university of groningen / faculty of science and engineering

MASTER'S THESIS

Cooperative distributed pick and place algorithm for mobile manipulators with camera feedback.

INDUSTRIAL ENGINEERING AND MANAGEMENT

IEM

June 28, 2019

Author:
M. (Marc) Capó Fuster
S3905772

Supervisors:
prof. dr. ir. M. (Ming) Cao
prof. dr. ir. B. (Bayu) Jayawardhana

Abstract

The main goal of this research is to perform a pick and place task done by several mobile robots equipped with a robotic arm by using a distributed algorithm and a formation control law for keeping the shape. Nowadays, the industry sometimes needs the cooperation of different robots to achieve what cannot be reached by a single robot. Sometimes, the fact of using only a single robot can be either really expensive or not powerful enough and it is worth to implement a system with several agents.

The studied case will be with four agents and it will be tested experimentally with the mobile nexus robots which are in the *DTPA* lab. The fact of being a task performed by several agents means that formation control theory will be taken into account, specifically the formation control Law designed by Garcia de Marina, Jayawardhana, and Cao [1]. Several studies and tests related to formation control have been performed in the PhD Nexus Group.

Furthermore, some research about pick and place task has also been studied but only for a single robot. Because of this reason, the aim of this research is to extrapolate the results of the pick and place task done for one robot to a formation of several agents by using the formation control law previously specified. Moreover once the algorithm is tested in a four agent formation is relatively easy to change the number of agents by simply modifying some parameters.

Challenges such as recognition of objects (Markers), tracking them (PI control) and keeping the formation of the agents (formation control theory) will be achieved by using the suitable sensors. For instance, a camera mounted on the robotic arm will be used for the recognition of objects, while a RPLidar Laser Scanner will be used for measuring the distances between robots and ensure that they keep the formation.

Acknowledgements

First of all I would like to thank prof. dr. Ming Cao and prof. dr. Bayu Jayawardhana for being my supervisors of this research and for the helpful meetings we did with the PhD Nexus Group.

I would also like to thank Carlo Cenedese for always being there for answering my theory questions and give me good advices about how to face up my research and the problems I had.

Simon Busman helped me with all the settings related to the Nexus Robots, showed me the basics of ROS and guided me on how to continue studying ROS. Furthermore, he also solved every technical problem I had, spending a huge amount of time. Martin Stokroos and Sietse Achterop also helped to ensure all the settings were the right ones.

Finally, I would like to thank my family and girlfriend for always being there and supporting me whenever I needed it.

Contents

Nomenclature

Abbreviations

AGV	Automated Guided Vehicles
CMOS	Complementary Metal–Oxide–Semiconductor
DoF	Degrees of Freedom
DTPA lab	Discrete Technology and Production Automation Laboratory
FPS	Frames Per Second
MAS	Multi-Agent System
MP	MegaPixel
PI control	Proportional and Integral control
PID control	Proportional Integral and Derivative control
ROS	Ros Operating System
RUG	RijksUniversiteit Groningen

Variables

Frame Corrections and Denavit-Hartenberg Parameters

θ_i	Joint angle
α_{i-1}	Link twist
a_{i-1}	Link length
d_i	Link offset
${}^{i-1}_{i}T$	Transformation relation between one frame (i-1) to another frame (i)
${}^{0}_{N}R$	Rotation matrix from frame (N) to frame (0) formed by the components r_{ij}
${}^{N}P_A$	Position vector A expressed in frame (N) formed by the components p_x, p_y and p_z

Formation Control

p_i	Position of an agent
$\mathbb{G}$	Indirect graph that describes the neighbor relationships
$\mathcal{V}$	Vertex set
$\mathcal{E}$	Ordered edge set
$\mathcal{N}_i$	Set of the neighbors of agent i
B	Incidence matrix
z	Vector of the sensed relative distances
$f_{\mathbb{G}}(p)$	Edge function
$R(z)$	Jacobian
$\mathcal{Z}$	Set of the possible formations
$\mathcal{R}$	Set of the rotational matrices
$\mathcal{D}$	Set of the resulting formations
u	Stacked vector of control inputs

$V(z)$	Potential function
e_k	Distance error for edge k
D_z	Block diagonal matrix of z
$D_{\tilde{z}}$	Block diagonal matrix of $\tilde{z}$
O_g	Global frame of coordinates
O_i	Local frame of coordinates for agent i
O_b	Body frame
p_c	Centroid of the body frame
μ_k	Motion parameter
$\tilde{\mu}_k$	Motion parameter
c	Gain for scaling the motion control parameters
A	Motion control matrix
w_b	Angular velocity of the rigid formation
μ_v	Motion parameter to assign translational velocity
μ_w	Motion parameter to assign rotational velocity

Approach Control

r	Reference signal
u	Control signal
e	Error signal
k_p	Proportional gain
k_i	Integral gain
k_i	Derivative gain
d_x	Distance to the tag in the x-direction
d_y	Distance to the tag in the y-direction
θ	Orientation angle between the formation and the tag
p	Output position and orientation of the nexus
ϕ	Vector of the angle values of joints 2, 3 and 5.

Centering of the tag and gravity compensation

e_{cx}	Horizontal error from the center of the camera frame and the one of the apriltag
e_{cy}	Vertical error from the center of the camera frame and the one of the apriltag
α_x	Position in x seen frome the camera
α_y	Position in y seen frome the camera
α_z	Position in z seen frome the camera
ϕ	Vector of the angle values of joints 2, 3 and 5.
p	Output position and orientation of the nexus
$\tilde{\phi}^*$	Vector of desired inputs of the angle values of joints 2, 3 and 5 for the servos.

1 Introduction

Over the last few decade, formation control algorithms for the control of multi-agent systems have received a lot of attention from the control community [2], [3]. The single, heavily equipped vehicle may require considerable power to operate. Compared to a single agent, a group of networked agents has the advantages of being flexible, redundant and fault tolerant [4] and as such can be employed to perform complex tasks [5] in a centralized or in a distributed manner. For instance, many coordinated robot tasks, such as enclosing a target [6], area exploration and surveillance [7], and vehicle platooning for energy efficiency [8], can be achieved by combining two different cooperative controls: multi-agent formation control and group motion control.

The main goal of this research is to perform a pick and place task done by several mobile robots equipped with a robotic arm by using a distributed algorithm and a formation control law for keeping the shape. Moreover, the project is based on the algorithm designed by Garcia de Marina, Jayawardhana, and Cao [1].

1.1 Research Background

In this section, a literature overview and a brief introduction of the main topics used in this thesis are proposed.

1.1.1 Multi-agent systems (MAS) and formation control

A multi-agent system (MAS) is a system composed of multiple interacting agents. In most of the cases the agents are considered intelligent and equipped with different sensors that allow them to interact with the environment. Multi-agent systems can be used to solve problems that are difficult or impossible for an individual agent to solve. The number of applications is wide, highlighting among others robotics, distributed computation or security.

According to Ferber (1999), "an agent can be a physical or virtual entity that can act, perceive its environment (in a partial way) and communicate with others, is autonomous and has skills to achieve its goals and tendencies. It is in a MAS that contains an environment, objects and agents (the agents being the only ones to act), relations between all the entities and a set of operations that can be performed by the entities" [9].

Within the topics related to MAS, formation control is one of the most studied. Formation control of MAS has obtained intensive focus due to its widely adaptability to solve different problems, e.g., in robotics, astronautics or aeronautics [10], [11]. A common objective in formation control is to maintain a desired (rigid) shape while moving the formation in a given direction. For instance, this objective is at issue when multiple mobile robots transport (heavy) items or when aircrafts are flying in a V-shape in order to save fuel [12].

(a) Formation control of autonomous drones.

(b) Formation control of aircrafts in order to save fuel.

Figure 1.1: Examples of formation control

1.1.2 Robotic manipulator

The mechanical structure of a robot manipulator consists of a sequence of rigid bodies (links) interconnected by means of articulations (joints) [13]; a manipulator is characterized by an arm that ensures mobility, a wrist that confers dexterity, and an end-effector, usually composed by a gripper, that performs the task required of the robot.

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 [1], 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.

The workspace is the set of points in the space which the manipulator's end-effector can access. Its shape and volume depend on the manipulator structure as well as on the presence of mechanical joint limits.

The degrees of freedom should be properly distributed along the mechanical structure in order to have a sufficient number to execute a given task. In the most general case of a task consisting of arbitrarily positioning and orienting an object in three-dimensional (3D) space, six DOFs are required, three for positioning a point on the object and three for orienting the object with respect to a reference coordinate frame. In the case that the number of actuators is less than the number of DOFs, the arm is under actuated. Otherwise, the arm is called fully actuated if it can reach all the positions in the workspace with an arbitrary orientation.

1.1.3 Automated guided vehicles (AGV)

An automated guided vehicle (AGV) is a driverless transport system used for planar movement of materials. Their use has grown enormously since their introduction, since the number of areas of application and variation in types has increased significantly. AGVs can be used in inside and outside environments, such as manufacturing, distribution, transshipment and (external) transportation areas [14]. At manufacturing areas, AGVs are used to transport all types of materials related to the manufacturing process.

An AGV follows along marked long lines or wires on the floor, or uses radio waves, vision cameras, magnets, or lasers for navigation. They are most often used in industrial applications to transport heavy materials around a large industrial building, such as a

[1] An open kinetic chain is defined as "a combination of successively arranged joints in which the terminal segments can move freely".

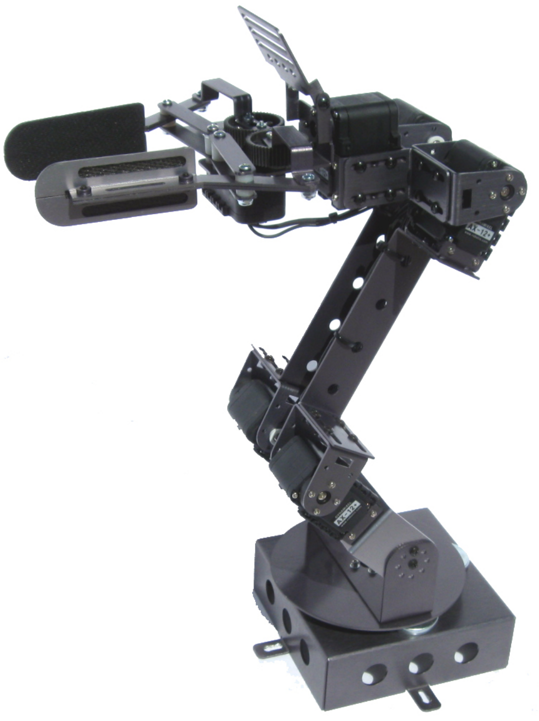


Figure 1.2: AX-18 Smart Industrial Robotic Arm used in this project.

factory or warehouse. In the case of this research, the AGV will use a vision camera to follow a marker.

Figure 1.3: Kiva robots used in amazon's warehouses [15].

1.2 Research Design

A research design (or research strategy) describes how the investigation will be approached for the dissertation. Thus, the methodology, main goal and research questions and stakeholders will be briefly discussed in this section.

1.2.1 Methodology

The Design Science will be the methodology [16] (Hevner) followed in this research project. The design science research can be briefly analyzed as an embodiment of three closely related cycles of activities. The Relevance Cycle inputs requirements from the contextual environment into the research and introduces the research artifacts into environmental field testing. The Rigor Cycle provides grounding theories and methods along with domain experience and expertise from the foundations knowledge base into the research and adds the new knowledge generated by the research to the growing knowledge base. The central Design Cycle supports a tighter loop of research activity for the construction and evaluation of design artifacts and processes. The recognition of these three cycles in a research project clearly positions and differentiates design science from other research paradigms. Figure 1.4 borrows the IS research framework found in Hevner, March, Park, *et al.* [17] and overlays a focus on the three inherent research cycles.

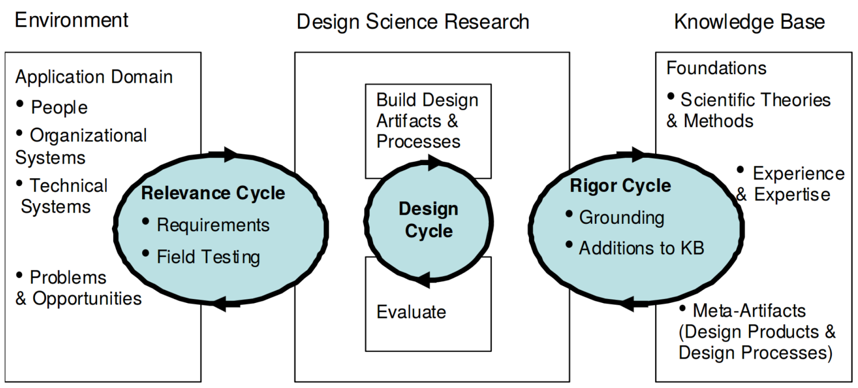


Figure 1.4: Design Science Research Cycles

1.2.2 Main goal and research questions

The main goal of this research is to achieve that a formation of four mobile agents are able to detect an object, track it, pick it and leave it in a different place. This main goal can be achieved by answering the following research questions:

Which is the most suitable formation control law to follow?
This question is basically answered in previous work done in the *DTPA lab* [18] concluding that the formation control law [1] desgined by Garcia de Marina, Jayawardhana, and Cao was the most suitable one. However, in Section 2.2 these reasons are briefly discussed.

Which previous research (useful for the project) has been done in the DTPA-lab?
This research is based mainly on two research master thesis done in the *DTPA lab*. On the one hand, Siemonsma [18] validated the formation control law [1] designed by Garcia de Marina, Jayawardhana, and Cao and implemented it in the Nexus mobile robots. On the other hand, Buursma [19] used one single Nexus mobile robot to implement a pick and place task.

Which hardware is available to perform the research?
The hardware available for this research is the one from the *DTPA lab*. The hardware which is used for this project is more specifically explained in the Chapter 5.

Which is the most suitable ROS configuration?
Since the power of the CPUs mounted in the Nexus mobile robots is limited and that can be a problem related to the frequency communication, some different configuration have been tried in order to run some ROS nodes in the WORKSTATION computer. This is explained in more detail in the Section 6.2.

Which are the algorithms and control used for this research?
Due to the wide variety of ways to face up this problem, the decision made is explained in Chapter 3.

Which is the most suitable sequence of steps that the formation should follow?
The steps which the formation is following in order to perform the task is described in Section 3.1 in more detail.

1.2.3 Stakeholders

In a corporation, a stakeholder is a member of "groups without whose support the organization would cease to exist" [20], as defined in the first usage of the word in a 1963 internal memorandum at the Stanford Research Institute.

Particularly, in this research project, two types of stakeholders can be distinguished. On the one hand, the RUG and more specifically the DTPA lab which provided me of all the material, help and knowledge necessary. On the other hand, Garcia de Marina, Jayawardhana, and Cao who designed the formation control law in which this research project is based.

1.3 Thesis Outline

The remainder of the thesis is organized as follows:

- *Chapter 2* explains the background of this research project. Two main topics are described in this chapter. First of all, the Smart robotic arm is explained by giving its DH parameters, forward kinematics and inverse kinematics. Secondly, it is also given an explanation for the formation control law [1] designed by Garcia de Marina, Jayawardhana, and Cao. This chapter becomes crucial for the well understanding of the project.
- *Chapter 3* provides a problem formulation and describes the algorithm developed to tackle the problem.
- *Chapter 4* describes the simulation setup and the experimental design and shows the results obtained from the simulation.
- *Chapter 5* describes the experimental setup and shows the experimental results.
- *Chapter 6* provides a discussion of the thesis specifying what has been achieved and the limitations which have arisen.
- *Chapter 7* concludes all the previous chapters highlighting the most important points of the thesis.
- *Chapter 8* give some recommendations about the future work to be done.

2 Literature study

2.1 Robotic Arm Control

2.1.1 Frame Corrections and Denavit-Hartenberg Parameters

In order to describe the motion of the manipulator the DH notation is used. These calculations have been performed according to [21]. The particular robotic arm adopted in this project can be seen in Figure 2.1 with its angles, distances and axes. The values of the DH parameters are also shown in table 2.1 (See Appendix B for further information).

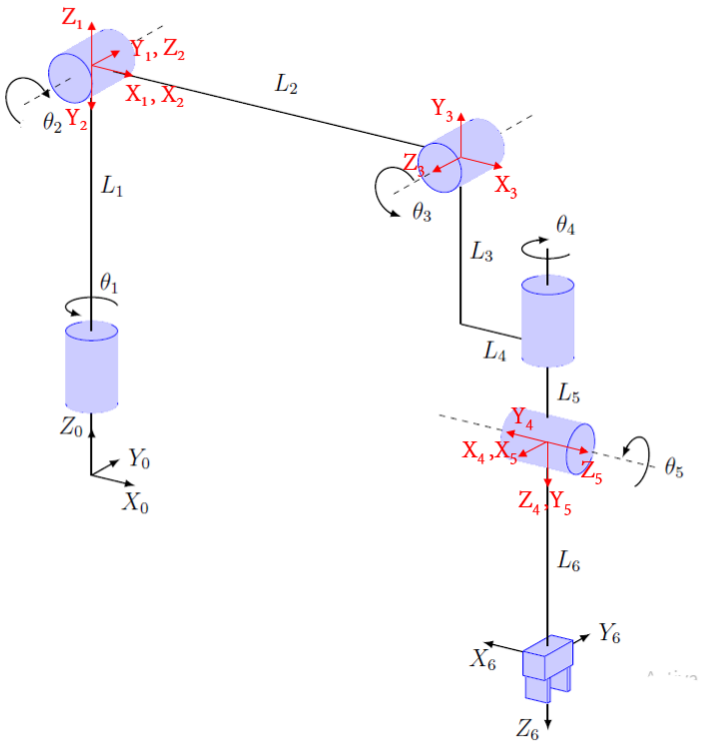


Figure 2.1: Schematic drawing of the CrustCrawler arm. The joint set depicted is: $\theta_1 = 0, \theta_2 = \pi/2, \theta_3 = -\pi/2, \theta_4 = 0, \theta_5 = 0$.

Link	α_{i-1}	a_{i-1}	θ_i	d_i
1	0	0	θ_1	L_1
2	$-\pi/2$	0	$-\pi/2 + \theta_2$	0
3	π	L_2	$\pi/2 + \theta_3$	0
4	$\pi/2$	L_4	$\pi/2 + \theta_4$	$L_3 + L_5$
5	$\pi/2$	0	θ_5	0
6	$-\pi/2$	0	$\pi/2$	L_6

Table 2.1: The standardized D-H parameters for the Robotic arm.

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \cos(\alpha_{i-1})\sin(\theta_i) & \cos(\alpha_{i-1})\cos(\theta_i) & -\sin(\alpha_{i-1}) & -d_i\sin(\alpha_{i-1}) \\ \sin(\alpha_{i-1})\sin(\theta_i) & \sin(\alpha_{i-1})\cos(\theta_i) & \cos(\alpha_{i-1}) & d_i\cos(\alpha_{i-1}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1} $$

To calculate the transformation matrix from one joint frame to the next one equation (2.1) is used. Taking into account that ${}^{i-1}_{i}T$ is the transformation relation between one frame (i-1) to another frame (i), is deduced:

$$ {}^{0}_{N}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \cdots {}^{N-1}_{N}T \tag{2.2} $$

Thus, from equations (2.1) and (2.2) the following matrix (2.3) can be calculated as:

$$ {}^{0}_{N}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3} $$

where r_{ij} determines the rotation matrix ${}^{0}_{N}R$, while p_x, p_y, p_z forms the position vector ${}^{0}P_N$.

2.1.2 Forward Kinematics

One of the main problems of this research is to transform what is seen from the camera mounted in the manipulator to the base of the robot in order to be able to control the position of the formation and move it towards the object. To perform the calculation shown in section 2.1.1 the different parts of the Robotic Arm have been measured obtaining the lengths shown in Table 5.1.

Taking into account these values (Table 5.1) and the values of the DH parameters (Table 2.1) the different transformations matrix can be calculated (see Appendix C to see these calculations).

Thus, the transformation matrix from frame C (camera) to frame 0 (base) is described in (2.4)

$$ {}^{0}_{C}T = {}^{5}_{C}T\,{}^{0}_{5}T \tag{2.4} $$

Taking into account that ${}^{C}P$ is the vector of the tag respect to the camera and ${}^{0}P$ is the vector of the tag respect to the base frame and that ${}^{0}P$ and ${}^{C}P$ are listed below as

$$ {}^{0}P = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} ; {}^{C}P = \begin{bmatrix} x_C \\ y_C \\ z_C \\ 1 \end{bmatrix} \tag{2.5} $$

The position of the tag respect to the base ${}^{0}P$ can be easily calculated considering the equations (2.4) and (2.5)

$$ {}^{0}P = {}^{0}_{C}T\,{}^{C}P \tag{2.6} $$

2.1.3 Inverse Kinematics

Once the system is in a position in which the robotic arm can reach the tag, means that the tag is in the **reachable workspace** [22]. Since the position of the tag respect to the base is known (${}^{B}P$), it is time to perform the Inverse Kinematics in order to know the different joint angle values. For simplicity and in order to have an analytic solution, the joint angles 4 and 5 have been set to 0 ($\theta_4 \equiv \theta_5 \equiv 0$). First of all, the transformation matrix ${}^{0}_{6}T$ has to be taken into account:

$$ {}^{0}_{6}T = \begin{bmatrix} {}^{0}_{6}R & {}^{0}p_6 \\ 0 & 1 \end{bmatrix} \tag{2.7}$$

where ${}^{0}p_6$ is:

$$ {}^{0}p_6 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.8}$$

These calculations are listed in more detail in Appendix D to finally obtain the following joint values:

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \arctan 2(y, x) \\ \arctan 2(s_2, c_2) \\ \arccos\left(\frac{c^2 - L_2^2 - b^2}{2L_2 b}\right) - \beta \end{bmatrix} \tag{2.9}$$

2.2 Formation Control

As highlighted in section 1.2, the other key focus of this thesis is **formation control** in order to keep a shape and move the formation towards the object. This is achieved via the algorithm developed by Garcia de Marina, Jayawardhana, and Cao [1] (2016). In the following, the motivation which drove us to the choice of this particular algorithm is described.

1. The formation shape and the motion control are achieved simultaneously while in other algorithms these two problems are usually tackled separately by using the gradient-based strategies for formation control and leader–follower coordination for motion control. In this last case, the leader moves according to a desired trajectory and the followers simply track the leader [23].
2. No extra sensors and estimators are needed which leads to a cheaper system because the formation shape and the motion control are achieved simultaneously.
3. The simplicity of the algorithm opens possibilities to solve difficult problems such as collective rotational motion, the enclosing of a moving target, and the formation coordination task for agents governed by higher order dynamics.
4. The addition of a new agent shouldn't be a problem since the algorithm is flexible enough. This is explained in more detail in Section 2.2.5.
5. Only a single sensor in one of the agents is needed to know the position of the formation in a global coordinate. This is because each agent i can work with only its own local frame O_i.

2.2.1 Preliminaries

In this section, some notations and basic concepts are introduced. For a given matrix $A \in \mathbb{R}^{n \times p}$, define $\overline{A} \triangleq A \otimes I_m \in \mathbb{R}^{nm \times pm}$, where the symbol $\otimes$ denotes the Kronecker product[1], $m = 2$ for $\mathbb{R}^2$ or otherwise 3 for $\mathbb{R}^3$, and I_m is the m-dimensional identity matrix. For a stacked vector $x \triangleq \text{col}(x_1, \cdots, x_k)$ with $x_i \in \mathbb{R}^n, i \in \{1, \cdots, k\}$, the diagonal matrix $D_x \triangleq \text{diag}(x_i)_{i\in\{1,\cdots,k\}} \in \mathbb{R}^{kn\times k}$ is defined. The cardinality of the set χ is denoted by $|\chi|$ and the Euclidean norm of a vector x is denoted by $||x||$. $\mathbf{1}_{n\times m}$ and $\mathbf{0}_{n\times m}$ are used to denote the all-one and all-zero matrix in $\mathbb{R}^{n\times m}$, respectively.

2.2.2 Formations and Graphs

A formation of $n \geq 2$ autonomous agents whose positions are denoted by $p_i \in \mathbb{R}^m$ is considered. The agents are able to sense the relative positions of its neighboring agents. The neighbor relationships are described by an indirect graph $\mathbb{G} = (\mathcal{V}, \mathcal{E})$ with the vertex set $\mathcal{V} = \{1, \cdots, n\}$ and the ordered edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The set $\mathcal{N}_i$ of the neighbors of agent i is defined by $\mathcal{N}_i \triangleq \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$. The elements of the incidence matrix $B \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$ for $\mathbb{G}$ by

$$b_{ik} \triangleq \begin{cases} +1, & if\ i = \mathcal{E}_k^{\text{tail}} \\ -1, & if\ i = \mathcal{E}_k^{\text{head}} \\ 0, & \text{otherwise} \end{cases} \tag{2.10}$$

where $\mathcal{E}_k^{\text{tail}}$ and $\mathcal{E}_k^{\text{head}}$ denote the tail and head nodes, respectively, of the edge $\mathcal{E}_k$, i.e., $\mathcal{E}_k = (\mathcal{E}_k^{\text{tail}}, \mathcal{E}_k^{\text{head}})$. A framework is defined by the pair $(\mathbb{G}, p)$ where $p = \text{col}(p_1, \cdots, p_n)$. The stacked vector of the sensed relative distances can then be described by

$$z = \overline{B}^\top p \tag{2.11}$$

Note that each vector $z_k = p_i - p_j$ in z corresponds to the relative position associated with the edge $\mathcal{E}_k^{head} = (i, j)$.

2.2.3 Infinitesimally and Minimally Rigid Formations and Their Realization

In this section, the concept of rigid formations and how to design them will be briefly reviewed. Most of the concepts explained here are covered in [1] and in more detail in [24], [25]. Define the edge function $f_{\mathbb{G}}(p) = col_k(||z_k||^2)$ and denote its Jacobian by $R(z) = D_z^\top \overline{B}^\top$, which is called the rigidity matrix in the literature. A framework $(\mathbb{G}, p)$ is infinitesimally rigid if $\text{rank}(R(z)) = 2n - 3$ when embedded in $\mathbb{R}^2$ or if $\text{rank}(R(z)) = 3n - 6$ when embedded in $\mathbb{R}^3$. Additionally, if $|\mathcal{E}| = 2n - 3$ in the 2-D case or $|\mathcal{E}| = 3n - 6$ in the 3-D case, then the framework is called minimally rigid. Roughly speaking, under the distance constraints, the only motions that one can perform over the agents in an infinitesimally and minimally rigid framework, while they are already in the desired shape, are the ones defining translations and rotations of the whole shape. In Figure 2.2 some examples in $\mathbb{R}^2$ and $\mathbb{R}^3$ of rigid and nonrigid frameworks are illustrated.

For a given stacked vector of desired relative positions $z^* = \text{col}(z_1^*\ z_2^* \cdots z_{|\mathcal{E}|}^*)$, $\mathcal{Z}$ describes the set of the possible formations, namely

$$\mathcal{Z} \triangleq \{(I_{|\mathcal{E}|} \otimes \mathcal{R})z^*\} \tag{2.12}$$

where $\mathcal{R}$ is the set of rotational matrices in $\mathbb{R}^2$ or $\mathbb{R}^3$. Roughly speaking, $\mathcal{Z}$ consists of all formation positions that are obtained by rotating z^*. If $(\mathbb{G}, p)$ is infinitesimally and minimally rigid, then, similar to the above, the set of the resulting formations $\mathcal{D}$ can be defined by

[1] Kronecker product, Wikipedia, `https://en.wikipedia.org/wiki/Kronecker_product` (accessed 1 April 2019)

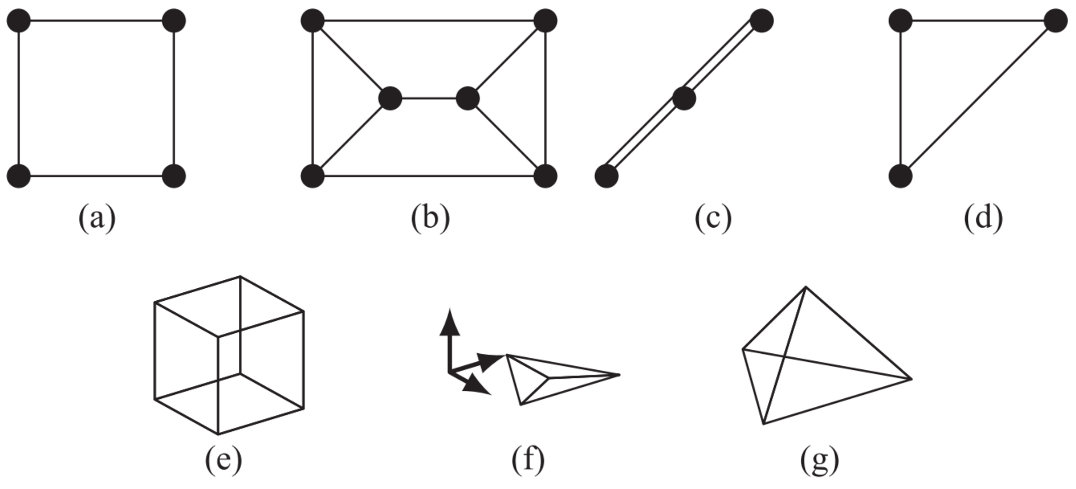


Figure 2.2: (a) Square without an inner diagonal is not rigid, since we can smoothly move the top two nodes, keeping the other two fixed without breaking the distance constraints. (b) Rigid but not an infinitesimally rigid framework. If we rotate the left inner triangle, then the right inner triangle can be counterrotated in order to keep the interdistances constant. (c) Minimally rigid but not an infinitesimally rigid framework since the nodes' positions are collinear. (d) Triangle is infinitesimally and minimally rigid. (e) Cube formed by squares without diagonals is not rigid. (f) Zero-volume tetrahedron is rigid but not infinitesimally rigid in $\mathbb{R}^3$, since all the nodes are coplanar. (g) Tetrahedron in $\mathbb{R}^3$ is infinitesimally and minimally rigid.

$$\mathcal{D} \triangleq \{z \mid ||z_k|| = d_k, k \in \{1, \cdots, |\mathcal{E}|\}\} \tag{2.13}$$

where $d_k = ||z_k^*||, k \in \{1, \cdots, |\mathcal{E}|\}$

Note that, in general, it holds that $\mathcal{Z} \subseteq \mathcal{D}$. For a desired shape, one can always design $\mathbb{G}$ to make the formation infinitesimally and minimally rigid. In fact, in $\mathbb{R}^2$, an infinitesimally and minimally rigid framework with two or more vertices can always be constructed through the Henneberg construction [26]. In $\mathbb{R}^3$, one can always obtain an infinitesimally and minimally rigid framework by the Henneberg-like tetrahedron insertions, as has been done in [23].

2.2.4 Gradient control in interagent distances

As explained in [1], many coordinated tasks can be achieved by combining two different cooperative controls: multiagent formation and group motion control. For formations of complicated shapes, these two problems are usually tackled separately, by using the gradient-based strategies for formation control and leader-follower coordination for motion control, in which for the latter, the leader moves according to a desired trajectory and the followers simply track the leader [27]. However, in this work, only gradient-based is used for both objectives.

In gradient-based formation control, stabilizable formations are identified by employing rigidity graph theory, in which the vertices of a graph represent the agents and the edges stand for the interagent distance constraints to define the shape of the formation. Rigid formations are associated with a potential function determined by the agents' relative positions. The potential has a minimum at the desired distances between the agents; thus, its gradient leads naturally to the formation controller that stabilizes rigid formations locally.

For a formation of n agents associated with the neighbor relationship graph $\mathbb{G}$, consider the following system where we model the agents as kinematical points:

$$\dot{p} = u \tag{2.14}$$

where u is the stacked vector of control inputs $u_i \in \mathbb{R}^m$ for $i = \{1, \cdots, n\}$.

For each edge $\mathcal{E}_k$, one can construct a potential function V_k with its minimum at the desired distance $||z_k^*||$ so that the gradient of such functions can be used to control interagent distances distributively.

A potential function is defined

$$V(\overline{B}^\top p) = V(z) = \sum_{k=1}^{|\mathcal{E}|} V_k(z_k) \tag{2.15}$$

and then the gradient descent control can be applied to each agent i in (2.14)

$$u_i = -\nabla_{p_i} \sum_{k=1}^{|\mathcal{E}|} V_k(z_k) \tag{2.16}$$

It can than be shown that the multi-agent formation will converge locally to the desired shape. According to [1], [28] and [29], the agent dynamics (2.14) under (2.16) can be written in the compact form

$$\dot{p} = -\overline{B}^\top \nabla_z V(z) \tag{2.17}$$

where $\nabla_z V(z)$ is the stacked vector of $\nabla_{z_k} V_k(||z_k||)$'s. Denoting the distance error for edge k by

$$e_k = ||z_k||^l - d_k^l \tag{2.18}$$

where $l \in \mathbb{N}$. It follows that

$$\nabla_{z_k} V_k(||z_k||) = z_k ||z_k||^{l-2} e_k \tag{2.19}$$

By substituting it into (2.17) and noting that

$$\dot{e}_k = l||z_k||^{l-1} \frac{d}{dt} ||z_k|| = l||z_k||^{l-2} z_k^\top \dot{z}_k \tag{2.20}$$

the closed-loop dynamics can be written in compact form as

$$\dot{p} = -\overline{B} D_z D_{\tilde{z}} e = -R(z)^\top D_{\tilde{z}} e \tag{2.21}$$

$$\dot{z} = \overline{B} \dot{p} = -\overline{B} R(z)^\top D_{\tilde{z}} e \tag{2.22}$$

$$\dot{e} = l D_{\tilde{z}} D_z^\top \dot{z} = l D_{\tilde{z}} R(z) R(z)^\top D_{\tilde{z}} e \tag{2.23}$$

where $e \in \mathbb{R}^{|\mathcal{E}|}$ is the stacked vector of e_k's, $\tilde{z} \in \mathbb{R}^{|\mathcal{E}|}$ is the stacked column vector consisting of all the $||z_k||^{l-2}$s, and the matrices D_z and $D_{\tilde{z}}$ are the block diagonal matrices of z and $\tilde{z}$, respectively, as defined in Section 2.2.1.

If the desired formation $\mathcal{D}$ is infinitesimally and minimally rigid, then the error signal e will locally go to zero.

2.2.5 Gradient-based formation-motion control

Notation of the frames of coordinates

The global frame of coordinates is denoted by O_g. The global frame is fixed at the origin of $\mathbb{R}^m$ with an arbitrary fixed orientation. Similarly, the local frame for agent i is denoted by O_i with some arbitrary orientation independent of O_g. The body frame is denoted by O_b. The body frame is fixed at the centroid p_c of the desired rigid formation. If we rotate the rigid formation with respect to O_g, then O_b is also rotated in the same manner. Recall that, ${}^ip_j$ defines the position of agent j with respect to O_i. In order to simplify the notation, the superscript is omitted whenever we discus an agents' variable with respect to O_g, e.g. $p_j \triangleq {}^gp_j$.

Inducing motion parameters in the gradient descent controller

In [1] a pair of motion control parameters are used in order to achieve a steady state translational and rotational motion. The motion control parameters are denoted by $\mu_k \in \mathbb{R}^2$ and $\tilde{\mu}_k \in \mathbb{R}^2$. These parameters are scaled by a gain c and added to the term d_k^l for each agent associated with the edge $\mathcal{E}_k = (i, j)$. Note that in [1], $l = 2$ is used throughout the analysis for the sake of simplicity and clarity of the notation (in the case of $l = 2$, $\mathcal{D}_{\tilde{z}}$ becomes the identity matrix). However, from now onwards the case for l = 1 is considered since this is more convenient during the experiments and since the stability analysis are omitted in this Thesis. Note that the main results of the analysis can be easily extended to any $l \in \mathbb{N}$. After scaling the motion control parameters and adding them to the term d_k, agent i uses a controlled distance of $d_k + \frac{\mu_k}{c}$ and agent j uses $d_k - \frac{\tilde{\mu}_k}{c}$. For the corresponding edge $\mathcal{E}_k = (i, j)$, these parameters are introduced in the gradient descent controller with the gain c, namely

$$u_i^k = -c\frac{z_k}{||z_k||}(||z_k|| - d_k) + \mu_k z_k \tag{2.24}$$

$$u_j^k = c\frac{z_k}{||z_k||}(||z_k|| - d_k) + \tilde{\mu}_k z_k \tag{2.25}$$

where u_i^k and u_j^k are the corresponding control inputs for agents i and j with edge $\mathcal{E}_k$. The equations above can be written in the following compact form

$$\dot{p} = -\overline{B}D_zD_{\tilde{z}}e + \overline{A}(\mu, \tilde{\mu})z \tag{2.26}$$

The elements a_{ik} of A are constructed in a very similar way as in the incidence matrix, namely

$$a_{ik} \triangleq \begin{cases} \mu_k, & if \ \ i = \mathcal{E}_k^{\text{tail}} \\ \tilde{\mu}_k, & if \ \ i = \mathcal{E}_k^{\text{head}} \\ 0, & \text{otherwise} \end{cases} \tag{2.27}$$

Therefore, μ and $\tilde{\mu} \in \mathbb{R}^{|\mathcal{E}|}$ are defined as the stacked vectors of μ_k and $\tilde{\mu_k}$ for all $k \in \{1, \cdots, |\mathcal{E}|\}$. Note that if $\mu = -\tilde{\mu}$, then $d_k + \frac{\mu_k}{c} = d_k - \frac{\tilde{\mu}_k}{c} = \tilde{d}_k$. This implies that when control law (2.26) is applied, while $\mu = -\tilde{\mu}$, only gradient-based formation shape control occurs with $\tilde{d}_k$ being the new stacked vector of prescribed distances. Furthermore, the gain c is a free design parameter for achieving exponential stability of the formation as shown in the stability analysis in [1].

One important property of (2.26) is that each agent i can work with only its own local frame O_i. This is shown in more detail in [1, Lemma 4.1 on p. 687]. In order to induce some desired steady-state motion of the formation in the desired shape, μ and $\tilde{\mu}$ can be manipulated at the equilibrium of (2.26). Then, the steady-state motion is a function of the desired shape $z \in \mathcal{Z}$ and $\mu, \tilde{\mu}$ namely

$$\dot{p}^* = \overline{A}(\mu, \tilde{\mu})z^* \tag{2.28}$$

Rigid body mechanics and decomposition of the motion parameters

Some notions from rigid body mechanics and the decomposition of the motion parameters are discussed below.

Some notions from rigid body mechanics in [30] can be used for describing agents' i velocity. As in the case of points in a rigid body, the steady-state velocity of every agent $\dot{p}_i^*$ at the desired rigid formation shape can be decomposed into

$$\dot{p}_i^* = \dot{p}_c^* + \underbrace{{}^b w \times {}^b p_i^*}_{\dot{p}_{i_w}^*} \tag{2.29}$$

where ${}^b w$ is the angular velocity of the rigid formation (similar to to that for the rigid body) and $\times$ denotes the cross product. In particular, in view of control law (2.26), and when the agents are in the desired shape $z^* \in \mathcal{Z}$, agents' i velocity (2.29) is given by

$$\dot{p}_c^* + \dot{p}_{i_w}^* = \sum_{k=1}^{|\mathcal{E}|} a_{ik} z_k^* \tag{2.30}$$

In order to achieve the translational and rotational movement of the whole formation, in [1] the motion parameters are descomposed into $\mu = \mu_v + \mu_w$ and $\tilde{\mu} = \tilde{\mu}_v + \tilde{\mu}_w$. Where $\mu_v, \tilde{\mu}_v \in \mathbb{R}^{|\mathcal{E}|}$ are used to assign the desired translational velocity and $\mu_w, \tilde{\mu}_w \in \mathbb{R}^{|\mathcal{E}|}$ are used to assign the desired rotational velocity. Using this decomposition, (2.30) can be rewritten for all the agents into the following compact form

$$\dot{p}^* = \underbrace{\overline{A}(\mu_v, \tilde{\mu}_v)z^*}_{\mathbf{1}_{|\mathcal{V}|\otimes 1}\dot{p}_c^*} + \underbrace{\overline{A}(\mu_w, \tilde{\mu}_w)z^*}_{\dot{p}_w^*} \tag{2.31}$$

where $\dot{p}_w^* \in \mathbb{R}^{m|\mathcal{V}|}$ is the stacked vector of all the rotational velocities $\dot{p}_{i_w}^*$, $i \in \{1, \cdots, |\mathcal{V}|\}$.

3 Problem formulation and algorithm development

In this chapter, a problem formulation is provided and the algorithms developed to tackle the problem will be discussed.

3.1 Problem formulation

The problem can be described as a finite state machine as shown in Figure 3.1. Thus, the automata can not go to the next state until certain conditions are met. These states and conditions are described below:

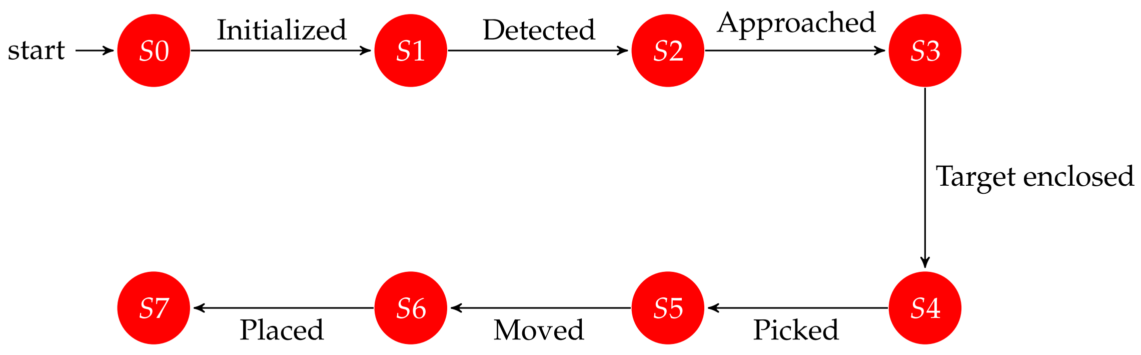
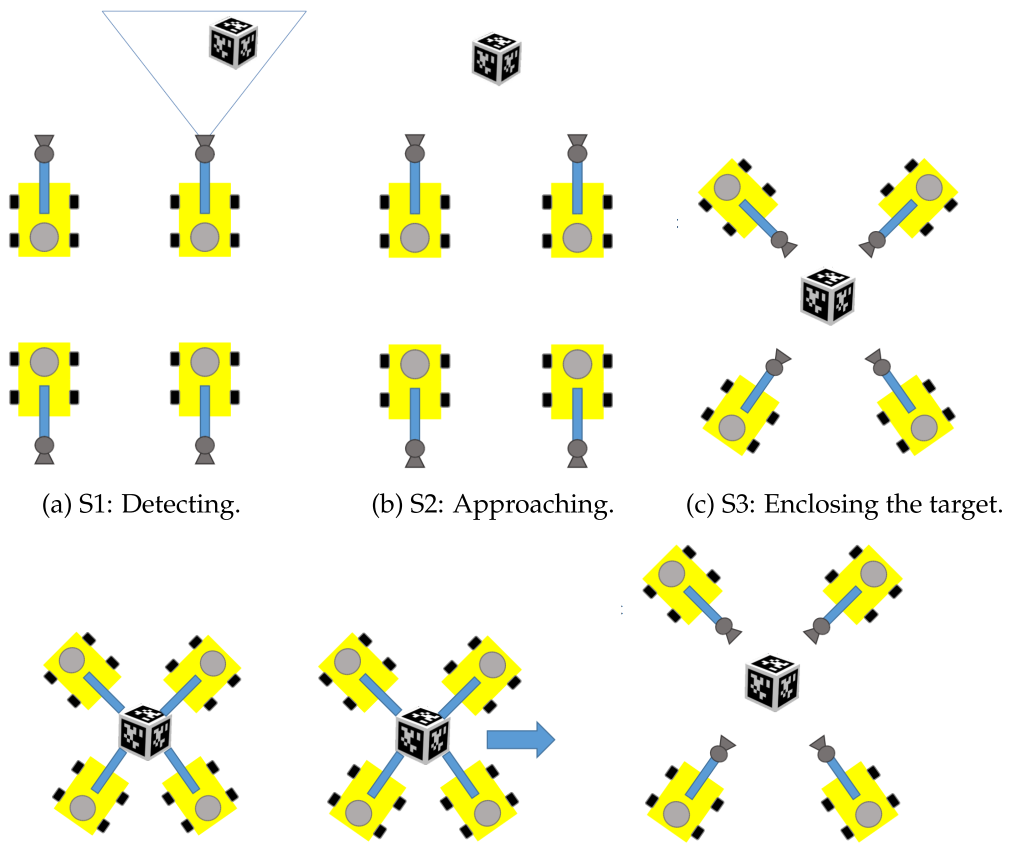


Figure 3.1: Finite state machine of the formation. Note that the loops in the same state are not shown in order to make the automata clearer

- *Initializing (S0)*: The four agents have to be correctly initialized. That consists of recognizing each other in order to keep the formation and to move the four robotic arms in order to keep the marker visible.
- *Detecting (S1)*: The four agents are trying to detect the marker. Only some of them will be able to detect it, then the agent which is closer to the marker becomes the leader of the formation. In the case the formation does not detect any object, since the camera mounted on each agent has a limited field of view, the formation will rotate until an object is detected.
- *Approaching (S2)*: The four agents move towards the object and stop in a desired distance in front of it.
- *Enclosing the target (S3)*: The four agents move around the object with an orientation in which the robotic arm is pointing the object.
- *Picking (S4)*: The four robotic arms pick the object in a open loop since no feedback is possible due to the camera is not pointing the object anymore. The joint values are known performing the Inverse Kinematics (Section 2.1.3).
- *Moving (S5)*: The formation moves to another place in an open loop because to receive feedback more sensors will be needed and the CPUs are not that powerful. Moreover, it is not the aim of this research project.

- *Placing (S6)*: The formation leaves the object in the place reached in state *S5*.
- *Shutdown (S7)*: The system is shutdown.

The steps described above are shown in a more illustrative way in the Figure 3.2.



(a) S1: Detecting. (b) S2: Approaching. (c) S3: Enclosing the target.

(d) S4: Picking. (e) S5: Moving. (f) S6: Placing.

Figure 3.2: Steps followed by the formation to perform the task.

3.2 Algorithm development

The block scheme of the closed loop system achieved in this research is shown below in the Figure 3.3.

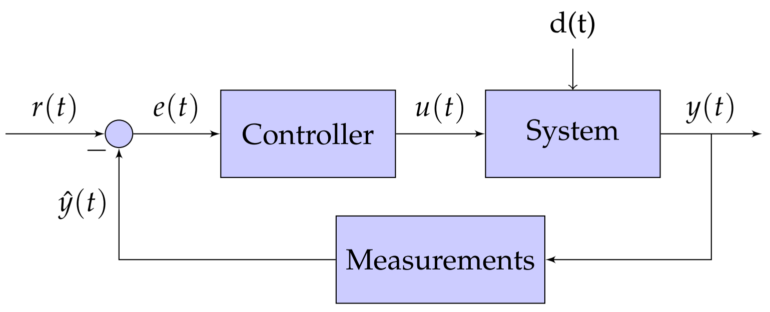


Figure 3.3: Main basic algorithm used in the research project.

Where $r(t)$ is the reference signal, $y(t)$ is the output of the system, $\hat{y}(t)$ is the measured output of the system, $e(t)$ is the error and $u(t)$ is the control input. This control scheme will be conjugated in different ways to solve different subproblems of the complex system considered. For instance, in Section 3.2.2 the control is a PI controller which controls the inputs (movement) of the Nexus mobile robot (system). Finally, the *Measurements* block provides an estimation of the output of the system obtained from some sensors.

3.2.1 Detection and leader decision

The formation will rotate until the object is detected. The agent whose camera first detects the object will become the leader. Thus, depending on which agent is the leader of the formation, Table 3.1 specifies which are the desired distances for the approaching control. The axis in the x and y directions are defined taking into account the movement of the agent 1 as described in Figure 3.4.

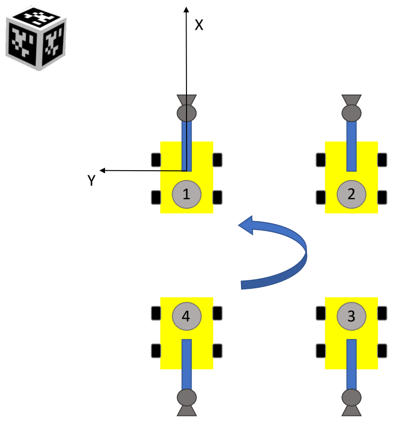


Figure 3.4: Detecting state. The formation starts to rotate. According to the direction of rotation, the leader would be agent 1 because is the first agent to see the object.

Agent	1	2	3	4
d_x^*	d^*	d^*	$-d^*$	$-d^*$
d_y^*	$-d_k/2$	$d_k/2$	$d_k/2$	$-d_k/2$

Table 3.1: Desired distances depending on the leader of the formation.

Where d^* is the desired distance in which the formation stops before starting to move around the object and d_k is the desired edge distance for any $k \neq 3$ (see Figure 4.1).

3.2.2 Approach control

Once the object is detected and the leader is defined, the formation is able to start approaching the object. This approach is done by using a PI controller and the desired distances shown in Table 3.1. The desired orientation of the formation, in this case, will be 0.

The PID controller is by far the most common control algorithm [31]. Most practical feedback loops are based on PID control or some minor variations of it. If $u(t)$ is defined as a control input, the algorithm form of a PID controller can be described as

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d \frac{de}{dt} \tag{3.1}$$

where u is the control signal and e is the error signal. The controller parameters are the proportional gain k_p, integral gain k_i and derivative gain k_d.

However, many controllers do not even use derivative action. In this particular case, only a PI controller is used in order to control the position of the formation by measuring the distances in the x-direction and y-direction and the angle by measuring the orientation in the z-direction. This results in the equations listed below:

$$u_x = -k_{px} e_x - k_{ix} \int_0^t e_x(\tau) d\tau \tag{3.2}$$

where $e_x = d_x - d_x^*$, being d_x and d_x^* the distance to the tag and the desired distance in the x-direction respectively.

$$u_y = -k_{py} e_y - k_{iy} \int_0^t e_y(\tau) d\tau \tag{3.3}$$

where $e_y = d_y - d_y^*$, being d_y and d_y^* the distance to the tag and the desired distance in the y-direction respectively.

$$u_\theta = -k_{p\theta} e_\theta - k_{i\theta} \int_0^t e_\theta(\tau) d\tau \tag{3.4}$$

where $e_\theta = \theta - \theta^*$. The orientation angle θ is defined as the angle between the x-axis of the formation and perpendicular to the tag. Thus, the desired value of this angle is $\theta^* \equiv 0$. In the appendix E is explained how to get the orientation of the apriltag relative to the base frame. Moreover, the inputs described above are negative due to Nexus setup and coordination frame.

The block scheme of the closed loop movement of the nexus is shown below in Figure 3.6

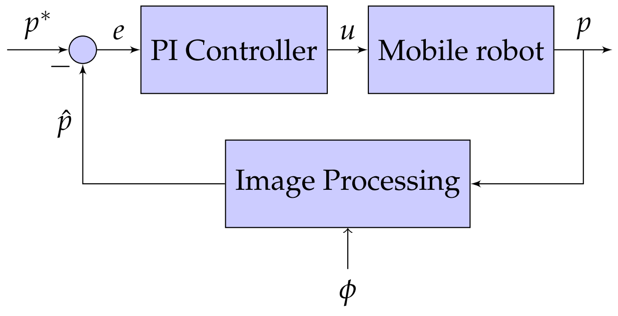


Figure 3.5: Particular algorithm used for the approaching control. The *Image processing* block is formed by the detection of the tag and the forward kinematics needed for computing the position of the mobile robot base frame instead of the camera frame.

where $p^* = [d_x^*\ d_y^*\ \theta^*]^\top$, $\hat{p} = [d_x\ d_y\ \theta]^\top$, $e = [e_x\ e_y\ e_\theta]^\top$, $u = [u_x\ u_y\ u_\theta]^\top$ and p is the output position and orientation of the nexus. The vector $\phi = [\phi_2\ \phi_3\ \phi_5]^\top$ represents the angle values of joints 2, 3 and 5 of the robotic arm, the ones responsible for centering the tag (see Section 3.2.3).

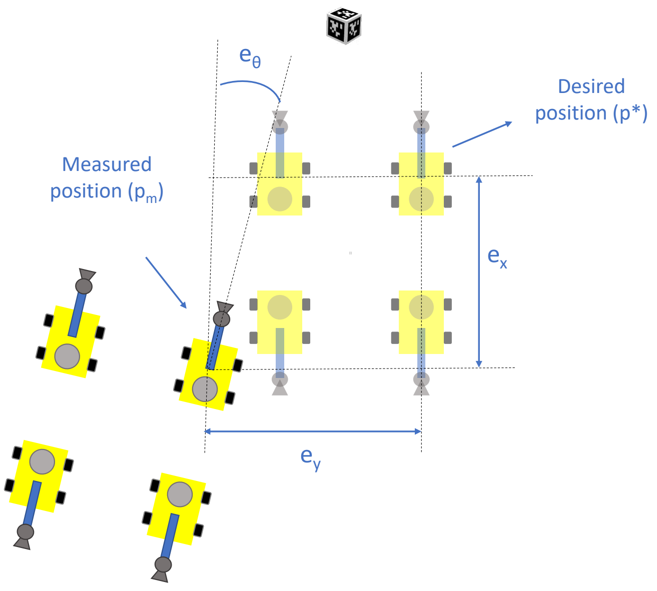


Figure 3.6: Formation approaching to the object with the three errors considered in the PI control.

3.2.3 Centering of the tag

The fact that the camera has a limited field of view means that a control of the robotic arm should be implemented in order to correct the position of its joints to keep the tag visible all the time. These errors are computed as

$$e_{cx} = \arctan \frac{\alpha_x}{\alpha_z} \tag{3.5}$$

$$e_{cy} = \arctan \frac{-\alpha_y}{\alpha_z} \tag{3.6}$$

where e_{cx} and e_{cy} are the horizontal and vertical error from the center of the camera frame and the one of the apriltag, respectively, see Figure 3.7. Moreover, α_x, α_y and α_z are the positions in x, y and z seen from the camera.

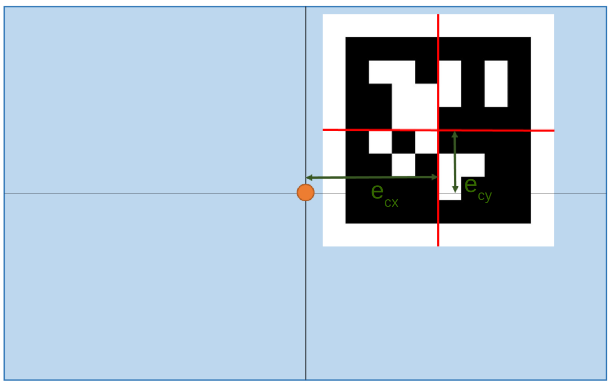


Figure 3.7: Errors between the camera and the center of the apriltag. The orange dot represents the center of the camera while the red cross represents de center of the apriltag.

The horizontal and vertical compensation are handled by the robotic arm in a decoupled way. The block diagram is represented below in Figure 3.8

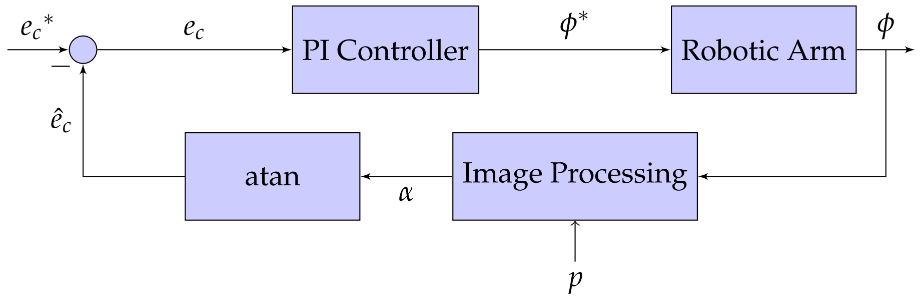


Figure 3.8: Block diagram used for the centering control.

where $e_c = \hat{e}_c = [e_{cx}\ e_{cy}]^\top$, $e_c^* = [0\ 0]^\top$, $\alpha = [\alpha_x\ \alpha_y\ \alpha_z]^\top$ and $\phi^* = [\phi_2^*\ \phi_3^*\ \phi_5^*]^\top$ is the vector of the desired angle values of the joints in order to keep the tag centered.

Gravity compensation

During the implementation of centering algorithm, it has been noticed that the real position of a joint was always different from the desired one and the error was always in the gravity force direction. This is because the robotic arm needs power for keeping a certain joint configuration with a specific window error. In Figure 3.9, the behaviour of the servos is represented.

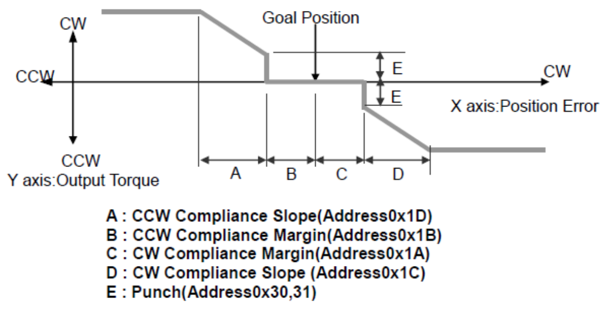
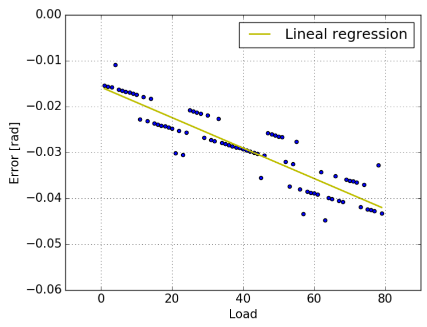
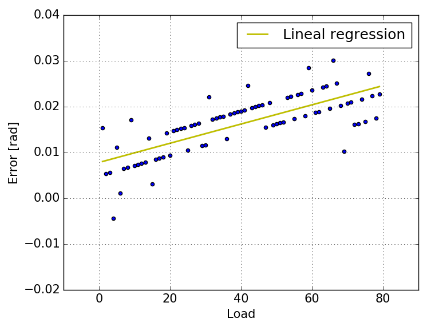

CW
CCW
CCW
Y axis:Output Torque
Goal Position
E
E
CW
X axis:Position Error
A
B
C
D
A : CCW Compliance Slope(Address0x1D)
B : CCW Compliance Margin(Address0x1B)
C : CW Compliance Margin(Address0x1A)
D : CW Compliance Slope (Address0x1C)
E : Punch(Address0x30,31)

Figure 3.9: How the servos reach the goal position.

When implementing the system with feedback of the robotic joint values, it has been noticed that the vertical error (e_{cy}) does not converge to zero but to an specific value (see Figure 3.12a). Furthermore, the higher is the load supported by the joint, the higher is the error (e) between the desired (ϕ^*) and real (ϕ) angle values of the joints. In Figure 3.10 the relationship between these two variables is shown.



(a) Shoulder Pitch Joint



(b) Elbow Flex Joint

Figure 3.10: Lineal regression between the error and load of each joint.

The implemented solution is to replace the Robotic Arm block of Figure 3.8 by the block diagram shown in Figure 3.11. The idea is to control the input to the servos via a proportional controller where $\tilde{\phi}^* = [\tilde{\phi}_2^* \ \tilde{\phi}_3^* \ \tilde{\phi}_5^*]$ is the new input vector to the servos. The improvement of the system is shown in Figure 3.12b.

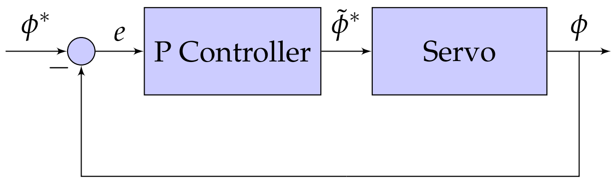
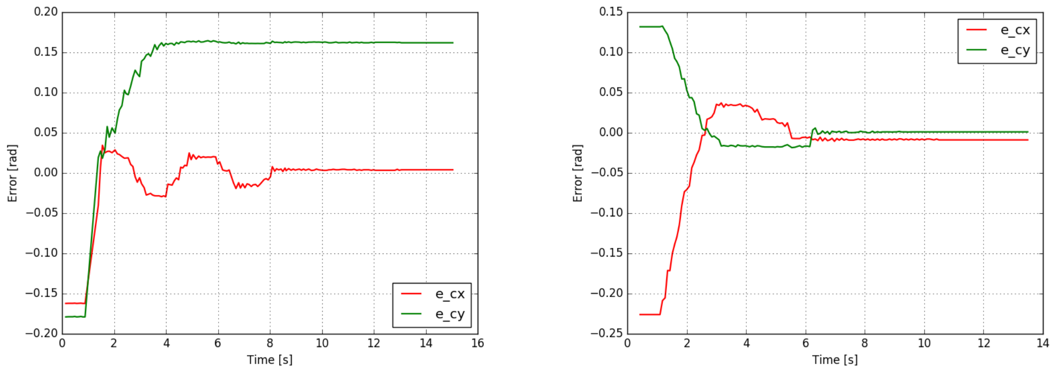


Figure 3.11: Robotic Arm block from Figure 3.8



(a) Without gravity compensation. (b) With gravity compensation.

Figure 3.12: e_{cx} and e_{cy} errors.

In Figure 3.13, the complete block diagram with gravity compensation is shown.

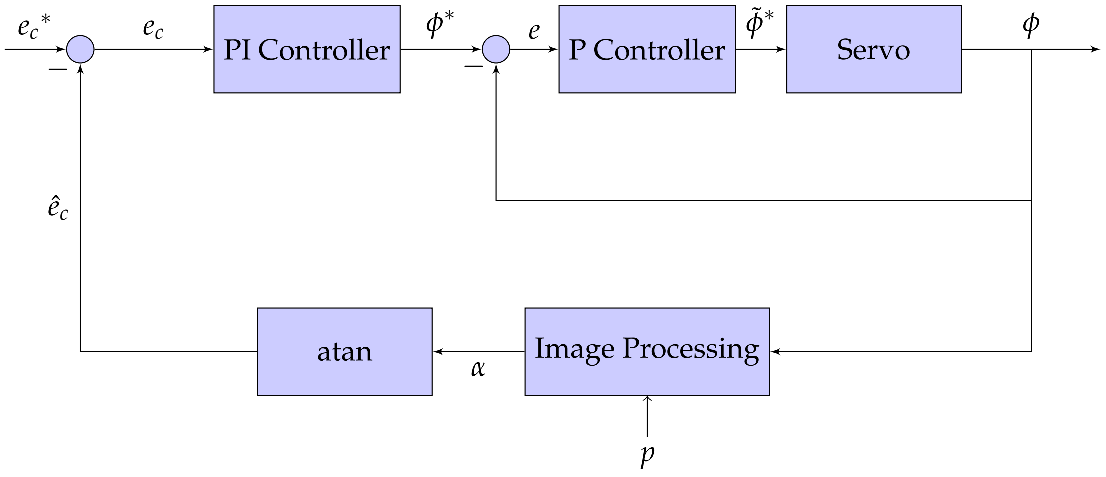


Figure 3.13: Complete block diagram used for the centering control.

4 Simulation setup, experimental design and simulation results

4.1 Simulation setup

The simulation is used to validate the algorithm developed for the approaching of the formation to the target. However, some differences can be distinguished between simulation and experimentation because of the ideality of the simulation. Thus, the assumptions and constraints listed below will be taken into account during the simulation:

The agents will be represented as points because it is not the aim of this simulation to implement the current real scenario but to prove that the algorithm works.

The agents will be supposed to be equipped with a camera of a $360°$ *field of view. Thus, they will be able to see the object wherever it is and no rotation of the formation is needed to detect the object in the initialization. Note that during the real experiments the field of view of the camera is* $70°$*. However, it is possible that in the future this algorithm can be implemented with camera which has a larger field of view.*

The time step of the simulation has been chosen taking into account the limitation of the slowest element of the real scenario. In this case, the update frequency of the real system is limited by the Laser Scanner. The Laser Scanner has an update frequency of $5,5Hz$*, so the time step of the system will be* $\Delta t = 0,18s$*.*

The enclosing algorithm in the simulation does not need to rotate the agent on themselves but it does in the real scenario. That is why during the simulation this algorithm only depends on the distance from the agents to the object.

The maximum velocity of the agents is limited by the maximum velocity of the Nexus cars ($0,6\ m/s$ *as seen in [32]. This is checked during the simulations as explained in Section 4.3.)*

The formation only approaches the object forward or backward.

4.2 Experimental design

4.2.1 Graph topology of the minimally rigid formation during the experiments

As discussed in the preliminaries in Section 2.2, the neighbor relationships between agents can be described by an undirected graph $\mathbb{G} = (\mathcal{V}, \mathcal{E})$ and its corresponding incidence matrix $B \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$. Four robots will be used throughout the experiments, therefore $|\mathcal{V}| = 4$. Recall from Section 2.2 that a formation is defined as minimally rigid in the 2D case if $|\mathcal{E}| = 2n - 3$. Therefore, five edges are needed to compose a minimally rigid formation of four robots. This entails that $|\mathcal{E}| = 5$. Furthermore, recall that the body frame fixed at the centroid ${}^{b}p_c$ of the desired formation is denoted by O_b. The framework of the formation with corresponding incidence matrix are shown in Figure

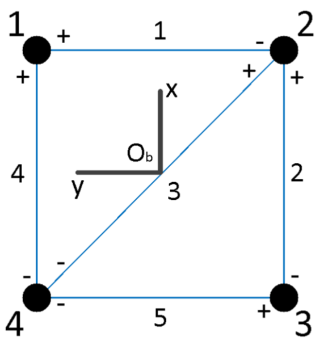


Figure 4.1: Minimally rigid formation of four agents with five edges. The body frame O_b is fixed at the centroid p_c of the desired formation.

4.1 and in (4.1) respectively. Note that in order to form a square, edges 1, 2, 4 and 5 should have a length of a, and edge 3 a length of $\sqrt{2}a$.

$$B = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ -1 & 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & -1 & -1 \end{bmatrix} \tag{4.1}$$

4.2.2 Design of μ and $\tilde{\mu}$ for achieving a desired translational and rotational formation velocity

The motion parameters μ and $\tilde{\mu}$ are induced in control law (2.26) in order to achieve formation motion. Recall from Section 2.2.5 that μ and $\tilde{\mu}$ can be decomposed into $\mu = \mu_v + \mu_w$ and $\tilde{\mu} = \tilde{\mu}_v + \tilde{\mu}_w$. Where μ_v and $\tilde{\mu}_v$ are used to assign the desired translational velocity and μ_w and $\tilde{\mu}_w$ are used to assign the desired rotational velocity. In addition to the decomposition proposed and discussed in [1] and Section 2.2.5, μ_v and $\tilde{\mu}_v$ is further decomposed in $\mu_{vx}, \tilde{\mu}_{vx}, \mu_{vy}$ and $\tilde{\mu}_{vy}$. Where μ_{vx} and $\tilde{\mu}_{vx}$ are used to assign the desired translational velocity in the x-direction of the body frame O_b and μ_{vy} and $\tilde{\mu}_{vy}$ are used to assign the desired translational velocity in the y-direction.

Below, three working examples are used to discuss the design of $\mu_{vx}, \tilde{\mu}_{vx}, \mu_{vy}, \tilde{\mu}_{vy}, \mu_w$ and $\tilde{\mu}_w$ respectively. The parameters s_x, s_y and s_w are used to rescale the formation velocity in the x-, y- and rotational direction respectively.

Translational formation velocity in the x-direction

The aim is to move the formation shown in Figure 4.1 in the x-direction of its own body frame O_b with a velocity of $1m/s$, i.e. ${}^b\dot{p}_c^* = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Following the algorithm described in Section 2.2.5, the motion parameters μ_{vx} and $\tilde{\mu}_{vx}$ needed are respectively

$$\mu_{vx} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{4.2}$$

$$\tilde{\mu}_{vx} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{4.3}$$

Translational formation velocity in the y-direction

The aim is to move the formation shown in Figure 4.1 in the y-direction of its own body frame O_b with a velocity of $1m/s$, i.e. ${}^b\dot{p}_c^* = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. Following the algorithm described in Section 2.2.5, the motion parameters μ_{vy} and $\tilde{\mu}_{vy}$ needed are respectively

$$\mu_{vy} = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{4.4}$$

$$\tilde{\mu}_{vy} = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{4.5}$$

Rotational formation velocity

The aim is to rotate the formation shown in Figure 4.1 with angular velocity ${}^b w = 2\ rad/s$ around its centroid ${}^b\dot{p}_c^*$. Following the algorithm described in Section 2.2.5, the motion parameters μ_w and $\tilde{\mu}_w$ needed are respectively

$$\mu_w = \begin{bmatrix} 1 & 1 & 0 & -1 & 1 \end{bmatrix} \tag{4.6}$$

$$\tilde{\mu}_w = \begin{bmatrix} 1 & 1 & 0 & -1 & 1 \end{bmatrix} \tag{4.7}$$

4.3 Simulation results

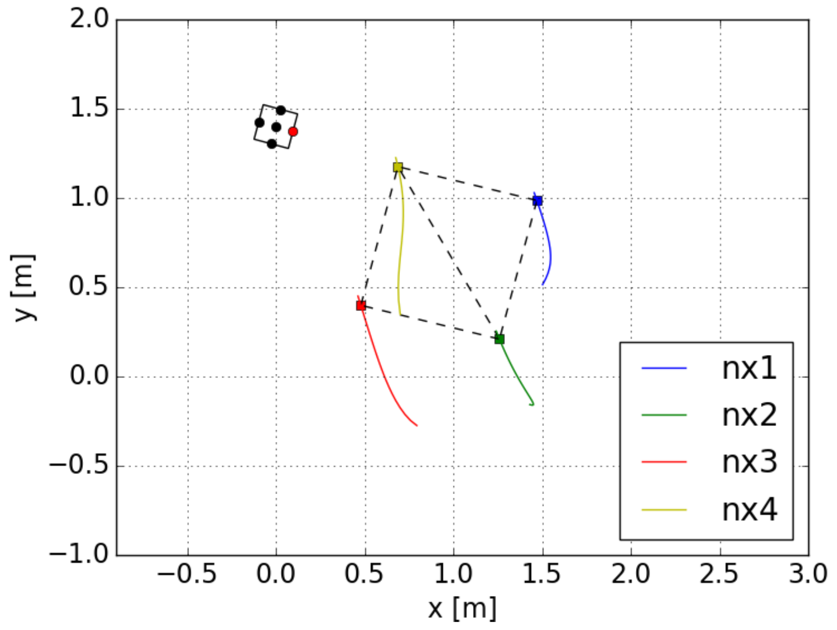
The simulation shows the trajectory of each agent and the lineal and orientation errors between the leader of the formation and the object. The leader of the formation is the closest agent to the object and the errors are between the closest face of the object and the leader of the formation. The center of the closest face is plotted as a red point while the object is plotted as a square. The different parameters used during this simulation are specified in Table 4.1. However, the simulations have also been tested with other different values to the ones specified in Table 4.1.

Parameter	**Value**
l	1
c	0.5
d_{1245}	0.8
d_3	$\sqrt{2}\, d_{1245}$
Simulation Time	60 s
Δt	0.18 s
d^*	0.3 m
Object orientation	15°
Object x-position	0 m
Object y-position	1.4 m
k_{px}	0.2
k_{ix}	0.0005
k_{py}	0.2
k_{iy}	0.0005
$k_{p\theta}$	0.005
$k_{i\theta}$	0.005

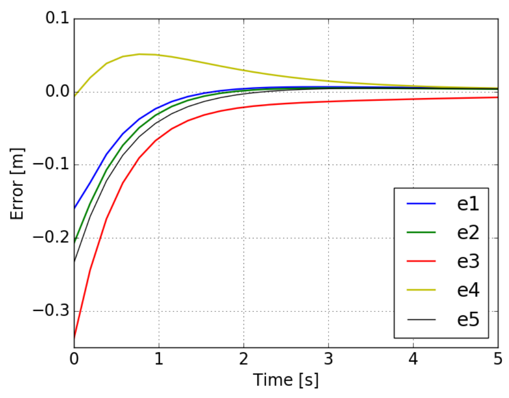
Table 4.1: Simulation parameters values used in the simulation.

4.3.1 Initialization

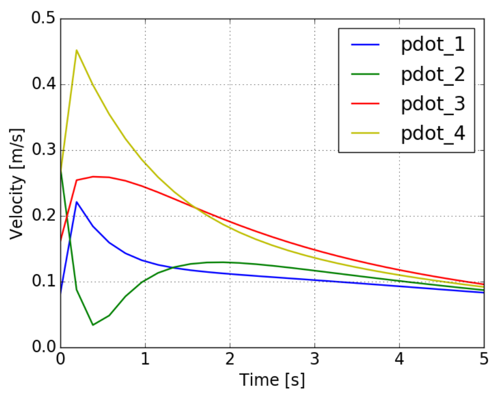
This simulation focus on the transition to the steady-state motion. Therefore, in this case, the velocity of each agent and the errors of the edges are shown in order to see how the steady-state motion is reached. Figure 4.2 also shows the trajectory of the formation towards the object.



(a) General overview of the formation during its initialization.



(b) Error of the length of each edge.



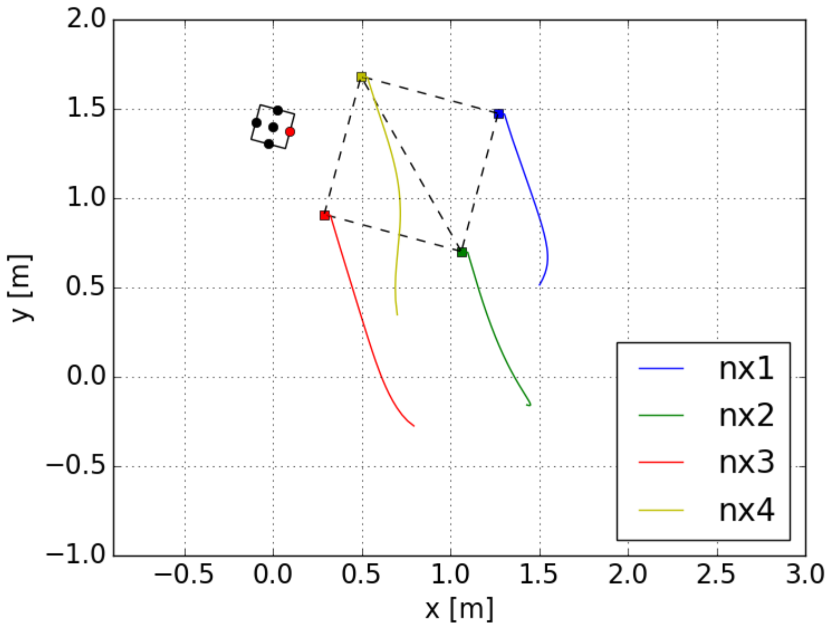
(c) Velocity of each agent.

Figure 4.2: Simulation of 5s of the initialization of the system.

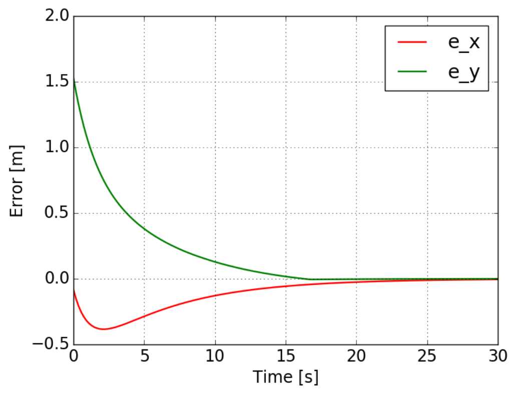
On the one hand, in Figure 4.2b, it can be seen how the error edges converges to 0 in a few seconds. On the other hand, in Figure 4.2c it is shown how the velocity of each agent converges to the desired velocity according to the inputs provided by the PI controller. It can also be proved that the velocity of each agent is never higher than 0.5 m/s and, consequently, it is not higher than the maximum velocity of the Nexus cars (0.6 m/s). Thus, because of the reasons explained above, it is proved that the steady-state motion is reached.

4.3.2 Approaching

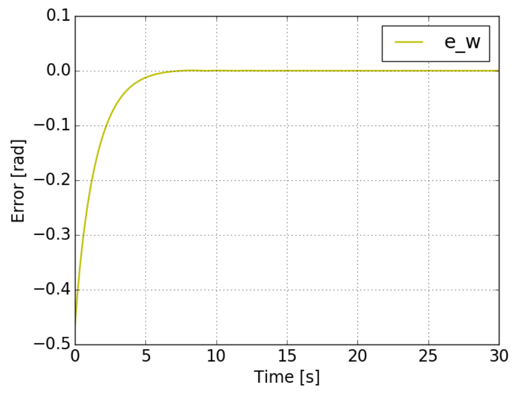
The system is approaching the object until a desired distance between the leader and the object is reached. These distances are specified in Table 3.1. Note that d^* is a parameter of the simulation and it could be modified.



(a) General overview of the formation during the approaching.



(b) Distance error in the x and y directions.



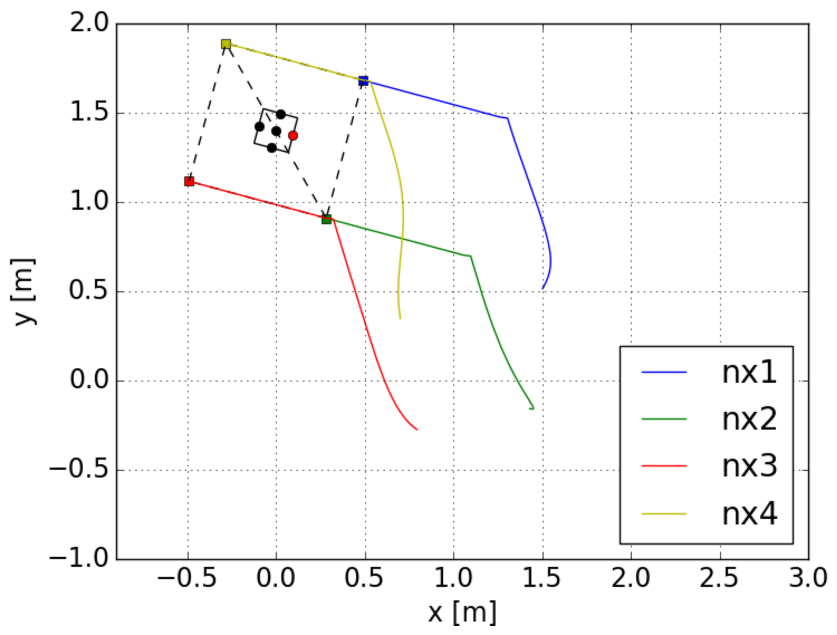
(c) Error of the orientation of the formation.

Figure 4.3: Simulation of 30s of the approaching state.

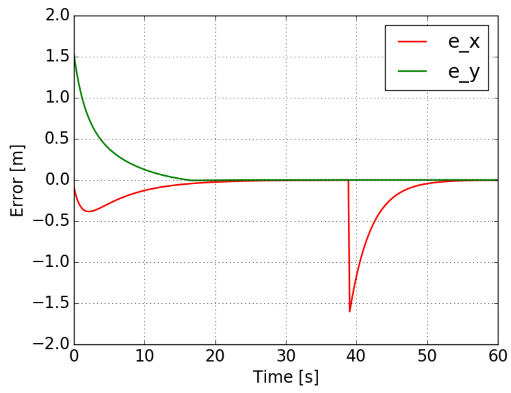
In Figures 4.3b and 4.3c, it is shown how the lineal and orientation errors converges to 0. Hence, the approaching state is completed.

4.3.3 Enclosing the target

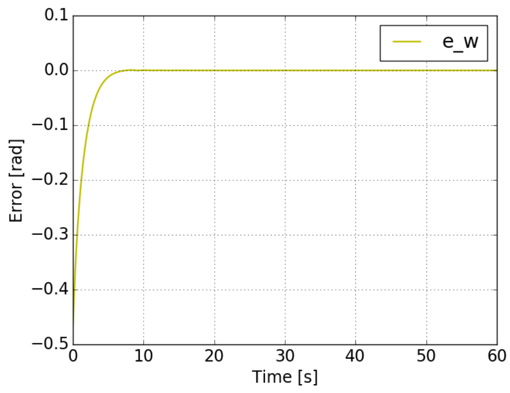
The system moves around until the object is completely centered. Note that in Figure 4.4b, around 40s, the error in the x-direction suddenly changes. This is because the approaching state has been completed and there is a new goal for the distance in the x-direction in the enclosing state.



(a) General overview of the complete simulation. The formation moves around the object until this is completely centered.



(b) Distance error in the x and y directions.



(c) Error of the orientation of the formation.

Figure 4.4: Simulation of 60s until the formation has enclosed the object.

From all the results shown above it can be concluded that the algorithm works for the initialization, approaching and enclosing states. Note that, because of the inherent ideality of the simulation, the results are not affected by any external noise.

5 Experimental setup and results

5.1 Experimental setup

In order to carry out this project several hardware and software have been used. In this section, the experimental setup will be shown.

5.1.1 Hardware (Robotic Agents)

The robotic agents used in this project are formed by a base or mobile platform, a laser scanner, a robotic arm and a camera (mounted in the same robotic arm) as shown in Figure 5.1.

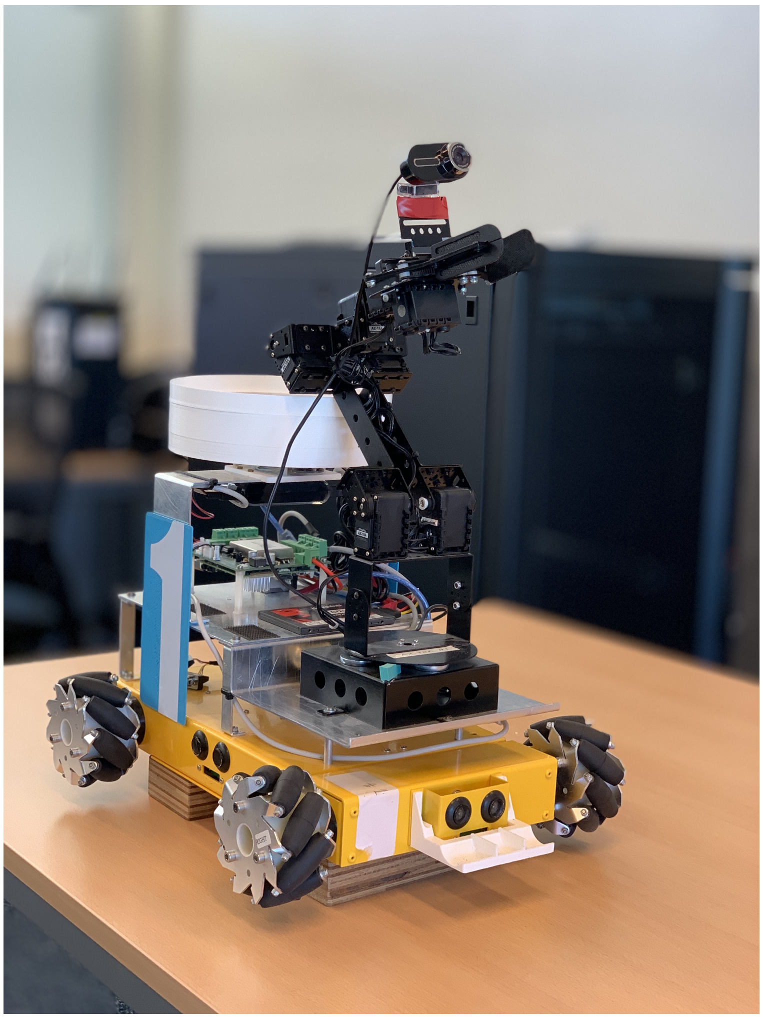
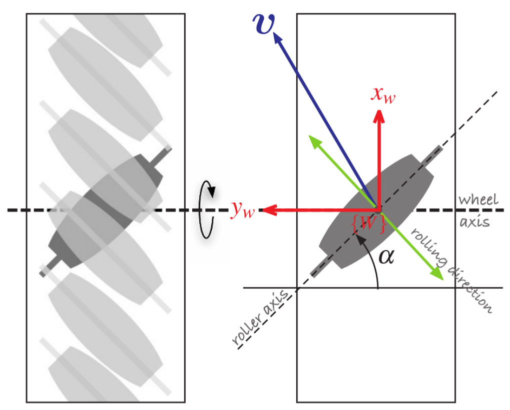


Figure 5.1: Robotic agent formed by its base (Nexus Robot), robotic arm (CrustCrawler AX-18), Laser Scanner and Camera.

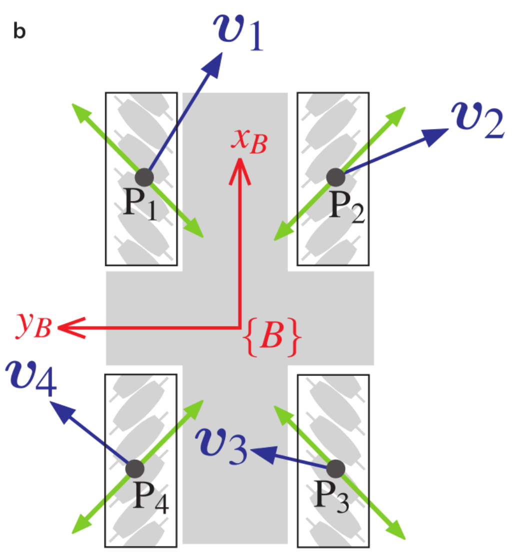
Mobile Nexus Robot

The Nexus Robot is an omnidirectional vehicle because it has omnidirectional wheels. An omnidirectional vehicle is formed by three or more omnidirectional wheels [33]. The schematic of how the wheel works is shown in Figure 5.2a

A single mecanum wheel does not allow any control in the rolling direction but for three or more mecanum wheels, suitably arranged, the motion in the rolling direction of any one wheel will be driven by the other wheels. Its pose is represented by the body frame B with its x-axis in the vehicle's forward direction and its origin at the centroid of the four wheels. A vehicle with four mecanum wheels is shown in Figure 5.2b



(a) The light rollers are on top of the wheel, the dark roller is in contact with the ground. The green arrow indicates the rolling direction.



(b) YouBot configuration.

Figure 5.2: Schematic of a mecanum wheel in plan view and schematic of a vehicle

The four wheel contact points indicated by grey dots have coordinate vectors ${}^{B}p_i$. For a desired body velocity ${}^{B}v_B$ and angular rate ${}^{B}w$ the velocity at each wheel contact point is:

$$ {}^{B}v_i = {}^{B}v_B + {}^{B}w\, \hat{z_B} \times {}^{B}p_i \qquad (5.1)$$

The different achievable behaviours of the vehicle [32] are illustrated in figure 5.3.

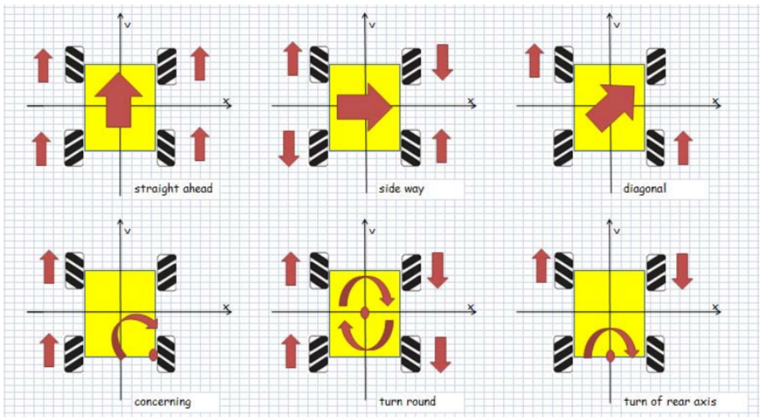


Figure 5.3: Co-effect of 4 mecanum wheels

The Robotic Arm (Crustcrawler AX-18A)

The Robotic Arm is a product from CrustCrawler Robotics [34], has 5 DoF and is equipped with 8 AX-18A servo motors (the *Shoulder Pitch Joint* and the *Elbow Flex Joint* are formed by 2 servo motors). It has an overall length of 56 centimeters and weights 1 kilogram. Every joint has a 300° range with 1024 steps (which means that the resolution is 0.29°/step = 0.00506 rad/step). The Robotic Arm consists of five joints and a gripper. In Figure 5.4 the Robotic Arm is depicted and the joints are named. The control of the arm only requires a serial connection to the computer through an USB cable and the USB2Dynamixel. All components are described in more detail on the CrustCrawler's website[1]. The different parts of the Robotic Arm have been measured obtaining the lengths shown in Table 5.1.

Part	Length [cm]
L_1	17
L_2	17
L_3	7
L_4	4
L_5	4
L_6	9

Table 5.1: Length of the different links specified in centimeters.

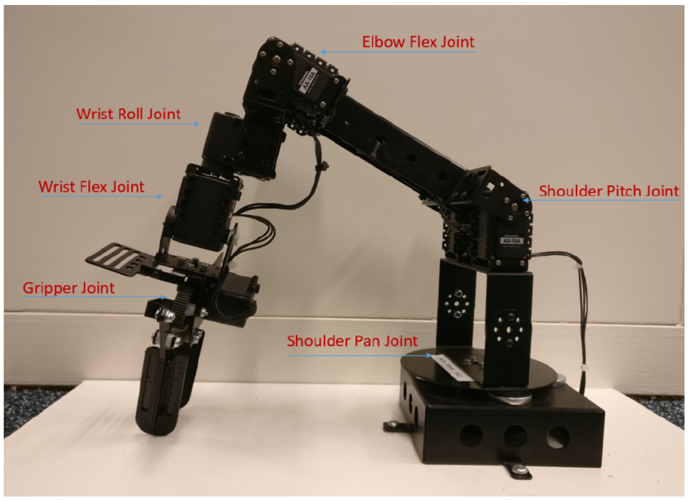


Figure 5.4: CrustCrawler AX-18A Smart Robotic Arm with joint names

In figure 5.5 the multi purpose mounting system above the gripper is shown. This system can be used, for instance, to place the webcam.

Laser Scanner

Every Nexus robot is provided with a LIDAR laser scanner. The laser produces a 2D point cloud data. This data is necessary to measure the relative distances from each

[1]CrustCrawler AX-18A, `https://www.iter.org/album/Media/7%20-%20Technical`(accessed 31 March 2019)

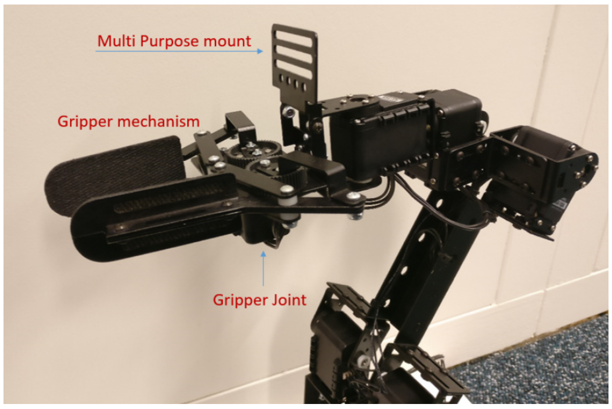


Figure 5.5: The multi purpose mounting system above the gripper

agent to their neighbors. Two important properties of the laser scanner are the maximum update frequency of 5.5Hz and the maximum rate of 360 data points per round [2].

Camera

The camera will be used in order to detect the apriltags and consequently the position of the object. In this case, the model of the camera is Philips SPC2050NC [3]. This camera has a field of view of 70°, a sensor of 2.0 MP CMOS and a maximum rate of 90 FPS. However, the real rate will be much lower due to communication limitations.

5.1.2 Software

Robot Operating System

The Robot Operating System (ROS) [35] is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

Among other concepts, it is really important to understand, at least, the following concepts:

- **Node**: Nodes are processes that perform computation. For example, one node controls a laser range-finder, one node controls the wheel motors, one node performs localization, one node performs path planning, one node provides a graphical view of the system, and so on.
- **Master**: The ROS Master provides name registration and lookup to the rest of the Computation Graph. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.
- **Messages**: Nodes communicate with each other by passing messages. A message is simply a data structure, comprising typed fields.
- **Topic**: Messages are routed via a transport system with publish / subscribe semantics. A node sends out a message by publishing it to a given topic. The topic

[2] RPLidar A1 360° Laser Range Scanner, http://www.slamtec.com/en/lidar/a1 (accessed 09 April 2019)

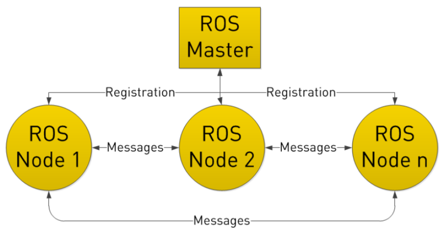
[3] Philips SPC2050NC, Philips, `https://www.philips.co.uk/c-p/SPC2050NC_00/webcam/specifications` (accessed 10 May 2019)

is a name that is used to identify the content of the message. A node that is interested in a certain kind of data will subscribe to the appropriate topic.

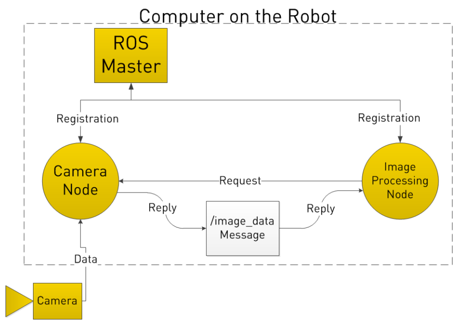
- **Services**: The publish / subscribe model is a very flexible communication paradigm, but its many-to-many, one-way transport is not appropriate for request / reply interactions, which are often required in a distributed system. Request / reply is done via services, which are defined by a pair of message structures: one for the request and one for the reply

In figure 5.6a it is shown the general working way of ROS. The Master allows all other ROS pieces of software (Nodes) to find and talk to each other. That way, it is not necessary to always specify “Send this sensor data to that computer at 127.0.0.1". It is possible to simply tell Node 1 to send messages to Node 2.

Figure 5.6b tries to emphasize in how services work, which means that there are a request and a reply message. A Node can register a specific service with the ROS Master, just as it registers its messages. In the below example, the Image Processing Node first requests */image_data*, the Camera Node gathers data from the Camera, and then sends the reply.



(a) Registration and exchange of messages between nodes.



(b) Exchange of information by a request/reply service.

Figure 5.6: ROS Overview

System overview: nodes and topics

Different code will be used in this project because of the complexity of the task. Instead of performing one big algorithm, it will be discomposed in small pieces of code in order to be more understable and easier to perform. That’s why code written by other people from the **DTPA** lab [18] [19] will be included. Figure 5.7 tries to show the different nodes (scripts) and topics used in the project.

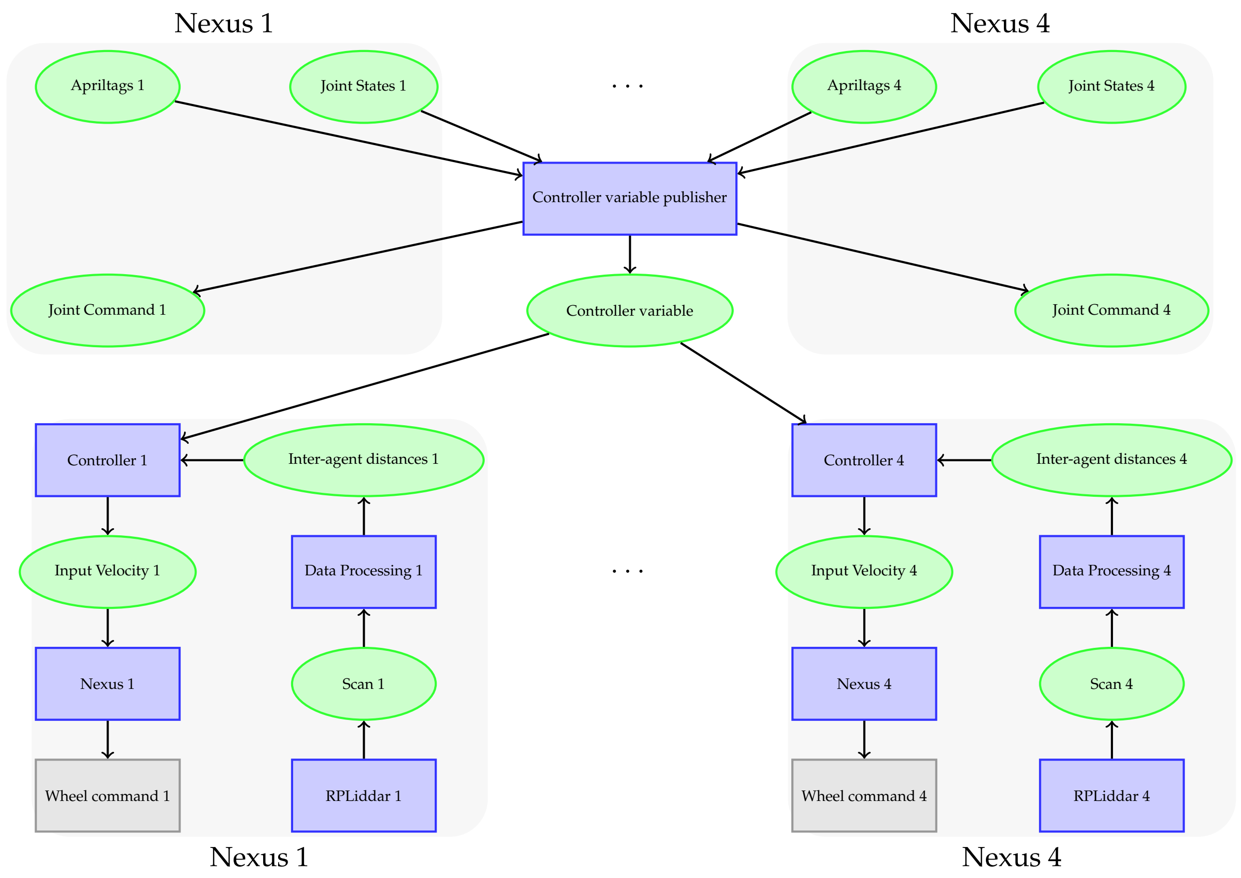


Figure 5.7: Schematic overview of the nodes (blue rectangle) and topics (green ellipses) used. For simplicity only agents 1 and 4 are shown

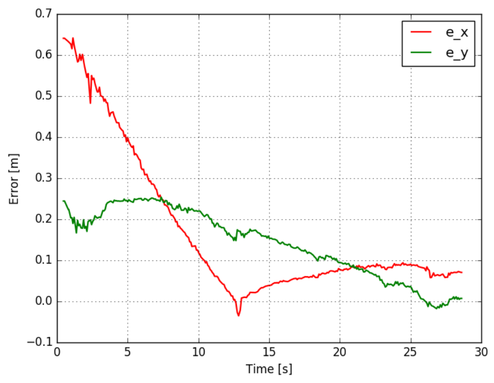
The node ***Controller variable publisher*** subscribes the topic */apriltags/detections* and the topic */JointStates* of each mobile agent i to obtain the position and orientation of the apriltag seen from the camera and the angle values of all the joints. This node is also publishing to the */JointCommand* topic of each mobile agent i the desired joint values in order to keep the tag visible. After performing the necessary calculations, the node ***Controller variable publisher*** publishes the inputs to move the formation to the topic *Controller variable*. This topic is subscribed by the controller of each agent.

The RP-lidar publishes 360 range measurements with a frequency of 5.5Hz. The data processing nodes translate these measurements into the inter-agent distances or z-values. Based on the inter-agent distances and the controller parameters, the controller node (employing control law discussed in Section 2.2.5) calculates the desired input velocities. The input velocities are translated into separate wheel commands which are communicated to the motors of the wheels. The communication to the wheels is done by software which is not in the ROS environment.

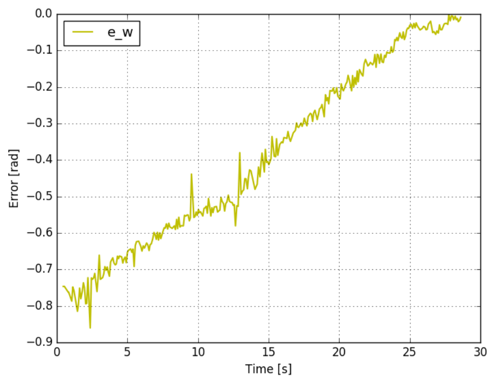
5.2 Experimental results

Even though, the idea was to implement the system with the formation, due to technical limitations, the experimental results are shown only for a single robot, being this robot the supposed leader of the formation. Therefore, it can be tested that the algorithm works and later apply it with the formation control. Moreover, the results shown below will be only for the approaching algorithm because to enclose the object four cameras (one mounted on each robot) are needed.

In the case that the robot does not detect any tag, it will start rotating until a tag is detected. Once the tag is detected, the robot will start approaching until it stops in front of him but displaced $d_k/2$, for any $k \neq 3$, to the left since the robot would be agent 1 and leader of the formation.



(a) Distance error in the x and y directions.



(b) Orientation error.

Figure 5.8: Experimental implementation of the approaching state.

In Figure 5.8a, the lineal errors (e_x, e_y) are represented. It can be seen that they both converge to 0 with a specific error margin. Because of this reason, the horizontal error (e_x) does not finally converge to 0 since the horizontal error margin is not really accurate due to there is no necessity to stop the robot at an accurate distance in the approaching algorithm. In Figure 5.8b, the orientation error (e_θ) is represented and it can also be checked that it converges to 0. From these results it can also be seen that the system is affected by some external noise.

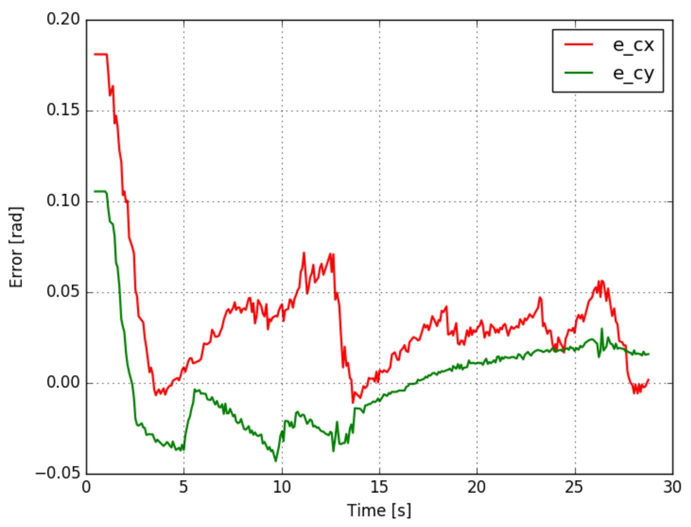


Figure 5.9: Camera centering errors during the approaching algorithm.

In Figure 5.9, it is shown the centering errors (e_{cx}, e_{cy}). The errors are conditioned by the movement of the robot and the robotic arm. Because of this reason, these errors do not converge clearly to 0 since the system needs more time to stabilize. One solution could be to move the robot slower in order to give time to the controller of the robotic arm to center completely the object and, consequently, have the errors closer to 0. Another solution could be to increase the gains of the robotic arm controller in order to move the robotic arm faster. This option led to a higher shaking of the robotic arm and, as a consequence, to sometimes lose the tag. Because of these reasons, the decision was to balance a enough fast system with a low shaking in which the tag was always visible but sometimes not perfectly centered. Moreover, the noise which appears in the signals is partially produced by the shaking phenomenon among other factors.

6 Discussion

The main goal is to perform, with a formation of four agents, a pick and place task. As explained in more detail in Section 3.1, in order to achieve this goal different states have to be reached. Unfortunately, due to technical limitations (see Section 6.2), it has not been possible to reach all these states, being the main goal partially achieved.

6.1 Evaluation of the main goal and research questions

Even though the main goal has not been completely reached due to technical problems and limitations, an approach to the goal has been done. All the research questions have been answered and the theory about the next states has been explained during the thesis giving the opportunity, in the future, to continue with this project when the technical issues are solved, being the possible implementation less complicated.

Moreover, a simulation has been performed in order to prove that the approaching and enclosing algorithms work. However, the following states have not been simulated due to the complexity of these states and the ideality of the simulation.

The real scenario has only been tested, with one robot, for the initialization, detection and approaching states since it was not possible to implement the system with a camera mounted on each agent (see Section 6.2), which is needed for the enclosing state in the real scenario.

6.2 Limitations

During the realization of the project several limitations have appeared preventing the proper performance of it. These limitations are described below.

Servos

The CrustCrawler AX-18A Smart Robotic Arm has some restrictions in order to avoid breaking the robotic arm. It basically means that each joint has a maximum torque or maximum temperature which can be reached for a specific time. Thus, because of the position of gravity center, the *Elbow Flex Joint* and, especially, *Shoulder Pitch Joint* are making a considerable effort. During the experiments this fact has been noticed preventing that these experiments last more than 1 or 2 minutes.

CPU Power

For the realization of this project a lot of peripherals, such as the camera or the laser Scanner, are used which means that a lot of computational power is required. The Nexus robot are equipped with quadcore Cortex-A9 processor which is not able to compute all calculations at the maximum possible frequency.

WiFi network

Once it was noticed that the CPU of each nexus robot was not enough to process all the data, it was tried to run the apriltags software in the PC. This led to a higher data flowing through the WiFi network. It was not a problem when only a single camera was running but with more cameras the communication was slow and unstable. The fact of not being able to have one camera on each robot prevented to implement the enclosing state and, consequently, the following states.

Camera

The current field of view of the camera limited to 70° makes the formation rotate in case the tag is not visible in the initial position. This could be fixed by buying cameras with a wider field of view. Nowadays, there are a lot of cameras even with a field of view of 360°.

7 Conclusion

This thesis aims that a formation of several agents equipped with a robotic arm performs a pick and place task by using a distributed algorithm and a formation control law for keeping the shape. This is achieved basing the algorithm on the formation control law designed by Garcia de Marina, Jayawardhana, and Cao [1]. Because of the variety of topics involved in this thesis, the challenge of merging all together becomes one of the biggest problems to overcome. Some of the pieces of this thesis have been tested, in previous work, in an isolated way working satisfactorily. However, several problems appear when the merge is performed due to the limitations explained in Section 6.2.

From the results presented in Chapters 4 and 5, it can be seen that it is possible to implement a system to achieve the main goal of this thesis, presented in Section 3.1. However, it is obvious that there is a difference between the ideal and the real world, being the real implementation harder due to, basically, communication problems caused by several reasons explained in Section 6.2.

Even though the real implementation has not been completely possible, this thesis can be useful as an starting point for improving the current system or for future work related to these topics (see Chapter 8). Theory related to the states which finally have not been implemented is also explained along the thesis in order to help other work related to the topics of this thesis.

8 Future Work

In this chapter two different ways of future work will be explained. First of all, and related to the improvement of the actual system (See Section 6.2), the system needs more powerful CPUs in order to perform the communication at the desired frequency. The solution could be either change the current CPU of each agent for a more powerful one or to implement a *Raspberry Pi* dedicated only to the apriltags processing. The second option leads to an scalable scenario, thus more cameras (and *Raspberry Pi*) can be added without affecting the nexus CPU. Moreover, in this second option, two cameras could be mounted on each agent in order to have feedback while performing the picking, moving and placing tasks since the camera mounted in the robotic manipulator is useless during these tasks.

Secondly, and also related to the improvement of the current system, some changes should be applied to the current manipulator. The thermal limit of its joints should be higher in order to perform longer experiments. This could be achieved by either changing the critical articulations for a more powerful ones or by changing the robotic manipulator. In the case of changing the manipulator some requirements described below should be taken into account:

- First of all, the thermal limits of the critical joints should be higher.
- Secondly, the fact that the robotic arm had some software prepared to perform the inverse kinematics would improve the accuracy of the picking task.
- Some force sensors in the gripper would help to balance the effort that each agent is making.
- 6 DoF would increase the reachable workspace.

The other way of future work is related to change the aspect of the actual system. One possibility, which probably would decrease the price of the system, could be to change the omnidirectional vehicle to regular ones. Thus, the agents would not be treated as simply points and the system should be rethought.

The other possibility, which would increase the flexibility and complexity of the system, could be to perform the same experiment but in a 3D space. The structure of the formation control should change and it could be tested, for instance, using drones.

References

[1] H. Garcia de Marina, B. Jayawardhana, and M. Cao, “Distributed rotational and translational maneuvering of rigid formations and their applications”, *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 684–697, Jun. 2016.

[2] D. B. Nguyen and K. D. Do, *Formation Control of Mobile Robots*. International Journal of Computers, Communications and Control, 2006.

[3] N. Chan, B. Jayawardhana, and J. Scherpen, “Distributed formation with diffusive obstacle avoidance control in coordinated mobile robots”, in *57th IEEE Conference on Decision and Control*, IEEE, Jan. 2019, pp. 4571–4576, ISBN: 978-1-5386-1395-5.

[4] G. Antonelli, F. Arrichiello, F. Caccavale, and A. Marino, “Decentralized time-varying formation control for multi-robot systems”, *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1029–1043, 2014.

[5] T. Mylvaganam, M. Sassano, and A. Astolfi, “A differential game approach to multi-agent collision avoidance”, *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 4229–4235, Aug. 2017.

[6] Y. Lan, G. Yan, and Z. Lin, “Distributed control of cooperative target enclosing based on reachability and invariance analysis”, *Systems & Control Letters*, vol. 59, no. 7, pp. 381–389, 2010.

[7] J. Yuan, Y. Huang, T. Tao, and F. Sun, “A cooperative approach for multi-robot area exploration”, pp. 1390–1395, 2010.

[8] S. Mastellone, D. M. Stipanović, C. R. Graunke, K. A. Intlekofer, and M. W. Spong, “Formation control and collision avoidance for multi-agent non-holonomic systems: Theory and experiments”, *The International Journal of Robotics Research*, vol. 27, no. 1, pp. 107–126, 2008.

[9] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999, ISBN: 0201360489.

[10] K.-K. Oh, M.-C. Park, and H.-S. Ahn, “A survey of multi-agent formation control”, *Automatica*, vol. 53, pp. 424–440, 2015.

[11] Z. Meng, B. D. Anderson, and S. Hirche, “Formation control with mismatched compasses”, *Automatica*, vol. 69, pp. 232–241, 2016.

[12] M. A. Lewis and K.-H. Tan, “High precision formation control of mobile robots using virtual structures”, in. Autonomous Robots, 1997, vol. 4.4, pp. 387–403.

[13] B. Siciliano, “Robotics”, in. Springer, 2009, pp. 4–10.

[14] I. F. Vis, “Survey of research in the design and control of automated guided vehicle systems”, *European Journal of Operational Research*, vol. 170, no. 3, pp. 677–709, 2006.

[15] Amazon, *Amazon robotics*, https://www.amazonrobotics.com, Accessed on 01-05-2019.

[16] A. R. Hevner, “A three cycle view of design science research”, *Scandinavian Journal of Information Systems*, vol. 19, 2007.

[17] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research", in. MIS Quarterly, 2004, vol. 28(1), pp. 75–105.

[18] J. Siemonsma, *Distributed translational and rotational control of rigid formations and its applications in industries*. RUG, 2017.

[19] J. Buursma, *Pick and place algorithms for mobile manipulators with camera feedback*. RUG, 2018.

[20] R. Freeman and D. Reed, "Stockholders and stakeholders: A new perspective on corporate governance", *California Management Review*, vol. 25, Apr. 1983.

[21] J. J. Craig, "Introduction to robotics", in. Pearson Education International, 2005, pp. 62–76.

[22] J. J. Craig, *Introduction to Robotics*. Pearson Education International, 2005.

[23] H. Garcia de Marina, M. Cao, and B. Jayawardhana, "Controlling rigid formations of mobile agents under inconsistent measurements", *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 31–39, 2015.

[24] L. Asimow and B.Roth, "The rigidity of graphs, ii", in. J.Math.Anal.Appl., 1979, pp. 171–190.

[25] B. Anderson, C. Yu, B.Fidan, and J. Hendrickx, "Rigid graph control architectures for autonomous formations", in. IEEE Control Syst. Mag, 2008, pp. 48–63.

[26] L. Henneberg, *Die graphische Statik der starren Systeme*. B. G. Teubner, 1911.

[27] H. Bai, M. Arcak, and J. Wen, *Cooperative Control Design*. Springer-Verlag New York, 2011.

[28] K. Oh and H. Ahn, "Distance-based undirected formations of single-integrator and double-integrator modeled agents in n-dimensional space", in. Int. J. Robust Nonlinear Control, 2014, vol. 24, pp. 1809–1820.

[29] P. A. Absil and K. Kurdyka, "On the stable equilibrium points of gradient systems", in. Systems & Control Letters, 2006, vol. 55, pp. 573–577.

[30] H. Goldstein, *Classical mechanics*. Addison-Wesley, 1951.

[31] K. J. Astrom and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008, pp. 293–294.

[32] N. Robot, *4wd 100mm mecanum wheel robot kit 10011*, https://www.nexusrobot.com/product/4wd-mecanum-wheel-mobile-arduino-robotics-car-10011.html, Accessed on 31-03-2019, 2012.

[33] P. Corke, "Robotics, vision and control", in. Springer, 2017, pp. 112–113.

[34] Unknown, *AX-12A / AX18A Smart Robotic Arm Specifications / Advantages*. Crustcrawler Inc., 2015.

[35] O. S. R. Foundation, *Ros website*, http://www.ros.org, Accessed on 01-04-2019.

[36] B. Siciliano, "Robotics", in. Springer, 2009, pp. 54–56.

[37] J. J. Craig, "Introduction to robotics", in. Pearson Education International, 2005, pp. 41–46.

Appendices

A How to run the code for a single robot

In order to run the code, download the workspace used for this project. After downloading the workspace follow the steps described below:

1. Preparation
   (a) Go to the home folder of the catkin workspace and run the following commands:
   catkin_make
   source devel/setup.bash
2. Initialization
   (a) Open one terminal and type *roscore*.
   (b) Open another terminal and connect it to the nexus: *ssh -l dtpa nexus1*. After that, in this same terminal, launch the Nexus and the arm:
   roslaunch nexus named_nexus_arm_laser.launch sim := false
   (c) In order to run the camera software, open another terminal and connect it to the ROS account of the nexus car: *ssh ros@nexus1* and launch the camera by running the following command: *source sap1*.
   (d) Open another terminal and launch the apriltags software in the computer: *roslaunch apriltags usb_cam_apriltags.launch*
3. Run the code
   (a) Run the initialization script: *rosrun some_tests init_marc.py*
   (b) Stop the script when the initialization is completed by typing *ctrl* + *C*.
   (c) Run the main script: *rosrun some_tests finalcode_marc.py*

B DH-Convetion

This section pretends to help to understand how the frame-coordinates are assigned and how get the DH-parameters. The explanation given here is based on the one explained in [21].

B.1 Convention for affixing frames to links

In order to describe the location of each link relative to its neighbors, a frame attached to each link is defined. The link frames are named by number according to the link to which they are attached. That is, frame $\{i\}$ is attached rigidly to link i.

B.2 Intermediate links in the chain

The convention used to locate frames on the links is as follows: The Z-axis of frame $\{i\}$, called Z_i, is coincident with the joint axis i. The origin of frame $\{i\}$ is located where the a_i perpendicular intersects the joint i axis. X_i points along a_i in the direction from joint i to joint $i+1$.

In the case of $a_i = 0$, X_1 is normal to the plane of Z_i and Z_{i+1}. α_i is defined as being measured in the right-hand sense about X_i and so we see that the freedom of choosing the sign of α_i in this case corresponds to two choices for the direction of X_i. Y_i is formed by the right-hand rule to complete the ith frame. Figure B.1 represents a Flow-Chart in order to assign the frames to links.

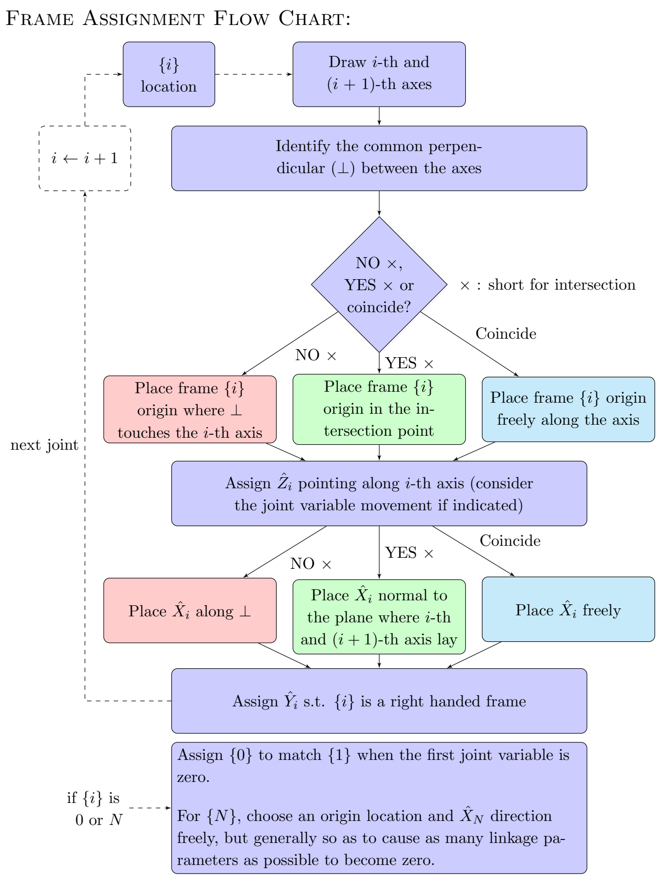

FRAME ASSIGNMENT FLOW CHART:
$\{i\}$ location
Draw i-th and $(i+1)$-th axes
$i \leftarrow i+1$
Identify the common perpendicular ($\perp$) between the axes
NO $\times$, YES $\times$ or coincide?
$\times$: short for intersection
NO $\times$
YES $\times$
Coincide
Place frame $\{i\}$ origin where $\perp$ touches the i-th axis
Place frame $\{i\}$ origin in the intersection point
Place frame $\{i\}$ origin freely along the axis
next joint
Assign $\hat{Z}_i$ pointing along i-th axis (consider the joint variable movement if indicated)
NO $\times$
YES $\times$
Coincide
Place $\hat{X}_i$ along $\perp$
Place $\hat{X}_i$ normal to the plane where i-th and $(i+1)$-th axis lay
Place $\hat{X}_i$ freely
Assign $\hat{Y}_i$ s.t. $\{i\}$ is a right handed frame
if $\{i\}$ is 0 or N
Assign $\{0\}$ to match $\{1\}$ when the first joint variable is zero.
For $\{N\}$, choose an origin location and $\hat{X}_N$ direction freely, but generally so as to cause as many linkage parameters as possible to become zero.

Figure B.1: Frame assignment flow chart created by Carlo Cenedese.

B.3 Link parameters in terms of the links frames

If the link frames have been attached to the links according to the DH convention, the following definitions of the link parameters are valid:

a_i = *the distance from* Z_i *to* Z_{i+1} *measured along* X_i

α_i = *the angle from* Z_i *to* Z_{i+1} *measured about* X_i

d_i = *the distance from* X_{i-1} *to* X_i *measured along* Z_i

θ_i = *the angle from* X_{i-1} *to* X_i *measured about* Z_i

Usually a_i is chosen $a_i > 0$, because it corresponds to a distance; however, α_i, d_i and θ_i are signed quantities.

C Forward kinematics calculations

The equations listed below are the transformations matrices for all the joints of the robotic manipulator:

$$ {}^0_1T = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 17 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

$$ {}^1_2T = \begin{bmatrix} \cos(\theta_2 - \pi/2) & -\sin(\theta_2 - \pi/2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin(\theta_2 - \pi/2) & -\cos(\theta_2 - \pi/2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \sin(\theta_2) & \cos(\theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \cos(\theta_2) & -\sin(\theta_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

$$ {}^2_3T = \begin{bmatrix} \cos(\theta_3 + \pi/2) & -\sin(\theta_3 + \pi/2) & 0 & 17 \\ -\sin(\theta_3 + \pi/2) & -\cos(\theta_3 + \pi/2) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\sin(\theta_3) & -\cos(\theta_3) & 0 & 17 \\ -\cos(\theta_3) & \sin(\theta_3) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

$$ {}^3_4T = \begin{bmatrix} \cos(\theta_4 + \pi/2) & -\sin(\theta_4 + \pi/2) & 0 & 4 \\ 0 & 0 & -1 & -11 \\ \sin(\theta_4 + \pi/2) & \cos(\theta_4 + \pi/2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\sin(\theta_4) & -\cos(\theta_4) & 0 & 4 \\ 0 & 0 & -1 & -11 \\ \cos(\theta_4) & -\sin(\theta_4) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4} $$

$$ {}^4_5T = \begin{bmatrix} \cos(\theta_5) & -\sin(\theta_5) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin(\theta_5) & \cos(\theta_5) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5} $$

$$ {}^5_6T = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 9 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6} $$

However, since the camera is not directly in center with the fifth frame, some other transformations have to be performed. First a translation over the Z-axis, a rotation over the X-axis of -90 degrees and another translation in the Z-axis. These transformation are listed below:

$$ {}^{5}_{C}T_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7} $$

$$ {}^{5}_{C}T_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-\pi/2) & \sin(-\pi/2) & 0 \\ 0 & \sin(-\pi/2) & \cos(-\pi/2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8} $$

$$ {}^{5}_{C}T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9} $$

Now, the total transformation from the camera to the fifth frame can be calculated as shown below:

$$ {}^{5}_{C}T = {}^{5}_{C}T_1\, {}^{5}_{C}T_2\, {}^{5}_{C}T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 48 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10} $$

D Inverse kinematics calculations

First of all, the following relation can be immediately obtained:

$$ \tan\theta_1 = \frac{y}{x} \Rightarrow \theta_1 = \arctan 2(y, x) \tag{11} $$

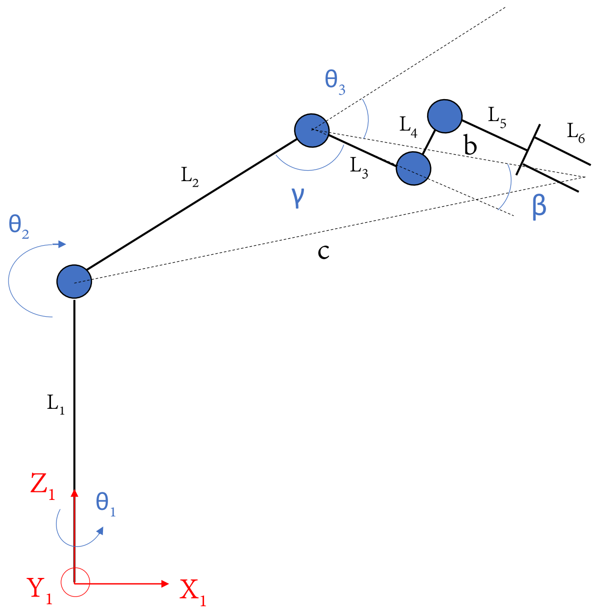


Figure D.2: Simplified version of the robotic arm.

After that, according to he following equations, θ_3 can be obtained applying the law of cosines [1]:

$$\theta_3 + \beta + \gamma = 180 \tag{12}$$

$$\cos\gamma = \frac{{L_2}^2 + b^2 - c^2}{2bL_2} \tag{13}$$

where:

$$b = \sqrt{{L_4}^2 + (L_3 + L_5 + L_6)^2} \tag{14}$$

$$c = \sqrt{x^2 + y^2 + (z - L_1)^2} \tag{15}$$

The values of b and c are the distances from the gripper to the joints 2 and 3 respectively. Finally, with the two next equations θ_3 can be computed:

$$\beta = \arctan 2(L_4, L_3 + L_5 + L_6) \tag{16}$$

$$\cos(\theta_3 + \beta) = \frac{c^2 - {L_2}^2 - b^2}{2L_2 b} \tag{17}$$

[1]Law of cosines, Wikipedia, `https://en.wikipedia.org/wiki/Law_of_cosines` (accessed 27 March 2019)

According to the equation shown in (2.7) and the angle sum and difference identities [2], assuming $\theta_4 \equiv \theta_5 \equiv 0$ and taking into account following notation:

$$^0p_6 = \begin{bmatrix} c_1((L_3+L_5+L_6)s_{2+3} - L_4c_{2-3} + L_2s_2) \\ s_1((L_3+L_5+L_6)s_{2+3} - L_4c_{2-3} + L_2s_2) \\ L_1 + (L_3+L_5+L_6)c_{2-3} + L_4s_{2+3} + L_2c_2 \end{bmatrix} \tag{18}$$

where:

$$cos(\theta_i) \equiv c_i \; and \; sin(\theta_i) \equiv s_i \tag{19}$$

So, the following relation can performed:

$$b_1 = \pm\sqrt{x^2+y^2} = s_2(L_2 - L_4s_3 + (L_3+L_5+L_6))c_3) - c_2(L_4c_3 + (L_3+L_5+L_6)s_3) \tag{20}$$

$$b_2 = z - L1 = c_2(L_2 - L_4s_3 + (L_3+L_5+L_6))c_3) + s_2(L_4c_3 + (L_3+L_5+L_6)s_3) \tag{21}$$

Which can be written as:

$$b_1 = k_1s_2 - k_2c_2 \tag{22}$$

$$b_2 = k_2s_2 + k_1c_2 \tag{23}$$

Where $k_1 = L_2 - L_4s_3 + (L_3+L_5+L_6))c_3$ and $k_2 = L_4c_3 + (L_3+L_5+L_6)s_3$
The equations shown before can be represented in its matrix as shown below:

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} k_1 & -k_2 \\ k_2 & k_1 \end{bmatrix} \begin{bmatrix} s_2 \\ c_2 \end{bmatrix} \Rightarrow \begin{bmatrix} s_2 \\ c_2 \end{bmatrix} = \begin{bmatrix} k_1 & -k_2 \\ k_2 & k_1 \end{bmatrix}^{-1} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{24}$$

Finally, we get:

$$\theta_2 = \arctan 2(s_2, c_2) \tag{25}$$

E Orientation of the apriltags seen from the base

During the experiments related with the orientation it was noticed that the apriltags had a different frame from the one from the sensor of the camera. This is shown in Figure E.3.

[2]Angle sum and difference identities, Wikipedia, `https://en.wikipedia.org/wiki/List_of_trigonometric_identities` (accessed 27 March 2019)

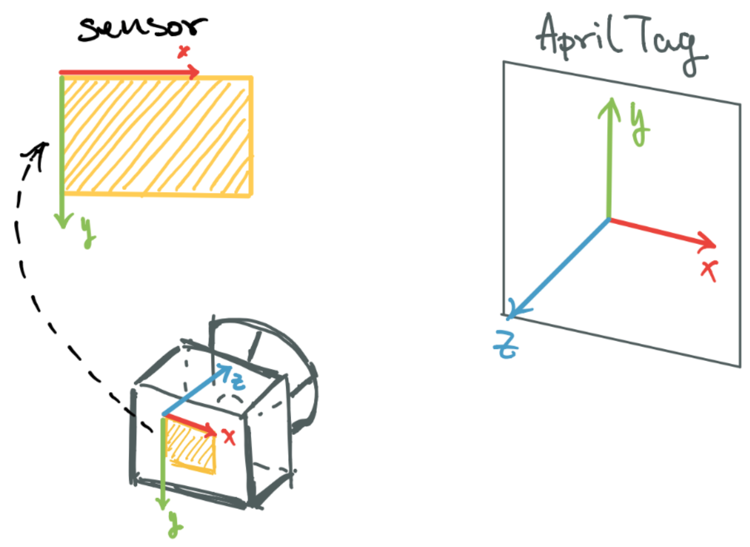


Figure E.3: Camera sensor and apriltags frame.

From Figure E.3, it can also be seen that between the camera (C) and apriltag (A) frames there is a rotation of 180° about the x axis, being the associated rotation matrix

$$ {}^{C}_{A}R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\pi) & -\sin(\pi) \\ 0 & \sin(\theta_1) & \cos(\pi) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{26} $$

From the apriltags software, the orientation is given as a quartenion in the apriltags frame (A). A quartenion is defined as $\mathcal{Q} = \{\eta, \epsilon\}$ where η is called the scalar part of the quaternion while $\epsilon = [\epsilon_x\ \epsilon_y\ \epsilon_z]^T$ is called the vector part of the quaternion [36]. The associated rotation matrix to the given quartenion in the apriltags frame (A) that describes the orientation (O) is calculated as follows:

$$ {}^{A}_{O}R = R(\eta, \epsilon) = \begin{bmatrix} 2(\eta^2 + \epsilon_x^2) - 1 & 2(\epsilon_x\epsilon_y - \eta\epsilon_z) & 2(\epsilon_x\epsilon_z + \eta\epsilon_y) \\ 2(\epsilon_x\epsilon_y + \eta\epsilon_z) & 2(\eta^2 + \epsilon_y^2) - 1 & 2(\epsilon_y\epsilon_z - \eta\epsilon_x) \\ 2(\epsilon_x\epsilon_z - \eta\epsilon_y) & 2(\epsilon_y\epsilon_z + \eta\epsilon_x) & 2(\eta^2 + \epsilon_z^2) - 1 \end{bmatrix} \tag{27} $$

Thus, the rotation matrix that describes the orientation of the apriltags relative to the camera is

$$ {}^{C}_{O}R = {}^{C}_{A}R \, {}^{A}_{O}R \tag{28} $$

Then, the rotation matrix which describes the orientation of the apriltags relative to the base is

$$ {}^{0}_{O}R = {}^{0}_{C}R \, {}^{C}_{O}R \tag{29} $$

After performing the calculation described above, the euler angles are used in order to get an easier reading of the orientation [37]. Being

$$ {}^{A}_{B}R_{ZYX}(\alpha, \beta, \gamma) = R_Z(\gamma)R_Y(\beta)R_X(\alpha) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{30} $$

the rotation matrix which gives the orientation of (B) relative to (A). The euler angles can be calculated as

$$\beta = Atan2(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}) \tag{31}$$

$$\alpha = Atan2(r_{21}/c\beta, r_{11}/c\beta) \tag{32}$$

$$\gamma = Atan2(r_{32}/c\beta, r_{33}/c\beta) \tag{33}$$