UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Dipartimento Ingegneria Industriale e Scienze Matematiche
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA
MECCANICA

Collaborative Object Transport Using Two Holonomic Mobile Robots

Trasporto collaborativo di oggetti con due robot mobili olonomi

Candidate:
Marco Fava

Advisor:
Prof. Luca Carbonari

Coadvisor:
Prof. Massimo Callegari
Prof. Annika Raatz

Academic Year 2021-2022

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Dipartimento Ingegneria Industriale e Scienze Matematiche
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA
MECCANICA

Collaborative Object Transport Using Two Holonomic Mobile Robots

Trasporto collaborativo di oggetti con due robot mobili olonomi

Candidate:
Marco Fava

Advisor:
Prof. Luca Carbonari

Coadvisor:
Prof. Massimo Callegari
Prof. Annika Raatz

Academic Year 2021-2022

Università Politecnica delle Marche
Dipartimento Ingegneria Industriale e Scienze Matematiche
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA MECCANICA
Via Brecce Bianche – 60131 Ancona (AN), Italy

Abstract

In recent times, the control of multiple mobile robots is becoming increasingly successful. Multi-robot structures can perform tasks that cannot be done by a single robot. Therefore, the study of the simultaneous motion of multiple robots is becoming more and more common.

Different kinds of robots that can be used for this purpose; in this thesis, two of them are studied, a classical non-holonomous type and another omnidirectional or holonomous one, that is capable of moving freely without necessarily change orientation. The project is inspired by an existing system capable of moving omni-directionally and supporting high loads. The robot is developed by the company KUKA and is called OmniMove. It is capable of transporting objects such as train wagons, aircraft parts, etc.

The study attempts to recreate a similar structure consisting of two holonomous robots connected with a platform. From the literature, the type of control used in structures already developed in academic institutions were studied.

Once the type of control to be implemented in our structure had been decided, we moved on to write a programme that could be implemented in ROS (Robot Operating System), a software capable of interact to real or virtual robots via our commands. The programme was written in Python and involves acquiring and sending data to control and monitor the robots. These will have to move in a synchronised manner, for whatever path is given to them.

The programme was then implemented with a closed-loop feedforeward controller that allows us to correct the robot's trajectory according to errors that can be measured during movement. Another method for measuring errors was studied and implemented, observing its advantages and drawbacks.

In addition to programming the movement of the two robots, a first prototype for connecting the robots was designed and realised. A platform that rests on three load-bearing spheres, one in the centre of each robot and one in the centre of the platform that supports it from the ground. These spheres allow the robots to deviate slightly from the trajectory described by the synchronism. If the platform's anchoring points were rigid, the robots' components could be ruined after only few applications. In order to support the load-bearing spheres and not move the platform under normal working conditions, soft material supports were created to allow the robots to move without repercussions and to support the platform. Several simulations were carried out both in the design area, to create and realise the soft components and to design the platform, and to control the navigation of the robots.

From the prototype, it was possible to observe the actual behaviour of the platform and deduce conclusions and future improvements applicable to the platform itself as well as to the entire system.

Sommario

Negli ultimi tempi il controllo di molteplici robot mobili sta riscontrando sempre più successo. Con strutture multi-robot si possano compiere compiti che non possono essere eseguiti da un singolo robot. Pertanto, lo studio della movimentazione simultanea di molteplici robot è sempre più frequente.

Diverse sono le tipologie di robot possono essere usati per a tale scopo. In questa tesi di laurea ne vengono studiati due, uno di tipo classico non olonomo e un altro omnirezionale o olonomo, capace cioè, di muoversi liberamente senza dover cambiare forzatamente orientamento. Il progetto si inspira ad un sistema già esistente capace di muoversi in maniera omnidirezionale e di supportare elevati carichi. Il robot in questione è sviluppato dall'azienda KUKA e si chiama OmniMove. È capace di trasportare oggetti come vagoni di treni, parti di aerei ecc.

Lo studio cerca di ricreare una struttura simile composta da due robot olonomi collegati con una piattaforma. Dalla letteratura sono state studiate le strutture già sviluppate in ambito accademico, concentrandosi sulla tipologia di controllo usata.

Una volta decisa la tipologia di controllo da implementare nella nostra struttura si è passati alla scrittura di un programma implementabile nel software ROS (Robot Operating System) in grado di far interagire i nostri comandi, scritti nei codici, con dei robot reali o virtuali. Il programma è stato scritto con Python e prevede l'acquisizione e l'invio di dati per controllare e monitorare i robot. Questi si dovranno muovere in maniera sincronizzata, per qualsiasi percorso fornitogli.

Si è poi implementato il programma con un controllore a ciclo chiuso in retroazione che ci permette di correggere la traiettoria del robot in base a degli errori misurabili nel corso della movimentazione. Un altro metodo di misurazione degli errori è stato studiato e implementato osservandone i pregi e i difetti.

Oltre alla programmazione della movimentazione dei due robot è stato progettato e realizzato un primo prototipo di connessione tra gli stessi. Una piattaforma che appoggia su tre sfere portanti, una al centro di ciascun robot e una al centro della piattaforma che tramite un supporto la sostiene dal terreno. Queste sfere consentono ai robot di potersi discostare leggermente dalla traiettoria descritta dal sincronismo. Se i punti di ancoraggio della piattaforma fossero stati rigidi la componentistica dei robot si sarebbe potuta rovinare dopo poche applicazioni. Per sostenere le sfere portanti e non far muovere la piattaforma nelle condizioni normali di lavoro, sono state create dei supporti di materiale morbido che consentono di far muovere i robot senza ripercussioni e, contemporaneamente, di supportare la piattaforma. Sono state compiute svariate simulazioni sia nell'ambito progettuale, per realizzare i supporti

morbidi e per progettare la piattaforma che per controllare la navigazione dei robot.

Dal prototipo è stato possibile osservare il comportamento reale della piattaforma, dedurne delle conclusioni e ipotizzare dei futuri miglioramenti, applicabili sia alla piattaforma stessa che all'intero sistema.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mobile robots and manipulators mounted on mobile platforms are becoming more and more useful in a large variety of indoor service applications: industrial, medical, military, and domestic. A partial list of environments where mobile platforms can assist ill, disabled and elderly people is: hospitals and clinics, dormitories for handicaped and elderly people, and sheldered workshops for physically disabled people. Moreover, they can be used in environments that are dangerous for humans, where extreme caution must be exercised because, for example, an unexploded bomb is being transported.

Three fundamental requirements of such robots are:

- maneuvrability,
- safety for people and machines,
- manipulability and dexterity.

Omnidirectional mobile robots are more and more used for this kind of strucure due to their extreme manoeuvrability which, returning to the case above, of the unexploded bomb, would make as few movements as possible in handling it.

Several omnidirectional platforms have been known to be realized by driving wheel with steering, universal wheels, spherical tires, or crawler mechanisms [7]. Groups of mobile robots can carry out tasks that a single robot could not perform. Most research in the area of formation control is focused on holonomic robots, since their superior mobility allows for better control and allows for the research on more sophisticated control techniques [8]. Holonomic robots in addiction can move freely in the space. Platforms where these kinds of robot are mounted can be useful for hazardous environment where space are limitated like wearhouse or chemical lab. After these considerations, it was decided to design and study a structure with two holonomic robots so that they could move in any direction without change orientation and could carry twice the payload of a single robot. Formation control was focused on two nonholonomic robots, because kinematics is more difficult, although that of the holonomic robot was not neglected.

The structure of this thesis is organised as follows. The following sections of this chapter explain the project goal and what has been studied in the literature from

the basics to understanding the workings of a holonomic robot. The second chapter describes the programming background required to replicate the work and the system used to control the robots. Furthermore, all simulation programmes are explained describing also the mathematics behind the structure's motion. Finally, the prototype built, the design programme used, and the tests developed are explained, along with their conclusions and future improvements.

1.1 Fundamentals Basics

The study of the state of the art allows us to understand the workings of the mechanisms to be studied and the solutions already present in the academic documentation. From these we can critically analyse the project by evaluating improvements or methodologies to be integrated into the prototype to be developed. The initial study also involved the entire robot programming part, in the Python language through the ROS Robot Operating System, then the formation control of platforms and the omnidirectional robots were studied.

1.1.1 Formation Control

Formation control of multiple autonomous mobile robots and vehicles has been studied extensively over the last decade for both theoretic research and practical applications. Various approaches and strategies have been proposed for the formation control of multiple robots. However, multi-robot coordination methods can be partitioned into three class approaches:

- virtual structure approach,
- behavioral approach,
- leader follower approach[9].

Each of them has several advantages and weaknesses.

Virtual Structure Approach

Consider a rigid body where the relative positions of points on the body are fixed. When the body moves in space with six degrees of freedom, points on the body continue to maintain their relative positions, although their positions in space change. If robots always maintain their relative positions with respect to a frame of reference, as the points of a rigid body do; they can be thought of as forming a rigid structure that does not physically exist. We call this a Virtual Structure. However, robots that forms a virtual structure with their position still have some degrees of freedoms, for example they can change their orientations. [10] The main advantage is that is easy to prescribe the behavior of the whole group and maintain the formation very well during the maneuvers. The main disadvantage of the current virtual structure

implementation is the centralization, which leads a single point of failure for the whole system.

Behavioral Approach

The approach of the controller in this case does not depend on the platform, so several effective controllers will exist. The idea we exploit is that tracking controllers can be designed independently of the coordination scheme, thus providing additional control power [11]. So several desired behaviors are prescribed for each robot such as obstacle avoidance, collision avoidance, goal seeking and formation keeping. The limitation of behavior-based approach is that it is difficult to analyze mathematically, therefore it is hard to guarantee a precise formation control.

Leader-Follower Approach

By leader following method, to prescribe a formation maneuver, we only need to specify the leader's motion and the desired relative positions between the leader and the followers. When the motion of the leader is known, the desired positions (desired distance and angle) of the followers relative to the leader can be achieved by local control law on each follower[11]. This approach is characterized by simplicity, reliability and no need for global knowledge and computation.

The last approach was the one used for this thesis. As there are only two robots to be moved simultaneously, it was considered the simplest and most effective. Many other researches used this method for formation control, and many implementations have been made, such as the fuzzy logic technique. The purpose of it is to keep the follower robot at a correct distance by monitoring it and giving values between 0 and 1 according with the distance, which will correct the robot's velocities[12].

1.2 Robots

The key feature of robots is their mechanical structure. Robots can be classified into *robot manipulators* that have a fixed base, and *mobile robots* where instead their base is movable. Robots are mechanical components linked together to perform tasks that help humans, in fact robotics comes from the term *robota* which in the Slavic language means executive work. These tasks, performed by robots, are more precise, reliable and safe than those performed by humans. At the same time, the control and programming of these robots can be difficult if flexibility is required or if a variety of tasks can be performed to achieve the same goal, so robots that can work with humans while respecting safety rules have become more popular in recent times. The capability to exert an action, is provided by an actuation system which animates the mechanical components of the robot. The concept of such a system refers to the context of motion control, dealing with servo motors, drives and transmissions.The capability for perception is entrusted to a sensory system which can acquire data on

the internal status of the mechanical system (proprioceptive sensors, such as position transducers) as well as on the external status of the environment (exteroceptive sensors, such as force sensors and cameras)[1].

Robot Manipulators

Robot manipulators consist of a series of rigid bodies (links) connected by joints; a manipulator consists of an arm that provides mobility, a wrist that provides dexterity, and the end effector that allows us to perform the tasks required by the robot.

The fundamental structure of robot manipulators can be an open kinematic structure or a closed kinematic structure. The former unlike the closed kinematic structure has only one sequence of links connecting two end parts of the chain. In the closed kinematic structure on the other hand, the sequence of links form a loop (see Figure 1.1), improving the performance of the structure: increasing load capacity, robustness and accuracy.

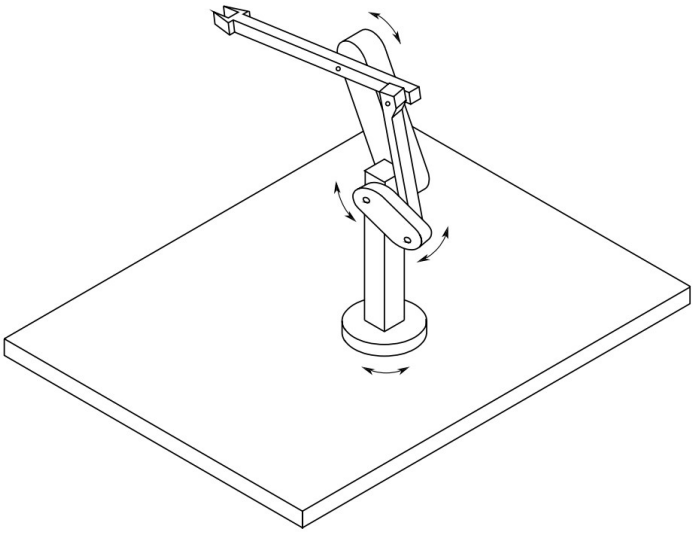


Figure 1.1: Example of robot with closed kinematic structure[1].

A manipulator's mobility is ensured by the presence of joints. The articulation between two consecutive links can be realized by means of either a prismatic or a revolute joint. In an open kinematic chain, each prismatic or revolute joint provides the structure with a single degree of freedom (DOF).

A prismatic joint creates a relative translational motion between the two links, whereas a revolute joint creates a relative rotational motion between the two links. Revolute joints are usually preferred to prismatic joints in view of their compactness and reliability. On the other hand, in a closed kinematic chain, the number of DOFs is less than the number of joints in view of the constraints imposed by the loop.

The degrees of freedom should be properly distributed along the mechanical structure in order to have a sufficient number of joints to execute a given task. In

the more general case of a task consisting of arbitrarily positioning and orienting an object in three-dimensional (3D) space, six DOF are required.

The kinematic structure allows us to position the robot wrist, the part that allows us to maneuver the end effector. Being, the wrist, the end part of the structure will have to be small, complicating the mechanical design.

The wrist can be called spherical if the axes of the 3 joints converge at the same point, with the advantage that the position and orientation of the end effector are decoupled, however, complicating the design of the component (see Figure 1.2)[1].

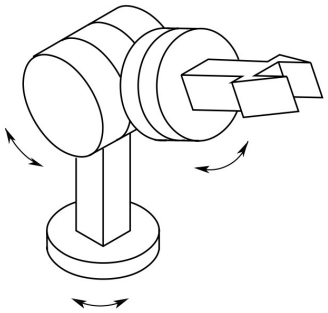


Figure 1.2: Spherical Wrist[1].

Mobile Robots

Mobile robots consist of a rigid structure equipped with a locomotion system that makes it capable of moving freely in space. This locomotion system can be moved in several ways:

- *Wheeled* which typically consist of a rigid body (Base or chasiss) and a wheel system that confer movement with respect to the ground. These will be the only types of robots that will have an in-depth discussion in this paper;
- *Legged* where the movement is inspired by living organisms, composed of multi rigid bodies interconnected with prismatic joints or more often with revolute joints;
- *Tracked locomotion* which allow movement in any kind of solid terrain;
- *Undulatory locomotion* that move with a snake-like wave motion, and can be used in narrow aquatic environments such as oil or gas installations (see Figure 1.3) [2].

We can classify robots into *holonomic* and *non-holonomic*.

This distinction is related to the way the robot moves, in other words the ratio between the number of controllable joints and the total number of degrees of freedom.

Figure 1.3: Snake Robot example[2].

If the number of controllable degrees is equal to the total number of degrees of freedom, the robot is called *holonomic*. It can move freely in any direction without forcibly changing orientation. A good example of this type of robot are those that have omnidirectional wheels or steerable drive wheels.

If the controllable degrees of freedom are less than the total degrees of freedom, the robot is known as *non-holonomic* drive. A car, for example, has three degrees of freedom, but we can only control the acceleration (forwards or backwards) and the steering angle of the steering wheel. In this way, we slip in curves, so we do not control everything, but rely on the friction between the wheels and the ground.

1.2.1 Conventional Wheels

Although the reliability, robustness and speed of robotic systems with four or more steered drive wheels has been validated and improved for years, generally, to avoid all the problems related to path traking, and steering flexibility [13], robots have drive systems with two fixed drive wheels, driven by electric motors, with usually caster wheels serving as stability points. An example of robot that use this configuration is the MIR 200, which was used in this study, in a virtual environment. The non-steered wheels, on the other hand, can be divided into three categories[14]:

- the *fixed wheels* which are rigidly mounted on the robot chassis and have a constant orientation over time;
- the *steerable wheels* which have two axes of rotation, the first one equal to that of the fixed wheels, while the second one, placed vertically, passing through the centre of the wheel, allows changing orientation with respect to the chassis;
- the *caster wheels* which also have two axes of rotation, but the vertical one does not pass through the wheel centre but is offset by a certain constant offset. This feature makes it possible to quickly change direction automatically by aligning the wheel to the chassis. This type of wheel is very often used to create a static balancing point without affecting the mobility of the mobile base.

An illustration of the previous wheels can be seen in Figure 1.4.

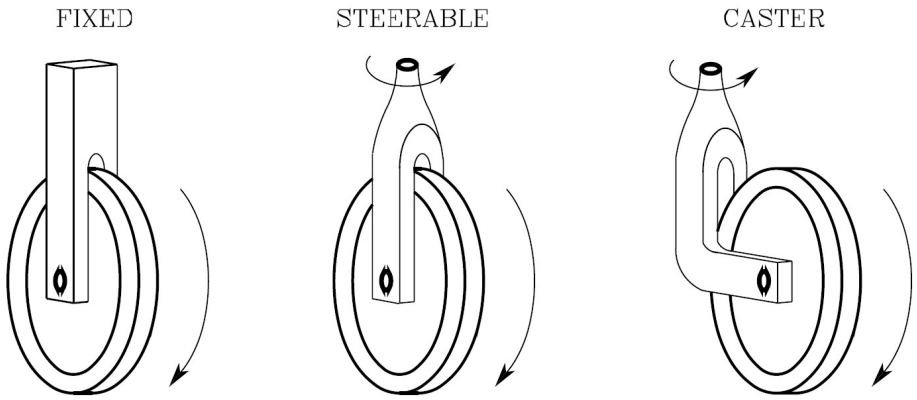


Figure 1.4: Conventional Wheels[1].

1.2.2 Conventional Robots - MIR 200

In this section the conventional robot used in the simulation is explained.

MIR 200 is a rectangular mobile robot 890 mm long and 580 mm wide (see Figure 1.5). This robot is equipped with 2 fixed stering wheel placed in its center, and four caster wheels on the corners. These allow all the movement of a non holonomic robot, in fact, is not possible going in the y direction without first turning. MIR 200 is equipped with two sick s300 laser scanners on opposing corners of the robot, giving it a 360-degrees detection radius without any blind spots[3]. One sick s300 laser scanner has a limitated range of detection (270-degrees), 90 degrees of blind spots, the exactly range that we do not need because we will scan the internal part of the robot. This type of robot is totally defined in a simulated environment, his model and all the funcionalities it has are available in Gazebo.

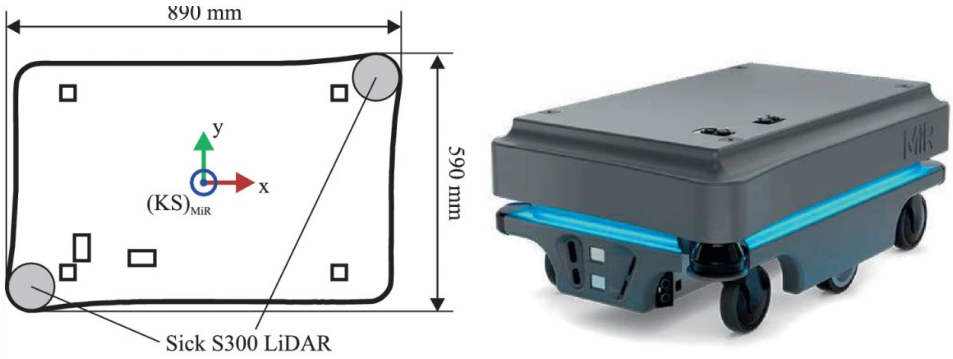


Figure 1.5: MIR200: schematic and manufacturer image [3].

1.2.3 Omnidirectional Wheels

Omnidirectional traction systems apply the rotational force of each individual wheel in one direction similarly to conventional wheels, with the difference, however, that they can slide freely in a lateral direction. The main advantage of using omnidirectional systems is to decouple the rotation from the translational motion, although this affects the speed of the operation, which will not be the fastest.

Mechanum wheels also called Ilon wheels or Swedish wheels are the most common wheels used in omnidirectional handling.

In a conventional wheel, the friction between the wheel surface and the ground generates the forces and torques that move the vehicle. However, in the case of the mecanum wheel, they are generated by the friction with the rollers of the wheel (see figure 1.6). The rollers, attached with a 45° angle, allow the wheel to generate a twisting force that makes the vehicle to move in 3 DOF (farward-backwards, right-left, rotation around the vertical axis).

The 45° angle and the shape of the rollers complicate the design of the wheels but provide a smoother contact with the surface and increase the load capacity supported by the single wheel itself [14].

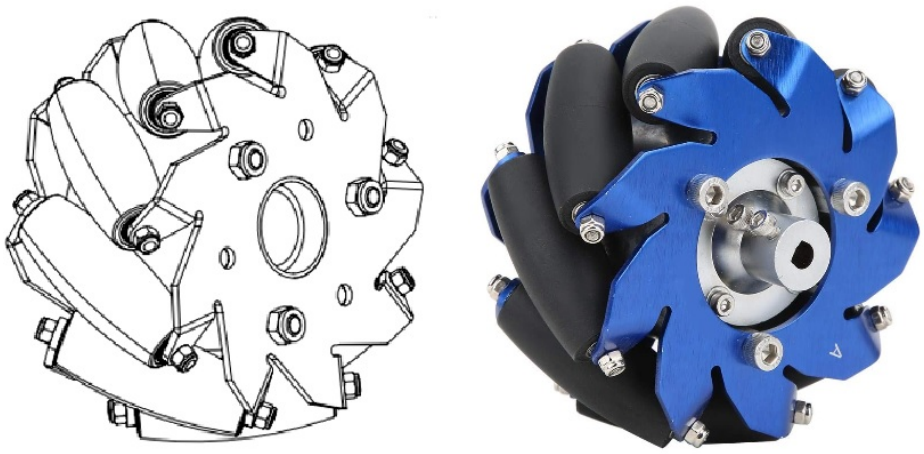
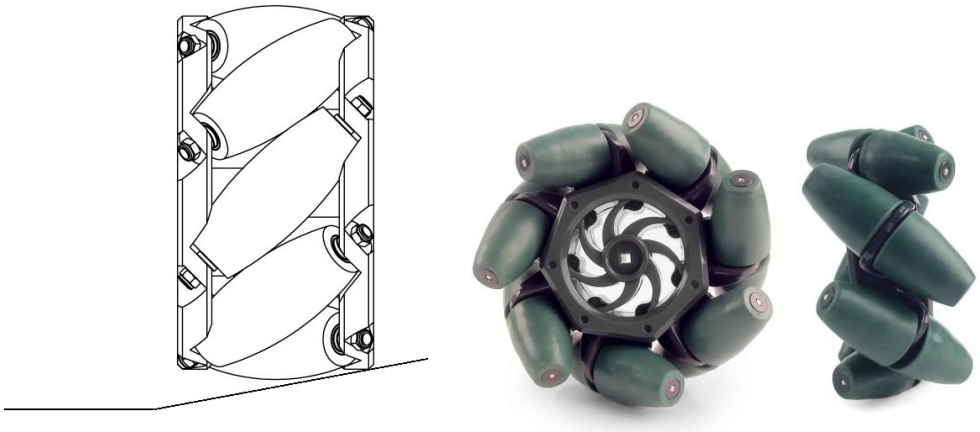


Figure 1.6: Omni Wheel Example [3]

There are different types of wheels, the most common, and the ones used in the robots available in the laboratory, are the traditional ones with the roller support on the side. They allow us to have an higher load that can be supported, but with the drawback of not being able to move on inclined surfaces. In these surfaces, in fact, the roller support could touch the surface, as we can see in Figure1.7a, compromising the omni-directional functionality, that is instead guaranteed even in these types of surfaces with Swedish wheels with centrally mounted rotating rollers (see Figure 1.7b) [4].

Recapitulating the Mecanum wheels have the following advantages: compact design, high load capacity, easy control, low speed and low pushing force when

moving diagonally. As disadvantages, we can mention: discontinuous contact, high sensitivity to locomotion surface irregularities, complex design [15].



(a) Traditional Mecanum wheel on inclined surface.

(b) Mecanum wheel with centrally mounted rollers.

Figure 1.7: Traditional and Centrally mounted rollers Mecanum wheels[4].

1.2.4 Omnidirectional Robot

Omnidirectional movement can be achieved by mounting four Mechanum wheels in the corners of a rectangular structure. We can see how to perform eight main movements without changing the orientation of the robot in the Figure 1.8.

Each wheel must be moved with the same angular speed at the same time during the operation. By changing the velocities of the diagonal wheels we achieved a motion between 0° to 360°. For example the forward motion is achieved when all four wheels are driven in the same direction (see Figure 1.8a).

For the displacement in sideways, diagonal pairs of wheels are driven in opposite directions (see Figures1.8e,f,g,h)[16].

AgileX- Scout mini rover

The omnidirectional robot available in the lab of the MATCH department is the one developed by AgileX Robotics, a Chinese company founded in 2016.

Scout Mini Rover is the smaller version of the Scout 2.0 robot. It is a robot with high manoeuvrability, speed, and considerable loading capacity. The robot can be equipped with omnidirectional wheels or not (see Figure 1.9).

In our case, the robot was equipped with Swedish wheels each with differential traction. In the specifications we can see that the maximum capacity of the robot is 20 kg for each omnidirectional wheel, so the maximum additional load that can be supported, subtracting the weight of the robot itself is approximately 50 kg [17].

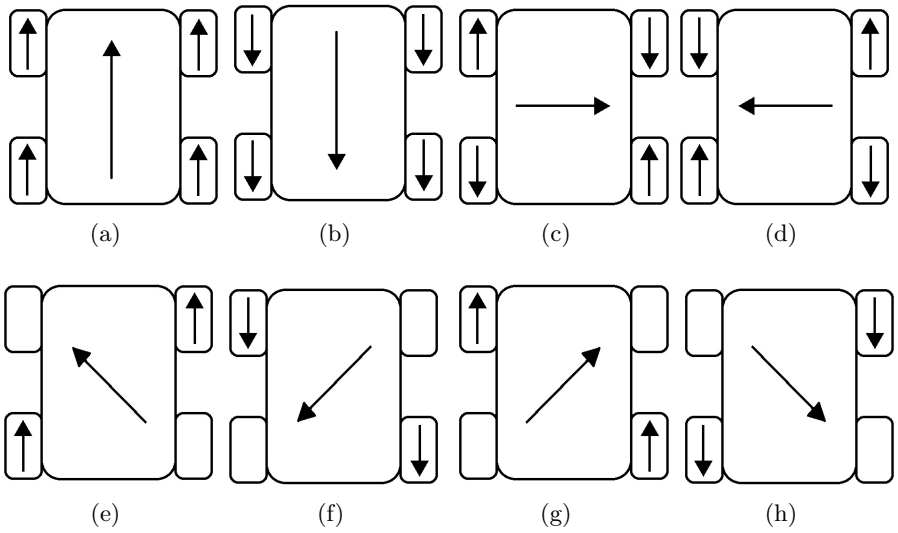


Figure 1.8: Motion of Omnidirectional platform

Figure 1.9: Wheel Equipment Available for Scout Mini

It is a ROS-compatible robot, so it is very versatile for academic and research experiments. The robot can be controlled via local reference system velocities or by controlling each individual wheel, describing trajectories such as those explained in the paragraph above. The robot is equipped with sliding guides that allow us to quickly connect the robot with other structures.

In addition, the robot is equipped with a LidaR laser, which will be discussed later, and with instrumentation, including wifi routers and other electronic equipment that allow us to communicate with the robot from a fixed computer. On the other hand they create encumbrances that will hinder the movement of the plate that will connect the two robots (see Figure: 1.10).

Figure 1.10: Scout Mini Robot with electronic equipment

1.3 Kinematic of a Rigid Body

To better understand the principle of this thesis project, it is good to recall one of the fundamentals of kinematics.

A rigid body is defined as a collection of particles with the property that the distance between the particles remains unchanged during a translation or rotation, barring negligible movements.

For a generic rigid body in a 3D frame with origin in "O" (see Figure 1.11a), we can take two casual points "i" and "j" owned to the rigid body. Defing the vector $\vec{r_i}$ and $\vec{r_j}$ as two time-dependent vectors that connect the origin to the point "i" and "j"; with the tip tail method we can define the vector ($\vec{r_i}$ - $\vec{r_j}$), that for the propperty of the rigid body will be a constant.

Considering for a better understanding the vector $(\vec{r_i} - \vec{r_j}) = \vec{a}$ we know that also $\vec{a}^2$ is a constant.

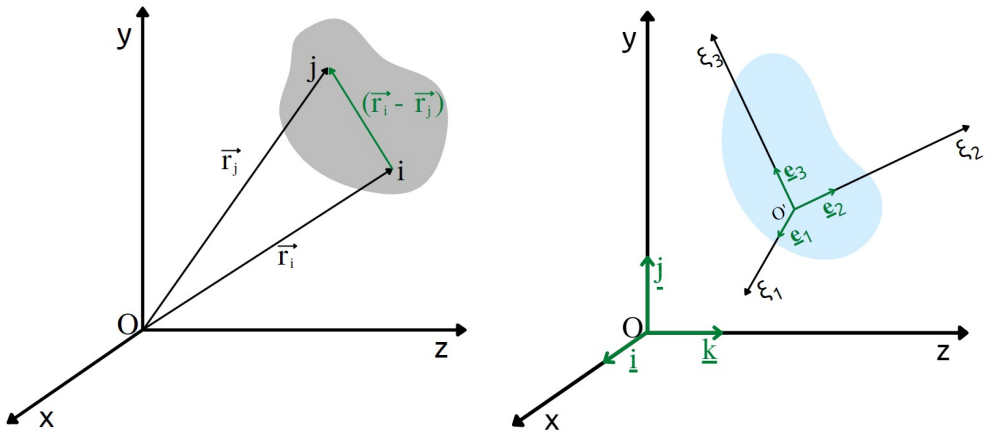
If we derive it we found
$(\vec{r_i} - \vec{r_j}) \cdot \frac{\mathrm{d}(\vec{r_i} - \vec{r_j})}{\mathrm{d}t} = (\vec{r_i} - \vec{r_j}) \cdot (\vec{v_i} - \vec{v_j}) = 0$ so if a scalar product of two vectors is zero it means that these vectors are perpendicular between them, $(\vec{r_i} - \vec{r_j}) \perp (\vec{v_i} - \vec{v_j})$.

For the propperties of a vector product if two vectors are perpendicular, it will exist another vector, that we will call $\vec{\omega}$ that will link them in the following way:

$$(\vec{v_i} - \vec{v_j}) = \vec{\omega} \times (\vec{r_i} - \vec{r_j}) \tag{1.1}$$

If we isolate $\vec{v_i}$ we will find the Fundamental Kinematic Equation of a Rigid Body:

$$\vec{v_i} = \vec{v_j} + \vec{\omega} \times (\vec{r_i} - \vec{r_j}) \tag{1.2}$$



(a) *Two points in a rigid body* (b) *A rigid body with a local frame*

Figure 1.11: Graph of a rigid body in a 3D space.

We do not know anything about the vector $\vec{\omega}$, so we can define a local frame in our body with versors $\underline{e_1}$, $\underline{e_2}$ and $\underline{e_3}$ each one time-dependent (see Figure 1.11b). In an analogous way if we do the same steps we did of the vector $\vec{a}$, we will see that $\underline{e_1} \perp \frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t}$ so it will exist a vector $\vec{\omega_1}$ that:

$$\frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} = \vec{\omega_1} \times \underline{e_1} \tag{1.3}$$

Similarly for the others versors so:

$$\begin{cases} \frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} = \vec{\omega_1} \times \underline{e_1} \\ \frac{\mathrm{d}\underline{e_2}}{\mathrm{d}t} = \vec{\omega_2} \times \underline{e_2} \\ \frac{\mathrm{d}\underline{e_3}}{\mathrm{d}t} = \vec{\omega_3} \times \underline{e_3} \end{cases} \tag{1.4}$$

We can now applying the follow formule for a vector product:
$\vec{x} \times \vec{b} = \vec{c}$
$\vec{x} = \frac{1}{b^2}\vec{b} \times \vec{c} + \lambda\vec{b} \quad \forall\lambda \in \mathbb{R}$, and $\vec{x} \perp \vec{b}$

$$\begin{cases} \vec{\omega_1} = \underline{e_1} \times \frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} + \lambda\ \underline{e_1} \\ \vec{\omega_2} = \underline{e_2} \times \frac{\mathrm{d}\underline{e_2}}{\mathrm{d}t} + \mu\ \underline{e_2} \\ \vec{\omega_3} = \underline{e_3} \times \frac{\mathrm{d}\underline{e_3}}{\mathrm{d}t} + \nu\ \underline{e_3} \end{cases} \tag{1.5}$$

Knowing that $\frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t}$ is perpendicular to $\underline{e_1}$ it is possible to write it as linear combination of $\underline{e_2}$ and $\underline{e_3}$.

$$\frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} = \mathbf{q}\ \underline{e_2} + \mathbf{w}\ \underline{e_3}$$

Multiplying once for $\underline{e_2}$ and then for $\underline{e_3}$ we can find $\mathbf{q}$ and $\mathbf{w}$, so the equation will be:

$$\frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} = (\frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} \cdot \underline{e_2})\ \underline{e_2} + (\frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} \cdot \underline{e_3})\ \underline{e_3}$$

Only now we can performing the vector product in the system of equations 1.5. The results will be:

$$\begin{cases} \vec{\omega_1} = \quad \lambda \quad \underline{e_1} + (-\frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} \cdot \underline{e_3})\ \underline{e_2} + \ (\frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} \cdot \underline{e_2})\ \ \underline{e_3} \\ \vec{\omega_2} = \ (\frac{\mathrm{d}\underline{e_2}}{\mathrm{d}t} \cdot \underline{e_3})\ \ \underline{e_1} + \quad \mu \quad \underline{e_2} + (-\frac{\mathrm{d}\underline{e_2}}{\mathrm{d}t} \cdot \underline{e_1})\ \underline{e_3} \\ \vec{\omega_3} = -(\frac{\mathrm{d}\underline{e_3}}{\mathrm{d}t} \cdot \underline{e_2})\ \underline{e_1} + \ (\frac{\mathrm{d}\underline{e_3}}{\mathrm{d}t} \cdot \underline{e_1})\ \ \underline{e_2} + \quad \nu \quad \underline{e_3} \end{cases} \tag{1.6}$$

We can demostrate that all the terms for each $\vec{\omega_1}$, $\vec{\omega_2}$, $\vec{\omega_3}$ in a single direction are the same. For example:

$$(-\frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} \cdot \underline{e_3}) = (\frac{\mathrm{d}\underline{e_3}}{\mathrm{d}t} \cdot \underline{e_1}) \tag{1.7}$$

That is because $\underline{e_1} \cdot \underline{e_3} = 0$ due to their perpendicularity.

If we derive it $\frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} \cdot \underline{e_3} + \underline{e_1} \cdot \frac{\mathrm{d}\underline{e_3}}{\mathrm{d}t} = 0$ so the equation 1.7 is verified.

This system of equation must be valid for every λ, μ, ν so we take them so that $\vec{\omega_1} = \vec{\omega_2} = \vec{\omega_3} = \vec{\omega}$. So we take:

$$\begin{cases} \lambda = \frac{\mathrm{d}\underline{e_2}}{\mathrm{d}t} \cdot \underline{e_3},\ \mu = \frac{\mathrm{d}\underline{e_3}}{\mathrm{d}t} \cdot \underline{e_1},\ \nu = \frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} \cdot \underline{e_2} \\ \vec{\omega} = \lambda\ \underline{e_1} + \mu\ \underline{e_2} + \nu\ \underline{e_3} \end{cases} \tag{1.8}$$

Finally we can rewrite the equations 1.4 finding the Poisson's Equations.

$$\begin{cases} \frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} = \vec{\omega} \times \underline{e_1} \\ \frac{\mathrm{d}\underline{e_2}}{\mathrm{d}t} = \vec{\omega} \times \underline{e_2} \\ \frac{\mathrm{d}\underline{e_3}}{\mathrm{d}t} = \vec{\omega} \times \underline{e_3} \end{cases} \tag{1.9}$$

Now if we take a casual point "P" in our body and we define the vectors $\vec{r_{o'}}$,$\vec{r_P}$ and $\vec{\xi_P}$ as shown in Figure 1.12 by carrying out the following calculations, we arrive at the fundamental law of kinematics for the rigid body knowing what the omega vector is.

$$\vec{r_P} = \vec{r_{o'}} + \vec{\xi_P}$$
$$\vec{\xi_P} = \vec{r_P} - \vec{r_{o'}}$$

$$\vec{\xi_P} = \xi_1\ \underline{e_1} + \xi_2\ \underline{e_2} + \xi_3\ \underline{e_3}$$

Where ξ_1, ξ_2 and ξ_3 are constants.

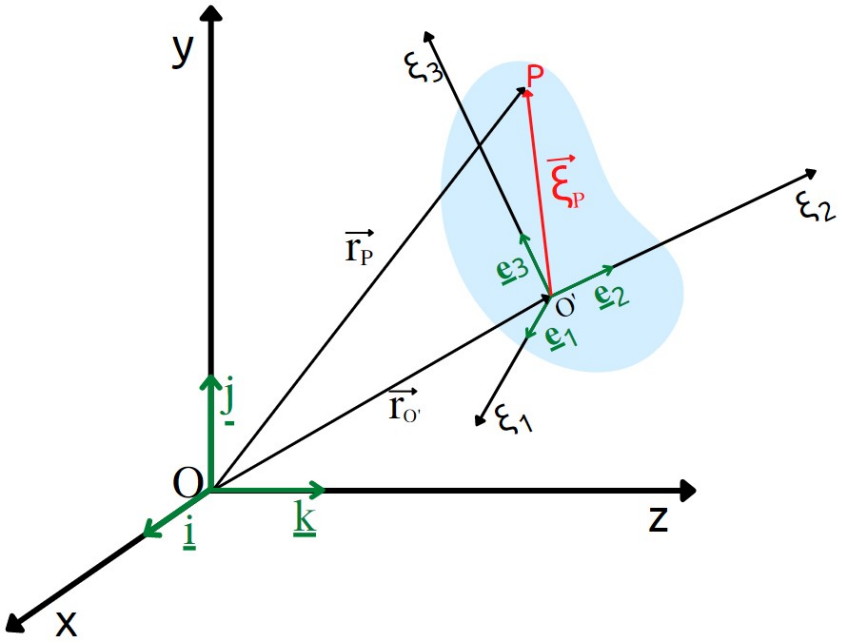


Figure 1.12: Another rapresentation of a rigid body in a 3D space

$$\vec{v_P} = \vec{v_{o'}} + \frac{\mathrm{d}\vec{\xi_P}}{\mathrm{d}t}$$
$$\frac{\mathrm{d}\vec{\xi_P}}{\mathrm{d}t} = \xi_1 \frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} + \xi_2 \frac{\mathrm{d}\underline{e_2}}{\mathrm{d}t} + \xi_3 \frac{\mathrm{d}\underline{e_3}}{\mathrm{d}t}$$

But for the Poisson's Equations:

$$\begin{aligned}\frac{\mathrm{d}\vec{\xi_P}}{\mathrm{d}t} = \xi_1\, \vec{\omega} \times \underline{e_1} + \xi_2\, \vec{\omega} \times \underline{e_2} + \xi_3\, \vec{\omega} \times \underline{e_3} = \\ = \vec{\omega} \times (\xi_1\, \underline{e_1} + \xi_2\, \underline{e_2} + \xi_3\, \underline{e_3}) = \\ = \vec{\omega} \times \vec{\xi_P}\end{aligned}$$

Finally we obtain:

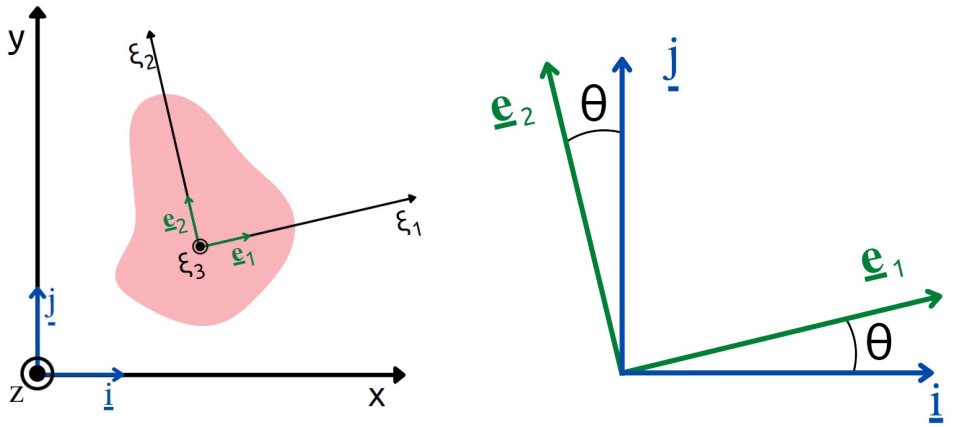
$$\begin{cases} \vec{v_P} = \vec{v_{o'}} + \vec{\omega} \times \vec{\xi_P} \\ \vec{v_P} = \vec{v_{o'}} + \vec{\omega} \times (\vec{r_P} - \vec{r_{o'}}) \end{cases} \tag{1.10}$$

$\vec{\omega}$ is indipendent from the origin O' chosed and it is unique. This last statement can be proved by absurdity.

1.3.1 Physic meaning of Omega

For better understand the meaning of the vector $\vec{\omega}$ is better to consider a 2D rigid body in a two-dimensional reference system (see Figure 1.13a), where the z-direction

is outgoing from the sheet.



(a) *Rigid body with a 2D local frame* (b) *Comparison between local and global frame*

Figure 1.13: Rigid body in 2D space

In that case: $\underline{e_3} = \underline{k}$ and does not change over time ($\frac{\mathrm{d}\underline{e_2}}{\mathrm{d}t} = 0$),

$$\begin{cases} \lambda = \frac{\mathrm{d}\underline{e_2}}{\mathrm{d}t} \cdot \underline{e_3}, \ \mu = 0, \nu = \frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} \cdot \underline{e_2} \\ \vec{\omega} = \lambda \ \underline{e_1} + \nu \ \underline{e_3} \\ \lambda = \frac{\mathrm{d}\underline{e_2}}{\mathrm{d}t} \cdot \underline{e_3} = -\frac{\mathrm{d}\underline{e_3}}{\mathrm{d}t} \cdot \underline{e_2} \\ \vec{\omega} = \frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} \cdot \underline{e_2} \ \underline{e_3} \end{cases} \tag{1.11}$$

If we now describe $\underline{e_1}$ and $\underline{e_2}$ as follow (see Figure 1.13b):
$\underline{e_1} = \cos\theta \ \underline{i} + \sin\theta \ \underline{j}$
$\underline{e_2} = -\sin\theta \ \underline{i} + \cos\theta \ \underline{j}$

$\frac{\mathrm{d}\underline{e_1}}{\mathrm{d}t} = \dot{\theta} \ (-\sin\theta \ \ \underline{i} + \cos\theta \ \underline{j}) = \dot{\theta} \ \underline{e_2}$

So $\vec{\omega}$ is the vector "angular velocity" perpendicular to the moving plane with module as the angular velocity: $\vec{\omega} = \dot{\theta} \ \underline{e_2} \cdot \ \underline{e_2} \ \underline{e_3} = \dot{\theta} \ \underline{k}$

To summarise, if we have a rigid body and we know the position of a point that belongs to this body we can know the position and the velocity of all points belonging to the rigid body due to the vector $\vec{\omega}$ and the vectors that connect a initial frame with the local frame of the body. The vector $\vec{\omega}$ is the angular velocity and in a 2D motion does not change, in all the point of the rigid body is the same.

Chapter 2

Programming Background

2.1 Introduction of ROS

The purpose of this chapter is to illustrate the systems used to create the code for the simultaneous movement of the two robots. First of all, two main tasks can be distinguished:

- the creation of the actual code for handling the robot;
- the methodology for tracking the distance between the robots, with speed correction.

The first involved the use of ROS (Robot Operating System) which is a system that allows us to communicate with the robot through different programming languages. The second was done using two different methodologies: one using LIDAR (Light Detection And Ranging), a laser located above the robot, and the other using the Adaptive Monte Carlo Localization (AMCL) method. Using in both cases the robot visualisation and localisation programme Rviz. These two tools, in combination, allow us to verify the behaviour of the robots in real time, allowing trajectory and speed correction with a feedforward control system.

2.1.1 ROS - Robot Operating System

ROS is an open-source, meta-operating system that provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management [5]. Unlike conventional operating systems, it can be used for numerous combinations of hardware implementation. ROS is a supporting system for controlling a robot and sensor with an hardware abstraction and developing robot application based on existing conventional operating systems. For that it could be defined as a meta operating system because it runs on existing operating systems(see Figure2.1).

Using ROS give us different advantages: ROS data communication is supported not only by one operating system, but also by multiple operating systems, hardware, and programs, making it highly suitable for robot development where various hardware

Figure 2.1: ROS as meta operating system [5].

are combined. In addiction, different types of programming languages can be used: the ROS program provides a client library to support various programming languages. The library can be imported in programming languages that are popular in the robotics field such as Python, C++, and Lisp as well as languages such as JAVA, C#, Lua, and Ruby. In other words, you can develop a ROS program using a preferred programming language. Package management: multiple processes having the same purpose are managed as a package so that it is easy to use and develop, as well as convenient to share, modify, and redistribute. Public repository: each package is made public to the developer's preferred public repository (e.g., GitHub) and specifies its licence.[5]. The main purpose of ros is to build an environment where we can develop robotic softwares that can communicate each other to reach a certain goal.

2.1.2 How Ros Works

To better understand how ROS works, it could be useful to explain the most used ROS terms.

ROS is developed in units of nodes. We can think nodes as an executable program, the smallest unit of processor running in ROS. Nodes are created for each purpose such as sensor data conversion, obstacle recognition, motor drive, encoder input, and navigation.

A node registers information such as name, message type, URI address and port number of the node. The registered node can act as a publisher, subscriber, service server or service client based on the registered information, and nodes can exchange messages using topics and services.

Messages are variables such as integer, floating point, and Boolean that can be sends or received between nodes. Without the *master* the connection between nodes

and messages is impossible. The *master* acts as a name server for node-to-node connections and message communication. The command *roscore* is used to run the master, and if you run the master, you can register the name of each node and get information when needed.

The *topic* can be imagined as a conversation topic. The publisher nodes give commands to the topic and subscriber nodes receive the information from the topic that are maybe changed because of the publisher command.

The term 'publish' stands for the action of transmitting relative messages corresponding to the topic. The *publisher node* registers its own information and topic with the master and sends a message to connected subscriber nodes that are interested in the same topic[5].

The *subscriber node* in fact, will receive the relative messages corresponding to the topic and will register its own information with the master and receive publisher information that publishes relative topic from the master. A subscriber is declared in the node and can be declared multiple times in one node, as well as in the publisher one.

For execute multiple nodes, we can create *launch files* that with the command *roslaunch* can execute different nodes at the same time and others operations such as configuring namespace and nodes changing package parameters, node names, environment variables etc.

Hence ROS is the system that allow the communication of robot and computer or other devices towards messages from different nodes to different topics. The type of communication could be different. For exchanging messages there are three different ways followed, described and summarized in the table 2.1:

- a topic which provides a unidirectional message transmission/reception;
- a service which provides a bidirectional message request/response;
- an action which provides a bidirectional message goal/result/feedback.

Table 2.1: Topics, services and actions in ROS[5].

Type	**Features**		**Description**
Topic	Asynchronous	Unidirectional	Used when exchanging data continuously
Service	Synchronous	Bi-directional	Used when exchanging data continuously
Action	Asynchronous	Bi-directional	Used when it is difficult to use the service due to long response times after the request or when an intermediate feedback value is needed

It is good to remark that topics transmit unidirectional continuous flow (that remain connected) to send or receive messages. They are suitable for sensor data that requires publishing messages periodically. On the other hand, services don't

maintain the connection so are useful to reduce the load of the network by replacing topic.

In the end, action servers perform the actions and send the feedback to the action client. They are often used to command complex robot tasks such as cancelling transmitted goal while the operation is in progress.

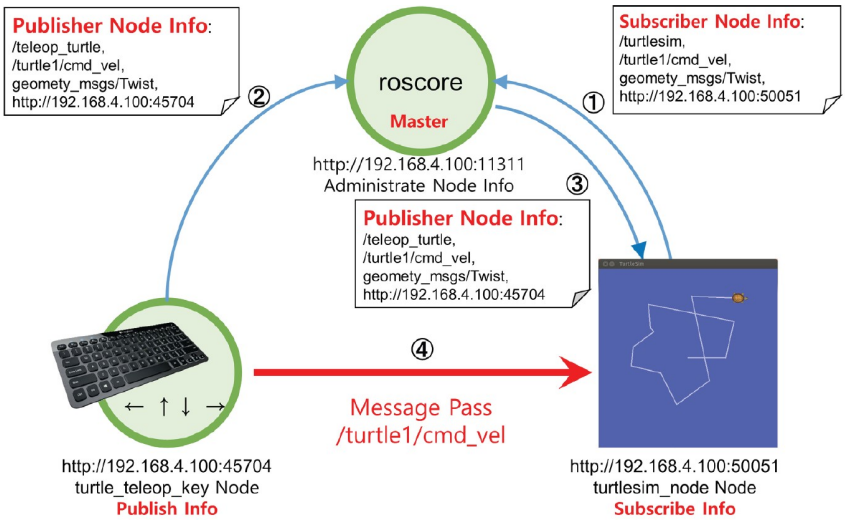


Figure 2.2: Message communication example [5].

In the figure above (see Figure 2.2) we can see an example of message communication that could be useful to understand how the communication between topics, subscribers and publishers work. In this example have been used two nodes, the master and the /turtle1/cmd_vel' topic. Turtlesim_node and turtle_teleop_key node are both connected with the master. Turtlesim node as a subscriber will give information to the master and that will provide the visual motion of the simulated robot. Turtle_teleop, instead, is written as a publisher; pressing the arrows in the keyboard it will give transational and rotational messages to the master but also directly to the virtual robot to perform the desired movement. The codes presented in this work are similar to this one, one or more publishers control the robot motion and subscribers take information from the simulated or real environment and from the robot itself to know the position or the speed of it.

2.2 Simulations

Simulations have been very usefull in this thesis project. There are a lot of software capable to simulate a real enviroment and a real behaviuor of the robot. However a very popular simulator in the field of robotics is Gazebo because of its high performance even though it is open source. It is strictly related to ROS because developed and distribuited from the same company. Other simulation sistem are used to the visualization to know and identificate the position of the robot or to

measure the distance from the obstacles etc. One of this type of system is Rviz, explained in the next section.

2.2.1 Gazebo Simulator

Gazebo is a 3D simulator that provides robots, sensors, environment models for 3D simulation required for robot development, and offers realistic simulation with its physics engine. Installing the package TurtleBot3 on Gazebo simulator we can access to different robot model we can upload in the environment. In addiction different simulated maps, already settled, can be placed. These will allow the robot to move around a place with obstacles, to detect them and to avoid them during the navigation. In our case the robot model uploaded was the MIR200, because as explained before there was not availabilty of a scout mini omnidirectional robot model.

As described in the section (1.2.2), MIR200 is equipped with two distance lasers and with a camera. We can see this type of model in Figure (2.3). The MIR200 model was placed in a "big square map" (as shown in Figure 2.4) that contains obstacles and allows to see topics that will be not visible in an empty map.

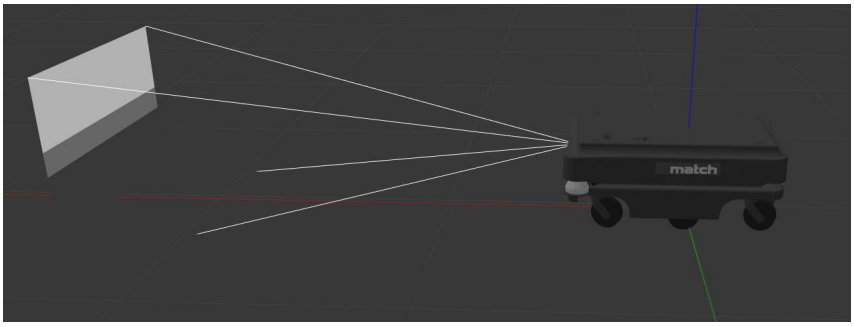


Figure 2.3: MIR200 model in Gazebo.

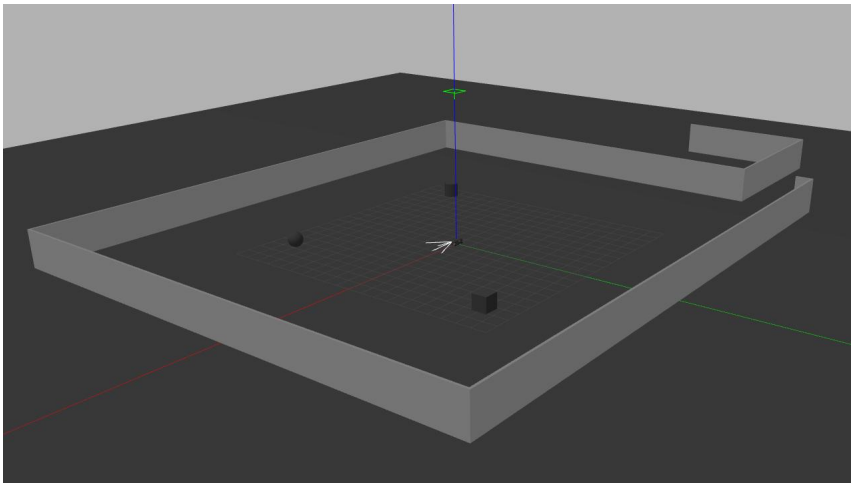


Figure 2.4: Example of a Map in Gazebo

2.2.2 Rviz

ROS can be used with different tools, these could be of visualization like the one described in this pharagraph. Rviz is a 3D visualization tool that transforms ROS messages in something we can watch virtually in a 3D space. For example it is possible to visualize objects near our robot due to a LSD (Laser Distance Sensor) or what a camera can see, if it is installed in the robot. With the laser has been possible to create a map of where the robot was placed, to identify the fixed obstacle, walls, desktops, computer etc and the movable one's like chairs people etc. A mobile robot model in Rviz can be displayed and received distance data from the LSD laser (see Figure 2.6), this option has been used in this thesis project for calculate the distance between another robot, but the details will be written later in this lecture. With this functionality the robot can localize itself and could be autonomous to go to a certain position without colliding objects. In general, it can be used in navigation, see figure 2.5.

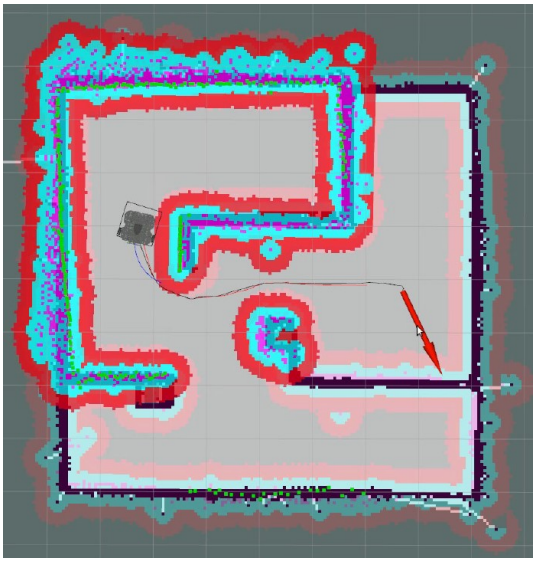


Figure 2.5: Example of navigation in Rviz [5].

In Rviz we can see the active topic that we had booted in ROS with Python nodes or launch files. The most interesting topic that has been used in this thesis are:

- Axes which display the xyz axes of what we need, such as the robot frame or a robot goal position.
- Laser scan display scan values, useful to see what the robot perceives near it.
- Map displays the occupancy map, used in navigation, on top of the ground Plane.
- Robot model displays the robot in a 3D shape already defined by the constructor of the robot or by developer that let his work open source.

- TF displays the coordinate transformation TF used in ROS. It is displayed with the xyz axes much like the previously mentioned axes, but each axis expresses the hierarchy with an arrow according to the relative coordinates.

- Pose Displays the pose (location + orientation) on 3D. The pose is represented in the shape of an arrow where the origin of the arrow is the position(x, y, z,) and the direction of the arrow is the orientation (roll, pitch, yaw). For instance, pose can be represented with the position and orientation of the 3D robot model, while it can be represented with the goal point.

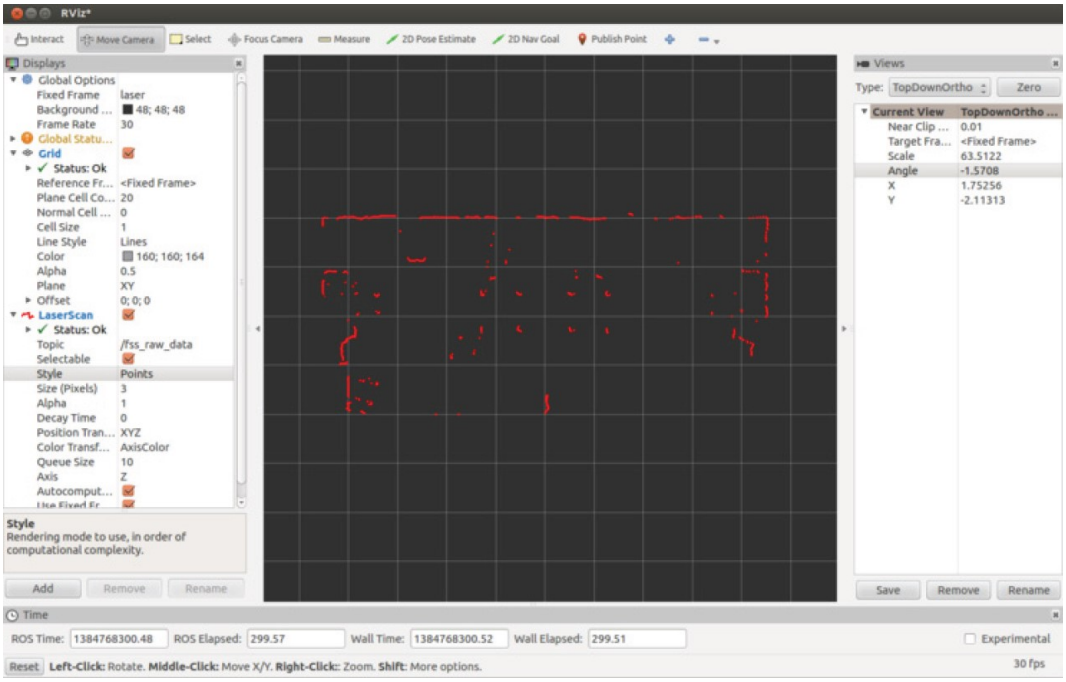
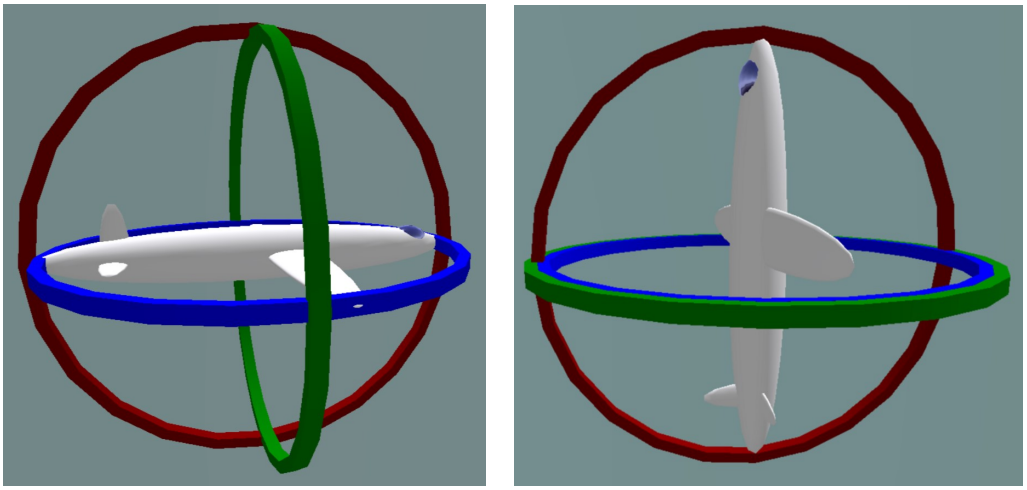


Figure 2.6: Measuring distance with LSD laser [5].

With them we controlled the actual and target position of the follower robot, in order to know if it followed the right path and to know how accurate it was to maintain the distance constant.

2.2.3 Orientation and Pose of the Robot

The pose of a robot can be described as a combination of positions and orientations. Here, the position is expressed by three vectors x, y, and z, and the orientation by four vectors x, y, z, and w, called a quaternion. The quaternion is not intuitive because they do not describe the rotation of three axes (x, y, z), such as the roll, pitch, and yaw angles that are often used. However, the quaternion form is free from the gimbal lock or speed issues that present in the Euler method of roll, pitch and yaw vectors. The gimbal lock is a problem that happens when two axis align together. It will cause a loss of one degree of freedom in a three-dimensional space.



(a) *A* normal three-gimbal set. (b) *A* gimbal lock.

Figure 2.7: Gimball lock illustration

As shown in Figure 2.7 when one axes overlaps with another (causing a lock) the model in the figure offset the blue one with a specific value, or changes the plane to the opposite direction, such as from 90 degree to -90 degree at once. The first scenario means when I rotate my device from left to right, my models may rotate from top to bottom as well, even though I don't rotate my device in such manner. Therefore, the quaternion type is preferred in robotics, and ROS also uses quaternion for this reason. Of course, functions to convert Euler values to quaternions are provided for convenience. In our case to visualize in a better way the orientation of our robots it has been used the Coordinate Transformation (TF) package that lets the user keep track of multiple coordinate frames over time and can easily transform from eulerian coordinate to quaternian ones and vice-versa.

Chapter 3

Controlling Two Robots

The structure studied in this thesis can be composed of two Scout Mini robots, so two holonomic robot, or two MIR200. The original project consisted on develop a structure composed by two holonomic robot connected with a platform, but during the internship there was not the availability of two fully functional holonomic robot, so we create a structure with two MIR200. However the kinematic of two holonomic synchronize robot has been studied.

3.1 Kinematics

Regarding the consideration made in section 1.3, our structure could be represented as a rigid body moving on the 2D space, where we need to know and control the velocity of two points. These two points are the baricentric point of our robots (see Figure 3.1).

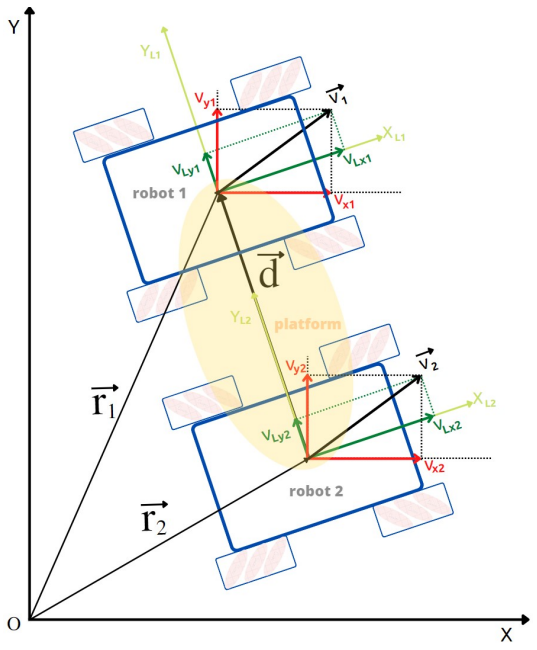


Figure 3.1: Schematization of the Structure.

Controlling the structure with a leader-follower aproach, we will control directly

the leader robot and the second one will follow the leader in a certain way.

However, we have to distinguish the original project from the one that was realised at the end.

3.1.1 Kinematics of a Structure of two Holonomic Robots

Taking in consideration the Figure (3.1) applying the fundamental law of kinematic (equation 1.10) we will know the global frame velocities of the second robot, knowing these of the leader (robot 1).
$\vec{v_2} = \vec{v_1} + \vec{\omega_z} \times \vec{d}$
$\vec{v_2} = \vec{v_1} + \vec{\omega_z} \times (\vec{r_2} - \vec{r_1})$

That into a matrix form is:

$$\vec{v_{L1}} = \begin{pmatrix} v_{Lx1} \\ v_{Ly1} \\ \omega_{L1} \end{pmatrix}$$

$$\vec{v_{G1}} = \begin{pmatrix} v_{Gx1} \\ v_{Gy1} \\ \omega_{G1} \end{pmatrix} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} v_{Lx1} \\ v_{Ly1} \\ \omega_{L1} \end{pmatrix}$$

$$\vec{v_{G2}} = \begin{pmatrix} v_{Gx2} \\ v_{Gy2} \\ \omega_{G2} \end{pmatrix} = \begin{pmatrix} v_{Gx1} \\ v_{Gy1} \\ \omega_{G1} \end{pmatrix} + \vec{\omega_1} \times \begin{pmatrix} (x_2 - x_1) \\ (y_2 - y_1) \\ 0 \end{pmatrix}$$

$$\vec{v_{L2}} = \begin{pmatrix} v_{Lx2} \\ v_{Ly2} \\ \omega_{L2} \end{pmatrix} = \begin{bmatrix} \cos\theta_2 & \sin\theta_2 & 0 \\ \sin -\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} v_{Gx2} \\ v_{Gy2} \\ \omega_{G2} \end{pmatrix} \tag{3.1}$$

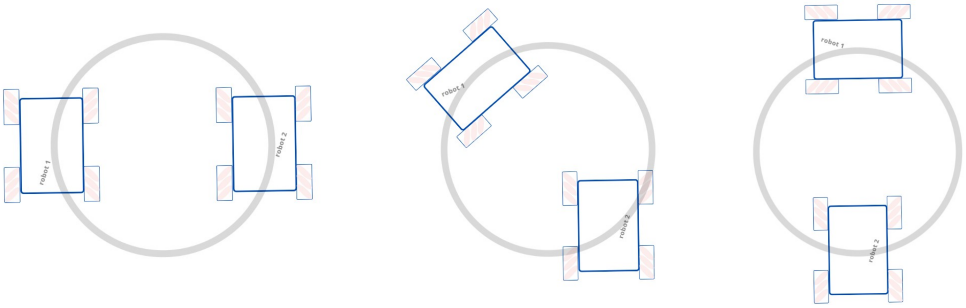
Each ω with any subscript are the same in a synchronize motion. If we know the velocities of the robot 1 we can give the velocities of the robot 2 in order to maintain costant the distance between the robot.

Omnidirectional robots can also maintain the distance between them by changing their orientation. Anyway this motion is a particular case that can be performed giving certains commands that will change the set-up of the second robot that will execute the different motion, changing its orientation (see Figure 3.2).

In this section we assumed that we can control the local velocities of the robot. Actually for the Scout mini AgileX is like this. Therefore it is important to know how the velocities are splitted for each mecanum wheel.

3.1.2 Kinematic of a Mecanum wheel System

A typical system of four mechanum wheels is shown in Figure 3.3. The parameters of this configuration are shown in table 3.1. In this configuration wheels sizes are the same[16].



(a) *S*tep 1. (b) *S*tep 2. (c) *S*tep 3.

Figure 3.2: Particular motion of a omnidirecional structure.

Table 3.1: Robot Parameters.

i	Wheels	α_i	β_i	γ_i	l_i	l_{ix}	l_{iy}
0	1sw	$\pi/4$	$\pi/2$	$-\pi/4$	l	l_x	l_y
1	2sw	$-\pi/4$	$-\pi/2$	$\pi/4$	l	l_x	l_y
2	3sw	$3\pi/4$	$\pi/2$	$\pi/4$	l	l_x	l_y
3	4sw	$-3\pi/4$	$-\pi/2$	$-\pi/4$	l	l_x	l_y

The configuration parameters and system velocities are defined as follows:

- x,y, θ robot's position (x, y) and its orientation angle θ (The angle between X and X_R);
- XGY, inertial frame; x,y are the coordinates of the reference point O in the inertial basis;
- $X_R OY_R$ robot's base frame; Cartesian coordinate system associated with the movement of the body center.
- $S_i P_i E_i$, coordinate system of ith wheel in the wheel's center point P_i;
- O,P_i, the inertial basis of the Robot in Robot's frame and $P_i = X_{Pi}, Y_{Pi}$ the center of the rotation axis of the wheel i;
- $\vec{OP_i}$, is a vector that indicates the distance between Robot's center and the center of the wheel ith;
- l_{ix}, l_{iy}, l_i,half of the distance between front wheels and l_{iy} half of the distance between front wheel and the rear wheels.
- l_{ix} distance between wheels and the base (center of the robot O);
- r_i, denotes the radius of the wheel i (Distance of the wheel's center to the roller center)
- r_r, denotes the radius of the rollers on the wheels.
- α_i, the angle between OP_i and X_R;
- β_i, the angle between S_i and X_R;

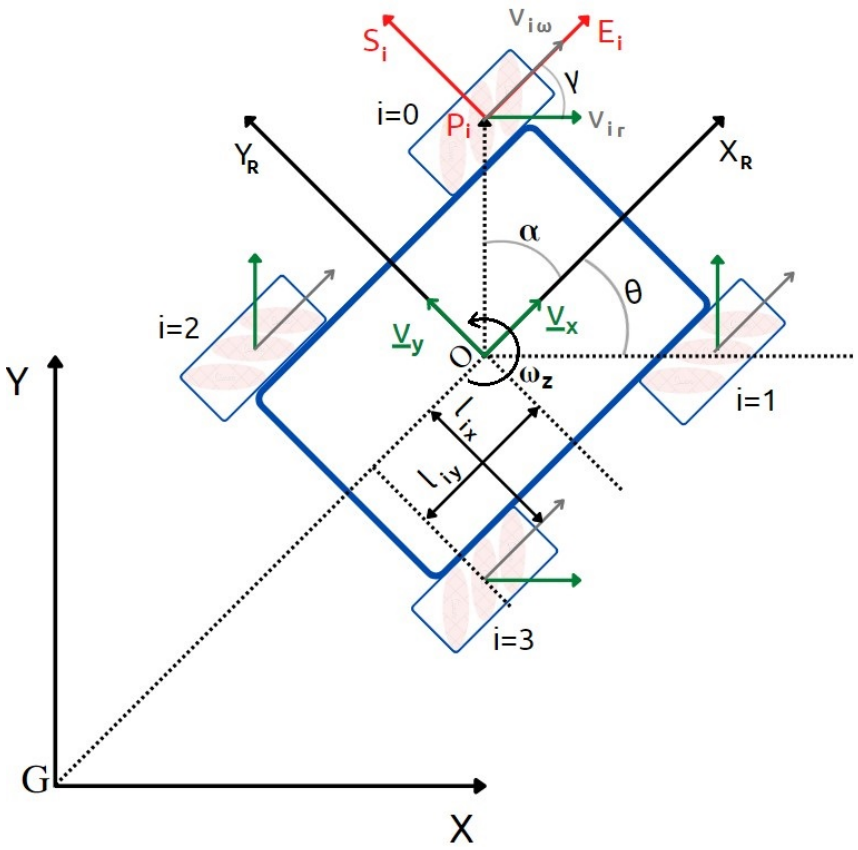
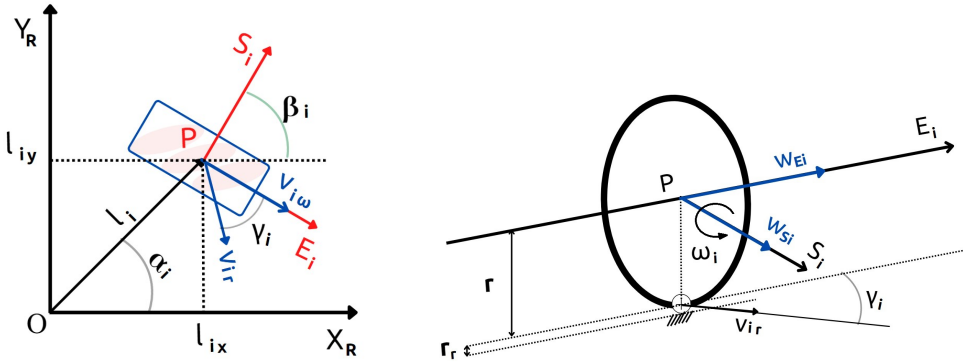


Figure 3.3: Wheels Configuration and Posture definition.

- γ_i, the angle between v_{ir} and E_i;
- ω_i [rad/s], wheels angular velocity;
- $v_{i\omega}$ [m/s], (i= 0,1,2,3) $\in$ R, is the velocity vector corresponding to wheel revolutions;
- v_{ir} the velocity of the passive roller in the wheel i;
- $[w_{si}\ w_{Ei}\ \omega_i]^{\mathrm{T}}$, Generalized velocity of point P_i in the frame $S_iP_iE_i$;
- $[v_{si}\ v_{Ei}\ \omega_i]^{\mathrm{T}}$, Generalized velocity of point P_i in the frame X_ROE_i;
- $v_x\ v_y$ [m/s] - Robot linear velocity;
- ω_z [rad/s], Robot angular velocity.

Developing the calculs, fully described in the article [16] we can arrive at the following results. They will allow us to control the velocity of each wheel knowing the velocity of the baricentrum of the robot.

$$\begin{pmatrix}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{pmatrix} = \frac{1}{r}\begin{bmatrix}1 & -1 & -(l_x+l_y)\\ 1 & 1 & (l_x+l_y)\\ 1 & 1 & -(l_x+l_y)\\ 1 & -1 & (l_x+l_y)\end{bmatrix} \cdot \begin{pmatrix}v_x\\ v_y\\ \omega_z\end{pmatrix}$$



(a) *W*heel i in the robot coordinate. (b) *W*heel i motion principle.

Figure 3.4: Parameters of ith wheel.

$$\begin{cases} \omega_1 = \frac{1}{r}(v_x - v_y - (l_x + l_y)\ \omega_z \\ \omega_2 = \frac{1}{r}(v_x + v_y + (l_x + l_y)\ \omega_z \\ \omega_3 = \frac{1}{r}(v_x + v_y - (l_x + l_y)\ \omega_z \\ \omega_4 = \frac{1}{r}(v_x - v_y + (l_x + l_y)\ \omega_z \end{cases}$$

We can also know how much the local velocities are in x and y direction knowing the angular velocities of each wheel, in other words using the inverse formula.

$$\begin{pmatrix} v_x \\ v_y \\ \omega_z \end{pmatrix} = \frac{r}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -\frac{1}{(l_x+l_y)} & \frac{1}{(l_x+l_y)} & -\frac{1}{(l_x+l_y)} & \frac{1}{(l_x+l_y)} \end{bmatrix} \cdot \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix}$$

If we know the velocities of the wheels and some of them are zero, we can come back to the basic movement of an holonomic robot (see Figure 1.9).

3.1.3 Kinematics of a structure with two MIR200 robots

Studying the kinematic of a non-holonomic structure we have to add the constrain that the angle θ_1 and θ_2 are the same.

Indicating with:

$$\vec{v_{L1}} = \begin{pmatrix} v_{Lx1} \\ v_{Ly1} \\ \omega_{L1} \end{pmatrix} = \begin{pmatrix} \dot{x_{L1}} \\ \dot{y_{L1}} \\ \dot{\theta_{L1}} \end{pmatrix}$$

and the other vectors with an analogous form we now want to control only the speed in direction x, and the agular velocity "ω" in the local frame of the robot, so it is obvious that the velocity in direction "y" is zero. The vector will come:

$$\vec{v_{L1}} = \begin{pmatrix} \dot{x_{L1}} \\ 0 \\ \dot{\theta_{L1}} \end{pmatrix}$$

Now we want to convert this vector in the global frame "$\vec{v_{G1}}$". Excluding the angular velocity ω, which will be the same in all configurations (both in the local and global framing of each robot) from the inverse rotation matrix, the vector will result:

$$\vec{v_{G1}} = \begin{pmatrix} \dot{x_{G1}} \\ \dot{y_{G1}} \end{pmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \cdot \begin{pmatrix} \dot{x_{L1}} \\ 0 \end{pmatrix} \tag{3.2}$$

$$\begin{cases} \dot{x_{G1}} = \cos\theta \ \dot{x_{L1}} \\ \dot{y_{G1}} = \sin\theta \ \dot{x_{L1}} \end{cases}$$

$$\vec{v_{G2}} = \begin{pmatrix} \dot{x_{G2}} \\ \dot{y_{G2}} \end{pmatrix} = \begin{pmatrix} \dot{x_{G1}} \\ \dot{y_{G1}} \end{pmatrix} + \vec{\omega} \times \begin{pmatrix} (x_2 - x_1) \\ (y_2 - y_1) \end{pmatrix} \tag{3.3}$$

$$\begin{cases} \dot{x_{G2}} = \cos\theta \ \dot{x_{L1}} - \omega \ (y_2 - y_1) \\ \dot{y_{G2}} = \sin\theta \ \dot{x_{L1}} + \omega \ (x_2 - x_1) \end{cases}$$

In the local frame they are:

$$\vec{v_{G2}} = \begin{pmatrix} \dot{x_{L2}} \\ \dot{y_{L2}} \\ \omega \end{pmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \dot{x_{G2}} \\ \dot{y_{G2}} \\ \omega \end{pmatrix} \tag{3.4}$$

that in an extended form is:

$$\vec{v_{L2}} = \begin{cases} \dot{x_{L2}} = \cos^2\theta \ \dot{x_{L1}} - \omega \ \cos\theta \ (y_2 - y_1) + \sin^2\theta \ \dot{x_{L1}} + \omega \ \sin\theta \ (x_2 - x_1) \\ \dot{y_{L2}} = -\omega \ \sin\theta \ (y_2 - y_1) + \omega \ \cos\theta \ (x_2 - x_1) \\ \vec{\omega} = \vec{\omega} \end{cases}$$

With two non-holonomic robot we can determine that:

$$(x_2 - x_1) = d \ \sin\theta$$

$$(y_2 - y_1) = d \ \cos\theta$$

So the entity on y direction will disappear as aspected.

$$\begin{cases} \dot{x_{L2}} = \dot{x_{L1}} + \omega \ d \ (1 - 2 \ \cos^2\theta) \\ \dot{y_{L2}} = \omega \ d \ \sin\theta \ \cos\theta - \omega \ d \ \cos\theta \ \sin\theta = 0 \\ \vec{\omega} = \vec{\omega} \end{cases}$$

3.2 Programs

Knowing the kinematic of the robot we can start to write a program that will control the second robot. First it has been studied how to perform basic movemnt of a single robot, that have been used later for controlling the leader robot. Basic programs had been developed in the first period of the thesis project. Then the OOP method (Object-Oriented Programming) has been integrated with the associated different advantages.

First of all we have to connect the robot with our computer or create a simulation environment in Gazebo. For the first request the MATCH institute in Leibniz University has developed commands to connect the computer to the Scout Mini robot. The simulation environment has been created with a launch file that will boot a "big square map" placing two MIR200 robots called mir1 and mir2 in a certain position. The leader robot (mir1) is placed in the origin and the follower robot is placed one meter far from the origin in the x direction.

The topics used as *Subscriber*, to get information from the robot are: ground_truth, amcl_pose, odom or similar. Those used as *Pubisher*, to publish information to the robots, are cmd_vel or similars.

To see an example of how they are linked together, we can see the Figure 3.5 where we can see that the two topics *ground* of the mir_1 and mir_2 robots go to the function circled in red, which performs the calculations and gives the speed messages to mir_2 via the topic *cmd_vel*. The latter topic will interact with gazebo so as to show the movement of the mir_2 robot.

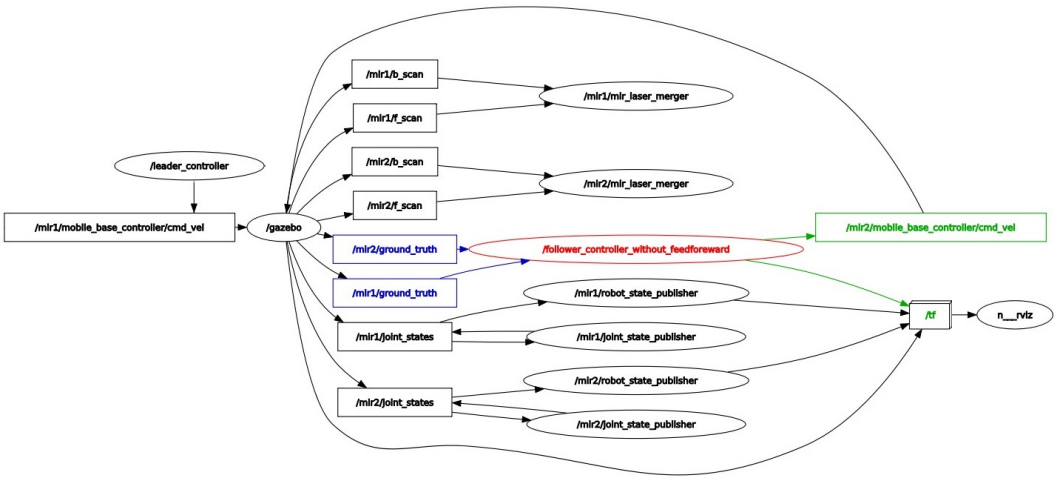


Figure 3.5: Node graph.

Several codes were written, some controlling the Scout Mini Rover robot others the two MIR200 robots in the simulation environment, essentially only the topics change, because the type of control is the same.

Analysing the code that controls the robot follower, we see that first of all, all the necessary packages have been added to implement a programme that can be used

on ROS and with which we can perform all the mathematical calculations. A class, according to the OOP method, dominates the code. It consists of several functions that perform certain tasks each. In the constructor "___init___", the attributes of the robot have been defined: the distance between the two robots and the appropriate publishers and subscribers. For this code, three *Subscribers* have been identified, each finding respectively: position and orientation of robot 1, position and orientation of robot 2 and speed of robot 1, where with robot 1 we define the leading robot and with robot 2 the follower.

All three *Subscribers* have as reference topics, from which they take data, as *ground_truth* topic of their respective robots. There is also a *Publisher* that provides speed information to robot 2, hence it is able to command it. Therefore the various messages to be exchanged between *Subscribers* and *Publishers* were described, as they will be in the form of position (Odometry) for the former and speed (Twist) for the latter.

Then, the callback functions were defined, which are useful for the *Subscribers* that must provide information on the position and orientation of the robots. The latter is expressed in quaternions so an appropriate function transforms it into radians. These callback functions will be applied to both robot 1 and robot 2. As for the *Subscriber*, which obtains information on the speed of robot 1, it will have an appropriate callback function that will have messages of type Twist with angular and linear velocities in all three spatial directions, although one will be zero (because we are in plane motion) and therefore the variable will not be called back.

The equation 3.3 was then used to calculate the global velocity of robot 2, and then transformed to local with the appropriate steps described in section 3.1.3, defining the local linear velocity of robot 2 in the x-direction and angular velocity in the z-direction.

The steps to obtain the global velocity of robot 1 from the local velocity were not done because the topic ground_truth is defined in the global reference system.

In the main function, the node name is defined, as programming for ROS requires, and the class with the described functions is executed in a loop. In this way, robot 2 will follow robot 1 in a synchronised manner, accelerating in a curve if outside and decelerating if inside, maintaining a constant distance between the robots.

3.2.1 Closed Loop Control Systems

In section 3.2, we described the programme to control the second robot which will follow the first one in a synchronised manner and maintain a constant distance equal to the initial one. With this method, however, we cannot guarantee that this distance remains the same over time, becouse external agents or noise and inaccuracy of all the sensors (IMU and encoder) could change the trajectory of the second robot, which could even collide with the leading robot. The robot control system was therefore implemented to prevent any misalignment due to mechanical issues such as wheel

slippage from leading the follower robot into an incorrect trajectory. For this, it was appropriate to include a closed-loop control system with feedback in the code.

A function was created based on the following schematisation (see Figure 3.6). In this function we have to provide three types of information: the actual position, the target position and the desired (target) speed of robot 2. We obtain information of the current position of robot 1 and 2 via the same topics used in the description in section 3.2 by defining them now as "Actual Position ROBOT". By defining the distance between the two robots as 'd' and decomposing it into the two directions 'x' and 'y' (d_x and d_y), the desired theoretical position (target position) of robot 2 in the local reference system was calculated by applying the rotation matrix.

$$\begin{cases} x_2 = x_1 + d_x \ \cos\theta_1 - d_y \ \sin\theta_1 \\ y_2 = y_1 + d_x \ \sin\theta_1 + d_y \ \cos\theta_1 \end{cases}$$

The orientation, on the other hand, was taken directly from the message indicating the orientation of robot 1, as they must be the same. The position of robot 2 was calculated in the local reference system because every feedforward control system is in the local reference system.

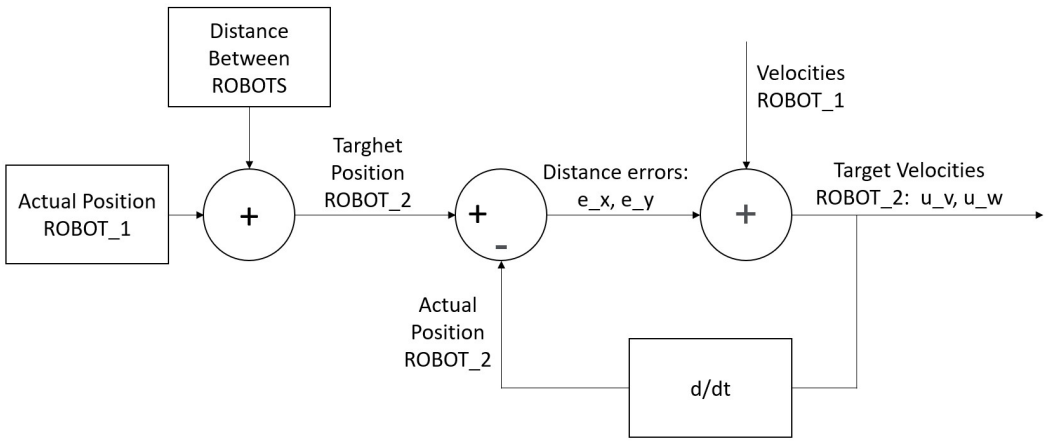
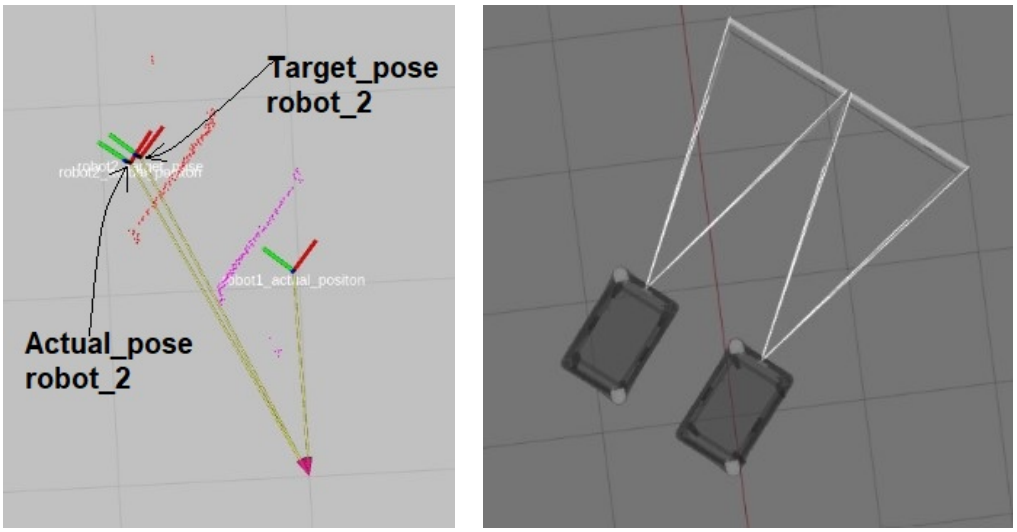


Figure 3.6: Closed loop control diagram.

Once the theoretical position was calculated, the deviation between this and the actual position was calculated in order to determine the errors e_x, e_y. The target velocity of robot 2 was calculated as in section 3.1.3. It depends on the current position of the two robots and the speed of robot 1. By entering all the parameters in the closed-loop control system function, we are able to obtain as output a linear and angular velocity that will be given to robot 2, which will be retroactive because it will be continuously updated and modified according to the two errors of distance and orientation in the two directions and will therefore adjust the robot's velocity until it reaches the desired theoretical velocity. To better understand, in the figure below we can see a deviation between the target position and the current position, which will result in an increase in robot2's linear speed until the desired position is



(a) *R*viz interface with actual and target frames.

(b) *G*azebo interface of two MIR robots moving.

Figure 3.7: Two MIR robots synchronised in a simulation environment.

reached.

In order to display the actual deviation of the second robot from the desired trajectory, the use of Gazebo was not sufficient; therefore, an additional parameter was added to the function concerning the current and target position of robot 2, which allows us to display in RVIZ the terns of axes that show us exactly the position and orientation of the function to which it has been applied. They are visible in the Figure 3.7a in green and red colour

3.3 LiDAR Sensor- amcl localization setup

The laser mounted on the Scout Mini Rover is produced by the SLAMTEC and it is the RPLidar A3 model. This laser is capable of scanning both indoor and outdoor environments. It is very thin, has a footprint of only 4 cm, and is silent as it uses brushless motors. It can scan distant objects up to 25 metres away and has a scanning range of 360°. RPLidar A3 adopts laser triangulation ranging principle, and with high-speed RPVision ranging engine,it measures distance data 16000 times per second and keeps its excellent performance in a long distance.

This laser was used to obtain the actual position information of the follower robot from the leader robot, a different approach which does not use amcl topic, and which can be compared with the latter. To get orientation and position in the x-direction, the acquisition of laser data was limited to a certain angular range so that only points in that range were acquired. The laser provides only one value for each point. This value is equal to the distance in polar co-ordinates between the point and the laser (radius). To know the angle associated with each point, the position in the data

(a) 360 Degree Omnidirectional Laser Range Scanning.

(b) Laser triangulation ranging principle.

Figure 3.8: RPLidar A3 illustrations [6]

acquisition range was taken into account. To explain it better, the laser rotates, and in a turn of 360° it acquires 1947 points and measures their distance. This means that every 0.184 degrees it takes a value from the objects around it. So knowing the position of the acquired value in the range 0-1947 we can easily derive the angle by multiplying this value with 0.184. Now knowing the angle and distance, it is possible to transform these values into cartesian coordinates, obtaining the x and y of each point. From those points, a linear regression was performed to obtain the line that best approximates the position of the points and thus its inclination. In addition to this through the RPLidar it is possible to measure the average distance of all points in this range in the x direction by multiplying the radius by the cosine of the angle of that point. It is possible to repeat the same steps to obtain a distance in the y direction without the necessity of doing a linear regression.

Scout Mini Rover has an irregular shape, so to be able to use such a system it would be necessary to create rigid, flat surfaces to be mounted on top of them, at the height of the laser so as to have more accurate data (see Figure 3.9).

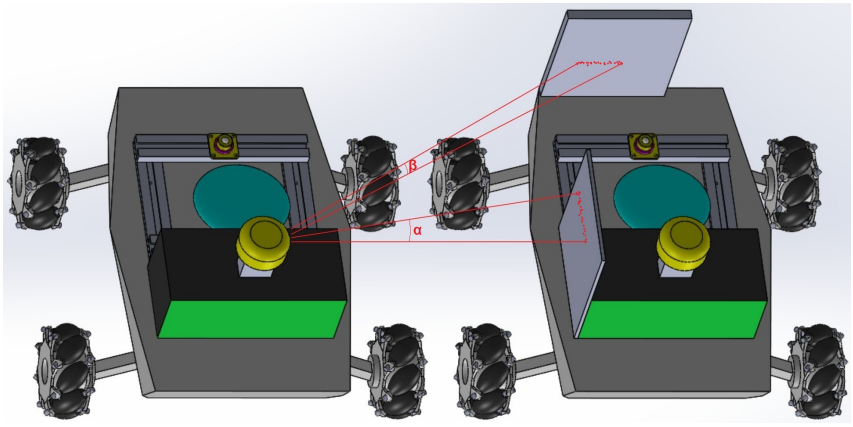


Figure 3.9: Illustration to get the actual position via RPLidar A3.

To perform linear regression, instead of creating a code with all the formulation to search for the line that minimises the sum of the deviations between the estimated and observed values, also known as regression residuals, and then defining for each

point:

$$\hat{y}_i = a + b\ x_i\ \forall i$$

the difference between the actual and predicted data is:

$$y_i = \hat{y}_i + \epsilon_i$$

so an error is made for each observation:

$$\epsilon_i = y_i - \hat{y}_i$$

The line that minimises the deviation will be:

$$\sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - a - b\ x_i)^2 = min$$

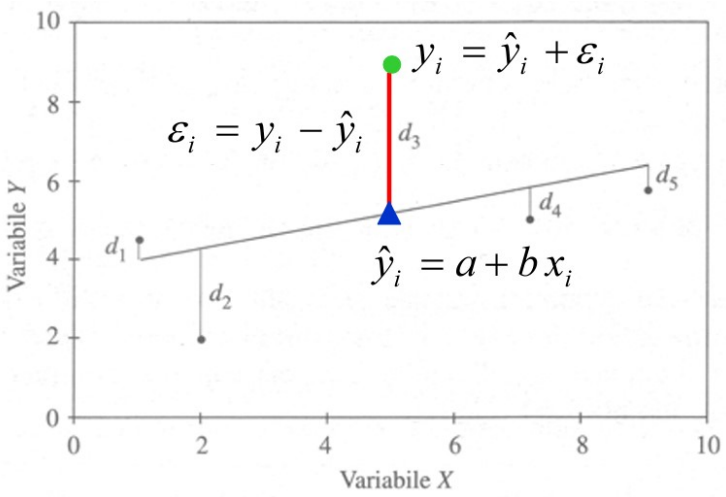


Figure 3.10: Simple Linear Regression Diagram.

it was preferred to use a commonly used Python package for this type of need. The package scikit-learn is a widely used Python library for machine learning, built on top of NumPy and some other packages. It provides the means for preprocessing data, reducing dimensionality, implementing regression, classifying, clustering, and more. Like NumPy, scikit-learn is also open-source. The system needs to be given the x- and y-coordinates of the points to be studied in the form shown in Figure 3.11, obtaining as results, via the "LinearRegression()" function, the coefficient of determination and the slope coefficient. With these two parameters we can understand how far the approximated line deviates from the values with which it was constructed, and through the angular coefficient we know the orientation in radians of the robot follower.

```
Python                                                                    >>>

>>> x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
>>> y = np.array([5, 20, 14, 32, 22, 38])
```

Figure 3.11: Input Scikit-learn package for Linear Regression.

3.4 Mapping the Environment

The first essential feature for navigation is the map. Like a navigation system, a robot needs a map, so we need to create a map and give it to the robot, or the robot should be able to create a map by itself. For this thesis project the map was created by the robot via the RPLidar on it. The Lidar can measure distance to the obstacle on the XY plane and a map can be created using the AMCL (Adaptive Monte Carlo Localization) aproach. AMCL is the upgrade of Monte Carlo Localization aproach (MCL).

When the robot moves, MCL generates N new samples that approximate the robot's position after the motion command. This high number of samples is also called particle filter that consists on three steps:

1. **Re-sampling:** Draw with replacement a random sample from the sample set according to the (discrete) distribution defined through the importance weights. This sample can be seen as an instance of the belief.

2. **Sampling:** Use previous belief and the control information to sample from the distribution which describes the dynamics of the system. The current belief now represents the density given by the product of distribution and an instance of the previous belief. This density is the proposal distribution used in the next step

3. **Importance sampling:** Weight the sample by the importance weight, the likelihood of the sample X given the measurement Z [18].

As you get additional measurements, you predict and update your measurements which makes your robot have a multi-modal posterior distribution. The number of samples required to achieve a certain level of accuracy varies drastically. During position tracking, on the other hand, the uncertainty is typically small and often focused on lower dimensional manifolds. Thus, many more samples are needed during global localization to accurately approximate the true density, than are needed for position tracking. An often criticized limitation of plain particle filters is their poor performance in higher dimensional spaces. This is because the number of particles needed to populate a state space scales exponentially with the dimension of the state space. That is where Adaptive feature comes to solve this issue. The AMCL adjusts the number of particles in the filter: when the robot's pose is highly uncertain, the number of particles is increased; when the robot's pose is well determined, the

number of particles is decreased. This enables the robot to make a trade-off between processing speed and localization accuracy[19]. Even though the AMCL package works fine out of the box, there are various parameters which one can tune based on their knowledge of the platform and sensors being used.

In the study of this thesis the amcl package has been modified to get a better result during the motion of the robot Scout Mini Rover setting the parameters of the RPLidar the metodolody used is the try and error one. A map is created and the AMCL package is now configured to get better results while the robot moves (see Figure 3.12). They may be useful to future application, such as autonomus navigation.

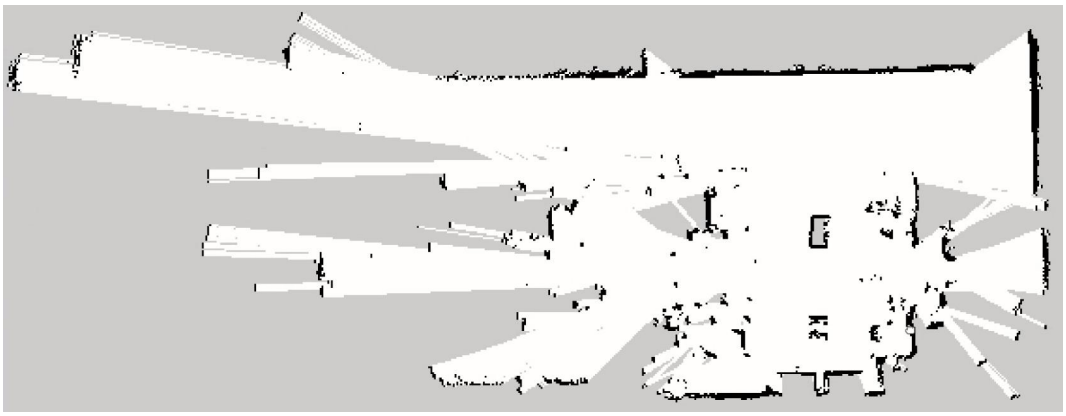


Figure 3.12: Map created with RPLidar A3 in MATCH department.

Chapter 4

Platform Design

This chapter will deal with the connection of the robots by describing the structure that has been implemented and prototyped. Besides being synchronised, the two robots will have to have a platform that connects them. This platform will have to be strong enough to support a static load equal to the sum of the load supported by each individual robot. Furthermore, it will have to prevent the two robots from hindering their motion. This is because the motors of the Scout Mini Rover robots are very fragile and an unplanned displacement could damage them. These displacements are due to the way we track the robot and the mechanics of the latter. As explained in the paragraph 1.2.3, wheel contact is discontinuous and there could be slippage due to ground-rollers contact.

Differents tools and methodologies were used to design this plate. First of all, the 3D modelling software used was SolidWorks, and all simulations were performed with SolidWorks Simulation. The idea was to create soft components that mounted on the plate would allow free movement between the robots, while still giving the structure some stability. 3D printing of soft material was used to observe the behaviour of the geometries assumed for the soft component, while FDM was used to create supporting components useful for assembling the structure.

Technical Specifications

It is good to specify the characteristics that our plate must have:

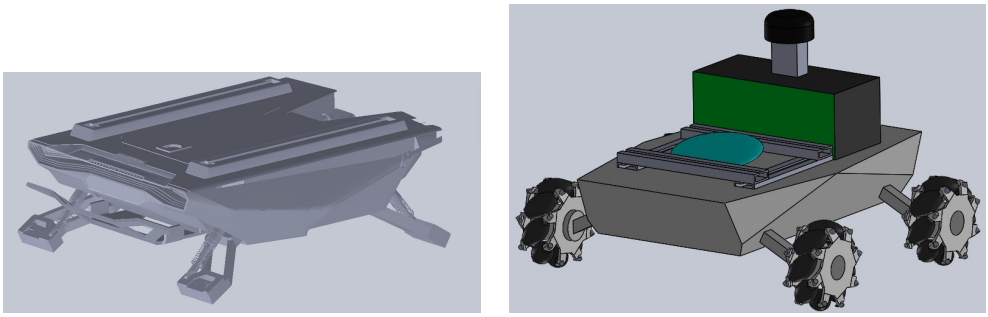
- It must be able to support a minimum weight of 100kg;
- It must allow for possible movement at the mounting points on the robots, an estimated 10 cm of movement.
- It must not hinder the cushioning of the robots following loads.
- It shall accommodate a standing person or another robot.
- It must not obstruct the movement of the robots by colliding with electrical equipment or surrounding objects.

4.1 SolidWorks

Solidworks is a 3D drawing tool with a variety of functionalities. A variety of objects with the strangest shapes can be drawn in a simulative environment. The software can be used for both solid and surface modelling. In this study, it was used for the evaluation of overall dimensions and for the design of robot connection plate components.

A 3D model of the chassis of the Scout Mini Rover robot without wheels can be find in the AgileX library on github. This model, however, in Solidworks is not seen as a single model but as many small surfaces and hundreds of objects (see Figure 4.1a). Due to the large number of components, it is therefore a large drawing file, making it very difficult to work on it with normal performance laptops.

Being a step file, it is not possible to have reference points and it was difficult to position the omni-directional wheels at the exact point of the end of the dedicated supports. Therefore, it was easier to create a rough three-dimensional model of the Scout Mini Rover robot, taking into account the overall dimensions and electronic components (see Figure 4.1b).



(a) *S*cout Mini Rover real model. (b) *S*cout Mini Rover rough 3D model.

Figure 4.1: Three dimensional Scout Mini Rover Models

To connect the robot to the platform, commercial components were used, which come from conveyor rollers for transporting industrial pallets. They are not actually rollers but spheres, and they are also called load-bearing spheres. These consist of a main ball that touches the platform, which is enclosed in a structure with other balls to reduce friction on the main one (see Figure 4.2). This technology provides good mechanical characteristics, such as a high load bearing capacity (80 kg load bearing capacity)[20] and allows us to move the robot underneath, with negligible platform movement.

The platform, however, supported with three load-bearing spheres, mounted only in the robot structure, would move for any slightest disturbance. This must be avoided. Therefore it was decided to use components made of soft material, which would not disturb the movement of the robots underneath but give stability to the structure.

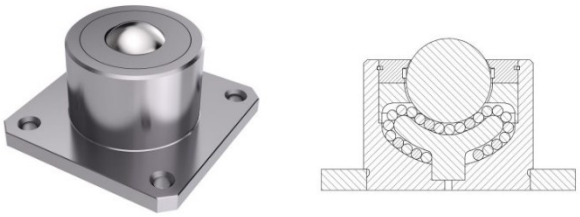
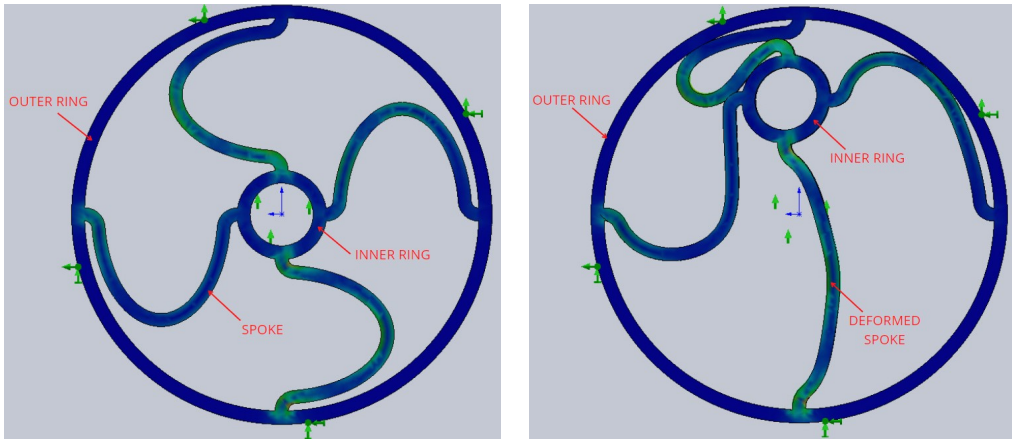


Figure 4.2: Heavy-duty ball caster with base flange (load-bearing spheres).



(a) Horizontal Wheel in resting position. (b) *D*eformed Horizontal Wheel

Figure 4.3: Horizontal Wheel movement.

The final shape of the spokes between the inner and outer rim was derived from several try and error tests with the help of the solidworks simulation environment and the 3D printer for soft materials. The only trick that we took in consideration was that the lenth of the spoke had to be at least long as the outer rim diameter less the inner rim diameter. The first two wheel prototypes were 3D printed on a reduced scale, due to the available printing surface of the printer. From these prototypes, we should have seen whether the geometry given to the spokes was suitable for bringing the inner rim closer to the outer rim. However, the material of the 3D printer was soft but not durable and by the second or third application, cracks were occurring (see highlighted parts in Figure 4.4). With this drawback and the fact that in any case the component created was of reduced dimensions, we moved on to others design approaches: the simulative one to verify the conformity of geometries and that of a moulding design to cast the material in a 3D printable mould and realised the *horizontal wheel* with the actual dimensions.

(a) First prototype of *Horizontal wheel*. (b) Second prototype of *Horizontal wheel*.

Figure 4.4: *Horizontal wheel* printed in a soft material.

4.2 Simulation Tests

Using SolidWork simulation, the displacements of the inner cylinder of the various *Horizontal wheels* created were simulated and it was seen which points were exposed to high stress that would trigger a breakage in the course of time.

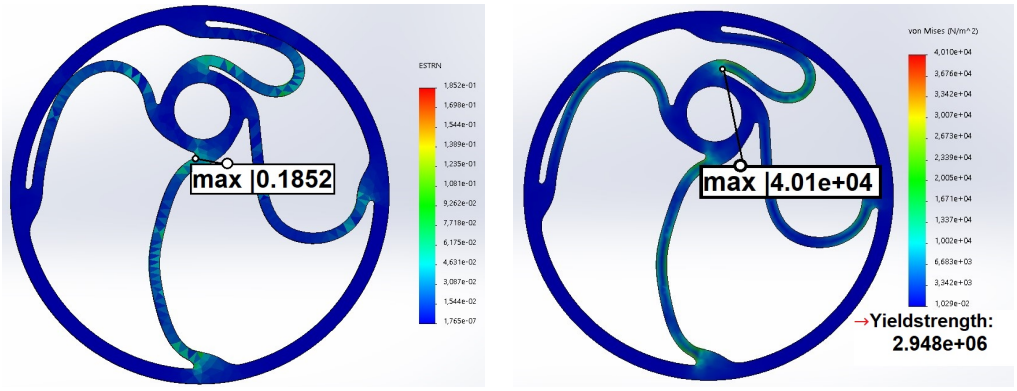
To do this, it was necessary to define the material. Therefore, a silicone material called Dragonskin 30 was chosen. This is a super-strong and flexible soft material suitable for our purpose, but it requires special precautions to make it. To realise the *Horizontal wheel*, once the male and female mould had been printed using the FDM 3D printer, the silicone material was prepared.

Composed of two liquid parts, these are placed and mixed in a container in equal quantities. The container is then placed in a depressurised environment, until the pressure of 0.1 [bar] is reached. Reaching this depression prevents the creation of bubbles inside the mould during solidification. The compound will then be poured into the previously lubricated mould and after a certain period of time it will solidify, giving the object the desired characteristics. After this step, at the end of the cure time, the compound will have solidified and the removal of the burrs (the excess material on the mould) will take place. Finally two horizontal wheel were made, one for hosting a single load-bearing ball and the second one for hosting two of them (see Figure 4.5).

As far as simulations are concerned, one of the first prototypes of the *Horizontal wheel* was simulated. In the Figure 4.3, its geometry can be seen and it can be deduced that the part of the spoke attached to the outer ring is subjected to excessive stress, which was then verified by the simulation values. Therefore, the prototype has been modified to the final version, where it does not have any parts subjected to stresses greater than the breaking stress because these parts have been appropriately resized (see Figure 4.6). The simulation, however, shows us that the displacement

Figure 4.5: *Horizontal wheel* casted with Dragon-skin 30.

of the inner ring is not as great as in the previous case, and that there is a section shrinkage of about 2 tenths of a millimetre, which is quite acceptable.



(a) *S*train simulation. (b) *S*tress simulation.

Figure 4.6: Final prototype of *Horizontal wheel*.

4.3 Prototype

The designed prototype of the platform consists of a large plate that supports the robots in two points by means of load-bearing spheres and in one point on the ground by means of a cushioned support that also has a load-bearing sphere in its extremity. The designed plate is assumed to be made of aluminium and will house all the components to be connected to the robots. In this plate, the stress resistance indicated in the specifications was verified with a FEM simulation, which showed that there are no stresses that lead the material to cracking, and the maximum displacements that occur are less than 0.7 mm, which is completely acceptable and

remediable. If more stability is required, an additional cushioned support can be added to the front. The virtual prototype can be seen in the Figure 4.7.

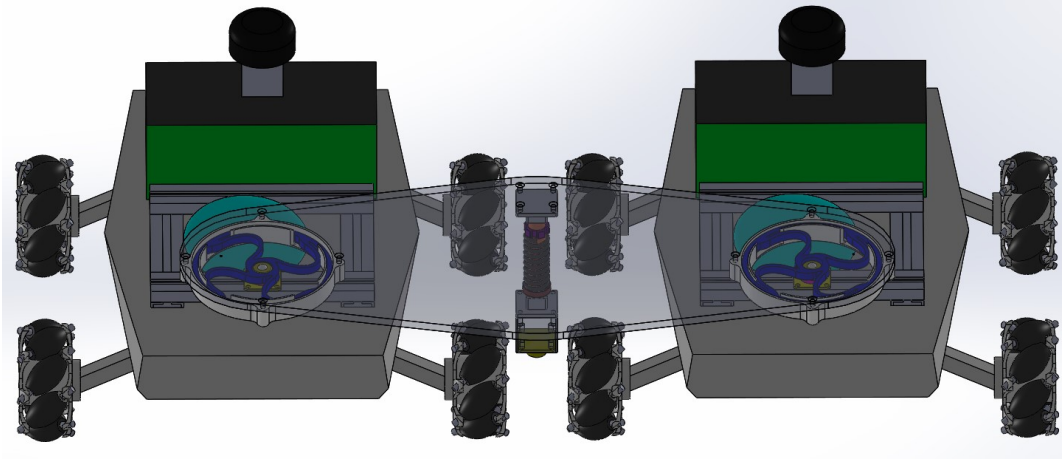
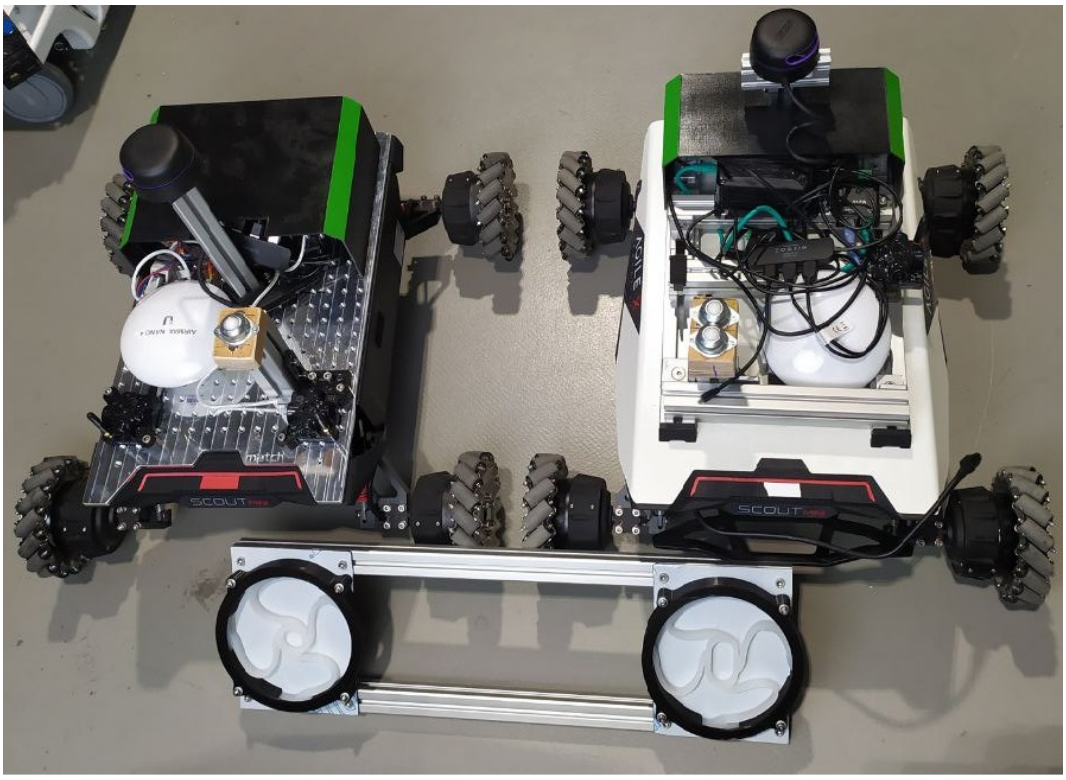


Figure 4.7: Virtual prototyope.

The first real prototype was made with the materials available in the MATCH department's warehouse and the FDM 3D printer. Two aluminium profiles and two plates are the basic components of the frame. In the aluminium plate, the support of the horizontal wheel is assembled and the two profiles attached. An initial version consisted of supporting the platform with three load-bearing spheres, all positioned on the robots.

In this case, it is not necessary to have an intermediate support point between the two robots so the support with the suspension damper that rest on the ground it would not be necessary anymore (see Figure 4.8).

On the other hand, the realisation of this prototype showed that the structure is unstable except for minimal movements. Therefore, a version with an intermediate support, similar to the one designed, but without a suspension damper, was chosen. The shock absorber was not included due to the timing of the suppliers. Although not satisfying a technical specification, the inserted support made it possible to visualise the behaviour of the plate to the movement of the robots.



(a) *P*latform supports on the robots, consisting of 3 supporting spheres.

(b) *A*ssembled platform without intermediate support.

Figure 4.8: Platform prototype without intermediate support.

Chapter 5

Results

As far as the robot positioning control system is concerned, this is very much influenced by the errors encountered during movement due to all the sensors (IMU and encoders) as well as the drift in odometry caused by the slip of the robot's wheels. This is why the closed-loop control system with retroaction was necessary to solve this type of problem. By acquiring positions using the Adaptive Monte Carlo Localisation method, the system will update the position error and correct the speed of the robot follower, allowing the target position to be reached in the shortest possible time.

On the other hand, the study was performed in a simulation environment where sensor-related problems are minimal, and mechanics-related problems such as wheel slippage do not occur. It is therefore difficult to distinguish in Gazebo or Rviz the type of code launched between the system with feedforward controller and the one that controls the robot follower with only kinematic formulation. It will therefore be necessary to verify the correctness of the codes in a real environment. One problem with the chosen formation control however remains. If for some reason the leader robot fails unexpectedly, for example, the follower robot will wait for it, whereas if the follower robot fails unexpectedly, the leader robot will continue its motion and damage the linking structure and the robots themselves.

The RPLidar laser localisation system of the Scout Mini Rover robot was useful for comparing the data acquired with the AMCL methodology in order to have a more truthful and less noise-dependent data. On the other hand, it will be necessary to arrange the robots of flat surfaces at the height of the laser. Therefore the current arrangement of the instrumentation will have to be changed, perhaps by adding a new laser in a lower position used only for this purpose. On the other hand, this type of data acquisition can be used with the MIR200 robots because they already have a parallelepiped shape. With the lasers placed in the back corner, it is possible to measure the distance in the x-direction and the orientation, while with the one in the front (and by analogy also the one in the back), it is possible to see whether the robot follower is going too fast or too slow, by acquiring on-off data. When, for example, MIR2 slightly overtakes the front of the leader robot, the front laser will perceive a drop in the current distance, which means that robot 2 is advancing too fast.

The platform connecting the robots is useful in preventing any movements or misalignments during the movement of the structure from causing the motors to strain and the robots to drag during their movement. The omission of certain components, such as the plate and the use of profiles to compensate this deficiency, actually proved to be a better alternative for prototypes like this. In fact, angle brackets can be fitted to the profiles, to which other profiles can be fitted, to create a more complex structure, without creating holes or anything else. This is precisely how the intermediate support was installed.

The use of wood was essential for the assembly of the load-bearing spheres, which could be positioned in any direction by drilling the wood directly with the screws. The two robots have different fairings, the most recent one, although not operative due to the breakage of a motor, has a perforated plate that allows multiple components to be mounted on it. However, it is lower than the original, so a wooden shim was installed to even out the difference in height in the support for the load-bearing spheres. In addition, the laser in this latest robot has been placed in the centre and hinders the movement of the plate. The latter should be positioned above all the components, but a profile that is too high could lead to problems with tip loading, so it would be better to review the arrangement of the electronic components by lowering them as much as possible while checking that the lasers still work at maximum performance. Furthermore, the plates used to rest on the load-bearing spheres are only coated with aluminium, they have a plastic material inside. Simply pressing the structure with the body weight will result in grooves that would not occur with an all-aluminium plate as in the virtual prototype.

The simulations for the *horizontal wheel* were useful but not totally. Those of the final *horizontal wheel* show that the inner ring does not reach the outer one with an imposed displacement of 50 mm. When the object is finished, however, the displacements of the inner ring are as desired (they reach the outer ring). The shape of the spokes, however, creates resistances in certain directions, resistances which are negligible because the mere weight of the robots counteracts them. The possible movement of the plate with the *horizontal wheels* is by a wide margin. Tests were carried out by moving only one robot, and it was able to do this without disturbing the stationary robot in the various directions, for about 10-15 cm if the robot rotates, it can make a rotation of about 180° before affecting the movement of the other robot.

The sudden stop of the robot's movement, however, causes an oscillation of the platform, which is dampened slightly by the *horizontal wheels*. It also allowed us to understand how to position this component so that it could move freely in space, guaranteeing all possible movements that a holonomic robot system could perform.

5.1 Conclusion

Analysing all results, it can be seen that with this thesis work, a first prototype structure for handling heavy objects in confined spaces has been developed.

The analysis of the state of the art analysed the use of platforms moved by holonomous robots, highlighting the main advantages and disadvantages. The type of formation control chosen was the leader-follower type. Programming training and the study of kinematics was essential for writing the control code. In this case, the lack of possibility of having fully functioning holonomous robots diverted the study of the platform towards a system controlled by two non-holonomous robots. This made the research more interesting from the point of view of kinematics, and considering and studying a more complex system will be easier to implement in holonomous robots in the future. The handling of the robots was implemented with a closed-loop control system with feedback so as to improve the synchronisation of the two robots. A further method for tracking the robot follower was developed, albeit with considerable disadvantages in terms of construction, but with the enormous advantage of simplicity. The structure if implemented with a cushioned support will be able to support a maximum load of 100 kg. The platform will allow possible asynchronous movements without impacting the robot components.

The mapping of the working environment in the MATCH department provides a map of a real environment, from which the Scout Mini Rover robots can locate themselves. As there was no second robot available, however, it was not possible to apply the codes created in this mode. Moreover, even if we wanted to verify the behaviour of the two MIR 200 robots in reality, the environment would have been too tight to move them without the risk of robots' collisions with objects placed in the laboratory. To understand the size of these robots in the mapped environment in Figure 3.12, we can see a well-defined rectangle in the centre of the room, this is a MIR200 robot that was placed there to carry out other research purposes. Considering that all the black dots are obstacles, it is evident that the synchronised movement of two parallel rectangles of this size, which moreover have no holonomic movement, is complex and dangerous.

The first prototype of the platform to connect the two robots provided the necessary indications to improve it, focusing on the most critical aspects, oscillation, damping, torque damping.

5.2 Future Implementations

This thesis project can be implemented in the future in several ways.

Firstly, if the robots remain two, it is necessary to have a joint that absorbs the torque and a support that has a damper. Another robot could be added and solve these problems. The formation control could remain the same or change, adopting more complex ones.

The robot follower tracking system using lasers can also be reused, but solving the problems described above. Nowadays, RPLidar lasers can recognise robots by their shape and deduce their position, so this method can also be implemented even if the complexity will increase.

The platform will have to have at least rigid plates in contact with the supporting spheres so as not to cause grooves and direct the movement in particular directions. If larger spaces are available, real robots can be tested and the methodology for mapping the environment can be replicated. An autonomous navigation system could also be implemented to the system, capable of reaching a given target position without colliding with surrounding objects, thus following the best path. Finally, this structure could be implemented to transport objects or people in warehouses or to support an anthropomorphic robot that will perform various tasks in a collaborative robot perspective.

Bibliography

[1] Luigi Villani Bruno Siciliano, Lorenzo Sciavicco and Giuseppe Oriolo. *Robotics modelling, planning and control.*, volume cit. on pp. 5-9, 15-18, 21. Springer Science Business Media, 2010.

[2] Daniele Costa Antonio Pinellia David Scaradozzi, Giacomo Palmieri. Bcf swimming locomotion for autonomous underwater robots: a review and a novel solution to improve control and efficiency. *Ocean Engineering*, vol. 130:pages 437–453, (cit. on page 448), 2017.

[3] Stüde M. Wielitzka M. Ortmaier T. Raatz A. Recker T., Zhou B. Lidar-based localization for formation control of multi-robot systems. *Springer, Cham*, page 363–372, 2022.

[4] Glen Bright Johan Potgeieter Sylvester Tlale Olaf Diegel, Aparna BAdve. Improved mecanum wheel design fro omni-directional robots. http://ftp.imp.fu-berlin.de/pub/Rojas/omniwheel/Diegel-Badve-Bright-Potgieter-Tlale.pdf, 2002.

[5] Leon Jung Darby Lim Yoonseok Pyo, Hancheol Cho. *ROS Robot Programming (English).* ROBOTIS, 12 2017.

[6] Shanghai Slamtec Co. Rplidar a3. 2022.

[7] SPYROSG. TZAFESTAS JUNTANG KEIGO WATANABE, YAMATO SHIRAISHI and TOSHIO FUKUDA. Feedback control of an omnidirectional autonomous platform for mobile service robots. *JournalofIntelligent and Robotic Systems*, page 315–330.

[8] Annika Raatz Tobias Recker, Malte Heinrich. A comparison of different approaches for formation control of nonholonomic mobile robots regarding object transport. *Elsevier B.V.*, pages 248–253, 2021.

[9] Abdelouahab Bazoula M. S. Djouadi Hichem Maaref. Formation control of multi-robots via fuzzy logic technique. *International Journal of Computers, Communications Control (IJCCC) III*, pages 179–184, 2008.

[10] Kar-Han Tan and M. A. Lewis. Virtual structures for high-precision cooperative mobile robotic control. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1, 1996.

[11] Magnus Egerstedt and Xiaoming Hu. Formation constrained multi-agent control. *IEEE Transactions on Robotics and Automation*, pages 947 – 951, 2002.

[12] A. Bazoula and Hichem Maaref. Fuzzy separation bearing control for mobile robots formation. *International Journal of Mechanical and Mechatronics Engineering*, 1, 2007.

[13] LiGao Pingxia Zhang and Yongqiang Zhu. Study on control schemes of flexible steering system of a multi-axle all-wheel-steering robot. *Advances in Mechanical Engineering*, 8(6):1–13, 2016.

[14] R.P.A van Haendel. Design of an omnidirectional universal mobile platform. *Technische Universiteit Eindhoven*, 2005.117:235–241, 2005.

[15] V. Spinu I. Doroftei, V. Grosu. *Studies in large plastic flow and fracture*, volume pp. 511-529. Advanced Robotic Systems International (Vienna) and ITech, 2007.

[16] Nurallah Ghaeminezhad Hamid Taheri, Bing Qiao. Kinematic model of a four mecanum wheeled mobile robot. *International Journal of Computer Applications*, Volume 113(3), 2015.

[17] AgileX Robotics(Shenzhen) Ltd. Scout mini. https://global.agilex.ai/products/scout-mini, 2021.

[18] Jekyll Minimal Mistakes. Particle filters in robotics. *Robotics Knowledgebase*, 2002.

[19] Sebastian Thrun. Particle filters in robotics. *Robotics Knowledgebase*, 2020.

[20] Schulz Stanztechnik GmbH. Heavy-duty ball caster with base flange. 2022.