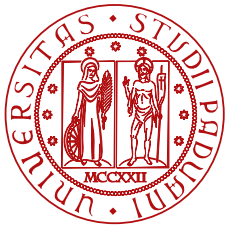

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Department of Information Engineering

Master's Degree in Control Systems Engineering

MPC-based cooperative transportation: towards real-world dynamic environments

Master Candidate

Kovachev Zlatko

Student ID 2129056

Supervisor

Prof. Cenedese Angelo

Co-supervisor

Prof. Michieletto Giulia

Dr. Cigarini Nicola

Academic Year 2024/2025

Graduation date 04/12/2025

To my parents
and friends

Abstract

Building on the work of Professor Cenedese Angelo and colleagues, titled "Model Predictive Control for Cooperative Transportation with Feasibility-Aware Policy," which proposes an MPC-based approach for cooperative navigation of a Multi-Robot System in environments with static and dynamic obstacles, this study adopts their project goal as a foundation to design a new system. In the new framework, omnidirectional robots are replaced by differential-drive robots, and a more sophisticated policy for managing edge cases is introduced. This document explains the motivations behind these changes and describes the development workflow, supported by simulations conducted in MATLAB. It also presents experiments where artificial noise is injected into the system, and concludes by outlining the groundwork for a future real-time implementation on physical robotic platforms.

Contents

List of Figures xi

List of Tables xiii

List of Acronyms xv

1 Introduction 1
- 1.1 Literature review . . . 1
  - 1.1.1 Multi-Robot Systems . . . 2
  - 1.1.2 Model Predictive Control . . . 3
- 1.2 Project premise . . . 4
- 1.3 Thesis outline . . . 4
- 1.4 Notation . . . 5

2 State of the art 7
- 2.1 Problem formulation . . . 8
  - 2.1.1 Robot model . . . 9
- 2.2 MPC approach . . . 9
  - 2.2.1 Leader objective function . . . 10
  - 2.2.2 Follower objective function . . . 11
- 2.3 Constraints . . . 12
  - 2.3.1 Obstacle avoidance . . . 12
  - 2.3.2 Constraint sets definition . . . 14
- 2.4 Feasibility-Aware Recovery Policy . . . 15
- 2.5 Project overview . . . 16

3 MPC approach with unicycle-model based robots 17
- 3.1 Robot definition . . . 17
  - 3.1.1 Load model . . . 19

3.1.2 Velocities constraints 19
3.2 MPC approach 20
3.2.1 Leader objective function 21
3.2.2 Follower objective function 23
3.3 Obstacle avoidance 23
3.3.1 Awareness range 24
3.4 Navigation policy 25
3.4.1 Activation criteria 25
3.4.2 Direction selection and goal generation 27
3.4.3 Reset zone 28
3.4.4 Policy cascade 29
3.4.5 Final thoughts 29
3.5 Virtual environments 30
3.5.1 Edge cases 34
3.6 Perturbation of the system: noise 38
3.6.1 Sensing noise 38
3.6.2 Actuation noise 39

4 Data assessments and validation **41**
4.1 Optimization data 41
4.2 Tests environments 43
4.3 Numerical results 45
4.3.1 Leader cost W 48
4.3.2 Leader cost Z 49
4.3.3 Leader cost R_l and R_a 50
4.3.4 Leader cost K 54
4.3.5 Follower cost C and β 56
4.3.6 Follower cost Q 59

5 Results and discussion **61**
5.1 MATLAB simulations 61
5.2 Comparison with the state of the art 67
5.3 Edge cases results 71
5.4 System robustness with noise 76

6 Conclusions **81**

Appendix **83**

Bibliography **87**

List of Figures

2.1 Example of safe area generation. 13

3.1 TurtleBot4 in two different configurations. 18
3.2 Load structure and vertices. 19
3.3 Awarness range example. 24
3.4 Example of a prediction collapse. 26
3.5 Trajectory simulation for both directions. 27
3.6 Visual representation of a reset zone. 28
3.7 Map (a) - No obstacles. 30
3.8 Map (b) - One obstacle. 31
3.9 Map (c) - Pathological case. 31
3.10 Map (d) - Two obstacles. 32
3.11 (e) - Three obstacles. 32
3.12 Map (f) - Six obstacles. 33
3.13 Map (g) - Valzer. 33
3.14 Map (h) - Moving obstacles. 34
3.15 Map (i) - Obstacle in front of the goal. 35
3.16 Map (ii) - Wall in front of the goal. 35
3.17 Map (iii) - Sharp bend. 36
3.18 Map (iv) - Goal boxed. 36
3.19 Map (v) - Goal boxed and left obstacle. 37
3.20 Map (vi) - Goal boxed and right obstacle. 37
3.21 Map (vii) - Navigation environment. 38

4.1 Map (α) - No obstacles. 43
4.2 Map (β) - Two obstacles. 43
4.3 Map (γ) - Direct wall. 44
4.4 Map (δ) - Goal blocked. 44

4.5 Base test on map ($\boldsymbol{\alpha}$). 46
4.6 Base test on map ($\boldsymbol{\beta}$). 46
4.7 Base test on map ($\boldsymbol{\gamma}$). 47
4.8 Base test on map ($\boldsymbol{\delta}$). 47
4.9 W test on map ($\boldsymbol{\delta}$). 49
4.10 R test on map ($\boldsymbol{\delta}$). $\mathbf{R_l} = 1$ and $\mathbf{R_a} = 0.1$. 53
4.11 R test on map ($\boldsymbol{\delta}$). $\mathbf{R_l} = 10$ and $\mathbf{R_a} = 5$. 54
4.12 C/β test on map ($\boldsymbol{\delta}$). $\mathbf{C} = 1000$ and $\beta = 0.9$. 58

5.1 Simulation of map (a). 62
5.2 Simulation of map (b). 63
5.3 Simulation of map (c). 64
5.4 Simulation of map (d). 64
5.5 Simulation of map (e). 65
5.6 Simulation of map (f). 66
5.7 Simulation of map (g). 66
5.8 Trajectory of the system in the map **(g)**. Snapshots are taken at $t = 30s$, $t = 68s$, $t = 102.5s$ and $t = 112s$. 67
5.9 Simulation of map (h). 68
5.10 Trajectory of the system in the map **(h)**. Snapshots are taken at $t = 14.5s$, $t = 21.5s$, $t = 26.5s$ and $t = 33s$. 68
5.11 Trajectory and formation error evolution of the system defined in [8]. 69
5.12 Trajectory of the system [8] in the Valzer map. Snapshots are taken at $t = 8s$, $t = 13.5s$, $t = 22.5s$ and $t = 30s$. 70
5.13 Formation error evolution of the system [8] in the Valzer map. . . 71
5.14 Simulation of map (i). 72
5.15 Simulation of map (ii). 73
5.16 Simulation of map (iii). 73
5.17 Simulation of map (iv). 74
5.18 Simulation of map (v). 75
5.19 Simulation of map (vi). 75
5.20 Simulation of map (vii). 76
5.21 Simulation with noise of map (a). 77
5.22 Simulation with noise of map (d). 78
5.23 Simulation with noise of map (iv). 78

5.24 Simulation with noise of map (i). 79
5.25 Simulation with noise of map (γ). 80

List of Tables

3.1 Velocities table 20

4.1 Gain values before optimization 42
4.2 Base results for optimization 45
4.3 Optimization testing of gain W 48
4.4 Optimization testing of gain Z 50
4.5 Optimization testing of R_l and R_a 51
4.6 Optimization testing of gain K 55
4.7 Optimization results for different values of C and β 57
4.8 Optimization results for different values of Q 59
4.9 Gain values after optimization 60

5.1 Simulation results on the standard environments 62
5.2 Simulation results on the edge cases environments 71
5.3 Simulation results with noise. 77

List of Acronyms

MRS Multi-Robot Systems

UAV Unmanned Aerial Vehicle

UGV Unmanned Groung Vehicle

AGV Automated Guided Vehicle

MPC Model Predictive Control

Chapter 1

Introduction

1.1 Literature review

In cooperative robotics, multiple robots work together to address a common task or objective, sharing information throughout the process to identify the most effective solution. Consequently, the exchange of information and sensor data is crucial [6]. Cooperative MRS has become an increasingly active research area because of its growing relevance in real-world applications such as logistics, search and rescue, and industrial operations [1].

This work concentrates on the cooperative transport of a heavy payload by two robots, which must compute and follow an optimal path to the goal while avoiding collisions with obstacles and preventing damage to the load [15]. To perform this task efficiently, the robots must preserve a fixed relative distance and, for instance, minimize the formation error, defined as the deviation between the desired and actual separation [20]. The trajectory planning must consider multiple variables for each robot and is influenced by the MRS architecture, its hierarchy, and the chosen control strategy [17][23].

The aim of this project is to build on the work presented in [8], by employing a different type of robot and enhancing the robustness and performance of the MRS through the introduction of new methods. These improvements are intended to be applicable in real scenarios where using a single robot would make transport challenging due to excessive payload weight or where the environment is too complex to navigate easily [6][15].

1.1.1 Multi-Robot Systems

A Multi-Robot System (MRS) is composed of two or more robots that must coordinate and collaborate to achieve a common goal, typically operating within the same environment [22]. By pooling the capabilities of several robots—each with potentially limited individual skills—an MRS can improve overall system performance, enabling interaction, communication, and the distribution and sharing of tasks among all robots [9]. In comparison with single-robot systems, MRSs provide advantages such as higher robustness, parallel task execution, and greater efficiency, making them particularly suitable for applications that demand redundancy and scalability [12]. For instance, robots in an MRS can exchange sensor readings or internal state information to gain a better understanding of their surroundings and to support sensor based control decisions [11].

MRSs can be realized through different architectures, but they are commonly categorized as either homogeneous, where all robots have the same design and capabilities, or heterogeneous, where robots differ in type and functionality so that complementary strengths can be leveraged [2]. Several types of MRSs can be designed [16]:

- **Swarms**: large groups of simple robots whose local interactions lead to emergent collective behavior.
- **Cooperative teams**: smaller groups of robots with more advanced capabilities.
- **Centralized systems**: a single global controller determines the actions of all robots.
- **Decentralized systems**: each robot has its own controller and makes local decisions.
- **Hybrid systems**: integrate global coordination mechanisms with individual robot autonomy.

In aerial scenarios, Unmanned Aerial Vehicles (UAVs) are typically employed, whereas Unmanned Ground Vehicles (UGVs) are adopted for ground-based operations [10]. A specific class of UGV is the Automated Guided Vehicle (AGV), which is tailored for material transport in structured, predictable environments. AGVs follow predefined routes or paths computed in real time by means of

laser guidance, LiDAR, or vision-based systems [19]. Navigation is further supported by sensors for obstacle detection [7], making AGVs particularly suited for repetitive tasks in known settings. In industrial contexts, they are extensively used in warehouses for moving raw materials such as paper, steel, and plastic; in manufacturing plants to feed and support assembly lines; and in logistics operations for container handling at ports, providing benefits such as reduced labor costs, higher throughput, and fewer human errors in sectors including automotive, food processing, and pharmaceuticals [3]. Their adoption also spans hospitals for internal goods distribution and agriculture for field logistics, highlighting their broad role in automating routine transport activities [13].

1.1.2 Model Predictive Control

Model Predictive Control (MPC) is an advanced control strategy that computes a sequence of control actions by minimizing a cost function over a finite prediction horizon, while continuously enforcing constraints on states, inputs, and outputs [14]. For AGVs, MPC is applied to generate trajectories, track reference paths, and support navigation in dynamic settings, enabling robots to accomplish sophisticated tasks such as obstacle avoidance, formation maneuvers, and energy-aware motion [5]. In cooperative scenarios, where several AGVs must move in a coordinated way to transport loads, avoid collisions, and follow a desired route, MPC ensures robust behavior by repeatedly solving an optimization problem at every sampling step and applying only the first control input of the optimal sequence [24].

MPC can be formulated in either linear or nonlinear form, allowing the selection of the most appropriate approach for a given application. Linear MPC relies on linearized vehicle models, which reduce computational effort and are well suited for real-time control when system behavior can be adequately represented without complex constraints or nonlinearities. For instance, Pengfei et al. [18] use a linear MPC framework for trajectory tracking in unmanned ground vehicles, separating motion control from path planning to enhance stability. Another work employs stochastic max-plus linear models combined with chance-constrained MPC for AGV scheduling, efficiently managing uncertainty in traffic networks [21].

Nonlinear MPC, on the other hand, is based on fully nonlinear dynamic models and therefore achieves higher fidelity for realistic systems, at the cost of greater computational demand. For AGVs operating in complex settings, nonlinear MPC improves trajectory tracking accuracy by accounting for kinematic nonlinearities and interactions with the environment [5]. It has also been applied to path tracking of wheeled mobile robots in indoor scenarios, providing accurate control performance under nonholonomic constraints [4].

1.2 Project premise

Building on the work of Elaamery et al. [8], this project focuses on designing an MRS composed of two robots in a Leader–Follower setup to cooperatively transport a heavy load through an environment containing both static and moving obstacles. The *Leader* is an UGV whose primary objective is to reach a target position while avoiding obstacles along its path, whereas the *Follower* is tasked with preserving a constant distance from the *Leader*, preventing collisions with obstacles, and additionally ensuring that the transported load is not damaged. The load is mounted on the *Follower* and linked to the *Leader* through a mechanism that permits translational motion. The project employs a nonlinear MPC to determine the optimal control inputs and trajectory that enable the MRS to achieve its goal; each robot runs its own MPC, relying on both its own measurements and information received from the other robot, thereby retaining a global understanding of the formation at all times. Finally, artificial noise was added to evaluate both the model's robustness and the practicality of real-time deployment on physical robots.

1.3 Thesis outline

In Chapter 2, we introduce the project by Elaamery et al. [8] and examine its functioning along with the underlying mathematical formulations. These concepts are then applied in Chapter 3 to present our own project, detailing the modifications and enhancements implemented. Chapter 4 describes the parameters used in the MPC and outlines how we determined their optimal values through a series of simulations. Finally, Chapter 5 presents the project results, and Chapter 6 concludes with a discussion of potential future developments.

1.4 Notation

Throughout this work, state vectors are represented by $\mathbf{q}$ and input vectors by $\mathbf{u}$. Vectors in $\mathbb{R}^n$ and matrices in $\mathbb{R}^{n \times n}$ are denoted using bold lowercase and bold uppercase letters, respectively. The MPC prediction horizon is always written as N. The system goal is given by $\mathbf{g}_f \in \mathbb{R}^3$, which typically corresponds to the map origin with zero orientation; when this goal is repeated N times, we write $\bar{\mathbf{g}}_f \in \mathbb{R}^{3N \times 1}$. Finally, $\otimes$ denotes the Kronecker product.

Chapter 2

State of the art

For the implementation of this project, the work by Elaamery et al. [8] has been adopted as the state of the art. Its primary goal is to design a control strategy for cooperative transport using an MRS. The study concentrates on two AGVs equipped with manipulators that must transport a shared load through environments populated with obstacles. The overarching objective is to guarantee that the system reaches the desired target while avoiding both static and moving obstacles, preserving the load's integrity and maintaining the formation.

To this end, a Leader–Follower architecture was employed, managed through a decentralized Model Predictive Control (MPC) framework in which each robot independently computes its trajectory. The *Leader* determines the path to the target while avoiding obstacles, whereas the *Follower* plans its own trajectory to keep a prescribed distance from the *Leader* and protect the load from collisions.

The distinctive feature of this project is the introduction of a recovery communication mechanism, termed the Feasibility-Aware Policy. The *Follower* can predict when the *Leader*'s planned trajectory is likely to become "infeasible" and signal this to the *Leader*. In response, the *Leader* temporarily abandons its usual optimal strategy and switches to a more conservative "recovery policy" (typically by reducing speed or altering the path), enabling the *Follower* to realign and safely maintain the formation.

The thesis project builds directly on the assumptions of this paper and in Chapter 3, we will transition from omnidirectional AGVs to AGVs modeled as unicycles.

2.1 Problem formulation

The system is composed of two agents, referred to as the *Leader* and the *Follower*. The problem is formulated under the assumption that both robots make compliant contact with the transported object [8]. For the decentralized control strategy introduced above to be applicable, the following fundamental assumptions are adopted:

- All measurements, state variables and obstacle locations are expressed with respect to the pose of the robot's End-Effector (EE).
- The low-level control layer is assumed to exert full control authority over both the manipulator and the mobile base, such that the EE dynamics can be approximated by a double integrator model.
- The load is transported at a fixed height; hence, the motion planning problem is restricted to the 2D plane.
- The robots can reliably estimate their own pose and communicate data over an ideal communication channel.

Let $\mathbf{x}_{\mathrm{L}}(t) \in \mathbb{R}^2$ and $\mathbf{x}_{\mathrm{F}}(t) \in \mathbb{R}^2$ denote the planar positions of the *Leader* and *Follower* EEs, respectively. The MPC computes the reference trajectories $\mathbf{x}_{\mathrm{L},d}(t)$ and $\mathbf{x}_{\mathrm{F},d}(t)$ that the two robots are required to track. In formal terms, the objective is to determine $\mathbf{x}_{\mathrm{L},d}(t)$ and $\mathbf{x}_{\mathrm{F},d}(t)$ such that:

$$\lim_{t\to\infty} \mathbf{x}_{\mathrm{L},d}(t) = \mathbf{x}^*_{\mathrm{L},d} \tag{2.1}$$

$$\left\|\mathbf{x}_{\mathrm{F},d}(t) - \mathbf{x}_{\mathrm{L}}(t)\right\| = d_{\mathrm{L,F}} \tag{2.2}$$

$$\left\|\mathbf{v}_{\bullet,i}(t) - \mathbf{o}_j(t)\right\| > r_j \qquad \bullet = L, F, \ell,\ i = 1...N_{\mathrm{V}},\ j = 1...M \tag{2.3}$$

In equation (2.2), $d_{\mathrm{L,F}}$ denotes the safety distance imposed by the load structure, ensuring that an appropriate grasp on the object is preserved at all times. Equation (2.3) encodes the obstacle avoidance requirement, which will be discussed in greater detail in Section 2.3.1.

2.1.1 Robot model

The robots are modeled as double integrators, which enables us to express the kinematics in state-space form using linear equations. The resulting equations of motion relate the input acceleration $\mathbf{u}(t)$ to the position as follows:

$$\ddot{\mathbf{x}}_{\mathrm{L}}(t) = \mathbf{u}_{\mathrm{L}}(t) \quad \text{and} \quad \ddot{\mathbf{x}}_{\mathrm{F}}(t) = \mathbf{u}_{\mathrm{F}}(t)$$

We define the full state vector $\mathbf{q}(t)$ as the concatenation of position and velocity:

$$\mathbf{q}_{\mathrm{L}}(t) := \begin{bmatrix} \mathbf{x}_{\mathrm{L}}(t) \\ \mathbf{v}_{\mathrm{L}}(t) \end{bmatrix} \in \mathbb{R}^4 \quad \text{with} \quad \mathbf{q}_{\mathrm{F}}(t) := \begin{bmatrix} \mathbf{x}_{\mathrm{F}}(t) \\ \mathbf{v}_{\mathrm{F}}(t) \end{bmatrix} \in \mathbb{R}^4$$

The system's evolution in continuous-time state space can be expressed as:

$$\dot{\mathbf{q}}(t) = \mathbf{A}_{\mathrm{c}}\mathbf{q}(t) + \mathbf{B}_{\mathrm{c}}\mathbf{u}(t), \quad \text{with} \quad \mathbf{A}_{\mathrm{c}} = \begin{bmatrix} \mathbf{0}_{2\times2} & \mathbf{I}_2 \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \end{bmatrix} \in \mathbb{R}^{4\times4}, \mathbf{B}_{\mathrm{c}} = \begin{bmatrix} \mathbf{0}_{2\times2} \\ \mathbf{I}_2 \end{bmatrix} \in \mathbb{R}^{4\times2}.$$

For implementing the MPC algorithm, the model is discretized using a sampling period of T_{s}, leading to the following discrete-time update law:

$$\mathbf{q}(t+1) = \mathbf{A}\mathbf{q}(t) + \mathbf{B}\mathbf{u}(t), \quad \text{with} \quad \mathbf{A} = e^{\mathbf{A}_{\mathrm{c}}T_{\mathrm{s}}} \in \mathbb{R}^{4\times4}, \mathbf{B} = \int_0^{T_{\mathrm{s}}} e^{\mathbf{A}_{\mathrm{c}}\tau}\mathbf{B}_{\mathrm{c}}d\tau \in \mathbb{R}^{4\times2}$$

2.2 MPC approach

Cooperative trajectory planning is tackled using a Model Predictive Control (MPC) framework. As outlined in Section 1.1.2, this approach seeks the optimal control sequence $\hat{\mathbf{u}}^*(\cdot)$ that minimizes a cost functional J, subject to system dynamics and to state and input constraints:

$$\hat{\mathbf{u}}^*(\cdot) = \underset{\hat{\mathbf{u}}(\cdot)}{\operatorname{argmin}} \quad J(\hat{\mathbf{q}}(\cdot), \hat{\mathbf{u}}(\cdot)) \tag{2.4}$$

$$\text{subject to} \quad \hat{\mathbf{q}}(k+1) = \mathbf{A}\hat{\mathbf{q}}(k) + \mathbf{B}\hat{\mathbf{u}}(k) \qquad \text{(model dynamics)} \tag{2.5}$$

$$\hat{\mathbf{q}}(0) = \mathbf{q}(t) \qquad \text{(initial condition)} \tag{2.6}$$

$$\hat{\mathbf{q}}(k) \in \mathcal{X}(t) \qquad \text{(state constraint)} \tag{2.7}$$

$$\hat{\mathbf{u}}(k) \in \mathcal{U} \qquad \text{(input constraint)} \tag{2.8}$$

where $k = 0, \ldots, N$ indexes the steps over the prediction horizon of length N. The vectors $\hat{\mathbf{q}}(k)$ and $\hat{\mathbf{u}}(k)$ denote the predicted state and input, respectively, while $\mathcal{X}(t)$ and $\mathcal{U}$ define the admissible state and input sets.

Because the MRS is organized in a decentralized fashion, each robot independently solves its own MPC problem with its own objective function at every time instant. Specifically, the optimization is carried out first by the *Leader* and subsequently by the *Follower*, ensuring the correct operation of the overall system.

2.2.1 Leader objective function

The leader's main goal is to reach the desired target position $\mathbf{x}^*_{\mathrm{L},d}$ while keeping the control effort as small as possible. This is formulated as a quadratic cost over the prediction horizon:

$$J_{\mathrm{L}}\left(\hat{\mathbf{q}}_{\mathrm{L}}(\cdot), \hat{\mathbf{u}}_{\mathrm{L}}(\cdot)\right) := \sum_{k=0}^{N-1} \left(\hat{\mathbf{q}}_{\mathrm{L}}^{\top}(k)\mathbf{W}\hat{\mathbf{q}}_{\mathrm{L}}(k) + \hat{\mathbf{u}}_{\mathrm{L}}^{\top}(k)\mathbf{R}\hat{\mathbf{u}}_{\mathrm{L}}(k)\right) + \hat{\mathbf{q}}_{\mathrm{L}}^{\top}(N)\mathbf{Z}\hat{\mathbf{q}}_{\mathrm{L}}(N) \tag{2.9}$$

This cost function consists of three main parts:

- The term $\hat{\mathbf{q}}_{\mathrm{L}}^{\top}(k)\mathbf{W}\hat{\mathbf{q}}_{\mathrm{L}}(k)$ penalizes the deviation of the state from the target, with weights given by $\mathbf{W} \in \mathbb{R}^{4\times4}$. This encourages the *Leader* to reach the target position placed at the origin.
- The term $\hat{\mathbf{u}}_{\mathrm{L}}^{\top}(k)\mathbf{R}\hat{\mathbf{u}}_{\mathrm{L}}(k)$ penalizes the control effort (i.e., energy usage), with weighting matrix $\mathbf{R} \in \mathbb{R}^{2\times2}$.
- The terminal term $\hat{\mathbf{q}}_{\mathrm{L}}^{\top}(N)\mathbf{Z}\hat{\mathbf{q}}_{\mathrm{L}}(N)$ defines the final-state penalty, with $\mathbf{Z} \in \mathbb{R}^{4\times4}$, to promote stable convergence.

For efficient numerical solution, the optimization problem is expressed in a compact matrix form. By stacking the N predicted states and inputs into the vectors $\hat{\bar{\mathbf{q}}}_{\mathrm{L}}$ and $\hat{\bar{\mathbf{u}}}_{\mathrm{L}}$, respectively, we obtain:

$$\hat{\bar{\mathbf{q}}}_{\mathrm{L}} := \begin{bmatrix} \hat{\mathbf{q}}_{\mathrm{L}}^{\top}(1) & \dots & \hat{\mathbf{q}}_{\mathrm{L}}^{\top}(N) \end{bmatrix}^{\top} \in \mathbb{R}^{4N}, \quad \hat{\bar{\mathbf{u}}}_{\mathrm{L}} := \begin{bmatrix} \hat{\mathbf{u}}_{\mathrm{L}}^{\top}(0) & \dots & \hat{\mathbf{u}}_{\mathrm{L}}^{\top}(N-1) \end{bmatrix}^{\top} \in \mathbb{R}^{2N},$$
$$\bar{\mathbf{W}} := \begin{bmatrix} \mathbf{I}_{N-1} \otimes \mathbf{W} & \mathbf{0}_{4(N-1)\times 4} \\ \mathbf{0}_{4\times 4(N-1)} & \mathbf{Z} \end{bmatrix} \in \mathbb{R}^{4N\times 4N}, \quad \bar{\mathbf{R}} := \mathbf{I}_N \otimes \mathbf{R} \in \mathbb{R}^{2N\times 2N}. \tag{2.10}$$

For both the *Leader* and the *Follower*, the evolution of the state can be written in terms of the initial state and the input sequence as

$$\hat{\mathbf{q}} = \bar{\mathbf{T}}\hat{\mathbf{q}}(0) + \bar{\mathbf{S}}\hat{\mathbf{u}} \tag{2.11}$$

where the prediction matrices $\bar{\mathbf{T}} \in \mathbb{R}^{4N\times 4}$ and $\bar{\mathbf{S}} \in \mathbb{R}^{4N\times 2N}$ are assembled from the system matrices $\mathbf{A}$ and $\mathbf{B}$ as

$$\bar{\mathbf{T}} = \begin{bmatrix} \mathbf{A} \\ \vdots \\ \mathbf{A}^N \end{bmatrix}, \quad \bar{\mathbf{S}} = \begin{bmatrix} \mathbf{B} & \mathbf{0}_{4\times 2} & \cdots & \mathbf{0}_{4\times 2} \\ \mathbf{AB} & \mathbf{B} & \cdots & \mathbf{0}_{4\times 2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \cdots & \mathbf{B} \end{bmatrix} \tag{2.12}$$

Inserting (2.11) into the cost functional allows us to rewrite the leader's optimization problem as a standard quadratic program in the decision variable $\hat{\mathbf{u}}_{\mathrm{L}}$.

$$J_{\mathrm{L}}\left(\hat{\mathbf{q}}_{\mathrm{L}}(0), \hat{\bar{\mathbf{u}}}_{\mathrm{L}}\right) = \frac{1}{2}\hat{\bar{\mathbf{u}}}_{\mathrm{L}}^{\top}\mathbf{H}\hat{\bar{\mathbf{u}}}_{\mathrm{L}} + \hat{\mathbf{q}}_{\mathrm{L}}^{\top}(0)\mathbf{F}\hat{\bar{\mathbf{u}}}_{\mathrm{L}} + \frac{1}{2}\hat{\mathbf{q}}_{\mathrm{L}}^{\top}\mathbf{Y}\hat{\mathbf{q}}_{\mathrm{L}}(0) \tag{2.13}$$

with:

$$\mathbf{H} = 2\left(\bar{\mathbf{R}} + \bar{\mathbf{S}}^{\top}\bar{\mathbf{W}}\bar{\mathbf{S}}\right) \in \mathbb{R}^{2N\times 2N} \tag{2.14}$$
$$\mathbf{F} = 2\bar{\mathbf{T}}^{\top}\bar{\mathbf{W}}\bar{\mathbf{S}} \in \mathbb{R}^{4\times 2N} \tag{2.15}$$
$$\mathbf{Y} = 2\left(\mathbf{W} + \bar{\mathbf{T}}^{\top}\bar{\mathbf{W}}\bar{\mathbf{T}}\right) \in \mathbb{R}^{4\times 4} \tag{2.16}$$

2.2.2 Follower objective function

The follower must preserve a specific formation distance $d_{\mathrm{L,F}}$ from the leader while simultaneously minimizing unnecessary movements. The formation error

is defined as the deviation between the squared Euclidean distance of the two agents and the desired squared distance:

$$e_{\mathrm{L,F}}(t) := \|\mathbf{x}_{\mathrm{L}}(t) - \mathbf{x}_{\mathrm{F}}(t)\|^2 - d_{\mathrm{L,F}}^2 \tag{2.17}$$

The follower's cost function J_{F} consists of two weighted contributions:

$$J_{\mathrm{F}}\left(\hat{\mathbf{q}}_{\mathrm{L}}(\cdot), \hat{\mathbf{q}}_{\mathrm{F}}(\cdot)\right) = cJ_{\mathrm{F},1}\left(\hat{\mathbf{q}}_{\mathrm{L}}(\cdot), \hat{\mathbf{q}}_{\mathrm{F}}(\cdot)\right) + J_{\mathrm{F},2}\left(\hat{\mathbf{q}}_{\mathrm{L}}(\cdot), \hat{\mathbf{q}}_{\mathrm{F}}(\cdot)\right) \tag{2.18}$$

1. The first component, $J_{\mathrm{F},1}$, penalizes formation error over the prediction horizon. A decay factor $\beta \in (0, 1]$ is introduced to prioritize minimizing the error in the immediate future steps:

$$J_{\mathrm{F},1}\left(\hat{\mathbf{q}}_{\mathrm{L}}(\cdot), \hat{\mathbf{q}}_{\mathrm{F}}(\cdot)\right) := \sum_{k=1}^{N} \beta^k \left(\|\mathbf{P}_{\mathbf{x}}\left(\hat{\mathbf{q}}_{\mathrm{L}}(k) - \hat{\mathbf{q}}_{\mathrm{F}}(k)\right)\|^2 - d_{\mathrm{L,F}}^2\right)^2 \tag{2.19}$$

2. The second component, $J_{\mathrm{F},2}$, penalizes the displacement of the follower between successive steps, thereby encouraging shorter trajectories and smoother motion:

$$J_{\mathrm{F},2}\left(\hat{\mathbf{q}}_{\mathrm{L}}(\cdot), \hat{\mathbf{q}}_{\mathrm{F}}(\cdot)\right) := \sum_{k=0}^{N-1} \|\mathbf{P}_{\mathbf{x}}\left(\hat{\mathbf{q}}_{\mathrm{F}}(k+1) - \hat{\mathbf{q}}_{\mathrm{F}}(k)\right)\|^2 \tag{2.20}$$

where $\mathbf{P}_{\mathbf{x}} := \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_{2\times 1} \end{bmatrix} \in \mathbb{R}^{2\times 3}$ is a selection matrix that extracts the position components from the state vector.

2.3 Constraints

2.3.1 Obstacle avoidance

To prevent collisions with environmental obstacles, all vertices of both the robots and the load must remain within a designated safe region. The obstacles have been modeled as circles with radius r_j and centers $\mathbf{o}_j$. Obstacle avoidance is then formally expressed as

$$\left\|\mathbf{v}_{\bullet,i}(t) - \mathbf{o}_j(t)\right\| > r_j \qquad \bullet = L, F, \ell,\ i = 1...N_{\mathrm{V}},\ j = 1...M \tag{2.21}$$

For the load, the vertex positions are functions of the joint configuration of the leader and followers:

$$\mathbf{v}_{\ell,i}(t) := h_\ell\left(\mathbf{x}_{\mathrm{L}}(t), \mathbf{x}_{\mathrm{F}}(t)\right) \in \mathbb{R}^2, \quad i = 1 \dots N_{\mathrm{v}} \tag{2.22}$$

Because this Euclidean distance constraint yields a non-convex nonlinearity that is not easily addressed in a linear MPC framework, we employ a geometric method based on local linearization. For each obstacle j, we determine the point $\mathbf{w}_j(t)$ on its boundary that is closest to the robot's center.

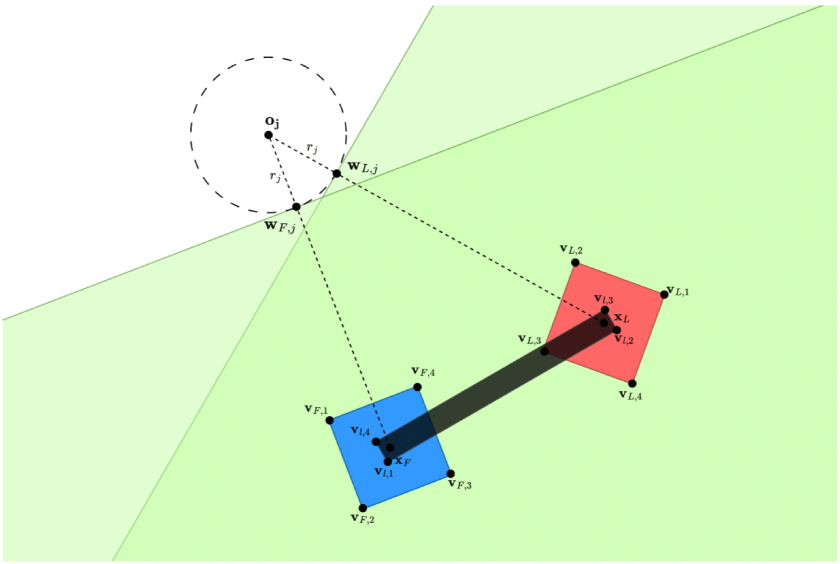

Figure 2.1: Example of safe area generation.

A half-space is constructed by taking the line tangent to the obstacle at $\mathbf{w}_j(t)$ and enforcing orthogonality with respect to the vector that links the robot's center to the obstacle.

The intersection of all such half-spaces, one for each of the M obstacles, yields a convex polygon that specifies the region in which the robots can safely move without collisions over the prediction horizon. Formally, for each vertex i and obstacle j, the safety constraint holds if:

$$\mathbf{g}_{o,j}^{\top}(t)\mathbf{v}_{\bullet,i}(t) \leq b_{o,j}(t) \tag{2.23}$$

where $\mathbf{g}_{o,j}(t)$ and $b_{o,j}(t)$ are the parameters describing the separating hyperplane at time t. Aggregating all obstacles leads to a matrix $\mathbf{G}_o$ and a vector $\mathbf{b}_o$ such that $\mathbf{G}_o(t)\mathbf{v}_{\bullet,i}(t) \leq \mathbf{b}_o(t)$.

2.3.2 Constraint sets definition

Although, from a conceptual standpoint, the constraints on the inputs (accelerations) and on the states (velocities) differ from obstacle-avoidance constraints, in the MPC implementation it is computationally advantageous to collect all inequalities into a single, compact matrix expression.

Using the prediction of the discrete linear model, the future state $\mathbf{x}(k)$ can be written as a function of the initial state $\mathbf{x}(0)$ and of the control input sequence $\mathbf{u}_k$ via the prediction matrices $\bar{\mathbf{T}}$ and $\bar{\mathbf{S}}$, so that

$$\mathbf{x}_h = \bar{\mathbf{T}}\mathbf{x}(0) + \bar{\mathbf{S}}\mathbf{u}_k. \tag{2.24}$$

This allows us to formulate a single linear inequality that incorporates all system constraints:

$$\mathbf{G}\mathbf{u}_k \leq \mathbf{W} + \mathbf{S}\mathbf{x}(0). \tag{2.25}$$

Expanding the matrices $\mathbf{G}$, $\mathbf{W}$, and $\mathbf{S}$ reveals how they encode the different constraint types:

$$\begin{bmatrix} \mathbf{A}_{bar}\bar{\mathbf{S}} \\ \mathbf{G}_{in} \\ \mathbf{A}_{vel}\bar{\mathbf{S}} \end{bmatrix} \mathbf{u}_k \leq \begin{bmatrix} \mathbf{B}_{bar} \\ \mathbf{B}_{in} \\ \mathbf{B}_{vel} \end{bmatrix} + \begin{bmatrix} -(\mathbf{A}_{bar}\bar{\mathbf{T}}) \\ \mathbf{0} \\ -(\mathbf{A}_{vel}\bar{\mathbf{T}}) \end{bmatrix} \mathbf{x}(0). \tag{2.26}$$

The detailed structure of these matrices is reported in Appendix 6.2. Examining each block in the inequality:

1. **Obstacle constraints:** The first block row of (2.26)
$$\mathbf{A}_{\text{bar}}\bar{\mathbf{S}}\mathbf{u}_k \leq \mathbf{B}_{\text{bar}} - \mathbf{A}_{\text{bar}}\bar{\mathbf{T}}\mathbf{x}(0) \tag{2.27}$$
corresponds to the obstacle-avoidance constraints introduced in Section 2.3.1. Substituting the expression for $\mathbf{x}_h$ given by equation (2.24), (2.27) can be rewritten as
$$\mathbf{A}_{\text{bar}}\mathbf{x}_h \leq \mathbf{B}_{\text{bar}}, \tag{2.28}$$
where $\mathbf{A}_{bar}\bar{\mathbf{S}}$ maps the system dynamics into the geometric constraints, and $\mathbf{B}_{bar}$ gathers the known terms defining the half-spaces. This component guarantees that the predicted trajectory stays inside the safe area.

2. **Input constraints:** The second block row of (2.26)
$$\mathbf{G}_{in}\mathbf{u}_k \leq \mathbf{B}_{in} \tag{2.29}$$
enforces bounds on the control inputs, avoiding actuator saturation by restricting the robot's admissible accelerations:
$$\mathbf{u}_{min} \leq \mathbf{u}(k) \leq \mathbf{u}_{max}.$$

3. **State constraints:** The last block row of (2.26)

$$\mathbf{A}_{\text{vel}}\bar{\mathbf{S}}\mathbf{u}_k \leq \mathbf{B}_{\text{vel}} - \mathbf{A}_{\text{vel}}\bar{\mathbf{T}}\mathbf{x}(0) \tag{2.30}$$

imposes bounds on the state variables, in particular on the robot's linear velocities, ensuring they remain within safe operating limits:

$$\mathbf{v}_{min} \leq \mathbf{v}(k) \leq \mathbf{v}_{max}.$$

2.4 Feasibility-Aware Recovery Policy

The decentralized strategy, while effective, carries the risk that the leader's optimal trajectory may be infeasible for the follower or may induce a formation error large enough to jeopardize the integrity of the load. This situation commonly arises in sharp turns or highly cluttered environments, where the follower's kinematic and environmental constraints become particularly restrictive.

To address this issue, a feasibility-aware recovery mechanism ("Recovery Policy") is introduced. The follower continuously predicts its own future state and, if it anticipates that the formation error will exceed a safety threshold ε within a specified prediction horizon, it signals this critical condition to the leader.

Upon receiving this alert, the leader switches to a conservative behavior aimed at restoring the formation. Concretely, the leader replaces its nominal, target-oriented cost function (J_{L}) with a recovery cost ($J_{\text{L,rec}}$). This alternative objective function focuses on minimizing the displacement between consecutive steps in the prediction horizon, effectively causing the leader to significantly slow down or halt:

$$J_{\text{L,rec}}\left(\hat{\mathbf{q}}_{\text{L}}(\cdot)\right) := \sum_{k=0}^{N-1} \left\|\mathbf{P}_{\mathbf{x}}\left(\hat{\mathbf{q}}_{\text{L}}(k+1) - \hat{\mathbf{q}}_{\text{L}}(k)\right)\right\|^2 = \sum_{k=0}^{N-1} \left\|\hat{\mathbf{x}}_{\text{L}}(k+1) - \hat{\mathbf{x}}_{\text{L}}(k)\right\|^2 \tag{2.31}$$

Implementing this policy requires re-running the MPC optimization within the same sampling interval. If infeasibility is detected, the leader discards the previously computed trajectory, replaces the cost with $J_{\text{L,rec}}$, and resolves the optimization problem. In this way, the system temporarily trades rapid convergence to the goal for enhanced stability and safe transport.

2.5 Project overview

The project described in [8] has achieved excellent numerical performance and has a good balance between detachment error and the speed of the two robots. Its implementation is easy because the robots are modeled as double integrators. In addition, the proposed Feasibility-Aware recovery policy efficiently reduces the overall error and helps to prevent load detachment.

Because the robots use omnidirectional wheels, changing direction is straightforward; however, this also introduces robustness challenges. When the *Leader* alters its direction too frequently, the *Follower* has difficulty keeping the desired distance, causing the recovery policy to be activated repeatedly and increasing the overall computation time. Moreover, the safe area for obstacle avoidance is computed only once at the beginning of each MPC cycle and remains fixed until the next iteration. This may compromise prediction accuracy, particularly in environments with moving obstacles.

Overall, the system offers both advantages and disadvantages. Nevertheless, the new robot architecture introduced in Chapter 3 is expected to significantly enhance the project, achieving very low tracking error while preserving strong robustness, even under noisy conditions.

Chapter 3

MPC approach with unicycle-model based robots

Building on the work of Elaamery et al. [8], presented in Chapter 2, this thesis aims to reproduce the same capabilities using robots based on the unicycle model. A new system has been designed following a Leader–Follower scheme, where the *Follower* is responsible to prevent the load from colliding with obstacles. The original Feasibility-Aware Recovery Policy has been discarded and replaced by a Navigation Policy, which enables the robots to compute more intelligent trajectories to bypass obstacles that would otherwise block them. This design decision will be discussed in detail in Section 3.4; however, it can already be stated that, after changing the robot model, the recovery policy has been found to be less relevant than in the work of Elaamery et al. [8].

In this chapter, we analyze the chosen robot model, describe how the objective functions for the MPC were adapted and refined, and examine the Navigation Policy and its role in managing particular edge cases. Finally, we outline the virtual test environments used to evaluate the system and highlight the importance of introducing noise to assess its robustness.

3.1 Robot definition

The robots introduced in Section (2.1.1) were modeled as double integrators, which yielded a system governed by linear equations. In the new setup, we instead have adopted differential-drive robots described by the unicycle model, where the state is given by $\mathbf{q}(t) = [x(t)\ \ y(t)\ \ \theta(t)]^T$, encoding the robot's position

and orientation in space. The control input is now defined by the linear and angular velocities used to drive the robot, namely $\mathbf{u}(t) = [v(t) \quad \omega(t)]^T$. The key distinction between these two robot models is that the state evolution can no longer be written in linear state-space form, because the update equations become nonlinear due to the *sin* and *cos* terms appearing in the position update, as shown in

$$\mathbf{q}(t+1) = \begin{cases} x(t) + v(t) \cdot cos(\theta(t)) \cdot T_s \\ y(t) + v(t) \cdot sin(\theta(t)) \cdot T_s \\ \theta(t) + \omega(t) \cdot T_s \end{cases} \tag{3.1}$$

Equation (3.1) is already written in discrete time with sampling period T_s. In the actual Matlab implementation, a *wrapToPi* operation is applied whenever the state component θ is updated, in order to preserve system stability. The following basic assumptions are made to ensure correct system behavior:

- All measurements, states, and obstacle positions are defined with respect to the robots' centers.
- The load is connected to the *Follower* through a revolute joint and rests on the *Leader* on a plane that permits longitudinal motion between the two robots without slipping away from the robot center.
- The obstacles are known a priori, and their locations are specified with respect to the positions of the two robots.

Figure 3.1: TurtleBot4 in two different configurations.

We have adopted the physical parameters of a **TurtleBot4** to specify the dimensions and maximum velocities of both robots, facilitating a potential real-world deployment of the code. The **TurtleBot4** is circular with a diameter of 0.35 m. To use it properly in this project, we have computed the vertices of an octagon circumscribed around the robot footprint, providing adequate coverage and preventing collisions with obstacles. These 8 vertices are then employed in the constraint calculations for obstacle avoidance. The maximum linear velocity of the **TurtleBot4** is 0.32 m/s, and its maximum angular velocity is 2.84 rad/s.

3.1.1 Load model

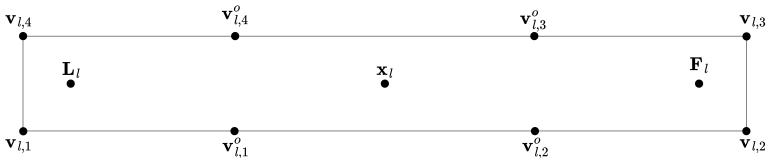

Figure 3.2: Load structure and vertices.

The load is modeled as a rectangle of one meter long (matching the required distance between the two robots) and 0.15 m wide. To ensure it does not collide with obstacles, four additional vertices are introduced $\mathbf{v}^o_{l,i}$, distinct from the physical corners of the load $\mathbf{v}_{l,i}$, since those lie within the region occupied by the two robots. These new vertices instead enable collision avoidance for the central area of the load, both against small obstacles and moving ones. They are used to build the obstacle-avoidance matrices and are combined with those already defined for the *Follower* to perform its MPC. The *Follower* is responsible for managing the load because it receives the *Leader*'s predicted states, allowing it to compute the load's pose at each MPC step and thus maintain a safe clearance from obstacles.

3.1.2 Velocities constraints

To realistically emulate the physical robots, we must constrain the input commands so that the velocities do not exceed the robots mechanical limits. The linear velocity is restricted to be non-negative and to remain below a specified upper bound, which prevents the robots from moving backward:

$$0 \leq v \leq v_{max}$$

In contrast, the angular velocity can assume both positive and negative values within a symmetric interval:

$$-\omega_{max} \leq \omega \leq \omega_{max}$$

These maximum values differ for the two robots:

	$\mathbf{v_{max}}$ (m/s)	$\boldsymbol{\omega_{max}}$ (rad/s)
LEADER	0.2	$\pi \cdot 0.6$
FOLLOWER	0.3	$\pi \cdot 0.9$

Table 3.1: Velocities table

From the table, we observe that the *Follower* can move faster than the *Leader*. This design choice gives the *Follower* a higher responsiveness to the *Leader*'s commands, enabling it to maintain its position and keep the load within the safety area, even in the presence of abrupt speed variations.

3.2 MPC approach

The key point to address is that a nonlinear MPC must be employed, because the state evolution is governed by the nonlinear equations in (3.1). In MATLAB, the nonlinear MPC is implemented using the *fmincon* function, which computes all N optimal control inputs for both robots; the first input in this sequence is then applied to update the robots' states. This function offers three main solvers: *interior-point*, *sqp*, and *active-set*. The **interior-point** method is the default and performs very well on linear problems, which is not suitable here. In contrast, **sqp** (Sequential Quadratic Programming) and **active-set** are better suited to nonlinear optimization. Although **active-set** is generally faster, tests on the system showed that **sqp** yields more stable solutions while still maintaining good computational speed.

The robots' objective functions have been adapted to the new nonlinear model and further refined to enhance both accuracy and efficiency, while continuously ensuring that the separation error between the two robots remained bounded.

3.2.1 Leader objective function

The cost function closely matches the one originally presented in [8]; the only change is the inclusion of an additional cost term, with its own gain **K**. This term provides the *Leader* with some information about the *Follower*'s predicted motion, allowing it to alter its path or reduce speed if the distance to the *Follower* becomes excessively large.

As an initial step, equation (2.9) is adopted instead of the condensed expression (2.13). This is because the construction of the matrices **T** and **S** in the earlier formulation relied on the state matrices **A** and **B** of the previous model. Since the current model is not available in state-space form, we revert to the first equation and adjust its gains to match the new dynamics. In the updated system, the state dimension is 3, whereas previously it was $\mathbf{q} = [x(t)\ y(t)\ v_x(t)\ v_y(t)]^T \in \mathbb{R}^4$. Because the input dimension remains unchanged, we can specify the dimensions of the new gains as follows:

- $\mathbf{W} \in \mathbb{R}^{3\times3}$, weight matrix applied to the state from step 1 to step $N-1$ and to the initial condition of the MPC.
- $\mathbf{Z} \in \mathbb{R}^{3\times3}$, weight matrix for the terminal state at step N.
- $\mathbf{R} \in \mathbb{R}^{2\times2}$, weight matrix for the predicted optimal input.

For the weight matrices W and Z, the element associated with the orientation θ in the state $\mathbf{q}$ is fixed to 0.01, as our objective is to reach the target position without enforcing a specific final heading for the robots. We then reformulate equations (2.10) for the updated system as follows:

$$\hat{\bar{\mathbf{q}}}_{\mathrm{L}} := \begin{bmatrix} \hat{\mathbf{q}}_{\mathrm{L}}^{\top}(1) & \dots & \hat{\mathbf{q}}_{\mathrm{L}}^{\top}(N) \end{bmatrix}^{\top} \in \mathbb{R}^{3N}, \quad \hat{\bar{\mathbf{u}}}_{\mathrm{L}} := \begin{bmatrix} \hat{\mathbf{u}}_{\mathrm{L}}^{\top}(0) & \dots & \hat{\mathbf{u}}_{\mathrm{L}}^{\top}(N-1) \end{bmatrix}^{\top} \in \mathbb{R}^{2N},$$
$$\bar{\mathbf{W}} := \begin{bmatrix} \mathbf{I}_{N-1} \otimes \mathbf{W} & \mathbf{0}_{3(N-1)\times3} \\ \mathbf{0}_{3\times3(N-1)} & \mathbf{Z} \end{bmatrix} \in \mathbb{R}^{3N\times3N}, \quad \bar{\mathbf{R}} := \mathbf{I}_N \otimes \mathbf{R} \in \mathbb{R}^{2N\times2N}. \tag{3.2}$$

Given these premises, the *Leader*'s overall cost function can be expressed as the sum of four distinct components:

1. The first term measures how far the initial state of the system is from the final goal:

$$J_{L,1}(\hat{\mathbf{q}}_L(t)) = (\hat{\mathbf{q}}_L(0) - \mathbf{g_f})^{\top} \mathbf{W} (\hat{\mathbf{q}}_L(0) - \mathbf{g_f}) \tag{3.3}$$

2. The second term penalizes the deviation of the N predicted states from the final goal:

$$J_{L,2}(\hat{\tilde{\mathbf{q}}}_L(t)) = \left(\hat{\tilde{\mathbf{q}}}_L - \bar{\mathbf{g}}_f\right)^\top \bar{\mathbf{W}} \left(\hat{\tilde{\mathbf{q}}}_L - \bar{\mathbf{g}}_f\right) \tag{3.4}$$

The distinctive feature of the weight matrix $\bar{\mathbf{W}}$ is that it embeds both the gain $\mathbf{W}$ for prediction steps 1 to $N-1$, and the gain $\mathbf{Z}$ for the terminal prediction step N.

3. The third term penalizes the control inputs computed by the MPC across the prediction horizon N:

$$J_{L,3}(\hat{\tilde{\mathbf{u}}}_L(t)) = \hat{\tilde{\mathbf{u}}}_L^\top \mathbf{R} \hat{\tilde{\mathbf{u}}}_L \tag{3.5}$$

4. The fourth term enables the *Leader* to incorporate information about the *Follower*'s predicted position, allowing it to plan its motion accordingly:

$$\begin{aligned} J_{L,4}(\hat{\tilde{\mathbf{q}}}_L(t), \hat{\tilde{\mathbf{q}}}_F(t-1)) =& \mathbf{K} \cdot \boldsymbol{\beta}_{N-1} \cdot \\ & \cdot \left(\mathbf{P_x} \|\hat{\tilde{\mathbf{q}}}_L(1:N-1) - \hat{\tilde{\mathbf{q}}}_F(2:N)\|^2 - d_{L,F}^2\right)^2 \end{aligned} \tag{3.6}$$

Here, the scalar gain $\mathbf{K} \in \mathbb{R}$ tunes the penalty on the distance error between the two robots, while $\boldsymbol{\beta}_{N-1} \in \mathbb{R}^{1\times N-1}$ is a vector of decay factors that emphasizes near-term predictions over those further in time:

$$\boldsymbol{\beta}_{N-1} = \begin{bmatrix} \beta & \beta^2 & \dots & \beta^{N-1} \end{bmatrix} \in \mathbb{R}^{1\times N-1} \qquad \text{with} \qquad \beta \in (0,1]$$

The remaining part of the expression computes the distance between the robots, offset by the load distance $d_{L,F}$, thus yielding the distance error. Note that the indexing selects only $N-1$ predicted states and over selected intervals. This arises because $\hat{\tilde{\mathbf{q}}}_F(t-1)$ corresponds to the prediction sequence from the iteration immediately preceding the one the MPC is currently solving. To synchronize the robots' predicted poses, we discard the first state (associated with time t) from the *Follower*'s sequence and omit the last state from $\hat{\tilde{\mathbf{q}}}_L(t)$. In this way, the two prediction sequences are temporally aligned, and the cost can be evaluated consistently.

Combining all these contributions yields the final expression of the *Leader*'s objective function:

$$\begin{aligned} J_L(\hat{\mathbf{q}}_L(t), \hat{\tilde{\mathbf{q}}}_L(t), \hat{\tilde{\mathbf{u}}}_L(t), \hat{\tilde{\mathbf{q}}}_F(t-1)) =& \left(\hat{\mathbf{q}}_L(0) - \mathbf{g_f}\right)^\top \mathbf{W} \left(\hat{\mathbf{q}}_L(0) - \mathbf{g_f}\right) + \\ &+ \left(\hat{\tilde{\mathbf{q}}}_L - \bar{\mathbf{g}}_f\right)^\top \bar{\mathbf{W}} \left(\hat{\tilde{\mathbf{q}}}_L - \bar{\mathbf{g}}_f\right) + \\ &+ \hat{\tilde{\mathbf{u}}}_L^\top \mathbf{R} \hat{\tilde{\mathbf{u}}}_L + \\ &+ \mathbf{K} \cdot \boldsymbol{\beta}_{N-1} \cdot \left(\mathbf{P_x} \|\hat{\tilde{\mathbf{q}}}_L(1:N-1) - \hat{\tilde{\mathbf{q}}}_F(2:N)\|^2 - d_{L,F}^2\right)^2 \end{aligned} \tag{3.7}$$

3.2.2 Follower objective function

For the *Follower*, the objective function remains largely analogous to that of the original system; only the equations structure have been adjusted to enable matrix-based implementation and to introduce a gain $\mathbf{Q}$. The formation error is defined as in (2.17):

$$e_{\mathrm{L,F}} := \|\mathbf{P_x}(\hat{\tilde{\mathbf{q}}}_L - \hat{\tilde{\mathbf{q}}}_F)\|^2 - d_{L,F}^2 \tag{3.8}$$

We start from (2.18) and alter its components as follows:

$$J_F(\hat{\mathbf{q}}_L(\cdot), \hat{\mathbf{u}}_F(\cdot)) = C \cdot J_{F,1}(\hat{\tilde{\mathbf{q}}}_L(\cdot), \hat{\tilde{\mathbf{q}}}_F(\cdot)) + Q \cdot J_{F,2}(\hat{\tilde{\mathbf{q}}}_F(t), \hat{\tilde{\mathbf{q}}}_F(t-1)), \tag{3.9}$$

1. The first term, $J_{F,1}$, penalizes deviations from the desired formation, thereby enforcing the correct spacing with respect to the leader:

$$J_{F,1}(\hat{\tilde{\mathbf{q}}}_L(\cdot), \hat{\tilde{\mathbf{q}}}_F(\cdot)) := \boldsymbol{\beta}_N \cdot e_{\mathrm{L,F}}(t)^2 \tag{3.10}$$

   where $\boldsymbol{\beta}_N = \begin{bmatrix} \beta & \beta^2 & \dots & \beta^N \end{bmatrix} \in \mathbb{R}^{1\times N}$ and $\beta \in (0, 1]$.

2. The second term, $J_{F,2}$, penalizes excessive motion of the robot, encouraging shorter trajectories and eliminating superfluous movements:

$$J_{F,2}(\hat{\tilde{\mathbf{q}}}_F(t), \hat{\tilde{\mathbf{q}}}_F(t-1)) := \|\mathbf{P_x}(\hat{\tilde{\mathbf{q}}}_F(t) - \hat{\tilde{\mathbf{q}}}_F(t-1))\|^2 \tag{3.11}$$

3.3 Obstacle avoidance

The construction of a safe area through half-spaces is a very clever approach and provides an effective way to cast the obstacle avoidance task as a convex optimization problem. Since we are using a nonlinear MPC, we must define a nonlinear function that replicates the role of the matrices $\mathbf{A}_{bar}$ and $\mathbf{B}_{bar}$ introduced in (2.28) for the linear MPC of the previous system. Starting from

$$\mathbf{A}_{bar} \cdot q \leq \mathbf{B}_{bar}, \tag{3.12}$$

we aim to define a function $c(q) \leq 0$ which, given the robot state, returns a set of inequalities that allows the MPC to verify whether the obtained solution results in a collision with obstacles or not. We define $c(q)$ as

$$c(q) := \mathbf{A}_{bar} \cdot q - \mathbf{B}_{bar} + \mathbf{S} \qquad s.t. \qquad c(q) \leq 0, \tag{3.13}$$

where the matrices $\mathbf{A}_{bar}$ and $\mathbf{B}_{bar}$ are defined in Appendix 6.2, and $\mathbf{S} \in \mathbb{R}^{NML \times 1}$ is a matrix[1] that introduces a safety margin in the half-space computation. This term shifts all half-spaces inward, toward the robot's center, thereby increasing robustness: uncertainties in the model and measurement noise have less impact on the construction of the safe area, reducing the risk of collisions with obstacles. The margin is identical for all inequalities in $c(q)$ and is set to 0.05 m for both the *Leader* and the *Follower*, and 0.01 m for the load's vertices.

To further enhance performance, these matrices are computed once at the beginning of each iteration for each robot. This, however, imposes a significant constraint when dealing with moving obstacles: their velocity must remain lower than that of the *Leader*. If this condition is not met, the obstacle may travel faster than what the robot's prediction accounts for during the MPC safe-area construction, substantially increasing the collision risk.

3.3.1 Awareness range

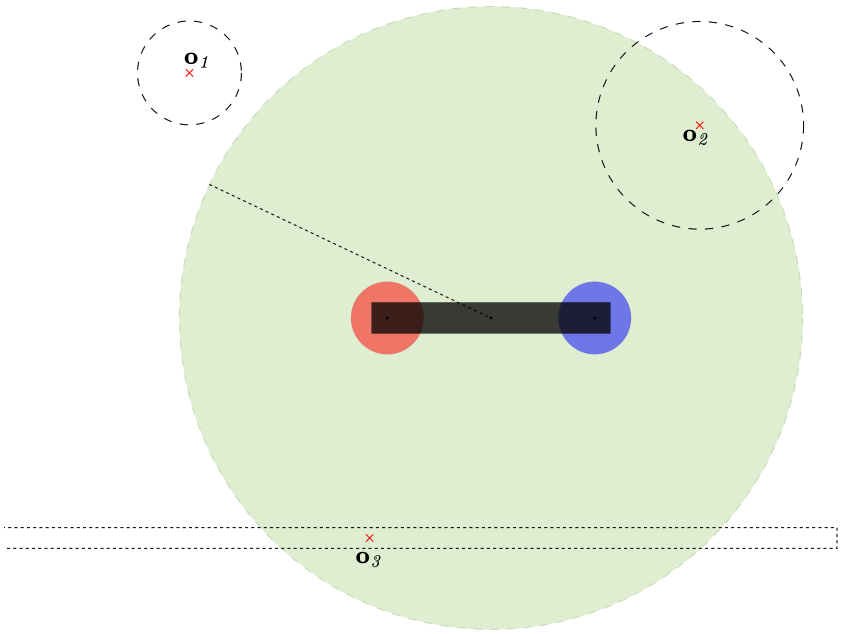

Figure 3.3: Awarness range example.

To improve performance in the obstacle-avoidance computations, we have leveraged the fact that all obstacles are known in advance. Specifically, a function

[1] N is the prediction horizon, M is the number of obstacles, and L is the number of vertices of the robot

has been implemented that creates a circular region around the MRS, centered at the load's center, to identify the nearest and most relevant obstacles. Any obstacle that intersects or lies within this region is retained for the computation of $c(q)$ described in the previous section; all others are disregarded for that iteration. This awareness range ensures that only the most influential obstacles are considered in the MPC computation, significantly reducing computation time, particularly in environments with numerous, widely spaced obstacles. An illustration of this mechanism is provided in Figure 3.3, where only obstacles $\mathbf{o}_2$ and $\mathbf{o}_3$ are retained, and obstacle $\mathbf{o}_1$ is omitted.

3.4 Navigation policy

The Navigation Policy is the key component of this system, enabling the robots to act more intelligently when they encounter obstacles that would otherwise block their path. It can detect when the *Leader* is about to become trapped, temporarily adjust the goal to avoid this, and then, once a reset zone is reached, reinstate the original goal, returning the system to its initial condition. This policy was developed while addressing the edge-case environments discussed in Section 5.3, and was subsequently refined into a basic navigation algorithm that still enables the MRS to reach its target, even when it is enclosed by multiple surrounding walls.

3.4.1 Activation criteria

Although this policy is very useful, it is not suitable for every situation. In essence, it evaluates the distance between two predicted states of the *Leader*, specifically state N and state $N - k_b$, where $k_b \in \mathbb{N}$, $1 \leq k_b \leq N - 1$, is a tunable scalar that determines how far apart the two predictions are. If k_b is too small, the policy may be triggered excessively; if it is too large, the policy may intervene too late, when the robot is already blocked in front of an obstacle. Based on extensive testing, $k_b = 5$ when $N = 20$, was selected as a suitable compromise.

The algorithm computes the distance between the positions of these two predictions. If this distance becomes very close to zero, it indicates that the predictions are converging to a single point and are unable to generate a new path, causing the MRS to remain stuck in front of an obstacle. This situation frequently arises when an obstacle is placed perpendicular to the robot's direction

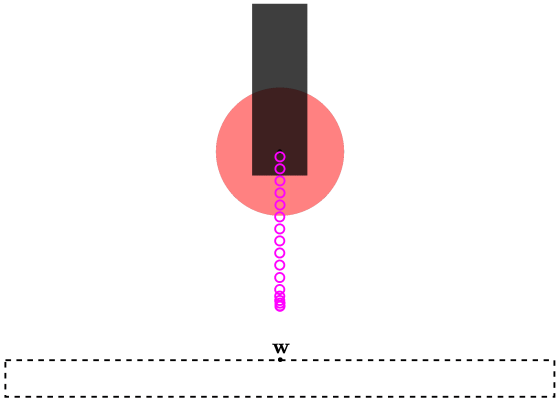

Figure 3.4: Example of a prediction collapse.

of travel, see Figure 3.4.

Because this policy should only be activated in this specific scenario—namely, when the *Leader* is at risk of becoming blocked in front of an obstacle—several conditions must be met before it can be triggered:

- **Distance from the goal**: the predictions are expected to converge when the goal is reached, so the policy must remain inactive if the Nth prediction is within 0.5 m of the goal.
- **Presence of obstacles**: even if it may appear obvious, it is necessary to verify that at least one obstacle is present within the awareness range. Due to possible MPC errors, the predictions might only collapse for a single iteration and then resume following the optimal path to the goal, for instance when the *Leader* slows down or stops, allowing the *Follower* to regain the correct formation without losing contact.
- **Proximity to an obstacle**: complementing the previous condition, it is also necessary to ensure that the Nth prediction lies close to an obstacle, thereby confirming that a collapse of the predictions could indeed occur.
- **Existence of a collapse**: finally, after checking all the previous conditions, it must be verified that an actual collapse of the predictions is taking place.

If all these conditions[2] are satisfied, the policy is activated and a new goal is assigned to the *Leader*, enabling it to circumvent the obstacle instead of remaining stuck in front of it.

[2] Additional conditions were introduced during the implementation of the code to guarantee correct behavior.

3.4.2 Direction selection and goal generation

When the policy detects that a prediction collapse is imminent, a new goal must be generated for the *Leader*. Before doing so, it is necessary to determine in which direction and at what position this goal should be placed relative to the robot's pose. Two candidate goals are created: one that would cause the *Leader* to turn left and another that would make it turn right. Using the direction θ_o, obtained from the direction of the Nth prediction and the point associated with the closest obstacle[3], these two goals are placed 15 m away along the directions $\theta_o + \alpha$ and $\theta_o - \alpha$, respectively. We set $\alpha = \frac{\pi}{5}$ as the deviation angle; this value was chosen to enable the system to avoid both circular obstacles and obstacles modeled as rectangular walls. Once these new goals are defined, we must select the most suitable direction for the MRS—namely, the one that maximizes the likelihood of avoiding deadlock and, crucially, reduces the risk of future blockages.

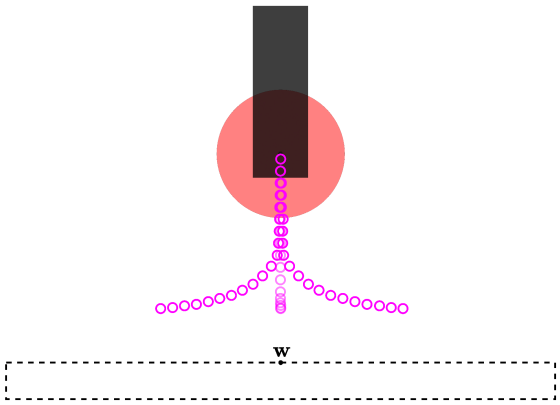

Figure 3.5: Trajectory simulation for both directions.

To make this choice, we simulate the *Leader*'s trajectory using the MPC for each of the two candidate goals, keeping the constraints fixed at the values they had when the policy was triggered, as seen in Figure 3.5. These simulations are computationally inexpensive because the constraints remain constant and only the *Leader*'s motion is propagated. We perform $\eta = 3$ MPC runs, ensuring that each subsequent run starts from the Nth state predicted by the previous one. After completing all η simulations for both goals, we count how many consecutive predictions occur before a collapse arises between two successive

[3]This direction may differ from the *Leader*'s heading θ_L.

predictions. If, in one of the two scenarios, the *Leader* travels significantly farther than in the other, we choose the corresponding goal as the new temporary target. If both simulations lead to the same traveled distance, we select the goal that requires the smallest change in direction with respect to the *Leader*'s heading at the time the policy was activated.

This procedure enables us to determine the *Leader*'s most promising direction, ensuring that it also evaluates whether its future motion could result in another blockage of the MRS.

3.4.3 Reset zone

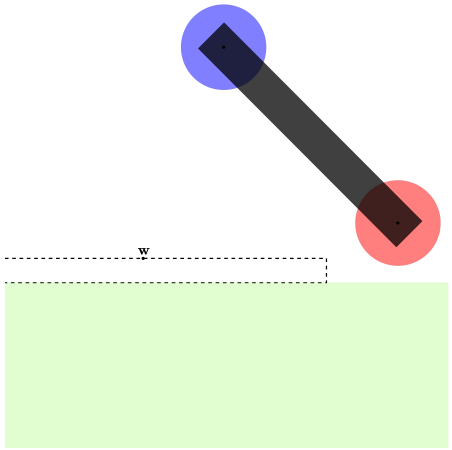

Figure 3.6: Visual representation of a reset zone.

Once the new temporary goal is set, a mechanism is required that enables the policy to stop and resume the original goal. To achieve this, a *"reset zone"* is introduced: as soon as the *Leader* enters this region, the policy is deactivated and the system switches back to its standard behavior. To construct the reset zone, we first determine a point **P**, defined as the position of prediction N at the instant the policy is triggered. From this point, we generate a line shifted by a margin[4] δ and in the direction of θ_o, namely

$$\mathbf{n} = \begin{bmatrix} \cos(\theta_o) \\ \sin(\theta_o) \end{bmatrix} \qquad \text{and} \qquad \mathbf{P} = \begin{bmatrix} x_N \\ y_N \end{bmatrix} + \delta \cdot n$$

[4]The margin is set to $\delta = 0.4$ m, chosen based on the robot's dimensions and the safety margin already employed in the obstacle avoidance section

Once these reset-zone parameters are obtained, we can state that, given the *Leader*'s position $\mathbf{q_P} = \begin{bmatrix} x_L & y_L \end{bmatrix}^\top$, if

$$\mathbf{n'} \cdot (\mathbf{q_P} - \mathbf{P}) \geq 0,$$

then the *Leader* lies inside the reset zone, which stops the policy and restores the original goal.

3.4.4 Policy cascade

A cascading policy mechanism has been added, that permits a Navigation Policy to be triggered even when another Navigation Policy is already running. This layered structure was developed to handle situations with multiple perpendicular walls. With this approach, if a prediction collapse occurs while a policy is active, the current policy is temporarily suspended, and a new goal along with a new reset zone is computed from the location of the new predicted collapse. After the *Leader* traverses the reset zone, the active policy ends and the previously suspended policy is resumed. This policy stack can be applied repeatedly, with up to four policies suspended. This cascading technique enhances the practical utility of the Navigation Policy and increases the overall robustness of the system when facing obstacles that would otherwise cause the MRS to become stuck.

3.4.5 Final thoughts

Integrating this policy into the system makes the entire MRS more resilient and better adapted to variations in simulated environments, enabling it to traverse fragmented corridors that would otherwise trap it at the first wall encountered while moving toward the goal. However, it should be stressed that this is not a sophisticated navigation algorithm; it is intended specifically for situations where the robots become trapped and cannot find a valid path. Nonetheless, this policy is highly effective, and its advantages will become evident when we examine the results obtained in the edge-cases environments in Section 5.3.

3.5 Virtual environments

Most of the virtual environments have been reproduced starting from those presented in [8], appropriately scaled down to account for the different speeds of the two systems.

To evaluate the correct behavior of the system, eight maps were designed, featuring circular obstacles and rectangular walls, both static and moving. In every environment, the goal is located at the origin, i.e., $(0, 0)$.

Figure 3.7: Map (a) - No obstacles.

(a) - No obstacles is the simplest map, containing no obstacles. This environment is used to check system stability when velocities can reach their maximum values, as well as to verify that the system operates correctly and that the goal is successfully reached.

(b) - One obstacle describes a scenario in which a single obstacle is placed between the MRS and the goal. This map is used to assess whether the construction of the constraint matrix $c(q)$ for obstacle avoidance is performed correctly.

(c) - Pathological case closely resembles the one obstacle map, except that the obstacle is placed exactly along the trajectory between the *Leader*'s initial position and the goal. As will be shown, without employing the Navigation Policy, this configuration causes the MRS to stop in front of the obstacle, unable to find a valid path to the goal.

(d) - Two obstacles introduces a second circular obstacle into the environment, creating a narrow passage between them. This map is particularly useful

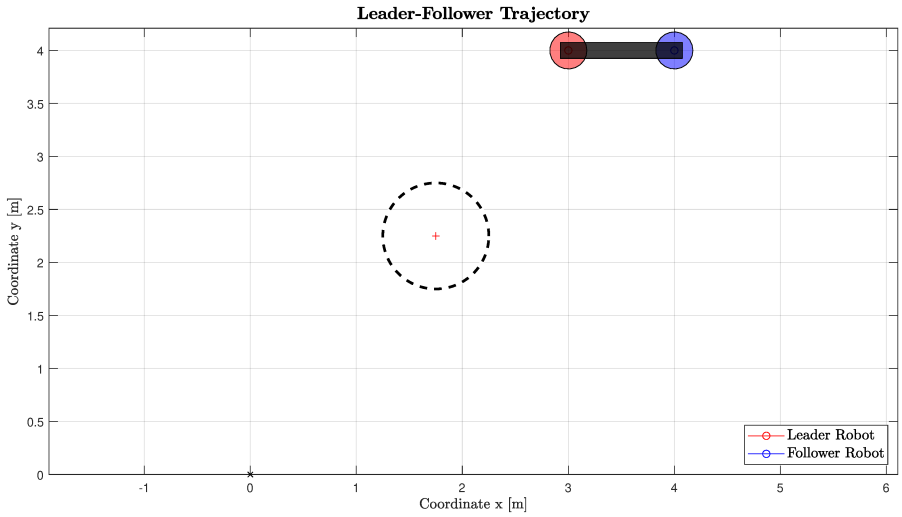

Figure 3.8: Map (b) - One obstacle.

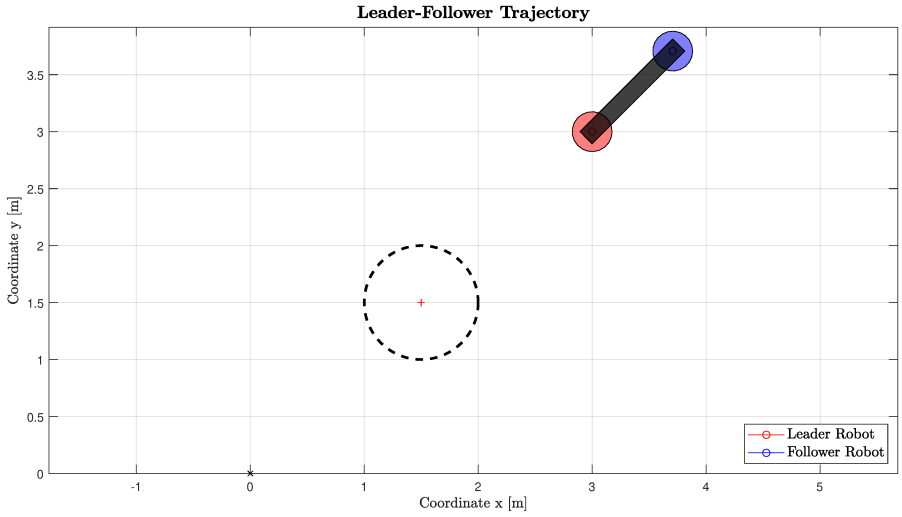

Figure 3.9: Map (c) - Pathological case.

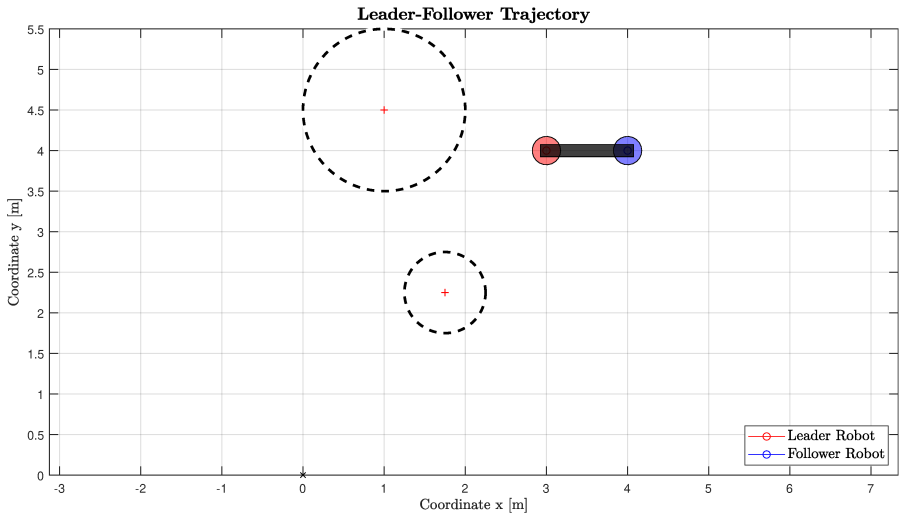

Figure 3.10: Map (d) - Two obstacles.

for validating the correct operation of the system, as it makes it possible to check whether the MRS can traverse a sufficiently tight corridor without getting stuck and without causing the load to collide along the path. This map will also be employed in Chapter 4 for tuning the MPC gains.

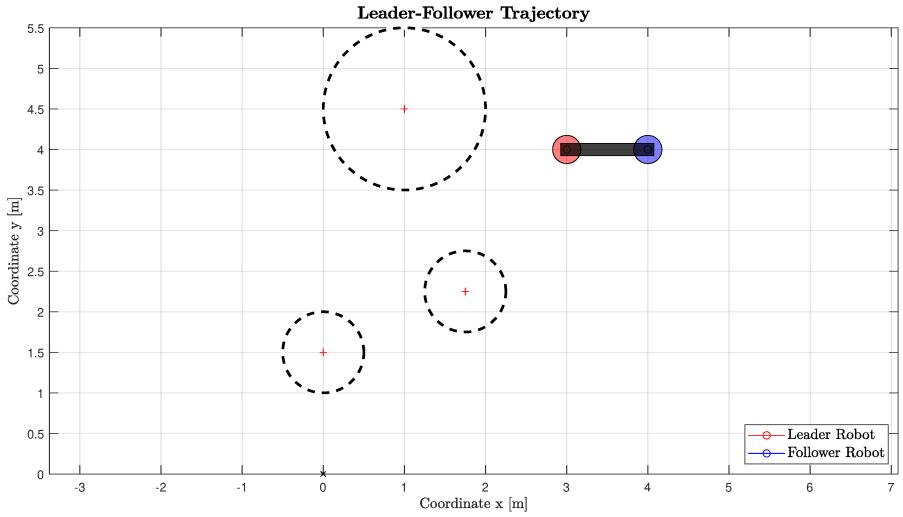

Figure 3.11: (e) - Three obstacles.

(e) - Three obstacles adds an additional obstacle to the previous map, positioned close to the goal, to assess whether the system can correctly manage obstacles located near the destination.

(f) - Six obstacles is among the largest maps developed for this project.

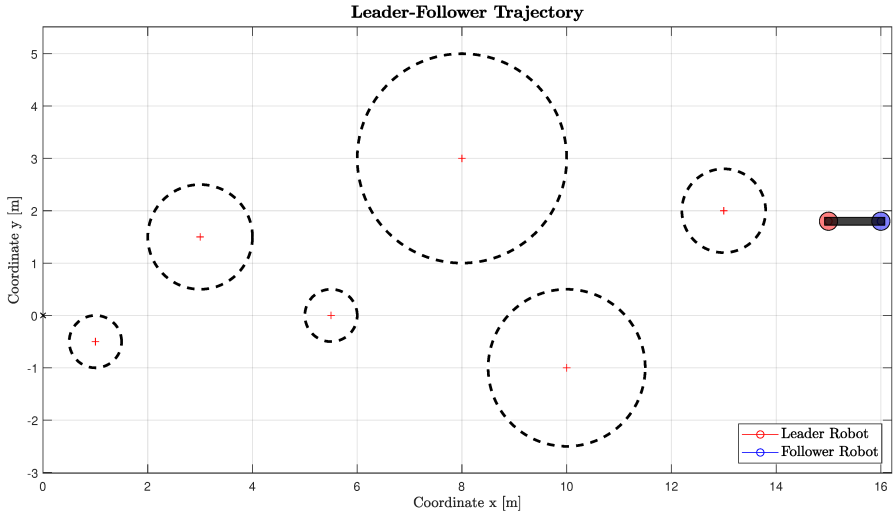

Figure 3.12: Map (f) - Six obstacles.

It includes six circular obstacles of varying radii arranged along the x-axis. This setup enables testing the system over longer travel distances and extended computation times.

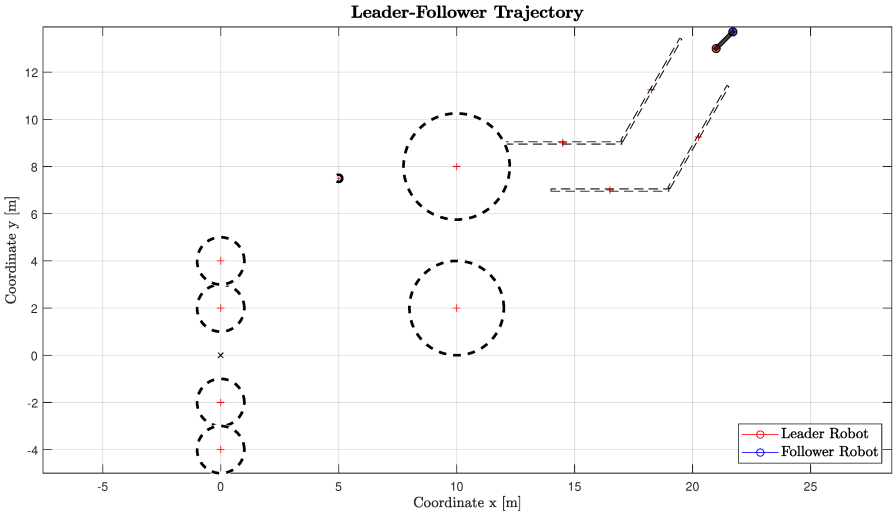

Figure 3.13: Map (g) - Valzer.

(g) - Valzer is the most challenging and realistic virtual environment designed. It comprises a long corridor leading to a narrow passage formed by two circular obstacles. After this passage, a moving obstacle must be avoided to reach the goal. Finally, additional obstacles are positioned near the destination, further complicating the MRS's approach.

Figure 3.14: Map (h) - Moving obstacles.

(h) - Moving obstacles features three obstacles moving in different directions. This map allows us to evaluate whether the system can cope with multiple moving obstacles simultaneously.

3.5.1 Edge cases

To evaluate the system in highly specific and unusual situations, seven test maps were designed to probe potential edge cases and determine whether the system can handle them or instead fails or crashes. Most of these scenarios, if not all, were successfully addressed through the Navigation Policy; therefore, these environments were primarily employed to validate the correct operation of the system when this policy is used.

(i) - Obstacle in front of the goal and **(ii) - Wall in front of the goal** are two maps with the same layout; the only distinction is that the former uses a circular obstacle, whereas the latter employs a rectangular wall. These obstacles lie on the straight line between the goal and the *Leader*'s initial position and are oriented perpendicular to the *Leader*'s motion. This setup makes it possible to check whether the Navigation Policy is correctly triggered for both obstacle shapes.

(iii) - Sharp bend is a map featuring a right-angle turn formed by rectangular walls. This environment was created to verify whether the MRS can traverse a narrow and intricate passage without compromising the payload.

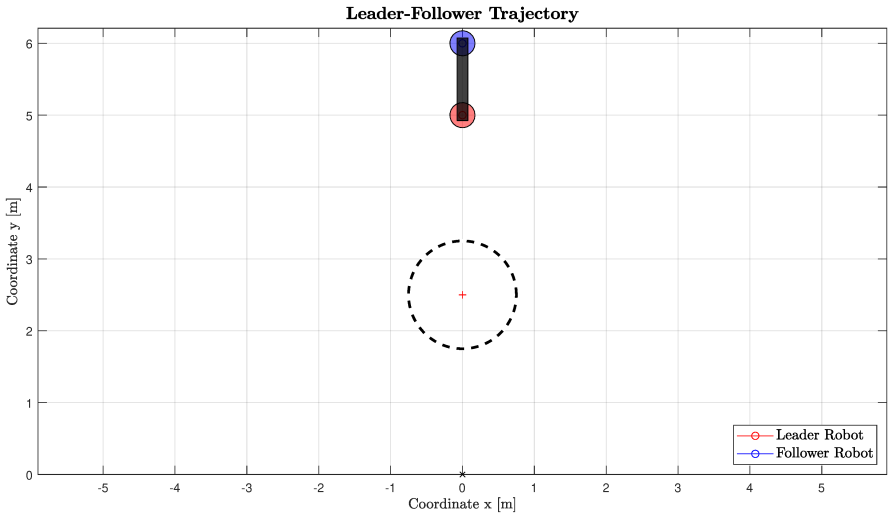

Figure 3.15: Map (i) - Obstacle in front of the goal.

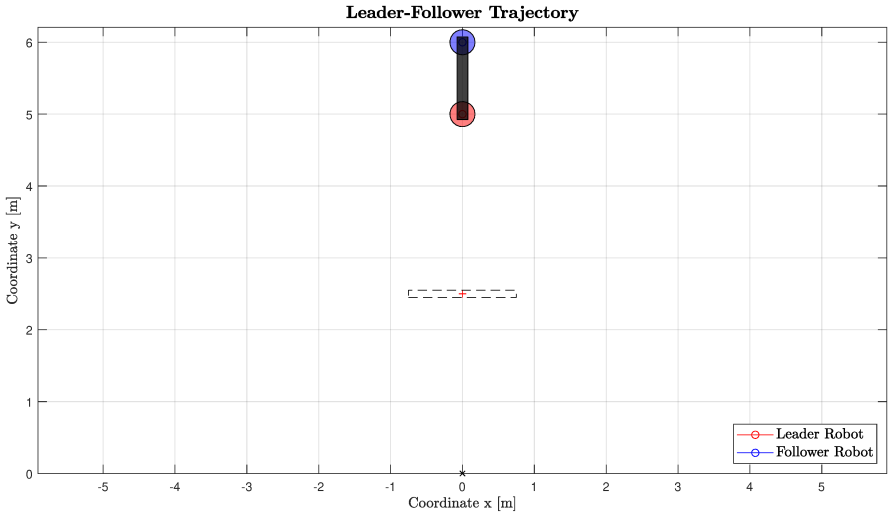

Figure 3.16: Map (ii) - Wall in front of the goal.

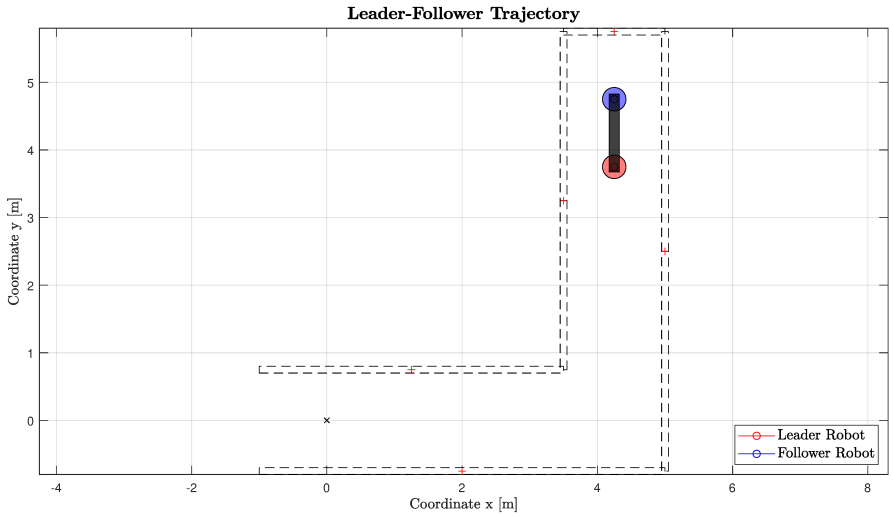

Figure 3.17: Map (iii) - Sharp bend.

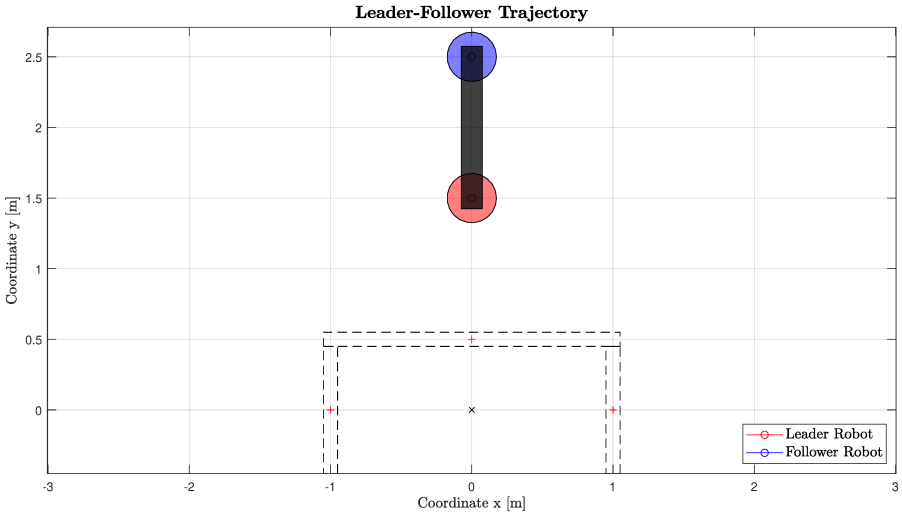

Figure 3.18: Map (iv) - Goal boxed.

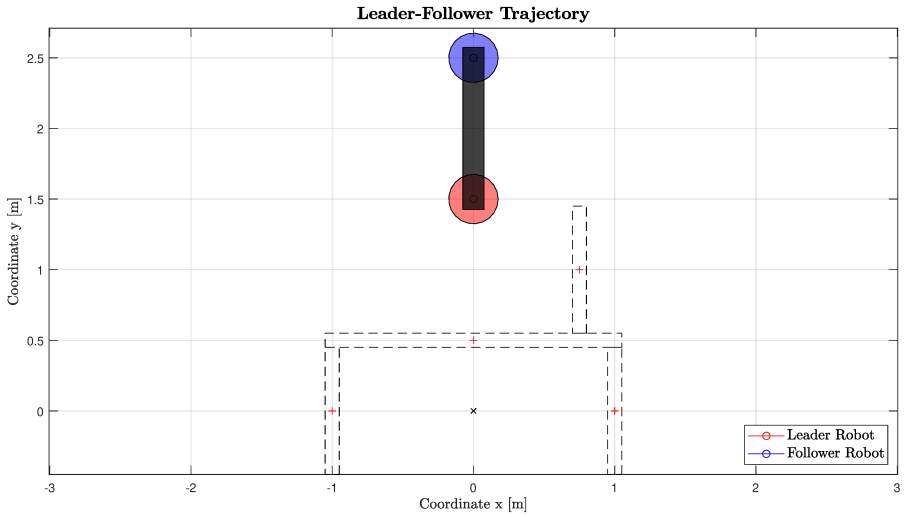

Figure 3.19: Map (v) - Goal boxed and left obstacle.

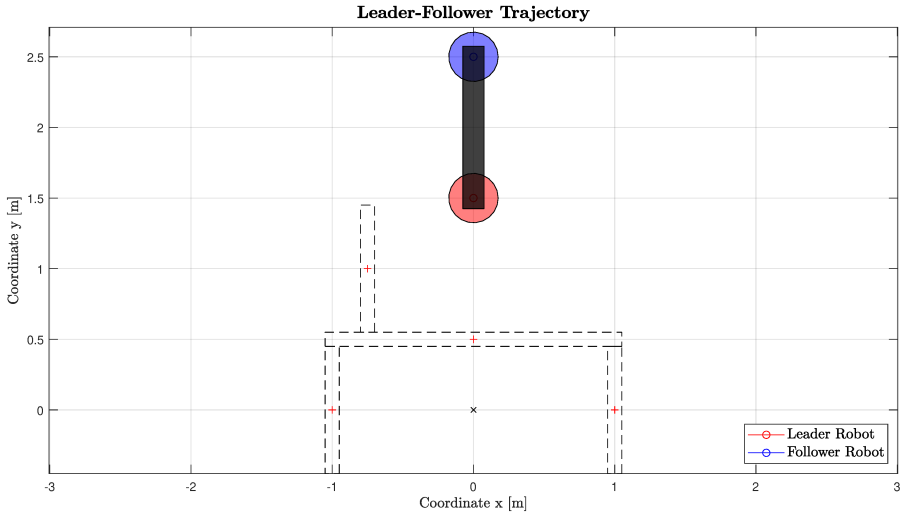

Figure 3.20: Map (vi) - Goal boxed and right obstacle.

(iv) - Goal boxed illustrates a rare but particularly interesting case. The goal is fully surrounded by three rectangular walls, with a single opening positioned on the opposite side of the map. This configuration forces the MRS to travel around the enclosure to reach the goal. Building on this map, two variants were defined: **(v) - Goal boxed and left obstacle** and **(vi) - Goal boxed and right obstacle**, used to assess whether the Navigation Policy can select the more efficient direction to reach the destination.

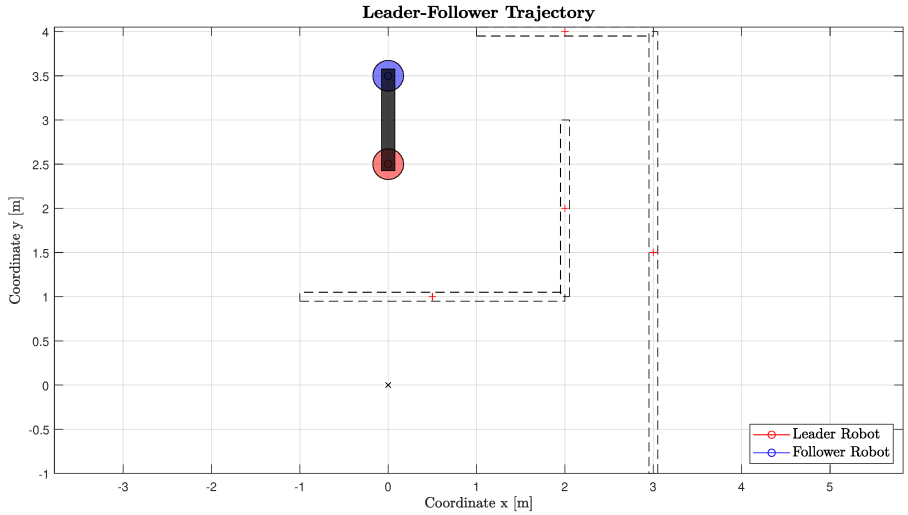

Figure 3.21: Map (vii) - Navigation environment.

(vii) - Navigation environment is used to examine the cascading behavior of the Navigation Policy. It comprises multiple rectangular walls arranged to form 90-degree turns, compelling the MRS to trigger several policies in sequence.

3.6 Perturbation of the system: noise

To evaluate the robustness of the system in simulation, and with an eye toward a future deployment on a physical robot, we conducted experiments in which artificial noise was injected into the system in two specific components: sensing and motor actuation.

3.6.1 Sensing noise

This noise models the system's accuracy in estimating obstacle positions relative to the robots. We aimed to reproduce the behavior of a real LiDAR

sensor: measurements taken close to the robot exhibit small deviations from the true distance, whereas measurements of faraway objects are affected by larger errors. To capture this effect, we implemented a function that adds distance-dependent noise to the true measurement:

$$\tilde{d} = \bar{d} + f(\bar{d}, \sigma^2) \quad \text{with} \quad \sigma \in [0.01, 0.05]$$

After several trials, we fixed $\sigma = 0.02$, and defined

$$f(\bar{d}, \sigma^2) = \sigma \cdot \bar{d}^{1.5}$$

The exponential presence on the distance yields low noise at short ranges and increasing noise as the distance grows.

3.6.2 Actuation noise

We also modeled the non-ideal behavior of real motors by perturbing the control inputs of both robots immediately after they are computed by the MPC. In this way, we obtain a more realistic simulation of a differential-drive platform such as the **TurtleBot4**. The noisy input is given by

$$\mathbf{u} = \mathbf{u_{MPC}} + \mathcal{N}(0, \Sigma) \qquad \text{with} \qquad \Sigma_v \in [0.005, 0.02], \Sigma_\omega \in [0.01, 0.05]$$

For both linear and angular velocities, we selected the smallest values in these intervals, namely $\Sigma_v = 0.005$ and $\Sigma_\omega = 0.01$.

Chapter 4

Data assessments and validation

The purpose of this section is to identify and confirm that the gains used in the MPC are optimal, i.e., they minimize the formation error between the two robots. While the system was initially tuned through extensive trial and error, a subsequent decision has been made to develop a method that provides a mathematical basis to justify these parameter choices.

All evaluations were conducted with a prediction horizon of $N = 20$ and with the Navigation Policy activated. The computations were run on a Galaxy Book3 laptop (connected to mains power), featuring an Intel® Core™ i7-1355U processor, integrated Intel® Iris® Xe Graphics, and 16 GB of LPDDR4x RAM. The complete set of tests has required roughly 11 hours, with each test batch executed automatically via a script that stored all relevant data and plots for every combination of optimization variables.

4.1 Optimization data

Eight parameters to be tuned were identified, involving both the *Leader* and the *Follower*. For the *Leader*, the following gains were considered:

- **W**: gain that regulates the distance between the robot and the goal. If it is too small, the *Leader* may deviate from the best path toward the target; if it is too large, the *Leader* will attempt to remain exactly on the optimal trajectory, with the risk of getting stuck in front of obstacles that could otherwise be bypassed.

- **Z**: gain that affects the last state predicted by the MPC and its distance from the goal. This parameter has a strong impact on the robot's behavior. When its value is high, it can "pull" the prediction toward a path that

differs from the nominal optimal one, but that allows the robot to avoid many obstacles, effectively attracting it like a magnet.

- $\mathbf{R_l}$ and $\mathbf{R_a}$: these two gains form the overall gain $\mathbf{R}$, which penalizes the control inputs chosen by the MPC. They are separated so that linear and angular velocities can be weighted independently, making it possible to prioritize one over the other. In principle, we aim to assign a higher weight to linear velocity than to angular velocity, so that the trajectory is preserved while retaining enough freedom to correct it without excessively reducing speed.

- $\mathbf{K}$: this gain penalizes formation error between the two robots. The larger its value, the more the *Leader* tends to slow down to maintain the correct distance from the *Follower*. The cost associated with $\mathbf{K}$ is also modulated by a vector of decay factors $\boldsymbol{\beta}_{N-1}$; however, we fixed $\beta = 0.95$ and did not vary it, since using a smaller value would decrease the importance of later predicted steps.

For the *Follower*, the following gains were defined:

- $\mathbf{C}$ and β: these two gains determine the *Follower*'s ability to stay at the desired distance from the *Leader* while keeping the formation error low. A larger $\mathbf{C}$ increases the impact of this error on the overall cost. We chose to vary the decay factors β to analyze how changes in this parameter affect the system's behavior.

- $\mathbf{Q}$: this gain shapes the *Follower*'s control inputs, influencing whether the robot performs large changes between successive iterations. If $\mathbf{Q}$ is too small, the robot tends to move excessively and unnecessarily; with a suitable value, it restricts motion to the most relevant displacements, with minimal variation between them, thus limiting superfluous motion and energy usage.

For all gains, the initial values were chosen heuristically, by manual tuning, without relying on a formal mathematical procedure, and were adjusted until the observed performance was satisfactory or improved.

Table 4.1: Gain values before optimization

W	**Z**	$\mathbf{R_l}$	$\mathbf{R_l}$	**K**	**C**	β	**Q**
10	1000	5	2,5	10	100	0,95	10

At the end of this chapter, we will see that these parameters have largely stayed the same, and that only a small subset has been modified to enhance the system.

4.2 Tests environments

Four virtual environments were defined to assess system stability, correct obstacle avoidance, and proper activation of the Navigation Policy for each selected gain set. The maps are:

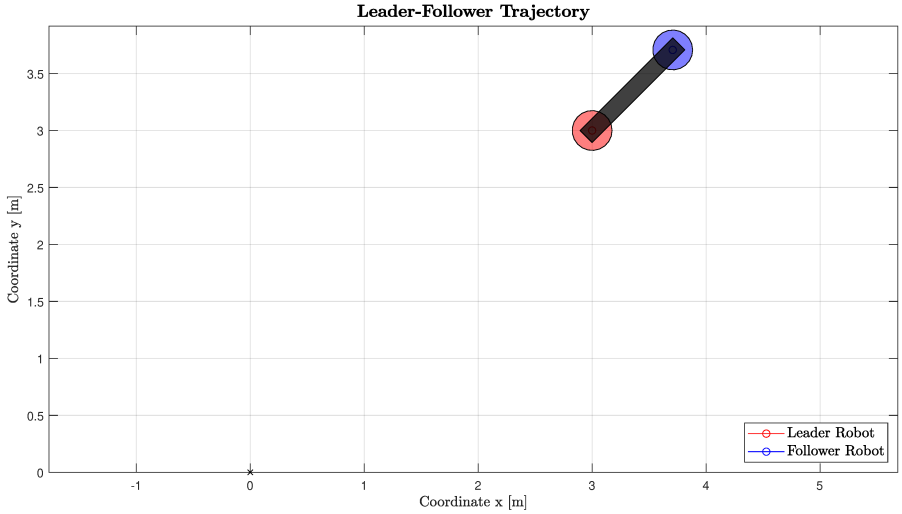

Figure 4.1: Map (α) - No obstacles.

(α) - No obstacles: a map without any obstacles, primarily used to verify that the system can reach the goal and to observe how the inputs vary to achieve this.

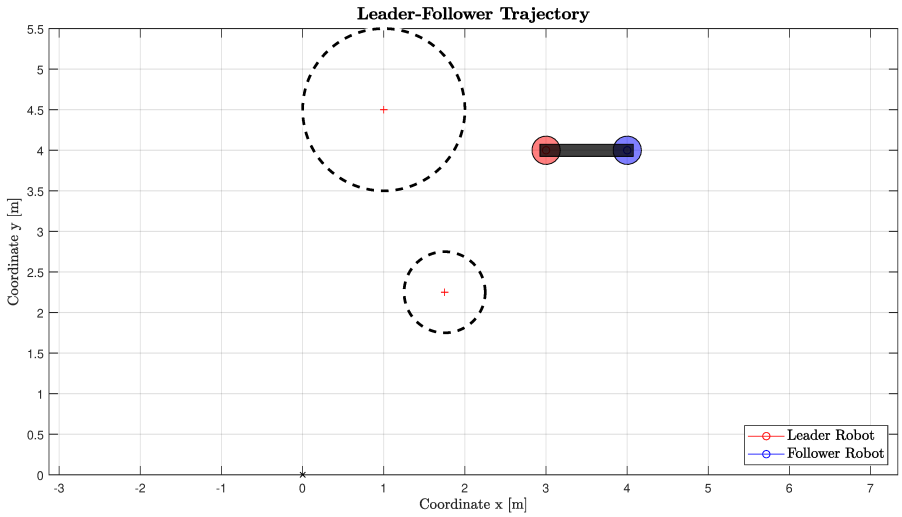

Figure 4.2: Map (β) - Two obstacles.

(β) - Two obstacles: as described in Section 3.5, this environment is crucial for checking correct obstacle avoidance and for evaluating how the MRS traverses a narrow passage—specifically, whether it can do so without becoming stuck and without triggering the Navigation Policy.

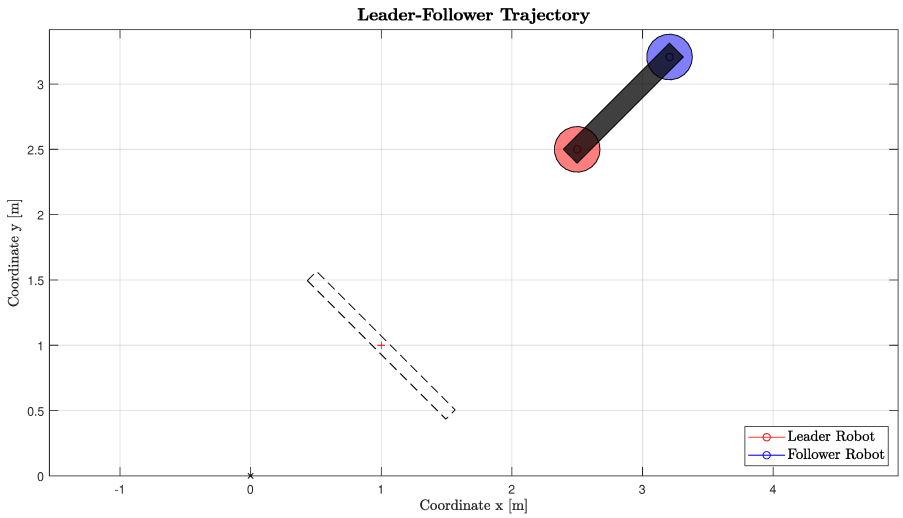

Figure 4.3: Map (γ) - Direct wall.

(γ) - Direct wall: a simple wall placed along the robot's path directly in front of the goal, used to confirm that the Navigation Policy is triggered correctly and that the robots can still reach the goal without damaging the load.

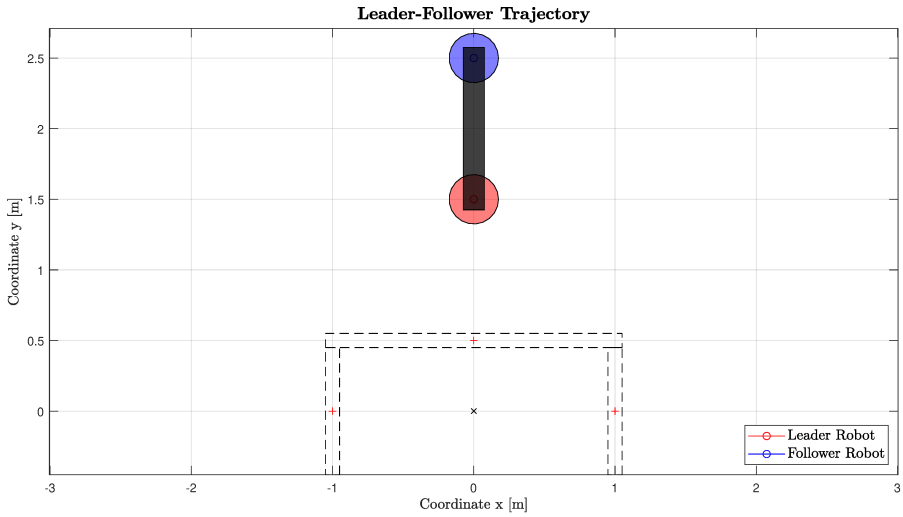

Figure 4.4: Map (δ) - Goal blocked.

(δ) - Goal blocked: identical to the previously discussed edge-case environ-

ment **(iv)**. This map was selected because it presents all the potential challenges the MRS may encounter, making it a powerful benchmark for assessing system performance. For each gain set, four tests were executed on these maps to verify correct behavior and ensure proper tuning of the variables.

4.3 Numerical results

For each gain, values spanning from very small to very large were selected to analyze how the system responds to these changes and how the MRS behavior depends on the numerical magnitude of the adjusted gain. The metrics evaluated for each test are:

- **Iters**: total number of system iterations. If this value reaches the upper limit of 500, the corresponding gain is discarded in advance, even if the remaining metrics for that test configuration appear satisfactory.
- **Time [s]**: number of computational seconds required to complete the test. This metric is mainly used to assess whether a given parameter value accelerates or slows down the overall computation.
- **MAX_e [m]**: the maximum formation error recorded during the test, expressed in meters. Ideally, this should not exceed 5 centimeters; as we will see, this threshold is rarely reached and is consistently undershot.
- **MEAN_e [m]**: the average formation error measured over the duration of the test.
- **Policy used**: number of times the Navigation Policy was triggered. If this count differs from the one reported in the baseline table, the corresponding gain value is rejected.
- **Policy iters**: number of iterations during which the policy remained active, used to compare possible variations across different tests.

As an initial configuration, we adopted the gains listed in 4.9, obtaining the following starting values:

Table 4.2: Base results for optimization

Scenario	Iters	Time [s]	MAX_e [m]	MEAN_e [m]	Policy used	Policy iters
No obstacles	209	11,72	6.35×10^{-5}	7.05×10^{-7}	0	0
Two obstacles	269	84,34	6.60×10^{-5}	4.85×10^{-7}	0	0
Direct wall	251	91,69	2.02×10^{-4}	-2.38×10^{-5}	1	83
Goal blocked	269	240,07	7.84×10^{-4}	-3.06×10^{-5}	2	172

The results are already very promising, showing a formation error of only a few tens of millimeters at most. The respective plots are shown in the images below

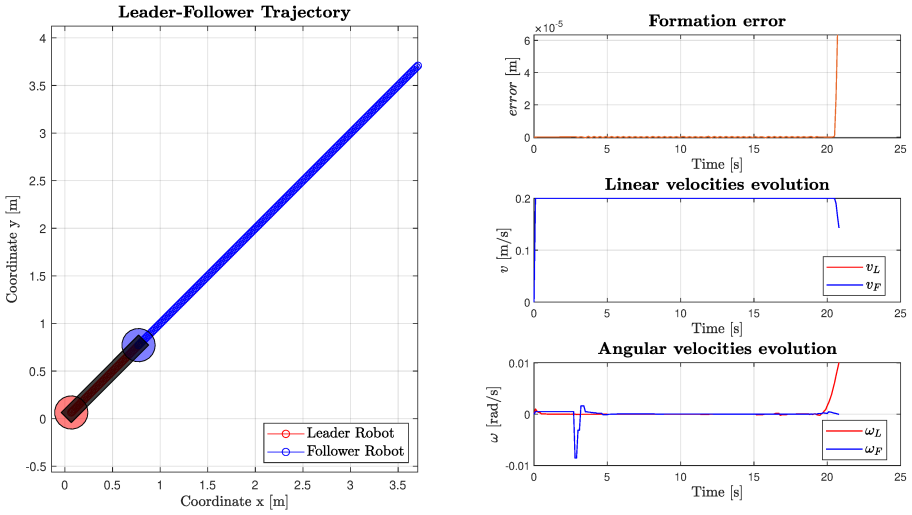

Figure 4.5: Base test on map (α).

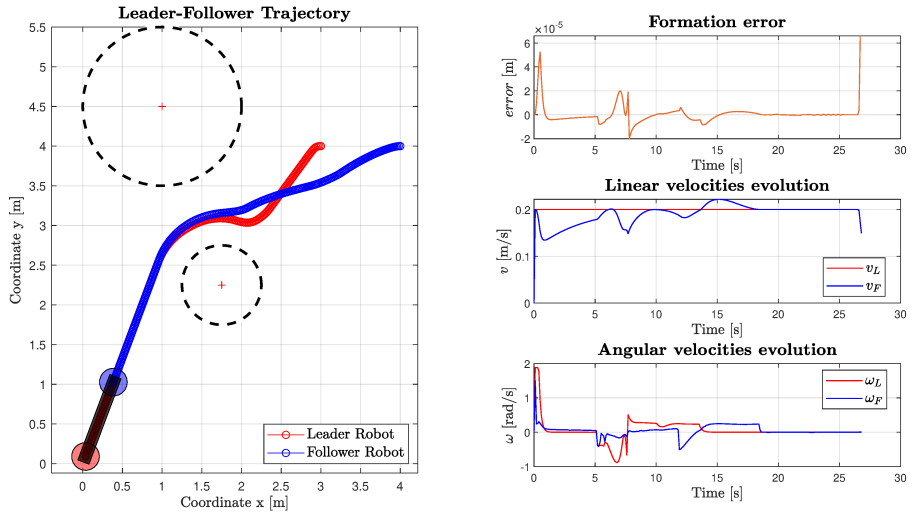

Figure 4.6: Base test on map (β).

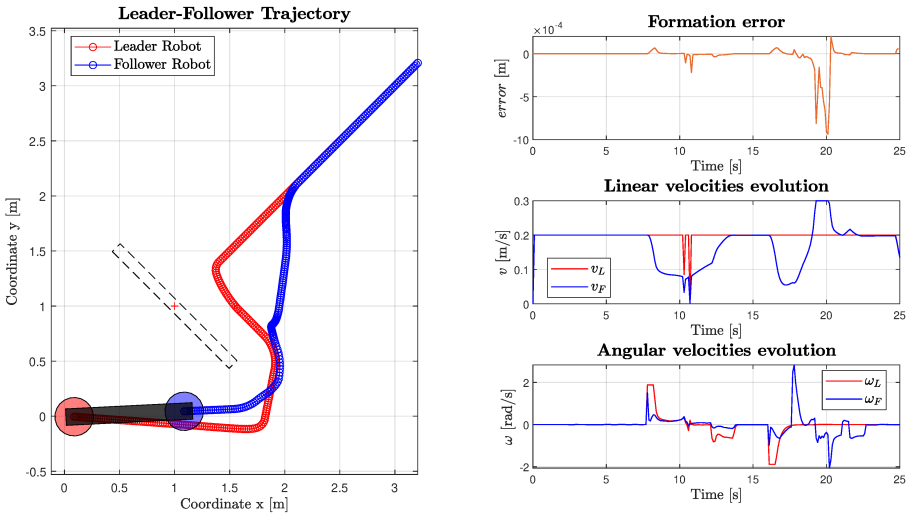

Figure 4.7: Base test on map (γ).

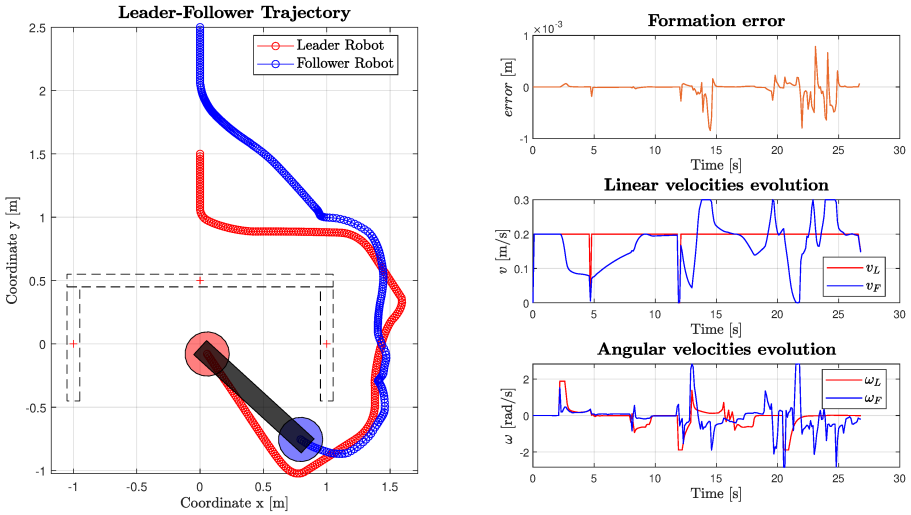

Figure 4.8: Base test on map (δ).

4.3.1 Leader cost W

To evaluate the effect of this gain, we tested five different values, taking the baseline value **10** as a reference, i.e.

$$\mathbf{W} \in \begin{bmatrix} 0.1 & 1 & 10 & 100 & 1000 \end{bmatrix},$$

for a total of 20 simulations, whose outcomes are all reported in Table 4.3.

Table 4.3: Optimization testing of gain W

Scenario	W	Iters	Time [s]	MAX_e [m]	MEAN_e [m]	Policy used	Policy iters
No obstacles	0,1	217	12,38	3.05×10^{-6}	2.20×10^{-7}	0	0
Two obstacles	0,1	277	79,92	4.92×10^{-5}	1.75×10^{-7}	0	0
Direct wall	0,1	271	102,14	5.55×10^{-4}	-2.01×10^{-5}	1	81
Goal blocked	0,1	429	408,38	2.68×10^{-2}	4.45×10^{-4}	2	156
No obstacles	1	214	13,19	1.52×10^{-5}	7.47×10^{-7}	0	0
Two obstacles	1	274	98,02	4.95×10^{-5}	5.96×10^{-7}	0	0
Direct wall	1	256	97,20	2.97×10^{-4}	-2.23×10^{-5}	1	81
Goal blocked	1	278	253,18	7.07×10^{-4}	-1.15×10^{-4}	2	166
No obstacles	10	209	10,71	6.35×10^{-5}	7.05×10^{-7}	0	0
Two obstacles	10	269	103,89	6.60×10^{-5}	4.85×10^{-7}	0	0
Direct wall	10	251	92,85	2.02×10^{-4}	-2.38×10^{-5}	1	83
Goal blocked	10	269	214,81	7.84×10^{-4}	-3.06×10^{-5}	2	172
No obstacles	100	209	10,32	4.57×10^{-7}	2.10×10^{-8}	0	0
Two obstacles	100	269	117,95	5.64×10^{-5}	2.04×10^{-8}	1	55
Direct wall	100	250	119,87	2.32×10^{-4}	-2.57×10^{-5}	1	83
Goal blocked	100	483	1222,00	2.13×10^{-1}	1.12×10^{-2}	4	342
No obstacles	1000	209	8,32	4.49×10^{-7}	2.67×10^{-9}	0	0
Two obstacles	1000	269	150,87	5.70×10^{-5}	2.89×10^{-8}	1	56
Direct wall	1000	250	120,47	2.12×10^{-4}	-8.90×10^{-6}	1	83
Goal blocked	1000	483	1346,18	3.16×10^{-1}	1.28×10^{-2}	4	346

From the table, we can immediately rule out $\mathbf{W} = 100$ and $\mathbf{W} = 1000$ for two main reasons: in both cases the policy is triggered during the test (β), whereas this never happens for the baseline. In addition, in the (δ) tests the maximum formation error exceeds the baseline value and even surpasses 20 centimeters for both of these gains.

Considering the results for $\mathbf{W} = 0.1$, we notice that in the (δ) test the number of iterations almost reaches the maximum allowed, and the maximum error is higher than in the baseline, reaching up to 2 centimeters. Consequently, this gain value must also be discarded.

To select a value between the two remaining candidates, we compare Figures

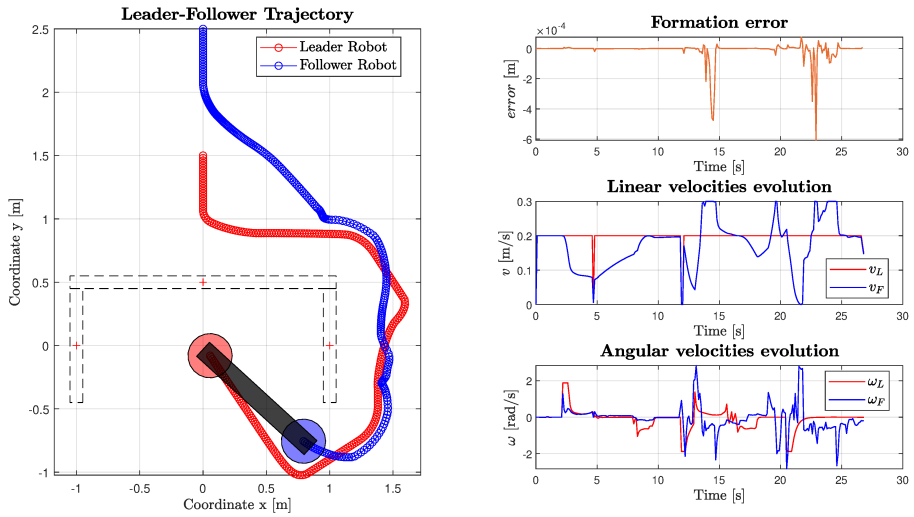

Figure 4.9: W test on map (δ).

(4.9) and (4.8). We have observed that the variance of the formation error in the former is significantly larger than in the latter, where it stays close to 0. Another distinctive aspect of the first figure is the very sharp angle at which the *Leader* initially activates the policy, as if it stops in front of the obstacle before turning to go around it. This behavior has been caused by the low value of **W** used in that test. For these reasons, the best choice for **W** is still the original value, namely 10.

4.3.2 Leader cost Z

The initial value of this gain is already set quite high because, during system development, it was found that a lower value caused the *Leader* to become stuck even in front of simple obstacles that were not perpendicular to its path. Therefore, five values below this initial setting were selected to validate this observation. The selected values are

$$\mathbf{Z} \in \begin{bmatrix} 0.1 & 1 & 10 & 100 & 1000 \end{bmatrix},$$

for a total of 20 simulations, whose outcomes are all reported in Table 4.4.

From the Table 4.4, it is immediately evident that setting **Z** below 1000 causes the policy to be triggered significantly more frequently and for a larger number

Table 4.4: Optimization testing of gain Z

Scenario	Z	Iters	Time [s]	MAX_e [m]	MEAN_e [m]	Policy used	Policy iters
No obstacles	0.1	209	14,43	6.45×10^{-5}	7.85×10^{-7}	0	0
Two obstacles	0.1	270	91,02	6.60×10^{-5}	-7.30×10^{-7}	1	56
Direct wall	0.1	252	102,82	1.23×10^{-4}	7.53×10^{-7}	1	83
Goal blocked	0.1	464	833,30	4.92×10^{-4}	-3.25×10^{-4}	5	325
No obstacles	1	209	20,88	6.44×10^{-5}	7.94×10^{-7}	0	0
Two obstacles	1	270	94,41	6.60×10^{-5}	-5.90×10^{-7}	1	56
Direct wall	1	253	117,91	8.42×10^{-5}	-8.03×10^{-7}	1	84
Goal blocked	1	464	895,86	4.91×10^{-4}	-3.17×10^{-4}	5	325
No obstacles	10	209	14,76	6.44×10^{-5}	7.81×10^{-7}	0	0
Two obstacles	10	270	126,50	6.19×10^{-5}	2.73×10^{-7}	1	56
Direct wall	10	252	117,26	8.33×10^{-5}	4.59×10^{-7}	1	83
Goal blocked	10	274	696,59	1.43×10^{-4}	-4.00×10^{-5}	2	175
No obstacles	100	209	11,53	6.41×10^{-5}	7.29×10^{-7}	0	0
Two obstacles	100	270	119,43	6.31×10^{-5}	-3.01×10^{-7}	1	56
Direct wall	100	253	99,10	2.18×10^{-4}	-1.26×10^{-6}	1	85
Goal blocked	100	/	/	/	/	/	/
No obstacles	1000	209	11,72	6.35×10^{-5}	7.05×10^{-7}	0	0
Two obstacles	1000	269	84,34	6.60×10^{-5}	4.85×10^{-7}	0	0
Direct wall	1000	251	91,69	2.02×10^{-4}	-2.38×10^{-5}	1	83
Goal blocked	1000	269	240,07	7.84×10^{-4}	-3.06×10^{-5}	2	172

of iterations. Under these conditions, in the map ($\boldsymbol{\beta}$), the Navigation Policy is activated, indicating that a low value of $\mathbf{Z}$ drives the system to select the shortest path to the goal, even when this path crosses the obstacles, without attempting to prevent a potential collapse of the prediction. Conversely, when $\mathbf{Z} = 1000$, the policy does not activate. By assigning greater weight to the final state of the prediction, the system is encouraged to move away from the obstacles, as this ultimately brings it closer to the goal over a longer horizon. For this reason, the value of this gain has been kept identical to that of the baseline.

4.3.3 Leader cost R_l and R_a

The gain $\mathbf{R}$ is split into two components so that it can independently affect the linear velocity, $\mathbf{R_l}$, and the angular velocity, $\mathbf{R_a}$. To determine suitable values, the following search range was selected:

$$\mathbf{R_{l,a}} \in \begin{bmatrix} 0.1 & 1 & 5 & 10 & 100 \end{bmatrix},$$

for both gains, resulting in 100 total simulations, all summarized in Table 4.5. In this setup, the baseline values $\mathbf{R_l} = 5$ and $\mathbf{R_a} = 2.5$ are not included among the tested configurations.

Table 4.5: Optimization testing of R_l and R_a

Scenario	R_l	R_a	Iters	Time [s]	MAX_e [m]	MEAN_e [m]	Policy used	Policy iters
No obstacles	0,1	0,1	209	10,73	4.28×10^{-7}	3.03×10^{-9}	0	0
Two obstacles	0,1	0,1	270	79,11	5.93×10^{-5}	-5.93×10^{-7}	0	0
Direct wall	0,1	0,1	249	83,01	2.15×10^{-4}	-2.18×10^{-5}	1	82
Goal blocked	0,1	0,1	270	194,18	7.87×10^{-3}	4.48×10^{-5}	2	170
No obstacles	0,1	1	209	9,71	4.31×10^{-7}	1.94×10^{-8}	0	0
Two obstacles	0,1	1	269	117,90	5.64×10^{-5}	2.46×10^{-8}	0	0
Direct wall	0,1	1	249	102,71	2.70×10^{-4}	-2.56×10^{-5}	1	82
Goal blocked	0,1	1	273	216,88	1.92×10^{-1}	1.20×10^{-2}	2	174
No obstacles	0,1	5	209	10,51	4.48×10^{-7}	1.08×10^{-8}	0	0
Two obstacles	0,1	5	269	122,98	4.97×10^{-5}	7.06×10^{-8}	0	0
Direct wall	0,1	5	250	98,67	3.14×10^{-4}	-8.73×10^{-6}	1	83
Goal blocked	0,1	5	270	235,47	5.40×10^{-4}	-3.48×10^{-5}	2	172
No obstacles	0,1	10	209	11,84	4.07×10^{-7}	1.57×10^{-9}	0	0
Two obstacles	0,1	10	284	138,92	8.99×10^{-3}	8.27×10^{-5}	1	56
Direct wall	0,1	10	250	97,05	2.05×10^{-4}	-3.04×10^{-6}	1	82
Goal blocked	0,1	10	271	253,04	3.26×10^{-4}	-4.39×10^{-5}	2	171
No obstacles	0,1	100	209	13,70	4.35×10^{-7}	1.14×10^{-9}	0	0
Two obstacles	0,1	100	289	166,72	3.41×10^{-3}	1.72×10^{-5}	1	58
Direct wall	0,1	100	500	128,49	2.50×10^{-4}	-5.04×10^{-7}	1	86
Goal blocked	0,1	100	500	404,77	3.26×10^{-5}	-7.94×10^{-8}	1	100
No obstacles	1	0,1	209	11,70	4.44×10^{-7}	3.02×10^{-8}	0	0
Two obstacles	1	0,1	270	110,41	5.95×10^{-5}	-5.85×10^{-7}	0	0
Direct wall	1	0,1	249	95,29	2.29×10^{-4}	-2.12×10^{-5}	1	82
Goal blocked	1	0,1	272	214,58	4.96×10^{-3}	-2.31×10^{-5}	2	170
No obstacles	1	1	209	11,47	4.43×10^{-7}	-3.34×10^{-8}	0	0
Two obstacles	1	1	269	112,57	5.64×10^{-5}	2.41×10^{-8}	0	0
Direct wall	1	1	249	92,58	2.61×10^{-4}	-3.99×10^{-5}	1	82
Goal blocked	1	1	273	345,17	1.86×10^{-1}	1.17×10^{-2}	2	174
No obstacles	1	5	209	12,38	4.38×10^{-7}	9.15×10^{-9}	0	0
Two obstacles	1	5	269	110,75	4.25×10^{-5}	3.87×10^{-8}	0	0
Direct wall	1	5	250	98,05	3.45×10^{-4}	-7.98×10^{-6}	1	83
Goal blocked	1	5	270	239,57	6.91×10^{-4}	-3.89×10^{-5}	2	172
No obstacles	1	10	209	12,38	4.30×10^{-7}	1.11×10^{-8}	0	0
Two obstacles	1	10	284	140,11	9.09×10^{-3}	8.35×10^{-5}	1	56
Direct wall	1	10	250	105,47	4.15×10^{-4}	4.09×10^{-7}	1	82
Goal blocked	1	10	271	232,40	3.30×10^{-4}	-5.50×10^{-5}	2	171
No obstacles	1	100	209	10,73	4.44×10^{-7}	9.64×10^{-9}	0	0
Two obstacles	1	100	289	176,94	3.05×10^{-3}	1.59×10^{-5}	1	58
Direct wall	1	100	500	145,88	1.01×10^{-4}	-3.49×10^{-7}	1	86
Goal blocked	1	100	500	267,95	3.58×10^{-5}	-3.44×10^{-8}	1	100
No obstacles	5	0,1	209	13,08	6.41×10^{-5}	6.56×10^{-7}	0	0
Two obstacles	5	0,1	270	112,41	5.99×10^{-5}	-1.78×10^{-7}	0	0
Direct wall	5	0,1	249	97,30	1.67×10^{-4}	-2.57×10^{-5}	1	82
Goal blocked	5	0,1	272	224,31	4.55×10^{-3}	-9.75×10^{-6}	2	168
No obstacles	5	1	209	10,41	6.40×10^{-5}	6.93×10^{-7}	0	0
Two obstacles	5	1	270	109,85	6.21×10^{-5}	6.14×10^{-7}	0	0
Direct wall	5	1	249	100,30	2.73×10^{-4}	-3.08×10^{-5}	1	82
Goal blocked	5	1	270	378,50	2.59×10^{-2}	6.64×10^{-4}	2	174
No obstacles	5	5	209	10,95	6.38×10^{-5}	6.96×10^{-7}	0	0
Two obstacles	5	5	269	132,21	6.46×10^{-5}	5.59×10^{-7}	0	0
Direct wall	5	5	251	101,14	3.20×10^{-4}	-8.32×10^{-6}	1	83
Goal blocked	5	5	270	242,89	1.14×10^{-3}	-3.88×10^{-5}	2	171
No obstacles	5	10	209	10,61	6.34×10^{-5}	7.03×10^{-7}	0	0
Two obstacles	5	10	284	134,93	8.68×10^{-3}	8.04×10^{-5}	1	56
Direct wall	5	10	250	102,24	2.91×10^{-4}	-9.56×10^{-7}	1	82
Goal blocked	5	10	271	241,71	2.79×10^{-4}	-3.58×10^{-5}	2	171
No obstacles	5	100	209	10,91	6.47×10^{-5}	7.07×10^{-7}	0	0
Two obstacles	5	100	290	165,71	2.95×10^{-3}	1.53×10^{-5}	1	58
Direct wall	5	100	500	158,26	4.80×10^{-5}	-2.80×10^{-7}	1	86
Goal blocked	5	100	500	223,13	3.26×10^{-5}	9.33×10^{-10}	1	100

Scenario	R_l	R_a	Iters	Time [s]	MAX_e [m]	MEAN_e [m]	Policy used	Policy iters
No obstacles	10	0,1	210	10,98	4.35×10^{-5}	1.09×10^{-6}	0	0
Two obstacles	10	0,1	272	113,04	5.94×10^{-5}	2.80×10^{-7}	0	0
Direct wall	10	0,1	251	95,66	1.94×10^{-4}	-2.30×10^{-5}	1	82
Goal blocked	10	0,1	272	222,87	7.69×10^{-3}	1.07×10^{-7}	2	164
No obstacles	10	1	210	10,16	4.46×10^{-5}	1.08×10^{-6}	0	0
Two obstacles	10	1	271	113,41	5.64×10^{-5}	8.79×10^{-7}	0	0
Direct wall	10	1	251	98,56	1.64×10^{-3}	-2.43×10^{-5}	1	82
Goal blocked	10	1	311	176,94	2.14×10^{-1}	3.39×10^{-2}	2	172
No obstacles	10	5	210	10,28	4.50×10^{-5}	1.05×10^{-6}	0	0
Two obstacles	10	5	270	147,13	4.37×10^{-5}	8.87×10^{-7}	0	0
Direct wall	10	5	252	101,83	3.47×10^{-4}	-8.44×10^{-6}	1	83
Goal blocked	10	5	271	232,26	8.59×10^{-4}	-3.08×10^{-5}	2	171
No obstacles	10	10	210	12,48	4.49×10^{-5}	1.07×10^{-6}	0	0
Two obstacles	10	10	286	129,42	8.55×10^{-3}	7.93×10^{-5}	1	56
Direct wall	10	10	252	102,60	2.84×10^{-4}	-4.12×10^{-7}	1	82
Goal blocked	10	10	274	331,50	7.60×10^{-4}	-5.07×10^{-5}	2	172
No obstacles	10	100	210	10,24	4.49×10^{-5}	1.08×10^{-6}	0	0
Two obstacles	10	100	291	174,91	2.68×10^{-3}	1.44×10^{-5}	1	58
Direct wall	10	100	500	157,16	3.34×10^{-5}	-3.06×10^{-7}	1	86
Goal blocked	10	100	500	170,07	3.26×10^{-5}	-2.37×10^{-9}	1	100
No obstacles	100	0,1	226	15,43	1.13×10^{-5}	9.82×10^{-7}	0	0
Two obstacles	100	0,1	287	118,47	5.93×10^{-5}	7.61×10^{-7}	0	0
Direct wall	100	0,1	270	112,33	2.36×10^{-4}	-1.89×10^{-5}	1	81
Goal blocked	100	0,1	337	490,22	2.00×10^{-1}	1.84×10^{-2}	2	161
No obstacles	100	1	226	15,22	1.16×10^{-5}	9.74×10^{-7}	0	0
Two obstacles	100	1	286	122,01	5.63×10^{-5}	7.84×10^{-7}	0	0
Direct wall	100	1	270	115,69	2.54×10^{-4}	-8.66×10^{-6}	1	81
Goal blocked	100	1	328	385,89	9.77×10^{-4}	-6.98×10^{-5}	2	156
No obstacles	100	5	226	14,33	1.15×10^{-5}	9.78×10^{-7}	0	0
Two obstacles	100	5	286	156,75	4.24×10^{-5}	8.42×10^{-7}	0	0
Direct wall	100	5	270	115,67	3.14×10^{-4}	-9.09×10^{-6}	1	81
Goal blocked	100	5	296	276,56	3.91×10^{-4}	-3.11×10^{-5}	2	166
No obstacles	100	10	226	15,61	1.14×10^{-5}	1.00×10^{-6}	0	0
Two obstacles	100	10	304	150,85	4.15×10^{-3}	2.20×10^{-5}	1	55
Direct wall	100	10	272	119,35	6.12×10^{-4}	1.64×10^{-6}	1	82
Goal blocked	100	10	298	321,49	8.14×10^{-4}	-4.07×10^{-5}	2	166
No obstacles	100	100	226	13,76	1.18×10^{-5}	9.75×10^{-7}	0	0
Two obstacles	100	100	308	182,36	1.24×10^{-3}	2.96×10^{-6}	1	57
Direct wall	100	100	500	184,03	8.18×10^{-5}	-6.87×10^{-7}	1	84
Goal blocked	100	100	500	177,84	4.01×10^{-5}	-1.43×10^{-6}	1	97

With 25 distinct pairs to consider, it is necessary to discard as many as possible before examining and comparing all 100 plots. From Table 4.5, we can immediately discard all cases with $\mathbf{R_a} = 100$, since for every such configuration the system reaches the maximum number of iterations in the (γ) and (δ) maps. We can also exclude all combinations with $\mathbf{R_l} = 0.1$, because they produce formation errors larger than the baseline and trigger policy activation in the map (β) in at least one trial.

To further narrow down the candidates, we observed that when $\mathbf{R_l} \leq \mathbf{R_a}$, the policy tends to activate when it should not, and the formation error increases; we therefore avoid these settings. Finally, configurations with $\mathbf{R_l} = 100$ are discarded, since such a large value would downplay other, more relevant costs, particularly those related to the distance from the goal. After applying all these filters, only 6 configurations remain to be examined.

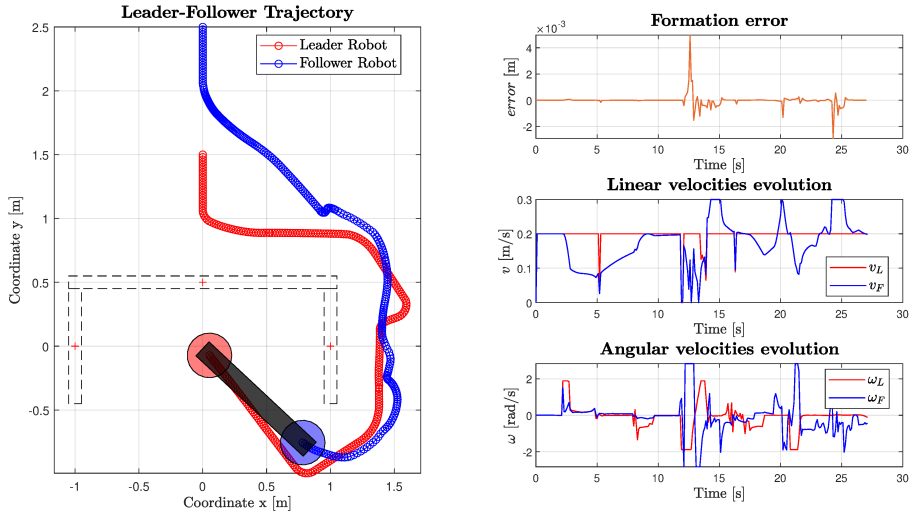

Figure 4.10: R test on map (δ). $\mathbf{R_l} = 1$ and $\mathbf{R_a} = 0.1$.

Examining the plots for the tests conducted in the (δ) environment, we observed that when setting $\mathbf{R_a} = 0.1$ in all three configurations, the *Follower* tends to spin on the spot, as illustrated for instance in Figure 4.10. This behavior is unexpected, because the gains being tuned should primarily affect the *Leader* rather than the *Follower*. A plausible explanation is that, since the MRS is a tightly coupled system in which the two robots strongly depend on each other, assigning a low weight to variations in the *Leader*'s angular velocity indirectly alters the *Follower*'s path. To keep tracking the *Leader* or to avoid collisions involving the load, the *Follower* is then forced to halt and rapidly change direction, which gives

rise to these "spin" trajectories.

We can also discard the configurations with $\mathbf{R_a} = 1$, because in both cases the formation error is greater than in the baseline. The only remaining option is $\mathbf{R_l} = 10$ and $\mathbf{R_a} = 5$, which has coincided with our initial parameter choice, namely $\mathbf{R_a} = \mathbf{R_l}/2$. Nonetheless, we have ultimately decided to retain the baseline gains: by comparing the formation error plots in Figure 4.8 with those for this configuration in Figure 4.11, it becomes clear that the variance is more stable in the baseline case.

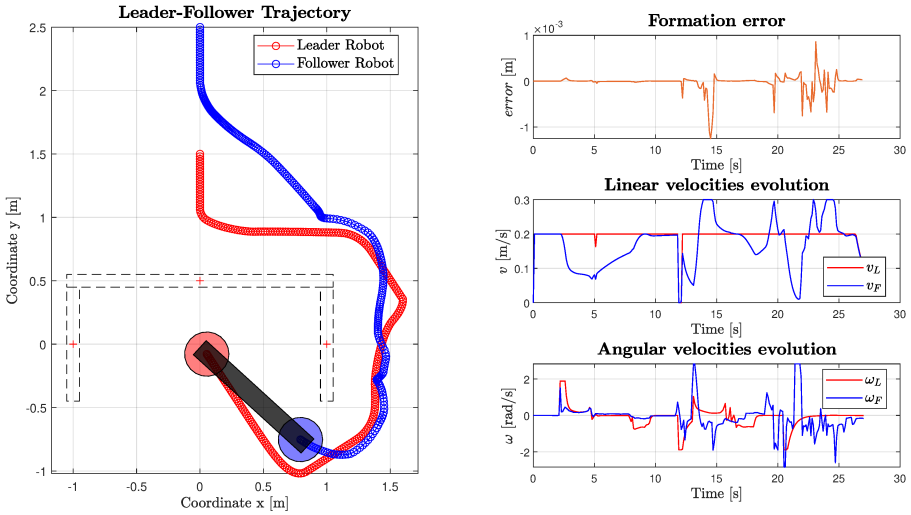

Figure 4.11: R test on map (δ). $\mathbf{R_l} = 10$ and $\mathbf{R_a} = 5$.

4.3.4 Leader cost K

To evaluate how this gain influences the system, we selected ten distinct values,

$$\mathbf{K} \in \begin{bmatrix} 0.1 & 1 & 5 & 10 & 100 & 200 & 400 & 600 & 800 & 1000 \end{bmatrix},$$

resulting in 40 simulations in total, all summarized in Table 4.6. This range was chosen because the gain directly affects the newly added cost associated with providing the *Leader* with information about the *Follower*'s position. Our goal was to determine whether this cost has a measurable impact on the system and, if so, to understand how variations in the gain modify its behavior.

Table 4.6: Optimization testing of gain K

Scenario	K	Iters	Time [s]	MAX_e [m]	MEAN_e [m]	Policy used	Policy iters
No obstacles	0,1	209	11,80	6.42×10^{-5}	7.15×10^{-7}	0	0
Two obstacles	0,1	269	78,70	6.63×10^{-5}	5.13×10^{-7}	0	0
Direct wall	0,1	251	83,84	2.33×10^{-4}	-2.47×10^{-5}	1	83
Goal blocked	0,1	270	179,44	6.96×10^{-4}	-2.65×10^{-5}	2	173
No obstacles	1	209	10,68	6.40×10^{-5}	7.20×10^{-7}	0	0
Two obstacles	1	269	115,78	6.57×10^{-5}	5.08×10^{-7}	0	0
Direct wall	1	251	87,71	1.74×10^{-4}	-2.31×10^{-5}	1	83
Goal blocked	1	271	214,47	6.73×10^{-4}	-3.74×10^{-5}	2	174
No obstacles	5	209	10,28	6.38×10^{-5}	7.12×10^{-7}	0	0
Two obstacles	5	269	96,29	6.62×10^{-5}	5.14×10^{-7}	0	0
Direct wall	5	251	90,14	2.34×10^{-4}	-2.52×10^{-5}	1	83
Goal blocked	5	269	197,64	4.83×10^{-4}	-4.49×10^{-5}	2	172
No obstacles	10	209	10,20	6.35×10^{-5}	7.05×10^{-7}	0	0
Two obstacles	10	269	107,13	6.60×10^{-5}	4.85×10^{-7}	0	0
Direct wall	10	251	88,61	2.02×10^{-4}	-2.38×10^{-5}	1	83
Goal blocked	10	269	197,36	7.84×10^{-4}	-3.06×10^{-5}	2	172
No obstacles	100	209	11,46	5.65×10^{-5}	5.29×10^{-7}	0	0
Two obstacles	100	269	113,53	5.89×10^{-5}	3.51×10^{-7}	0	0
Direct wall	100	249	91,06	2.77×10^{-4}	-2.59×10^{-5}	1	82
Goal blocked	100	270	229,43	7.91×10^{-4}	-3.29×10^{-5}	2	173
No obstacles	200	209	11,47	5.63×10^{-5}	3.59×10^{-7}	0	0
Two obstacles	200	269	86,00	5.73×10^{-5}	2.91×10^{-7}	0	0
Direct wall	200	249	87,06	2.80×10^{-4}	-1.28×10^{-5}	1	82
Goal blocked	200	269	213,07	5.62×10^{-4}	-4.43×10^{-5}	2	172
No obstacles	400	209	11,62	4.48×10^{-5}	2.24×10^{-7}	0	0
Two obstacles	400	269	123,76	5.25×10^{-5}	1.23×10^{-7}	0	0
Direct wall	400	251	97,26	3.06×10^{-4}	-2.72×10^{-5}	1	83
Goal blocked	400	271	194,28	4.64×10^{-4}	-3.82×10^{-5}	2	172
No obstacles	600	209	10,94	3.07×10^{-4}	1.59×10^{-6}	0	0
Two obstacles	600	270	92,55	5.23×10^{-5}	-1.89×10^{-6}	0	0
Direct wall	600	251	91,65	2.59×10^{-4}	-1.97×10^{-5}	1	83
Goal blocked	600	493	465,79	1.68×10^{-3}	-2.40×10^{-3}	4	336
No obstacles	800	209	11,86	2.50×10^{-4}	1.23×10^{-6}	0	0
Two obstacles	800	270	113,97	5.28×10^{-5}	-1.43×10^{-6}	1	53
Direct wall	800	250	92,56	9.14×10^{-4}	-9.16×10^{-6}	1	82
Goal blocked	800	275	232,83	5.08×10^{-4}	-2.27×10^{-5}	2	172
No obstacles	1000	209	13,09	2.10×10^{-4}	6.82×10^{-7}	0	0
Two obstacles	1000	270	130,87	5.36×10^{-5}	-5.00×10^{-8}	1	53
Direct wall	1000	254	101,75	7.94×10^{-3}	8.42×10^{-5}	1	83
Goal blocked	1000	497	341,55	4.30×10^{-4}	-1.28×10^{-2}	4	342

With the exception of $\mathbf{R} = 600, 800, 1000$, Table 4.6 alone does not allow us to draw strong conclusions without jointly analyzing the simulation plots. These three cases were set aside due to their excessive use of the Navigation Policy and the large number of iterations required in some instances. By examining both the Table and the plots, we have noticed a distinctive pattern: higher gain values reduce the formation error, confirming that the cost has been implemented correctly. At the same time, increasing the gain systematically leads to longer computation times in all configurations involving obstacles. Given these observations, it is necessary to select a compromise value, which is precisely $\mathbf{K} = 10$, the baseline chosen for our study.

4.3.5 Follower cost C and β

These gains influence the primary cost term governing the *Follower*'s MPC, as they enable the maintenance of the desired distance from the *Leader* by keeping the formation error as small as possible. For the gain $\mathbf{C}$, five values were considered:

$$\mathbf{C} \in \begin{bmatrix} 1 & 10 & 100 & 1000 & 5000 \end{bmatrix},$$

whereas for the decay factors β, four values were selected:

$$\beta \in \begin{bmatrix} 0.85 & 0.9 & 0.95 & 1 \end{bmatrix},$$

yielding a total of 80 simulations, all summarized in Table 4.7. In these experiments, the objective is not only to identify the optimal gain value, but also to analyze how varying the decay factors affects the overall system behavior.

By looking at the Table 4.7, we can immediately discard $\mathbf{C} = 1, 10, 5000$, since in all tests these values yield a maximum formation error substantially larger than the baseline. This outcome is expected: reducing $\mathbf{C}$ relative to the baseline reduces the weight assigned to the cost of preserving the MRS formation. For the same reason, we also exclude all configurations with $\beta = 1$ as well as the pairs $[100, 0.9]$, $[1000, 0.95]$, and $[1000, 0.85]$.

By examining the trajectories for all configurations in the ($\boldsymbol{\delta}$) scenario, we observe that changes in β also affect the *Follower*'s path: the closer β is to 1, the

Table 4.7: Optimization results for different values of C and β

Scenario	C	β	Iters	Time [s]	MAX_e [m]	MEAN_e [m]	Policy used	Policy iters
No obstacles	1	1	209	5,46	3.97×10^{-3}	6.94×10^{-5}	0	0
Two obstacles	1	1	270	83,50	3.97×10^{-3}	3.32×10^{-4}	0	0
Direct wall	1	1	251	82,32	5.02×10^{-3}	6.71×10^{-4}	1	83
Goal blocked	1	1	270	191,78	7.20×10^{-3}	1.32×10^{-3}	2	172
No obstacles	1	0,95	209	5,56	4.39×10^{-3}	8.68×10^{-5}	0	0
Two obstacles	1	0,95	270	109,62	4.38×10^{-3}	3.70×10^{-4}	0	0
Direct wall	1	0,95	250	81,29	5.46×10^{-3}	5.95×10^{-4}	1	82
Goal blocked	1	0,95	270	197,11	7.18×10^{-3}	1.36×10^{-3}	2	172
No obstacles	1	0,9	209	4,79	4.89×10^{-3}	1.49×10^{-4}	0	0
Two obstacles	1	0,9	270	92,63	4.90×10^{-3}	4.20×10^{-4}	0	0
Direct wall	1	0,9	250	85,56	6.17×10^{-3}	7.31×10^{-4}	1	82
Goal blocked	1	0,9	270	196,83	8.11×10^{-3}	1.33×10^{-3}	2	172
No obstacles	1	0,85	209	4,60	5.62×10^{-3}	4.18×10^{-4}	0	0
Two obstacles	1	0,85	270	102,61	5.60×10^{-3}	5.32×10^{-4}	0	0
Direct wall	1	0,85	250	77,02	1.86×10^{-2}	1.31×10^{-3}	1	82
Goal blocked	1	0,85	270	197,91	2.03×10^{-2}	2.13×10^{-3}	2	172
No obstacles	10	1	209	8,19	5.34×10^{-4}	6.63×10^{-6}	0	0
Two obstacles	10	1	269	93,47	5.35×10^{-4}	3.38×10^{-5}	0	0
Direct wall	10	1	251	87,93	4.50×10^{-3}	1.89×10^{-4}	1	83
Goal blocked	10	1	269	223,82	2.25×10^{-3}	3.02×10^{-4}	2	172
No obstacles	10	0,95	209	6,11	5.70×10^{-4}	7.19×10^{-6}	0	0
Two obstacles	10	0,95	269	103,08	5.68×10^{-4}	3.62×10^{-5}	0	0
Direct wall	10	0,95	251	89,56	2.87×10^{-3}	1.58×10^{-4}	1	83
Goal blocked	10	0,95	269	208,01	2.38×10^{-3}	2.77×10^{-4}	2	172
No obstacles	10	0,9	209	5,88	6.10×10^{-4}	7.79×10^{-6}	0	0
Two obstacles	10	0,9	269	137,80	6.06×10^{-4}	4.12×10^{-5}	0	0
Direct wall	10	0,9	250	87,68	1.60×10^{-3}	1.32×10^{-4}	1	82
Goal blocked	10	0,9	269	204,93	2.54×10^{-3}	2.78×10^{-4}	2	172
No obstacles	10	0,85	209	5,47	6.51×10^{-4}	8.51×10^{-6}	0	0
Two obstacles	10	0,85	269	126,04	6.47×10^{-4}	4.35×10^{-5}	0	0
Direct wall	10	0,85	250	88,71	1.37×10^{-3}	1.25×10^{-4}	1	82
Goal blocked	10	0,85	269	222,88	2.60×10^{-3}	2.75×10^{-4}	2	172
No obstacles	100	1	209	14,15	6.07×10^{-5}	8.31×10^{-7}	0	0
Two obstacles	100	1	269	115,76	6.35×10^{-5}	3.54×10^{-6}	0	0
Direct wall	100	1	251	105,12	1.60×10^{-4}	4.84×10^{-5}	1	83
Goal blocked	100	1	270	223,41	3.81×10^{-3}	1.24×10^{-4}	2	173
No obstacles	100	0,95	209	10,19	6.35×10^{-5}	8.28×10^{-7}	0	0
Two obstacles	100	0,95	269	120,89	6.60×10^{-5}	3.74×10^{-6}	0	0
Direct wall	100	0,95	251	94,28	9.37×10^{-4}	3.51×10^{-5}	1	83
Goal blocked	100	0,95	269	202,48	8.44×10^{-4}	7.07×10^{-5}	2	172
No obstacles	100	0,9	209	10,71	6.75×10^{-5}	8.39×10^{-7}	0	0
Two obstacles	100	0,9	269	103,69	7.01×10^{-5}	3.95×10^{-6}	0	0
Direct wall	100	0,9	252	96,74	1.19×10^{-3}	3.32×10^{-5}	1	84
Goal blocked	100	0,9	269	213,36	7.38×10^{-4}	6.04×10^{-5}	2	172
No obstacles	100	0,85	209	7,26	7.16×10^{-5}	8.76×10^{-7}	0	0
Two obstacles	100	0,85	269	123,75	7.43×10^{-5}	4.43×10^{-6}	0	0
Direct wall	100	0,85	251	91,56	5.81×10^{-4}	2.67×10^{-5}	1	83
Goal blocked	100	0,85	269	200,42	7.37×10^{-4}	5.47×10^{-5}	2	172
No obstacles	1000	1	209	15,06	6.19×10^{-6}	2.03×10^{-7}	0	0
Two obstacles	1000	1	269	139,96	6.41×10^{-6}	3.93×10^{-7}	0	0
Direct wall	1000	1	251	127,64	3.38×10^{-3}	9.49×10^{-5}	1	83
Goal blocked	1000	1	270	253,63	1.98×10^{-3}	6.17×10^{-5}	2	173
No obstacles	1000	0,95	209	18,25	6.51×10^{-6}	1.83×10^{-7}	0	0
Two obstacles	1000	0,95	269	122,98	6.77×10^{-6}	3.89×10^{-7}	0	0
Direct wall	1000	0,95	251	131,62	3.72×10^{-3}	9.14×10^{-5}	1	83
Goal blocked	1000	0,95	269	236,48	1.23×10^{-3}	4.31×10^{-5}	2	172

Scenario	C	β	Iters	Time [s]	MAX_e [m]	MEAN_e [m]	Policy used	Policy iters
No obstacles	1000	0,9	209	17,70	6.89×10^{-6}	1.78×10^{-7}	0	0
Two obstacles	1000	0,9	269	119,35	7.16×10^{-6}	4.19×10^{-7}	0	0
Direct wall	1000	0,9	251	116,43	9.19×10^{-4}	2.04×10^{-5}	1	83
Goal blocked	1000	0,9	269	219,98	6.07×10^{-4}	1.75×10^{-5}	2	172
No obstacles	1000	0,85	209	13,20	7.29×10^{-6}	1.74×10^{-7}	0	0
Two obstacles	1000	0,85	269	124,40	7.58×10^{-6}	4.38×10^{-7}	0	0
Direct wall	1000	0,85	251	110,19	1.39×10^{-4}	4.35×10^{-6}	1	83
Goal blocked	1000	0,85	270	215,20	1.02×10^{-3}	1.54×10^{-5}	2	173
No obstacles	5000	1	209	34,22	1.30×10^{-6}	7.01×10^{-8}	0	0
Two obstacles	5000	1	269	199,06	1.54×10^{-6}	1.08×10^{-7}	0	0
Direct wall	5000	1	250	134,37	3.69×10^{-3}	1.22×10^{-4}	1	82
Goal blocked	5000	1	269	283,86	1.47×10^{-3}	3.66×10^{-5}	2	172
No obstacles	5000	0,95	209	27,95	1.37×10^{-6}	7.05×10^{-8}	0	0
Two obstacles	5000	0,95	269	155,06	1.37×10^{-6}	9.24×10^{-8}	0	0
Direct wall	5000	0,95	250	134,32	2.09×10^{-3}	6.01×10^{-5}	1	82
Goal blocked	5000	0,95	269	256,51	9.80×10^{-4}	2.76×10^{-5}	2	172
No obstacles	5000	0,9	209	36,16	1.38×10^{-6}	5.82×10^{-8}	0	0
Two obstacles	5000	0,9	269	170,70	2.70×10^{-6}	1.15×10^{-7}	0	0
Direct wall	5000	0,9	250	131,34	9.38×10^{-4}	2.40×10^{-5}	1	82
Goal blocked	5000	0,9	270	250,71	1.94×10^{-3}	2.21×10^{-5}	2	173
No obstacles	5000	0,85	209	21,86	1.46×10^{-6}	9.82×10^{-8}	0	0
Two obstacles	5000	0,85	269	152,34	1.53×10^{-6}	1.03×10^{-7}	0	0
Direct wall	5000	0,85	250	124,32	4.82×10^{-4}	7.21×10^{-6}	1	82
Goal blocked	5000	0,85	269	254,76	8.78×10^{-4}	2.14×10^{-5}	2	172

more the predicted trajectories attempt to preserve the nominal distance from the *Leader*'s predictions, resulting in a path that tends to be strongly attracted to the *Leader*'s trajectory. In contrast, for smaller values of β, the *Follower* does not try to match the *Leader*'s path exactly but, when necessary, takes longer routes to better preserve the formation.

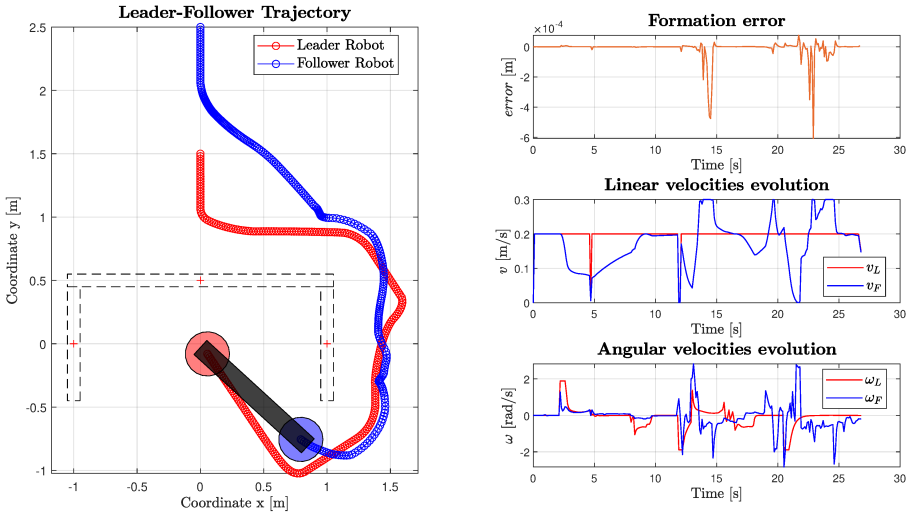

Figure 4.12: C/β test on map (δ). **C** = 1000 and β = 0.9.

After these exclusions, only three parameter combinations remained to be

evaluated. Among them, we selected $\mathbf{C} = 1000$ and $\beta = 0.9$, thereby modifying the gain parameters. Comparing Figure 4.12 with the baseline in Figure 4.8, we see that the trajectory is improved and the variance of the formation error is also reduced.

4.3.6 Follower cost Q

The last gain examined is the cost associated with the *Follower*'s inputs generated by the MPC. As before, 5 values were selected:

$$\mathbf{W} \in \begin{bmatrix} 0.1 & 1 & 10 & 100 & 500 \end{bmatrix},$$

leading to 20 simulations in total, whose results are all listed in Table 4.8. Note that the largest value considered is smaller than the one used for gain $\mathbf{C}$, since we aim to prioritise the latter, even at the expense of increased *Follower* motion.

Table 4.8: Optimization results for different values of Q

Scenario	Q	Iters	Time [s]	MAX_e [m]	MEAN_e [m]	Policy used	Policy iters
No obstacles	0,1	209	13,11	6.78×10^{-7}	9.25×10^{-8}	0	0
Two obstacles	0,1	269	101,15	3.27×10^{-7}	1.93×10^{-8}	0	0
Direct wall	0,1	251	97,34	3.74×10^{-3}	4.06×10^{-5}	1	83
Goal blocked	0,1	269	239,17	1.22×10^{-3}	2.67×10^{-5}	2	172
No obstacles	1	209	17,28	6.83×10^{-7}	8.88×10^{-8}	0	0
Two obstacles	1	269	130,52	7.23×10^{-7}	6.86×10^{-8}	0	0
Direct wall	1	251	107,82	1.35×10^{-3}	1.99×10^{-5}	1	83
Goal blocked	1	270	261,47	8.58×10^{-3}	8.85×10^{-5}	2	173
No obstacles	10	209	17,42	6.89×10^{-6}	1.78×10^{-7}	0	0
Two obstacles	10	269	131,49	7.16×10^{-6}	4.19×10^{-7}	0	0
Direct wall	10	251	112,23	9.19×10^{-4}	2.04×10^{-5}	1	83
Goal blocked	10	269	222,53	6.07×10^{-4}	1.75×10^{-5}	2	172
No obstacles	100	209	23,84	6.67×10^{-5}	7.70×10^{-7}	0	0
Two obstacles	100	269	186,80	7.09×10^{-5}	4.24×10^{-6}	0	0
Direct wall	100	251	121,99	7.80×10^{-4}	3.55×10^{-5}	1	83
Goal blocked	100	269	218,08	7.87×10^{-4}	5.60×10^{-5}	2	172
No obstacles	500	209	13,84	3.19×10^{-4}	3.84×10^{-6}	0	0
Two obstacles	500	269	176,05	3.21×10^{-4}	1.97×10^{-5}	0	0
Direct wall	500	251	116,61	1.94×10^{-3}	9.35×10^{-5}	1	83
Goal blocked	500	269	229,65	1.24×10^{-3}	1.65×10^{-4}	2	172

From Table 4.8, we observe that the baseline tests yield the lowest maximum and average formation errors. After also inspecting the plots and confirming the absence of significant trajectory variations, we have concluded that the baseline value $\mathbf{Q} = 10$ is the optimal choice.

In summary, the optimal values identified from these tests are:

Table 4.9: Gain values after optimization

W	**Z**	$\mathbf{R_l}$	$\mathbf{R_a}$	**K**	**C**	β	**Q**
10	1000	5	2,5	10	1000	0,9	10

Chapter 5

Results and discussion

In this chapter, we have presented and analyzed the results produced by the system when applied to the virtual environments introduced in Section 3.5, using the optimal gains identified in Chapter 4 and a prediction horizon of $N = 20$.

We have also briefly compared these outcomes with those of the original system to assess whether, under the same overall concept but with different robots and execution details, the performance has changed. As will become evident, the results are highly encouraging.

Furthermore, we have examined how the system and the implemented Navigation Policy handle challenging edge cases, and we will conclude by demonstrating the system's robustness through the injection of artificial noise.

5.1 MATLAB simulations

For each environment, the complete simulation has been executed and, to validate its proper functioning, two types of plots have been produced afterward: a dynamic plot and a composite plot. The first consists of a video that illustrates the motion of the two robots and the load within the simulated environment, showing all positions and predictions at every time step. This video is useful for checking the simulation and visually inspecting the robots' behavior on the map; however, because it is a video, it will not be included in the written thesis.

The second type of plot combines a diagram of the virtual environment—showing the trajectories created by the two robots—with three time-based graphs (time on the x-axis): the first graph reports the *formation error* between the two robots, while the second and third depict the evolution of the *linear and angular velocities*, respectively. These plots provide access to all key information about the system and allow us to analyze the trajectory achieved by the MRS during the simulation.

As in Chapter 4, the following table shows all of the system's results obtained after running the simulations on the maps that will be presented:

Table 5.1: Simulation results on the standard environments

Scenario	Iters	Time [s]	MAX_e [m]	MEAN_e [m]	Policy used	Policy iters
No obstacles	221	37,59	6.49×10^{-6}	8.00×10^{-8}	0	0
One obstacle	269	93,00	7.22×10^{-6}	6.29×10^{-8}	0	0
Two obstacles	269	100,07	7.16×10^{-6}	5.82×10^{-8}	0	0
Three obstacles	275	130,80	6.43×10^{-6}	8.41×10^{-8}	0	0
Pathological	238	100,32	7.27×10^{-6}	3.45×10^{-8}	1	57
Six obstacles	779	420,76	7.33×10^{-6}	1.06×10^{-8}	0	0
Valzer	1262	436,46	1.71×10^{-5}	-1.38×10^{-7}	0	0
Moving obstacles	386	827	5.32×10^{-5}	-4.60×10^{-5}	1	57

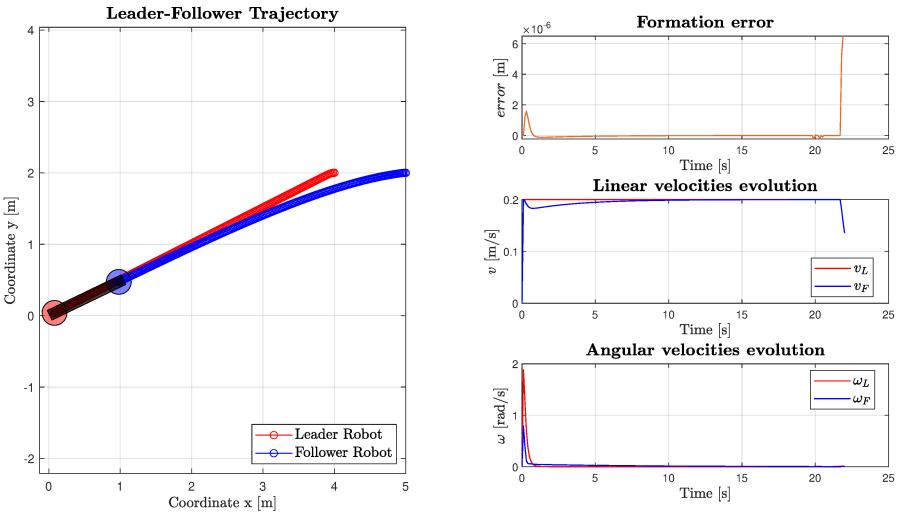

Figure 5.1: Simulation of map (a).

This map, though empty, is central to the system's development: if the MRS cannot reach the goal even in the absence of obstacles, the implementation is inherently flawed. Using this map, we have successfully implemented the robot model and have observed the project operating correctly for the first time. From

Figure 5.1, we can see the *Leader* moving toward the goal while the *Follower* travels behind it, effectively tracing its path. As the robots approach the goal, the *Follower* stays precisely behind the *Leader*, matching its linear velocity to maintain the load at the correct distance. The simulation results are excellent: the maximum formation error is 6.49×10^{-6} m and the mean error is 8×10^{-8} m, both of which are practically imperceptible to the human eye.

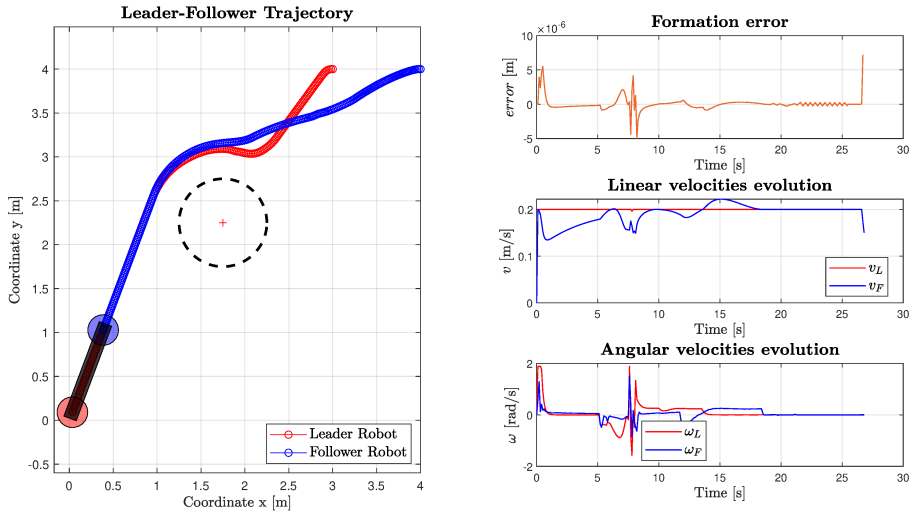

Figure 5.2: Simulation of map (b).

In the first plot of Figure 5.2, the system's ability to detect and navigate around obstacles in the environment is illustrated. The *Leader* promptly adjusts its path upon encountering the obstacle in order to circumvent it, while the *Follower* successfully tracks the *Leader*, maintaining both itself and the load at an appropriate distance from the obstacle to avoid any collision. The *Follower*'s linear velocity is initially lower than that of the *Leader*, as it allows the *Leader* time to identify a suitable trajectory. Once the obstacle has been cleared, the *Follower* increases its speed to regain the correct relative position and to ensure that the load does not collide with the obstacle. Even in this case, with an obstacle present, the tracking error remains very small, on the order of 1×10^{-6} m.

Figure 5.3 presents a scenario analogous to the one just discussed; however, as previously noted, the obstacle is now placed exactly perpendicular to the *Leader*'s trajectory. From Table 5.1, we can see that the Navigation Policy is correctly triggered, preventing the system from becoming stuck due to a failure in the prediction. Aside from this aspect, the results are largely consistent with those obtained in the previous scenario.

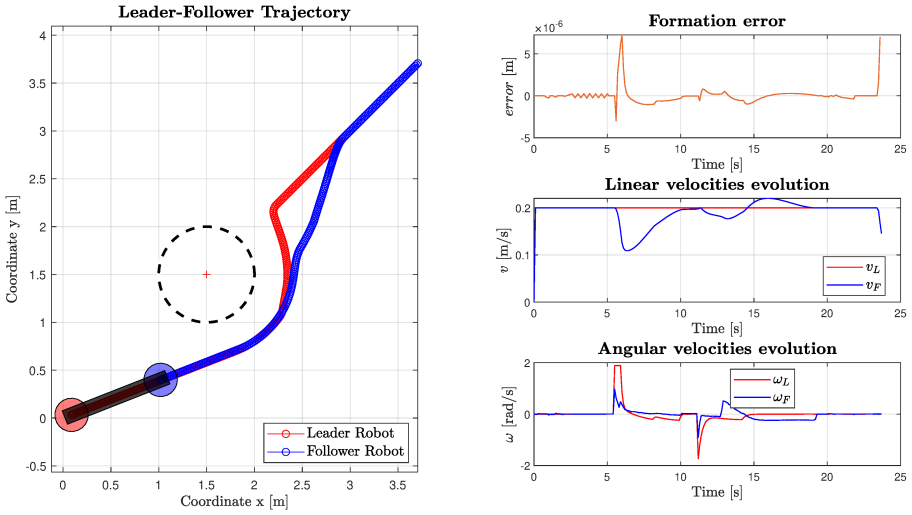

Figure 5.3: Simulation of map (c).

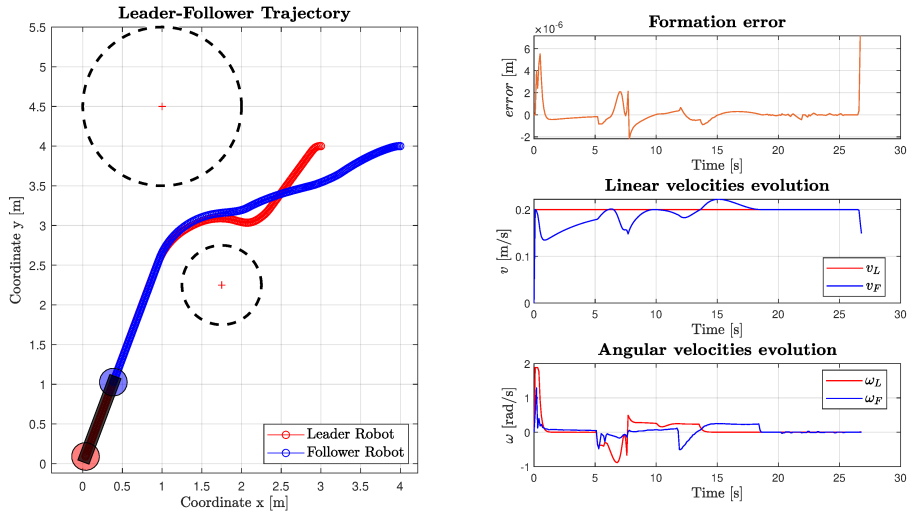

Figure 5.4: Simulation of map (d).

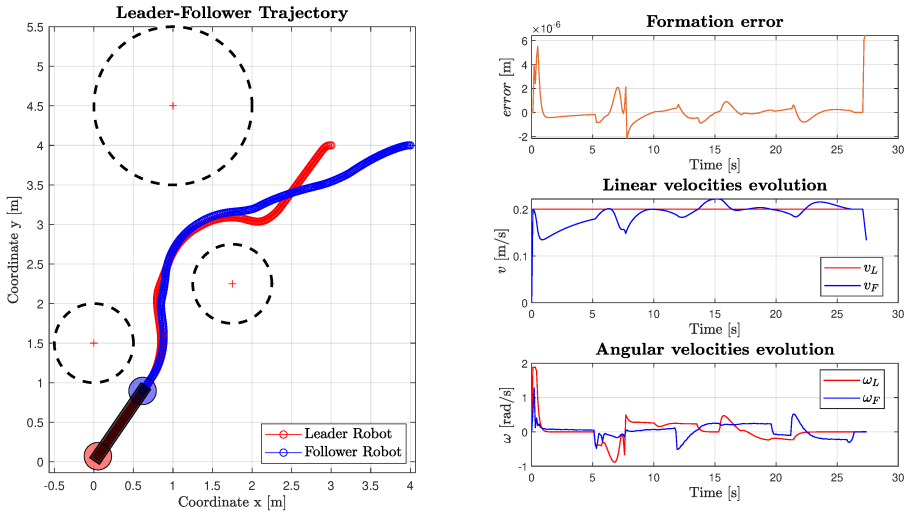

Figure 5.5: Simulation of map (e).

In Figure 5.4, a second obstacle has been introduced into the environment, creating a narrow passage that the MRS must traverse without damaging the load. This scenario is crucial for evaluating the system's capability to move through several obstacles simultaneously. The *Leader*'s trajectory remains largely similar to the previous cases, whereas the *Follower*'s path has been adjusted to prevent the load from striking the lower obstacle and to avoid the *Follower* itself colliding with the upper one. Here as well, the results are highly satisfactory, confirming that the inter-robot distance stays well regulated for the entire simulation. Introducing an additional obstacle near the goal, as shown in Figure 5.5, does not notably affect the final performance or the overall formation error, but it does alter both robots' trajectories so that they avoid colliding with it. Despite this new obstacle being positioned very close to the goal and in a strategically challenging location, the system appears to adapt effectively without significant issues.

To evaluate the system's robustness over long computation times, the environment illustrated in Figure 5.6 has been designed, featuring a large simulated map with many widely spaced obstacles. The system has successfully simulated this scenario without significant problems, maintaining the variance of the formation error stable and very close to zero, except for a peak occurring when the *Leader* reaches the goal. This peak results from the abrupt speed change of the *Leader* on arrival and the *Follower*'s stronger deceleration to prevent a collision. Nevertheless, the resulting error remains very small, so this peak is practically

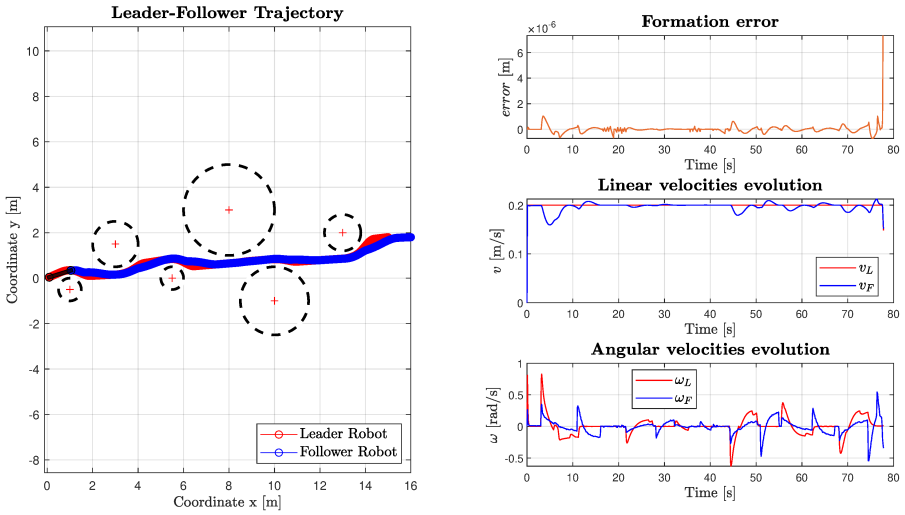

Figure 5.6: Simulation of map (f).

negligible in the context of the entire simulation.

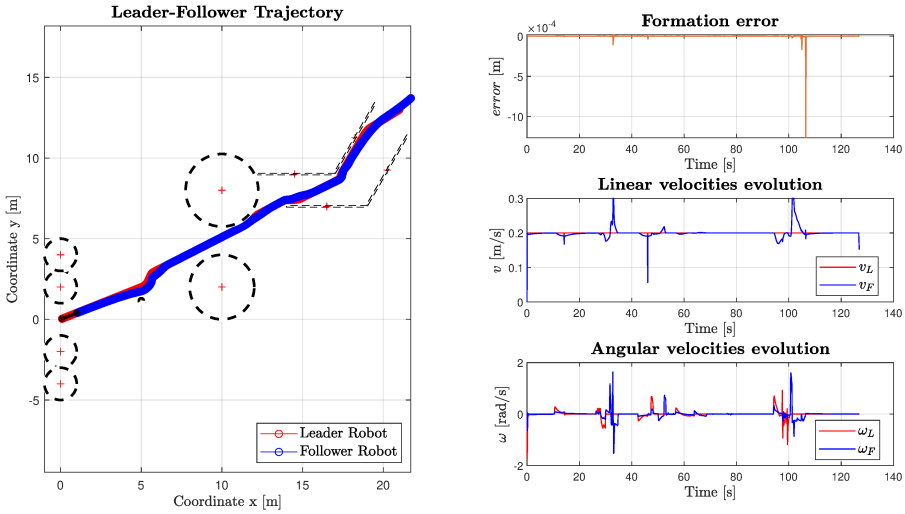

Figure 5.7: Simulation of map (g).

The map in Figure 5.7, inspired by the one in [8], offers the most realistic approximation of an actual operating environment. The system can navigate the corridor without significant difficulties, apart from a spike in formation error triggered by a sudden acceleration of the *Follower*. This occurs when the *Follower* attempts to keep the load from striking the corridor corner after the *Leader* has

already passed it. To adjust its path, both linear and angular velocities change sharply as soon as the robot detects that the load's center could collide with the corridor walls. After exiting the corridor, the MRS has successfully traversed the narrow gap between the two obstacles and has reliably detected and avoided the moving obstacle in the environment without any collision. This obstacle travels at 0.1 m/s, which enables the *Leader* to overtake it, as the robot is capable of moving at a higher speed.

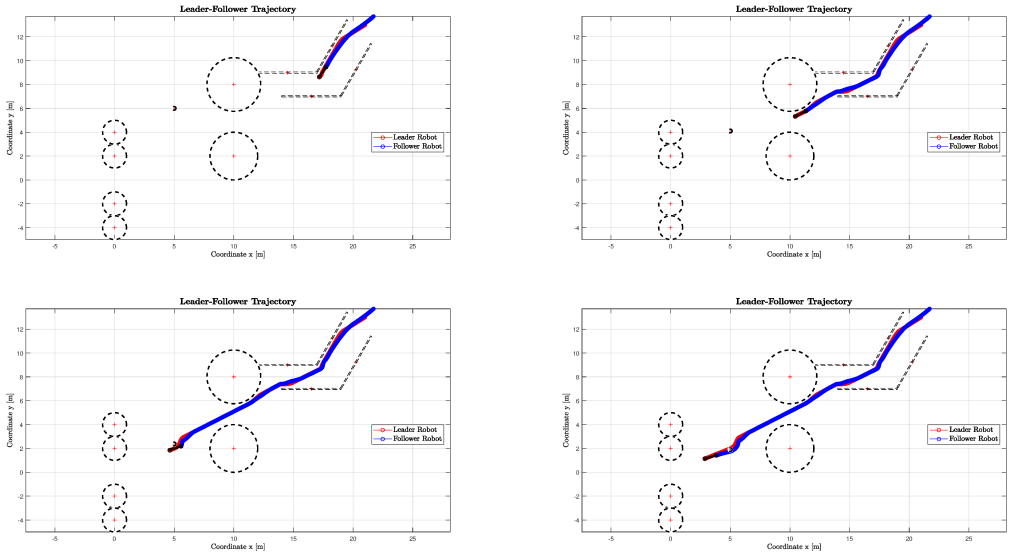

Figure 5.8: Trajectory of the system in the map **(g)**. Snapshots are taken at $t = 30s$, $t = 68s$, $t = 102.5s$ and $t = 112s$.

Finally, in Figure 5.9, we illustrate a configuration containing only moving obstacles. To better interpret the evolution of the simulation, we consider multiple time instants in the Figure 5.10. We observe that the *Leader* has successfully avoided all obstacles by rapidly adjusting its trajectory, while the *Follower* must repeatedly increase its speed to prevent any obstacle from impacting the payload. Overall, the results are impressive given the large number of moving obstacles, and achieving a maximum formation error of only a few millimeters represents an excellent milestone.

5.2 Comparison with the state of the art

In the previous section, through the use of the graphs together with Table 5.1, we have shown that the formation error remains very low in all the simulated

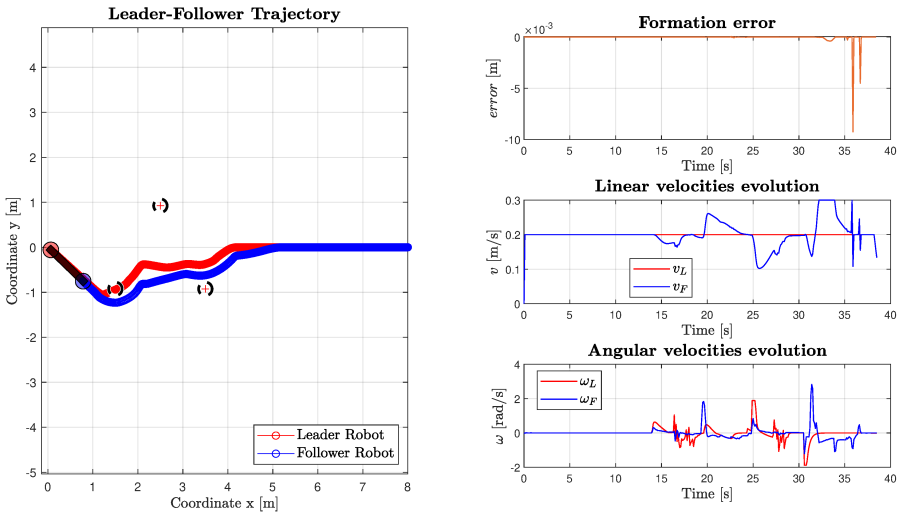

Figure 5.9: Simulation of map (h).

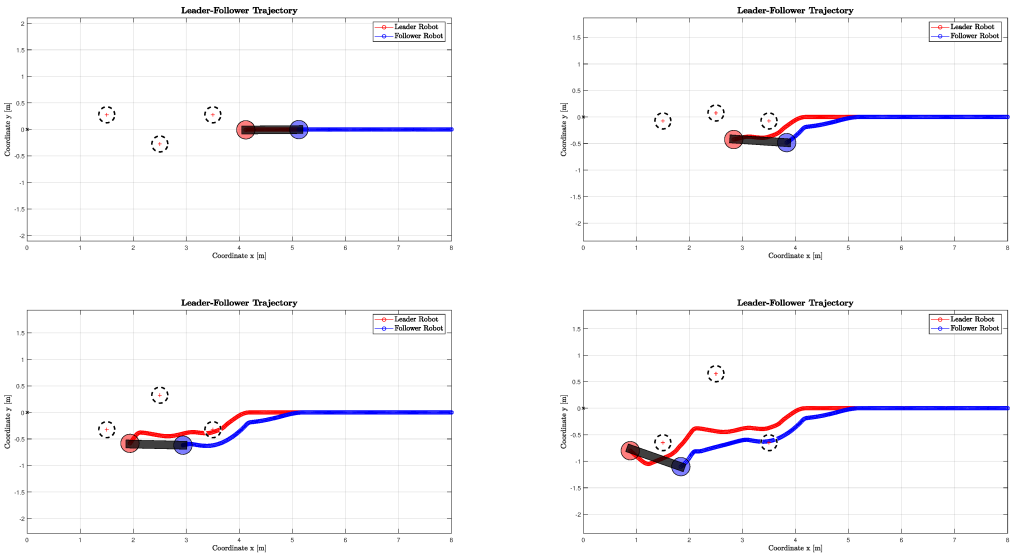

Figure 5.10: Trajectory of the system in the map **(h)**. Snapshots are taken at $t = 14.5s$, $t = 21.5s$, $t = 26.5s$ and $t = 33s$.

scenarios using our system. A natural follow-up question is whether the newly proposed system actually outperforms the original one, despite the differences in both implementation and robotic platforms.

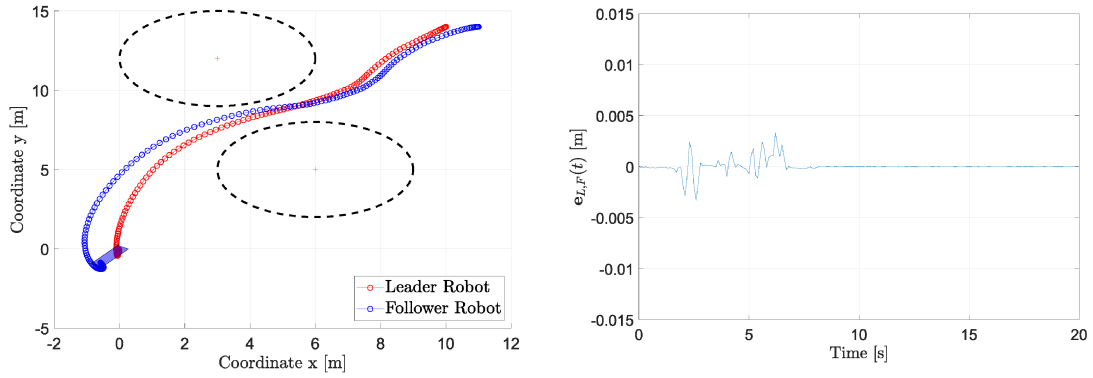

Figure 5.11: Trajectory and formation error evolution of the system defined in [8].

The first map on which we can perform meaningful evaluations is the Two obstacles scenario, simulated and shown by our system in Figure 5.4 and, for the state-of-the-art system, in Figure 5.11. Although some differences are present, the environment itself remains unchanged except for the scale, which differs because the robot speed in the previous system was 1 m/s, while in ours it is 0.2 m/s. Consequently, the new map, as well as the robots' trajectories, appears smaller.

A key difference between the two simulations is the configuration of the MRS when the *Leader* reaches the goal. In the first case, once the obstacles are passed, the *Leader* moves almost directly toward the goal, whereas in the second case its trajectory appears nearly parabolic. One plausible explanation is the distinct robot models: in [8] the robots follow a double-integrator model, and this simplification likely leads to the observed behavior of both the *Leader* and the *Follower*, the latter tracing an even more clearly defined parabola as it tries to follow the former. In contrast, in our system the robot is modeled by nonlinear equations that emulate a differential-drive platform, thereby enforcing physical constraints on the motion.

Regarding the results, in the previous system Figure 5.11/b shows a maximum formation error of approximately 0.035 m. In our case, as reported in Table 5.1, the maximum value for this map is 7.16×10^{-6} m, i.e., an error up to 4 orders of magnitude smaller. This is a significant improvement, and a likely reason for

this discrepancy is again the underlying robot model. With differential-drive robots, the heading cannot be changed instantaneously from one iteration to the next; the robot must first stop and then rotate in place. By contrast, robots with a double-integrator model are designed to emulate omnidirectional motion, allowing instantaneous changes of direction, but at the cost of increased instability of the overall MRS, as this example illustrates.

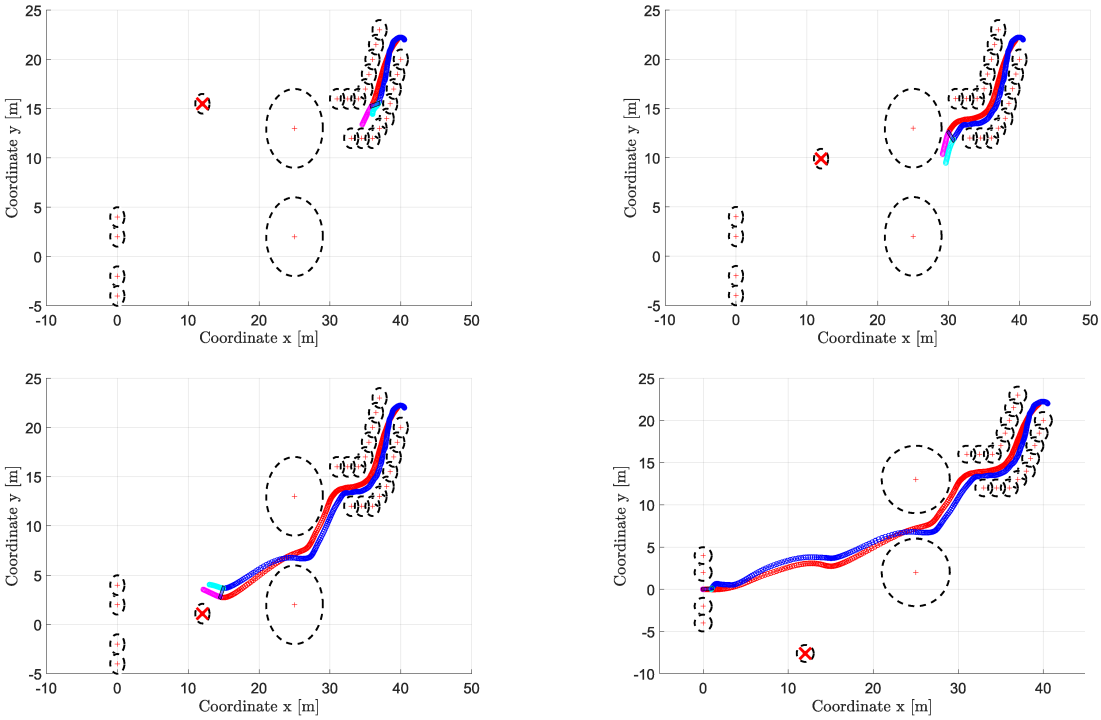

Figure 5.12: Trajectory of the system [8] in the Valzer map. Snapshots are taken at $t = 8s$, $t = 13.5s$, $t = 22.5s$ and $t = 30s$.

We can also compare the results obtained on the Valzer map, shown for our system in Figure 5.7 and for the reference system in Figure 5.12. As in the previous example, the maps differ, but the general principle is the same, and the behavior of the two systems is consistent with what was observed earlier. Both reach the goal and successfully avoid the moving obstacle, although the trajectories of the two MRSs are not identical. In this scenario, the maximum formation error of our system is 1.71×10^{-5} m, whereas for the second system it is approximately 5.2×10^{-3} m. Once again, the error of the new system is significantly lower than that of the reference one. Furthermore, Figure 5.13 shows that the error for the reference system fluctuates continuously during the

simulation, unlike in Figure 5.7, where—aside from a few spikes—the variation remains very limited.

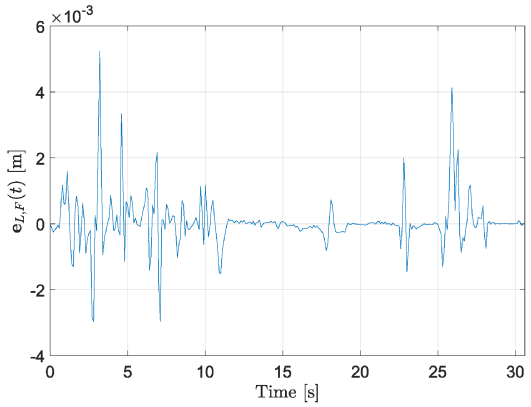

Figure 5.13: Formation error evolution of the system [8] in the Valzer map.

By changing the robot platform and introducing a small number of modifications to the overall architecture, we have achieved results that substantially outperform those of the reference system, even without the Feasibility-Aware Recovery Policy. This is a key result, as it demonstrates that we have advanced the state of the art by equipping the system with well-targeted, distinctive features.

5.3 Edge cases results

In this section, we show how introducing the Navigation Policy has increased the robustness and intelligence of the MRS, enabling it to manage specific situations that would otherwise have caused it to stall before reaching its objective. In all the environments analyzed, the policy was activated in order to keep the *Leader* from getting stuck, while still delivering excellent final performance across all simulations, as summarized in the table below.

Table 5.2: Simulation results on the edge cases environments

Scenario	Iters	Time [s]	MAX_e [m]	MEAN_e [m]	Policy used	Policy iters
Direct obstacle	285	108,88	7.29×10^{-6}	5.07×10^{-8}	1	61
Direct wall	312	119,40	2.79×10^{-5}	3.63×10^{-8}	1	83
Sharp bend	369	228,01	2.80×10^{-5}	-7.97×10^{-7}	1	44
Goal boxed	269	231,72	7.59×10^{-5}	-1.25×10^{-5}	2	172
Block left	269	445,73	9.04×10^{-5}	-2.24×10^{-5}	2	172
Block right	270	257,28	1.52×10^{-4}	-1.39×10^{-5}	2	173
Navigation	528	703.61	1.59×10^{-2}	-4.19×10^{-4}	2	345

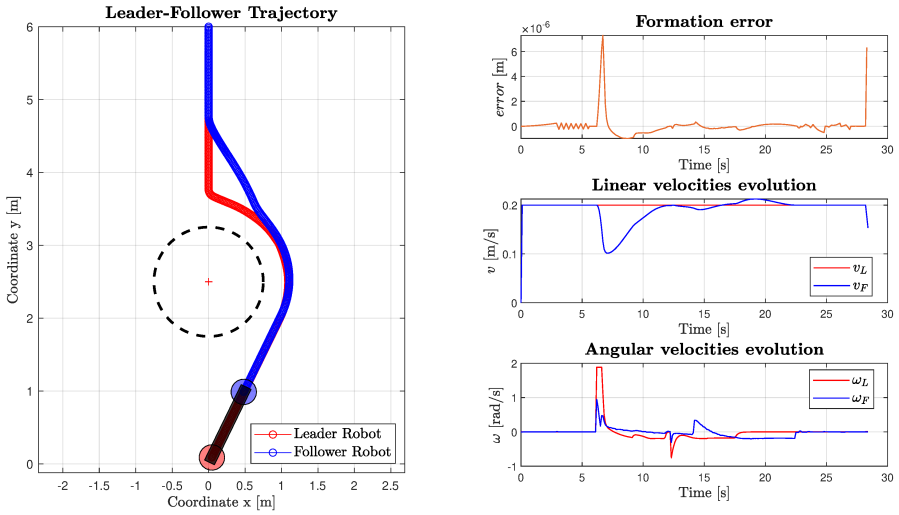

Figure 5.14: Simulation of map (i).

The first two maps we examine differ only in the type of obstacle: in Figure 5.14 a circular obstacle is present, whereas in Figure 5.15 it has been replaced by a rectangular wall. These configurations have been created during the implementation of the Navigation Policy and were subsequently used to validate its correct behavior with various obstacle types. In both scenarios, the policy enables the MRS to smoothly navigate around the obstacle and then reach the target. The second scenario is particularly noteworthy, because the wall's position causes the load to move very close to it when the *Leader* has already passed the obstacle and is heading toward the goal, increasing the likelihood of a collision. Nonetheless, the plots show that the *Follower*'s trajectory and velocities adapt significantly, introducing a deviation that allows the load to clear the wall without any contact.

A specific scenario is the simulation of a corridor with a 90-degree turn, illustrated in Figure 5.16. This map has been designed to analyze how the system behaves when the initial position is surrounded by obstacles on all sides, and to assess its ability to operate in this type of environment, which is particularly interesting. From the plot, it is evident that, from the outset, the *Leader* struggles to reach the goal because it lies beyond the bend in the corridor. Nevertheless, we can observe that the *Leader*, by following the wall closely, is able to reach the turn without difficulty, then rotate and proceed directly toward the goal. In this test, the *Follower* experiences difficulties only when the *Leader* changes heading after rounding the corner, as indicated by the peak in the error plot. Overall,

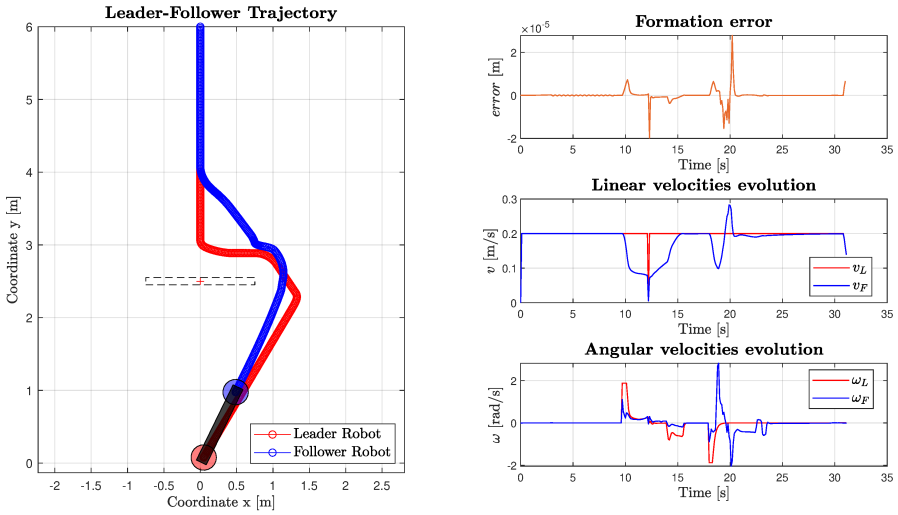

Figure 5.15: Simulation of map (ii).

however, the formation error stays very low, allowing us to conclude that the system can successfully navigate narrow, relatively simple corridors.

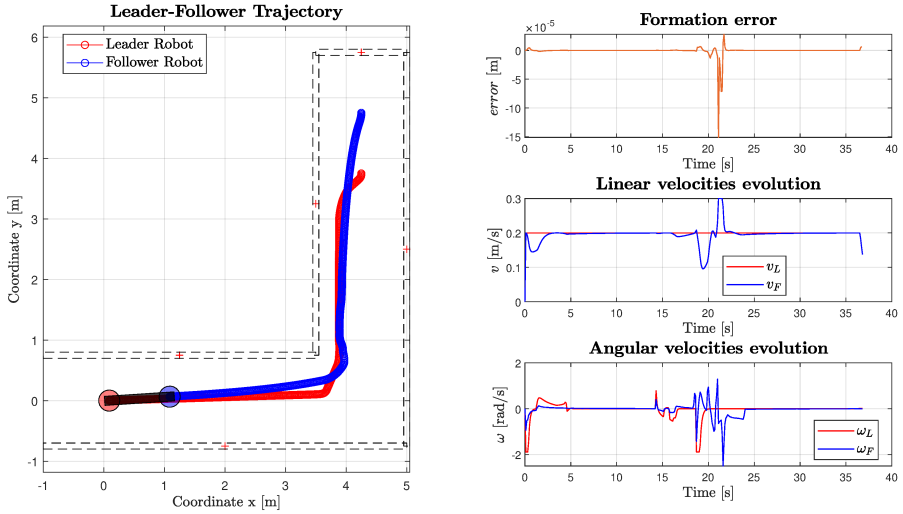

Figure 5.16: Simulation of map (iii).

One of the most specific situations that can arise occurs when the goal is surrounded by obstacles on all sides except one, forcing the MRS to move around them to reach its target. The system must be intelligent enough to detect this configuration and adapt its trajectory, preventing it from getting stuck in front of any obstacle. Figure 5.17 illustrates this scenario and shows how, thanks to the Navigation Policy, our system can determine the most suitable path to the

goal.

In this experiment, the policy is triggered twice; however, by inspecting the plot of the formation error, we observe two peaks that correspond to the moments when the policy is deactivated, not when it is activated. During the first activation, the *Follower* remains sufficiently distant from the obstacles, allowing it to replicate the *Leader*'s trajectory change without difficulty. When it enters the reset zone, it abruptly changes direction, which produces a peak in the formation error. This peak is also influenced by the *Follower*'s attempt to keep the load away from the wall edges to avoid collisions.

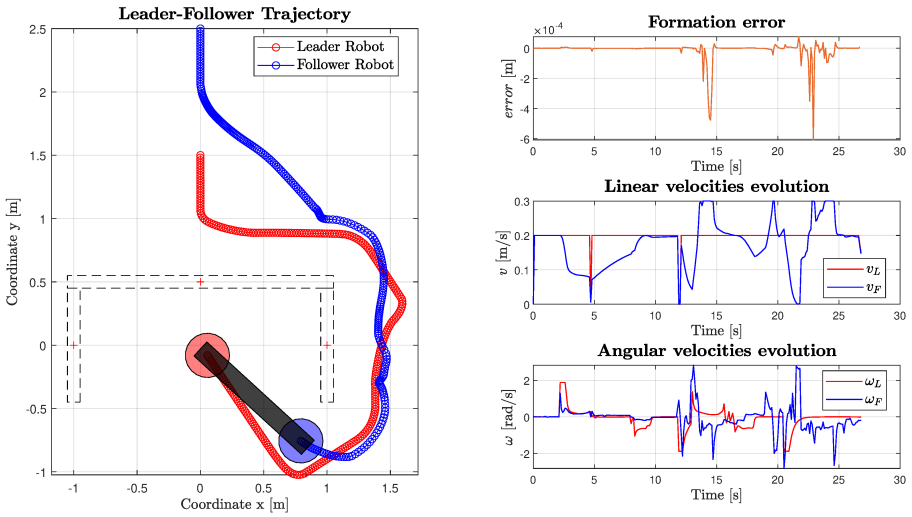

Figure 5.17: Simulation of map (iv).

A comparable behavior is observed during the second activation: once again, the peak in the formation error coincides with the end of the activation. Here, the configuration resembles the Wall in front of the goal scenario described earlier, where the load is positioned very close to the end of the wall, and the *Follower* actively tries to move it away.

From this base map, two additional maps have been created: they are identical to the original but include one extra obstacle to the left (Figure 5.18) and one to the right (Figure 5.19) of the initial obstacle. The introduction of these obstacles allows us to verify whether the Navigation Policy can, through its MPC-based prediction, select the most advantageous direction. As shown by the plots, the policy correctly detects the potential failure of its current prediction due to the new obstacle and consequently chooses the opposite direction, enabling the system to reach the goal as in the original map.

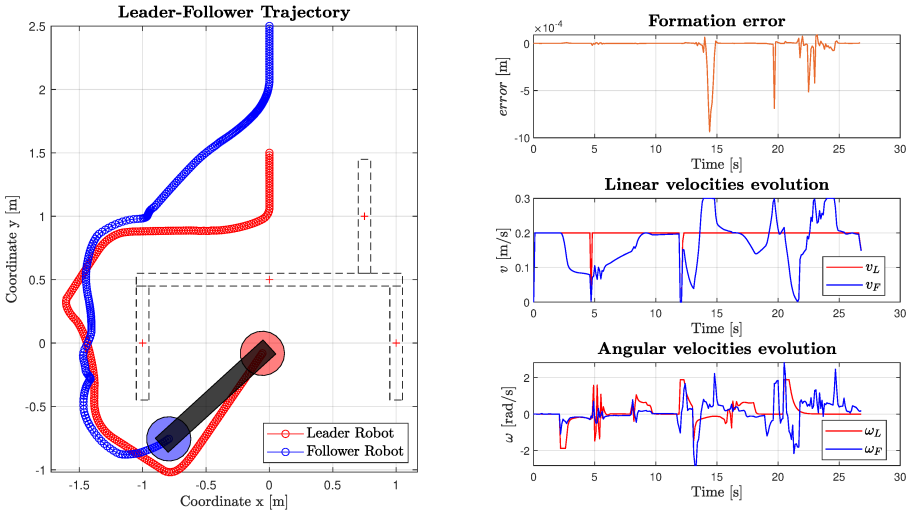

Figure 5.18: Simulation of map (v).

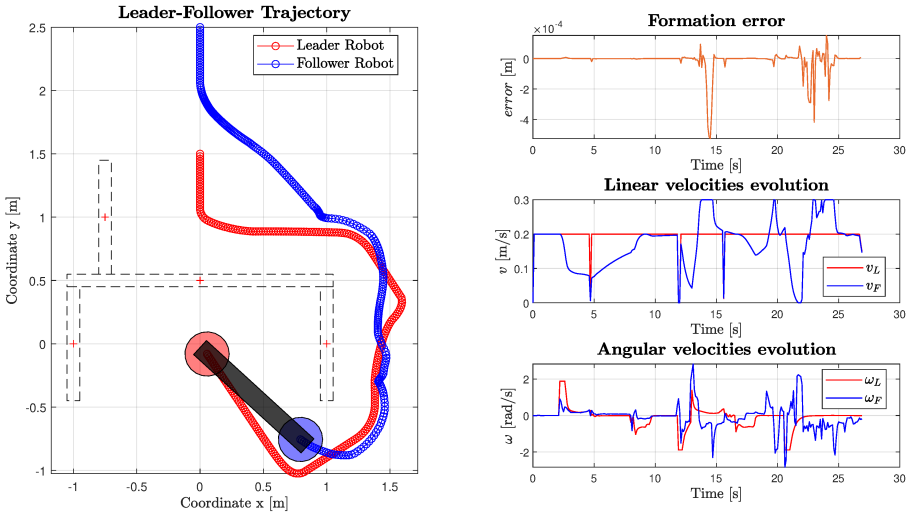

Figure 5.19: Simulation of map (vi).

The final map under consideration is the one in Figure 5.20, which represents the most complex scenario analyzed so far. This environment was purposefully constructed to evaluate the cascaded Navigation Policy, and therefore features very narrow turns that compel the MRS to identify paths it would be unable to find without this policy. The results show that the system successfully reaches the target without any collisions involving either the robots or the load. The only limitation is that, in this experiment, we recorded the highest formation error among all tests, approaching 2 centimeters. Fortunately, such an error margin would still not lead to a system failure.

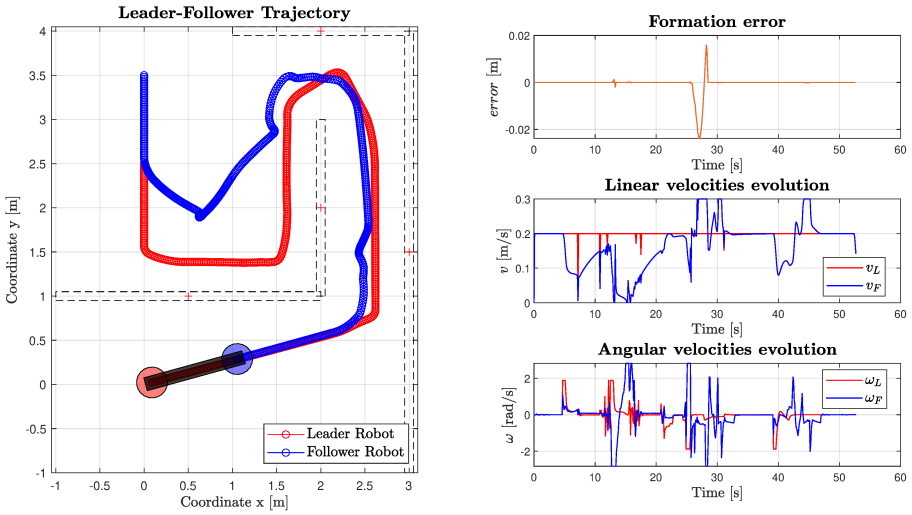

Figure 5.20: Simulation of map (vii).

5.4 System robustness with noise

As discussed in Section 3.6, considering a possible realistic implementation of the system and evaluating its robustness, we have introduced artificial noise into the system. The resulting test outcomes are highly encouraging: despite noise in both sensing and actuation, the system has consistently reached the target in every trial and has typically preserved a low formation error. The detailed results are presented in the table below:

Because the plots from the noisy system closely resemble those from the ideal system, we focus our discussion on the tests where the discrepancy between them is largest. In each figure, both scenarios are displayed to facilitate

Table 5.3: Simulation results with noise.

Scenario	Iters	Time [s]	MAX_e [mm]	MEAN_e [mm]	Policy used	Policy iters
No obstacles	223	89,24	1.15×10^{-3}	1.77×10^{-4}	0	0
One obstacle	273	198,42	1.34×10^{-3}	1.83×10^{-4}	0	0
Two obstacles	272	241,58	1.28×10^{-3}	1.93×10^{-4}	0	0
Three obstacles	277	285,82	1.43×10^{-3}	2.10×10^{-4}	0	0
Pathological	241	189,54	1.69×10^{-3}	1.98×10^{-4}	1	57
Six obstacles	788	809,55	1.56×10^{-3}	2.08×10^{-4}	0	0
Valzer	1286	1712,74	1.82×10^{-3}	1.81×10^{-4}	0	0
Moving obstacles	390	926,80	1.40×10^{-3}	1.21×10^{-4}	1	58
Direct obstacle	288	215,79	1.24×10^{-3}	2.24×10^{-4}	1	62
Direct wall	314	234,19	1.33×10^{-3}	1.78×10^{-4}	1	83
Elbow	375	817,58	1.15×10^{-3}	2.05×10^{-4}	1	43
Goal boxed	273	323,64	1.46×10^{-3}	1.73×10^{-4}	2	173
Block left	273	313,73	1.39×10^{-3}	1.51×10^{-4}	2	174
Block right	272	472,09	1.50×10^{-3}	1.70×10^{-4}	2	173

direct visual comparison.

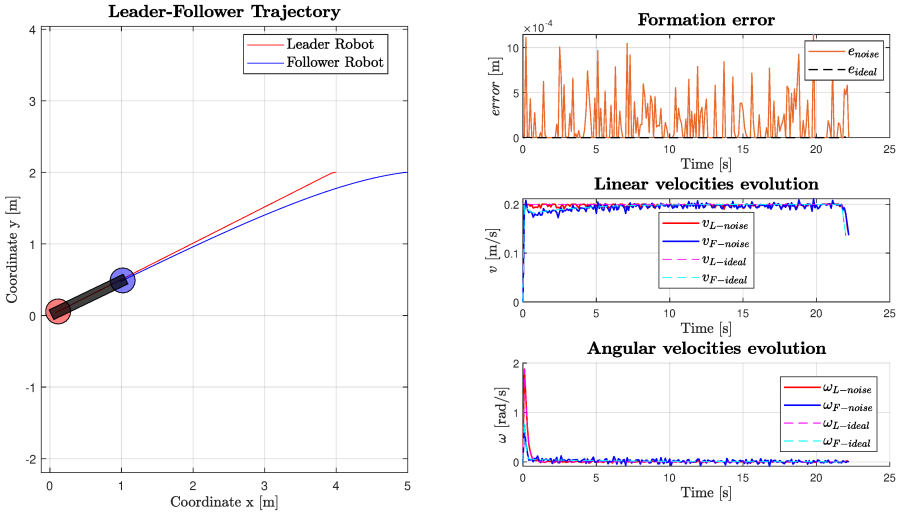

Figure 5.21: Simulation with noise of map (a).

We first consider the obstacle-free scenario, as it enables us to isolate the impact of actuation noise on the system. In Figure 5.21, the formation error in the noisy system exhibits a high variance, whereas in the ideal case it is nearly a flat line along the zero axis. This result is crucial because it confirms that introducing noise has a tangible effect on the system. Although the overall trajectories of the two robots appear similar in both cases, the linear and angular velocities in the noisy system fluctuate around the corresponding values in the ideal system. This effect arises because noise is injected directly into the robots' inputs; nonetheless, the system's robustness enables us to preserve an accurate

trajectory even when the inputs vary at every iteration.

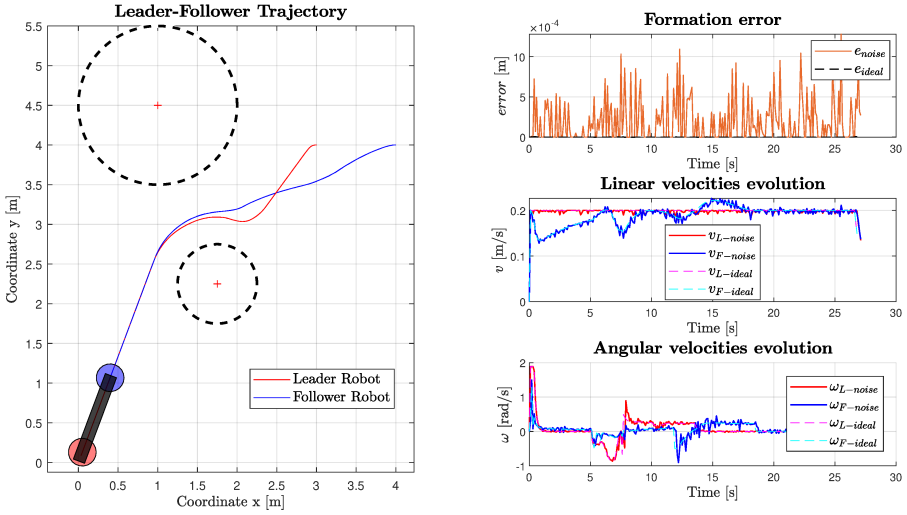

Figure 5.22: Simulation with noise of map (d).

To show that the system can still operate correctly when sensing is affected by noise, the environment **(d)** was selected as a first test case. As illustrated in Figure 5.22, the same distinctive behaviors observed in the No obstacles scenario reappear here. Despite the noisy measurements, the safety margins embedded in the obstacle-avoidance functions allow the system to pass through the bottleneck created by the two obstacles without any collisions involving either the robots or the load.

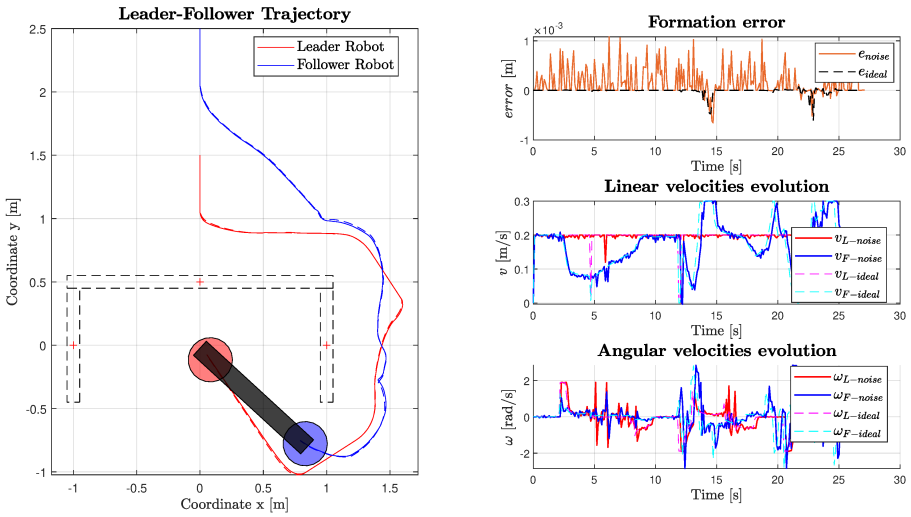

Figure 5.23: Simulation with noise of map (iv).

To verify instead that the Navigation Policy also works properly under noisy conditions, the map **(iv)** was used as a reference case. The main concern in the presence of noise is that the policy might fail to activate because a potential collapse is not correctly anticipated in the prediction. This concern arises from the fact that noise causes the predicted trajectory to shift slightly at each iteration—by a few millimetres—which could, in principle, delay the policy's activation. However, Figure 5.23 shows that the policy is triggered correctly on both occasions and that noise in the system does not hinder its operation. The noisy system effectively tracks the trajectory of the ideal system, and the evolution of the noise and velocities remains in line with the patterns previously observed.

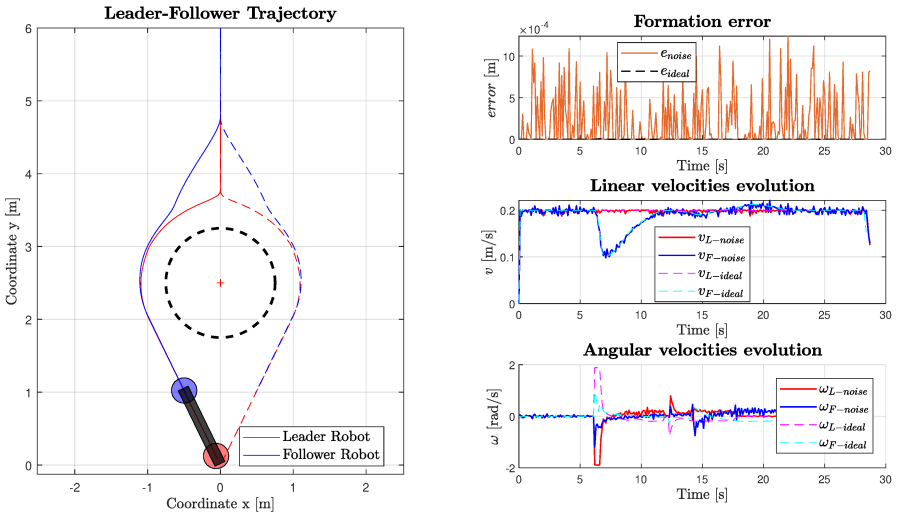

Figure 5.24: Simulation with noise of map (i).

To conclude with the practical examples, a curious feature has emerged in the experiments conducted in the maps **(i)** and **(ii)**. As illustrated in Figures 5.24 and 5.25, respectively, the trajectory of the noisy system is the mirror image of that of the ideal system. The introduction of noise has altered the *Leader*'s heading, making it differ from the ideal one; since, within the Navigation Policy, both possible directions are weighted equally, the policy selects the direction that extends the *Leader*'s current path. In this situation, we can observe how this choice profoundly modifies the evolution of the simulation. At the same time, even though the selected heading differs, the overall behavior of the system remains analogous, ultimately yielding mirrored trajectories. A noteworthy observation concerns the linear velocities, which—apart from fluctuations due

to noise—are nearly identical in both systems. By contrast, the angular velocities are reflected about the x-axis, as the two systems choose opposite directions.

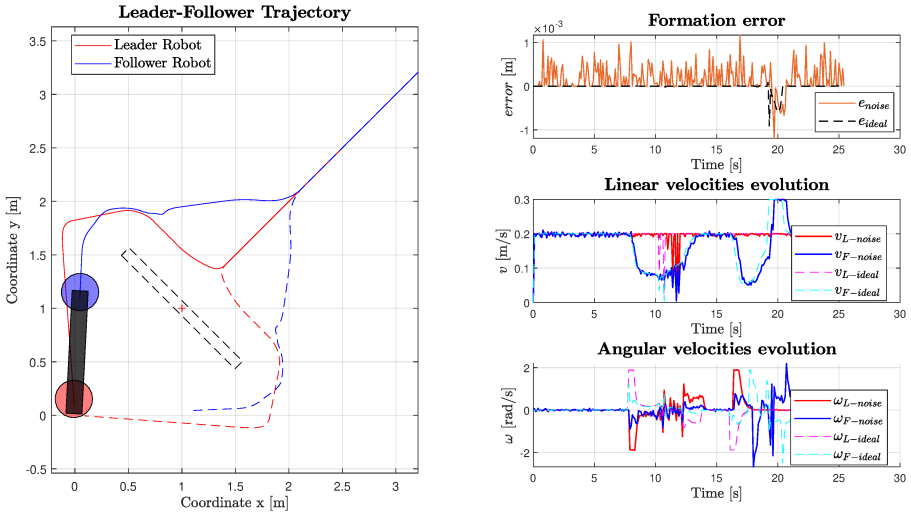

Figure 5.25: Simulation with noise of map (γ).

Chapter 6

Conclusions

Building on the project presented in ([8]) and taking it as the state of the art, we have set out to design an MRS that pursues the same objective as the original system. By employing differential-drive robots modeled as unicycles, we have enhanced both the sensitivity and robustness of the framework, obtaining very good results in multiple simulated scenarios and achieving low formation errors. This improvement has made it possible to eliminate the Feasibility-Aware Recovery Policy, which had been specifically introduced to reduce formation error. We have replaced it with a Navigation Policy, an algorithm that can temporarily modify the system's goal to prevent the MRS from getting stuck and to increase the robots' autonomy, enabling them to bypass most obstacles they encounter. Thanks to this policy, many edge cases have been successfully handled, substantially extending the system's capabilities. Additional experiments have been conducted by injecting artificial noise into the system to assess its robustness and its suitability for real-world deployment. These tests have also yielded very positive results, supporting the feasibility of an implementation with real robots.

Despite its strong potential, driven by the low formation error and the inclusion of the dedicated Navigation Policy, the project still has some limitations. The computation time per simulation varies between 60 and 300 seconds for simulated timed of 25 to 80 seconds. These performance levels are not yet sufficient for real-time applications, but they could be improved through several strategies: distributing the MPC computations across multiple machines to run

them in parallel, using a more powerful computer than the one adopted in this work, optimizing the code by generating MEX functions from the MATLAB implementation, or rewriting the entire project in C++ to achieve higher efficiency.

Another limitation concerns obstacle detection. A key assumption during development has been that all obstacles are known in advance; however, for deployment on real robots, LiDAR sensors would be required. We have attempted to integrate LiDARs into the system, but due to the way the obstacle-avoidance constraints were formulated, the MRS is unable to correctly reconstruct obstacle shapes and therefore tends to collide with them.

If this project is to be executed in real time on **TurtleBot4** robots, substantial work is still needed. Nevertheless, the results obtained so far are very encouraging and support the prospect of a future real-world implementation.

Appendix

Constraint matrices

$$A_{con} = \begin{bmatrix} q_{1x} - x(0) & q_{1y} - y(0) & 0 & 0 \\ q_{1x} - x(0) & q_{1y} - y(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ q_{2x} - x(0) & q_{2y} - y(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ q_{Mx} - x(0) & q_{My} - y(0) & 0 & 0 \end{bmatrix} \in \mathbb{R}^{ML \times n} \tag{6.1}$$

$$A_{bar} = \mathbf{I}_N \otimes A_{con} = \tag{6.2}$$

$$= \begin{bmatrix} \begin{bmatrix} q_{1x} - x(0) & q_{1y} - y(0) & 0 & 0 \\ q_{1x} - x(0) & q_{1y} - y(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ q_{2x} - x(0) & q_{2y} - y(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ q_{Mx} - x(0) & q_{My} - y(0) & 0 & 0 \end{bmatrix} & & \mathbf{0}_{ML \times n} \\ & \ddots & \\ \mathbf{0}_{ML \times n} & & \begin{bmatrix} q_{1x} - x(0) & q_{1y} - y(0) & 0 & 0 \\ q_{1x} - x(0) & q_{1y} - y(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ q_{2x} - x(0) & q_{2y} - y(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ q_{Mx} - x(0) & q_{My} - y(0) & 0 & 0 \end{bmatrix} \end{bmatrix} \in \mathbb{R}^{NML \times nN} \tag{6.3}$$

$$A_{vel} = \mathbf{I}_N \otimes \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} = \mathbf{I}_N \otimes V_l = \begin{bmatrix} V_l & & \cdots & \mathbf{0}_{2v\times n} \\ \vdots & \ddots & & \vdots \\ \mathbf{0}_{2v\times n} & \cdots & & V_l \end{bmatrix} \in \mathbb{R}^{2vN\times nN} \tag{6.4}$$

$$G_{in} = \mathbf{I}_N \otimes \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} = \mathbf{I}_N \otimes U_l = \begin{bmatrix} U_l & & \cdots & \mathbf{0}_{2m\times m} \\ \vdots & \ddots & & \vdots \\ \mathbf{0}_{2m\times m} & \cdots & & U_l \end{bmatrix} \in \mathbb{R}^{2mN\times mN} \tag{6.5}$$

$$b_{con} = \begin{bmatrix} \begin{bmatrix} q_{1x} - x(0) & q_{1y} - y(0) \end{bmatrix} \begin{bmatrix} q_{1x} - v_{1x} \\ q_{1y} - v_{1y} \end{bmatrix} \\ \begin{bmatrix} q_{1x} - x(0) & q_{1y} - y(0) \end{bmatrix} \begin{bmatrix} q_{1x} - v_{2x} \\ q_{1y} - v_{2y} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} q_{2x} - x(0) & q_{2y} - y(0) \end{bmatrix} \begin{bmatrix} q_{2x} - v_{1x} \\ q_{2y} - v_{1y} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} q_{Mx} - x(0) & q_{My} - y(0) \end{bmatrix} \begin{bmatrix} q_{Mx} - v_{Lx} \\ q_{My} - v_{Ly} \end{bmatrix} \end{bmatrix} \tag{6.6}$$

$$= \begin{bmatrix} \left[q_{1x} - x(0)\right]\left[q_{1x} - v_{1x}\right] + \left[q_{1y} - y(0)\right]\left[q_{1y} - v_{1y}\right] \\ \left[q_{1x} - x(0)\right]\left[q_{1x} - v_{2x}\right] + \left[q_{1y} - y(0)\right]\left[q_{1y} - v_{2y}\right] \\ \vdots \\ \left[q_{2x} - x(0)\right]\left[q_{2x} - v_{1x}\right] + \left[q_{2y} - y(0)\right]\left[q_{2y} - v_{1y}\right] \\ \vdots \\ \left[q_{Mx} - x(0)\right]\left[q_{Mx} - v_{Lx}\right] + \left[q_{My} - y(0)\right]\left[q_{My} - v_{Ly}\right] \end{bmatrix} \in \mathbb{R}^{ML\times 1} \tag{6.7}$$

$$B_{bar} = \text{repmat}\,(b_{con}, N, 1) = \begin{bmatrix} \left[q_{1x} - x(0)\right]\left[q_{1x} - v_{1x}\right] + \left[q_{1y} - y(0)\right]\left[q_{1y} - v_{1y}\right] \\ \vdots \\ \left[q_{Mx} - x(0)\right]\left[q_{Mx} - v_{Lx}\right] + \left[q_{My} - y(0)\right]\left[q_{My} - v_{Ly}\right] \\ \left[q_{1x} - x(0)\right]\left[q_{1x} - v_{1x}\right] + \left[q_{1y} - y(0)\right]\left[q_{1y} - v_{1y}\right] \\ \vdots \\ \left[q_{Mx} - x(0)\right]\left[q_{Mx} - v_{Lx}\right] + \left[q_{My} - y(0)\right]\left[q_{My} - v_{Ly}\right] \\ \vdots \end{bmatrix} \in \mathbb{R}^{NML \times 1} \tag{6.8}$$

$$B_{in} = \begin{bmatrix} u_{\max} \\ u_{\max} \\ u_{\max} \\ u_{\max} \\ \vdots \\ u_{\max} \\ u_{\max} \\ u_{\max} \\ u_{\max} \\ \vdots \end{bmatrix} \in \mathbb{R}^{2mN \times 1} \quad B_{vel} = \begin{bmatrix} v_{\max} \\ v_{\max} \\ v_{\max} \\ v_{\max} \\ v_{\max} \\ v_{\max} \\ \vdots \\ v_{\max} \\ v_{\max} \\ v_{\max} \\ v_{\max} \\ \vdots \end{bmatrix} \in \mathbb{R}^{2vN \times 1} \tag{6.9}$$

Bibliography

[1] Javier Alonso-Mora, Stuart Baker, and Daniela Rus. "Multi-robot formation control and object transport in dynamic environments via constrained optimization". In: *The International Journal of Robotics Research* 36.9 (2017), pp. 1000–1021. DOI: 10.1177/0278364917719333. URL: https://doi.org/10.1177/0278364917719333.

[2] Prathamesh Barawkar et al. "Ant-Inspired Strategies for Multi-Robot Collaborative Transportation—An Ockham's Razor". In: *IFAC-PapersOnLine* 54.20 (2024), pp. 231–236. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2021.11.040.

[3] Johan Berglund, Gabriel Kappenman, Alvaro Sánchez, et al. "Automated Guided Vehicles in Material Handling: A Review". In: *Procedia CIRP* 119 (2023). Open Access, pp. 101–106. ISSN: 2212-8271. DOI: 10.1016/j.procir.2023.05.016.

[4] Andrea Bozzi et al. "Nonlinear Model Predictive Control for Wheeled Mobile Robots in Indoor Settings". In: *IEEE Robotics and Automation Letters* 8.2 (2023), pp. 789–796. ISSN: 2377-3774. DOI: 10.1109/LRA.2023.3234567.

[5] Guanpu Chen et al. "Simultaneous Trajectory Planning and Tracking Using an MPC Method for Cyber-Physical Systems: A Case Study of Obstacle Avoidance for an Intelligent Vehicle". In: *IEEE Transactions on Industrial Informatics* 14.9 (2018), pp. 4273–4283. ISSN: 1551-3203. DOI: 10.1109/TII.2018.2817392.

[6] Jianing Chen et al. "Occlusion-Based Cooperative Transport with a Swarm of Miniature Mobile Robots". In: *IEEE Transactions on Robotics* 31.2 (2015), pp. 307–321. DOI: 10.1109/TRO.2015.2400731.

[7] Dries Derinck et al. “Real-Time Path Planning and Collision Avoidance for Autonomous Mobile Robots and AGVs in Dynamic Environments”. In: *IEEE Access* 8 (2020). Open Access, pp. 217845–217860. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3042618.

[8] Badr Elaamery et al. “Model Predictive Control for Cooperative Transportation with Feasibility-Aware Policy”. In: *Robotics* 10.3 (2021). ISSN: 2218-6581. DOI: 10.3390/robotics10030084. URL: https://www.mdpi.com/2218-6581/10/3/84.

[9] Alessando Farinelli, Luca Iocchi, and Daniele Nardi. “Multirobot Systems: A Classification Focused on Coordination”. In: *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society* 34 (Nov. 2004), pp. 2015–28. DOI: 10.1109/TSMCB.2004.832155.

[10] Giuseppe Fragapane et al. “Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda”. In: *European Journal of Operational Research* 295.2 (2021), pp. 405–426. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2021.01.019.

[11] Uwe Jahn et al. “A Taxonomy for Mobile Robots: Types, Applications, Capabilities, Implementations, Requirements, and Challenges”. In: *Robotics* 9.4 (2020). ISSN: 2218-6581. DOI: 10.3390/robotics9040109. URL: https://www.mdpi.com/2218-6581/9/4/109.

[12] Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. “A Framework for Multi-Robot Cooperative Exploration”. In: *International Journal of Advanced Robotic Systems* 9.5 (2012). Open access, p. 178. ISSN: 1729-8806. DOI: 10.5772/45692.

[13] Pramod Kumbhar, Shweta Jadhav, et al. “Automated Guided Vehicles in Healthcare and Agriculture: Applications and Challenges”. In: *International Journal of Engineering Research & Technology* 10.5 (2021). Open Access, pp. 123–130. ISSN: 2278-0181. DOI: 10.17577/IJERTV10IS050234.

[14] Nicola Lissandrini, Giacomo Michieletto, and Angelo Cenedese. “Model Predictive Control for Cooperative Transportation with Feasibility-Aware Policy”. In: *Robotics* 10.3 (2021). Open Access, p. 84. ISSN: 2218-6581. DOI: 10.3390/robotics10030084.

[15] Daniel Mellinger et al. "Cooperative Grasping and Transport Using Multiple Quadrotors". In: *Distributed Autonomous Robotic Systems: The 10th International Symposium*. Ed. by Alcherio Martinoli et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 545–558. ISBN: 978-3-642-32723-0. DOI: 10.1007/978-3-642-32723-0_39. URL: https://doi.org/10.1007/978-3-642-32723-0_39.

[16] Yara Rizk, Mariette Awad, and Edward W. Tunstel. "Cooperative Heterogeneous Multi-Robot Systems: A Survey". In: *ACM Computing Surveys* 52.2 (2019). Article No. 29, 29:1–29:31. ISSN: 0360-0300. DOI: 10.1145/3303848.

[17] Kazuki Shibata et al. *Learning Locally, Communicating Globally: Reinforcement Learning of Multi-robot Task Allocation for Cooperative Transport*. 2022. arXiv: 2212.02692 [cs.RO]. URL: https://arxiv.org/abs/2212.02692.

[18] Pengfei Song and Vijay Kumar. "A Potential Field Based Approach to Multi-Robot Manipulation". In: *IEEE Transactions on Robotics* 39.1 (2023). Adapted for linear MPC in UGVs context, pp. 456–467. ISSN: 1552-3098. DOI: 10.1109/TRO.2022.3214567.

[19] Iris F. A. Vis. "Survey of research in the design and control of automated guided vehicle systems". In: *European Journal of Operational Research* 170.3 (2006), pp. 677–709. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2004.09.004.

[20] Heng Xie, Kaixu Dong, and Pakpong Chirarattananon. "Cooperative Transport of a Suspended Payload via Two Aerial Robots With Inertial Sensing". In: *IEEE Access* 10 (2022), pp. 81764–81776. DOI: 10.1109/ACCESS.2022.3194932.

[21] Ning Xu et al. "Interval Trajectory Tracking for AGV Based on MPC". In: *Procedia CIRP* 83 (2019). Focus on stochastic elements in AGV scheduling, pp. 513–518. ISSN: 2212-8271. DOI: 10.1016/j.procir.2019.03.088.

[22] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. "A Survey and Analysis of Multi-Robot Coordination". In: *International Journal of Advanced Robotic Systems* 10.12 (2013), p. 399. DOI: 10.5772/57313. URL: https://doi.org/10.5772/57313.

[23] Heng Zhang et al. "Cooperative Transportation With Mobile Manipulator: A Capability Map-Based Framework for Physical Human–Robot Collaboration". In: *IEEE/ASME Transactions on Mechatronics* 27.6 (2022), pp. 4396–4405. DOI: 10.1109/TMECH.2022.3155601.

[24] Houchun Zheng et al. "Longitudinal and Lateral Control of Autonomous Vehicles in Multi-Vehicle Driving Environments". In: *IEEE Transactions on Intelligent Vehicles* 3.2 (2018), pp. 213–226. ISSN: 2379-8858. DOI: 10.1109/TIV.2018.2821153.