Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Control System Engineering

Collaborative Transportation with Mobile Manipulators and Human interaction

Master Candidate

Achille Policante

Student ID 2005825

Supervisor

Prof. Alessandro Chiuso

University of Padova

Co-supervisor

Dr. Pouria Tajvar

Royal institute of Technology

Academic Year

2021/2022

Abstract

Nowadays multi-robot systems (MRS) are becoming more and more popular in the industrial field taking advantage of its improved robustness and efficiency compared to single-robot systems. In fact, many tasks, such as object transportation, can benefit from MRS's flexibility and reliability to extend the load-carrying capacity. Above all, mobile manipulators, essentially any type of arm mounted in a base capable to move in the ground, are the most suitable to emulate human behaviour since permit to take, carry and leave any type of object without substantial changes in the environment. In this work, a centralized controller for three mobile manipulators is proposed to provide the reference velocities for each arm grasping the object and a fixed formation for the bases is exploited to carry and keep it in the center. The object position displacement w.r.t. the initial position is used to move the bases of the manipulators in the direction of the displacement. In this way, the higher controller only has to move the object w.r.t. the bases to control the velocity of the entire formation. Furthermore, this approach allows a human to directly command the formation, by acting as the higher controller, pushing or pulling the object. Experiments using Hebi Rosies mobile manipulators are demonstrated in order to validate the theory.

Acknowledgements

I'm extremely grateful to professor Chiuso for giving me the opportunity to begin this wonderful experience and for introducing me to the prestigious Royal Institute of Technology in Sweden.

Here I could not have undertaken this journey without professor Dimorogonas, my examiner and head of the Division of Decision and Control Systems in KTH.
I'd also like to express my deepest appreciation to dr. Tajvar, who, as my supervisor, provides me with all of the tools, papers, and methodologies I need to complete the assignment.

Special thanks to my girlfriend Sophia, my parents Renato and Lorena and my sister Anita. Their belief in me has kept my spirits and motivation high throughout this project.

Lastly, I'd like to mention my colleague, friend and flatmate Gianmarco for his participation in this work, which included helpful advice, feedback sessions, and moral support.

Acronyms

MAS	Multi Agent System
MPC	Model Predictive Control
LQR	Linear Quadratic Regulator
LQGR	Linear Quadratic Gaussian Regulator
DOF	Degree of Freedom
QP	Quadratic Problem
APF	Artificial Potential Field
EE	End-Effector
ROS	Robotic Operating System
MCS	Motion Capture System
PWM	Pulse Width Modulation

Contents

Chapter 1

Introduction and State of the Art

Cooperation and coordination of multi-robot systems have been the focus of significant research efforts in recent years. It is becoming more and more common to substitute huge and expensive robots with multiple tiny ones, whose production can be standardized in order to decrease costs [13]. An example can be found in [14], where Multi Agent System (MAS) is applied to a real-world production line for automotive transmission forks. Besides this, multi-robot systems can perform tasks more efficiently than a single robot or accomplish tasks not executable by a single one [4]. Moreover, other advantages can also be supplied like increasing tolerance to possible vehicle fault, providing flexibility to the task execution, or taking advantage of distributed sensing and actuation. One area that can benefit from these improvements is the transportation manufacturing industry, in which a single mobile robot can no longer meet the requirements of some tasks in many cases [32].
In summary, there are two distinct categories of cooperative transportation: pushing only strategies and grasping strategies, where both pushing and pulling are allowed [26]. The first one usually is characterized by cheaper hardware, since the mobile robotic bases don't need to be equipped with expensive and precise robotic arms and End-Effector (EE)s. Nevertheless, a more sophisticated controller must be designed to keep contact with the transported object by the robots. A novel control architecture is discussed in [19], in

which cluster space control to maintain formation and explicit force control are combined to transport an object. The proposed architecture was shown to be capable of dynamically controlling the desired locations of the robots relative to the box to improve the ability to turn. The second method, on the other hand, involves employing certain EEs to grab the object. In this way, the object is always attached to each arm and it's easier to develop a controller to manipulate it. This is the approach that will be adopted for this project. Other works that resort to this approach are [30] and [29], where a push-pull strategy based on force control is presented.

Another aspect that must be considered concerns the communication between the robots, which can be full, partial, or zero. In the third case is appropriate to refer to it as implicit communication, in which the robots can't directly send messages or share their own position with the others, but is possible to structure a communication based on the movements of the other robots [22],[30].

Exploiting the same advantages of multiple robots w.r.t. a single one, the control and estimation can be split too. Dividing in this way between all the robots, the computational power increases, and robustness is improved. These approaches are named decentralized and in [1] and [25] two different implementations are showed.

First of all, in order to move the entire formation, a high-level controller is required to find a path to reach the target position in which the object should be offloaded. In [28] and [31] an Artificial Potential Field (APF) method is applied in order to resolve the path planning problem. In APF, obstacles and the robots are modeled as electrical charges of the same sign, instead, the target position is considered as a charge with the opposite sign. In this way is possible to find a path to the target point avoiding collisions with obstacles, following the gradient of the potential field. It is shown in [27] how this concept can be applied to collaborative transportation. However, this approach is only able to find local minima and so stuck phenomena can arise frequently. To overcome this issue many techniques can be exploited, such as adding a virtual obstacle in presence of local minima to escape from there [21].

Instead of minimizing an artificial electric field, other techniques are based

on finding an optimal control input w.r.t. a cost function, as presented in [12]. Moreover, the cost function can be subject to constraints, equalities or inequalities that must be satisfied inside the minimization problem. These can be related to limitations of the maximum velocities of the robots or can be established to avoid collisions with the obstacles. On the other hand, this technique is very sensitive to uncertainties in the knowledge of the robot's model and object's parameters. Furthermore precise and expensive force sensors are required in a real implementation.
The proposed controller is based on resolving some optimization problems too, but resorting to a pipeline control strategy based on many Model Predictive Control (MPC)s that work in cascade.
As well as becoming faster, stronger, and less expensive, the next generation of robots has to take on a new challenge: working in direct contact with humans [20].In fact, due to the elderly population dominance of most industrialized nations, the desire to automate routine daily tasks, and the scarcity or high cost of local human expertise, the application domains for robotics are expanding from factories to human environments.
Figure 1.0.1 shows different scenes of collaboration and interaction between robots and an operator. In this work, the presence of a human operator is exploited to control the entire formation through small movements of the object, while this is lifted by the robots.

1.1 Problem

First, a centralized controller that coordinates the three robots moving them as a single unit while manipulating the object to be transported must be constructed in order to facilitate cooperation amongst the robots. The focus of this work's second section will be on information exchange, specifically between robots and a human operator. The literature describes various methods of robot-robot communication, as was mentioned in the introduction. This is essential while working alongside humans and operating outside of manufacturing lines. However, it is still difficult to establish an implicit non-verbal communication

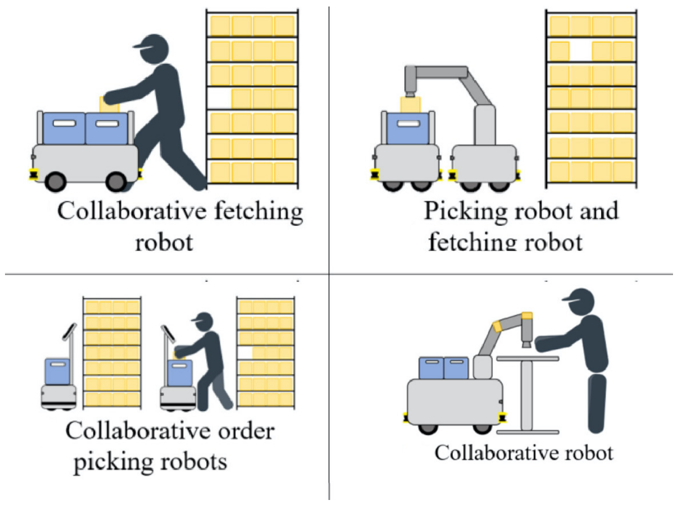

Collaborative fetching robot
Picking robot and fetching robot
Collaborative order picking robots
Collaborative robot

Figure 1.0.1: Different types of collaborative transportation and human interaction [6]

between a robot and an operator. In [11] the role of learning is exploited. To summarize, an answer to the following question will be presented in this work: *"How can the human operator coordinate the robots to move together and transmit directions to the formation in order to move each robot and the object being carried?"*

1.2 Outline

In this section, a brief explanation of the work's structure is presented. Starting from section 2, formulas, theorems, and proofs will be covered in detail to provide basic theoretic knowledge to the reader in order to understand the next computations. In particular, two theoretical areas will be summarized. The first one concerns the derivation of the arm dynamics and how to use that to develop a controller for the arm in the Cartesian space. In the second part, the MPC strategy and the related proof of stability will be presented.
Section 3 provides an overview of the Robotic Operating System (ROS) environment, the models of the mobile base and the arm, and the real hardware specifications of the motors adopted in the experimental setup. A low-level control strategy related to the robotic arm, with proof of stability, will be also presented.

In Section 4 the proposed controller, with a pipeline structure, is explained, resorting also to the real implementation in the ROS environment.
Section 5 will cover all the appropriate experiments, useful to validate some aspect of the controller in its part, to get to the end of the complete test in which the whole controller is working. Other experiments are conducted to investigate human collaboration and obstacle avoidance. The last section 6 will conclusively provide a short recap of the work, leading to a discussion of the results and to some consideration and improvements for future research.

Chapter 2

Theoretical Background

In this chapter, a detailed description of the degree project's background is presented. Relevant theorem and proof, useful to better understand the basis of the controller, are provided too.
In the first part, a description of the arm dynamics is provided and will be used to develop a controller. A review of Lyapunov stability theory is also presented since adopted in 3 to prove the stability of the proposed arm controller. Instead, the second section offers the MPC strategy's foundations, describing the benefits w.r.t. classical state-space-model-based controllers and PIDs.

2.1 Arm dynamics

The generic dynamical model of the i th robotic arm, with n_i Degree of Freedom (DOF), can be formulated, according to Newton Lagrange Formulation, as the following :

$$\boldsymbol{M}_i(q_i)\ddot{q}_i + \boldsymbol{C}_i(q_i, \dot{q}_i)\dot{q}_i + \boldsymbol{F}_i\dot{q}_i + g_i(q_i) = \tau_i - \boldsymbol{J}_{A-i}^T(q_i)f_{e-i} \tag{2.1}$$

where $q_i \in \mathbb{R}^{n_i}$is the joint position vector, $\tau_i \in \mathbb{R}^{n_i}$is the joint torque vector, $\boldsymbol{M_i}(q_i) \in \mathbb{R}^{n_i} \times \mathbb{R}^{n_i}$ is the symmetric positive definite inertia matrix, $\boldsymbol{C_i}(q_i, \dot{q}_i) \in \mathbb{R}^{n_i} \times \mathbb{R}^{n_i}$ is the centrifugal and Coriolis terms matrix, $\boldsymbol{F_i}(q_i) \in \mathbb{R}^{n_i} \times \mathbb{R}^{n_i}$ is the

matrix modeling viscous friction, $g_i(q_i) \in \mathbb{R}^{n_i}$ is the vector of gravity terms, and $f_{e-i} \in \mathbb{R}^6$ is the vector of interaction forces between the robot's end-effector and the environment.
From this time forth, since all the arms have the same structure and so apparently identical, considering that also the dynamics of each doesn't affect the behaviour of the others, the i subscript can be neglected and a generic arm can be considered for the future discussions.
For the purpose of this work, the most important matrix to be evaluated is the analytical Jacobian $\boldsymbol{J_A}(q)$ that can be derived as:

$$x = k(q) = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \psi \\ \theta \end{bmatrix} \quad \boldsymbol{J_A}(q) = \frac{\partial k(q)}{\partial q} = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \frac{\partial x}{\partial q_3} & \cdots & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \frac{\partial y}{\partial q_3} & \cdots & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \frac{\partial z}{\partial q_3} & \cdots & \cdots & \frac{\partial z}{\partial q_n} \\ \frac{\partial \phi}{\partial q_1} & \frac{\partial \phi}{\partial q_2} & \frac{\partial \phi}{\partial q_3} & \cdots & \cdots & \frac{\partial \phi}{\partial q_n} \\ \frac{\partial \theta}{\partial q_1} & \frac{\partial \theta}{\partial q_2} & \frac{\partial \theta}{\partial q_3} & \cdots & \cdots & \frac{\partial \theta}{\partial q_n} \\ \frac{\partial \psi}{\partial q_1} & \frac{\partial \psi}{\partial q_2} & \frac{\partial \psi}{\partial q_3} & \cdots & \cdots & \frac{\partial \psi}{\partial q_n} \end{bmatrix} \tag{2.2}$$

This matrix is fundamental since provides the direction in the joint space $\mathbb{R}^{n_i}$ that minimizes the error between the actual EE position and the target one in the Cartesian space.
Applying Taylor expansion to the previous:

$$x^d = k(q) = \lim_{q^k \to q} k(q^k) + \boldsymbol{J_A}(q^k)(q - q^k) + o||q - q^k||^2 \tag{2.3}$$

$$x^d \approx k(q^k) + \boldsymbol{J_A}(q^k)(q - q^k) \tag{2.4}$$

and applying $\boldsymbol{J_A}^\dagger = \boldsymbol{J_A^T}(\boldsymbol{J_A}\,\boldsymbol{J_A}^T)^{-1}$ on both sides, an iterative formula can be obtained :

$$q(k+1) = q(k) + \boldsymbol{J_A}^\dagger(x^d - k(q(k))) \tag{2.5}$$

also rewritten as :

$$\tilde{q} = \boldsymbol{J_A}^\dagger(x) \tag{2.6}$$

In comparison to other techniques, such as those based on the transposition of the Jacobian, this controller formula is shown in [24] to have a quadratic convergent rate, making it faster. On the other hand, this technique suffers the presence of singularities and works properly as long as x is close enough to x^d.

2.1.1 Lyapunov Stability

The following background information, extracted from [18], about various types of stability and Lyapunov technique is reported in order to assess the stability of the suggested controller, which will be presented in section 3.

Equilibrium points

Consider a dynamical system which satisfies:

$$\dot{x} = f(x,t) \quad x(t_0) = x_0 \quad x \in \mathbb{R}^n \tag{2.7}$$

Assume that $f(x,t)$ satisfies the standard conditions for the existence and uniqueness of solutions, i.e. $f(x,t)$ is Lipschitz continuous with respect to x, uniformly in t, and piecewise continuous in t. Consider $x^* = 0$ is an equilibrium point for the system.

Definition 1 (Lyapunov stability).
The equilibrium point $x^* = 0$ is stable (in the sense of Lyapunov) at $t = t_0$ if for any $\epsilon > 0$ there exists a $\delta(t_0, \epsilon) > 0$ such that

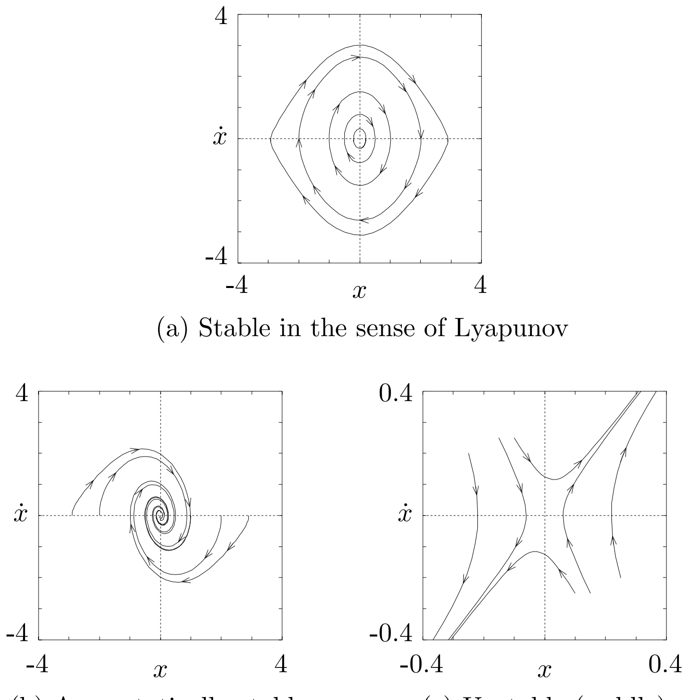
$$\|x(t_0)\| < \delta \quad \Longrightarrow \quad \|x(t)\| < \epsilon, \quad \forall t \geq t_0.$$

Lyapunov stability is a very mild requirement on equilibrium points. For instance, in a real case, it means that a robot arm is guaranteed to stay inside the target position's boundaries but not to reach the desired location. The differences between Lyapunov stability and the asymptotic one (described above) are illustrated visually in 2.1.1.

Definition 2 (Asymptotic stability).
An equilibrium point $x^* = 0$ is asymptotically stable at $t = t_0$, if $x^* = 0$ is stable and $x^* = 0$ is locally attractive, i.e. there exists $\delta(t_0)$ such that:

$$\|x(t_0)\| < \delta \quad \Longrightarrow \quad \lim_{t\to\infty} x(t) = 0$$

Since they describe how a system acts while it is getting close to equilibrium, the definitions 2 and 1 are considered local definitions. An equilibrium point x^* is referred to as globally stable if it remains stable under all initial conditions $x0$ in $\mathbb{R}^n$.



(a) Stable in the sense of Lyapunov

(b) Asymptotically stable

(c) Unstable (saddle)

Figure 2.1.1: Phase portraits for stable and unstable equilibrium points.

Lyapunov's direct method

Lyapunov's direct method permits access to a system's stability without explicitly integrating the differential equation 2.7. The approach is a generalization of the notion that if a system contains some "measure of energy", stability can be determined by looking at the rate at which the system's energy changes. Resorting to the well-known definitions of positive definite, locally positive definite, and decrescent functions the next theorem is stated.

Theorem 1 (Basic theorem of Lyapunov)**.**
Let $V(x,t)$ be a non-negative function with derivative $\dot{V}(x,t)$ along the trajectories of the system, namely :

$$\dot{V}\Big|_{\dot{x}=f(x,t)} = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f.$$

1. *If $V(x,t)$ is locally positive definite and $\dot{V}(x,t) \leq 0$ locally in x and for all t, then the origin of the system is locally stable (in the sense of Lyapunov).*
2. *If $V(x,t)$ is locally positive definite and decrescent, and $\dot{V}(x,t) \leq 0$ locally in x and for all t, then the origin of the system is uniformly locally stable (in the sense of Lyapunov).*
3. *If $V(x,t)$ is locally positive definite and decrescent, and $-\dot{V}(x,t)$ is locally positive definite, then the origin of the system is uniformly locally asymptotically stable.*
4. *If $V(x,t)$ is positive definite and decrescent, and $-\dot{V}(x,t)$ is positive definite, then the origin of the system is globally uniformly asymptotically stable.*

2.2 Model Predictive Control

MPC is a class of control algorithms built on the knowledge of the system model. The first approaches based on state space model theory, Linear Quadratic Regulator (LQR) and Linear Quadratic Gaussian Regulator (LQGR), showed that a linear time-invariant state feedback control is the optimal controller for a linear time-invariant MIMO system. A number of very desirable characteristics of this linear feedback control law, such as existence, uniqueness, and asymptotic stability have been demonstrated and a clear formula, reasonably

simple to calculate, is given for the feedback gain.
However, it only ended up being used in a very small number of actual implementations. The primary cause of that is due to practical considerations; in fact, LQR regulators may severely malfunction in the presence of non-linearities like input saturation (which are frequently present in real applications) and under conditions of constraints like state or control limitations required to achieve the desired goal. Talking instead about robustness, it is immediately noticeable that, in comparison to the other optimum strategies stated above, MPC has instantaneous feedback from the plant.LQR is a closed loop approach too, since the sequence of inputs is a sequence of gains that multiply the actual state error, but once computed offline, these gains are fixed and will be adopted without any possible modifications in real-time. This can lead to poor performance if the model is not very accurate or if disturbances are present. As will be discussed below, MPC, may totally solve these circumstances and get around these restrictions by calculating a new control gains sequence at each new iteration.

2.2.1 Receding Horizon principle

Starting from the knowledge of the model is possible to make predictions at each iteration about future plant outputs. In particular, the number of steps of looking ahead, in a discrete version of the algorithm, is indicated with N and represents the prediction horizon. Then the optimization problem is solved based on measured input data and the prediction of the output for different input scenarios. This procedure is repeated at each new time step while satisfying a set of constraints. The resultant optimal control sequence $[u(t)^\star, u(t+1)^\star \dots u(t+N)^\star]$ is applied to only the next time step, i.e. only the first input $u(t)^\star$ is adopted, while the MPC is solving the optimization problem again to drive the predicted plant outputs as close as possible to the desired references. This approach is the so-called Receding Horizon and image 2.2.1 displays how it works.

By this iterative strategy, MPC has the ability to anticipate and optimize the

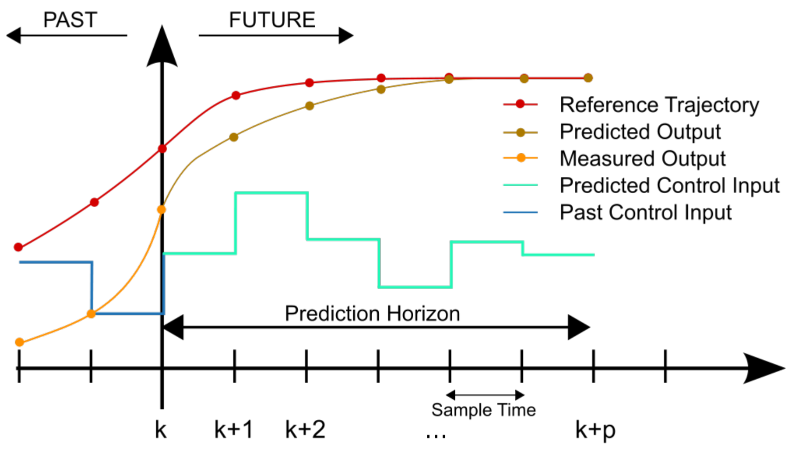


Figure 2.2.1: Reciding Horizon visual representation [10]

future process trajectory and can take control actions accordingly, before the process actually arrives at this future point.

The two different approaches involved in this work, the well-known PID controller and the MPC algorithm, are compared in figure 2.2.2. While PID is widely adopted for its simplicity and for the possibility of tuning empirically without a model of the plant, MPC can achieve high control performance in terms of efficiency and tracking. Furthermore, w.r.t. PID, its easier handling of constraints and competing objectives as well as the tuning of the controller that is more intuitive [17].

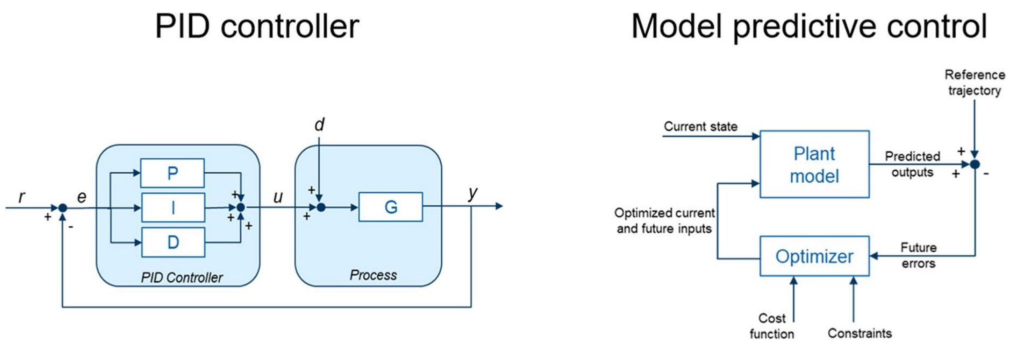


Figure 2.2.2: Comparison between PID controller and MPC

To make the technique work, it is important to explain the criteria required to deal with the two additional challenges that occur with this approach: stability and persistent feasibility. The concept of ”stability” refers to Lyapunov

stability, but ”persistent feasibility” means that the controller controller provides feasibility for all future steps (i.e. there is a feasible ”solution for all future steps”).
First of all, basic formulations with proof of stability is presented.
Given the generic discrete state space model:

$$x(t+1) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t) \tag{2.8}$$

where the two terms represent respectively the system's unforced and forced state evolution. The control input $u(t)$ is provided by minimizing the cost function J w.r.t all the sequence of the inputs from $u(t)$ to $u(t+N)$:

$$\begin{aligned} \min_u V(t) =& \min_u \sum_{\tau=t+1}^{t+N} J(\tau) \\ \text{s.t.} \quad & x(\tau+1) = \boldsymbol{A}x(\tau) + \boldsymbol{B}u(\tau) \\ \text{considering : } J(\tau) =& (x(\tau)-x^d)^T \boldsymbol{Q} (x(\tau)-x^d)^T + u(\tau-1)^T \boldsymbol{R} u(\tau-1) \end{aligned} \tag{2.9}$$

where x^d is the target state and $\boldsymbol{Q}, \boldsymbol{R}$ are the weight matrices.
The predictive system can be modeled as :

$$X = \mathbf{G}x(t) + \mathbf{H}U + \mathbf{F}u(t) \tag{2.10}$$

where

$$X = \begin{bmatrix} x(t+1) \\ \vdots \\ x(t+N) \end{bmatrix} \quad U = \begin{bmatrix} u(t+1) \\ \vdots \\ u(t+N) \end{bmatrix} \quad \boldsymbol{G} = \begin{bmatrix} A \\ \vdots \\ A^n \end{bmatrix} \tag{2.11}$$

$$\boldsymbol{H} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ B & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-2}B & A^{N-3}B & \dots & 0 \end{bmatrix} \quad \boldsymbol{F} = \begin{bmatrix} B \\ AB \\ \vdots \\ A^{N-1}B \end{bmatrix} \tag{2.12}$$

and so the 2.9 can be rewritten as :

$$\begin{aligned} \min_u V(t) = \min_u X^T \bar{\boldsymbol{Q}} X + U^T \bar{\boldsymbol{R}} U \\ \text{s.t.} \quad X = \mathbf{G}x(t) + \mathbf{H}U + \mathbf{F}u(t) \end{aligned} \tag{2.13}$$

where $\bar{\boldsymbol{Q}}$ and $\bar{\boldsymbol{R}}$ are the block diagonal matrices in which the blocks in the diagonal are respectively N times $\boldsymbol{Q}$ and N times $\boldsymbol{R}$.
With this formulation, the problem is a Quadratic Problem (QP) and so can be solved resorting to the Riccati equation.
However adding more constraints, as was done in the next sections, the problem is no more linear, so other solution approaches must be exploited. The following theorem states that also in the case of non-linearity, due to the constraints, is possible to find a solution that leads to the stability (see [7] for a more in-depth explanation).

Theorem 2 (Maciejowski)**.**
Given an MPC algorithm, namely the receding horizon method is applied with only the first input element used from the optimization problem, if the constraints are feasible, and assuming that $x(t+N) = 0$ and $u(t+N) = 0$, $J(x,u) \geq 0$ and $J(x,u) = 0$ iff $x = \underline{0}$ and $u = \underline{0}$, then $x = \underline{0}$ is a stable point and the optimization problem in 2.8 is feasible and solved in each step.

Proof. Let $V_0^*(t)$ be the optimal value of $V(t)$ corresponding to the optimal input signal u_0^*. Clearly, $V_0^*(t) \geq 0$ and $V_0^*(t) = 0$ only if $x(t) = 0$. In fact if $x(t) = 0$, then the optimal solution is to set $u(t+i) = 0$ for all i. To show that $V_0^*(t+1) \leq V_0^*(t)$, and hence that $V_0^*(t)$ is a Lyapunov function, let consider:

$$\begin{aligned}
V_0^*(t+1) =& \min_u \sum_{i=1}^{N} J(x(t+1+i), u(t+i)) \\
=& \min_u \left[\sum_{i=1}^{N} J(x(t+i), u(t-1+i)) - J(x(t+1), u(t)) \right] \\
&+ J(x(t+1+N), u(t+N)) \\
\leq& - J\left(x(t+1), u_0^*(t)\right) + V_0^*(t) + \\
&\min_u [J(x(t+1+N), u(t+N))]
\end{aligned}$$

since the optimum cannot be worse that keeping the optimal solution found at time t. But assuming as terminal constraint $x(k+N) = 0$, thus $u\left(t+N_p\right) = 0$ and remain at $x = 0$. This gives

$$\min_u \{J(\boldsymbol{x}(k+1+N), u(k+N))\} = 0$$

Since $J\left(x(k), u_0^*(k) \geq 0\right.$, then

$$V_0^*(k+1) \leq V_0^*(k)$$

and therefore, $V_0^*(k)$ is a Lyapunov function. □

Proved the stability, persistent feasibility should be to discussed. Considering X_k as the set of all feasible states at prediction step k, some useful definition are provided in the following :

Definition 3 (MPC persistent feasibility)**.**
Starting from any initial state $x(0) \in X_0$ persistent feasibility is achieved if, under MPC control law, feasibility is guaranteed at all time (i.e. $x(t) \in X_0 \quad \forall k \in \mathbb{Z}_+$).

Definition 4 (Control Invariant set)**.**

A set $C \subseteq X$ is called a Control Invariant set for a dynamic system if

$$x(t) \in C \Rightarrow \exists u(t) \in U \ s.t. \ f(x(t), u(t)) \in C, \forall t \in \mathbb{Z}_+$$

The next theorem is rapidly defined with the help of these premises.

Theorem 3.
If X_f is control invariant, then the receding horizon optimization algorithm is persistently feasible.

Proof. considering $X_N = X_f$ the set of all feasible states at prediction step N, since X_N is control invariant, namely $\exists u \in U \ s.t. \ f(x,u) \in X_N$, $X_N \subseteq X_{N-1}$. Proceeding at the same way $X_N \subseteq X_{N-1} \subseteq X_{N-2} \cdots \subseteq X_0$. So for any $x \in X_0$, applying the first input u_0 from MPC control law, $x(1) = f(x, u_0) \in X_1 \subseteq X_0$ and so on obtaining $x(t) \in X_0 \forall t \in \mathbb{Z}_+$ $\square$

2.2.2 MPC discretization

In order to implement a feasible algorithm, a discretization of continuous implementation of a real plant model as:

$$\dot{x} = \boldsymbol{A}x + \boldsymbol{B}u$$

must be considered. Given a time step Δt, a first simple discretization consists of applying the Euler method:

$$x(k+1) = x(k) + (\boldsymbol{A}x(k) + \boldsymbol{B}u(k)\ )\Delta t \tag{2.14}$$

Runge–Kutta discretization

Instead of Euler method, a more sophisticated discretization, the Runge-Kutta, also known as ”RK4”, is proposed [2]. Given a generic function :

$$\frac{dy}{dt} = f(t, y), \quad y(0) = y_0 \tag{2.15}$$

is possible to rewrite the next state $y(k+1)$ as the previous one plus a summation of 4 weighted terms. In this method, to have consistency, the sum of the weights must be unitary.

$$
\begin{aligned}
& y(k+1) = y(k) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\Delta t \\
\text{with :}\quad & k_1 = f(t, y(k)) \\
& k_2 = f(t + \tfrac{\Delta t}{2}, y(k) + k_1\tfrac{\Delta t}{2}) \\
& k_3 = f(t + \tfrac{\Delta t}{2}, y(k) + k_2\tfrac{\Delta t}{2}) \\
& k_4 = f(t + \Delta t, y(k) + k_3\Delta t)
\end{aligned}
\tag{2.16}
$$

An example is shown in figure 2.2.3. Applying the previous to the 2.8 and exploiting the linearity of the function $f()$:

$$
\begin{aligned}
& x(k+1) = x(k) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\Delta t \\
\text{with :}\quad & k_1 = \boldsymbol{A}x(k) + \boldsymbol{B}u(k) \\
& k_2 = \boldsymbol{A}(x(k) + k_1\tfrac{\Delta t}{2}) + \boldsymbol{B}u(k) \\
& \quad = \boldsymbol{A}x(k) + \tfrac{\Delta t}{2}\boldsymbol{A}^2x(k) + \tfrac{\Delta t}{2}\boldsymbol{AB}u(k) + \boldsymbol{B}u(k) \\
& \quad = (I_n + \tfrac{\Delta t}{2}\boldsymbol{A})\boldsymbol{A}x(k) + (I_n + \tfrac{\Delta t}{2}\boldsymbol{A})\boldsymbol{B}u(k) \\
& \quad = (I_n + \tfrac{\Delta t}{2}\boldsymbol{A})k_1 \quad \text{so with the same procedure:} \\
& k_3 = (I_n + \tfrac{\Delta t}{2}\boldsymbol{A})k_2 \\
& k_4 = (I_n + \Delta t\boldsymbol{A})k_3
\end{aligned}
\tag{2.17}
$$

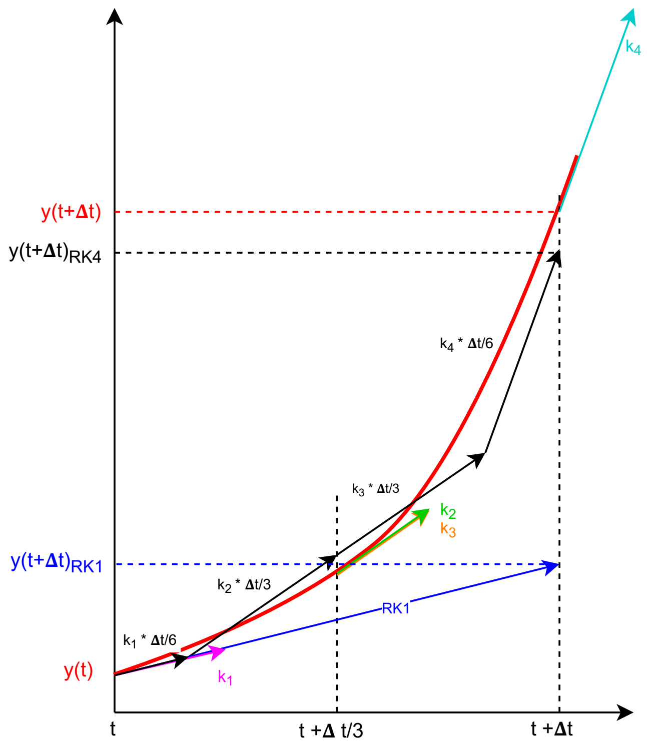


Figure 2.2.3: Runge–Kutta method compared to Euler one.

A recursive and thus computationally efficient formula for determining all the terms of the summation can be found. Because it is a fourth-order method, the approximation is more precise, and the truncation error is on the order of $O(\Delta t^5)$. It is important to recognize that this formulation can be generalized to any number of terms within the summation, and thus the "RK1" version, which considers only the term k_1 with a unitary coefficient for that term, is equivalent to using Euler's approach. Figure 2.2.3 demonstrates why the discretization computed using the 4^{th} order technique is more precise than the Euler method. Below an algorithmic implementation of the previous discrete MPC-based controller is proposed (algorithm 1).

Algorithm 1 MPC algorithm

```
Require: $t = 0, T, \delta x^d, f(x,u)$    ▷ time,time step,prediction horizon, target position, dynamics
  $U \leftarrow$ vect$(u(t), u(t+1), \dots, u(t+T))$
  $X \leftarrow$ vect$(x(t), x(t+1), \dots, x(t+T)$
  while $||\hat{x} - x^d|| \leq$ treshold do
    $\hat{x} \leftarrow x(t)$    ▷ measure actual state
    find $U$ to minimize $J(X, U)$
      subject to :    $X(0) = \hat{x}$ ,    $x(t+1) = f(x(t), u(t))$
    $u^\star \leftarrow U(0)$ apply $x^\star$ to the system for a time $\delta$
    $t \leftarrow t + \delta$
  end while
```

It is critical to remember that a termination condition is introduced to the algorithm in order to stop it. In practice, this is essential to limiting oscillations around the final state, especially when a lot of effort is expended to reduce state error.

Chapter 3

Methods and Tools

This section describes the environment used to design the controller as well as the relevant models and hardware adopted to implement these controllers.
ROS is involved in the development of the entire controller structure. This is primarily due to the ease with which different hardware can be used and made to work together at the same time. With ROS, it is possible to connect the Motion Capture System (MCS), which provides the position of all the objects, to the PCs inside the mobile robots (called *Robot Control Computers* and equipped with Ubuntu) and the lab PC (*Development Computer*), which runs all the scripts concerning the task menage control.
Following a brief explanation of how ROS works, the model of the mobile base is presented, along with the related formulas that associate the speeds of the wheels with the velocity of the body.
Similarly, after some details about the motor used in the project, the Carthesian controller developed for the arm is explained and its stability is demonstrated. The MCS behavior is described in the final section.

3.1 Environment : ROS

ROS is a set of open-source software libraries designed to provide services such as low-level device control, message-passing between processes, and hardware

abstraction.
Each process is represented in the graph structure as a *node*. The edges that connect the nodes instead are called *topics*. In order to start a communication, a node can publish some data in a topic, and each other node can subscribe to that topic to read the information. This decentralized architecture lends itself well to robots, which often consist of a subset of networked computer hardware. Nevertheless, to set up peer-to-peer communication, a *Master* node must always be present. This is useful for setting up node-to-node communication for topics and controlling parameter server updates. The principle of working in this environment is depicted graphically in Figure 3.1.1.

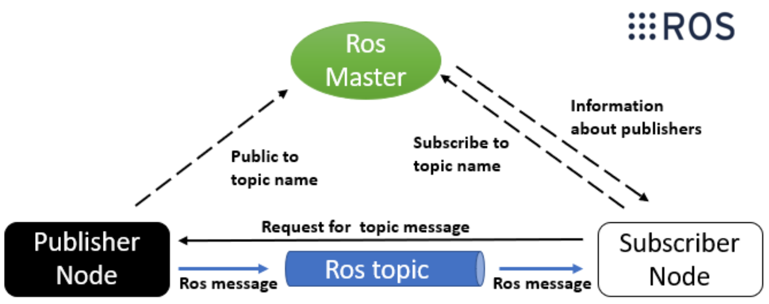


Figure 3.1.1: Ros structure and working behavior

Each node can be written in either the C++ or Python programming languages. In this work, only Python will be adopted.
The scripts the contain the instructions to run the nodes are usually incorporated in *packages*, where a file called *CMakeList.txt* contains the set of directives and instructions describing source files and targets of the package (executable, library and messages). For further information about ROS see introduction in [23].
Going into full depth of this work, the ROS distribution employed is *ROS 1* with versions *Melodic* for the Development Computer and *Noetic* for the Robot Control Computer. Aside from all the code developed individually in a customized package , this project makes extensive use of the package already produced by Hebi robotics to handle and control the motors, relying on the

equipped sensors and the already implemented inner controller.
When it comes to Python libraries, in addition to the commonly used *NumPy, SciPy* and *Pandas*, *RosPy*, which is required to code with ROS, and *Casadi*[1], a symbolic toolbox essential for managing derivatives and optimisation problems, must be included.

3.2 Model : Omni Base

The base is a holonomic drive system since it is equipped with omni wheels (figure 3.2.1). The movement across all the directions is possible by the presence of small discs (called rollers) around the circumference which are perpendicular to the turning direction. In figure 3.2.2 is shown the structure of the base with the velocities of each wheel and, accordingly, the velocity of the base.
From the following equations for the forward kinematic:

$$\begin{aligned} V_x^m &= \frac{2V_2 - V_1 - V_3}{3} \\ V_y^m &= \frac{\sqrt{3}V_3 - \sqrt{3}V_1}{3} \\ \omega_p &= \frac{V_1 + V_2 + V_3}{3L} \end{aligned} \tag{3.1}$$

is possible to derive, for the inverse dynamics :

$$\begin{aligned} V_1 &= -\frac{V_x^m}{2} - \frac{\sqrt{3}V_y^m}{2} + L\,\omega_p \\ V_2 &= V_x^m + L\omega_p \\ V_3 &= -\frac{V_x^m}{2} + \frac{\sqrt{3}V_y^m}{2} + L\,\omega_p \end{aligned} \tag{3.2}$$

[1]for more details see Casadi's web page : `https://web.casadi.org/`

Furthermore, it is useful to calculate the body velocity in relation to the world:

$$V^B = \begin{bmatrix} V_x^m \\ V_y^m \\ \omega_p \end{bmatrix} \qquad V^W = \mathcal{R}_z(\theta)V^B \tag{3.3}$$

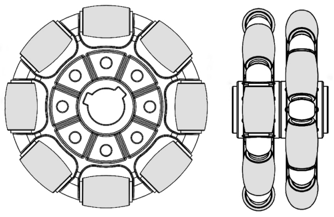


Figure 3.2.1: Graphical representation of an omni wheel

Thus, inside the base is already implemented a controller that takes as an input the velocity V^B and provides the velocities for each wheel. The wheel velocities are directly sent to the inner controller inside each motor, the same type of motor used for the arm in figure 3.3.3.

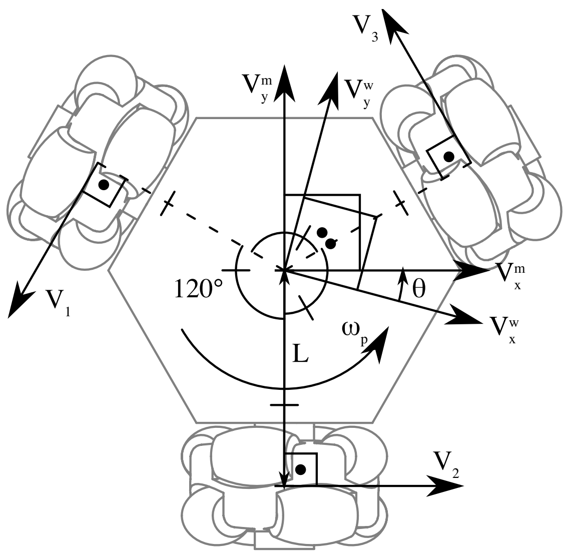


Figure 3.2.2: Geometry structure of the mobile base

There are some reasons why these types of wheels are not widely used in industry, with the simpler "turtle-bot" structure being preferred. This is primarily due to cost, as omni wheels are more expensive and less durable than standard wheels. Another reason is inefficiencies caused by energy loss in a direction normal to the movements through the peripheral rollers [5]. Even through the above, to provide a simpler experimental environment, these are widely used in laboratories.

3.3 Model : Arm

The robotic arm under consideration for this work is an articulated 6 DOF robot with only revolute joints. The structure is essentially the same as *A-2085-06-6DoF X-Series-Arm*, proposed on the Hebi robotics web site (see [8] for complete assembly instructions), with a few exceptions, such as the absence of a gas spring and a different type of EE.
According to the Denavit-Hartenberg convention, the disposition of the motors and relative reference frames is shown in figure 3.3.1.
The reference frame attached to m_5 deviates from the convention and is chosen to be aligned with the base frame when the arm is in the *home* configuration.
Another aspect to take into account is that 6 DOF is the bare minimum for the complete manipulation of an object, as it allows the controller to move the EE in the 3D environment with any possible orientation in the three axis. From Hebi, some example code are provided in `https://github.com/HebiRobotics/hebi_cpp_api_ros_examples`, where the ROS package *hebi_cpp_api_ros_examples* is adopted in this project to provide the low level controller of the base and the arm. The details of the motors and the network will be explained in the following section.

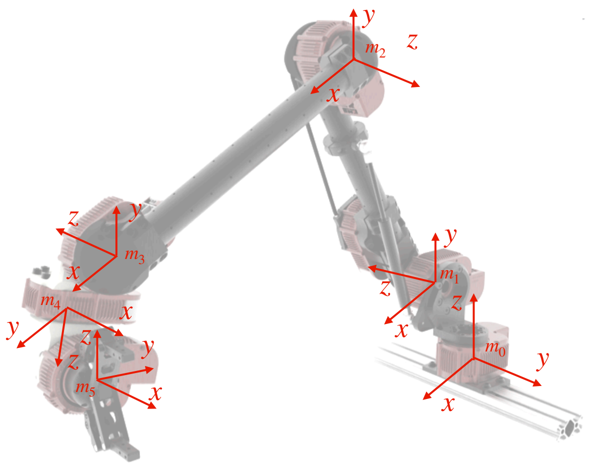


Figure 3.3.1: Arm model with the reference frames of each joint reported.

3.3.1 Hebi motors

HEBI modules are Ethernet-capable devices that can be used to construct a robot or an automated system. These modules include physical motion actuators, namely the motors employed for this project, as well as devices for interacting with other sensors and I/O. Modules communicate using firmware over a standard Ethernet connection, linking to host PCs and receiving an IP address. In the network configuration adopted in this project (see figure 3.3.2) the motors are connected through a wired network, each with its own router set for DCHP and an on-board computer (Robot Control Computer). The Rosie mobile base is equipped with a PC running Ubuntu, so the three wheels are already rigidly connected. Instead, the arm's motors are linked together in a chain via Ethernet cables, with the first (m_0) wired to the base. Off-board development is possible by connecting to either the robot network or the office network via a separate Pc (Development Computer).

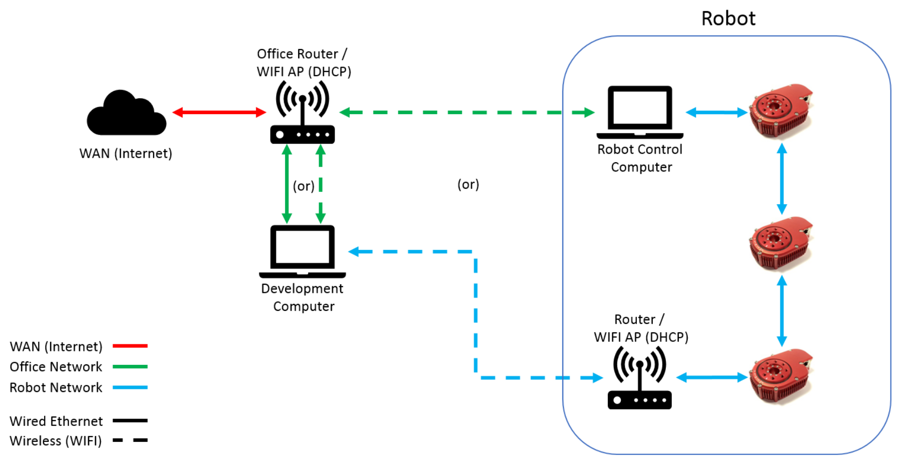


Figure 3.3.2: Network configuration adopted for this project

The actuators chosen for this project are the X-Series (in figure 3.3.3) that They combine a brushless motor, geartrain, spring, encoders, and control electronics in a small package that operates on standard DC voltages and communicates via standard 10/100Mbps Ethernet. These actuators are intended to be full-featured robotic components that can control position, velocity, and torque at the same time. Each actuator also includes a variety of other sensors to measure the following quantities:

- Angular Position (multi-turn absolute, +/- 4 turns)
- Angular Velocity
- Output Torque
- 3-axis Acceleration (Accelerometer / Gyro)
- Temperature
- Voltage
- Current

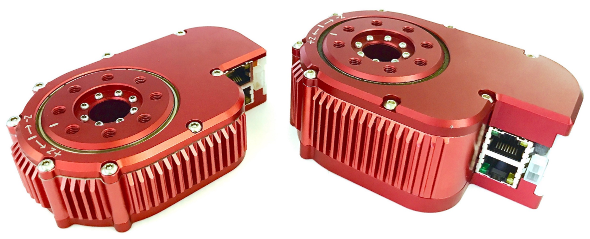


Figure 3.3.3: X-Series motors provided by Hebi Robotics

3.3.2 Control strategy

As stated in section 2, a Cartesian space controller is used in conjunction with the inner controller within the motors. According to the user manual provided by Hebi Robotics (see complete manual in [9]), strategy 3 is chosen, where Position, Velocity, and Effort PID controllers all directly sum to generate motor Pulse Width Modulation (PWM) commands. Figure 3.3.4 summarize the all the other possible strategies available to control the motors.

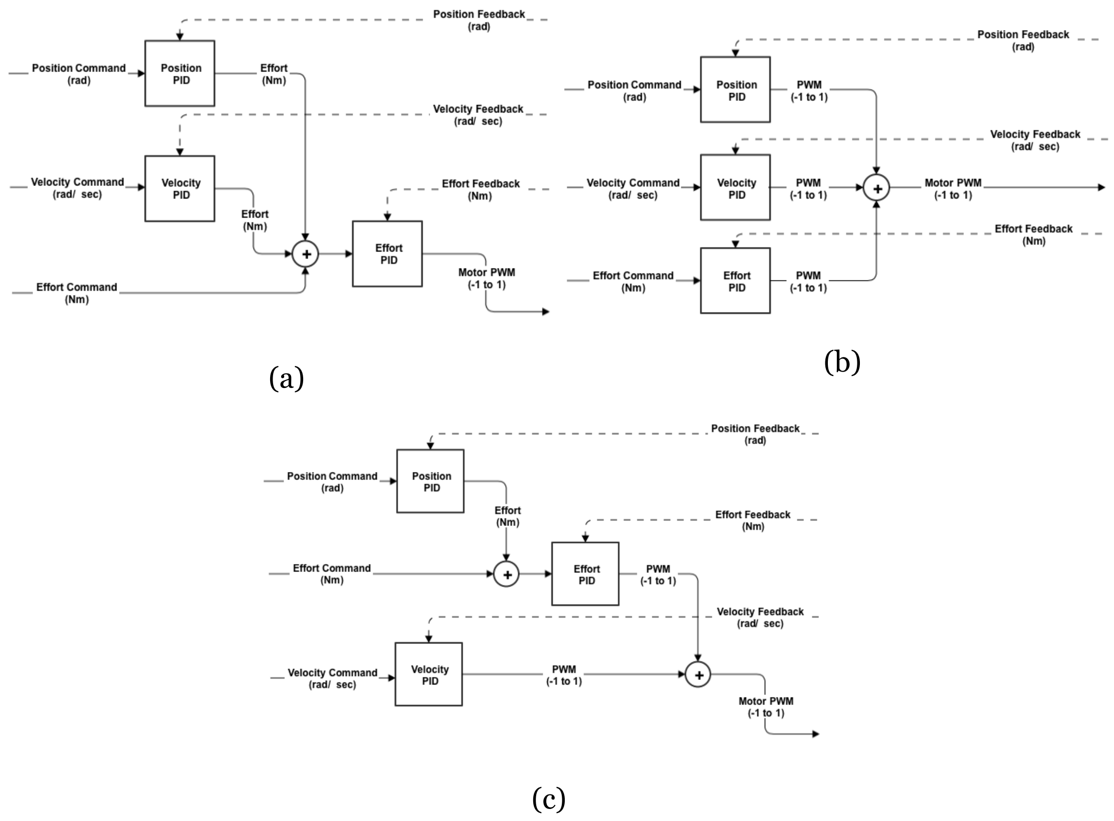


Figure 3.3.4: Different strategies proposed for the inner controller by Hebi Robotics. In order strategy 2, 3 and 4 are reported.

A brief digression about PWM, for sake of completeness, is required before discussing the controller. PWM, that stands for Pulse-Width Modulation, is a method of providing an arbitrary effective voltage to the load (in this case, the motor) by switching on and off the DC supply. The output voltage is proportional to the percentile of time when the switch is turned on (called duty cycle). The

PWM switching frequency must be high enough so that the load is not affected. The frequency for X-series motors are set to $20kHz$ which is appropriate for DC motor control.
Following the calculation of the final PWM values a series of safety controllers are run in all of the control strategies. These safety controllers may change the PWM value based on factors such as bus voltage, motor winding temperature, motor velocity, and position limits.
Moving to the proposed controller, the structure is presented in figure 3.3.5.

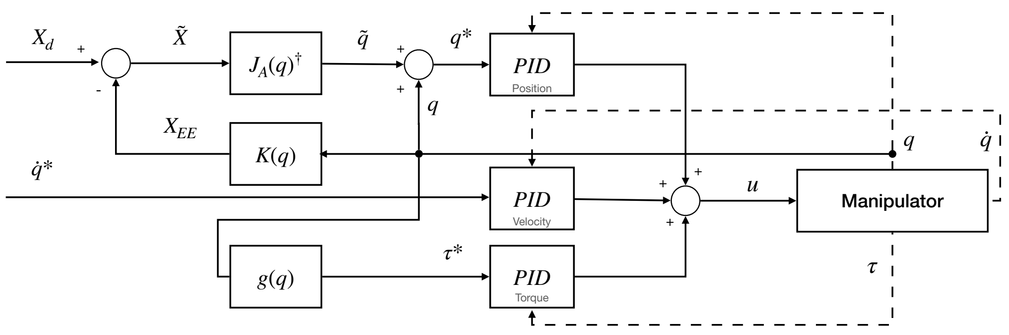


Figure 3.3.5: Cartesian position controller adopted for the arms.

Analysis of Stability

As a first step, only the proportional terms within the PIDs are considered for the following computations. Another assumption is that the input u and the real torque provided by the motors τ are approximately equal. If the torque constant K_τ is well estimated, this is reasonable (that is quite simple with both torque and current sensors available).

So the input u can be expressed as :

$$\begin{aligned}
&u = g(q) + \boldsymbol{K}_p^{\tau}\,(\,g(q) - \tau) + u_q \\
&\text{considering :} \quad \tau = u \\
&u\,(\boldsymbol{I_n} + \boldsymbol{K}_p^{\tau}) = g(q)\,(\boldsymbol{I_n} + \boldsymbol{K}_p^{\tau}) + u_q \\
&\boldsymbol{K}_{u_q} = (\boldsymbol{I_n} + \boldsymbol{K}_p^{\tau})^{-1} \\
&u = g(q) + \boldsymbol{K}_{u_q} u_q
\end{aligned} \tag{3.4}$$

where $(\boldsymbol{I_n} + \boldsymbol{K}_p^{\tau})$ is clearly invertible, considering a diagonal gains matrix, and u_q is the contribution of the controller given by the position and velocity proportional controller for the joint states q :

$$u_q = \boldsymbol{K_P^{pos}} \boldsymbol{J_A}^{\dagger}(q)(x^d - x) + \boldsymbol{K_P^{vel}}(\dot{q}_d - \dot{q}) \tag{3.5}$$

and considering $\dot{q} = 0$, u_q can be rewritten as :

$$u_q = \boldsymbol{K_P^{pos}} \boldsymbol{J_A}^{\dagger}(q)\tilde{x} - \boldsymbol{K_P^{vel}}\dot{q} \tag{3.6}$$

Resorting to the Lyapunov theory explained in 2.1.1, a suitable function to prove the stability of the controller is :

$$\begin{aligned}
&V(\dot{q}, \tilde{q}\,) = \frac{1}{2}\dot{q}^T \boldsymbol{B}(q)\dot{q} + \frac{1}{2}\tilde{q}^T \boldsymbol{K}\tilde{q} \\
&V(\dot{q}, \tilde{q}\,) = 0\;, \qquad V(\dot{q}, \tilde{q}\,) > 0 \quad \forall \dot{q}, \tilde{q} \neq \underline{0}
\end{aligned} \tag{3.7}$$

where $\tilde{q}$ is the Cartesian error, $\boldsymbol{B}$ the inertia matrix considered in 2.2 and $\boldsymbol{K}$ a positive-definite matrix, namely $\forall x \neq \underline{0} \quad x^T \boldsymbol{K} x > 0$.
Deriving w.r.t time the Lyapunov function :

$$\begin{aligned}
&\dot{V}(\dot{q}, \tilde{q}\,) = \dot{q}^T \boldsymbol{B}\ddot{q} + \frac{1}{2}\dot{q}^T \dot{\boldsymbol{B}}\dot{q} + \dot{\tilde{q}}^T \boldsymbol{K}\tilde{q} \\
&\qquad \text{using :} \quad \boldsymbol{B}\ddot{q} = u - \boldsymbol{C}(q, \dot{q})\dot{q} - \boldsymbol{F}\dot{q} - g(q) \\
&\qquad \text{and} \quad \dot{q}^T \left(\frac{1}{2}\dot{\boldsymbol{B}} - \boldsymbol{C}(q, \dot{q})\right)\dot{q} = 0 \\
&\dot{V}(\dot{q}, \tilde{q}\,) = -\dot{q}^T \boldsymbol{F}\dot{q} + \dot{q}^T\,(u - g(q)\,) + \dot{\tilde{q}}^T \boldsymbol{K}\tilde{q}
\end{aligned} \tag{3.8}$$

and resorting to 2.6 and considering that $\dot{\tilde{q}} = -\dot{q}$, is possible to rewrite :

$$\dot{V}(\dot{q}, \tilde{q}\ ) = -\dot{q}^T \boldsymbol{F}\dot{q} + \dot{q}^T \left(u - g(q) - \boldsymbol{K}\boldsymbol{J_A}^{\dagger}(q)\tilde{x}\right) \tag{3.9}$$

Now, using the controller input from 3.4 and 3.6:

$$\dot{V}(\dot{q}, \tilde{q}\ ) = -\dot{q}^T \boldsymbol{F}\dot{q} + \ \dot{q}^T \left(g(q) + \boldsymbol{K}_{u_q}\boldsymbol{K_P^{pos}}\tilde{q} + \boldsymbol{K}_{u_q}\boldsymbol{K_P^{vel}}\dot{q} - g(q) - \boldsymbol{K}\boldsymbol{J_A}^{\dagger}(q)\tilde{x}\right) \tag{3.10}$$

considering $K = \boldsymbol{K}_{u_q}\boldsymbol{K_P^{pos}}$, only the quadratic term relating to velocity is left :

$$\dot{V}(\dot{q}, \tilde{q}\ ) = -\dot{q}^T \left(\boldsymbol{F} + \boldsymbol{K}_{u_q}\boldsymbol{K_P^{vel}}\right)\dot{q} \tag{3.11}$$

that is always negative for any $\dot{q} > 0$. So the system with the proposed control input will converge to $\dot{q} = \underline{0}$, as fast as $\left\|\boldsymbol{K}_{u_q}\boldsymbol{K_P^{vel}}\right\|$ increases. To evaluate the error $\tilde{q}$ instead is sufficient to evaluate the 2.2 with 0 velocity and acceleration, so:

$$\begin{aligned} &g(q) = g(q) + \boldsymbol{K}_{u_q}\boldsymbol{K_P^{pos}}\boldsymbol{J_A}(q)^{\dagger}\tilde{x} \\ &\boldsymbol{J_A}(q)^{\dagger}\tilde{x} = \underline{0} \quad \Leftrightarrow \quad \tilde{x} \in \ker \boldsymbol{J_A}(q)^{\dagger}(q) \end{aligned} \tag{3.12}$$

If $\boldsymbol{J_A}(q)$ is a non-singular square matrix, $\tilde{x}$ must converge to $\underline{0}$.
Regarding the position's integral action, proof of semiglobal practical stability can be found in proposition 2 of [3]. The trial and error method is used to obtain the coefficients, and the values chosen will be presented in the following chapter.

3.4 Model : Gripper

The EE adopted for the arm, as mention above, is not provided by Hebi but 3D-printed and built directly in the laboratory. It is an adaptative gripper which employs a Dynamixel[2] AX-12A servomotor as actuator. A photo of the

[2]web site :`http://www.dynamixel.com/list.php?dxl=x`

gripper is in 3.4.1. The communication with the servomotor is handled with an integrated USB2AX and the power is provided by a DC-DC buck converter. All the mount instruction can be found in the official KTH-SML github repository : `https://github.com/KTH-SML/kino_gripper`.

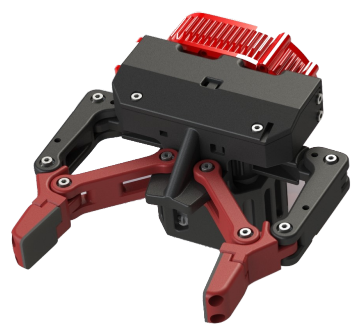


Figure 3.4.1: Portrait of the Kino gripper

The schematic and some computation are presented here. The following kinematics equations provide the gripper opening distance as a function of the servo motor angle ψ. The inverse kinematics is not computed analytically but all the possible values for ψ are pre-calculated and stored in a Look-Up table.
Starting from the four points :

$$\begin{gathered} D = \Big(u + (f-e)\cos(\gamma),\ v + (f-e)\sin(\gamma)\Big) \quad B = \Big(u + p + b\cos(\psi),\ v + q + b\sin(\psi)\Big) \\ A = \Big(u + a\cos(\theta),\ D_y + h\cos(-\gamma)\Big) \quad C = \Big(u - r,\ v + s\Big) \end{gathered} \tag{3.13}$$

considering u, p, v, q, b, a, c, g as constant and known values, some of these can be rewritten as functions of the coordinates of the previously mentioned point:

$$
\begin{aligned}
&c^2 = (A_x - B_x)^2 + (A_y - B_y)^2 \\
&f^2 = (u - B_x)^2 + (v - B_y)^2 \\
&\quad \text{and resorting to law of cosines: } f^2 = \left(g^2 + b^2 - 2gb\cos(\pi - \sigma - \psi)\right) \\
&h = \sqrt{c^2 - e^2} \\
&\gamma = \text{acos}\left(g^2 + f^2 - b^2/2gf\right) - \sigma \\
&\sigma = \text{acos}(p/g) \\
&e = \left(a^2 + f^2 + c^2\right)/2gf
\end{aligned}
\tag{3.14}
$$

is possible to find A_x as function of ψ without knowing θ:

$$
\begin{aligned}
&c^2 - f^2 = A_x^2 - 2A_xB_x + A_y^2 - 2A_yB_y - u^2 + 2uB_x - v^2 + 2vB_y \\
&\quad \text{and defining : } z = \left(A_y^2 - 2A_yB_y - u^2 + 2uB_x - v^2 + 2vB_y - c^2 + f^2\right) \\
&\quad \text{considering : } A_y = D_y + h\cos(-\gamma) \\
&A_x = B_x - \sqrt{(B_x^2 - z)}
\end{aligned}
\tag{3.15}
$$

And so the opening distance is $2C_x = A_x - r$

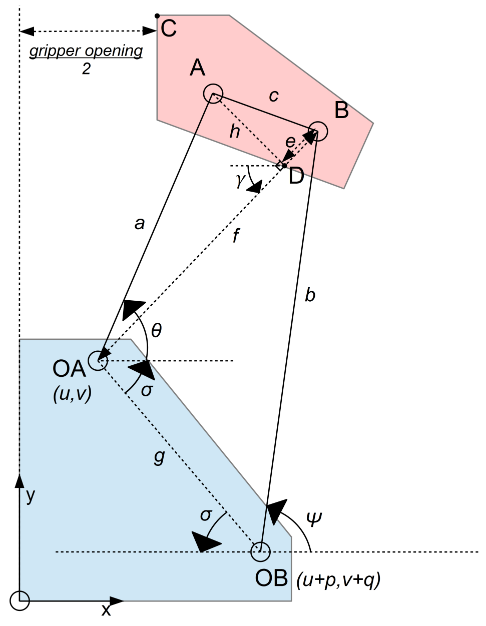


Figure 3.4.2: Schematic representation of the gripper.

Moreover in the KTH-SML github repository is already provided the ROS code to implements the Kinematics model of the gripper and the related Look-Up table for the inverse problem. So for each gripper a node will run to commands the servo motor, receiving as input the gripper opening distance.

3.5 Mocap Capture Qualisys

The estimation of each object's true location and orientation throughout the experimental setup is required for the job, hence a MCS is used. The one adopted is produced by Qualisys (figure 3.5.1) and it makes use of up to 12 cameras to estimate each marker's location in the arena.
Because the markers are all equal and lack communication technology (passive markers), the software must associate each body to a certain pattern and shape of the markers attached to it in order to distinguish a specific entity. Clearly, at least three markers are required to define an object, generating a reference frame centered in the middle of the surface bounded by the three points with

the x, y-directions lying on this surface and the z-axis perpendicular. However, establishing the z-axis direction is not unique. To avoid this uncertainty, an additional marker is introduced that must be not coplanar with the others. In a real-world setting, the number of markers is generally higher for two reasons. The first is about the system's robustness; if one marker is lost from the cameras or detaches from the body for any reason, the experiment has to continue without interruptions. The second cause, on the other hand, is explained by the presence of several similar objects. The greater the possibility that the pattern of the markers is similar to other items, the more difficult it is to distinguish between them. Thus increasing the number of markers the possibility of having similar geometries decrease and so the system becomes more reliable.

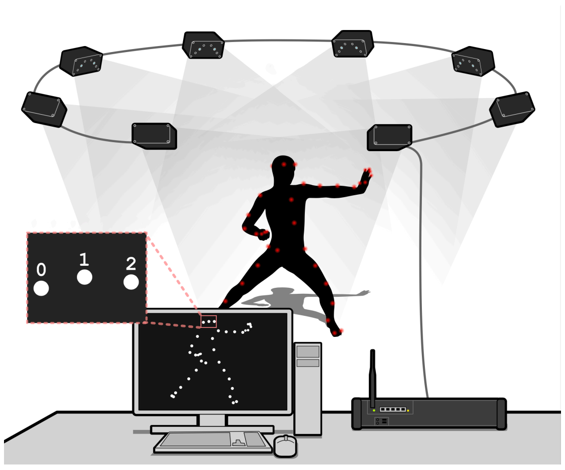


Figure 3.5.1: MCS provided by Mocap Qualisys, URL: `https://www.qualisys.com/`

Chapter 4

The proposed controllers

Most of the literature regards mobile manipulators as a unified object in which the DOF of the base and the arm are summed and so only one controller is involved to satisfy the tasks [16].
Instead, the base and the arm are viewed as two distinct entities in this work. As a result, both the structures can be flexibly interchanged with different hardware without needing to modify the entire controller, but only the part concerning the structure replaced. For example, if, instead of the presented ”omnibase,” a ”Turtle-bot” base or a ”4-wheels” base with its own control scheme is substituted, the command structure should remain unchanged.
So essentially 2 macro controllers work in parallel to manage the entire formation. The first one aims to move all the robot bases with the goal of keeping the object in the formation’s geometrical center. The second, on the other hand, is designed to move the grasped object inside the formation in the same direction that the formation must move to reach the target position via commands to the arms. It is significant to observe that the two controllers are not totally independent because they both rely on the object position, which depends on the commands from both controllers.
Moreover, as explained in section 3, all the position measurements are provided by the MCS and so each robot has access to the position of the all others and to the position of the object to carry.

The presence of MPCs, whose working principle was thoroughly explained in chapter 2, recurs frequently in the controller's structure. As previously stated, this technique provides numerous benefits such as efficiency, reliability, and the ability to asset some cost constraints. Because the models used in the majority of cases are very simple, as will be illustrated shortly, a PID could also achieve good performance in such cases. So, the ability to add constraints drives the decision to use an MPC. This will be used for obstacle avoidance, input limitations, and state limitations, such as the carried object's maximum velocity and angular velocity.

4.1 Formation Bases Controller

The formation base controller is responsible for the motion of all the bases. In figure 4.1.1 is possible to visualize the structure . Velocity commands are sent to all the bases, which have the inner controller (see 4.2.3) to commute the velocity references v^d_{base-i} in velocity references for each wheel V_1, V_2, V_3 , resorting to the 3.2.

In order to evaluate the object displacement, the *Formation Position Estimator* block computes the center of the formation starting from the position of all the bases $p_{ba\hat{s}e-i}$, *with* $i = 0, 1, 2$.

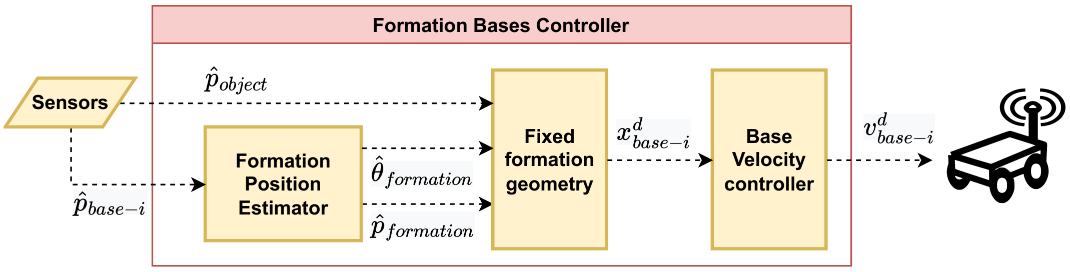


Figure 4.1.1: Graphical representation of the formation base controller

A graphical representation of the operating principle is provided in figure 4.1.2 : when the object is moved away from the formation's center, a position displacement $\hat{\delta}$ occurs and so a desired pose x^d_{base-i} is computed for each base.

In the following, the three main parts of the controller will be described in detail.

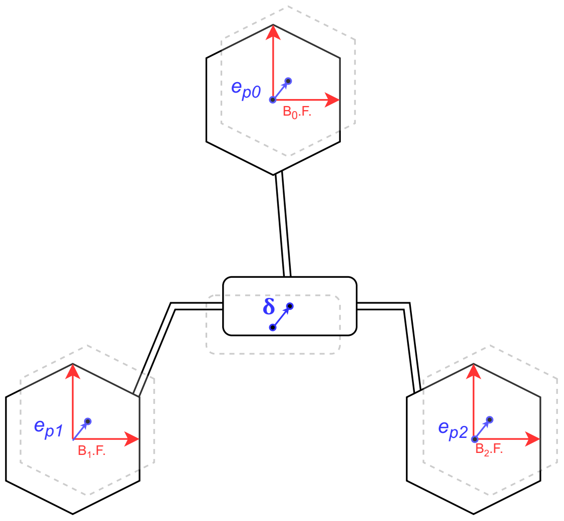


Figure 4.1.2: Operating principle of the Formation Bases Controller

4.1.1 Formation position estimator

The chosen configuration of the robots for the formation is an equilateral triangle, in this way the position of the center is nothing but the average of the 3 robot positions (see figure 4.1.3).

$$\hat{p_{formation}} = \frac{1}{3}\sum_{i=0}^{2} p_i \tag{4.1}$$

To compute the orientation, the average method is applied too. Renaming for simplicity $p_i = \hat{p_{base-i}}$, $\theta_i = \hat{\theta_{base-i}}$ and taking into account fixed angles inside the formation :

$$\begin{aligned}\theta_i &= \arctan\left(\frac{-p_i^y + \hat{p_{formation}}^y}{-p_i^x + \hat{p_{formation}}^x}\right) - i\frac{2\pi}{3}\\ \hat{\theta_{formation}} &= \frac{1}{3}\sum_{i=0}^{2}\theta_i\end{aligned} \tag{4.2}$$

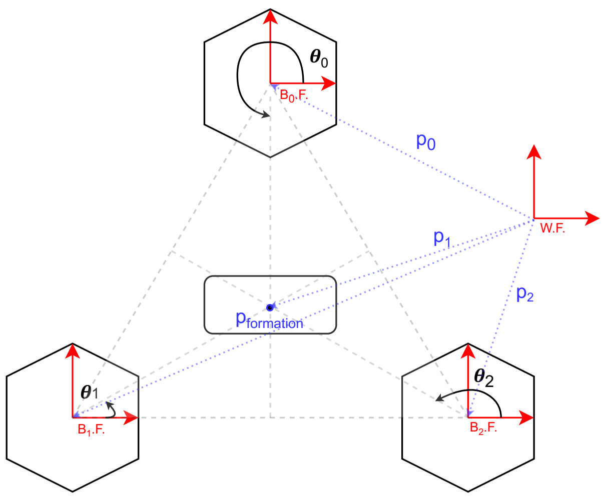


Figure 4.1.3: The chosen formation structure, where the robots take up the position of a equilateral triangle's vertices

As a result of using 4.1 in conjunction with other computations, the following formula is deduced:

$$\hat{\theta}_{formation} = \arctan\left(\frac{p_0^y + p_1^y}{p_0^x + p_1^x}\right) + \arctan\left(\frac{p_1^y + p_2^y}{p_1^x + p_2^x}\right) + \arctan\left(\frac{p_2^y + p_0^y}{p_2^x + p_0^x}\right) - \frac{2\pi}{3} \quad (4.3)$$

4.1.2 Fixed formation geometry

As previously stated, the chosen formation is an equilateral triangle, and the robots always remain in the triangle's vertices. The side of the triangle is denoted with L, the distance from the robots and the center is instead d, computed as $d = L/\sqrt{3}$. From this, the following formula is derived:

$$\hat{p}_{base-i}{}^{d} = \hat{p}_{object} + \mathcal{R}(\theta_{object})\, a(i) \quad \text{with } i = 0, 1, 2$$
$$\text{where :} \quad a(i) = \left[d\cos\left(\frac{\pi}{2} + i\frac{2\pi}{3}\right),\ d\sin\left(\frac{\pi}{2} + i\frac{2\pi}{3}\right) \right] \quad (4.4)$$

where $\mathcal{R}(\theta) \in S0(2)$ is the rotational matrix w.r.t. the object angle $\hat{\theta}_{object}$.
In this way, the orientation of the entire formation is aligned with that of the

object. It is thus possible to rotate the formation by rotating the object slightly. As a consequence of this fixed displacement between the formation angle and the object, the entire formation continues to rotate until the object is not realigned by the operator.

Base Orientation

The orientation of each base is computed independently from the formation:

$$\theta^{d}_{base-i} = \angle(p_{\hat{object}} - p_{\hat{base-i}}) \tag{4.5}$$

In this way, each robot is always pointed at the object all the time.
Nevertheless, angle tracking is not always straightforward because commands to achieve the desired orientation may involve an entire rotation in the opposite direction. For instance, because the angle provided by the MCS is always between $-\pi$ and π, when the robots are close to the limit boundary, as π, small oscillations of the object could cause this problem. To overcome the issue, a function called *get-minum-angle (gma)* was created to return the smallest angle between the current one and the target:

$$gma(\theta) = \begin{cases} \theta & = \theta - 2\pi \quad \text{if } \theta > \pi \\ \theta & = \theta + 2\pi \quad \text{if } \theta < -\pi \\ \theta & = \theta \quad \text{otherwise} \end{cases} \tag{4.6}$$

And so, the desired target angle that will be adopted becomes:

$$\theta^{d}_{base-i} = \theta_{base-i} + gma(\angle(p_{\hat{object}} - p_{\hat{base-i}}) \ - \theta_{base-i}) \tag{4.7}$$

4.1.3 Bases velocity controller

This part of the controller aims to send commands in velocity to each base, given its designed position. As mentioned in 3, a low-level controller is already implemented. Thus, it is possible to model each base as a simple integrator, being directly commanded in velocity. However, the velocity input v^d_{base-i} should be evaluated w.r.t. the $i-th$ base frame. In this case, considering a state space, the model becomes slightly more complex and not more linear. Nevertheless, according to 2, is possible to build an MPC that carries the system to stability. Resorting to the cost function in 2.9, the minimization problem is :

$$\begin{aligned} \min_u V(t) = \min_u \sum_{\tau=t+1}^{t+N} & J(\tau) \\ \text{s.t.} \quad & x(\tau+1) = \boldsymbol{A}x(\tau) + \boldsymbol{B}u(\tau) \\ \text{s.t.} \quad & ||u(\tau)[0:1]||_\infty{}^1 < v_{max} \\ \text{s.t.} \quad & |u(\tau)[2]| < \omega_{max} \end{aligned} \tag{4.8}$$

where state under consideration is $x = [p_{\hat{base-i}}, \theta_{\hat{base-i}}] \in \mathrm{R}^3$ and the following matrices are taken into account:

$$\boldsymbol{A} = \boldsymbol{0}_3 \qquad\qquad \boldsymbol{B} = \boldsymbol{\mathcal{R}}_{W-Bi}$$

$$\boldsymbol{Q} = \begin{bmatrix} 2000 & 0 & 0 \\ 0 & 2000 & 0 \\ 0 & 0 & 1000 \end{bmatrix} \quad \boldsymbol{R} = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 100 \end{bmatrix} \tag{4.9}$$

The rotational matrix is computed from body to world frame and the values within the weight matrices Q and R are empirically determined. The other MPC parameters are chosen empirically too. In particular, the number of steps N selected to looking forward in the future is 4 and the step horizon Δt considered in the discretization in 2.17 is set to $0.1s$.

About the input limitations, the maximum linear velocity v_{max} and the maximum angular velocity ω_{max} are respectively $0.1m/s$ and $0.4rad/s$.

[1]L-∞ norm is defined as $||v||_\infty = \max_n |v[n]|$

Rotations

A problem that has arisen during the experiments concerns rotations. Figure 4.1.4 illustrates the problem in a graphical manner to aid comprehension. When the angle displacement between the formation and the object is big enough, i.e. when the small angle approximation becomes no more valid, the shortest path (the one in blue) leads to reducing the distance between the vertices and the center (from d_1 to d_2 in figure). Because the arms controller is designed to keep an object in the center of the equilateral triangle with an external radius of d_1, reducing that distance to d_2 shrinks the formation and therefore more pressure is applied to the object.

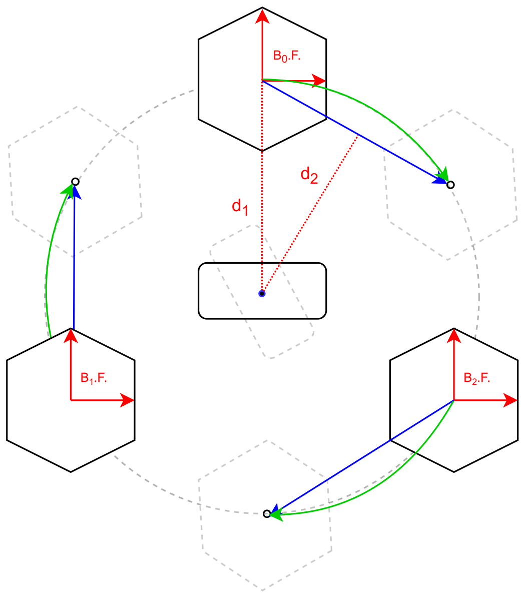


Figure 4.1.4: Representation of the rotation's issue

As a necessary consequence, a distance $d = d_1$ must remain constant throughout the movement. To accomplish this, a term related to the distance $J^\star$ is added to the cost function 2.9.

$$J^\star = \sum_{\tau=t+1}^{t+N} Q_d(||x(\tau) - x_{box}(\tau)||_2 - d)^2 \tag{4.10}$$

Both the distance maintenance and obstacle avoidance terms can be viewed as strong constraints that must be met along the minimization path, but this can lead to infeasibility issues in practice. As a result, the alternative proposed above is to directly add a quadratic term to the cost function that is equal to zero when the constraint is met and not when it is not. Another similar approach would be to use strict constraints that can be relaxed by including a slack variable that is minimized in the optimal solution.

4.2 Object Controller

The so-called Object Controller is responsible for the object motion within the formation. So basically, it simulates the behavior of the human operator on pushing or pulling the object in the desired direction to follow.
It's characterized, as is possible to see from figure 4.2.1, by a pipeline structure. At the top level, the *Navigation Controller* block provides the object's desired displacement δ^d , below stands the *Displacement Controller* that computes velocity references v^d_{arm-i} for each arm, and for each, the *Velocity Controller* computes the target positions for the related joints q^d_i . So essentially, when the human operator wants to move the formation, this circuit is interrupted and so the arms are commanded directly.

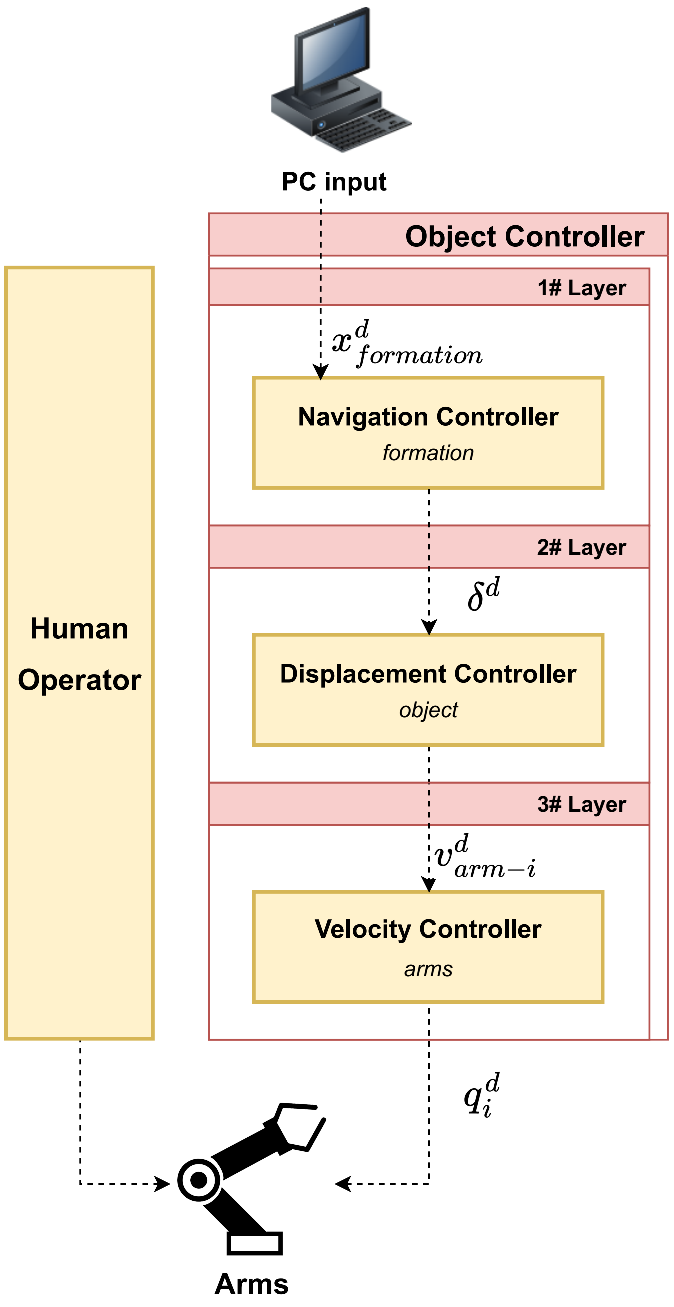


Figure 4.2.1: Graphical representation of the Object Controller

4.2.1 Navigation Controller

The Navigation controller, at the highest level, performs the same functions as the Bases velocity controller described above. In fact the central idea behind this block is to build an MPC designed for the whole formation that, given a target pose $x^d_{formation}$, provides the displacement reference for the box δ^d. So as was done above, resorting to the same minimization problem :

$$\begin{aligned} \min_u V(t) = \min_u & \sum_{\tau=t+1}^{t+N} J(\tau) \\ \text{s.t.} \quad & x(\tau+1) = \boldsymbol{A}x(\tau) + \boldsymbol{B}u(\tau) \\ \text{s.t.} \quad & ||u(\tau)[0:1]||_\infty < \delta_{xy-max} \\ \text{s.t.} \quad & |u(\tau)[2]| < \delta_{\theta-max} \end{aligned} \tag{4.11}$$

where state under consideration is $x = [\hat{p_{formation}}, \hat{\theta_{formation}}] \in \mathrm{R}^3$ computed as in 4.1-4.2. The entire formation is modeled as a simple integrator considering the displacement of the box δ^d w.r.t. the world frame. Thus, the following matrices are taken into account:

$$\boldsymbol{A} = \boldsymbol{0}_3 \qquad \boldsymbol{B} = \boldsymbol{I_3}$$

$$\boldsymbol{Q} = \begin{bmatrix} 1100 & 0 & 0 \\ 0 & 1100 & 0 \\ 0 & 0 & 550 \end{bmatrix} \quad \boldsymbol{R} = \begin{bmatrix} 700 & 0 & 0 \\ 0 & 700 & 0 \\ 0 & 0 & 300 \end{bmatrix} \tag{4.12}$$

with the following chosen empirically MPC parameters :
$N = 12,\ \Delta t = 0.1,\ \delta_{xy-max} = 0.06,\ \delta_{\theta-max} = \pi/16$.

Obstacle Avoidance

Bring into play an MPC, is possible to include additional terms within the cost function to satisfy other tasks, for instance, such as avoiding collisions with obstacles. In particular the following term :

$$J^{\star\star} = \sum_{\tau=t+1}^{t+N} \frac{Q_o}{(||x(\tau) - x_{obstacle}(\tau)||_2 - d)^2} \tag{4.13}$$

is added to 2.9, where Q_o is the weight given to the obstacle avoidance constraint and d the minimum distance allowed between the center of the formation and the center of the obstacle (d_1 in figure 4.1.4). The value chosen for $Q_o = 1000$.

4.2.2 Displacement Controller

Another MPC is also involved in this level of the structure. To keep things simple, only the kinematics of the object grasped (basically another integrator) is considered, so other calculations to provide convenient inputs and outputs are required. A graphical representation of the pre and post computation required for the work process is reported in 4.2.2.

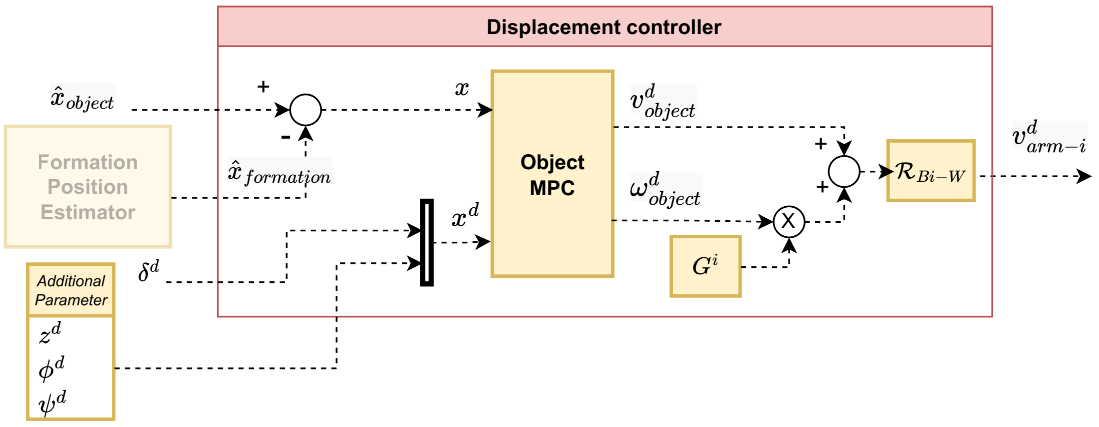


Figure 4.2.2: Displacement controller structure

Starting with δ^d from the previous block, the target state is:

$$x^d = \left[\delta^d_x,\ \delta^d_y,\ z^d,\ \phi^d,\ \psi^d,\ \delta^d_\theta\right] \in \mathbb{R}^6$$

where the additional parameters $z^d,\ \phi^d,\ \psi^d$ can be freely chosen. For all the experiments are respectively set to $0.1m,\ 0,\ 0$.
The state instead is evaluated as the following:

$$\begin{aligned} x &= \left[\hat{\delta}_x,\ \hat{\delta}_y,\ \hat{z}_{object},\ \hat{\phi}_{object},\ \hat{\psi}_{object},\ \hat{\delta}_\theta\right] \in \mathbb{R}^6 \\ \text{where}: \hat{\delta} &= \left[\hat{\delta}_x,\ \hat{\delta}_y,\ \hat{\delta}_\theta\right] = \left[p_{o\hat{b}ject} - p_{form\hat{a}tion},\ \theta_{obj\hat{e}ct} - \theta_{form\hat{a}tion}\right] \end{aligned} \tag{4.14}$$

At last, the input $\left[u = v^d_{object},\ \omega^d_{object}\right] \in \mathbb{R}^6$ is regarded as the object's desired velocity (linear and angular) w.r.t the world frame.

Thus, resorting to the same structure of the MPC described above:

$$
\begin{aligned}
\min_u V(t) = \min_u & \sum_{\tau=t+1}^{t+N} J(\tau) \\
\text{s.t.} \quad & x(\tau+1) = \boldsymbol{A}x(\tau) + \boldsymbol{B}u(\tau) \\
\text{s.t.} \quad & ||u(\tau)[0,1,2]||_\infty < v_{max} \\
\text{s.t.} \quad & ||u(\tau)[3,4,5]||_\infty < \omega_{max} \\
\text{s.t.} \quad & ||v^d_{arm-i}||_\infty < v_{max} \quad \text{for } i = 0,1,2
\end{aligned}
\tag{4.15}
$$

with the following matricies considered:

$$
\begin{aligned}
\boldsymbol{A} = \boldsymbol{0}_6 \qquad & \boldsymbol{B} = \boldsymbol{I}_6 \\
\boldsymbol{Q} = \text{diag}&\{8,8,8,2,2,2\} \\
\boldsymbol{R} = \text{diag}&\{0.5,0.5,0.5,4,4,4\}
\end{aligned}
\tag{4.16}
$$

As pointed out previously, the MPC are chosen empirically : $N = 4$, $\Delta t = 0.01$, $v_{max} = 0.3$, $\omega_{max} = 0.1$.
The constraints are associated to the maximum velocity of the object as well as the maximum velocity of the arms. The block uses the Grasp Matrix $\boldsymbol{G}$ knowledge to estimate them. This matrix is created by simply stacking the grasping position points of the EE with respect to the object's center. These positions are evaluated at the start and remain relatively constant throughout all movements. Knowing this, the desired velocities of the arms can be calculated as follows:

$$
v^d_{arm-i-WF} = v^d_{object} + \omega^d_{object} \times \boldsymbol{G}^i \tag{4.17}
$$

That formula is obtained by assuming that in order to move the box with the desired linear velocity, all of the arms must move at the same velocity. To obtain an angular velocity in the object, another contribution is required (different for each arm).
However, the velocity must then be evaluated in relation to the reference frame

of the $i - th$ Rosie before being sent to the next and final block, so:

$$v^d_{arm-i} = \mathcal{R}_{Bi-W}\ v^d_{arm-i-WF} \tag{4.18}$$

This system enables the object to be manipulated by defining any possible orientation to be tracked. Because only linear velocity commands are considered in the following block, the velocity sent in this manner is a vector in $\mathbb{R}^3$, but to be processed it is transformed into a vector in $\mathbb{R}^6$ as the following :

$$v^d_{arm-i} \rightarrow [v^d_{arm-i}, 0, 0, 0] \in \mathbb{R}^6$$

The decision to send velocities rather than directly position was made at first to avoid instabilities caused by inaccuracies in the model's knowledge and to make the movements smoother. Some initial trials were performed using an effectively position controller, but the results were as expected, so the method based on velocities was chosen. Furthermore, when sending velocities, the position of each arm is not taken into account, increasing the system's robustness in the event of an underestimation of that.
Another notable feature of this object model, which does not account for dynamics and instead relies on the possibility of imprinting a velocity, is its independence from the object mass. It is thus possible to manipulate various objects without having to perform a complex estimation procedure at the outset.

4.2.3 Velocity Controller

The closing block is associated to processing the joint state reference q^d_i for each arm with $i = 0, 1, 2$, beginning with the desired velocity for the EE.
To track a velocity reference, the Cartesian position to be sent to the arms is computed with an integrator, and any position is maintained whenever the received velocity is 0. In order to provide a starting position for the arms position controller in 3.3.2 a desired position x^d_i is considered at the start. Instead, the ϕ^d, ψ^d, θ^d angles are chosen to follow the orientation of the object and keep the

grippers aligned at all times.
Considering directly the discrete implementation:

$$x_i^\star(k) = x_i^d(k) + v_{arm-i}^d(k)\Delta t \tag{4.19}$$

where δt is the step time at which v_{arm-i}^d is updated and the desired state is used instead of the actual one. This is extremely useful for controlling the orientation of the end effector without interfering with the velocity controller. Starting from an equilibrium point $x_i(k) \approx x_i^d(k)$, without sending any other commands, the result should be the same as using the actual position. As a result, the following are used to send angular references , in particular for the θ angle, which varies the most during the experiments:

$$\begin{aligned} \theta_i^d &= \text{gma}(\theta_{object} - \theta_{base-i} + \pi) \quad \text{with: } i = 0 \\ \theta_i^d &= \text{gma}(\theta_{object} - \theta_{base-i}) \quad \text{with: } i = 1, 2 \end{aligned} \tag{4.20}$$

where gma is the same function described in 4.6.
Using 2.5, the reference for the joint states is thus :

$$q_i^d(k) = q_i(k) + \boldsymbol{J_A}^\dagger \left(x_i^d(k) + v_{arm-i}^d(k) - x_i(k)\ \right) \Delta t \tag{4.21}$$

Gravity Compensator:

The actuator torque required to maintain a fixed position, caused by the weight of robot links, can be a significant problem. A common method used in robot design to establish balance throughout the range of motion is gravity compensation. With the same procedure adopted in [15], an offline estimation of the gravity and friction forces is computed in order to obtain a reliable gravity compensator term inside the controller. First of all, to easily compute the input u, the controller structure (in 3.3.5) is simplified using only a proportional controller K_p in the joint space. Thus the contribution of the gravitational force is parameterized through the vector θ and at an equilibrium point q the following applies:

$$W(q)\theta + f_s = K_p(q^\star - q) \tag{4.22}$$

where f_s is the friction force, $q^\star$ is the desired position and

$$W(q) = \begin{bmatrix} w_{11}(q) & w_{12}(q) & \dots & w_{1(n-1)}(q) & w_{1n}(q) \\ 0 & w_{22}(q) & \dots & w_{2(n-1)}(q) & w_{2n}(q) \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & w_{(n-1)(n-1)}(q) & w_{(n-1)n} \\ 0 & 0 & \dots & 0 & w_{nn}(q) \end{bmatrix} \quad w_{ij}(q) = \bar{g}^{\mathrm{T}} \left(\partial R_B^j / \partial q_i\right) p_j \tag{4.23}$$

with R_B^i the rotation matrix of frame i with respect to the base frame, pi the directional vector in frame i pointing to the center of mass mi of link i and $\hat{g}$ = $[0, 0, 9.81]^T$.

At this point evaluating 4.22 at 2 different points $q_1^\star, q_2^\star$ and considering the 2 errors e_1, e_2 w.r.t. the actual positions q_1, q_2, is possible to cancel out the friction term f_s, that is quite the same for both the two positions, just making the difference:

$$\begin{aligned} W(q_2)\theta - W(q_1)\theta &= K_p e_2 - K_p e_1 \\ W(q_2) - W(q_1)\theta &= K_p(e_2 - e_1) \\ \hat{\theta} &= K_p(W(q_2) - W(q_1))^\dagger(e_2 - e_1) \end{aligned} \tag{4.24}$$

then the friction force can be estimated :

$$\hat{f}_s = K_p e_1 + W(q_1)\hat{\theta} \tag{4.25}$$

4.3 Starting Procedure and Flowchart

The starting procedure used during the experimental setup will be explained in this segment, as will the flowchart in 4.3.1.

First, the robots wait for a connection with the MCS, which provides the position of all the robots and the object. The Arms then begin to move to a predefined

configuration, opening the grippers. Meanwhile, the formation base controller begins to fulfill its part by providing instructions to the bases in order for them to reach the target position around the object.
Once the final formation geometry has been completed, a request to close the grippers is committed; if the grippers are still open, they are closed. The arms then set a small velocity along the x direction of the box to generate force and reduce the possibility of losing the grasped item (*”push box”* block in the figure). When a desired position and then a target displacement are received, the arms start moving in order to produce the object’s displacement. This causes the bases that are no longer in the target position to be triggered. As a result, until the target displacement is set to zero, the formation will move to the desired position.

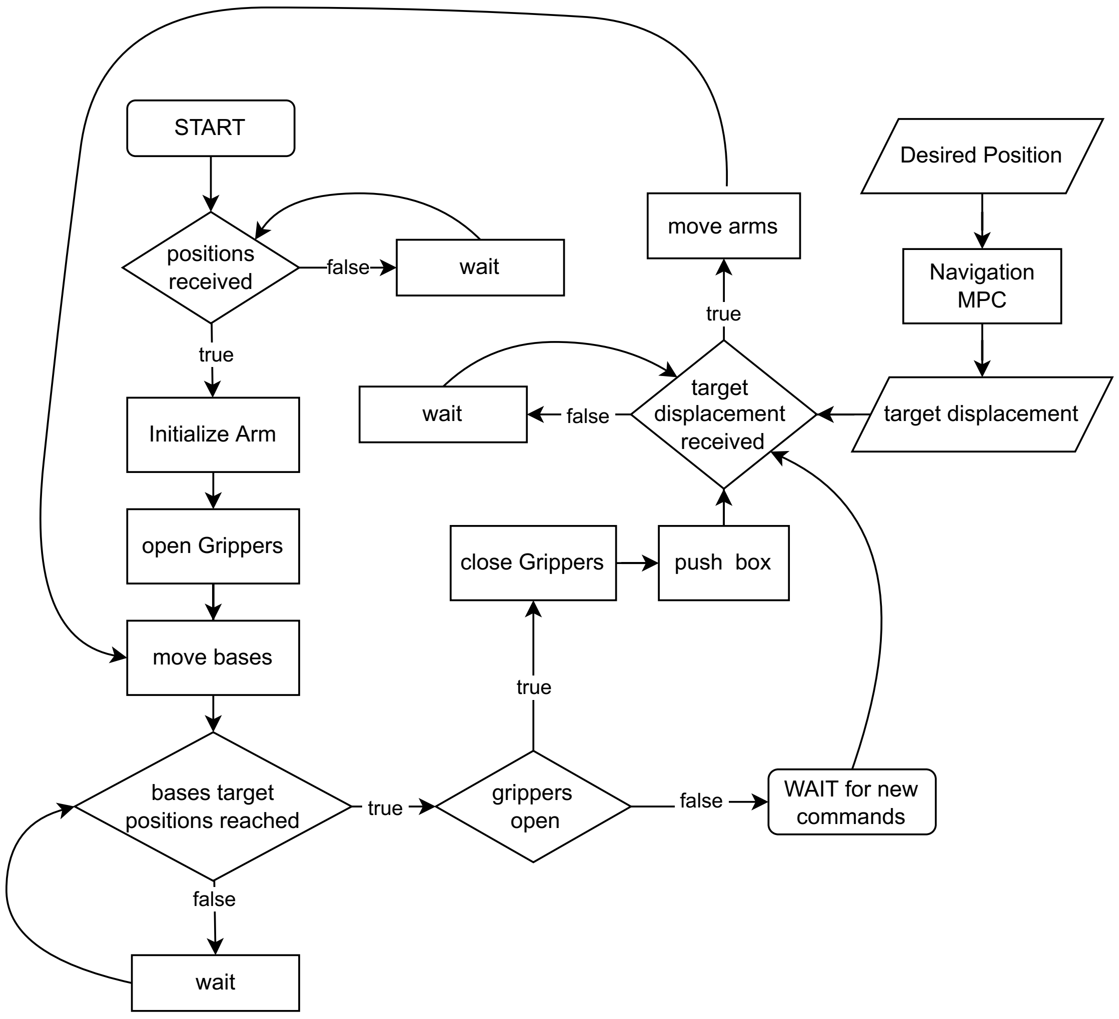


Figure 4.3.1: Work flow described graphically by a flowchart.

4.4 ROS implementation

The ROS realization of the control scheme described above is presented below (figure 4.4.1 depicts that structure). Starting from the lowest level, a node to control the gripper (*"RosieX-gripper-node"*), the arm (*"RosieX-arm-node"*) and the base (*"RosieX-base-node"*) is implemented for each *"RosieX"* with $X = 0, 1, 2$. These nodes are executed directly inside the Robot Control Computer, which is present on each base, using the network structure described in 3.3.2. Instead, all of the other nodes run in parallel on a lab's remote PC.
The decision to assign each part of the controller to a separate node and have

them communicate via dedicated topics was made specifically to allow all parts of the code to run simultaneously. The PC cannot handle every script at the same time, so this is clearly not true simultaneity, but a high CPU clock can achieve comparable results.
Because each of nodes' roles was thoroughly explained above, only a few notes about the code implementations will be reported here.

4.4.1 Code Features

First of all a topic called *"/Formation/init"* is created to establish a communication channel between most of the nodes. Text messages are used to initiate or terminate different operations. For instance, when a *"RosieX-base-MPC"* reaches the target position around the object, a message *"close-gripper-X"* is sent to *"Arm-node"*. If the grippers are not already closed, the latter collects these messages until it receives them from all the three nodes (namely the aforementioned message with $X = 0, 1, 2$). At which point the node is allowed to send commands to close all the gripper simultaneously.
Another method adopted to improve safety and prevent problems caused by lack of communication is reported in the following. The data from the topics is stored in variables within each node and thus remains constant if not updated. Thus a counter is implemented to track the time between receiving critical data messages. If this time exceeds a certain threshold (say, half a second), the node is stopped until new data arrives. It is therefore possible to avoid potentially dangerous situations in the event of data loss between nodes. Such is extremely crucial, for example, with velocity commands sent from *"BOX-MPC"* to *"Arm-node"* because if transmission fails, the arm may continue to move without control. Since all of the nodes are connected in cascade, this feature makes it possible to halt the system by simply interrupting the communications at the top level.
The entire code, with all the scripts and the classes adopted can be found at `https://github.com/Achille1998/Collaborative_Transportation.git`.
In particular four classes are developed: one for the arm, one for the base, one

for the object and one for the entire formation. Inside each class is present all the code to subscribe and publish in the related topics as well as the code to run the algorithms (like the MPC algorithm proposed in 1 . In this way the scripts are simplified, initializing only the object and running methods in loops.

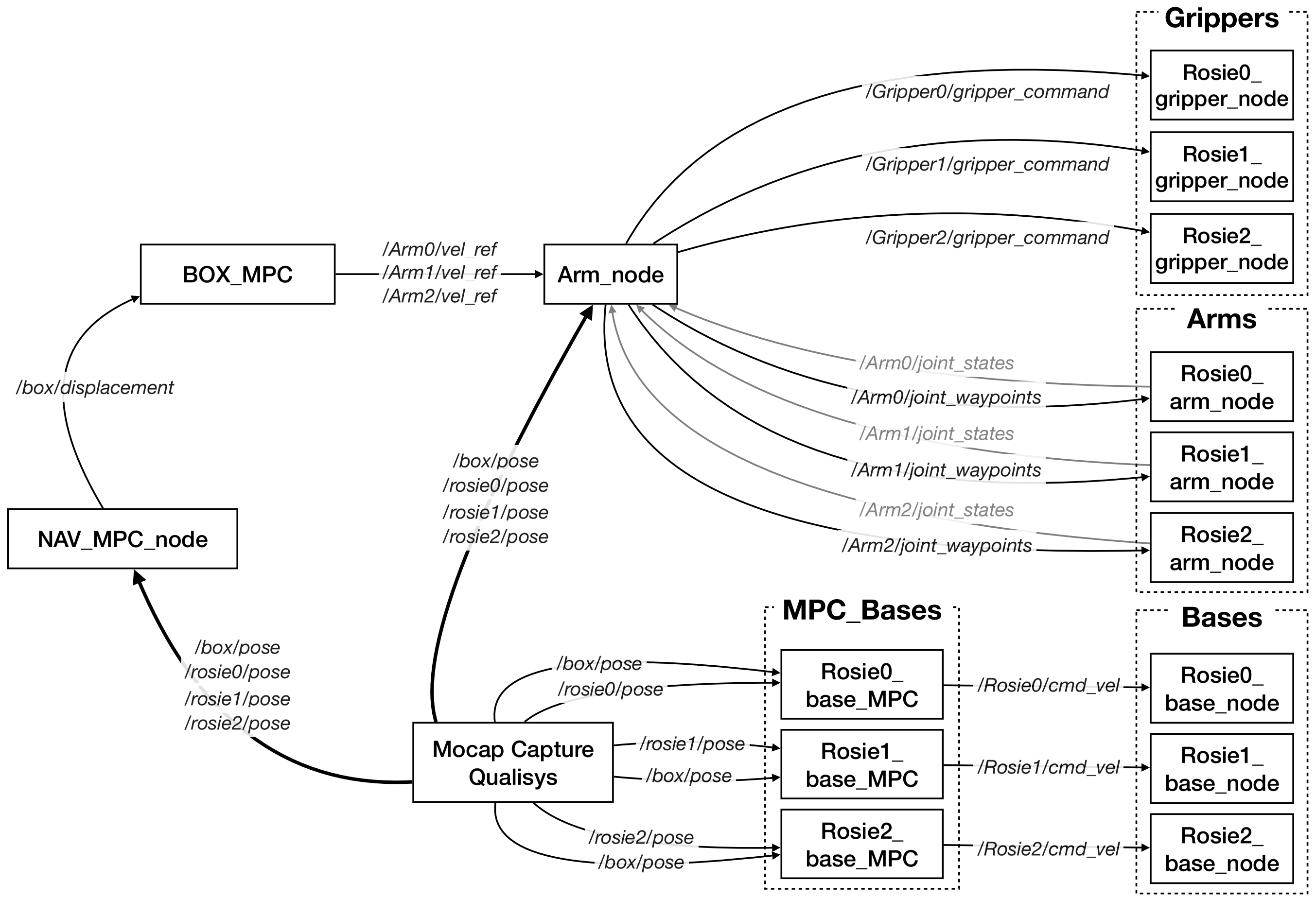


Figure 4.4.1: Ros structure of nodes and topics involved for the proposed controller. To simplify the notations the *"/Formation/init"* linkage are not reported.

Chapter 5

Experimental Results

In this chapter, the numerical results and related graphs are presented to demonstrate the controller's feasibility. The hardware and settings are extensively described in section 3, but a brief summary is provided here.
Three mobile manipulators called *rosie*, designed by Hebi Robotics, are employed for the purpose of the experiments. Each *rosie* is made by an holonomic mobile base with 3 omni-wheels and a 6 DOF robotic arm. The EEs, on the other hand, are adaptative grippers, called Kino gripper that are not provided by Hebi but 3D-printed and built directly in the laboratory.
All the data from the experiments are saved resorting to *rosbag* package, which allows saving all the topics from the Ros environment in order to replay the information flow offline. Some videos have also been recorded and can be found at the URL: `https://www.youtube.com/channel/UCBgQH-vvFeMrxny7d6Ch-fw`.
All of the tests take place in the Smart Mobility Lab (SML),which is located on the KTH campus, in Stockholm. Here an effective combination of automatic control, computer science, and mechatronics is being used to develop the next generation of mobile systems for intelligent transportation and smart driving.[1].
Figure 5.0.1 shows a view of the LAB from the top. The dashed line represents the working limits of the motion capture and the reference frame is reported

[1]The web page of the laboratory can be found in `https://www.kth.se/is/dcs/research/control-of-transport/smart-mobility-lab/smart-mobility-lab-1.441539`

too. In particular the dimensions are, respectively, in the x, y directions, $5m$ and $4m$.

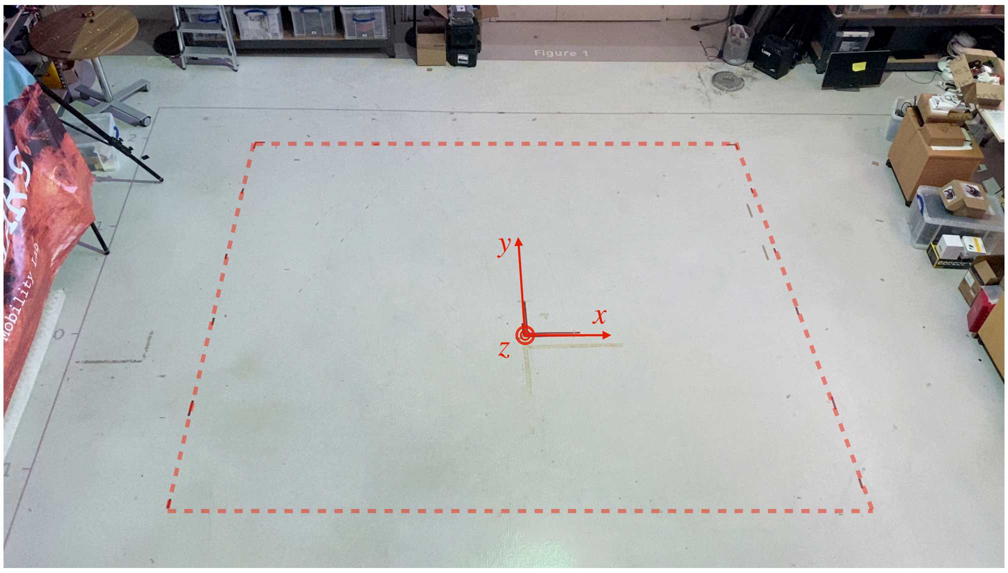


Figure 5.0.1: Picture of the Smart Mobility Lab from the top.

5.1 Assumptions

As previously stated, some assumptions are made in order to avoid considering aspects that are unnecessary to this work and to simplify the experimental setup. For the sake of convenience, all the assumptions are summarized below :

- Along with all the trajectories, the formation geometry remains static. The bases cannot move too far away from the vertices of the equilateral triangle with the object in the center.
- The position of all the robots and the object, as well as the obstacles, is provided by the MCS. As a result, all the agents are aware of the object's location, shape, the target position for the entire formation, and the position of the other robots and obstacles.
- In order to lift the object, the grasping points where the EE tries to pick the object are predetermined and computed as relative positions w.r.t. object's center.

In the next paragraphs, single elements of the controller scheme will be tested separately to reach, in the end, the final experiments, in which all the parts are involved simultaneously.

5.2 Test arm velocity controller

In this first test, the velocity arm controller, described in 4.2.3 is subjected to analysis to demonstrate the accuracy and absence of latency. This is critical because the MPC designed for the object in the Object controller, in order to function properly, must be able to imprint velocity using to the arms. To evaluate this, some markers of the MCS was added only for the purpose of measuring the velocity of the EE and relating it with the commands sent.
Figure 5.2.1 shows this comparison.

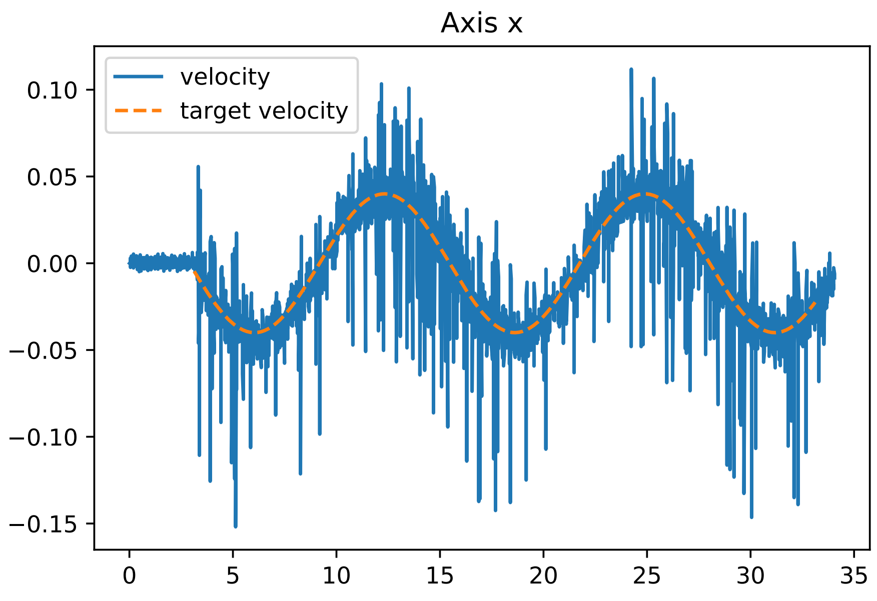


Figure 5.2.1: Test concerning the velocity tracking by the arms: $[m/s][s]$

The horizontal axis represents the time in seconds, while the vertical axis represents velocity in meters per second. The velocity trajectory to be followed is a sinusoid with a frequency of $5\ rpm$ and an amplitude of $0.5\ m/s$. Despite the presence of a significant noise in the measurements, due to marker position oscillations, it's clear that the robot is capable of good tracking, and thus this first part of the controller is validated .

5.3 Test gravity compensator

Secondly, the subject of the test will be the gravity compensator. In order to validate the procedure explained in 4.2.3, twenty different randomly chosen joint states are considered. In figures 5.3.1, 5.3.2 is represented the comparison between the real torque and the estimated one by the gravity compensator. To

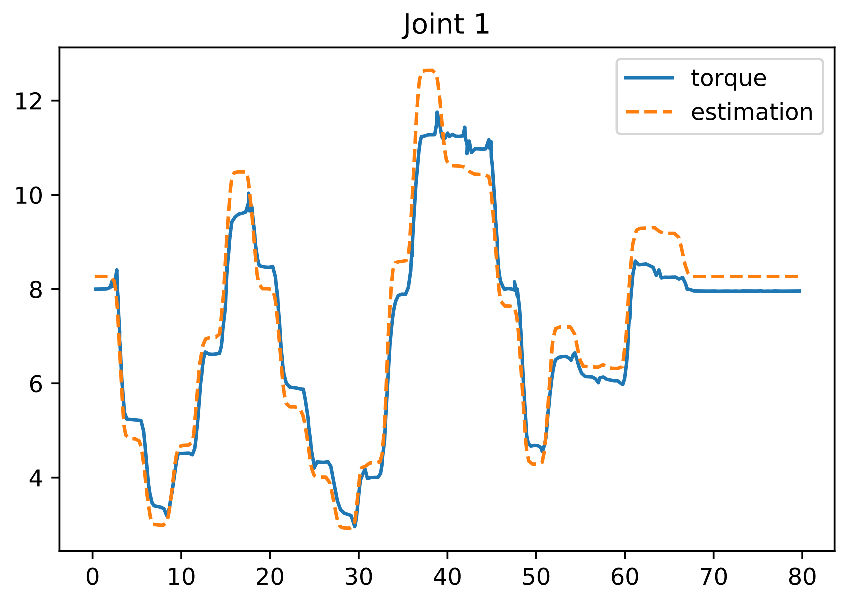


Figure 5.3.1: Test of the gravity compensator performance on joint 1: $[N/m][s]$

examine the difference at the steady state, a few seconds are added after each movement to evaluate the contribution of only the gravitational term. Because it is greater at the top of the arm , with the entire structure to be lifted, the second and third joints were chosen for the test.
As previously stated, the x-axis considers time in seconds while the torque is measured in N/m. The latter is obtained using torque sensors, which are

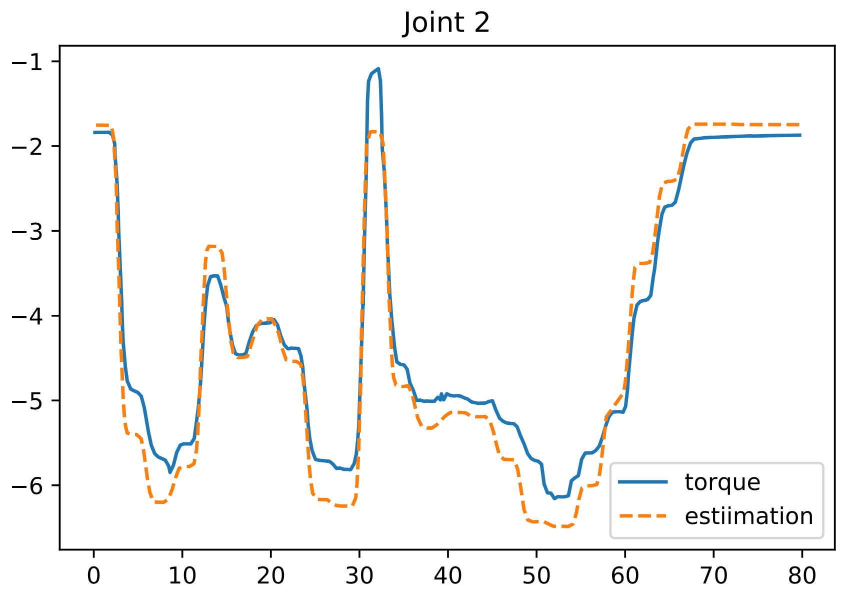


Figure 5.3.2: Test of the gravity compensator performance on joint 2: $[N/m][s]$

standard on all motors (for more information, see section 3.3.1).
The testing indicates that the estimate is close enough to the true value. Because the integral action exists within the controller, even a rough estimate is sufficient to achieve a null steady-state error.

5.4 Test displacement controller

The movements in the 3D space of the box through the three arms together are considered in the third test. Because the entire controller is now up and running, a portion of the disturbances are caused by the movements of the bases during the test. Figures 5.4.1, 5.4.2, 5.4.3 presents the reference provided by the Navigation controller, described in 4.2.1 and the real displacement of the box, estimated as using 4.14 and 4.1.

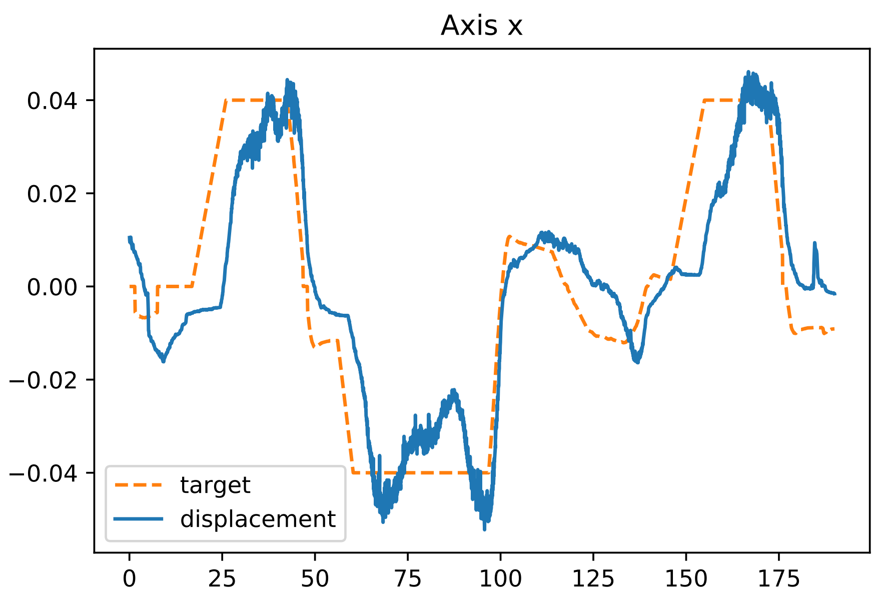


Figure 5.4.1: Test concerning the displacement on axis x: $[m][s]$

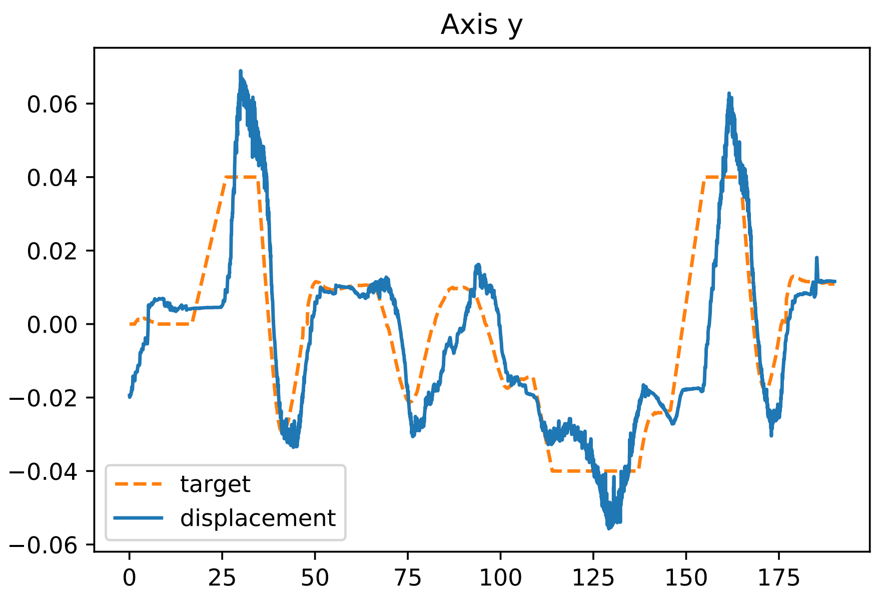


Figure 5.4.2: Test concerning the displacement on axis y: $[m][s]$

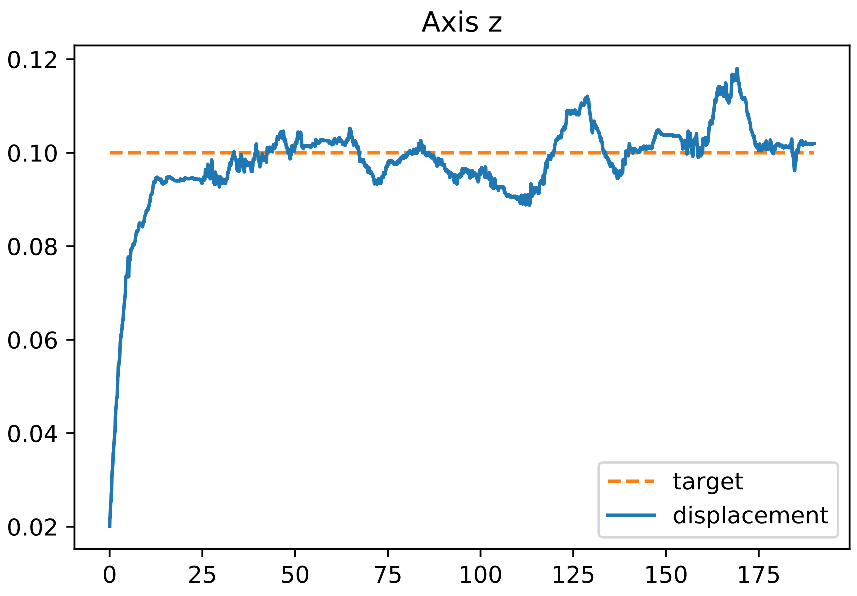


Figure 5.4.3: Test concerning the displacement on axis z: $[m][s]$

The time in seconds is considered for the horizontal axis while the target and measured displacements are expressed in meters.
Even if the tracking appears to be imprecise, it is not particularly important in practice because what matters most is the direction of the movements.
Furthermore, the presence of delay can be explained by the maximum velocity constraints for the arms and the maximum velocity permitted for the bases. These limits are set to prevent the box from falling due to high mechanical stress situations.

5.5 Test base velocity controller

The final component to be evaluated separately is the Base velocity controller described in 4.1.3. The target position is computed as in 4.4, taking rosie1 into account ($i = 1$ in the formula), starting from the position measured by the Motion Capture System, which provides the robot's position too. Graphs 5.5.1 and 5.5.2 show the results.

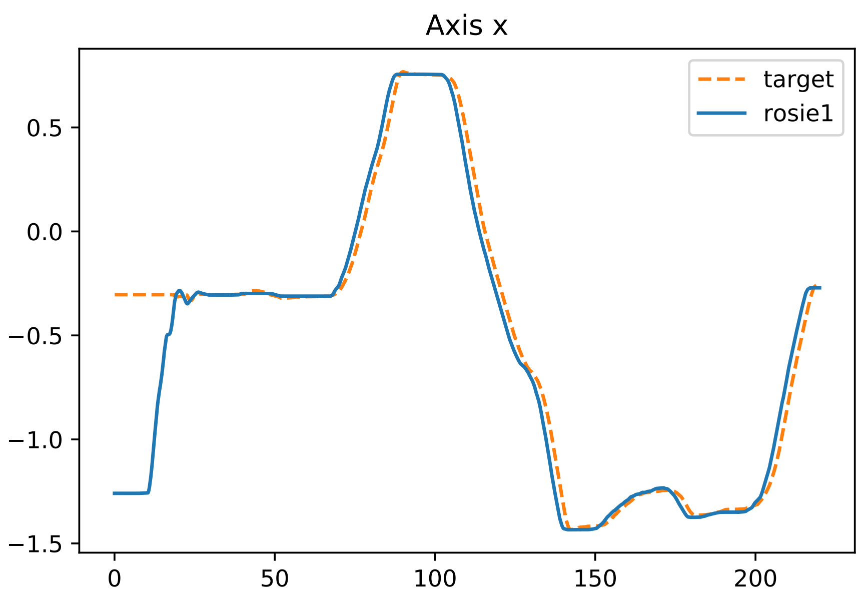


Figure 5.5.1: Test to verify the tracking by the MPC base on the x axis: $[m][s]$

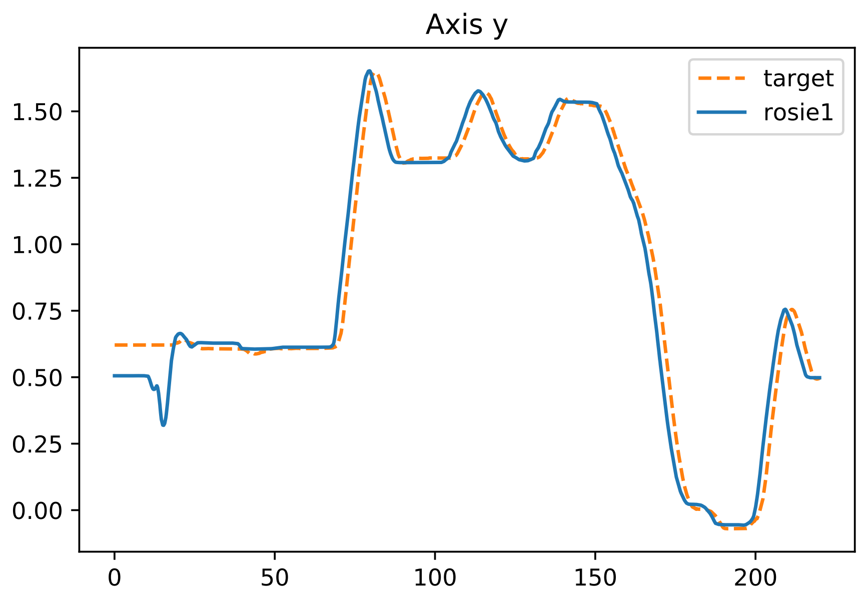


Figure 5.5.2: Test to verify the tracking by the MPC base on the y axis: $[m][s]$

The x-axis represents time in seconds, while the y-axis indicates the position, in meters, of the mobile base rosie 1 (the blue solid line) and the target position

(the orange dashed one).
Because the object was not attached to the base at first, the distance to cover was considerable. After that, the position error remains always very small because the arms'maximum displacement is $0.06m$. So, as expected, the tracking is flawless, attributable to the object and the robot attached.

5.6 Test circle tracking with fixed orientation

To assess the formation's ability to reach different points in the plane, the target positions sent to the robots in this test follow a circular trajectory:

$$x^d_{formation} = \left[r\cos(\omega t)\,, r\sin(\omega t),\ \frac{\pi}{2} \right] \qquad \omega = \frac{2\pi}{T}, \quad T = 180\,s, \quad r = 0.5\,m$$

considering $x^d_{formation}$ as defined in 4.2.1, which represent the position in the x, y directions and the desired formation orientation around the z axis. In this case, is fixed and set to $\frac{\pi}{2}$. The reference for the z-axis is instead set to $0.1\,m$. Figure 5.6.1 and **??** show the comparison between the reference and the position of the formation obtained as in 4.1.

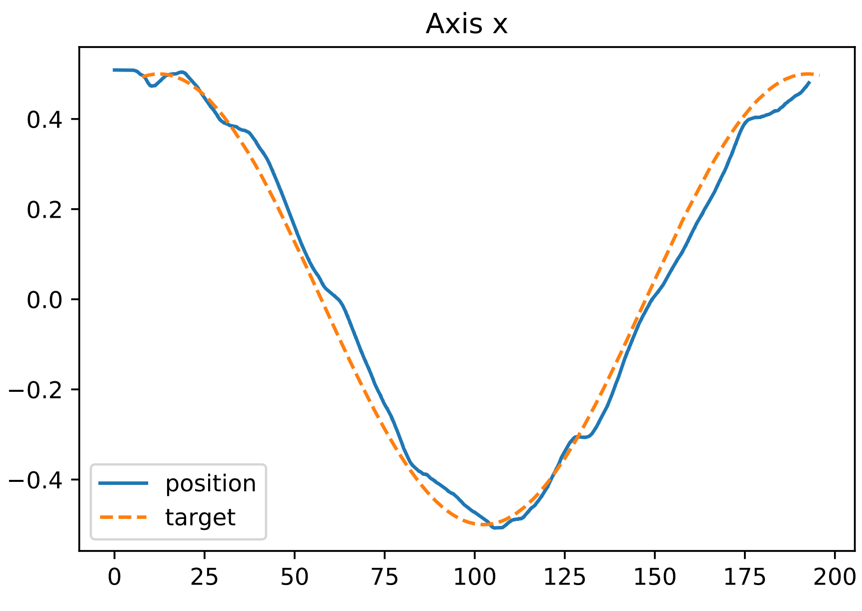


Figure 5.6.1: Circle tracking by the entire formation in the the x axis :$[m][s]$

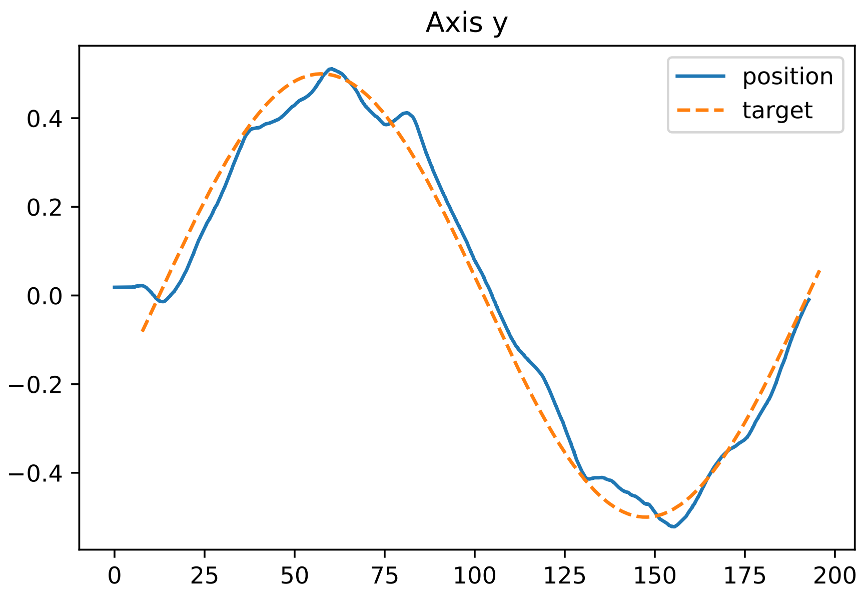


Figure 5.6.2: Circle tracking by the entire formation in the the y axis :$[m][s]$

Some errors in the tracking are due mostly to arms's difficulty in precisely moving the object, as observed in the test dedicated above to validating the displacement controller.
Throughout this text, next figures 5.6.3 and 5.6.4 highlight the mechanism of the Displacement controller. The same structure as the graphs described above is used, and the controller's expected behavior is also reported (the dashed green line in the figure).
To be specific, in order to track the circle, the object should either compute a circle with the same frequency ω and amplitude defined by the maximum allowable displacement:

$$\hat{\delta}^{d}_{formation} = \left[\delta_{xy-max}\cos\left(\omega t + \frac{\pi}{2}\right), \delta_{xy-max}\sin(\omega t),\ 0\right]$$

It is critical to remember that this behavior is what we could expect from an ideal model, without taking into account all of the real-world interactions, such as delays and elastic and compression factors during the manipulation of the box. Despite this, the results are satisfactory, and the object's movements are close

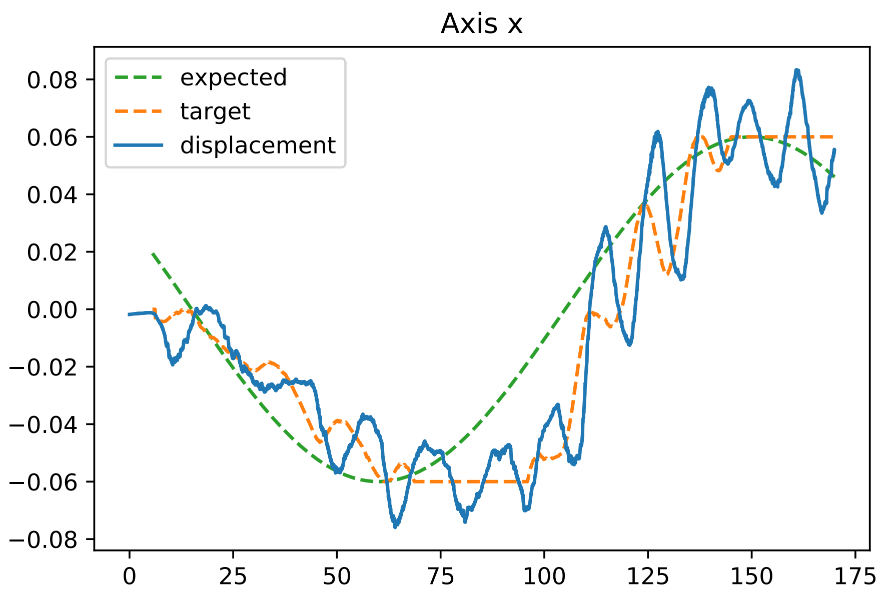


Figure 5.6.3: Displacement of the object during the circle tracking respect the target and the expected behavior in the x axis: $[m][s]$

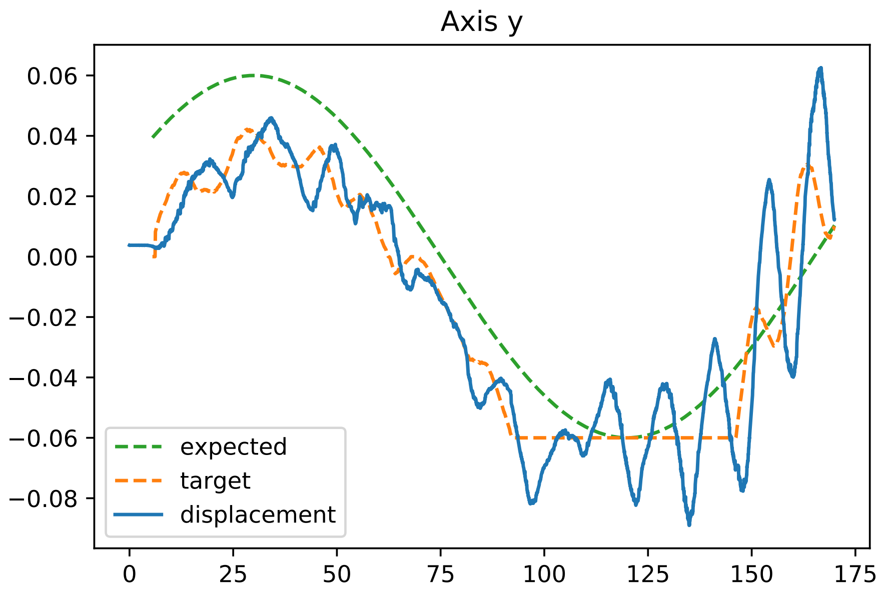


Figure 5.6.4: Displacement of the object during the circle tracking respect the target and the expected behavior in the y axis: $[m][s]$

to the expected performance.

5.7 Test circle tracking with orientation

The preceding test is repeated, but with a non-constant reference for the angle of the formation and using the same parameters as before:

$$x^d_{formation} = \left[r\cos(\omega t), r\sin(\omega t), 0.1, \frac{\pi}{2} + \omega t\right]$$

Figure 5.7.1 depicts the trajectory tracking in the $xy-$plane, whereas figure 5.7.2 captures some frames of the robots moving along the circle[2].

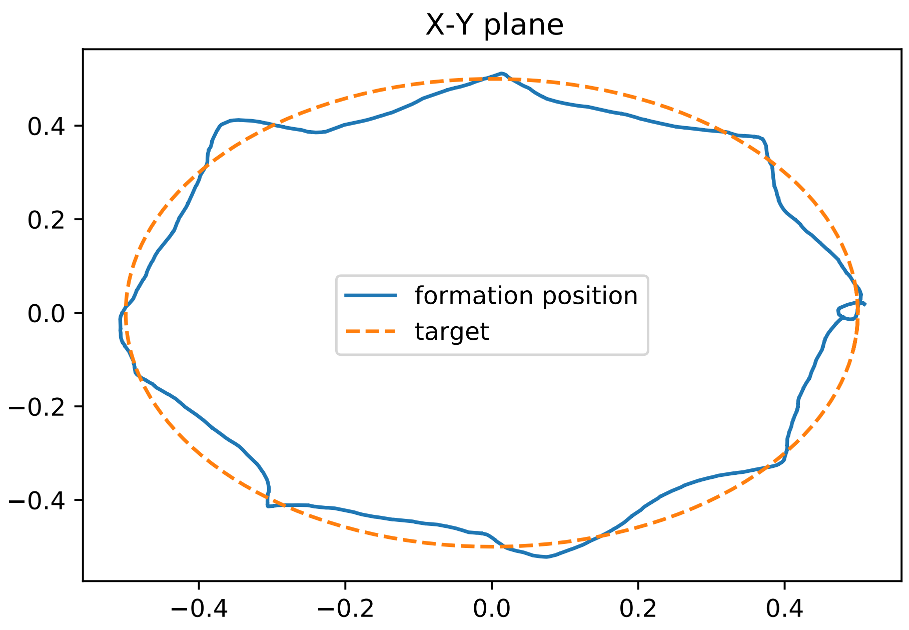


Figure 5.7.1: Tracking of a circle by the entire formation with a non constant orientation reference in the $x - y$ plane : $[m][m]$

In this test, maintaining a fixed displacement for the box (in the x direction relative to the formation's reference frame) while the orientation is changed to

[2]Due to technical difficulties during the recording, a full video of this test is not available at the URL mentioned above.

follow the circle is where the robots will encounter the most trouble.
Problems with the quantification of the formation's angle in particular also arise because every robot's movement can influence the estimate. Despite all that, the results are quite good and demonstrate that this controller can compute turning maneuvers.

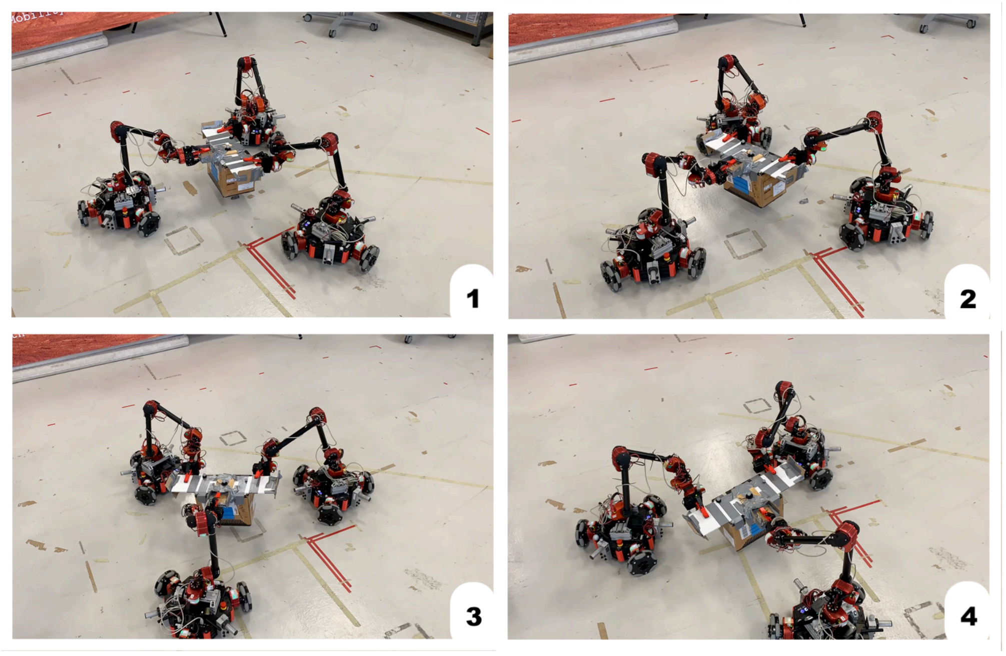


Figure 5.7.2: Some snapshots of the test concerning the tracking of a circle with a non-constant formation reference angle.

5.8 Test obstacle avoidance

The last experiment, which does not require the presence of a human operator, demonstrates the system's ability to change the path to reach the target position in the presence of an obstacle. The trajectories computed are illustrated in the image 5.8.1. Different lines are drawn to illustrate how the formation behaves differently depending on whether or not there are obstacles present. The solid red line is the way to reach $(-1.0, -1.5)$ starting from $(1.0, 1.0)$, affected by the obstacle's presence. In contrast, the dashed red line indicates the path calculated by the formation from $(-1.0, -1.5)$ to $(1.0, 1.0)$ without the obstacle.
Obviously, this is not an optimal path, but it is sufficient to avoid colliding

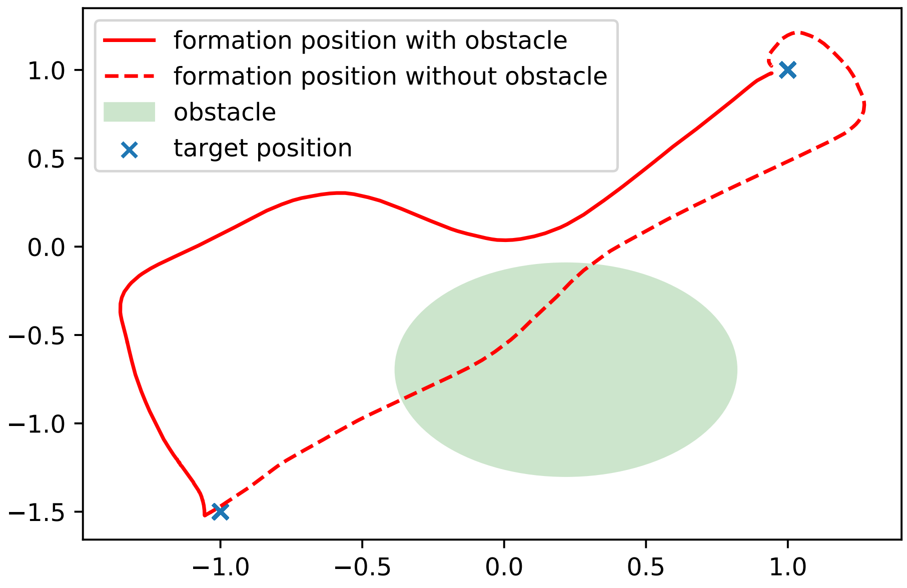


Figure 5.8.1: Representation in the $xy-$plane to prove the capability of the system to avoid an obstacle : [m][m]

with the obstacle and remain in a safe region. Other tests with obstacles in movements have been conducted, but the size of the arena in which the motion capture is capable of providing measurements makes it difficult to achieve satisfactory results.

5.9 Test human interaction

Ultimately, a human operator moves the formation directly in this final test. The gravity compensator mode is activated as soon as the robots lift the object, and the new positions of the object are preserved as long as the operator maintains contact.

Image 5.9.1 reveals a frame of the video with the related URL in the description, while figures 5.9.2 and 5.9.3 report the position of the formation and the displacement decided by the operator.

In this representation, the real displacement was multiplied by a factor of 10 to make both lines visible in the same graph.

The box is pushed in the x direction at the start ($T \approx 10\,s$), and after a delay of approximately 5 seconds, the formation begins moving in that direction until the displacement returns close to zero ($T \approx 25\,s$). As a result, the operator has complete control over the formation, allowing him or her to move it to any desired location.

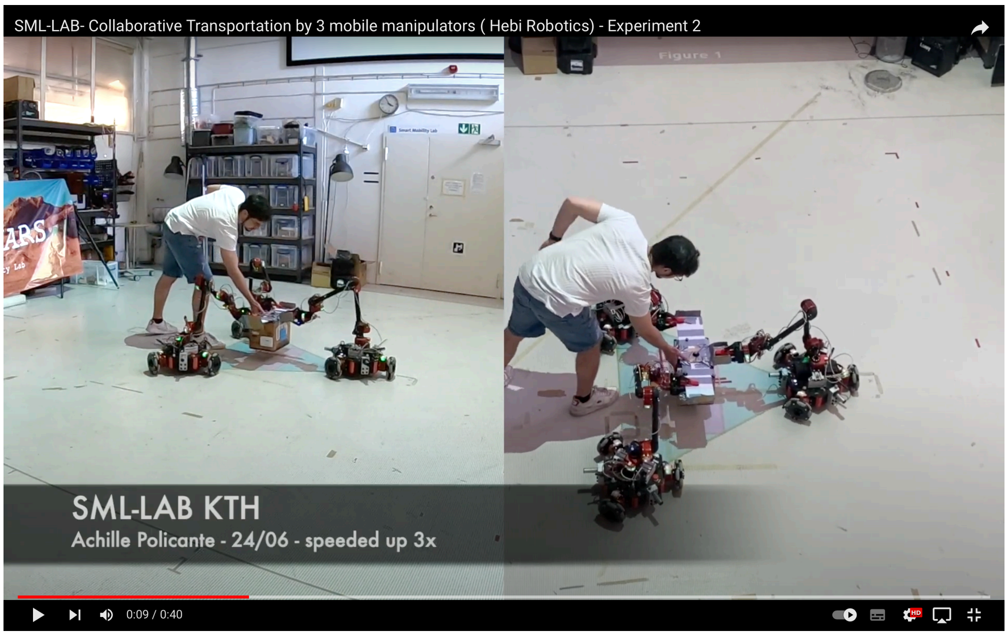

SML-LAB- Collaborative Transportation by 3 mobile manipulators (Hebi Robotics) - Experiment 2
SML-LAB KTH
Achille Policante - 24/06 - speeded up 3x
0:09 / 0:40

Figure 5.9.1: The URL of the complete video can be found at:
`https://youtu.be/mfdn_poNz6s`

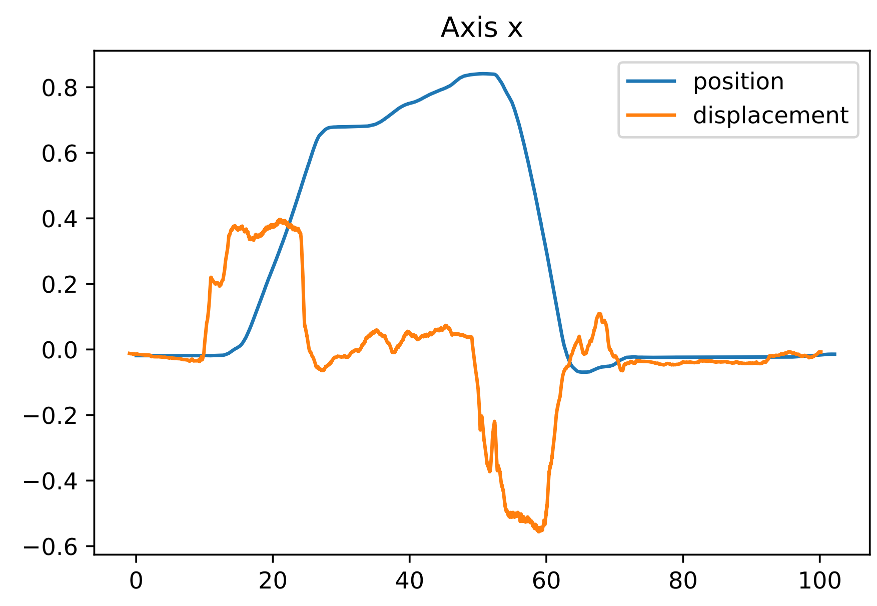


Figure 5.9.2: Position and displacement under human control on axis x: $[m][s]$

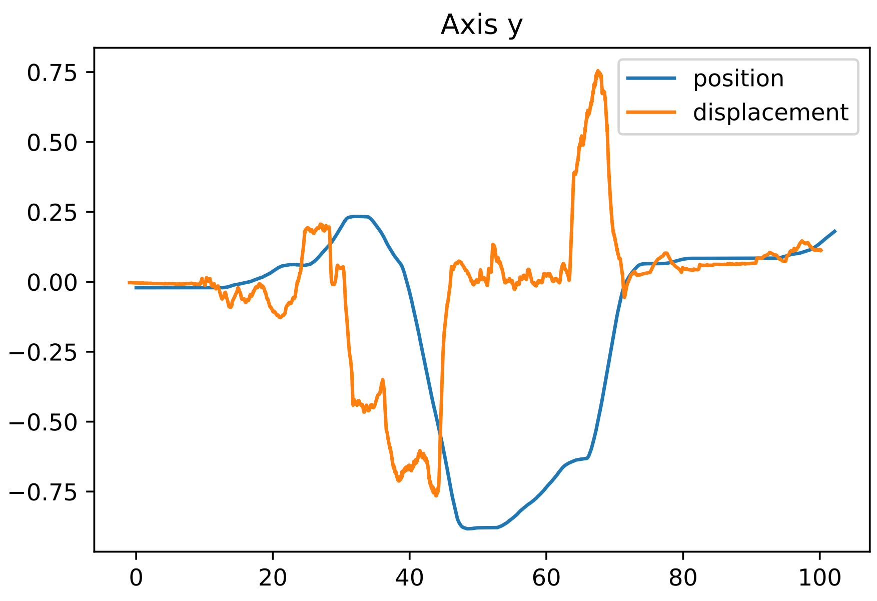


Figure 5.9.3: Position and displacement under human control on axis y: $[m][s]$

Chapter 6

Conclusions

A centralized controller to supervise three mobile manipulators is presented in this work. It is demonstrated that the controller is able to achieve the two main challenges described in the introduction: enabling collaborative transportation and allowing a human operator to directly control the formation.
Starting through the existing research, both pushing and pulling approaches are used, being in the presence of grippers. The geometry of the formation is static and an equilateral triangle is chosen, with one robot at each of its vertices and the object being handled and so lifted in the middle.
Two main blocks make up the planned controller: one deals with the bases (*Formation Bases Controller*), the other with object manipulation (*Object Controller*). Model Predictive Control techniques are used in both blocks, but mainly for manipulation, where multiple ones work in cascade, adopting a pipeline structure.
The controller's essential function is as follows: the motions of the object signal the direction in which the bases must move, and the positions of the bases are feedback that govern the position of the object within the formation. The benefit of this construction is the possibility to easily replace the object controller with a Human operator who can accomplish the same job, moving the object to control the entire formation.
In this sense, the displacement of the object within the formation is the implicit

chosen method that the operator can use to communicate his intentions to the robots.
The controller's feasibility is validated in a real-world implementation through using ROS environment and mobile manipulators from Hebi robotics. Initially, each component of the controller was tested separately to assess its behavior. The results of the experiments demonstrated that all part performs satisfactorily. Following that, the entire controller is tested, exhibiting the predicted behavior, basically allowing the formation to achieve any goal position and orientation, while effectively preventing the object from falling. Moreover another test validated the controller's capacity to avoid collisions with obstacles whose positions were unknown in advance.
In the last experiment, a human operator demonstrates complete control of the formation, allowing him or her to maneuver it to any desired place.

6.1 Future Work

Additional avenues of future research include the possibility of developing an algorithm based on a decentralized control system. In this scenario, the robots cannot have complete information and must rely on local estimates.
An alternative approach would be to increase the amount of information available at the start about the environment in order to perform path planning based on sample methods, such as RRT.
In addition, future implementations could include the ability to manipulate heavier objects while operating close to arm configuration limits.
The final aspect that could be improved is collision avoidance. To increase safety between the robots, an obstacle avoidance term could be included in the minimization problem inside each base controller.
In this way, a less rigid geometry can be achieved, allowing bottlenecks to be overcome and obstacle collisions to be avoided without relying on the global navigation controller for the entire formation.

Bibliography

[1] Antonelli, Gianluca, Arrichiello, Filippo, Caccavale, Fabrizio, and Marino, Alessandro.
"A Decentralized Controller-Observer Scheme for Multi-Agent Weighted Centroid Tracking". In: *IEEE Transactions on Automatic Control* 58.5 (2013), pp. 1310–1316. doi: 10.1109/TAC.2012.2220032.

[2] Butcher, J.C. *B-Series: Algebraic Analysis of Numerical Methods*. Springer Series in Computational Mathematics. Springer International Publishing, 2021. isbn: 9783030709556. url: https://books.google.it/books?id=3aM3zgEACAAJ.

[3] Cervantes, Ilse and Alvarez-Ramirez, Jose. "On the PID tracking control of robot manipulators". In: *Systems & Control Letters* 42.1 (2001), pp. 37–46. issn: 0167-6911. doi: https://doi.org/10.1016/S0167-6911(00)00077-3. url: https://www.sciencedirect.com/science/article/pii/S0167691100000773#PROP2.

[4] Cheng, Cheng, Yu, Xin-Yi, Ou, Lin-Lin, and Guo, Yong-Kui. "Research on multi-robot collaborative transportation control system". In: *2016 Chinese Control and Decision Conference (CCDC)*. 2016, pp. 4886–4891. doi: 10.1109/CCDC.2016.7531868.

[5] Diegel, Olaf, Badve, Aparna, Bright, Glen, Potgieter, Johan, and Tlale, Sylvester. "Improved mecanum wheel design for omni-directional robots". In: *Proc. 2002 Australasian conference on robotics and automation, Auckland*. 2002, pp. 117–121.

[6] Fragapane, Giuseppe, de Koster, René, Sgarbossa, Fabio, and Strandhagen, Jan Ola. “Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda”. In: *European Journal of Operational Research* 294.2 (2021), pp. 405–426. issn: 0377-2217. doi: `https://doi.org/10.1016/j.ejor.2021.01.019`. url: `https://www.sciencedirect.com/science/article/pii/S0377221721000217`.

[7] Franklin, G.F., Powell, J.D., and Emami-Naeini, A. *Feedback Control of Dynamic Systems*. What’s New in Engineering. Pearson, 2019. isbn: 9780134685717. url: `https://books.google.it/books?id=f3DatAEACAAJ`.

[8] HEBI Robotics, Inc. Pittsburgh. *6-DOF Arm Kit- Assembly Instructions*. 2021. url: `https://docs.hebi.us/resources/kits/assyInstructions/A-2085-06_Manual.pdf`.

[9] HEBI Robotics, Inc. Pittsburgh. *Hebi robotics: Documentation*. url: `https://docs.hebi.us/hardware.html#x-series-actuators`.

[10] Jalili, Sadegh, Rezaie, Behrooz, and Rahmani, Zahra. “A novel hybrid model predictive control design with application to a quadrotor helicopter”. In: *Optimal Control Applications and Methods* 39.4 (2018), pp. 1301–1322. doi: `https://doi.org/10.1002/oca.2411`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/oca.2411`. url: `https://onlinelibrary.wiley.com/doi/abs/10.1002/oca.2411`.

[11] Klingspor, Volker, Demiris, John, and Kaiser, Michael. “Human-robot communication and machine learning”. In: *Applied Artificial Intelligence* 11.7 (1997), pp. 719–746.

[12] Koung, Daravuth, Kermorgant, Olivier, Fantoni, Isabelle, and Belouaer, Lamia. “Cooperative Multi-Robot Object Transportation System Based on Hierarchical Quadratic Programming”. In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 6466–6472. doi: `10.1109/LRA.2021.3092305`.

[13] Leitão, Paulo. "Multi-agent systems in industry: current trends & future challenges". In: 2013.

[14] Leusin, Matheus E., Kück, Mirko, Frazzon, Enzo M., Maldonado, Mauricio U., and Freitag, Michael. "Potential of a Multi-Agent System Approach for Production Control in Smart Factories". In: *IFAC-PapersOnLine* 51.11 (2018). 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018, pp. 1459–1464. issn: 2405-8963. doi: `https://doi.org/10.1016/j.ifacol.2018.08.309`. url: `https://www.sciencedirect.com/science/article/pii/S2405896318314332`.

[15] Liu, Ming and Quach, Nghe. "Estimation and Compensation of Gravity and Friction Forces for Robot Arms: Theory and Experiments". In: *Journal of Intelligent and Robotic Systems* 31 (Aug. 2001), pp. 339–354. doi: `10.1023/A:1012089607929`.

[16] Marino, Alessandro. "Distributed adaptive control of networked cooperative mobile manipulators". In: *IEEE Transactions on Control Systems Technology* 26.5 (2017), pp. 1646–1660.

[17] Morari, Manfred and Lee, Jay H. "Model predictive control: past, present and future". In: *Computers & Chemical Engineering* 23.4-5 (1999), pp. 667–682.

[18] Murray, Richard M, Li, Zexiang, and Sastry, S Shankar. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[19] Neumann, Michael A, Chin, Matthew H, and Kitts, Christopher A. "Object manipulation through explicit force control using cooperative mobile multi-robot systems". In: *Proceedings of the World Congress on Engineering and Computer Science*. Vol. 1. 2014.

[20] Nobile, C. *Robotics business review perspectives 2013: Outlook for next-gen new-gen industrial co-worker robotics*. Dec. 2017. url: `https://www.roboticsbusinessreview.com/manufacturing/outlook_for_next_gen_new_gen_industrial_co_worker_robotics/..`

[21] Park, Min and Lee, Min Cheol. “A new technique to escape local minimum in artificial potential field based path planning”. In: *Journal of Mechanical Science and Technology* 17 (Dec. 2003), pp. 1876–1885. doi: `10.1007/BF02982426`.

[22] Pereira, Guilherme AS, Pimentel, Bruno S, Chaimowicz, Luiz, and Campos, Mário FM. “Coordination of multiple mobile robots in an object carrying task using implicit communication”. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*. Vol. 1. IEEE. 2002, pp. 281–286.

[23] Robert, John. *Hands-On Introduction to Robot Operating System(ROS) | trojrobert*. July 2020. url: `https://trojrobert.github.io/hands-on-introdution-to-robot-operating-system%28ros%29/`.

[24] Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2010. isbn: 9781846286414. url: `https://books.google.it/books?id=jPCAFmE-logC`.

[25] Sugar, T. and Kumar, V. “Decentralized control of cooperating mobile manipulators”. In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*. Vol. 4. 1998, 2916–2921 vol.4. doi: `10.1109/ROBOT.1998.680672`.

[26] Tuci, Elio, Alkilabi, Muhanad HM, and Akanyeti, Otar. “Cooperative object transport in multi-robot systems: A review of the state-of-the-art”. In: *Frontiers in Robotics and AI* 5 (2018), p. 59.

[27] Wada, Masayoshi and Torii, Ryotaro. “Cooperative transportation of a single object by omnidirectional robots using potential method”. In: *2013 16th International Conference on Advanced Robotics (ICAR)*. 2013, pp. 1–6. doi: `10.1109/ICAR.2013.6766543`.

[28] Wang, Jia, Wu, Xiao-bei, and Xu, Zhi-liang. “Decentralized Formation Control and Obstacles Avoidance Based on Potential Field Method”. In:

2006 International Conference on Machine Learning and Cybernetics. 2006, pp. 803–808. doi: 10.1109/ICMLC.2006.258457.

[29] Wang, Zhi-Dong, Takano, Y., Hirata, Y., and Kosuge, K. “A pushing leader based decentralized control method for cooperative object transportation”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 1. 2004, 1035–1040 vol.1. doi: 10.1109/IROS.2004.1389489.

[30] Wang, Zijian and Schwager, Mac. “Kinematic multi-robot manipulation with
no communication using force feedback”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 427–432. doi: 10.1109/ICRA.2016.7487163.

[31] Ying, Zhang and Xu, Li. “Leader-follower formation control and obstacle avoidance of multi-robot based on artificial potential field”. In: *The 27th Chinese Control and Decision Conference (2015 CCDC)*. 2015, pp. 4355–4360. doi: 10.1109/CCDC.2015.7162695.

[32] You, Bo, Liu, Si-Yuan, Zhang, Tian-Yong, Pang, Zhong-Hua, and Li, Xiaohu. “Design of Collaborative Transportation System via Multiple Mobile Manipulators”. In: *2020 39th Chinese Control Conference (CCC)*. 2020, pp. 4765–4770. doi: 10.23919/CCC50068.2020.9189550.