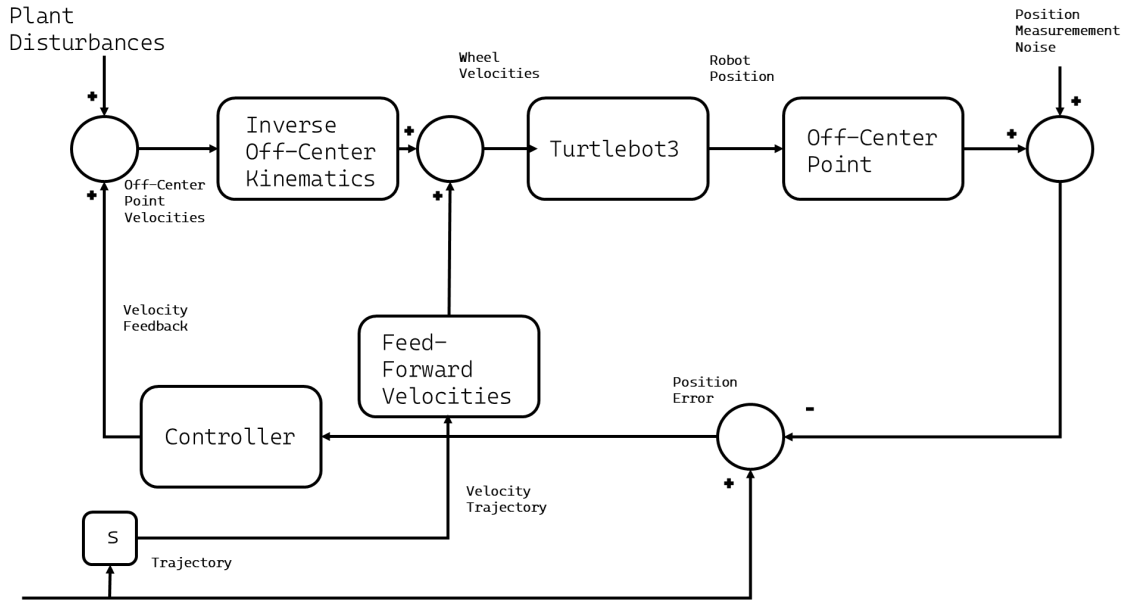


Figure 5.84: System Plant

5.5.3.1 Off-Center Control

Given a non-holonomic mobile robot with wheels, its kinematic equations of motion are described by assuming no wheel slip:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \mathbf{J}_r(\theta) \cdot \mathbf{v} \quad (5.25)$$

where x , y and θ represent the positions in coordinates x and y and the orientation of the robot, respectively, and v and ω are the linear and angular velocities of the base. These equations model the motion of the robot by taking into account the non-holonomic constraint, which limits the degrees of freedom of the system and forces the robot to move only in the direction along its longitudinal axis. The goal of the control, in the case of trajectory tracking, is to control the position x and y of the robot, while the orientation is indirectly specified by the trajectory itself.

Given a reference position in coordinates X and Y , expressed as functions of time $x^*(t)$ and $y^*(t)$, and considering their derivatives $\dot{x}^*(t)$, $\dot{y}^*(t)$, $\ddot{x}^*(t)$ and $\ddot{y}^*(t)$, the linear and angular feedforward velocities for the base of a nonholonomic robot are obtained from the following expressions, respectively:

$$v^{ff}(t) = \sqrt{\dot{x}^*(t)^2 + \dot{y}^*(t)^2} \quad (5.26)$$

$$\omega^{ff}(t) = \frac{\ddot{y}^*(t)\dot{x}^*(t) - \ddot{x}^*(t)\dot{y}^*(t)}{(v^{ff}(t))^2} \quad (5.27)$$

Accordingly, the left (v_L) and right (v_R) wheel speeds for a differential robot can be calculated using the following expressions:

$$\omega_L^{ff}(t) = v^{ff}(t) - b \cdot \omega^{ff}(t) \quad (5.28)$$

$$\omega_R^{ff}(t) = v^{ff}(t) + b \cdot \omega^{ff}(t) \quad (5.29)$$

where b represents half the distance between the wheels.

To obtain the feedback signals, the system is linearized to implement Off-Center Point based control. This approach allows a non-holonomic wheeled mobile robot to be controlled in any direction X - Y using wheel speed commands. From a design perspective, the goal is to adjust the position of the off-center point by treating it as two decoupled integrators, one for each of the directions to be controlled. In this way, the controller compensates for system disturbances and measurement noise.

The most common choice is to position the off-center point at a distance e in the direction of forward motion. Therefore, the position of the eccentric point is defined by:

$$\mathbf{x}_e = \begin{bmatrix} x_e \\ y_e \end{bmatrix} = \begin{bmatrix} x + e \cdot \cos \theta \\ y + e \cdot \sin \theta \end{bmatrix} \quad (5.30)$$

The e parameter affects the control of deviations perpendicular to the trajectory. In this work, this parameter has been set to the distance between the LiDAR of the robot and the wheel axis, and is considered as the controlled variable in the proposed design.

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \end{bmatrix} = \begin{bmatrix} 1 & 0 & -e \cdot \sin \theta \\ 0 & 1 & e \cdot \cos \theta \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \mathbf{J}_e(\theta) \cdot \dot{\mathbf{x}} \quad (5.31)$$

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \end{bmatrix} = \begin{bmatrix} \cos \theta & -e \cdot \sin \theta \\ \sin \theta & e \cdot \cos \theta \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \mathbf{J}_e(\theta) \cdot \mathbf{J}_r(\theta) \cdot \mathbf{v} \quad (5.32)$$

It can be verified that, for $e \neq 0$, the Jacobian $(\mathbf{J}_e(\theta) \cdot \mathbf{J}_r(\theta))^{-1}$ is invertible. Consequently, the linearized system, shown in Fig. 5.84, behaves as follows:

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \end{bmatrix} = \begin{bmatrix} \frac{1}{s} & 0 \\ 0 & \frac{1}{s} \end{bmatrix} \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \end{bmatrix}$$

5.5.4 H_∞ Controller Design

The system to be controlled can be modeled as an integrator, since it is a kinematic controller designed to eliminate disturbances and noise. This control is based on the following generalized plant:

$$\begin{bmatrix} e \\ u \\ \tilde{e} \end{bmatrix} = \begin{bmatrix} \frac{1}{s} & 0 & \frac{1}{s} \\ 0 & 0 & 1 \\ \frac{1}{s} & 1 & \frac{1}{s} \end{bmatrix} \cdot \begin{bmatrix} d \\ n \\ u \end{bmatrix}$$

Where:

- u is the control action corresponding to the desired velocity of the off-center point,
- d is the input disturbance to the process (e.g., a different velocity of the off-center point from the theoretically commanded velocity),
- n is the measurement noise,
- e and $\tilde{e}$ are the position error and the measured position error, respectively (the sign is changed to obtain negative feedback control, as is the norm).

Since the control in coordinates X and Y is decoupled, a separate controller is designed for each of the coordinates.

Along with the generalized plan, a set of filters (or frequency weights) related to the disturbance, measurement noise, error, and control action will be designed. These filters will provide a weighted generalized map that defines the frequency response of the signals to be considered.

The generalized input filters are:

$$G_d(s) = \frac{0.033s + 0.033}{10s + 1} \quad G_n(s) = 0.0333 \quad (5.33)$$

Low frequency disturbances were considered with a maximum amplitude of 3.3 cm/s, which represents the difference between the commanded speed and the speed actually achieved by the robot. At high frequencies, the amplitude is ten times smaller. The Bode plot of this filter is shown in Fig. 5.85(left).

The maximum expected noise in the position measurement is 3.3 cm.

Filters for generalized outputs are defined as the inverse of these "models". They reflect the maximum amplitude of the desired error and controlled variable when the perturbations have a maximum amplitude defined by the frequency filters $G_d(s)$ and $G_n(s)$.

$$G_e(s) = 0.05 \quad G_u(s) = \frac{0.1 \cdot \left(\frac{\pi}{T_s/20} s + 1 \right)}{\frac{\pi}{T_s/2} s + 1} \quad (5.34)$$

Where $T_s = 0.125$ s represents the sampling period used for filter discretization in the computer implementation.

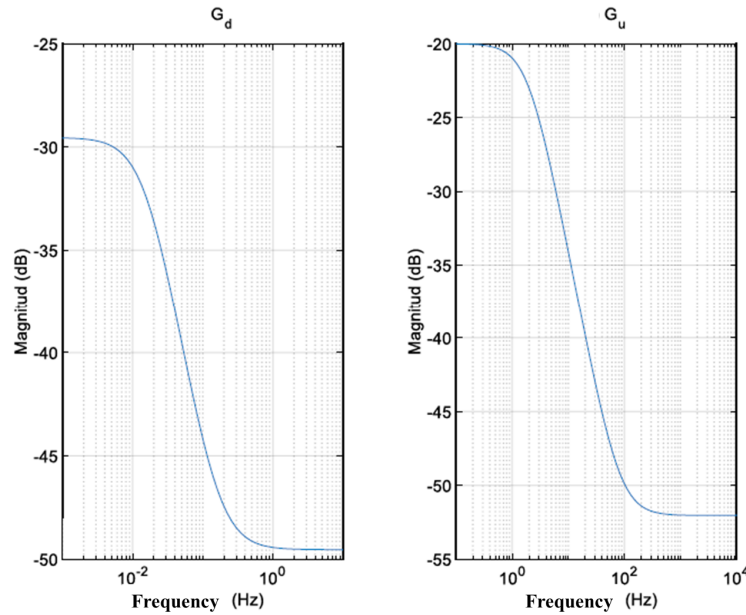


Figure 5.85: Frequency Response of the filters $G_d(s)$ and $G_u(s)$

The pole of the input filter was chosen so that its activity decreases from a frequency equal to 50% of the Nyquist frequency. In this way, the discrete implementation of the design is not affected by high frequency effects or aliasing caused by sampling.

The bode diagram of the input filter is shown in Fig.5.85 (right). The maximum allowable error was set to 5 cm in the worst case.

Several key aspects were considered when designing the filter:

- All filters must satisfy the condition $\|G(\omega)\|_\infty \neq 0$ to ensure that the designed controller/observer does not have infinite gains at any frequency, avoiding malfunctions in practical applications.
- The filters for the generalized plant input (noise and measurement noise) have been chosen according to the frequency content expected in practical experiments. They should be modified if the controllers excite unwanted signals at certain frequencies.
- Simple filters have been designed to avoid an excessive increase in the order of the weighted generalized system. For example, the filter for measurement noise was kept constant, because by limiting the high frequencies of the actuator, the problem is adequately treated in the SISO case.

Based on this specification, three H_∞ controllers were designed to compare their performance:

- Proportional controller with low-pass filter:

$$G_c(s) = \frac{10K_p}{s + 10}$$

The cutoff frequency was chosen to be 10 rad/s to filter out measurement noise.

- PID controller:

$$G_c(s) = K_p + \frac{K_i}{s} + \frac{K_d s}{T_f s + 1}$$

The three controller constants (K_p , K_i , K_d) and the low pass filter frequency (T_f) are chosen by an optimizer.

- Full-order controller: Designed to ensure an optimal solution.

Although the H_∞ methodology is not the only option for designing PIDs (first two options), controllers were designed using this metric for comparison:

- For PIDs, `hinfstruct` command by MATLAB was used to select the structure of the controller.
- The full-order controller was designed using the `hinfsv` command, which guarantees an optimal solution.

The Tab. 5.4 shows the infinite norm ($\|G\|_\infty$) of the various designed controllers. All of the controllers have γ values less than 1, ensuring that the error and control signals remain less than 1 (in scaled magnitudes) with disturbances and noise. In addition, with maximum disturbances described by $G_d(s)$ and $G_n(s)$, errors and control actions are less than $G_e(s)$ and $G_u(s)$ in the original physical magnitudes.

Controller	γ
P (filter)	0.7649
PID	0.7535
Full-order	0.7038

Table 5.4: Infinity norm values ($\|G\|_\infty$) for the designed controllers

The actually designed controls and their parameters are listed in Tab. 5.5.

Fig. 5.86 shows the closed-loop Bode plot of the designed controllers. It is observed that the PID controller is the only one to introduce an integrator, but its frequency response between 0.02 rad/s and 0.3 rad/s reaches a gain of 2.1, while the other controllers maintain a gain of 4 up to 1 rad/s.

At high frequencies, all controllers smooth out the control signal to avoid exciting unwanted high-frequency signals.

The full-order controller, despite not having an integrator, shows high gain at low frequencies, helping to reduce the position error compared with the proportional controller. At high frequencies, all controllers effectively filter signals, including unwanted signals, thus ensuring a robust and stable response.

Fig. 5.87 shows the dynamic behavior in response to a disturbance and noise step on the error and control signals.

Finally, to verify that the closed-loop design meets the specifications required by the filters, Fig. 5.88 shows the closed-loop Bode plot for the different controllers.

Controller	Transfer Function / State-Space Representation
P (filter)	$G_c(s) = \frac{19.8}{s+10}$
PID	$G_c(s) = 1.42 + \frac{0.0992}{s} - \frac{0.442s}{0.325s+1}$
Full-order	$A = \begin{bmatrix} -14.38 & -18.16 & -0.6821 \\ 1.349 & -46.13 & 0.06692 \\ 0.009605 & -61.44 & -0.08782 \end{bmatrix},$ $B = \begin{bmatrix} 0 \\ -65.18 \\ -87.02 \end{bmatrix}, C = [0.9537 \quad -0.03546 \quad -0.001332],$ $D = 0$

Table 5.5: Designed controllers and their representations

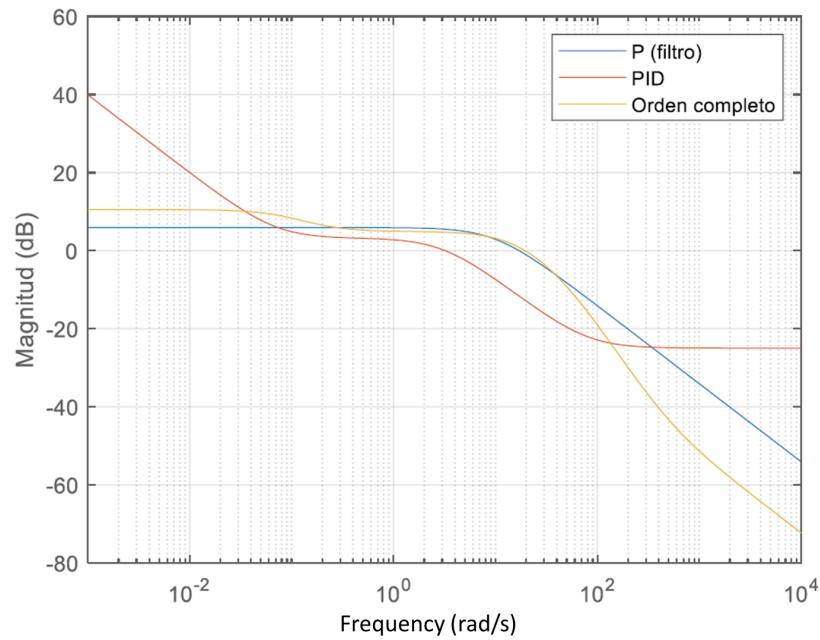


Figure 5.86: Frequency Response for the Controllers

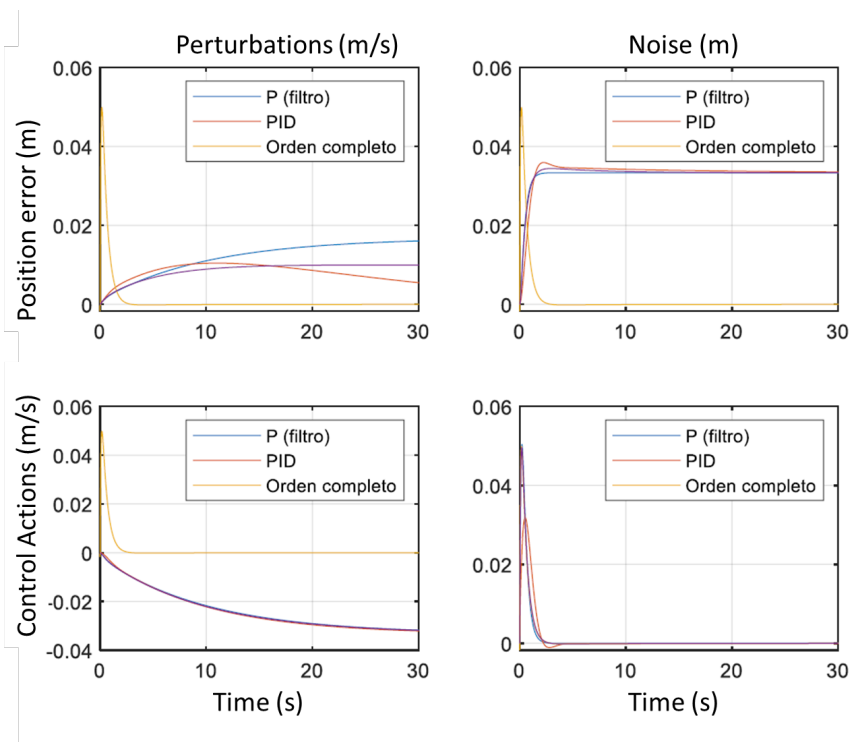


Figure 5.87: Response for different controllers in the presence of system disturbances and measurement noise.

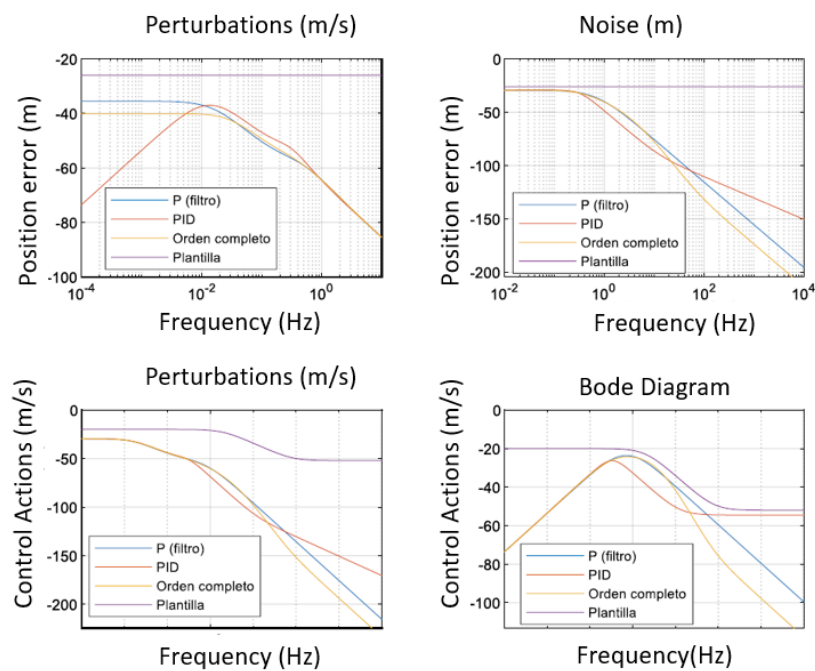


Figure 5.88: Closed Loop Linear System Frequency Response

5.5.5 Validation of H_∞ Control Design

To validate the designed controllers, a kinematic control was implemented so that the system follows a trajectory defined by the following mathematical expression:

$$\begin{bmatrix} x^*(t) \\ y^*(t) \end{bmatrix} = \begin{bmatrix} \cos \phi_0 & -\sin \phi_0 \\ \sin \phi_0 & \cos \phi_0 \end{bmatrix} \cdot \begin{bmatrix} A_x \sin\left(\frac{2\pi t}{T}\right) \\ A_y \sin\left(\frac{4\pi t}{T}\right) \end{bmatrix}$$

Where $T = 120$ s is the period of the trajectory, $A_x = 1.5$ and $A_y = 0.6$ are the lengths of the main axes of the trajectory, and this is rotated by an angle $\phi_0 = -\text{atan}\frac{2A_y}{A_x}$ for convenience.

As an implementation detail, it is showed that the controllers in Tab. 5.5 were actually designed in discrete mode, with a sampling period $T_s = 0.125$ s. The schedule was discretized using the ZOH (Zero Order Hold) discretization method, while the filters were discretized using Tustin's method.

In the Fig. 5.89 it can be seen that all the designed controllers show good trajectory tracking behavior. The applied wheel speeds in the case of the full-order controller are

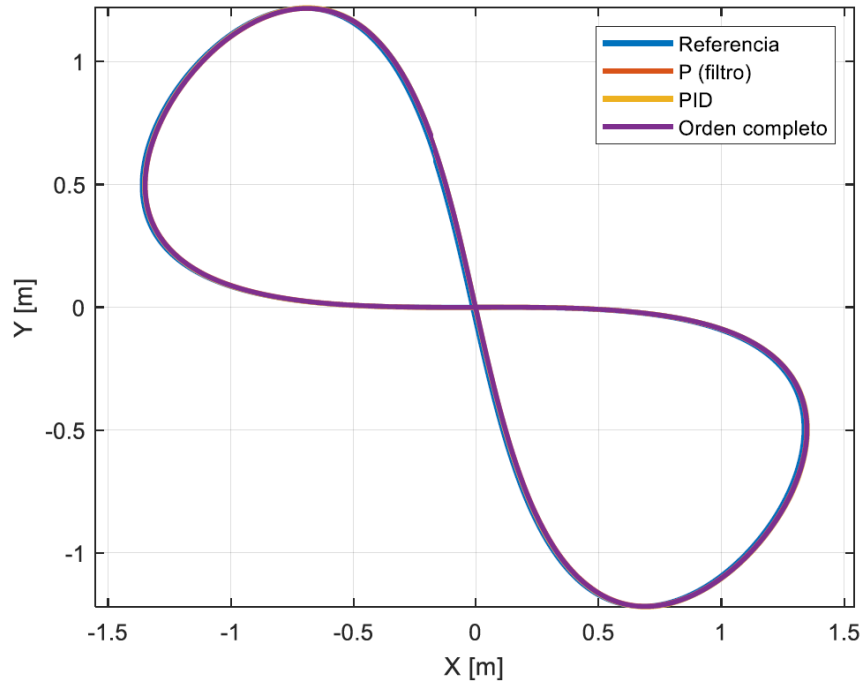


Figure 5.89: X-Y reference trajectory and robot position

shown in Fig. 5.90. This figure shows, on the one hand, the feedforward component and, on the other hand, the actual speed, calculated as the sum of the feedforward and feedback components.

It can be seen that the commanded speed exceeds the feedforward speed. This happens because the speed control of the robot motors has a position error caused by friction.

This behavior is due to the fact that, with the default values, the motors use proportional control to adjust the speed.

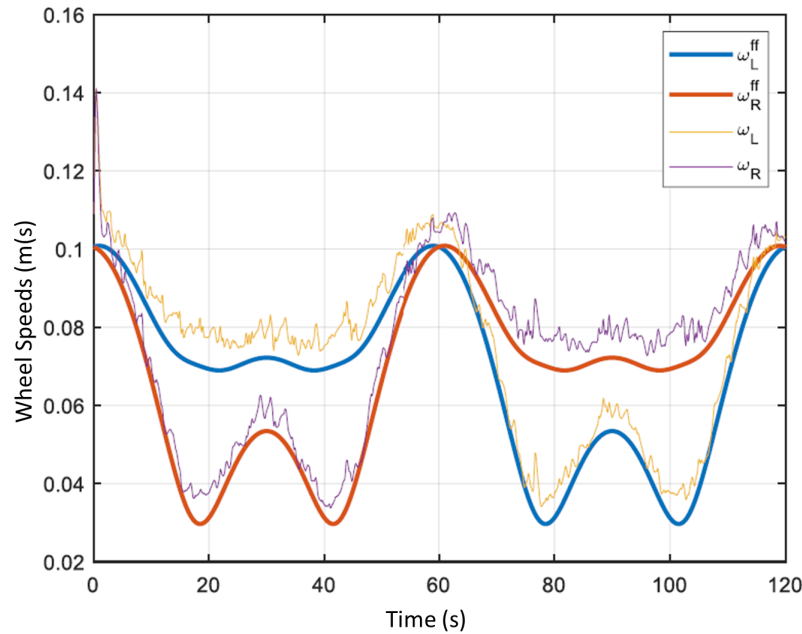


Figure 5.90: Full-order wheel speeds

The errors in the X and Y coordinates during trajectory tracking are shown in Fig.5.91, together with the feedback components obtained from the full-order controller.

The frequency spectrum of the error signals in the coordinate X for different controllers is shown in Fig.5.92. It can be observed that the PID and full-order controllers have a lower average error component (zero frequency error), while having a similar overall frequency response.

Tab.5.6 shows the RMS error values for the different controllers. It can be seen that the full-order controller achieves a slightly lower error than the others, while remaining in the same order of magnitude. However, this result is associated with higher high frequency activity than the other options. This phenomenon, known as the "waterbed effect," is common in control: to reduce the low-frequency error, it is necessary to act faster in the higher frequency bands (117).

Controller	X Coordinate RMS Error	Y Coordinate RMS Error
P (filter)	0.0032	0.0027
PID	0.0031	0.0024
Full Order Controller	0.0028	0.0021

Table 5.6: RMS error values for the different controllers

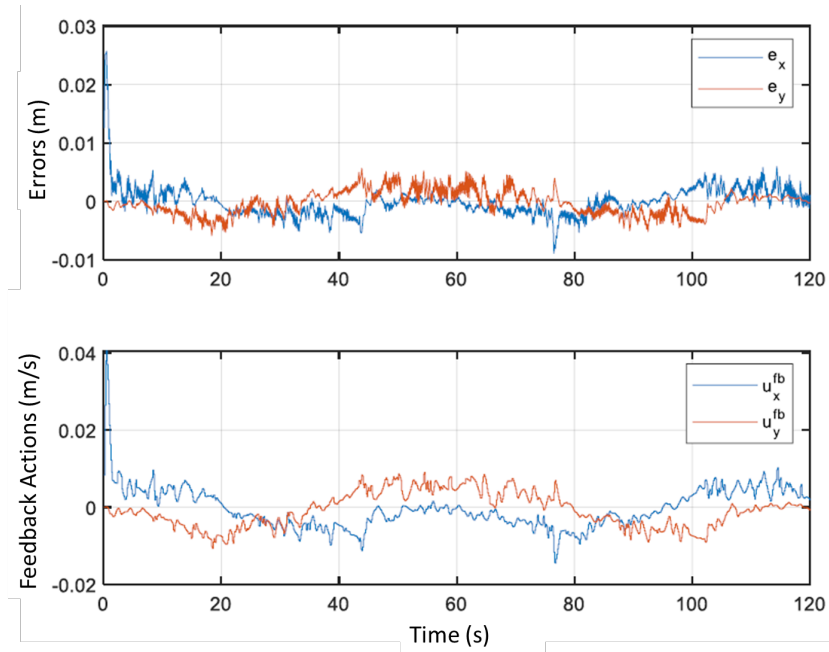


Figure 5.91: Full-order controller position errors and control actions (of the off-center point) generated by the feedback action

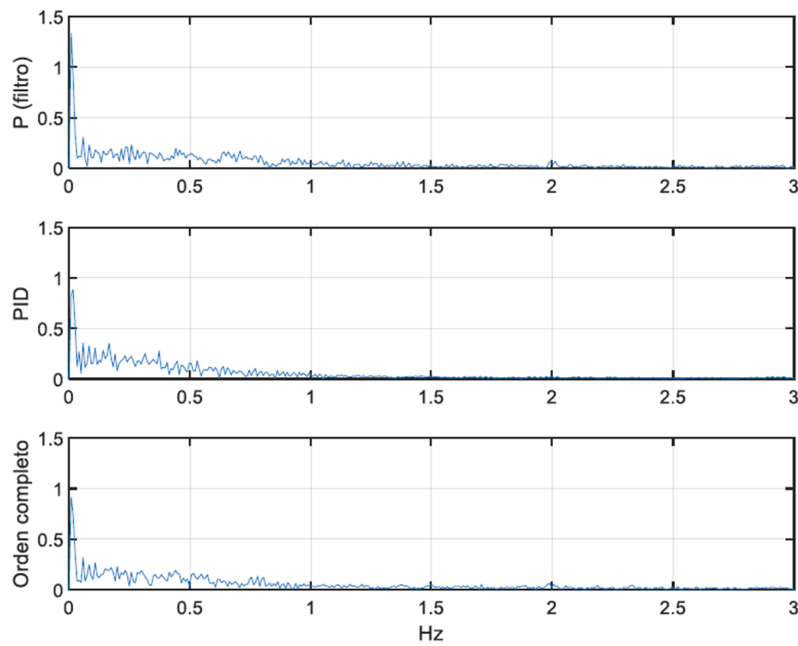


Figure 5.92: FFT of the X-Position Error

5.5.6 Conclusions

In this work, different controller design strategies based on H_∞ techniques were compared, from the full-order optimal controller to simpler approaches (P, PID) using structured H_∞ . The study showed that the application of H_∞ control theory to non-holonomic robotic systems is a robust solution for trajectory tracking in the presence of disturbances and structural uncertainties. The approach used, which integrates feedforward and feedback signals, allows minimizing position errors and keeping the system stable even under complex operating conditions.

Experimental results, confirmed by validation tests, highlight the effectiveness of the designed controllers. In particular, the full-order controller showed better RMS error limitation than solutions based on PID or proportional filters, while maintaining accurate high-frequency response.

This work provides a basis in the field of robust control for mobile robots, with potential applications spanning from logistics to autonomous exploration. The proposed methods can be further developed by integrating advanced adaptive techniques and multi-agent systems.

–6–

Conclusion

This PhD thesis dealt with the development, modelling and validation of a mobile robotic system, a six-wheeled rover developed within the AVELICA project, for waste collection and environmental clean-up. The project, which is part of the European REACT-EU programme, aims to promote sustainable technological solutions in line with the objectives of the European Green Deal.

The work was carried out in several phases, contributing to the design and optimization of a robust and flexible system. One of the main achievements was the creation of detailed multibody models of the rovers, designed as Digital Twins. These models allowed accurate analysis of the dynamics of the systems, load distribution and interactions with rough terrain. Validation, through experimental testing on the small prototype, demonstrated the reliability of the model in representing the behavior of the system under operational conditions. The use of the Digital Twins reduced the time and cost of the design and development phase.

A step-by-step approach, involving the development of a Small Rover prototype and then a Full-Scale prototype, provided for the gradual validation of the design solutions and optimization of the technologies implemented. This method proved effective in combining technological innovation and practical applicability.

The rover was equipped with a rocker bogie suspension system, which improved the ability of the system to adapt to uneven terrain, ensuring even load distribution and greater stability during movement. The system has been thought to be equipped with a multi-function robotic arm, which should be used both for specific harvesting tasks and to modify the attitude of the rover when crossing obstacles. In parallel, a hand-eye calibration algorithm was developed to increase the accuracy of the interaction between the vision system and the robotic arm, improving the effectiveness of autonomous operations. Autonomous navigation applications were also carried out by using other mobile robots, with tests in representative scenarios to evaluate the performance of the algorithms in several environments.

The rover is a good standard to environmental automation, providing a practical solution for litter collection on beaches and other natural areas. Future prospects include the improvement of autonomous navigation systems, with a focus on multi-agent config-

urations, which could improve operational efficiency and enable large-scale collaborative applications. In addition, the integration of IoT technologies and advanced communication systems could extend the functionality of the rover, enabling real-time monitoring and greater interaction with other autonomous devices such as drones and remote control stations. Another possible improvement is the use of the Digital Twin for predictive system maintenance. Through advanced modelling, it would be possible to identify and prevent potential failures, improving operational reliability and reducing maintenance costs.

By ensuring a strong interconnection between theoretical modelling, simulation, and experimental validation, this thesis offers a comprehensive framework in which each chapter plays a crucial role in the overall development of the rover. The work has demonstrated the possibility of developing advanced robotic systems to address environmental challenges, contributing to the protection of ecosystems and sustainability. Although there remains room for improvement, the AVELICA project represents a solid basis for further developments, opening up new opportunities in the field of mobile and environmental robotics.

–A–

Hardware Components and Systems Integration

A.1 Small Prototype Hardware

This section describes the design of the small-scale prototype, including its mechanical structure, electronic components and control system. The rover is built using an Actobotics® chassis and DC motors to provide stability and manoeuvrability for experimental validation. The integration of sensors, communication protocols and power management solutions is detailed to ensure efficient system operation.

A.1.1 Mechanical Components Design

This section presents an analysis of the essential mechanical components of the prototype rover, with a particular focus on the Actobotics® chassis and motors with gearboxes. These components play a critical role in the operation of the Rocker-Bogie suspension and the even distribution of load on the wheels.

A.1.1.1 Chassis

In order to investigate the behaviour of the rover, an initial prototype was built using an Actobotics® chassis, specifically the Bogie Runt Rover™ model as in Fig. A.1.

This frame was chosen for its simplicity, low cost, and lightweight design, and it was 3D-printed in ABS (acrylonitrile butadiene styrene). In accordance with the specifications outlined by the manufacturer and in alignment with the intrinsic characteristics of the Rocker-Bogie suspension, the chassis components are designed in a manner that ensures uniform distribution of the vertical load when all wheels are positioned on a single horizontal plane. This configuration is intended to guarantee consistent traction capacity across the wheels.



Figure A.1: Chassis of the Small-size prototype (118)

A.1.1.2 Motor-Wheel Assembly

The wheels are non-steerable, which restricts the ability of the rover to turn but simplifies the process of programming the logic. The tread on these rubber wheels features four rows of 24 circumferentially arranged studs. The chassis is equipped with six DC electric motors, each coupled with a 48:1 gearbox, which enables the output shafts—on which the wheels are mounted—to rotate at a relatively low speed. This configuration is crucial for the effective operation of a suspension system such as the Rocker-Bogie.



Figure A.2: DC motor with gearbox

The gearboxes reduce rotational speed and change the rotational axis direction of the output shaft, utilizing a right-angle gearbox (Fig. A.2). This configuration has the effect of minimizing the motor assembly footprint, thus allowing for a more compact arrangement along the ends of the rocker and bogie links. The motor-gearbox assembly datasheet, in Fig. A.3, is provided below, as it is a fundamental component of the MultiBody modeling process in Chapter 2.

SPECS	
Output Shaft Style	Double Flats
Motor Type	Brushed DC
Gear Material	Plastic
Wire Length	12"
Weight	1.02 oz (29g)
Wire Gauge	26awg
Voltage (Nominal)	6VDC
No-Load Speed @ 6VDC	230RPM
No-Load Current @ 6VDC	190mA
Stall Current @ 6VDC	250mA
Stall Torque @ 6VDC	11.11 oz-in (0.8 kg-cm)
Gearbox Style	Face Gear / Straight Out Spur
Connector Type	Male Header Pin
Gear Ratio	48:1

Figure A.3: Data sheet of the DC Motor

A.1.2 Electronic Components Integration

This section presents the Arduino Uno WiFi Rev 2 board, which was selected as the central control unit for the prototype due to its versatility and ease of programming. To facilitate the management of the six DC motors integrated into the rover, two Adafruit Motor Shield V2s were incorporated into the design. This integration enables the simultaneous control of multiple motors, addressing the requisite functionality of the system.

A.1.2.1 Arduino Board

The Arduino Uno WiFi Rev 2, in Fig.A.4 was selected as the central electronic board for the prototype. The control board is equipped with an 8-bit ATmega4809 microcontroller. The selection of Arduino®, an open-source platform, was driven by the potential to accelerate the prototyping process through its integrated development environment (IDE), which supports programming in a language derived from C/C++. The prototype hardware is based on a printed circuit board that integrates the ATmega4809 microcontroller. The circuit comprises:

- 14 input/output pins, 5 of which are configured for pulse-width modulation (PWM) and 6 are designated for analog inputs.
- A 5-volt voltage regulator.
- A 16 MHz piezoelectric crystal oscillator for clocking.
- An ICSP programming connector.

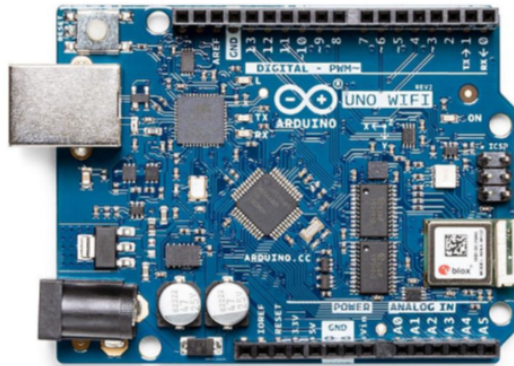


Figure A.4: Arduino Board (119)

- A reset button.

The USB interface enables serial communication with the computer for the purposes of programming, while the WiFi capability facilitates the straightforward implementation of remote control.

A.1.2.2 Adafruit Motor Shield V2

As the Arduino® control board is unable to directly drive all six DC motors in the prototype, two Motor Shields have been employed. The Adafruit Motor Shield V2 (Fig.A.5) was selected for testing purposes, as it is capable of being stacked directly onto Arduino pins. Each Adafruit Motor Shield is capable of controlling up to four DC motors and one servo motor simultaneously. The utilization of two such boards allows for the independent control of up to eight DC motors, exceeding the necessary system requirements by two.

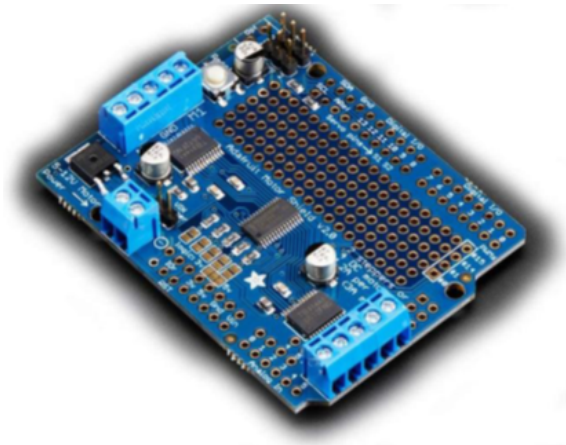


Figure A.5: Adafruit Motor Shield V2 Motor-Shield (120)

A.1.3 Embedded Sensors Implementation

A.1.3.1 LSM6DS3TR IMU

The prototype employs a single sensor, specifically the LSM6DS3TR 6-axis IMU, which is integrated on the Arduino® control board. The sensor comprises both a gyroscope and an accelerometer. The accelerometer is capable of measuring accelerations along the three Cartesian axes, whereas the gyroscope is able to record angular velocities. By processing the data from the sensors in a joint manner through the application of sensor fusion techniques, the attitude of the rover can be determined by calculating three orientation angles. While the IMU sensor accurately computes roll and pitch angles, the yaw angle is more subject to inaccuracies due to numerical integration errors, which can result in significant measurement drift. The aforementioned drift in yaw calculation could be mitigated by incorporating a magnetometer, as it directly measures yaw without being influenced by integration errors.

The data collected, comprising accelerations, angular velocities, and orientation angles, are acquired and processed in real time. Each measurement records ten variables: the time in milliseconds, the three accelerations along the Cartesian axes (Fig. A.6, the three angular velocities around the axes, and the computed roll, pitch, and yaw angles. The aforementioned measurements facilitate a comprehensive examination of the dynamic behavior of the system during experimental trials conducted in controlled settings.

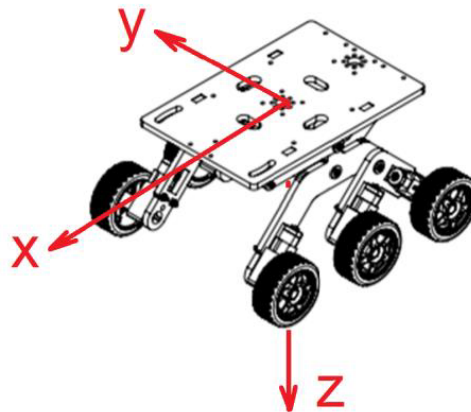


Figure A.6: Reference System of the IMU (118)

In the reference system of the IMU, the x, y, and z axes are aligned with the longitudinal, transverse, and vertical axes of the rover, respectively. The data collected by the IMU sensor are employed not only in post-processing but can also be utilized directly on the Arduino® board to implement a fundamental control system for the rover and its trajectory.

The LSM6DS3TR sensor in the prototype is preset to sample at 104 Hz. To improve the signal quality utilized for trajectory monitoring, the library parameters were modified to elevate the sampling rate to 416 Hz, with the potential for additional increases. It is

evident that an increase in the sampling rate of the IMU results in a corresponding rise in energy consumption.

A more comprehensive description of the operational principles of a MEMS IMU is provided hereafter.

The IMU, or "inertial measurement unit," is a device that incorporates inertial MEMS (MicroElectroMechanical Systems) sensors, such as accelerometers, gyroscopes, and magnetometers. An inertial measurement unit (IMU) is a device that incorporates inertial microelectromechanical (MEMS) sensors, including accelerometers, gyroscopes, and magnetometers. The sensors facilitate the monitoring of the dynamics of a moving vehicle, providing valuable data for use by onboard computers or ECUs. IMUs have a multitude of applications across various fields, including aeronautics, marine, and space, particularly in inertial guidance systems. Additionally, they are employed in motor vehicles for electronic attitude control, significantly enhancing vehicle safety and stability. MEMS accelerometers are principally constituted of a mass supported by translational springs, as depicted in Fig. A.7. In the context of MEMS sensors, the sensing component

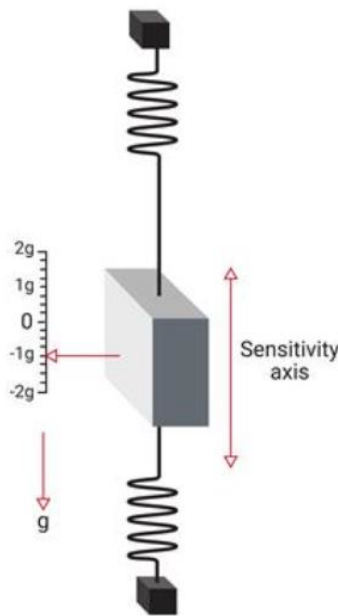


Figure A.7: MEMS Accelerometer Structure

is referred to as the "test mass," and the direction along which it can move is designated as the "axis of sensitivity." The test mass is observed to undergo a shift proportional to the intensity of the applied acceleration. Similarly, MEMS gyroscopes are composed of masses and springs, as illustrated in Fig. A.8. The masses are subjected to a force along the x-axis that causes them to oscillate while on the z-axis there are angular oscillations at velocity ω . Because of this in the y direction the masses will be subjected to Coriolis forces that cause measurable displacements proportional to the velocity ω .

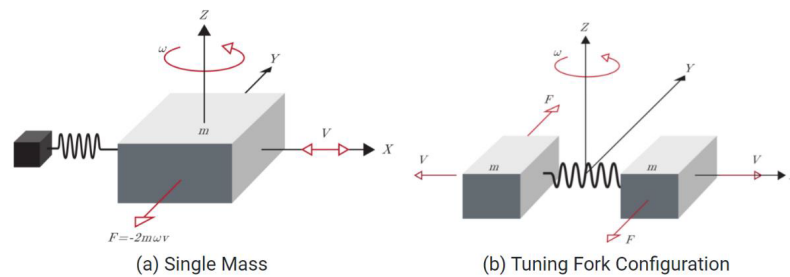


Figure A.8: MEMS gyroscope structure: (a) single mass configuration, (b) tuning fork configuration

A.1.3.2 MPU 6050

The MPU6050 module (Fig.A.9) is a microelectromechanical system (MEMS) that contains a three-axis accelerometer and a three-axis gyroscope. This enables the measurement of a multitude of parameters related to the motion of a system or object, including acceleration, velocity, orientation, displacement, and numerous others. The angular acceleration of a body on its own axis can be measured by the gyroscope, while the acceleration of a body along a direction can be measured by the accelerometer. The device is highly accurate, employing a 16-bit ADC (analog-to-digital converter) for each channel. Once

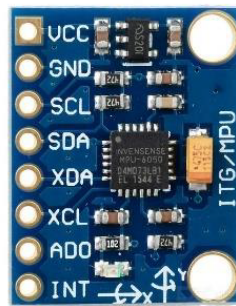


Figure A.9: MPU6050 Sensor

the requisite data has been obtained from the IMU sensor, it is clearly essential to proceed with the transfer of data. The data obtained from the IMU sensor provides a physical representation of the motion of the rover, and by performing continuous acquisition, it is possible to obtain data that varies over time. It is precisely through the acquisition of time-varying data that an output signal can be reconstructed, which can then be subjected to post-processing analysis.

A.1.4 Data Acquisition System

The data obtained from the IMU sensor represent the dynamic behavior of the rover during motion. The time-varying data are continuously acquired, resulting in a signal that can be analyzed in post-processing.

The limited memory capacity of the Arduino board precluded the feasibility of storage on the board. Initially, data transfer over a WiFi connection was considered, but this approach proved inadequate due to network delays, resulting in insufficient sampling rates. Consequently, two distinct acquisition methodologies were incorporated into the code: one via USB and another utilizing a micro-SD card.

A.1.4.1 USB Logging

In order to facilitate the acquisition of data via USB cable, PuTTY, an SSH client, was employed to facilitate the logging of data transmitted over the Arduino virtual serial port. It is of significant importance to ensure that the baud rate is synchronized between PuTTY and the Arduino in order to guarantee optimal functionality. The initial configuration of the baud rate on the Arduino was set to 9600, which proved to be insufficient for effective data transmission, consequently reducing the sampling rate. As a result, the baud rate was increased to 57,600, the maximum supported by the board, with the objective of accelerating the transmission process.

The Arduino code was modified to enable the printing of sensor readings in a serial format. Although the sensor operates at a sampling rate of 416 Hz, the limited dynamic memory of the Arduino board and the volume of command strings being processed reduce the effective sampling rate to approximately 100 samples per second for each of the 10 monitored quantities. This rate is sufficient for the embedded electronics. It is noteworthy that this reduction in sampling rate is exclusive to the serial printing process. If the sensor data is utilized directly on the device, the sampling rate remains at 416 Hz.

A.1.4.2 SD Data Acquisition

An additional electronic module, connected to the Arduino via ICSP connectors, was employed for the acquisition of data. Its name is micro-SD card (Fig. A.10).



Figure A.10: Micro-SD module

In order to enhance the efficiency of data storage, the code was modified to incorporate a buffer of predefined capacity, which is utilized to store a specified number of captures prior to their periodic transfer to the micro-SD card. The library utilized for micro-SD writing necessitates three consecutive operations for each save: opening the file, writing, and closing the file. Of these, the open and close operations are especially time-consuming.

The buffer was implemented to avoid the necessity of opening and closing the file for each 10-data acquisition (comprising time, 3 accelerations, 3 angular velocities, and 3 angles), which had previously resulted in a reduction of the sampling rate to approximately 5 Hz due to delays associated with file handling. The implementation of a buffer reduced the number of file openings and closings, enabling a sampling rate of approximately 60 samples per second when saving to the micro-SD card.

However, it was found that the serial connection offered a more efficient means of data saving. In cases where complex trajectories are involved and a USB cable is not a practical solution, saving to the SD card remains a potential avenue for data storage. In both cases, the data is saved in a single file in append mode, with each 10-data set added to the existing data queue.

A.1.4.3 Power Supply System Analysis

In the tests in which data were acquired by serial, the Arduino was powered directly through the USB connection. In this configuration, the six motors, along with the two Adafruit Motor Shields, were powered by a battery pack comprising four 1.5-volt AA-size stylus batteries. However, in tests in which data were saved to micro-SD, powering the Arduino from the same battery pack used for the motors resulted in notable voltage drops, which frequently led to a board reset, rendering the system unresponsive. A solution was implemented that involved the use of two distinct power systems. The Adafruit motors and Motor Shields were powered by the battery pack, while the Arduino was powered by a 5V and 2A USB power bank. This configuration ensured system stability during testing, as demonstrated in the final prototype configuration, which was prepared for experimental testing (see Fig. 3.1).

A.2 Small Prototype Robotic-Arm Hardware

This section describes the robotic arm hardware components integrated into the small prototype rover. The design prioritises lightweight materials, modularity and high-precision actuation to ensure efficient performance in different test scenarios.

The mechanical structure of the arm is built using high-torque servo motors selected for their compactness, reliability and precise motion control. Different servo models are integrated, each performing a specific function within the kinematic chain of the arm, from basic rotation to end-effector actuation. The section also describes the electronic control interfaces that ensure seamless communication between the robotic arm and the main processing unit.

A.2.1 Mechanical Components Design

This sections describe the structural and mechanical design of the robotic arm integrated into the small prototype. The architecture of the arm is optimised for lightness, modularity and precision to ensure effective interaction with the environment. The following subsections outline the main mechanical components, including the frame and actuation systems.

A.2.1.1 Robotic arm

The robotic arm used in this experiment is the Lynx 6 from LynxMotion, as shown in Fig. A.11



Figure A.11: Robotic arm (121)

The Lynx 6 robotic arm is capable of performing rapid, precise, and consistent movements. The product was selected on the basis of its cost-effectiveness and robust, durable design. The arm is constructed using high-strength, laser-cut Lexan structural components, which exhibit considerable durability and resistance to deformation. The servomotors,

which are responsible for movement, are connected to an Arduino board that controls the commands. Four Hitec HS-475 servomotors are used to power the links of the arm, while a Hitec HSR-5990TG motor is employed to drive the base. This motor is capable of delivering high torque and ensures stability at the base joint. Lastly, the gripper is actuated by HS-85MG and HS-81 servomotors.

A.2.1.2 HITECH HS-475

Despite its compact size, this analog servo motor is capable of delivering approximately 53 Ncm of torque at 6.0V, making it an optimal choice for applications that require high performance in confined spaces. The response speed increases to 0.18 seconds per 60 degrees at maximum voltage, demonstrating a high level of reactivity, which is ideal for dynamic robotics or model-building tasks. However, the use of plastic gears and a single bearing support may not be optimal for the handling of heavy loads or for extended periods of operation. Additionally, the large 200-degree turning radius offers greater flexibility compared to standard models that stop at 180 degrees, allowing for more extensive and precise movements. The specifications for this motor are shown in Fig. A.12.



Figure A.12: HITECH HS-475 motor specifications (122)

A.2.1.3 HITECH HSR-5990TG

The digital servo motor depicted in the image represents a high-performance model, capable of delivering exceptionally high torque, reaching approximately 294 Ncm at 7.4V. This makes it particularly well-suited to applications that require substantial force and precision. These characteristics make it an optimal choice for the base of the device,

which requires both strength and stability. The coreless motor design ensures more efficient operation with reduced vibration. With a speed of 0.14 seconds per 60 degrees at 7.4V, the motor offers extremely fast response times, making it ideal for applications requiring immediate reactions, such as advanced robotics. The titanium gears provide superior strength and durability compared to plastic gears, while the dual bearings ensure smoother and more stable motion. The specifications for this motor are shown in Fig. A.13.

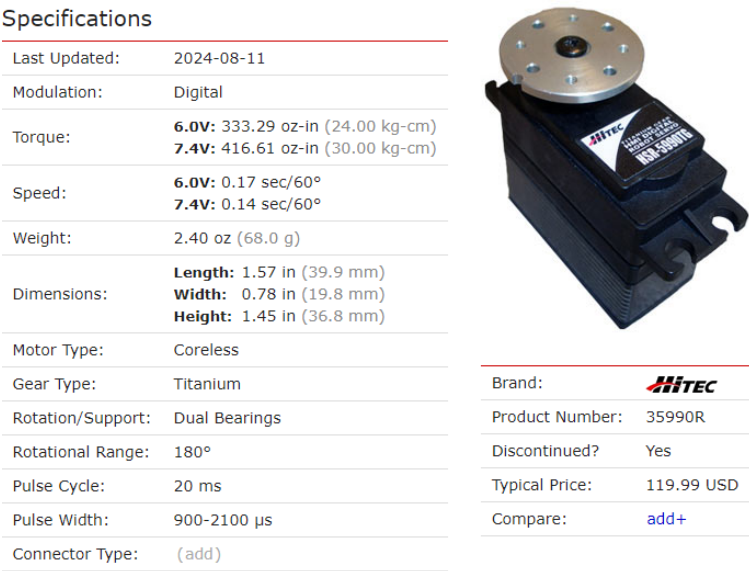


Figure A.13: HSR-5990TG motor specifications (122)

A.2.1.4 HITECH HS-85MG

The analog servo motor, as illustrated in Fig. A.14 is distinguished by its remarkably compact dimensions and minimal weight of 0.020 kg. This renders it particularly well-suited for applications where spatial constraints and weight considerations are significant, such as in the context of compact devices. The device exhibits a torque of approximately 35 Ncm at 6.0V, which is remarkable given its compact dimensions. This output is sufficient for light-duty tasks. With a rotation speed of 0.14 seconds per 60 degrees at 6.0V, the device ensures quick responsiveness, which is valuable in applications requiring fast reactions. The metal gears offer enhanced durability and strength compared to their plastic alternatives. However, the single-bearing mount may not be the most robust in applications requiring high-frequency or load-bearing capabilities.

A.2.1.5 HITECH HS-81

This analog servo motor offers a favorable balance of lightweight design and rapid performance, weighing in at only 0.02 kg and capable of achieving a rotation speed of 0.09

Specifications	
Modulation:	Analog
Torque:	4.8V: 42.00 oz-in (3.02 kg-cm) 6.0V: 49.00 oz-in (3.53 kg-cm)
Speed:	4.8V: 0.16 sec/60° 6.0V: 0.14 sec/60°
Weight:	0.77 oz (21.8 g)
Dimensions:	Length: 1.14 in (29.0 mm) Width: 0.51 in (13.0 mm) Height: 1.18 in (30.0 mm)
Motor Type:	3-pole
Gear Type:	Metal
Rotation/Support:	Single Bearing
Rotational Range:	(add)
Pulse Cycle:	20 ms
Pulse Width:	900-2100 μ s
Connector Type:	(add)



Brand:	HITEC
Product Number:	32085S
Typical Price:	29.95 USD
Compare:	add+

Figure A.14: HS-85MG motor specifications (122)

seconds per 60 degrees at 6.0V, positioning it among the fastest in its class. While the torque of 30 Ncm at 6.0V is not among the highest, it is sufficient for applications that do not require extreme forces, such as small robots, as in this case. The plastic gearing and bushing support may prove to be a limiting factor in terms of durability when the motor is subjected to continuous strain. However, the key advantage of this motor is its remarkably compact size and light weight, which make it an ideal choice for applications where space and weight reduction are critical, as illustrated in the case being examined. The specifications for this motor are presented in Fig. A.15.

A.2.2 Electronic Components and Interfaces

The electronic components employed in the experiments described in this experiment and analyzed in this phase are identical to those utilized in all previous experiments and have been previously described at the beginning of this chapter. The primary distinction in this instance is the integration of a robotic arm mounted on the rover, which made it necessary to use multiple batteries to power the entire system. Accordingly, the power supply was divided, with one battery designated for the DC motors of the rover and another for the servomotors controlling the joints of the robotic arm. In the simulation, a voltage of 6V was planned for both the gear motors and the robotic servomotors; thus, two 6V, 3500 mAh Ni-MH batteries were selected (Fig. A.16).

Specifications

Modulation:	Analog
Torque:	4.8V: 36.00 oz-in (2.59 kg-cm) 6.0V: 42.00 oz-in (3.02 kg-cm)
Speed:	4.8V: 0.11 sec/60° 6.0V: 0.09 sec/60°
Weight:	0.58 oz (16.4 g)
Dimensions:	Length: 1.17 in (29.7 mm) Width: 0.47 in (11.9 mm) Height: 1.16 in (29.5 mm)
Motor Type:	3-pole
Gear Type:	Plastic
Rotation/Support:	Bushing
Rotational Range:	180°
Pulse Cycle:	20 ms
Pulse Width:	900-2100 µs
Connector Type:	(add)



Brand:	HITEC
Product Number:	31081S
Typical Price:	17.95 USD
Compare:	add+

Figure A.15: HITECH HS-81 motor specifications (122)



Figure A.16: Ni-MH 6V 3500mAh battery

A.3 Full-Scale Rover Prototype

A.3.1 Rapid Prototyping Techniques

Rapid prototyping (RP) comprises a range of technologies that facilitate the creation of three-dimensional objects through an additive, layer-by-layer process. Although the conceptual foundations of RP can be traced back to the 1890s, its commercial introduction occurred only in the 1980s. The evolution of RP is inextricably linked to advancements in computer and software technologies, with computer-aided design (CAD) systems playing a pivotal role in enabling the development of contemporary RP systems. The initial objective of RP systems was to produce prototypes in significantly reduced timeframes, ranging from a few hours to several days, thereby expediting the product development cycle. However, following over three decades of technological advancement, rapid prototyping has evolved beyond its original scope. The applications of rapid prototyping are now diverse, encompassing a number of sectors including, but not limited to, design, manufacturing, automotive, aerospace, biomedical, jewellery and coin production. This expansion highlights the versatility of RP and its essential function in contemporary industrial and creative processes.

A.3.1.1 Additive Manufacturing

3D printing, also referred to as additive manufacturing, is a process whereby three-dimensional objects are constructed based on a CAD or digital 3D model. This process entails the regulated deposition, union, or solidification of materials under the direction of a computer (123). The material, which may be any of a number of substances including plastics, liquids, and powder grains, is typically added in successive layers to build the final structure. The first instances of 3D printing occurred in the 1980s, at which time it was primarily regarded as a tool for creating functional or aesthetic prototypes. The term "rapid prototyping" was widely used to describe its application at that time. By 2019, advancements in precision, repeatability, and the range of printable materials had significantly expanded the potential of 3D printing, rendering it a viable industrial production technology. Consequently, the terms "additive manufacturing" and "3D printing" are now frequently used interchangeably. One of the most significant advantages of this technology is its ability to produce highly complex shapes and geometries, including hollow structures and components with internal trusses, which would otherwise be unfeasible to construct manually. This capability enables the creation of innovative designs and substantial weight reductions, thereby broadening the applicability of 3D printing across diverse fields (124).

One of the key advantages of 3D printing is its ability to produce highly complex shapes or geometries that would otherwise be impossible to construct manually. This includes hollow parts and components with internal truss structures, which are ideal for reducing weight.

There are numerous branded additive manufacturing processes, which can be grouped

into seven main categories:

- Vat photopolymerization
- Material jetting
- Binder jetting
- Powder bed fusion
- Material extrusion
- Directed energy deposition
- Sheet lamination

The primary differences between these processes lie in the methods used to deposit layers and the materials employed. Each technique has its own set of advantages and limitations. For instance, some manufacturers offer options to use powders or polymers as build materials, while others may use standard, off-the-shelf business paper to create durable prototypes.

Key considerations in selecting a 3D printer typically include printing speed, the cost of the printer and printed prototypes, material choice and expense, and color capabilities. Printers capable of working directly with metals are generally expensive; however, less costly alternatives can produce molds, which can then be used to create metal parts.

One of the most popular techniques, Fused Filament Fabrication (FFF), also known by the trademarked term Fused Deposition Modeling (FDM), originated from processes such as automatic polymeric foil hot air welding, hot-melt gluing, and automatic gasket deposition. In FDM, models or parts are created by extruding small beads or streams of material that harden immediately to form layers. A filament of thermoplastic or other low melting-point material is fed into an extrusion nozzle (3D printer extruder), where it is heated to its melting temperature and deposited onto a build platform (see Fig. A.17).

Plastic remains the most common material for this type of printing, with a variety of polymers available, including acrylonitrile butadiene styrene (ABS), polycarbonate (PC), polylactic acid (PLA), high-density polyethylene (HDPE), PC/ABS blends, polyphenylsulfone (PPSU), and high-impact polystyrene (HIPS). Typically, these polymers are provided in filament form and are made from virgin resins, ensuring high quality and consistency.

Raise3D Pro 2 Plus

The Raise3D Pro2 Plus (Fig. A.18) is one of the best affordable large-format 3D printers. Thanks to its substantial build volume of 305 × 305 × 605 mm, the Pro2 Plus allows for the printing of large-dimension components or a large number of small-dimension components.

The most important advantages of this printer are:

- Excellent quality and accuracy of up to 10 microns (0.01 mm).

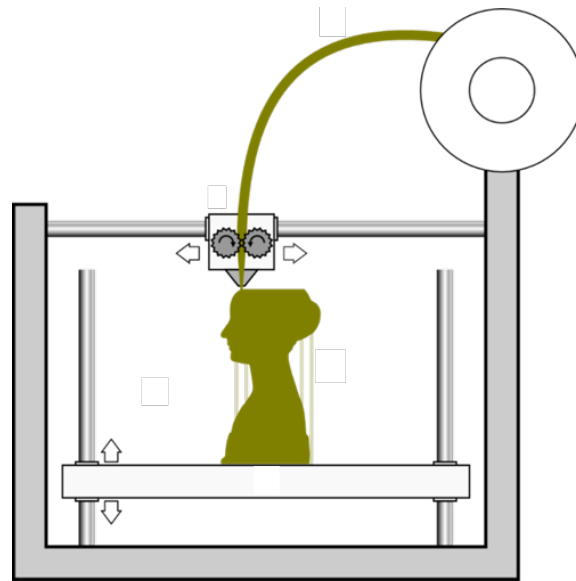


Figure A.17: Representation of the material extrusion process in additive manufacturing

- Improved print quality thanks to the extruder lifting system.
- The filament sensor ensures the printer automatically pauses when the filament runs out.
- Huge build volume of $305 \times 305 \times 605$ mm.
- All-metal hot end for print temperatures up to 300°C , allowing for a wide choice of filaments.
- Wireless control via Wi-Fi.
- Power failure is no longer a problem thanks to the power failure protection.
- Monitor the 3D print job anywhere, anytime with the built-in camera.

Ideamaker Software

IdeaMaker is a 3D slicer software with a user-friendly interface. It provides limitless customization. Creating an STL file by the CAD file, it is possible to import the component in IdeaMaker. Once the component has been imported, the software automatically generates support structure to sustain the component. The software has four default printing modes, as shown in Fig. A.19. Each one of them has different parameters for inner speed, external speed, density, layer thickness, geometry etc... IdeaMaker also allows to arrange the component in the workspace in limitless way: it is possible to try more alternatives to get the best configuration. Finding the best solution allows to save time and material, as different configurations can have different support structures, as shown in Fig. A.20 and Fig.A.21.



Figure A.18: Raise3D Pro 2 Plus (125)

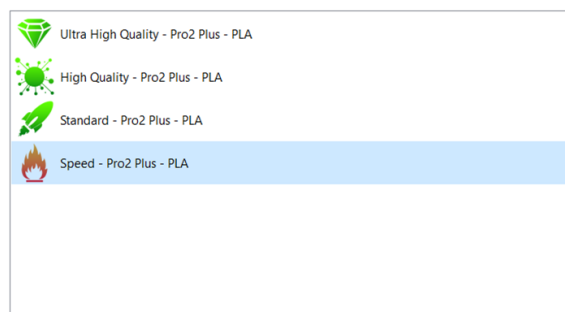


Figure A.19: Raise3D Pro 2 Plus default printing modes (125)

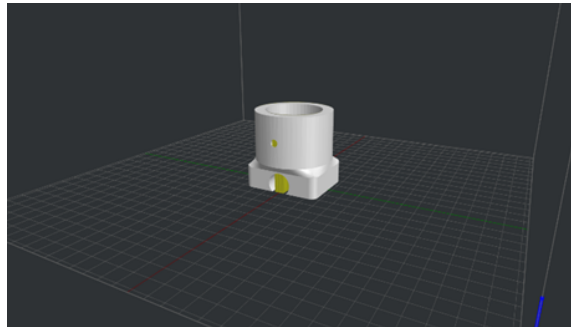


Figure A.20: Raise3D Pro 2 Plus default printing modes: top-view (125)

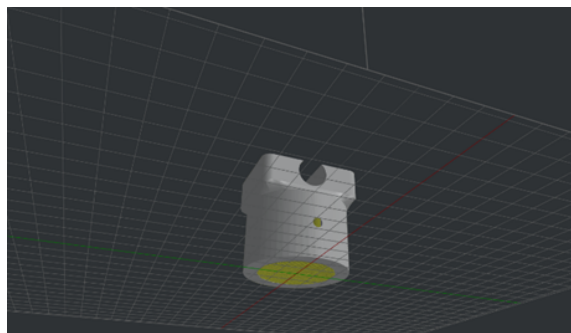


Figure A.21: Raise3D Pro 2 Plus default printing modes: bottom-view (125)

A.3.2 Motor and Control Systems

In order to effectively manage the power units of the rover, with particular attention to the motor-wheel system, it is essential to integrate a number of key components. These include the control unit, which in this case is represented by an Arduino Mega microcontroller, the drivers for power management and motor control signals, and finally the motors themselves. The control unit plays a central role in the system, acting as the primary processing and coordinating entity. As previously described in detail in the preceding sections, the Arduino Mega microcontroller was selected for its processing capabilities, versatility, and compatibility with the drivers and sensors utilized in the project. The drivers, in contrast, represent the interface between the control unit and the motors. They are responsible for translating control signals from the control unit into commands that are appropriate for the motors, while also managing the power required for their operation. This involves regulating the voltage and current supplied to the motors, as well as controlling the direction and speed of rotation. The motors represent the final actuators of the system, transforming electrical signals into mechanical movement, which is necessary for the traction of the rover. The selection of motors is of critical importance in order to guarantee the requisite power and efficiency, particularly in the context of rough terrain applications. This section will concentrate on the drivers and motors, exploring their functionality, technical specifications and the part they play in ensuring the correct operation of the motor-wheel system.

A.3.2.1 Drivers

In the context of motor interfacing with controllers, one of the primary requirements is that the controller operates at a low voltage and draws a small amount of current, whereas the motors require high voltage and current for their operation. In other words, the output of the controller is unable to drive the motor directly. For this reason, direct interfacing between controllers and motors is not a viable solution.

To address this issue, a motor driver circuit is employed. Motor driver circuits serve the function of current amplifiers, acting as a bridge between the controller and the motor in a motor drive system. These circuits are typically constructed using discrete components integrated into an integrated circuit (IC).

The input to a motor driver circuit is a low-current signal derived from the controller. The primary function of the circuit is to use that signal to manage the motor with the power signals. This enables the motor to operate in an optimal manner, regardless of whether it is a brushless DC motor, a brushed DC motor, a stepper motor, or another type of DC motor.

In order to facilitate the project, it has been used DBLS-01S motor drivers (Fig. A.22). The function of these drivers is to act as an interface between the controller and the motors, ensuring optimal operation by delivering the requisite voltage and current. This driver has a maximum power of 200W and can work between 24V and 48V, as shown in Tab. A.1.



Figure A.22: DBLS-01S Brushless DC Motor Driver (126)

Voltage Input	24V DC	36V DC	48V DC
Continuous Output Current	8.3A	5.5A	4.2A
Max. Output Power	200W	200W	200W
Peak Output Current	13A/3S	13A/3S	13A/3S

Table A.1: DBLS-01S Datasheet

A.3.2.2 Brushless DC Motors

The DC motors utilized in this project were procured from three hoverboards, exhibiting a design comparable to the one in Fig. A.23. These motors were selected on the basis of their high efficiency, compact design, and ability to deliver the requisite torque and power for the operation of the rover. It was necessary to disassemble the DC motor-wheel system from the hoverboard, as illustrated in Fig. A.24. This stage of the process ensured the integration of the motor-wheel assembly into the design of the rover, while maintaining its functionality.

Three of the seven wires are dedicated to the three-phase connection, while the remaining four wires are used to obtain the encoder signal.

An encoder is a sensing device that provides feedback to a motion control system. The function of the encoder is to convert motion into an electrical signal that can be interpreted by a control device, such as a counter or programmable logic controller (PLC). The feedback signal transmitted by the encoder is fundamental for the determination of



Figure A.23: Hoverboard



Figure A.24: DC motor-wheel subsystem used for propulsion in the Full-Scale Rover prototype

essential parameters, including position, count, speed, and direction (in the event that the encoder comprises two code disks).

For a more detailed understanding of the functioning of an encoder, please refer to Fig.A.25. The figure illustrates the process by which motion is translated into an electrical signal, highlighting the interaction between the moving parts of the encoder and its sensing mechanism. A beam of light is emitted from an LED and passes through the code disk, which is patterned with opaque lines (resembling the spokes on a bicycle wheel). As the encoder shaft rotates, the light beam from the LED is interrupted by the opaque lines on the code disk, preventing it from reaching the photodetector assembly. This results

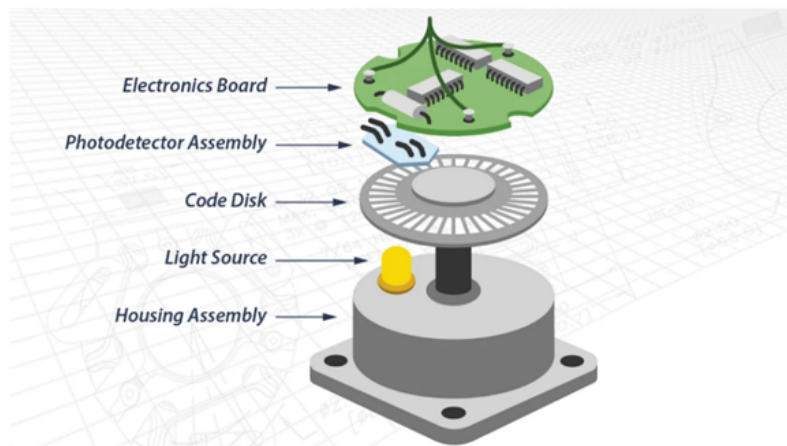


Figure A.25: Schematic representation of an encoder (127)

in the generation of a pulse signal, whereby the presence of light is indicative of a 'on' state, while the absence of light corresponds to an 'off' state. The signal is conveyed to the counter or controller, which then transmits the signal to initiate the desired function. Given that the drivers are less powerful than the motors, it is not feasible for the motors to reach the maximum value of torque, which is the stall torque, as illustrated in Fig. A.26.

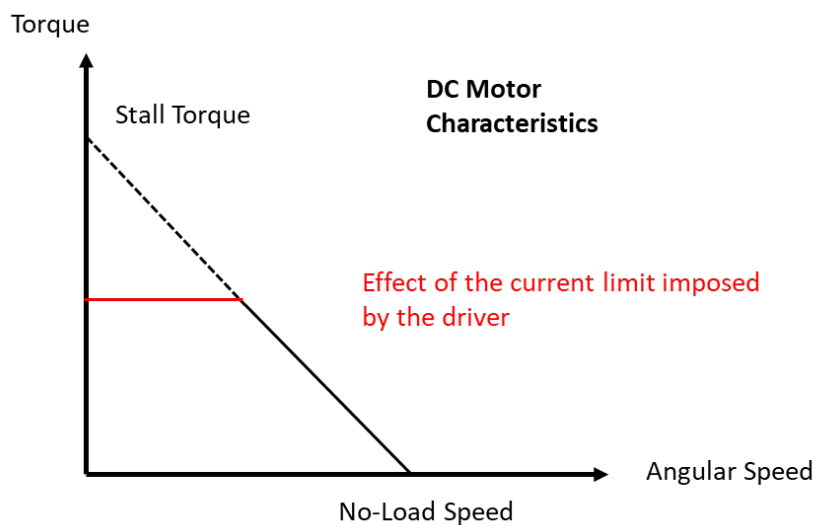


Figure A.26: DC motor characteristics with the effect of the current limit imposed by the driver on torque and angular speed

A.3.3 Lithium-Ion Battery System

Lithium-ion batteries (Fig. A.27) represent one of the most prevalent rechargeable battery chemistries in current use. Lithium-ion batteries are used extensively to power a wide

range of devices, including mobile phones, laptops, and electric vehicles, due to their high energy density and long cycle life.

A lithium-ion battery is typically composed of one or more lithium-ion cells in conjunction with a protective circuit board. The circuit board is of great importance in ensuring the safety and reliability of the battery, as it prevents overcharging, over-discharging, and excessive current flow, which could otherwise damage the cells or create hazardous conditions.

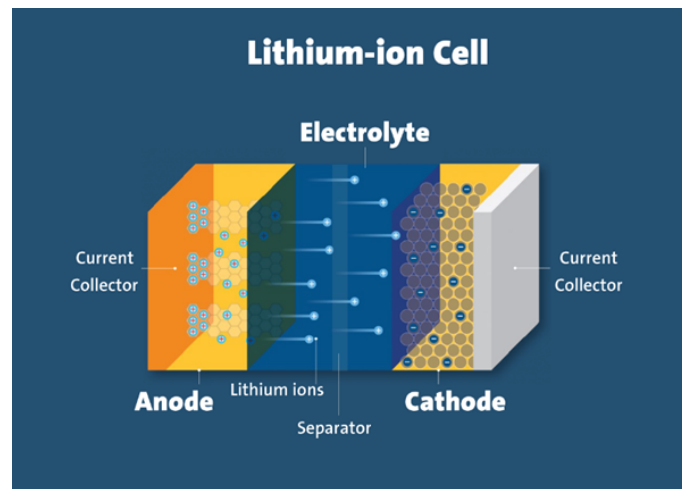


Figure A.27: Components of a lithium-ion battery: anode, cathode, electrolyte, and separator (128)

A Lithium-Ion battery is made up of:

- **Electrodes:** The positively and negatively charged ends of a cell. Attached to the current collectors.
- **Anode:** The negative electrode.
- **Cathode:** The positive electrode.
- **Electrolyte:** A liquid or gel that conducts electricity.
- **Current collectors:** Conductive foils at each electrode of the battery that are connected to the terminals of the cell. The cell terminals transmit the electric current between the battery, the device, and the energy source that powers the battery.
- **Separator:** A porous polymeric film that separates the electrodes while enabling the exchange of lithium ions from one side to the other.

In a lithium-ion battery (Fig. A.28), lithium ions (Li^+) migrate internally between the cathode and anode, while electrons flow in the opposite direction through the external

circuit. The migration of lithium ions generates an electrical current, which in turn powers the connected device.

During the discharge phase, the anode releases lithium ions to the cathode, thereby creating a flow of electrons in the external circuit that supplies energy to the device. The process is reversed during charging: lithium ions are released by the cathode and move to the anode, where they are stored until the battery is discharged again. The lithium-ion battery used in this project was sourced from the hoverboard. It has a nominal voltage of 25.2V and a rated capacity of 4Ah, providing sufficient power to meet the operational requirements of the rover.

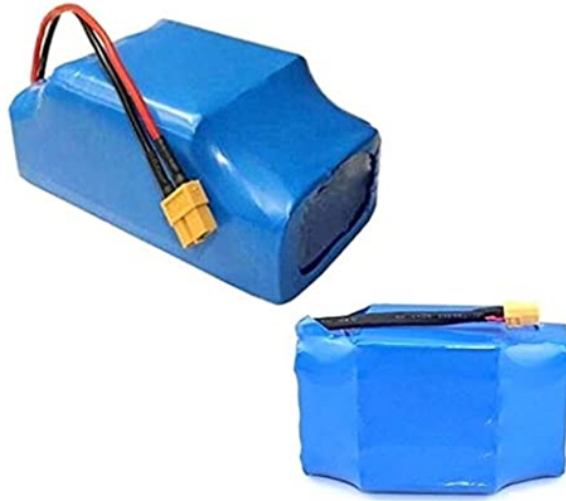


Figure A.28: Lithium-Ion Battery

The power supply system represents a critical component in the design of any mechatronic system, as it ensures the delivery of the energy required for the operation of motors, controllers, and sensors. For the rover, a stable and efficient power supply is essential to ensure the continued functionality of the vehicle, particularly during demanding tasks such as traveling uneven terrain or performing extended operations.

This section presents an overview of the physics underlying the lithium-ion battery utilized in the rover model, accompanied by an exposition of the mathematical equations employed to simulate its operational behavior. The aforementioned equations address key aspects, including charge and discharge dynamics, internal resistance, and voltage-current relationships. This provides insight into the manner in which the battery supports the overall performance of the rover.

For the Li-ion battery, the Discharge and Charging model are as in eq. A.1 and eq. A.2.

$$f_1(i_t, i^*, i) = E_0 - K \cdot \frac{Q}{Q - i_t} \cdot i^* - K \cdot \frac{Q}{Q - i_t} \cdot i_t + A \exp(-B \cdot i_t) \quad (\text{A.1})$$

$$f_2(i_t, i^*, i) = E_0 - K \cdot \frac{Q}{i_t + 0.1Q} \cdot i^* - K \cdot \frac{Q}{Q - i_t} \cdot i_t + A \exp(-B \cdot i_t) \quad (\text{A.2})$$

with:

- E_0 [V]: the constant voltage, in V.
- K [V/Ah]: the polarization constant, in V/Ah, or polarization resistance, in Ohms.
- i_t [A]: the low-frequency current dynamics, in A.
- i^* [A]: the battery current, in A.
- i [A]: the extracted capacity, in Ah.
- Q [Ah]: the maximum battery capacity, in Ah.
- A [V]: the exponential voltage, in V.
- B [1/Ah]: the exponential capacity, in Ah⁻¹.

A typical discharge curve is shown in Fig. A.29.

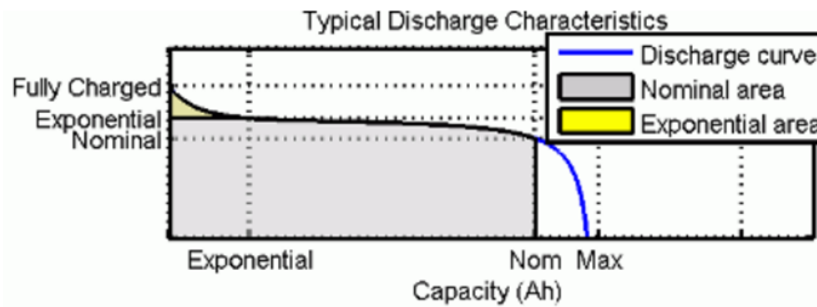


Figure A.29: Discharge curve for a Li-ion battery (129)

The curve is made up of three sections:

- The first section represents the exponential voltage drop when the battery is charged. The width of the drop depends on the battery type.
- The second section represents the charge that can be extracted from the battery until the voltage drops below the battery nominal voltage.
- The third section represents the total discharge of the battery, when the voltage drops rapidly.

When the battery current is negative, the battery recharges, following a charge characteristic as in Fig. A.30.

The state of charge (SOC) for a battery is a measure of the battery's charge, expressed as a percent of the full charge. The depth of discharge (DOD) is the numerical complement of the SOC, such that:

$$\text{DOD} = 100\% - \text{SOC}.$$

For example, if the SOC is:

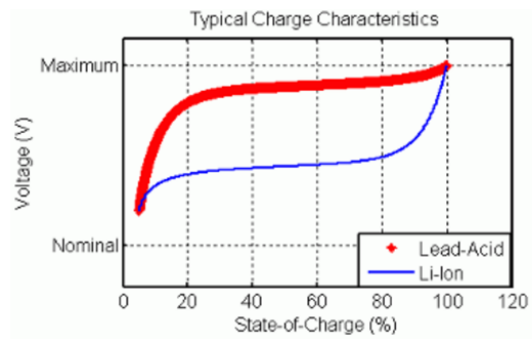


Figure A.30: Charging curve for a Lead acid (red) and a Li-ion (blue) battery (129)

- **100%:** The battery is fully charged, and the DOD is 0%.
- **75%:** The battery is 3/4 charged, and the DOD is 25%.
- **50%:** The battery is 1/2 charged, and the DOD is 50%.
- **0%:** The battery is completely discharged, and the DOD is 100%.

A.4 Hardware for Hand-EYE Calibration

This section presents the hardware components used for hand-eye calibration, which is essential to accurately align the coordinate system of the robotic arm with the vision system. The setup includes a robotic arm, a camera system and calibration targets to ensure precise spatial correspondence for autonomous operation.

A.4.1 Robotic Arm Integration

The Dobot Magician (see Fig. A.31) is utilized as the robotic manipulator. This robot, which has four axes of movement, is recognised for its precision and versatility in educational and light industrial applications. The key specifications of the device are as follows:

- Payload Capacity: 0.5 kg, suitable for handling lightweight tools and markers.
- Maximum Reach: 320 mm, allowing operation within a compact workspace.
- Repeatability: ± 0.2 mm, ensuring consistent positioning accuracy.



Figure A.31: Dobot Magician (130)

The user-friendly control interface and consistent performance of Dobot Magician make it an appropriate choice for calibration experiments that require precise and repeatable movements. The straightforward implementation of the system allows for rapid deployment in experimental setups. Although the Dobot Magician is an effective platform for this study, the calibration methodology is designed to be adaptable and can be transferred to more advanced robotic systems with higher performance capabilities.

A.4.2 Vision System Configuration

The vision system consists of the Intel RealSense D415 depth camera (Fig. A.32), which is known for its high-resolution depth sensing capabilities. The system boasts a number of key features, including:

- Depth Technology: Active stereo with a standard field of view, providing depth perception through dual infrared sensors and an RGB sensor (as showed in Fig. A.33).
- Depth Resolution: Up to 1280×720 pixels, delivering detailed depth information.
- Depth Range: Optimal performance between 0.16 m to 10 m, accommodating various workspace sizes.



Figure A.32: Intel Realsense D415 camera (131)

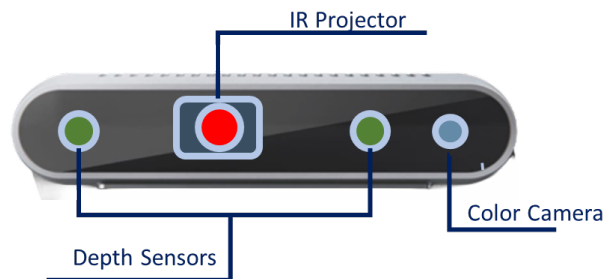


Figure A.33: Stereo Active technology in the Intel Realsense D415 (131)

The capacity of the D415 to capture both depth and RGB data enables the effective detection of markers and the conduct of spatial analysis during the calibration process.

A.5 Hardware for Navigation Applications

This section outlines the hardware components used for autonomous navigation, including visual perception systems, computing units and localisation sensors. The selected devices provide real-time data processing, environmental mapping and precise motion planning for efficient and adaptive rover navigation.

A.5.1 Materials for Visual Perception Systems

The cameras used are part of the Intel RealSense D400 series, a family of products that integrate the most advanced hardware and software solutions developed by Intel® for depth sensing, and are designed to be easily integrated into a variety of applications.

The hardware of both cameras is supported by an open source software development kit (SDK) called 'librealsense', which provides comprehensive and intuitive software support for the entire range of D400 Series cameras. This SDK allows users to take advantage of the functionality of the cameras to develop custom applications.

The cameras from the Intel RealSense D400 series will be used in this work: the D435, which will be used to implement visual SLAM (vSLAM), and the D415, already described in the A.4.2 section, which will be used to detect the movement of the rover from a fixed external position. The latter will be used to monitor the movement of the rover in its operational environment. Both Intel RealSense D400 series cameras are based

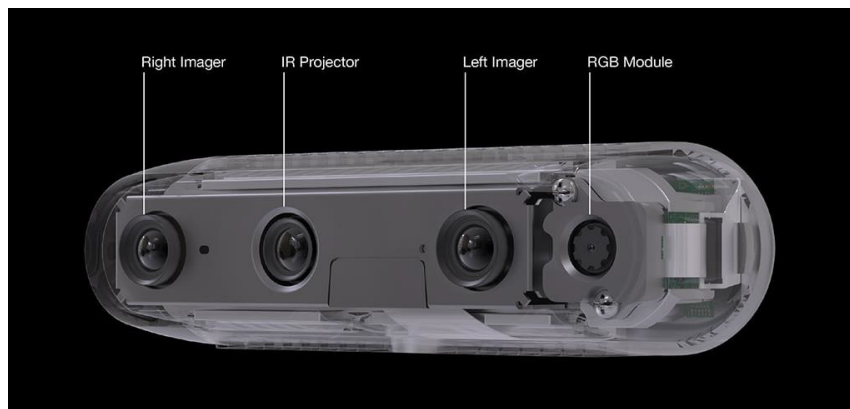


Figure A.34: D400 Modules in both cameras (131)

on the principle of active stereoscopy for depth calculation. The hardware structure of the cameras includes components like RGB Camera, left and right imager and IR Projector (see Fig.A.34). The RGB Camera is used to capture high-resolution colour images, providing detailed visual data that can be combined with depth information for advanced machine vision applications. Both cameras are equipped with two identical image sensors, called the 'left imager' and 'right imager' respectively, designed to capture stereo images. These sensors work in synergy to calculate depth using stereo disparity techniques, exploiting the differences between the images taken from the two slightly

different viewpoints and thanks to the intrinsics compute the depth. This configuration is essential for accurate and reliable depth detection. In addition, the D435 is equipped with an Infrared Projector that projects an invisible IR pattern onto the scene, improving depth measurement accuracy in low-light or low-contrast scenes. The left and right cameras capture the scene simultaneously and send the image data to the on-board processor. The processor uses the received data to calculate depth values for each pixel by correlating the corresponding points between the left camera image and the right camera image. The result of this processing is a depth image that associates each pixel with a value representative of its distance from the camera. The accuracy of the depth calculation depends on the correct configuration and calibration of the sensors, in particular the intrinsic parameters of the cameras, which can be obtained either from data sheets or from calibration. Incorrect configuration or failure to fix the sensors can compromise the calibration between depth and RGB images, negatively affecting the quality and reliability of the measurements.

A.5.1.1 Depth Camera D435: Specifications and Integration

The Intel RealSense™ D435 depth camera features a wide Field of View (FOV), the largest in the D400 series. This feature makes it ideal for a wide range of applications, including robotic navigation, object recognition, and augmented and virtual reality, where capturing the largest possible scene is critical. Another key advantage of the D435 (Fig. A.35) is its global shutter, which makes it ideal for fast motion applications. It offers excellent sensitivity in low-light conditions, enabling robots to navigate in dark environments and reducing motion blur in captured images. Some important key features are:



Figure A.35: D435 Model (131)

- Versatile Usage: Suitable for both indoor and outdoor environments.

-
- **Global Shutter Technology:** With $3\ \mu\text{m} \times 3\ \mu\text{m}$ pixel size, ensuring high-quality imaging.
 - **Depth Range:** Effective up to approximately 10 meters, with a minimum depth distance of just 0.105 meters.
 - **Resolution and Frame Rate:** Depth images can reach up to 1280×720 resolution at 90 frames per second, while RGB images are captured at 1920×1080 resolution at 30 frames per second.
 - **Design and Connectivity:** Compact design ($90\ \text{mm} \times 25\ \text{mm} \times 25\ \text{mm}$) with a USB-C 3.1 connection and multiple mounting options.

These features make the D435 a versatile and powerful tool for a wide range of vision-based applications.

A.5.2 Hardware and Computational Resources for LiDAR-Based SLAM for Single and Multi-Agent Systems

This section describes the hardware and computational resources used for LiDAR-based SLAM, allowing precise mapping and localisation in both single and multi-agent robotic systems. The setup includes high performance LiDAR sensors and embedded computing platforms to ensure efficient real-time processing and autonomous navigation.

A.5.2.1 Scout Mini AgileX Overview

The AgileX Scout Mini (Fig. A.36) is a compact and versatile UGV designed to operate effectively in difficult terrain. With four-wheel drive and independent suspension, it offers good maneuverability, with the ability to turn on itself with a zero turning radius thanks to the use of skid-steering as a steering method. In Fig. A.37 are some of the specifications of



Figure A.36: Scout Mini AgileX (132)

the robot; the way to interface with the systems is to use the ROS framework to facilitate the integration of sensors and actuators (80).

Dimensions	612mm x 580mm x 245mm
Weight	26kg
Payload	10kg
Operating Temperature	-20°C to 60°C
IP Rating	IP22
Battery	24V15Ah
Power Supply Output	24V15A
Range (No Load)	10km
Communication	CAN and Serial Interfaces
Maximum Speed (No Load)	2.7m/s
Maximum Climbable Obstacle Height (No Load)	70mm
Maximum Climbable Slope (No Load)	30°
Minimum Ground Clearance	115mm
Motors	Four 150W Brushless Motors

Figure A.37: Specifications of the Scout Mini AgileX robot (133)

A.5.2.2 Velodyne LiDAR Puck VLP-16 Specifications

The Velodyne LiDAR Puck VLP-16 (Figure A.38) is the LiDAR used to generate and utilize the point cloud within the Li-SLAM algorithm used. The VLP-16 is a compact and lightweight 3D LiDAR sensor designed to provide accurate and reliable real-time measurements: widely used in mapping, robotics, security, and industrial automation. Solid-state LiDAR uses a technology called MEMS (Micro-Electro-Mechanical Systems) to direct the laser beam. The Puck sensor from Velodyne is an example of solid-state LiDAR. The Puck uses an array of lasers and detectors to emit pulses of light and measure the time it takes for them to return after hitting an object. This data is then processed to create a detailed three-dimensional map of the environment.

Some of the main technical features are:

- 16 channels
- 100 m maximum range
- Accuracy of ± 3 centimeters
- Approximately 300,000 points per second
- 360° horizontal field of view
- Rotating frequency between 5-20 Hz

The VLP-16 has a built-in web server that makes it easy to monitor and configure itself.



Figure A.38: Velodyne LiDAR Puck VLP-16 (134)



Figure A.39: NVIDIA Jetson AGX Xavier (135)

As mentioned above, the VLP-16 creates three-dimensional maps, provides robots with an accurate perception of their surroundings to facilitate autonomous navigation and obstacle avoidance, and is integrated into autonomous driving systems for object detection and classification.

In this project, the VLP-16 was fundamental to the Li-SLAM algorithm, not only in generating a detailed map but also in providing precise information for the generation of the odometry measurement, as will be seen later.

A.5.2.3 NVIDIA Jetson AGX Xavier for Processing

The integration of the NVIDIA Jetson AGX Xavier (Fig.A.39) platform with the Scout Mini robot provides a good solution for robotics development and research, in particular for the application that will be presented in this chapter.

This combination allows the robot to perform complex workloads, such as visual odometry, sensor fusion, localization and mapping, obstacle detection and path planning, which are essential in order to test the Li-SLAM algorithm under investigation.

Some of the key features of the NVIDIA Jetson AGX Xavier include:

- A GPU with NVIDIA Volta architecture, featuring 512 CUDA cores and 64 Tensor cores running at a maximum frequency of 1.377 GHz.
- As CPU, the NVIDIA Carmel ARM v8.2 64-bit 8-core processor with 8MB L2 cache and 4MB L3 cache, running at up to 2.0 GHz.
- A memory of 32GB LPDDR4x with a 256-bit bus and 136.5GB/s bandwidth.
- Configurable power profiles of 10W, 15W, and 30W to balance performance and power consumption.

The integration of the NVIDIA Jetson AGX Xavier on board the Scout Mini is a good fit for this project, providing excellent capabilities within the system to achieve its goals.

A.5.3 Exploration with Multi-Agent and Collaborative Robotic Systems Hardware

This section presents the hardware components used for multi-agent and collaborative robotic systems that provide coordinated exploration and interaction. The setup includes mobile robotic platforms, communication interfaces and sensor units that ensure effective team navigation and task execution. The system used for this application is a TurtleBot 4 (Fig. A.40) equipped with sensors to support autonomous navigation. The TurtleBot 4 is a ROS 2 based mobile robot designed for educational and research applications. Its hardware configuration includes an iRobot® Create3 mobile base, a Raspberry Pi 4 running ROS 2, a 2D LiDAR, and several other integrated sensors. An Intel® RealSense™ D435i (the same explained in the past of this thesis but with IMU) camera has been added to this equipment to further improve its capabilities.

With these sensors, TurtleBot 4 can map its environment, navigate autonomously, run artificial intelligence models through the camera, and perform many other advanced tasks.

A.5.3.1 iRobot® Create3 for Multi-Agent Framework

The iRobot® Create® 3 (Fig.A.41) is an advanced mobile robotics platform with a complete set of integrated sensors and actuators designed to enable users to effectively develop and test robotic algorithms. A key element of this platform is its integration with ROS 2, which provides a robust and flexible software foundation. Data collected from sensors is published to specific ROS 2 topics, while servers and subscriptions facilitate control of actuators and management of user requests. The robot is also equipped with pre-programmed autonomous behaviors, such as docking to the docking station, following the wall, and handling obstacles. These behaviors can be activated or configured through ROS 2 actions and parameters, making the system highly adaptable to different application scenarios. Sensors integrated into the platform include two front impact sensors that allow the robot to detect direct contact with objects. For odometry estimation, there are two



Figure A.40: Turtlebot 4 (109)

dedicated wheel tracking encoders for accurate motion estimation, a downward facing optical flow sensor, and a 3D gyroscope and 3D accelerometer to provide orientation and acceleration measurements. There are also four infrared sensors for edge detection and six infrared sensors for obstacle detection. The robot is equipped with three wheels, one

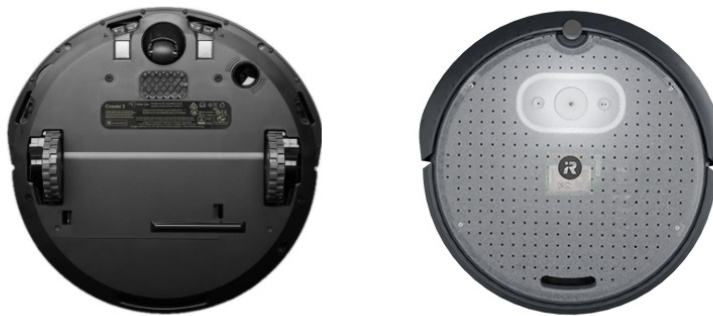


Figure A.41: iRobot® Create® 3 (109)

of which is idle. This configuration makes the steering system of the differentiated wheel type. The two independently driven motorized wheels allow the robot to change direction by changing the relative speed of rotation of the wheels, ensuring precise and effective maneuverability.

A.5.3.2 Raspberry Pi 4 and User Interface PCBA

The Raspberry Pi 4 Model B (see Fig. A.42) is a single-board computer designed by the Raspberry Pi Foundation to deliver high performance in a compact, low-cost format. It is equipped with a Broadcom BCM2711 processor, a 64-bit quad-core ARM Cortex-A72 running at 1.5GHz. The memory is available in different configurations with 1GB, 2GB, 4GB and 8GB LPDDR4-3200 SDRAM options, providing more flexibility to meet specific user needs. In this case is by 4GB. For connectivity, Raspberry Pi 4 offers Gigabit Ethernet, support for dual-band 802.11ac (2.4 GHz and 5.0 GHz) wireless networks, and Bluetooth 5.0. The board has two USB 3.0 ports and two USB 2.0 ports, allowing to connect a wide range of peripherals. It also has two micro-HDMI ports, each capable of supporting a display with up to 4K resolution. Raspberry Pi 4 maintains compatibility with previous versions, including the 40-pin GPIO header, making it easy to integrate with existing hardware and accessories (136).



Figure A.42: Raspberry Pi 4B (136)

In the case of the TurtleBot 4, the Raspberry Pi is paired with a User Interface Board to further extend its functionality. This combination allows the user to provide intuitive and direct control of the Create® 3 robot.

The User Interface Board (Fig. A.43) not only facilitates robot control, but also acts as a versatile expansion platform. This board can be used to connect and manage additional components, sensors, and other devices, making the system highly customizable. This approach provides the flexibility to adapt the TurtleBot 4 to specific application needs, allowing new features to be implemented and the capabilities of the robot to be expanded based on project requirements.

A.5.3.3 RPLIDAR A1M8 Integration

The RPLidar A1M8 (Fig. A.44) is a 2D state-of-the-art 360 degree laser sensor developed by Slamtec. With its ability to perform full scans of the environment within a radius of 12 meters, this device is particularly suitable for applications such as mapping and implementation of SLAM algorithms. The accuracy and reliability of the sensor make it a good tool for robotics and automation in both static and dynamic environments. Unlike

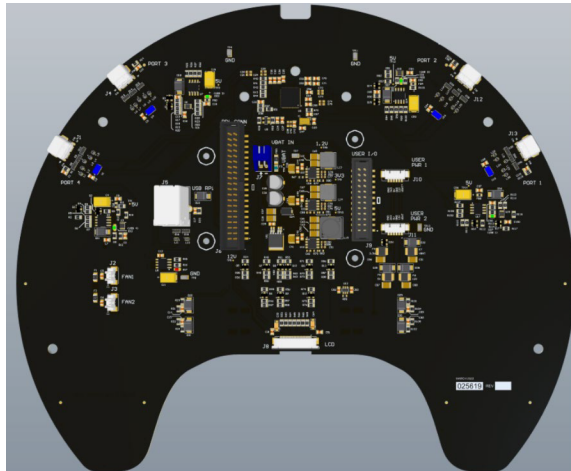


Figure A.43: User Interface PCBA (109)



Figure A.44: RPLidar A1M8

the LiDAR in the previous application, this LiDAR uses moving mechanical component technology. Its operation is based on laser triangulation (Fig. A.45): the sensor emits pulses of laser light and measures the time it takes for the reflections to return to the sensor after hitting surrounding objects. Through this process, LiDAR can acquire highly accurate data and create a detailed 2D map of the environment and objects present, making it a key tool for applications such as robotics, mapping, and autonomous navigation.

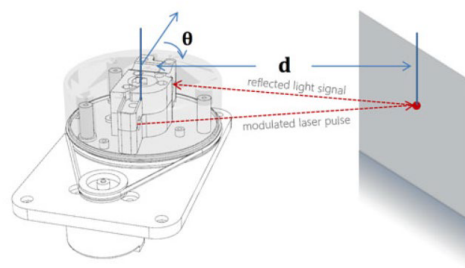


Figure A.45: Triangulation for the LiDAR

List of Figures

1.1	Classification of robotic mobility systems based on locomotion types (4)	6
1.2	Nanokhod (5)	7
1.3	Lobhaz DT3 (KIST)	8
1.4	Spot, Boston Dynamics (6)	8
1.5	Robot RHex (7)	9
1.6	Sojourner (8)	10
1.7	CRAB suspension equipped on a Rover (9)	10
1.8	Working Scheme of the CRAB Suspension(9)	11
1.9	Rocker-Bogie Suspension System	11
1.10	Ackermann Steering Principle (11)	13
1.11	Explicit Configuration (Left) vs Skid Steering Configuration (Right) (12)	13
1.12	Computer vision system process, from input image to object recognition output	15
1.13	Pinhole model (17)	17
1.14	Prospective model showing the projection of 3D coordinates (P) onto the image plane (P')	17
1.15	Thin Lens physics	19
1.16	Meaning of Extrinsic and Intrinsic (19)	19
1.17	Projection Model (18)	20
1.18	Stereo Camera	22
1.19	RB-VOGUI+ (42)	25
1.20	Schematic representation of a basic PID control scheme	28
1.21	Generic proportional controller scheme	29
1.22	Phase portrait illustrating the dynamics of Sliding Mode Control (46)	30
1.23	Sliding surface, reaching mode, and high-frequency switching in Sliding Mode Control (47)	31
1.24	Block diagram of the Sliding Mode Control system (47)	32
1.25	Autonomous Rover (Left) and Teleoperated Rover (Right)	33
1.26	Waymo-operated Jaguar I-Pace (50)	35
1.27	Automated Guided Vehicles during a facility task (51)	35
1.28	Robot in hazardous environment (42)	36
1.29	Mobile robot for agricultural operations	36

1.30	Robot cleans up litter (55)	37
1.31	NASA Rover Perseverance (57)	37
1.32	A vehicle taking relative measurements to environmental landmarks (58)	38
1.33	Flight of the robobee	40
1.34	Rover Lunkkhod	41
1.35	Rover Marskhod 1 (66)	42
1.36	Perseverance (57)	43
1.37	Mobile Robots in Wharehouses	43
1.38	BeBot (69)	44
1.39	Contact Force in the Simscape Multibody Model (70)	48
1.40	Variation of the friction coefficient with tangential velocity (70)	49
1.41	Structural components of an electric motor, including the stator, rotor, winding, commutator, and brushes(74)	50
1.42	Right-hand rule showing directions of force, magnetic field, and current (74)	50
1.43	Diagram of an electric motor showing the interaction of permanent magnets, electromagnetic forces, and current flow (74)	51
1.44	Equivalent circuit of a DC motor	52
1.45	Representation of a generic signal $S(t)$ over time with discrete sampling points	56
1.46	Representation of a signal in the time domain and its corresponding frequency domain	58
1.47	Different types of USB ports and connectors	61
1.48	I ² C protocol network showing master-slave communication over SDA and SCL lines (77)	62
1.49	Example of the I ² C wiring diagram illustrating master-slave connections (78)	63
2.1	Rocker-bogie suspension system of a mobile robot	67
2.2	Obstacle Crossing: 1st phase (85)	68
2.3	Obstacle Crossing: 2nd phase (85)	69
2.4	Obstacle Crossing: 3rd phase (85)	69
2.5	Example of Rocker-Bogie system with steerable wheels	71
2.6	Kinematic diagram of the rocker-bogie suspension system	71
2.7	Composition of angles in the rocker-bogie mechanism	73
2.8	Geometric characteristics of the small prototype rover	74
2.9	Load distribution in the rocker-bogie suspension system under uniform weight (W)	75
2.10	Equilibrium analysis of the rocker-bogie system in the absence of obstacles	75
2.11	Load distribution in the rocker-bogie suspension system as Front Wheel approaches Obstacle)	76
2.12	Vertical reaction forces as a function of F_c	77
2.13	Configuration when $F_c > F_c^*$	77
2.14	Load distribution as Wheel B approaches the obstacle ($h_1 = r, h_2 = h_3 = 0$)	78

2.15	Load distribution as Wheel C approaches the obstacle	79
2.16	Rollover Boundary Condition: (a) on flat ground, (b) ascending an incline, (c) descending an incline	80
2.17	Stability analysis of the rocker-bogie mechanism: (left) on flat terrain, (right) at the tipping point during an obstacle climb	81
2.18	Multibody model of the rover system	84
2.19	Layout of the rover	85
2.20	3D model of the rover chassis	86
2.21	3D model of the Motor-Wheel	87
2.22	Simscape Drive Unite macroblock	88
2.23	DC Motor Simscape model	88
2.24	Specifications of the DC Motor	89
2.25	DC motor characteristic curves: (a) Slope variation indicates motor resis- tance changes; (b) Rightward shift indicates a voltage increase	90
2.26	Representation of Pulse Width Modulation (PWM) signal	91
2.27	Electronic tachometer used to measure the rotational velocity of a wheel	92
2.28	Stall torque characterization test for the motor of the rover	93
2.29	Torque-Velocity ($M-\omega$) plane for the DC motor at different voltage levels	94
2.30	Motor-wheel velocity for the Simscape Model	95
2.31	Measured Velocity Wheel $\hat{y}$	96
2.32	Parameters Optimization	97
2.33	Friction Coefficients: ground and obstacle step	97
2.34	Soil Model	98
2.35	Wheel Convex Hull	99
2.36	High-accuracy mesh model of the wheel	102
2.37	Medium-accuracy mesh model of the wheel	102
2.38	Low-accuracy mesh model of the wheel	103
2.39	Point cloud representation of a wheel in Simulink	103
2.40	Multibody simulation of the rover in straight-line motion	105
2.41	Center of gravity motion in the multibody model along X, Y, and Z axes over time.	106
2.42	Multibody simulation of the rover crossing a triangular obstacle with Bogie Front	107
2.43	Translation of the center of gravity in the case of the Bogie Front	107
2.44	Angular Velocities - Bogie Front	108
2.45	Multibody simulation of the rover crossing a triangular obstacle with Rocker Front	108
2.46	Angular Velocities - Rocker Front	108
2.47	Frequency spectrum of a ConvexHull-Wheel at a rotational speed of 3 rad/s	110
2.48	Zoom of Spectrum of a ConvexHull-Wheel at 3 rad/s	111
2.49	Frequency spectrum generated from point clouds in each simulation test	111
2.50	Zoom of Spectrum of a Point Cloud for each simulation test	112
2.51	FFT at 10 rad/s along Y Axis	113

2.52	Zoom FFT at 10 rad/s	113
2.53	FFT at 20 rad/s	114
2.54	Zoom FFT at 20 rad/s	114
2.55	FFT at 30 rad/s	114
2.56	Zoom FFT at 30 rad/s	115
2.57	FFT at 15 and 10 rad/s	115
2.58	Zoom in FFT at 15 and 10 rad/s	116
2.59	FFT at 15 and 5 rad/s	117
2.60	Zoom in of FFT at 15 and 5 rad/s	117
2.61	Schematic diagram of the rover showing the imbalance caused by the robotic arm	120
2.62	Rocker-bogie reaction forces in the absence of obstacles	120
2.63	Reaction forces as a function of eccentricity (e)	121
2.64	Configuration of the rover as it approaches the obstacle with Wheel A	122
2.65	Reaction forces as a function of eccentricity (e) while Wheel A approaches the obstacle	123
2.66	Nonlinear motion causing the unloading of the central wheel	124
2.67	Configuration of the rover as it approaches the obstacle with Wheel B	124
2.68	Parameters of the rover configuration as it approaches the obstacle with Wheel B	125
2.69	Configuration of the rover as it approaches the obstacle with Wheel C	127
2.70	Configuration of the rover as it approaches the obstacle with Wheel C, avoiding tipping	128
2.71	Multibody model of the robotic arm in Simscape	129
2.72	Simscape model of a robotic arm	130
2.73	Rover model with an empirical robotic arm	132
2.74	Simulation frames of the rover model with an "empirical" robotic arm	133
2.75	Simulation frames of the rover model with an "analytical" robotic arm	134
2.76	Mass distribution of the robotic arm	135
2.77	Configuration of the robotic arm during a climbing-up maneuver	137
2.78	Robotic-Arm anti-overturning configuration	139
2.79	CAD model of the initial rover design	143
2.80	Side view of the initial rover	143
2.81	Front view of the initial rover	143
2.82	CAD model of a single Rocker-Bogie suspension	144
2.83	CAD model of the Rocker subsystem	144
2.84	Components of the Rocker subsystem	145
2.85	CAD model of the Bogie subsystem	145
2.86	Components of the Bogie subsystem	146
2.87	Uniform rotation of the Bogies	146
2.88	Opposite rotation of the Bogies	146
2.89	Independent rotation of the Bogies	147
2.90	CAD model of the differential system	147

2.91	CAD model of the differential rod	147
2.92	Detailed view of the pivot in the differential system	148
2.93	Differential Mechanism in Simscape environment: Isometric view	148
2.94	Differential Mechanism in Simscape environment: Side view	148
2.95	Connection using spherical joints in the linkage system	149
2.96	Visualization of spherical joint rotation with reference frames	149
2.97	Wheel-Motor subsystem: Isometric view	150
2.98	Wheel-Motor subsystem: Side view	151
2.99	Wheel-Motor Subsystem with tubular bar attachment for structural support	151
2.100	Macro-blocks structure of the rover model illustrating the modular con- figuration of the system	152
2.101	Multibody structure of the rover highlighting its modular and intercon- nected components	154
2.102	Multibody structure of the Central Body, illustrating the key components and their interconnections	155
2.103	Reference frame for the upper cap of the differential joint	156
2.104	Reference frames along the Differential Bar	156
2.105	Multibody model representation of the spherical joint link, illustrating the structural and dynamic connections	157
2.106	Multibody model of the rocker-bogie subsystem, showing its components	157
2.107	Multi-domain model of the DC motor, integrating mechanical and electri- cal systems	158
2.108	Multibody model of the wheel (Case 1) illustrating the subsystem design with a File Solid block for CAD model import and a point cloud representation	158
2.109	Graphical representation of the wheel using a point cloud with 100 cir- cumferential points	159
2.110	Zoomed view of the graphical representation of the point cloud on the wheel surface	159
2.111	Multibody model of the wheel (Case 2) illustrating the subsystem design with a File Solid block for CAD model import as Convex Hull	160
2.112	Comparison between the true geometry and its convex hull approximation	160
2.113	Control system logic diagram for each wheel, illustrating the PID con- troller and voltage regulation	161
2.114	Parameter settings for the "Spatial Contact Force" block, including normal force and frictional force configurations	162
2.115	Grid surface representation illustrating X, Y grid vectors and Z height values	163
2.116	Representation of a surface using a Brick Solid block in the simulation environment	163
2.117	IMU Sensor sub-block integrated within the multibody model	164
2.118	Six-DOF joint block enabling unconstrained 3-D translation and rotation	164
2.119	Point cloud representation highlighting the geometry and surface details of the component	166
2.120	Simulation 1: 200 points	167

2.121 Simulation 7: 12800 points	167
2.122 Relationship between the number of circumferential points and simulation time, including a fitted curve for better trend analysis	168
2.123 Angular distance θ	168
2.124 Simulation 1: 31 circumferential lines	169
2.125 Simulation 5: 1 circumferential line	169
2.126 Fitted curve showing the relationship between the number of circumfer- ential lines and the simulation time	170
2.127 Simulation 3: $\theta = 1^\circ$	171
2.128 Simulation 5: $\theta = 0.01^\circ$	171
2.129 Simulation 5: $\theta = 0.01^\circ$, zoom	171
2.130 Fitted curve showing the relationship between the angular distance θ and the simulation time	172
2.131 Torque profile of the front motors during obstacle crossing, with a refer- ence speed of 1 rad/s for each wheel	173
2.132 Obstacle crossing, Bogie Front: Case 1	174
2.133 Torque profiles of the front, middle, and rear motors during obstacle crossing with Bogie Front: Case 1	175
2.134 Obstacle crossing, Bogie Front: Case 2	175
2.135 Torque profiles of the front, middle, and rear motors during obstacle crossing with Bogie Front: Case 2	176
2.136 Obstacle crossing, Rocker Front: Case 1	176
2.137 Torque profiles of the front, middle, and rear motors during obstacle crossing with Rocker Front: Case 1	177
2.138 Obstacle crossing, Rocker Front: Case 2	177
2.139 Torque profiles of the front, middle, and rear motors during obstacle crossing with Rocker Front: Case 2	178
2.140 Representation of an irregular surface using a grid-based model	178
2.141 Representation of an irregular surface using a grid-based model	179
2.142 Position of the center of the rover over time along the x, y, and z axes during traversal of uneven terrain	179
2.143 Zoomed view of the position of the rover along the z_c and y_c axes over time	180
2.144 Speed profiles of the six wheels during traversal of uneven terrain	180
2.145 Battery parameters configuration	181
2.146 Energy consumption in Case 1, showing the state of charge for the front, middle, and rear batteries over time	181
2.147 Rover navigating an inclined surface with a slope of approximately 20° , used to evaluate performance and energy consumption under challenging conditions	182
2.148 Energy consumption in Case 2, showing the state of charge for the front, middle, and rear batteries over time	183
2.149 Rover position over time along the x, y, and z axes during traversal of an inclined surface	183

2.150 First Prototype Configurations	184
2.151 Speed reference profile over time, illustrating the target rotational speed in rad/s	185
2.152 Visualization of speed differences across the wheels for the second config- uration	185
2.153 Visualization of speed components for the third configuration, highlight- ing individual wheel velocities	186
2.154 Speed for First Prototype Configurations	187
2.155 Front Motors Torque for the four configurations	188
2.156 Middle Motors Torque for the four configurations	189
2.157 Rear Motors Torque for the four configurations	190
2.158 Curve scheme for the two-wheeled robot, illustrating the skid-steering behavior with differential drive	191
2.159 Speed Reference	192
2.160 (a) Speed trend for the six motors and (b) trajectory curve for the first configuration	193
2.161 (a) Speed trend for the six motors and (b) trajectory curve for the second configuration	194
2.162 (a) Speed trend for the six motors and (b) trajectory curve for the third configuration	195
2.163 (a) Speed trend for the six motors and (b) trajectory curve for the fourth configuration	196
2.164 Torque comparison of the front left (FL) and front right (FR) motors for the four configurations	197
2.165 Torque comparison of the middle left (ML) and middle right (MR) motors for the four configurations	198
2.166 Torque comparison of the rear left (RL) and rear right (RR) motors for the four configurations	199
2.167 (a) Speed and (b) torque trends for the six motors; (c) trajectory curve for the hybrid configuration	201
2.168 Hybrid configuration traveling a turned obstacle	202
2.169 JAKA Zu 5s (88)	204
2.170 Center of gravity of the first model rover shown from side and front views	204
2.171 Sketch illustrating the new dimensions of the rover design	205
2.172 Side view of the new rover model with updated dimensions	205
2.173 Front view of the new rover model with updated dimensions	206
2.174 Center of gravity of the new model rover shown from side and front views	206
2.175 Comparison of the center of gravity positions between the two rover models	207
2.176 First model of the rover frame design	207
2.177 Plate constructed using triangular aluminum corner pieces to enhance stability and structural integrity	208
2.178 Isometric and side view of the Bosch	209
2.179 Plate made with the use of triangular angles	209

2.180 Connector angle enabling the assembly of Bosch profiles for the central body plate	210
2.181 Angles used for "L" and "T" connections in the assembly of Bosch profiles	210
2.182 Plate-suspension system connection using L-joints	211
2.183 Connection with L-joints rotated 180 degrees	211
2.184 Plate-suspension system connection via L-joints	212
2.185 Connection with 'L' joints rotated 180	212
2.186 Plate-suspension system connection using a collar	213
2.187 Collar and cap used for connecting the panel to the suspension system .	213
2.188 Comparison of the old and new connections between the central body plate and the suspension system	214
2.189 L-shaped GAR-joint with a sectional view	214
2.190 GAR-Collar joint with sectional view	215
2.191 Tilting of the suspension system induced by shaft movement	215
2.192 GAR-collar cap linkage	216
2.193 Old spherical joint used in the suspension system connection	216
2.194 New joint used in the suspension system connection	217
2.195 Frontal View of the first prototype with the 'turret' highlighted	218
2.196 Side view of the latest rover prototype with the 'turret' highlighted . . .	219
2.197 Central body of the first rover	220
2.198 Central body of the latest rover	220
2.199 Comparison between the old and new links in the rocker-bogie suspension system	221
2.200 Comparison between the old and new GAR joint designs	222
2.201 Alignment between the center wheel axis and the GAR joint axis	223
2.202 Rocker-Bogie angle in the first model	224
2.203 Rocker-Bogie angle in the new model	225
2.204 Torques at GAR of the two banks: external wheels version	226
2.205 High torques experienced at the GAR during operation for the first rover	226
2.206 Torques at GAR of the two banks: rocker wheels internal version	227
2.207 Torques at of the two banks: central wheels external version	227
2.208 Torques at of the two banks: new rover model	227
2.209 Torques at of the two banks: new rover model with external central wheels	228
2.210 Torques at of the two banks: new rover model with robotic arm	228
2.211 Torques at of the two banks: new rover model with robotic arm and central external wheels	229
2.212 Motor-wheel	230
2.213 Front View and side view of the old support connecting the wheels to the suspension	230
2.214 Front View and side view of the new support connecting the wheels to the suspension	231
2.215 GRS Cap	231
2.216 "Grid Surface" specifications block	233

2.217	Ground in Simulation	233
2.218	Imposed speed Law for each motor	234
2.219	Starting position of the rovers	234
2.220	Motor impulse response at $t = 2$ s comparing the new and old front motors	235
2.221	Torque profile of the front wheels in the first model of the rover	235
2.222	Obstacle crossing phases for the first model of the rover at different time intervals	236
2.223	Comparative analysis of the torque generated by the front wheel motors in the new and old models	237
2.224	Comparative analysis of the torque generated by the middle wheel motors in the new and old models	237
2.225	Comparative analysis of the torque generated by the rear wheel motors in the new and old models	238
3.1	Configuration of the rover with two different battery packs	243
3.2	Configuration diagram of the prototype	244
3.3	Dashboard of the developed web server for remote control of the mobile robot	245
3.4	Osoyo App for remote control of the mobile robot	246
3.5	Acceleration plot at 100 PWM in the straight-line test, with Bogie Front .	247
3.6	Accelerations Straight-Line PWM: 100, 150, 200, 255	248
3.7	Angular velocities at 100 PWM input for the Bogie front configuration (Serial Port Acquisition)	249
3.8	Angular velocities at 100 PWM input for the Rocker front configuration (Serial Port Acquisition)	249
3.9	Prototype angles trajectory at Bogie front configuration as 100 PWM . .	250
3.10	Prototype angles trajectory at Rocker front configuration as 100 PWM .	250
3.11	Prototype angles trajectory at Rocker front configuration as 255 PWM .	251
3.12	Angles Straight-Line PWM: 150, 200, 255	252
3.13	Angles plot for steering maneuver (150 left 75 right) with bogie front configuration	253
3.14	Angular velocities Plot for steering maneuver (150 left 75 right) with bogie front configuration	254
3.15	Experimental setup illustrating the robot crossing an obstacle	254
3.16	Angle variations (roll, pitch, and yaw) during obstacle crossing at 150 PWM	255
3.17	Tachometer Rs-Pro Rm-1501	257
3.18	DC motor setup for sensor testing and calibration	258
3.19	FFT Analysis with major peak representation	258
3.20	Configuration diagram of the prototype for the tests	259
3.21	Angular velocities of the left and right wheels under PI controller, comparing reference and controlled outputs	261
3.22	Error trends of the left and right wheels under PI control	261
3.23	Voltage input trends for the left and right wheels under PI control	262

3.24	Angular velocities of the left and right wheels under Sliding Mode Control, comparing reference and controlled outputs	264
3.25	Error trends of the left and right wheels under Sliding Mode Control . .	265
3.26	Voltage input trends for the left and right wheels under Sliding Mode Control	265
3.27	Robot demonstrating obstacle crossing on the left bank	266
3.28	Angular velocity trends during obstacle crossing on the left bank under Sliding Mode Control	267
3.29	Error trends during obstacle crossing on the left bank under Sliding Mode Control	268
3.30	Voltage input trends during obstacle crossing on the left bank under Sliding Mode Control	268
3.31	FFT analysis of the slow bank	270
3.32	FFT analysis of the fast bank	271
3.33	Angular velocities of the left and right wheels during the steering test under Sliding Mode Control	272
3.34	Error trends of the left and right wheels during the steering test under Sliding Mode Control	272
3.35	Voltage inputs for the left and right wheels during the steering test under Sliding Mode Control	273
3.36	Rover prototype equipped with a robotic arm for enhanced obstacle overcoming capabilities	275
3.37	Configuration diagram of the prototype for crossing tests	276
3.38	Robotic Arm Scheme	276
3.39	Command flow	276
3.40	Test 1: Rover prototype positioned at the initial stage of the obstacle . . .	280
3.41	Measured pitch angle during the failure crossing test: raw signal (blue line) and Kalman-filtered signal (red line)	280
3.42	Obstacle crossing with a robotic arm in active configuration, wide obstacle: six consecutive frames illustrating the crossing process	281
3.43	Measured pitch angle during obstacle crossing with a robotic arm in active configuration: raw signal (blue line) and Kalman-filtered signal (red line)	282
3.44	Measured accelerations during obstacle crossing along (a) x-axis and (b) z-axis	283
3.45	Obstacle crossing with a robotic arm in active configuration, short obstacle: six consecutive frames illustrating the crossing process	284
3.46	Measured pitch angle during obstacle crossing with a robotic arm in active configuration, short obstacle: raw signal (blue line) and Kalman-filtered signal (red line)	284
3.47	Rover tilting during crossing stages: four consecutive frames.	285
3.48	3D printed model with a grid infill pattern at 20% density	289
3.49	3D printed model with a grid infill pattern at 50% density	289
3.50	3D printed model with a hexagonal infill pattern at 20% density	289

3.51	(left) Upper bogie joint and (right) lower bogie joint used in the suspension system	290
3.52	Body components – Rocker bogie link (left) and central joint (right) . . .	290
3.53	Wheel attachments and rocker joint	290
3.54	Bogie Joint subsystem	292
3.55	Rocker-Bogie subsystem	292
3.56	Body structure	293
3.57	Rover structure: isometric view	293
3.58	Rover structure: side view	293
3.59	Warping phenomenon observed during 3D printing	297
3.60	Control Panel of the 3D printer	298
3.61	Printing progress	299
3.62	Rocker-Bogie Suspension for the new prototype	300
3.63	Central body plate built from Bosch profiles and ABS corners	300
3.64	Isometric view of the Rover	301
3.65	Side view of the Rover	301
3.66	Frontal view of the Rover	302
3.67	Last rover prototype	302
4.1	Visual servo configurations as a pose graph. The coordinate frames are: world 0, end effector E, camera C, and goal G for the end-point open-loop configuration	308
4.2	Pose Graph for case study application	310
4.3	Experimental bench setup: (A) robot and camera reference frames, and (B) red color marker located on the robotic arm	311
4.4	Workbench setup for the case study	312
4.5	Sequence of images illustrating the movement of the robotic arm in the workspace	313
4.6	RGB Image	315
4.7	Depth Image	315
4.8	Invariant Calibrator	316
4.9	Color segmentation algorithm: (1) Red extraction, (2) grayscale image, (3) subtraction of steps 1 and 2, (4) binary image result	317
4.10	Fitting of the plane to the 3D data points: acquired points, ideal points, and marker center	319
4.11	Size of the artificial point cloud	321
4.12	Sheaf of parallel planes	322
4.13	LSQ Trend	324
4.14	Bull-eye diagram	324
5.1	Testbench for light detection	330
5.2	Acquired RGB frame from the external camera	331
5.3	Gray-scale image	332

5.4	Binary Image obtained from the segmentation process	332
5.5	Example of following binary image for motion perception	333
5.6	Setup for the vSLAM test	335
5.7	FAST method for keypoint extraction (99)	336
5.8	Image Decomposition in layers (100)	337
5.9	vSLAM Steps (100)	338
5.10	First keyframe	339
5.11	Keypoints in the Keyframe	339
5.12	Matching the keypoint from the RGB to the Depth	339
5.13	Tracking Phase	340
5.14	Graph of covisibility	341
5.15	Environment Map	341
5.16	Mobile, fixed, and world reference frames represented in the experimental setup	343
5.17	Common world reference system with visual markers for spatial alignment	344
5.18	Tracking performance for a straight path	346
5.19	Tracking performance for a turn path	347
5.20	vSLAM performance for a straight path	349
5.21	vSLAM performance for a straight path	350
5.22	Illustrative representation of the ROS network architecture, showing how nodes interact through topics and services to facilitate data exchange and communication (103)	353
5.23	Single agent scheme for the Scout Mini	355
5.24	Diagram illustrating the overall functioning of the agent in the environment	356
5.25	Raw point cloud data captured by the LiDAR, representing the scanned environment in three dimensions	357
5.26	Navigation transformation frames chain	358
5.27	Direct LiDAR Odometry (DLO) map displaying the environment with time-variant pose estimation	359
5.28	Occupancy grid map representing the position of the robot and detected obstacles within the environment	359
5.29	Purpose of AMCL (108)	360
5.30	Example of AMCL convergence: (L) Low convergence and (R) High con- vergence	361
5.31	IP table of the multi-agent network	364
5.32	Diagram illustrating the overall functioning of the multi-agent in the environment	364
5.33	Turtlebot Display	367
5.34	Simple Discovery configuration network (109)	369
5.35	ROS 2 topic list for multiple robots	370
5.36	Output of the /cmd_vel topic in ROS 2, displaying linear and angular velocity commands for controlling the movement of the robot	370
5.37	Output of the /scan topic in ROS2	371

5.38	Output of the /odom topic in ROS2	371
5.39	Graphical interface of Gazebo	373
5.40	Rviz2	374
5.41	Keyboard teleoperation interface	375
5.42	Rqt graph	375
5.43	Bluetooth joystick utilized for teleoperation	375
5.44	Leader and follower setup with Turtlebot4 robots	376
5.45	ArUco, id = 23, DICT_6X6_250 (110)	377
5.46	Prospective removal (110)	377
5.47	Marker Cells (110)	377
5.48	Example of Aruco Poses	378
5.49	Representation of the transformation from the RealSense frame to the base link frame in the robotic system	379
5.50	Transformation matrix detailing the translation and rotation between frames	379
5.51	Controlled Parameters: d and α	380
5.52	Control loop diagram showcasing the calculation of feedback distance d_{feedback} and the voltage adjustment for maintaining desired robot distance	381
5.53	Chasing Test: Gazebo Simulation	383
5.54	Chasing Test: Simulation Trajectories	383
5.55	Chasing Test: Error	384
5.56	Chasing Test: Test sequence	384
5.57	Chasing Test: Trajectories in Test	385
5.58	Chasing Test: Error in Test	385
5.59	Chasing Test: Percentage Error Gazebo	386
5.60	Chasing Test: Percentage Error in Test	386
5.61	Law of motion for Turtlebot leader	388
5.62	Law of Motion Test: Error in Simulation	389
5.63	Law of Motion Test: Percentage Error in Simulation	389
5.64	Law of Motion Test: Velocity along x robot axis	390
5.65	Law of Motion Test: Zoom in for velocity along x robot axis V_x trend . .	390
5.66	Law of Motion: V_x under PI Control	391
5.67	Law of Motion: Error under PI Control	391
5.68	Law of Motion: Percentage Error under PI Control	392
5.69	Law of Motion: Test under PI	392
5.70	Law of Motion: V_x under PI test	393
5.71	Law of Motion: Error in PI Test	393
5.72	Law of Motion: Percentage Error in Test	394
5.73	Law of Motion: Difference of the velocities in Simulation	394
5.74	Law of Motion: Difference of the velocities in Test	395
5.75	Law of Motion: Test with SMC	396
5.76	Law of Motion: Error in SMC test	397
5.77	Law of motion: Difference of speeds under PI	397
5.78	Law of motion: Difference of speeds under SMC	398

5.79	Obstacle Avoidance: Test sequence	399
5.80	Obstacle Avoidance Test: Trajectories	400
5.81	Obstacle Avoidance Test: Linear Velocities	401
5.82	Obstacle Avoidance Test: Angular Velocities	401
5.83	ROS2 architecture of the system	404
5.84	System Plant	406
5.85	Frequency Response of the filters $G_d(s)$ and $G_u(s)$	409
5.86	Frequency Response for the Controllers	411
5.87	Response for different controllers in the presence of system disturbances and measurement noise.	412
5.88	Closed Loop Linear System Frequency Response	412
5.89	X-Y reference trajectory and robot position	413
5.90	Full-order wheel speeds	414
5.91	Full-order controller position errors and control actions (of the off-center point) generated by the feedback action	415
5.92	FFT of the X-Position Error	415
A.1	Chassis of the Small-size prototype (118)	420
A.2	DC motor with gearbox	420
A.3	Data sheet of the DC Motor	421
A.4	Arduino Board (119)	422
A.5	Adafruit Motor Shield V2 Motor-Shield (120)	422
A.6	Reference System of the IMU (118)	423
A.7	MEMS Accelerometer Structure	424
A.8	MEMS gyroscope structure: (a) single mass configuration, (b) tuning fork configuration	425
A.9	MPU6050 Sensor	425
A.10	Micro-SD module	426
A.11	Robotic arm (121)	428
A.12	HITECH HS-475 motor specifications (122)	429
A.13	HSR-5990TG motor specifications (122)	430
A.14	HS-85MG motor specifications (122)	431
A.15	HITECH HS-81 motor specifications (122)	432
A.16	Ni-MH 6V 3500mAh battery	432
A.17	Representation of the material extrusion process in additive manufacturing	435
A.18	Raise3D Pro 2 Plus (125)	436
A.19	Raise3D Pro 2 Plus default printing modes (125)	436
A.20	Raise3D Pro 2 Plus default printing modes: top-view (125)	437
A.21	Raise3D Pro 2 Plus default printing modes: bottom-view (125)	437
A.22	DBLS-01S Brushless DC Motor Driver (126)	439
A.23	Hoverboard	440
A.24	DC motor-wheel subsystem used for propulsion in the Full-Scale Rover prototype	440

A.25 Schematic representation of an encoder (127)	441
A.26 DC motor characteristics with the effect of the current limit imposed by the driver on torque and angular speed	441
A.27 Components of a lithium-ion battery: anode, cathode, electrolyte, and separator (128)	442
A.28 Lithium-Ion Battery	443
A.29 Discharge curve for a Li-ion battery (129)	444
A.30 Charging curve for a Lead acid (red) and a Li-ion (blue) battery (129) . .	445
A.31 Dobot Magician (130)	446
A.32 Intel Realsense D415 camera (131)	447
A.33 Stereo Active technology in the Intel Realsense D415 (131)	447
A.34 D400 Modules in both cameras (131)	448
A.35 D435 Model (131)	449
A.36 Scout Mini AgileX (132)	450
A.37 Specifications of the Scout Mini AgileX robot (133)	451
A.38 Velodyne LiDAR Puck VLP-16 (134)	452
A.39 NVIDIA Jetson AGX Xavier (135)	452
A.40 Turtlebot 4 (109)	454
A.41 iRobot® Create® 3 (109)	454
A.42 Raspberry Pi 4B (136)	455
A.43 User Interface PCBA (109)	456
A.44 RPLidar A1M8	456
A.45 Triangulation for the LiDAR	457

List of Tables

1.1	Time intervals and corresponding signal values for $S(t)$	56
2.1	Geometric coordinates of key points in the rocker-bogie mechanism . . .	72
2.2	Geometric parameters of the small prototype rover	74
2.3	Rocker-bogie configuration parameters with wheel A on an obstacle step	78
2.4	Rocker-Bogie Configuration with wheel A and wheel B on obstacle step	79
2.5	PWM values, corresponding voltages, and measured no-load speeds of the DC motor	93
2.6	PWM values and corresponding measurements of weight and stall torque for the DC motor	94
2.7	Suspension angle configuration with the robotic arm as Wheel B approaches the obstacle	125
2.8	Suspension angle configuration with the robotic arm as Wheel C approaches the obstacle	127
2.9	Suspension angle configuration with the robotic arm in a tipping condition	128
2.10	Masses of the link.	136
2.11	Distances of the centers of mass relative to the reference axis.	138
2.12	Distances of the centers of mass relative to the reference axis for anti-overturning configuration.	139
2.13	Simulation time required for varying numbers of circumferential points, illustrating the computational cost as the number of points increases . .	167
2.14	Simulation time required for different numbers of circumferential lines .	169
2.15	Simulation time required for different angular distances θ	170
3.1	Comparison of values between the left and right banks under PI control .	262
3.2	Comparison of values between the left and right banks under Sliding Mode Control	266
5.1	Chasing Test: Gazebo vs Experimentation	387
5.2	Law of Motion Test: Gazebo vs Experimentation	393
5.3	Law of Motion Test: PID vs SMC	397
5.4	Infinity norm values ($\ G\ _\infty$) for the designed controllers	410
5.5	Designed controllers and their representations	411

5.6	RMS error values for the different controllers	414
A.1	DBLS-01S Datasheet	439

Bibliography

- [1] Gregory Dudek and Michael Jenkin. *Computational principles of mobile robotics*. Cambridge university press, 2024.
- [2] Junyan Hu, Parijat Bhowmick, Inmo Jang, Farshad Arvin, and Alexander Lanzon. A decentralized cluster formation containment framework for multirobot systems. *IEEE Transactions on Robotics*, 37(6):1936–1955, 2021.
- [3] Junyan Hu, Ali Emre Turgut, Barry Lennox, and Farshad Arvin. Robust formation coordination of robot swarms with nonlinear dynamics and unknown disturbances: Design and experiments. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 69(1):114–118, 2021.
- [4] Magda Cazacu. A short overview on terrestrial hybrid locomotion mobile robots.
- [5] S Klinkner, CG-Y Lee, C Wagner, W Hlawatsch, A-M Schreyer, and H-P Röser. Destination moon and beyond for the microrover nanokhod. 2007.
- [6] Boston Dynamics. Spot - il robot mobile agile per ispezioni dinamiche e industriali.
- [7] Kevin C Galloway, Galen Clark Haynes, B Deniz Ilhan, Aaron M Johnson, Ryan Knopf, Goran Lynch, Benjamin Plotnick, Mackenzie White, and Daniel E Koditschek. X-rhex: A highly mobile hexapedal robot for sensorimotor tasks. *University of Pennsylvania, Tech. Rep*, 2010.
- [8] J Matijevic. Sojourner: The mars pathfinder microrover flight experiment. *Space Technology*, 3(17):143–149, 1997.
- [9] Thomas Thueer, Pierre Lamon, Ambroise Krebs, and Roland Siegwart. *CRAB-Exploration rover with advanced obstacle negotiation capabilities*. Swiss Federal Institute of Technology (ETHZ), Autonomous Systems Lab, 2006.
- [10] Donald B Bickler. Articulated suspension system. 1989.
- [11] J Hrbáček, T Ripel, and J Krejsa. Ackermann mobile robot chassis with independent rear wheel drives. In *Proceedings of 14th International Power Electronics and Motion Control Conference EPE-PEMC 2010*, pages T5–46. IEEE, 2010.

-
- [12] Benjamin Shamah. Experimental comparison of skid steering vs. explicit steering for a wheeled mobile robot. *MSc Thesis, The Robotics Institute of Carnegie Mellon University*, 1999.
 - [13] Olaf Diegel, Aparna Badve, Glen Bright, Johan Potgieter, and Sylvester Tlale. Improved mecanum wheel design for omni-directional robots. In *Proc. 2002 Australasian conference on robotics and automation, Auckland*, pages 117–121, 2002.
 - [14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
 - [15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.
 - [16] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
 - [17] André Mewes. *Projector-based Augmented Reality and Touchless Interaction to Support MRI-Guided Interventions*. PhD thesis, Otto von Guericke University Magdeburg - OVGU, 2019.
 - [18] P Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer, 2011.
 - [19] MathWorks. Camera calibration.
 - [20] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.
 - [21] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.
 - [22] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*, volume 1, pages 886–893. Ieee, 2005.
 - [23] Ross Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.
 - [24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.
 - [25] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

- [26] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.
- [27] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [28] Yang Cong, Ronghan Chen, Bingtao Ma, Hongsen Liu, Dongdong Hou, and Chenguang Yang. A comprehensive study of 3-d vision-based robot manipulation. *IEEE Transactions on Cybernetics*, 53(3):1682–1698, 2023.
- [29] Anna Karwasz and Igor Wawrzynowicz. Using visions systems and manipulators in industry 4.0. In *International Scientific-Technical Conference MANUFACTURING*, pages 165–177. Springer, 2024.
- [30] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.
- [31] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of field robotics*, 37(3):362–386, 2020.
- [32] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.
- [33] Andreas Kamilaris and Francesc X Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, 147:70–90, 2018.
- [34] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- [35] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.
- [36] Jim Mainprice, Rafi Hayne, and Dmitry Berenson. Goal set inverse optimal control and iterative replanning for predicting human reaching motions in shared workspaces. *IEEE Transactions on Robotics*, 32(4):897–908, 2016.
- [37] Ronald T Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments/MIT press*, 1997.

-
- [38] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011.
- [39] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.
- [40] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.
- [41] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.
- [42] Robotnik Automation S.L. Rb-vogui: Autonomous mobile robot for indoor and outdoor.
- [43] Khashayar Asadi, Rahul Jain, Ziqian Qin, Mingda Sun, Mojtaba Noghabaei, Jeremy Cole, Kevin Han, and Edgar Lobaton. Vision-based obstacle removal system for autonomous ground vehicles using a robotic arm. In *ASCE International Conference on Computing in Civil Engineering 2019*, pages 328–335. American Society of Civil Engineers Reston, VA, 2019.
- [44] Liuping Wang. Basics of pid control. 2020.
- [45] B. B. Musmade, Y. P. Patil, R. K. Munje, and B. M. Patre. Design of sliding mode control scheme for nonlinear processes for improved performance. In *2015 International Conference on Industrial Instrumentation and Control (ICIC)*, pages 992–996, 2015.
- [46] KS Holkar and LM Waghmare. Sliding mode control with predictive pid sliding surface for improved performance. *International Journal of Computer Applications*, 78(4), 2013.
- [47] Emre Hasan Dursun and Akif Durdu. Speed control of a dc motor with variable load using sliding mode control. *International Journal of Computer and Electrical Engineering*, 8(3):219, 2016.
- [48] Hari Maghfiroh, Augustinus Sujono, Chico Hermanu Brillianto Apribowo, et al. Basic tutorial on sliding mode control in speed control of dc-motor. *Journal of Electrical, Electronic, Information, and Communication Technology*, 2(1):1–4, 2020.
- [49] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

- [50] Waymo. Meet our newest self-driving vehicle: the all-electric jaguar i-pace.
- [51] Dürr AG. Profleet - automated guided vehicles for all applications.
- [52] Robin R Murphy. *Disaster robotics*. MIT press, 2017.
- [53] Chao Cheng, Jun Fu, Hang Su, and Luquan Ren. Recent advancements in agriculture robots: Benefits and challenges. *Machines*, 11(1):48, 2023.
- [54] Matthew Dunbabin and Lino Marques. Robots for environmental monitoring: Significant advancements and applications. *IEEE Robotics & Automation Magazine*, 19(1):24–39, 2012.
- [55] Meijer. Meijer is first retailer to unveil beach and water cleaning drones in the great lakes in partnership with the council of the great lakes region.
- [56] Charles R Weisbin and Guillermo Rodriguez. Nasa robotics research for planetary surface exploration. *IEEE Robotics & Automation Magazine*, 7(4):25–34, 2000.
- [57] NASA. Mars 2020: Perseverance rover.
- [58] MWM Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on robotics and automation*, 17(3):229–241, 2001.
- [59] Y Wang and Y Zhao. Multiple ships cooperative navigation and collision avoidance using multi-agent reinforcement learning with communication. *arXiv preprint arXiv:2410.21290*, 2024.
- [60] David Wilkie, Jur Van Den Berg, and Dinesh Manocha. Generalized velocity obstacles. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5573–5578. IEEE, 2009.
- [61] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE international conference on robotics and automation*, pages 1928–1935. Ieee, 2008.
- [62] Aakriti Agrawal, Senthil Hariharan, Amrit Singh Bedi, and Dinesh Manocha. Dc-mrta: Decentralized multi-robot task allocation and navigation in complex environments. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11711–11718. IEEE, 2022.
- [63] Yue Jin, Yaodong Zhang, Jian Yuan, and Xudong Zhang. Efficient multi-agent cooperative navigation in unknown environments with interlaced deep reinforcement learning. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2897–2901, 2019.

-
- [64] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7:1–41, 2013.
- [65] Kota Kondo, Claudius T Tewari, Mason B Peterson, Annika Thomas, Jouko Kinnari, Andrea Tagliabue, and Jonathan P How. Puma: Fully decentralized uncertainty-aware multiagent trajectory planner with real-time image segmentation-based frame alignment. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13961–13967. IEEE, 2024.
- [66] A Kemurdjian, V Gromov, V Mishkinyuk, V Kucherenko, and P Sologub. Small marsokhod configuration. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 165–168. IEEE, 1992.
- [67] Paul G Backes, Kam S Tso, and Gregory K Tharp. Mars pathfinder mission internet-based operations using wits. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 1, pages 284–291. IEEE, 1998.
- [68] S Aswath, VJ Unnikrishnan, T Kalathil Sreekuttan, Hariprasad Abin Simon, P Mathew Deepu, Maya Menon, Praveen Basil, Ragesh Ramachandran, Abhay Sengar, Arjun Balakrishnan, et al. Design and development of an intelligent rover for mars exploration. In *The 18th Annual International Mars Society Convention*, 2016.
- [69] The Searial Cleaners. Bebot, the beach cleaning robot.
- [70] MathWorks. Modeling contact force between two solids.
- [71] Vincent Acary and Bernard Brogliato. *Numerical methods for nonsmooth dynamical systems: applications in mechanics and electronics*. Springer Science & Business Media, 2008.
- [72] Peter Wriggers and Tod A Laursen. *Computational contact mechanics*, volume 2. Springer, 2006.
- [73] Friedrich Pfeiffer and Christoph Glocker. *Multibody dynamics with unilateral contacts*, volume 421. Springer Science & Business Media, 2000.
- [74] Orientalmotor. *Technical Manual Series: Brushed DC Motor Structure and Principles*.
- [75] Luca Zaccarian. Dc motors: dynamic model and control techniques. *Lecture notes., roma, italy*, 2012.
- [76] Henri J Nussbaumer and Henri J Nussbaumer. *The fast Fourier transform*. Springer, 1982.

- [77] Samuele Crivellaro. *Protocollo di comunicazione I²C*.
- [78] ScanImage®. I2c data recording.
- [79] FlexiHub. Can bus - cos'è e come funziona.
- [80] Agilex Robotics. Scout mini - agilex robotics.
- [81] Chiara Cosenza, Vincenzo Niola, Stefano Pagano, and Sergio Savino. Theoretical study on a modified rocker-bogie suspension for robotic rovers. *Robotica*, 41(10):2915–2940, 2023.
- [82] JW Umland, GR Gillis-Smith, CW Buck, and HJ Eisen. Mechanical design of the mars pathfinder mission. 1997.
- [83] Pierangelo Malfi. Studio, modellazione multibody e analisi sperimentale sulla pianificazione dei moti di un robot mobile a sei ruote basato su sistema di sospensione rocker-bogie. Master Thesis.
- [84] Herve Hacot. *Analysis and traction control of a rocker-bogie planetary rover*. PhD thesis, Massachusetts Institute of Technology, 1998.
- [85] Moshe P Mann and Zvi Shiller. Dynamic stability of a rocker bogie vehicle: Longitudinal motion. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 861–866. IEEE, 2005.
- [86] Armando Nicolella, Vincenzo Niola, Stefano Pagano, Sergio Savino, and Mario Spirto. An overview on the kinematic analysis of the rocker-bogie suspension for six wheeled rovers approaching an obstacle. In *The International Conference of IFToMM ITALY*, pages 86–93. Springer, 2022.
- [87] Maria Chiara Gervasio. Studio cinematico e dinamico di un robot mobile dotato di braccio robotico nel superamento degli ostacoli. Master Thesis.
- [88] Jaka. Jaka zu 5s collaborative robot.
- [89] Luc Avérous. Polylactic acid: synthesis, properties and applications. In *Monomers, polymers and composites from renewable resources*, pages 433–450. Elsevier, 2008.
- [90] M Kolybaba, LG Tabil, S Panigrahi, WJ Crerar, T Powell, and B Wang. Biodegradable polymers: past, present, and future. In *ASABE/CSBE north central intersectional meeting*, page 1. American Society of Agricultural and Biological Engineers, 2006.
- [91] ME Adams, DJ Buckley, and RE Colborn. *Acrylonitrile-butadiene-styrene polymers*, volume 70. iSmithers Rapra Publishing, 1993.

-
- [92] Zixiang Weng, Jianlei Wang, T Senthil, and Lixin Wu. Mechanical and thermal properties of abs/montmorillonite nanocomposites for fused deposition modeling 3d printing. *Materials & design*, 102:276–283, 2016.
 - [93] Seth Hutchinson, Gregory D Hager, and Peter I Corke. A tutorial on visual servo control. *IEEE transactions on robotics and automation*, 12(5):651–670, 1996.
 - [94] Chiara Cosenza, Pierangelo Malfi, Armando Nicolella, Vincenzo Niola, and Sergio Savino. Hand-eye calibration using invariant calibrator placed to a robotic arm. In *The International Conference of IFToMM ITALY*, pages 343–350. Springer, 2024.
 - [95] Ikenna Enebuse, Mathias Foo, Babul Salam Ksm Kader Ibrahim, Hafiz Ahmed, Fhon Supmak, and Odongo Steven Eyobu. A comparative review of hand-eye calibration techniques for vision guided robots. *IEEE Access*, 9:113143–113155, 2021.
 - [96] Velibor Đalić, Vedran Jovanović, and Petar Marić. Submillimeter-accurate marker-less hand-eye calibration based on a robot’s flange features. *Sensors*, 24(4):1071, 2024.
 - [97] Leihui Li, Xingyu Yang, Riwei Wang, and Xuping Zhang. Automatic robot hand-eye calibration enabled by learning-based 3d vision. *Journal of Intelligent & Robotic Systems*, 110(3):130, 2024.
 - [98] Yu Guo, Yuanyan Xie, Yue Chen, Xiaojuan Ban, Balqies Sadoun, and Mohammad S Obaidat. An efficient object navigation strategy for mobile robots based on semantic information. *Electronics*, 11(7):1136, 2022.
 - [99] Yue Wu. Research on feature point extraction and matching machine learning method based on light field imaging. *Neural Computing and Applications*, 31(12):8157–8169, 2019.
 - [100] MathWorks. Visual slam with rgb-d camera.
 - [101] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.
 - [102] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 573–580. IEEE, 2012.
 - [103] MathWorks Robotics Team. Mathworks robotics github repository.
 - [104] ROS Community. Ros wiki.
 - [105] Kenny Chen, Brett T Lopez, Ali-akbar Agha-mohammadi, and Ankur Mehta. Direct lidar odometry: Fast localization with dense point clouds. *IEEE Robotics and Automation Letters*, 7(2):2000–2007, 2022.

- [106] Kevin Murphy and Stuart Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Sequential Monte Carlo methods in practice*, pages 499–515. Springer, 2001.
- [107] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 2432–2437. IEEE, 2005.
- [108] ROS. Amcl.
- [109] Clearpath Robotics. Turtlebot 4 - mobile robot platform.
- [110] OpenCV. Detection of aruco markers using opencv.
- [111] Niraj Kumar Goswami and Prabin Kumar Padhy. Sliding mode controller design for trajectory tracking of a non-holonomic mobile robot with disturbance. *Computers & Electrical Engineering*, 72:307–323, 2018.
- [112] DS Flamm and HONG Yang. $H/\sup$ infinity/-optimal mixed sensitivity for general distributed plants. In *29th IEEE Conference on Decision and Control*, pages 134–139. IEEE, 1990.
- [113] Ahmad Taher Azar, Fernando E. Serrano, Ibrahim A. Hameed, Nashwa Ahmad Kamal, and Sundarapandian Vaidyanathan. Robust h-infinity decentralized control for industrial cooperative robots. In Aboul Ella Hassanien, Khaled Shaalan, and Mohamed Fahmy Tolba, editors, *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2019*, pages 254–265, Cham, 2020. Springer International Publishing.
- [114] Amir Salimi Lafmejani, Hamed Farivarnejad, and Spring Berman. H-optimal tracking controller for three-wheeled omnidirectional mobile robots with uncertain dynamics. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7587–7594, 2020.
- [115] Jose Ceron Neto and Roberto Santos Inoue. Deep adaptive nonlinear h control for wheeled mobile robots. In *2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE)*, pages 385–390, 2023.
- [116] Robin Amsters and Peter Slaets. Turtlebot 3 as a robotics education platform. In *Robotics in Education: Current Research and Innovations 10*, pages 170–181. Springer, 2020.
- [117] Ricardo Núñez, Armando Nicolella, Leopoldo Armesto, Antonio González, and Antonio Sala. Diseño de controladores h-infinito para el seguimiento de trayectorias con robots móviles con ruedas. *Jornadas de Automática*, (45), 2024.

-
- [118] Servocity. Bogie runt rover™.
- [119] Arduino. Arduino board.
- [120] Adafruit Industries. Adafruit.
- [121] Lynxmotion. Lynx 6 arm.
- [122] Hitec. Hitech motors.
- [123] Jon Excell and Stuart Nathan. The rise of additive manufacturing. *The engineer*, 24, 2010.
- [124] Hugo KS Lam, Li Ding, TCE Cheng, and Honggeng Zhou. The impact of 3d printing implementation on stock returns: A contingent dynamic capabilities perspective. *International Journal of Operations & Production Management*, 39(6/7/8):935–961, 2019.
- [125] Raise3D. Raise3d pro plus.
- [126] StepperOnline. *DBLS-01S*.
- [127] Robotics Tomorrow. What is an encoder?
- [128] Chimicamo. Batterie per cellulari li-ion.
- [129] Deepak Patel, Vishal Singhal, Aditya Kumar, Kanishk Bansal, Awadhesh Kumar Maurya, Anuj Goyal, and Anurag Dixit. Dc fast charging technique for automated guided vehicle. In *Advances in Interdisciplinary Engineering: Select Proceedings of FLAME 2020*, pages 303–312. Springer, 2021.
- [130] Dobot. Dobot magician.
- [131] Intel. Intel realsense.
- [132] Agile-X. Agile-x scout mini.
- [133] Trossen Robotics. Scout mini specifications - trossen robotics.
- [134] Velodyne. Velodyne lidar puck vlp-16.
- [135] Nvidia. Nvidia jetson agx xavier.
- [136] Raspberry Pi Foundation. Raspberry pi 4 model b specifications.



UNIONE EUROPEA
Fondo Sociale Europeo



*Ministero dell'Università
e della Ricerca*



PON
RICERCA
E INNOVAZIONE
2014 - 2020

REACT EU

La borsa di dottorato è stata cofinanziata con risorse del
Programma Operativo Nazionale Ricerca e Innovazione 2014-2020, risorse FSE REACT-EU
Azione IV.4 “Dottorati e contratti di ricerca su tematiche dell’innovazione”
e Azione IV.5 “Dottorati su tematiche Green”