UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI TECNICA E GESTIONE DEI SISTEMI INDUSTRIALI

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA MECCATRONICA

A Variable Prediction Horizon MPC Approach for Leader-Follower Transportation in Presence of Obstacles

Relatore
Prof. Michieletto Giulia

Correlatore
Prof. Antonello Riccardo
Dott. Bertoni Massimiliano

Laureando
Civiero Matteo
Matr. 2087500

ANNO ACCADEMICO 2023-2024

Data di laurea 3/12/2024

Un ringraziamento speciale a tutte le persone che mi sono sempre state accanto, che hanno condiviso e che sono sicuro continueranno a condividere con me esperienze e passioni.
Ringrazio la relatrice, Giulia Michieletto, e il correlatore, Riccardo Antonello, per avermi supportato in questo progetto.
Ringrazio Massimiliano e i migliori compagni di tesi di sempre per la disponibilità e l'aiuto nei momenti più critici.
"Il meno è fatto".

Abstract

Cooperative transportation using a Multi-Robot System has emerged as a relevant research topic in recent years, primarily due to its potential in challenging workspaces, including transporting large payloads in cluttered environments. In this case, an obstacle avoidance feature is essential to guarantee the safe navigation of both agents and payload.

This thesis addresses the problem of cooperative transportation in environments populated by static and dynamic obstacles by implementing a feasibility-aware leader-follower Model Predictive Control based algorithm. Compared to the existing state-of-art, a more realistic model of agents dynamics is considered. A geometrical approach is employed to define the hard constraints for obstacle avoidance, while soft constraints are represented by a potential repulsive field functional cost, which acts to guide the agents away from the obstacles.

The results achieved through numerical simulations in MATLAB® show that a variable prediction horizon MPC can guarantee improved performance in trajectory planning with a reduced computational effort, compared to a fixed prediction horizon MPC. Algorithm robustness to disturbance and delay is also assessed, showing that it depends both on MPC parameters and obstacle placement. Lastly, Gazebo simulations are performed to include physics and implement more realistic experiments, showing that this algorithm is able to fulfill its tasks successfully.

Contents

List of Figures

Chapter 1

Introduction to Multi-Robot System transportation

1.1 Literature review

A Multi-Robot System (MRS) consists on a group of robots that can execute tasks in the same workspace, or even work on the same task. To successfully cooperate, they need to interact with each other, such as exchanging information collected by onboard sensors about the environment or about their status. Cooperative MRS has become an important research topic, because of its large variety of useful applications. The focus of this work is *cooperative transportation*, that implements the transportation of an object to a target position by planning a feasible path for the agents. Agents have to keep a certain distance between each other to accomplish this task and to avoid payload damaging. To evaluate this ability, formation error, or else the difference between ideal formation distance and actual agents distance, is considered. The trajectory computation takes into account different criteria for each robot and depends on the composition of the MRS, the agents' hierarchy and the used algorithm.

1.1.1 Multi-Robot Systems

Most of the MRS studied in literature use Unmanned Aerial Vehicles (UAVs) or Unmanned Ground Vehicles (UGVs), as they are the most versatile and useful agents for military, civil and industrial purposes. In particular, UAVs are mainly used in combination with cables to grasp the payload [1], [2], [3], [4], [5], rather than manipulators that are more commonly used with UGVs [6]. Moreover, UAVs are preferred for spatial problems due to their high mobility, while UGVs usage depends on the type of manipulators or end-effectors mounted on agents. Mixed agent MRS has also been considered [7], aiming to analyze more generic spatial problems with increased complexity, due to actuated manipulators equipped on both agents, that guarantee a flexible and efficient system, even in cluttered and complex environments.

Tasks division in a MRS can be carried out with different approaches. *Leader-follower* approach is probably the most common, where one agent is the leader and has to compute the trajectory to move the payload to the target, while the remaining agents are followers and they can have different strategies to fulfill their task. For instance, in [6], [8], [9] a precise distance between the agents has to be maintained. In [7], [1], [2] forces on payload are minimized, to avoid stretches or damages on it. These objectives are implemented as functional costs to minimize. In addition, other kind of optimization can be introduced, such as energy consumption [3], [9], time spent moving [3] and so on. *Feasibility-aware*, or else *recovery*, policy is an interesting *leader-follower* feature developed in [6]. This policy is actuated when formation error is higher than a certain limit and it slows down the leader to let followers recover their position in formation.

1.1.2 Model Predictive Control approach

Model Predictive Control (MPC) is probably the control architecture that best suits the cooperative transportation problem. The goal of MPC is to optimize a cost function that dictates the behaviour of the agents, ensuring the solution satisfies some hard constraints. These constraints are defined as equations, inequalities, or boundaries involving variables such as agent state or input. MPC facilitates trajectory planning and computes control inputs. It also seamlessly incorporates both hard and soft constraints. Specifically, soft constraints are handled by adding them as terms in the cost function.
A generic MPC problem consists on a functional cost $J(\mathbf{x}(\cdot), \mathbf{u}(\cdot))$ to optimize in a prediction horizon N, defined as a set of N following time steps, while satisfying some constraints, such as model dynamics, state and input constraints:

$$\mathbf{u_{opt}} = argmin_{\mathbf{u}(\cdot)}\big(J\left(\mathbf{x}(\cdot), \mathbf{u}(\cdot)\right)\big) \tag{1.1}$$

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) \qquad (model\ dynamics) \tag{1.2}$$

$$\mathbf{x}(k) \in \mathcal{X} \qquad (state\ constraints) \tag{1.3}$$

$$\mathbf{u}(k) \in \mathcal{U} \qquad (input\ constraints) \tag{1.4}$$

$$k \in \mathbb{Z},\ k \in [0, N] \qquad (time) \tag{1.5}$$

The MPC problem has different forms, as it is shown in Table 1.1, and, according to its properties, using a precise MPC scheme presents different pros and cons. Talking about the MPC formulation, *linear MPC* clearly uses a simplified or linearized model, that can cause approximation problems, and both functional costs and constraints are linear, so usually easy to optimize. On the other hand, *non-linear MPC* is tougher to implement, having to manage non-linear functions and inequalities, but it is more precise. Moreover, solution modality defines the scalability of the architecture. In a *centralized MPC* a single agent computes trajectories for all the agents. This approach can be really efficient but has to deal with scalability and robustness problems. In *decentralized MPC* schemes, often based on a *leader-follower* approach, every single agent executes the algorithm, producing a scalable and robust system.

Table 1.1: Characterization of MPC scheme.

MPC formulation	Linear MPC [6], [8], [5]	Non-Linear MPC [7], [1], [2], [10], [11], [9]
Solution modality	Centralized [1], [2], [5]	Decentralized [6], [7], [8], [11], [9]

1.1.3 Obstacle avoidance policy

Obstacle avoidance is another key component of this project, because it represents the MPC constraints used in environments where various obstacles can be placed. A large amount of approaches have been formulated and tested in literature. First of all, there are two main categories of constraints: hard constraints and soft constraints (Table 1.2). Hard constraints can represent a set of non feasible solutions and are incorporated to the algorithm as equations or inequalities related to the optimization variable. Soft constraints are usually written as functional costs, such as barrier function or repulsive function, and are added to the path-generation functional costs.

The geometrical approach [6] is the one used and analyzed in this thesis. Each agent path, which is predicted throughout the horizon, satisfies the constraints when the agent shape, in

each position of this path, completely lays on a convex space. Simple minimum distance between agents and obstacles defines the constraints in [1], [2], [9].
Soft constraints are used [7] to implement obstacle avoidance too, performing a *potential repulsive field* that guide the agents away from the obstacles.
Otherwise, input, velocity and state constraints are always set as hard constraints, such as boundaries.

Table 1.2: Obstacle avoidance approach.

Hard constraints	Soft constraints	Mixed constraints
[6], [1], [2], [8], [11], [9]	[7]	[10]

A particularly interesting constraint formulation is presented in [10], where a *Chance Constrained MPC* is developed, considering the agent-agent and agent-obstacle collision probability to define hard constraints and two functional costs that produce a penalty and a barrier function. The result is a smoother path produced by the algorithm and a reduced computational effort.
Finally, the algorithm presented in [8] is able to control more than one formation of robots working in the same area. It implements collision avoidance between robots in the same formation and between different formations, allowing a great scalability and the transportation of more payloads at the same time.

Linear MPC Framework for a 2 degrees of freedom MRS (2-DOF LMPC) exposed in [6] results to be the most interesting work to study and improve. In fact, the agents model used is a planar double integrator, way simpler than other papers that considered spatial problems [7] and cable tension models [1], [2]. Moreover, it developed a more intuitive obstacle avoidance constraints definition with respect to [10], [8] and definitely more interesting than [1], [2], [9]. Lastly, this paper presents excellent cooperative transportation performance, such as the ability to avoid collisions, reach the target and keep the formation error bounded below few centimeters.

1.2 Contribution

In this thesis, the *leader-follower feasibility-aware* 2-DOF LMPC controlling two UGVs with an actuated manipulator presented in [6] is analyzed and enhanced. The improvements include incorporating agent heading into the model, which necessitates transitioning from linear MPC

to non-linear MPC, and implementing a variable prediction horizon to optimize computational efficiency. So, a Non-Linear MPC Framework for a 3 degrees of freedom MRS (3-DOF NLMPC) is developed based on the 2-DOF LMPC framework.
More precisely, linear constraints used in geometrical approach are converted to non-linear constraints, so that they are compatible with the new 3 degrees of freedom (DOF) model. After that, *potential repulsive field* functional cost similar to [7], for both agents, and *control effort* functional cost, for the follower, are added to the functions used in 2-DOF LMPC.

In literature, every paper considered a fixed prediction horizon and analyzed which could be the optimal length, depending on the environment and the algorithm used. It is intuitive that the longer the prediction horizon is, the bigger the computational cost for each algorithm iteration will be. Moreover, the latter increases with the number of obstacles in the workspace. The novel idea to improve the performance of this system is to implement a *state machine*, where the MPC uses a longer prediction horizon N when the MRS is far from the obstacles, so that obstacle avoidance can be ignored, while it uses a much shorter prediction horizon when obstacle avoidance is necessary to prevent collisions. Obstacles position in the environment needs to be fully known or sensed in well advance. State machine ideally optimizes computational effort and keeps the same behaviour of the system in [6]. By reducing computation when unnecessary and using a wider prediction horizon to anticipate the best path, it balances efficiency and performance. MATLAB® simulations comparing fixed and variable prediction horizons validate this intuition. The robustness to disturbance of the system is also tested and the algorithm proves to be effective in presence of a reasonably sized noise.
After that, Simulink® simulations are performed to assess the behaviour of the algorithm with respect to the instrinsic input delay due to computation. These tests show that an input delay significantly affects the effectiveness of this algorithm. Lastly, Gazebo simulations introduce real-world physics and delays due to ROS2™ communication between Simulink® and Gazebo environment, confirming the same behaviour proven in Simulink®.

1.3 Thesis structure

In Chapter 2 all the assumptions and MPC architecture used in [6] are reported. Chapter 3 details the enhancements made to the system. Results of numerical simulations conducted in MATLAB® are shown in Chapter 4. Chapter 5 reports results obtained in Simulink® and Gazebo simulations. Chapter 6 discusses the results obtained and suggests future work.

1.4 Notation

In this work, calligraphic letters define sets, for instance $\mathcal{V}$ is a set of vertices. Scalars in $\mathbb{R}$ are represented with lowercase letters, such as x. Vectors in $\mathbb{R}^k$ are written in bold lowercase letters, such as $\mathbf{u} \in \mathbb{R}^2$. Matrices in $\mathbb{R}^{k\times h}$ are written in uppercase letters, for example $A_c \in \mathbb{R}^{4\times 4}$, $k, h \in \mathbb{N}$. Superscripts indicate that a variable is referred to a specific element in the environment, such as leader, follower or load. For example f^* is referred to agent $* = L, F, load$. Symbol $\otimes$ represents the Kronecker product between two matrices.

Chapter 2

Linear Model Predictive Control Framework for a 2 degrees of freedom Multi-Robot System: description and analysis

In this chapter, an analysis of the 2-DOF LMPC framework used in [6] is made, to point out some details that are essential for implementing the more complex 3-DOF NLMPC framework, which will be further developed in Chapter 3.

The objective of this 2-DOF LMPC framework is to control a leader-follower MRS composed of two UGVs transporting a bar payload to the goal in a 2-D cluttered environment, with static and dynamic obstacles, ensuring the minimization of the formation error, or else maintaining the inter-agent distance required to successfully carry the payload, and enabling obstacle avoidance. A recovery policy is incorporated to improve formation error performance in particularly cluttered environments.

2.1 Application scenario assumptions

2-DOF LMPC framework is based on assumptions presented in the following.
The considered MRS accounts for two UGVs, each of them composed of a mobile base and a manipulator. Note that the manipulator can be simplified into a fixed joint between the agent and the payload, since the problem is planar and the bar is assumed to be at a constant height. The mobile base is equipped with omnidirectional wheels (Figure 2.1).

These two robots have to grasp and transport a bar payload in a 2-D environment where $M \geq 0$ circular obstacles are placed. The latter can be either static or dynamic with a uniform rectilinear motion.
It is also assumed that:

- agents are assumed to already be grasping the payload, so the grasping task is not considered.
- the agent is assumed to be controlled by a low-level controller, acting on all DOF, that applies agent input computed by the MPC to the real robot. Basically, it converts agent accelerations into wheels motor commands.
- the grasping between the end-effector and the load is compliant, so if a formation error between the agents occurs, a wrench, proportional to the formation error, is applied to the load. To avoid damaging the payload, one of the goal in this application scenario is to minimize the formation error.
- the end-effector is placed at the center of the agent, so formation error is measured as the distance between agents center.
- agents have direct access to exact obstacles position and to the other agent state. These variables are referenced to a global inertial frame $\mathcal{F}_w$, placed in the target position at $(0, 0)$ for simplicity, without loss of generality.

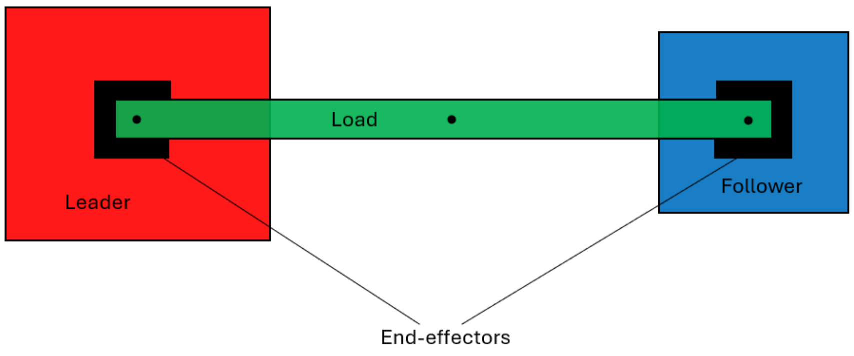


Figure 2.1: MRS scheme used in this thesis.

2.1.1 Agents model

The dynamics of both UGVs composing the considered MRS is described by a double integrator model. In general, each agent state is defined as $\mathbf{x}^* \in \mathbb{R}^n$ and incorporates agent position $\mathbf{p}^* \in \mathbb{R}^2$ and velocity $\mathbf{w}^* \in \mathbb{R}^v$, where n is the state dimension and $v = \frac{n}{2}$ is the velocity vector dimension. Agent input is $\mathbf{u}^* \in \mathbb{R}^m$, where m is the input dimension, and it represents agent acceleration. $* = L, F$ defines if it refers respectively to leader or follower.

In this 2-DOF model, state vector is defined as

$$\mathbf{x}^*(t) = \begin{bmatrix} x^*(t) & y^*(t) & v_x^*(t) & v_y^*(t) \end{bmatrix}^T \in \mathbb{R}^4 \tag{2.1}$$

$$\mathbf{p}^*(t) = \begin{bmatrix} x^*(t) & y^*(t) \end{bmatrix}^T \in \mathbb{R}^2 \qquad \mathbf{w}^*(t) = \begin{bmatrix} v_x^*(t) & v_y^*(t) \end{bmatrix}^T \in \mathbb{R}^2 \tag{2.2}$$

$\mathbf{p}^*(t)$ is the position and $\mathbf{w}^*(t)$ is the velocity vector of the agent, both expressed in the global inertial frame $\mathcal{F}_w$ (world frame). $t \in \mathbb{R}$ is a generic time step.
In addition, according to the double integrator model, the control input of the $*$ agent corresponds to the linear acceleration of the agent along the x- and y-axes of the world frame, namely

$$\mathbf{u}^*(t) = \begin{bmatrix} u_x^*(t) & u_y^*(t) \end{bmatrix}^T \in \mathbb{R}^2 \tag{2.3}$$

The continuous-time model of both leader and follower agent dynamics can be described as:

$$\dot{\mathbf{x}}^*(t) = A_c \mathbf{x}^*(t) + B_c \mathbf{u}^*(t) \tag{2.4}$$

$$\mathbf{y}^*(t) = \mathbf{x}^*(t) \tag{2.5}$$

where the system matrices are:

$$A_c = \begin{bmatrix} 0_{2\times2} & I_{2\times2} \\ 0_{2\times2} & 0_{2\times2} \end{bmatrix} \in \mathbb{R}^{4\times4} \qquad B_c = \begin{bmatrix} 0_{2\times2} \\ I_{2\times2} \end{bmatrix} \in \mathbb{R}^{4\times2} \tag{2.6}$$

This continuous model can be discretized with a sampling time T_s, resulting in the discrete-time representation:

$$\mathbf{x}^*(s+1) = A\mathbf{x}^*(s) + B\mathbf{u}^*(s) \tag{2.7}$$

$$\mathbf{y}^*(s) = \mathbf{x}^*(s) \tag{2.8}$$

where $s = \lfloor \frac{t}{T_s} \rfloor$ is the discrete time step, with t being the continuous time and $s \in \mathbb{Z}$.

The sample time chosen for this algorithm is $T_s = 0.1\ s$, as this value was also used in [6], [7] and it has proven to be a compatible value with the computational effort required by the 3-DOF NLMPC framework in most tested environments (Chapter 4). In general, a lower T_s can guarantee more frequent path updates, considering that mobile obstacles are present. Thus, a trade-off between update rate and algorithm execution time is necessary.

2.1.2 Obstacles model

Obstacles are modeled as circles on the plane, with their centers represented as $\mathbf{o}_i = \begin{bmatrix} o_{x_i} & o_{y_i} \end{bmatrix}^T \in \mathbb{R}^2$ and radii $r_i \in \mathbb{R}$, where $i = 1 \dots M$ and $M \in \mathbb{N}$ is the total number of obstacles in the environment. Obstacles can move in the workspace with a constant velocity defined by $\mathbf{v}_i^{obs} = \begin{bmatrix} v_{x_i}^{obs} & v_{y_i}^{obs} \end{bmatrix}^T \in \mathbb{R}^2$.
At the beginning of the simulation, obstacles are placed in the environment, and their positions are updated at the end of each time step based on their velocities. Thus, agents consider obstacles to be stationary during each MPC iteration and across the entire prediction horizon. This because agents have complete access to obstacles position but they cannot predict their trajectory.

2.2 Linear Model Predictive Control approach

To face the cooperative transportation problem in the described application scenario, the authors of [6] propose a linear MPC with an incorporated recovery policy and a geometrical obstacle avoidance constraints approach.

The generic MPC problem is re-proposed and adapted to this specific application. It consists on a functional cost $J^*(\mathbf{x}_h^*, \mathbf{u}_k^*)$ to optimize in a prediction horizon N, defined as a set of N following time steps, with respect to the optimization variable $\mathbf{u}_k^*$, while satisfying some constraints, such as model dynamics, state and input constraints:

$$\mathbf{u}_\mathbf{k}^* = argmin_{\mathbf{u}_k^*}\left(J^*\left(\mathbf{x}_h^*, \mathbf{u}_k^*\right)\right) \tag{2.9}$$

$$\mathbf{x}^*(k+1) = A\mathbf{x}^*(k) + B\mathbf{u}^*(k) \qquad (model\ dynamics) \tag{2.10}$$

$$\mathbf{x}^*(h) \in \mathcal{X} \qquad (state\ constraints) \tag{2.11}$$

$$\mathbf{u}^*(k) \in \mathcal{U} \qquad (input\ constraints) \tag{2.12}$$

(2.13)

The optimization variable of this algorithm is the stacked vector containing optimal inputs of the $* = L, F$ agent over the entire prediction horizon, expressed in world frame, defined as:

$$\mathbf{u}_k^* = \begin{bmatrix} u_x^*(0) & u_y^*(0) & \dots & u_x^*(N-1) & u_y^*(N-1) \end{bmatrix}^T \in \mathbb{R}^{mN} \tag{2.14}$$

The index $k = 0 \dots (N-1)$ indicates that these inputs correspond to the current time step ($k = 0$) and the subsequent $N-1$ steps, where $N \in \mathbb{N}$ represents the length of the prediction horizon.
$\mathbf{x}^*(0) \in \mathbb{R}^n$ is the state of the agent at current time step. Then, the stacked vector containing the predicted state of the $* = L, F$ agent, in world frame, over the entire prediction horizon in which $\mathbf{u}_k^*$ is applied, is:

$$\mathbf{x}_h^* = \begin{bmatrix} x^*(1) & y^*(1) & v_x^*(1) & v_y^*(1) & \dots & x^*(N) & y^*(N) & v_x^*(N) & v_y^*(N) \end{bmatrix}^T \in \mathbb{R}^{nN} \tag{2.15}$$

$h = 1 \dots N$ represents the index of the state throughout the prediction horizon.
Model dynamics of the agents over the prediction horizon is regulated by the equation

$$\mathbf{x}_h = \bar{T}\mathbf{x}(0) + \bar{S}\mathbf{u}_k \tag{2.16}$$

where:

$$\bar{S} := \begin{bmatrix} \bar{S_1} \\ \vdots \\ \bar{S_N} \end{bmatrix} = \begin{bmatrix} B & 0_{n\times m} & \dots & 0_{n\times m} \\ AB & B & \dots & 0_{n\times m} \\ \vdots & \vdots & \vdots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix} \in \mathbb{R}^{nN\times mN} \tag{2.17}$$

$$\bar{T} := \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} \in \mathbb{R}^{nN\times n} \tag{2.18}$$

In Figure 2.2, the control scheme of this system is illustrated. The control architecture is composed of two MPC controllers, one for each agent: a Leader MPC and a Follower MPC.

- Leader MPC receives as inputs the target position $\mathbf{p}_{goal}^L = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$, its current state $\mathbf{x}^L$, and obstacles information, namely their center positions and radii, alongside the coordinates of the point on each obstacle surface nearest to the leader center, and the distance between these points. Using this information, the Leader MPC computes the agent acceleration inputs over the prediction horizon, $\mathbf{u}_k^L$, to generate an optimal trajectory. Only the first acceleration input $\mathbf{u}^L$ is applied by the low-level controller.

- Follower MPC takes as inputs the predicted states of the leader over the prediction horizon $\mathbf{x}_h^L$, its current state $\mathbf{x}^F$, and obstacles information referenced to the follower center. Similar to the Leader MPC, the Follower MPC computes follower acceleration inputs over the prediction horizon $\mathbf{u}_k^F$ and applies only the first input $\mathbf{u}^F$ using the low-level controller.

After the inputs computation a "recovery policy check" is performed receiving agents predicted states $\mathbf{x}_h^L$ and $\mathbf{x}_h^F$ as inputs. If recovery policy actuation criteria is satisfied, input computation is repeated with leader recovery function J_{rec}^L replacing standard leader functional cost J^L. Recovery policy is defined in Section 2.2.3.
When the MPC has finished its computation and agent accelerations are given as output, ideal low-level controllers apply $\mathbf{u}^*$ to the real robots. The agents state in the next step $\mathbf{x}^*(t+1)$ are then measured, assuming ideal measurements, and used as inputs for the next iteration of the MPC algorithm.
This means that MPC elaborates online the agent input that the real system needs to apply through low-level controllers. Since low-level controllers and measurement system are assumed to be ideal, $\mathbf{x}^*(t+1)$ perfectly follows the dynamics described by equation (2.7).

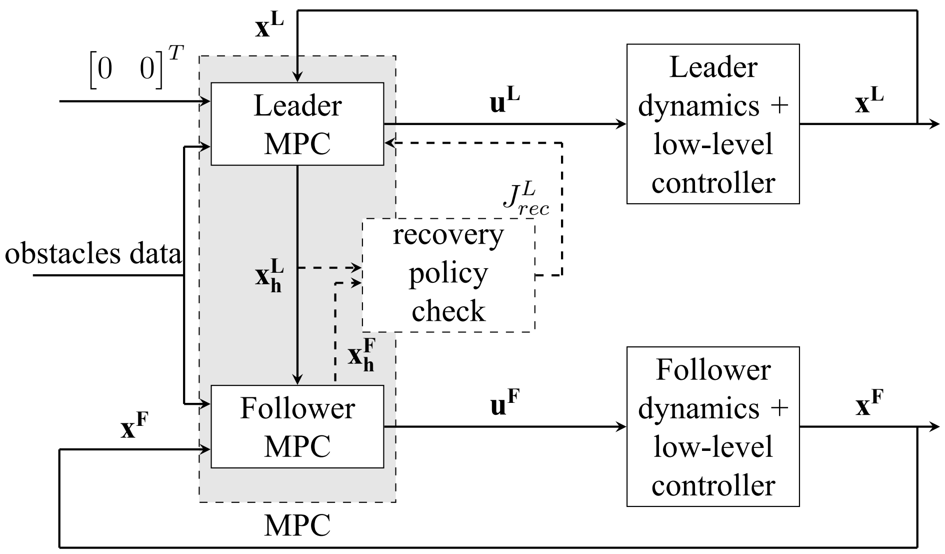


Figure 2.2: 2-DOF LMPC architecture.

These MPC controllers are different for leader and follower in terms of functional cost, because the aim of the leader is to plan the best path to reach the goal, while the follower needs to keep a fixed distance to the leader to correctly transport the load, moving on the shortest path possible. In addition, both agents need to avoid obstacles, so obstacle avoidance for both Leader and Follower MPC is provided by hard constraints.

2.2.1 Leader functional cost

Leader MPC objective is to plan the best path to reach the goal. As a consequence, leader functional cost is a quadratic cost function defined as below:

$$J^L\left(\mathbf{x}^L(k),\mathbf{u}^L(k)\right) = \sum_{k=0}^{N-1}\left((\mathbf{x}^L(k))^T W\mathbf{x}^L(k) + (\mathbf{u}^L(k))^T R_L\mathbf{u}^L(k)\right) + (\mathbf{x}^L(N))^T Z\mathbf{x}^L(N)$$
$$W \in \mathbb{R}^{n\times n} \quad R_L \in \mathbb{R}^{m\times m} \quad Z \in \mathbb{R}^{n\times n} \tag{2.19}$$

- $(\mathbf{x}^L(k))^T W\mathbf{x}^L(k)$ represents leader state cost throughout the prediction horizon, that weights the distance of leader state to the goal state $\begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}^T$, so position $(0,0)$ with null velocities, and W is its weight matrix.
- $(\mathbf{u}^L(k))^T R_L\mathbf{u}^L(k)$ represents leader control effort throughout the prediction horizon, that is used to reduce the energy spent to move the leader, and R_L is its weight matrix.
- $(\mathbf{x}^L(N))^T Z\mathbf{x}^L(N)$ is leader state cost of the last prediction horizon step and Z is its weight matrix.

The expression can be simplified [6]:

$$J^L(\mathbf{u}^L_k) = J^L_1(\mathbf{u}^L_k) = \frac{1}{2}(\mathbf{u}^L_k)^T H\mathbf{u}^L_k + \mathbf{x}^L(0)F\mathbf{u}^L_k + \frac{1}{2}(\mathbf{x}^L(0))^T Y\mathbf{x}^L(0) \tag{2.20}$$
$$H := 2\left((I_N \otimes R_L) + \bar{S}^T\bar{W}\bar{S}\right) \in \mathbb{R}^{mN\times mN} \tag{2.21}$$
$$\bar{W} := \begin{bmatrix} I_{N-1}\otimes W & 0_{n(N-1)\times n} \\ 0_{n\times n(N-1)} & Z \end{bmatrix} \in \mathbb{R}^{nN\times nN} \tag{2.22}$$
$$F := 2\bar{T}\bar{W}\bar{S} \in \mathbb{R}^{n\times mN} \tag{2.23}$$
$$Y := 2(W + \bar{T}^T\bar{W}\bar{T}) \in \mathbb{R}^{n\times n} \tag{2.24}$$

Since the term $(\mathbf{x}^L(0))^T Y\mathbf{x}^L(0)$ in equation (2.20) is independent to the optimization variable, it can be omitted.

2.2.2 Follower functional cost

Follower agent has to navigate through the environment keeping a certain distance to the leader to avoid damaging the payload, moving on the shortest path possible. For this reason Follower

MPC functional cost can be decomposed in two components, each corresponding to a specific task that the agent is required to accomplish:

1. Firstly, the squared formation error is defined as:

$$e_{L,F} = \left|\left|\mathbf{p}^L(t) - \mathbf{p}^F(t)\right|\right|^2 - d_{L,F}^2 \tag{2.25}$$

   where $d_{L,F}$ is the correct distance leader and follower center should have, or else the length of payload.
   The first component J_1^F accounts for the minimization of the squared formation error, so that follower agent keeps the correct distance $d_{L,F}$ with respect to the leader, in order to successfully transport the payload:

$$J_1^F(e_{L,F}(h)) = \sum_{h=1}^{N} \beta^h e_{L,F}^2(h) \tag{2.26}$$

   $\beta \in (0, 1] \subset \mathbb{R}$ is a decaying weight factor that prioritizes the minimization of squared formation error during the initial steps of the prediction horizon. This is particularly relevant since only the first predicted input is applied at each iteration.

2. The second component J_2^F minimizes the travelled distance over the prediction horizon, to avoid excessive movements:

$$J_2^F(\mathbf{p}^F(k)) = \sum_{k=0}^{N-1} \left|\left|\mathbf{p}^F(k+1) - \mathbf{p}^F(k)\right|\right|^2 \tag{2.27}$$

Thus, the total functional cost for the follower is:

$$J^F = CJ_1^F + J_2^F \tag{2.28}$$

where $C \in \mathbb{R}$ is the squared formation error weight. A higher value of C reduces the formation error.

This functional cost needs to be expressed in a different way, so that optimization variable

$\mathbf{u}_k^F$ is explicit.

$$J_1^F\left(\mathbf{u}_k^F\right) = \sum_{k=1}^{N} \beta^k \left(\left|\left| P\left(\mathbf{x}^L(k) - \bar{T}_h \mathbf{x}^F(0) - \bar{S}_h \mathbf{u}_k^F\right) \right|\right|^2 - d_{L,F}^2 \right)^2 \tag{2.29}$$

$$J_2^F(\mathbf{u}_k^F) = \left(\bar{S}' \mathbf{u}_k^F \bar{T}' \mathbf{x}^F(0)\right)^T \bar{P}^T \bar{P} \left(\bar{S}' \mathbf{u}_k^F \bar{T}' \mathbf{x}^F(0)\right) \tag{2.30}$$

$$\bar{S}' := \begin{bmatrix} B & 0_{n\times m} & \dots & 0_{n\times m} \\ (A - I_n)B & B & \dots & 0_{n\times m} \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-2}(A - I_n)B & A^{N-3}(A - I_n)B & \dots & B \end{bmatrix} \in \mathbb{R}^{nN\times mN} \tag{2.31}$$

$$\bar{T}' := \begin{bmatrix} A - I_n \\ A(A - I_n) \\ \vdots \\ A^{N-1}(A - I_n) \end{bmatrix} \in \mathbb{R}^{nN\times n} \tag{2.32}$$

$$\bar{P} := I_N \otimes P \in \mathbb{R}^{mN\times nN} \tag{2.33}$$

In equation (2.29) $\bar{T}_h \in \mathbb{R}^{n\times n}$ and $\bar{S}_h \in \mathbb{R}^{n\times mN}$ are the h-th row group, each composed of n rows, of matrices introduced in equations (2.18) and (2.17), respectively. As a matter of fact, the equation $\bar{T}_h \mathbf{x}^F(0) + \bar{S}_h \mathbf{u}_k^F = \mathbf{x}^F(h)$ selects the predicted state at time step h of the prediction horizon, namely the h-th block of n rows of the vector $\mathbf{x}_h^F$, according to equation (2.16).

2.2.3 Recovery policy

The decentralized nature of this algorithm means that the leader could plan a trajectory that is infeasible for the load or the follower, potentially causing system failure. To prevent this, a recovery policy can be implemented. Formation error is defined as

$$fe_{L,F}(h) = ||\mathbf{p}^L(h) - \mathbf{p}^F(h)|| - d_{L,F} \tag{2.34}$$

with $\mathbf{p}^L(h)$ and $\mathbf{p}^F(h)$ predicted positions computed by leader and follower standard functional costs in a certain algorithm iteration, with $h = 1 \dots N$. If the absolute value of formation error $|fe_{L,F}(h)|$ exceeds a certain value, ϵ_{loose_grip}, for at least one of the first k_{loose_grip} steps in the prediction horizon, both agents input are recomputed with a modified leader functional cost that slows the leader down, allowing the follower to recover and reduce the formation error. In other words, recovery policy is actuated if standard inputs computation results to predict a system path with an excessive formation error.
However, this approach could still fail due to really strict obstacle avoidance constraints and an

incorrect value of the formation error weight C, for example when the system encounters an extremely narrow passage between obstacles. In such cases, the leader will slow down, but if leader manages to pass through the narrow passage while the load or follower is blocked, the formation error will diverge.
The new leader functional cost used during recovery is:

$$J^L_{rec}(\mathbf{p}^L(k)) = J^L_{rec,1}(\mathbf{p}^L(k)) = \sum_{k=0}^{N-1} \left|\left|\mathbf{p}^L(k+1) - \mathbf{p}^L(k)\right|\right|^2 \tag{2.35}$$

This cost effectively slows down the leader, reducing the space navigated at each step throughout the prediction horizon.

2.2.4 Linear leader and follower constraints

An analysis of the linear constraints used in [6] is essential for understanding this approach and its non-linear extension described in Section 3.3.4.

To analyze obstacle avoidance constraints, it is necessary to introduce additional variables that describe the agents shape and the interaction between agents and obstacles.
M is the number of obstacles in the workspace. The vector

$$\mathbf{q}^*_i = \begin{bmatrix} q^*_{i_x} & q^*_{i_y} \end{bmatrix}^T \in \mathbb{R}^2 \tag{2.36}$$

represents the coordinates of the point on the surface of the i-th obstacle, in world frame, that is the nearest to the agent $* = L, F$ at step 0 of the current prediction horizon. $i = 1 \dots M$ is the obstacle index. Obstacles, and so $\mathbf{q}^*_i$, are considered static by the agents at each MPC iteration. Each robot has a set of vertices $\mathcal{V}^*$ with cardinality $L^* \in \mathbb{N}$, $* = L,F$. For both the leader and the follower, the location of each vertex $\mathbf{v}^*_j$ is expressed in the agent body frame $\mathcal{F}_B$, a reference system centered in the agent center of mass:

$$\mathbf{v}^*_j = \begin{bmatrix} v^*_{j_x} & v^*_{j_y} \end{bmatrix}^T \in \mathbb{R}^2 \tag{2.37}$$

Since in this 2-DOF LMPC framework a fixed joint is considered between agents and load, at the beginning of each MPC iteration vertices $\mathbf{v}^*_j$ are rotated by θ^{load}, or else the angle corresponding to payload rotation. θ^{load} directly depends on agents current center position and is defined as:

$$\theta^{load}(0) = atan2\big((y^F(0) - y^L(0)), (x^F(0) - x^L(0))\big) \tag{2.38}$$

$\theta^{load} = 0$ when both agents center lay on the same y coordinate and follower stands on leader right.
Load vertices are not considered in these constraints.
This vertices model implies that the position of the j-th vertex in world frame at step t is $\mathbf{p}^*(t) + \mathbf{v}_j^*$.

To simplify the notation, hereafter, the superscripts are omitted since these matrices are the same for both leader and follower.
Constraints are divided in obstacle avoidance constraints and agents acceleration and velocity boundaries. Obstacle avoidance constraints build a safe area where leader and follower can move over the prediction horizon, making sure their vertices do not collide with obstacles by computing a set of infeasible positions. Agents acceleration and velocity constraints apply saturation on acceleration, and consequently on control force, and limit agents dynamic. Constraints are defined by the following inequality:

$$G\mathbf{u}_k \leq W + S\mathbf{x}(0) \tag{2.39}$$

G, W and S are composed of blocks with a precise meaning, that are made explicit in the following

$$\begin{bmatrix} A_{bar}\bar{S} \\ G_{in} \\ A_{vel}\bar{S} \end{bmatrix} \mathbf{u}_k \leq \begin{bmatrix} B_{bar} \\ B_{in} \\ B_{vel\_constr} \end{bmatrix} + \begin{bmatrix} -(A_{bar}\bar{T}) \\ 0_{2vN\times n} \\ -(A_{vel}\bar{T}) \end{bmatrix} \mathbf{x}(0) \tag{2.40}$$

A separate analysis for each block provides a clearer understanding of how the constraints are constructed. All matrices used to build these constraints are presented in Appendix .1.

1. The first row block of equation (2.40) is

$$A_{bar}\bar{S}\mathbf{u}_k \leq B_{bar} - A_{bar}\bar{T}\mathbf{x}(0) \tag{2.41}$$

and represents obstacle avoidance constraints. Since state predictions over prediction horizon are related to the initial state and the predicted input over the same horizon by $\mathbf{x}_h = \bar{T}\mathbf{x}(0) + \bar{S}\mathbf{u}_k$, then it holds that:

$$\begin{gathered} A_{bar}\mathbf{x}_h - A_{bar}\bar{T}\mathbf{x}(0) \leq B_{bar} - A_{bar}\bar{T}\mathbf{x}(0) \\ A_{bar}\mathbf{x}_h \leq B_{bar} \end{gathered} \tag{2.42}$$

By expliciting matrix A_{bar} containing constraints definition and the predicted state $\mathbf{x}_h$, the result is:

$$\begin{bmatrix} A_{constr} & \dots & 0_{ML\times n} \\ \vdots & \ddots & \vdots \\ 0_{ML\times n} & \dots & A_{constr} \end{bmatrix} \begin{bmatrix} x(1) \\ y(1) \\ v_x(1) \\ v_y(1) \\ \vdots \\ x(N) \\ y(N) \\ v_x(N) \\ v_y(N) \end{bmatrix} \leq B_{bar} \tag{2.43}$$

After multiplying matrices, and by using their definitions in Appendix .1, it follows that each row defines a scalar inequality:

$$\begin{bmatrix} [q_{1_x} - x(0)]x(1) + [q_{1_y} - y(0)]y(1) \\ [q_{1_x} - x(0)]x(1) + [q_{1_y} - y(0)]y(1) \\ \vdots \\ [q_{2_x} - x(0)]x(1) + [q_{2_y} - y(0)]y(1) \\ \vdots \\ [q_{1_x} - x(0)]x(2) + [q_{1_y} - y(0)]y(2) \\ \vdots \end{bmatrix} \leq \begin{bmatrix} [q_{1_x} - x(0)][q_{1_x} - v_{1_x}] + [q_{1_y} - y(0)][q_{1_y} - v_{1_y}] \\ [q_{1_x} - x(0)][q_{1_x} - v_{2_x}] + [q_{1_y} - y(0)][q_{1_y} - v_{2_y}] \\ \vdots \\ [q_{2_x} - x(0)][q_{2_x} - v_{1_x}] + [q_{2_y} - y(0)][q_{2_y} - v_{1_y}] \\ \vdots \\ [q_{1_x} - x(0)][q_{1_x} - v_{1_x}] + [q_{1_y} - y(0)][q_{1_y} - v_{1_y}] \\ \vdots \end{bmatrix} \tag{2.44}$$

To clarify the inequality and give it a more geometrically interpretable form, it is suitable to focus on individual rows, reordering terms step by step to extract the underlying geometrical meaning. Note that the outcomings are valid for each i-th obstacle, j-th vertex and h time step.

$$\left(\begin{bmatrix} q_{i_x} \\ q_{i_y} \end{bmatrix} - \begin{bmatrix} x(0) \\ y(0) \end{bmatrix}\right)^T \begin{bmatrix} x(h) \\ y(h) \end{bmatrix} \leq \left(\begin{bmatrix} q_{i_x} \\ q_{i_y} \end{bmatrix} - \begin{bmatrix} x(0) \\ y(0) \end{bmatrix}\right)^T \left(\begin{bmatrix} q_{i_x} \\ q_{i_y} \end{bmatrix} - \begin{bmatrix} v_{j_x} \\ v_{j_y} \end{bmatrix}\right) \tag{2.45}$$

$$-\left(\begin{bmatrix} q_{i_x} \\ q_{i_y} \end{bmatrix} - \begin{bmatrix} x(0) \\ y(0) \end{bmatrix}\right)^T \begin{bmatrix} x(h) \\ y(h) \end{bmatrix} + \left(\begin{bmatrix} q_{i_x} \\ q_{i_y} \end{bmatrix} - \begin{bmatrix} x(0) \\ y(0) \end{bmatrix}\right)^T \left(\begin{bmatrix} q_{i_x} \\ q_{i_y} \end{bmatrix} - \begin{bmatrix} v_{j_x} \\ v_{j_y} \end{bmatrix}\right) \geq 0 \tag{2.46}$$

$$\left(\begin{bmatrix} q_{i_x} \\ q_{i_y} \end{bmatrix} - \begin{bmatrix} x(0) \\ y(0) \end{bmatrix}\right)^T \left(\begin{bmatrix} q_{i_x} \\ q_{i_y} \end{bmatrix} - \left(\begin{bmatrix} x(h) \\ y(h) \end{bmatrix} + \begin{bmatrix} v_{j_x} \\ v_{j_y} \end{bmatrix}\right)\right) \geq 0 \tag{2.47}$$

$$\left(\begin{bmatrix}x(0)\\y(0)\end{bmatrix}-\begin{bmatrix}q_{i_x}\\q_{i_y}\end{bmatrix}\right)^T\left(\left(\begin{bmatrix}x(h)\\y(h)\end{bmatrix}+\begin{bmatrix}v_{j_x}\\v_{j_y}\end{bmatrix}\right)-\begin{bmatrix}q_{i_x}\\q_{i_y}\end{bmatrix}\right)\geq 0 \tag{2.48}$$

$$(\mathbf{vec_1})^T(\mathbf{vec_2})\geq 0 \tag{2.49}$$

$$\mathbf{vec_1}:=\left(\begin{bmatrix}x(0)\\y(0)\end{bmatrix}-\begin{bmatrix}q_{i_x}\\q_{i_y}\end{bmatrix}\right)\qquad \mathbf{vec_2}:=\left(\left(\begin{bmatrix}x(h)\\y(h)\end{bmatrix}+\begin{bmatrix}v_{j_x}\\v_{j_y}\end{bmatrix}\right)-\begin{bmatrix}q_{i_x}\\q_{i_y}\end{bmatrix}\right) \tag{2.50}$$

The vector $\mathbf{vec_1}$ links the point on i-th obstacle surface $\mathbf{q}_i$ to the center of the robot at step 0 of the prediction horizon. $\mathbf{vec_2}$ links $\mathbf{q}_i$ to the position of the j-th vertex of the robot in world frame at step h. Inequality (2.48) holds when $\mathbf{vec_1}$ and $\mathbf{vec_2}$, which are 2-D vectors, are both pointing out from the half-plane containing the i-th obstacle, tangent to the latter at $\mathbf{q}_i$ and normal to $\mathbf{vec_1}$ (Figure 2.4).
Geometrically, this means that each vertex during the entire prediction horizon cannot lie inside the half-plane containing the i-th obstacle. This half-plane is static throughout the horizon. When considering all obstacles, the combination of all their half-planes forms a convex region, known as the safe area, within which the robot must remain to avoid collisions, similar to the one represented in Figure 2.3. Consequently, the constraints applied to a single half-plane must hold for the entire safe area to ensure the robot trajectory remains free from obstacles.

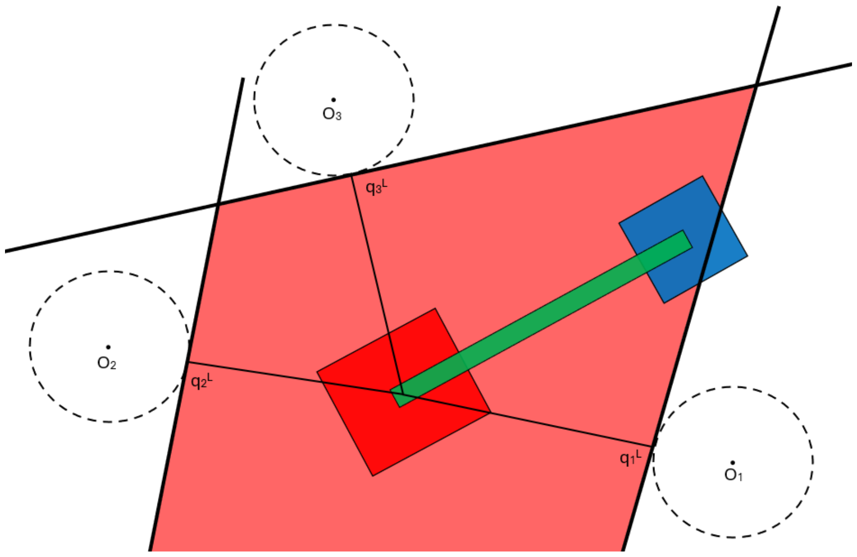
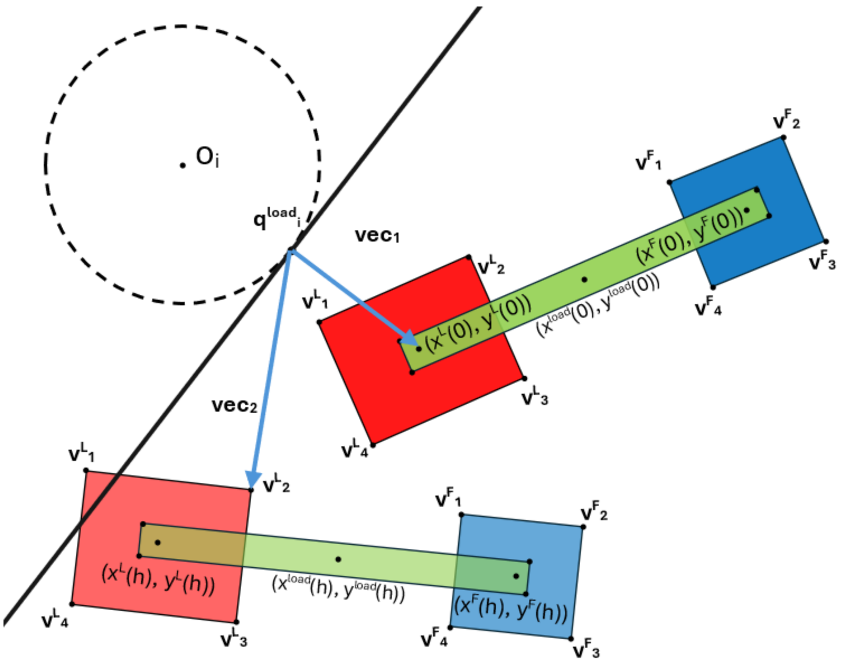
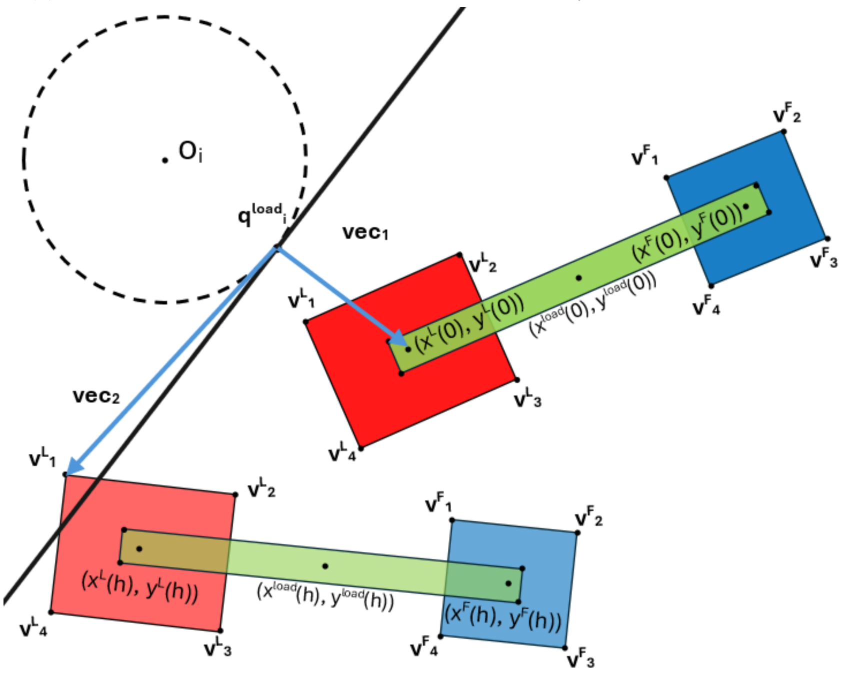


Figure 2.3: Example of obstacle avoidance convex area for leader agent. Leader predicted positions throughout the current prediction horizon need to lay inside the red area, delimited by half-planes tangent to obstacles. At next iteration of the MPC, the half-planes, and consequentially the safe area, will change. Leader and follower have different safe areas, but the definition is the same.



(a) Constraint satisfied, because both vectors lay inside the safe area.



(b) Constraint violated, because $\mathbf{vec_2}$ lays outside the safe area.

Figure 2.4: Examples of linear obstacle avoidance constraints. The system in bright colors is the system at time step 0 of the current prediction horizon, while the system in faded colors represents the system at a generic step h of the current prediction horizon. $\mathbf{vec_1}$ and $\mathbf{vec_2}$ for vertices $\mathbf{v}_1^L$ and $\mathbf{v}_2^L$, as an example, are drawn in light blue.

The total amount of obstacle avoidance constraints are:

- NML^L for leader, where L^L is the number of leader vertices.
- NML^F for follower, where L^F is the number of follower vertices.

2. The second row block of inequality (2.40), i.e. inequality

$$G_{in}\mathbf{u}_k \leq B_{in} \tag{2.51}$$

accounts for input constraints, namely $u^*_{x/y}(k) \in [-u^*_{max}, u^*_{max}]$. As a matter of fact, expanding matrices in inequality (2.51) with matrices in Appendix .1, we obtain:

$$\begin{bmatrix} U_l & \dots & 0_{2m\times m} \\ \vdots & \ddots & \vdots \\ 0_{2m\times m} & \dots & U_l \end{bmatrix} \begin{bmatrix} u_x(0) \\ u_y(0) \\ \vdots \\ u_x(N-1) \\ u_y(N-1) \end{bmatrix} \leq \begin{bmatrix} \begin{bmatrix} u_{max} \\ u_{max} \\ u_{max} \\ u_{max} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} u_{max} \\ u_{max} \\ u_{max} \\ u_{max} \end{bmatrix} \\ \vdots \end{bmatrix} \tag{2.52}$$

where U_l is the matrix selecting each control input, once for the positive and once for the negative bound, within $\mathbf{u}^*(k)$ (see Appendix .1). Multiplying matrices leads to:

$$\begin{bmatrix} u_x(0) \\ -u_x(0) \\ u_y(0) \\ -u_y(0) \\ \vdots \\ u_x(N-1) \\ -u_x(N-1) \\ u_y(N-1) \\ -u_y(N-1) \end{bmatrix} \leq \begin{bmatrix} u_{max} \\ u_{max} \\ u_{max} \\ u_{max} \\ \vdots \\ u_{max} \\ u_{max} \\ u_{max} \\ u_{max} \end{bmatrix} \tag{2.53}$$

The total number of input constraints for each agent results to be $2mN$.

3. The third row block of inequality (2.40), i.e. inequality

$$\begin{aligned} A_{vel}\bar{S}\mathbf{u}_k &\leq B_{vel} - A_{vel}\bar{T}\mathbf{x}(0) \\ A_{vel}\mathbf{x}_h - A_{vel}\bar{T}\mathbf{x}(0) &\leq B_{vel} - A_{vel}\bar{T}\mathbf{x}(0) \\ A_{vel}\mathbf{x}_h &\leq B_{vel} \end{aligned} \tag{2.54}$$

accounts for velocity constraints, namely $v^*_{x/y}(h) \in [-v^*_{max}, v^*_{max}]$. Once again, expanding matrices in inequality (2.54) with matrices found in Appendix .1:

$$\begin{bmatrix} V_l & \dots & 0_{2v\times n} \\ \vdots & \ddots & \vdots \\ 0_{2v\times n} & \dots & V_l \end{bmatrix} \begin{bmatrix} x(1) \\ y(1) \\ v_x(1) \\ v_y(1) \\ \vdots \\ x(N) \\ y(N) \\ v_x(N) \\ v_y(N) \end{bmatrix} \leq \begin{bmatrix} \begin{bmatrix} v_{max} \\ v_{max} \\ v_{max} \\ v_{max} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} v_{max} \\ v_{max} \\ v_{max} \\ v_{max} \end{bmatrix} \\ \vdots \end{bmatrix} \tag{2.55}$$

where V_l is the matrix selecting each linear velocity, once for the positive and once for the negative bound, within $\mathbf{x}^*(h)$ (see Appendix .1). The result is:

$$\begin{bmatrix} v_x(1) \\ -v_x(1) \\ v_y(1) \\ -v_y(1) \\ \vdots \\ v_x(N) \\ -v_x(N) \\ v_y(N) \\ -v_y(N) \end{bmatrix} \leq \begin{bmatrix} v_{max} \\ v_{max} \\ v_{max} \\ v_{max} \\ \vdots \\ v_{max} \\ v_{max} \\ v_{max} \\ v_{max} \end{bmatrix} \tag{2.56}$$

The total number of velocity constraints for each agent results to be $2vN$.

2.3 Discussion and remarks

This 2-DOF LMPC framework proved to have good performance in [6] regarding minimization of formation error and obstacle avoidance. Hereafter, pros and cons are listed to point out strengths and weaknesses of this algorithm, and consequently introduce the 3-DOF NLMPC.

- Pros:

  1. Simple double integrator UGVs dynamic model.
  2. Effective choice of leader and follower functional costs to efficiently reach the goal and to keep formation error bounded.
  3. Great enhancement in formation error performance thanks to recovery policy, shown in [6].
  4. Obstacle avoidance approach implemented through hard constraints, guaranteeing safety and reliability.

- Cons:

  1. Double integrator UGVs model does not consider agents independent rotation with respect to the payload, causing the system to lack on flexibility and affecting obstacle avoidance performance.
  2. There is no penalty function that penalizes robot vertices too near obstacles, so agent-obstacle distance tends to be dangerously small. This might affect obstacle avoidance robustness.
  3. Follower does not have a control effort functional cost, leading this agent to carry out unnecessary movements.
  4. Recovery policy might be computationally expensive if evaluated too often, due to the fact it expects to execute input computation twice in an MPC iteration (see Section 2.2.3).
  5. This obstacle avoidance approach lacks in robustness when dynamic obstacles are present in the environment. This is due to the fact that constraints consider each obstacle to be static during the prediction horizon. This assumption is ensured exclusively when obstacle dynamics is slow enough.
  6. The fact that half-planes composing the safe area are static during the prediction horizon may affect obstacle avoidance flexibility in particularly cluttered environments (see Section 3.3.5).

Consequently, 3-DOF NLPMC framework is developed to enhance UGVs model to consider agent rotation. In addition, it adds a potential repulsive field function to penalize robot vertices too near obstacles and a follower control effort function that limits follower agent movements.

Chapter 3

Non-Linear Model Predictive Control Framework for a 3 degrees of freedom Multi-Robot System: design and development

3-DOF NLMPC framework is developed and discussed in this chapter. This algorithm builds on the solution presented in Chapter 2 by incorporating the robots 3-DOF dynamic model, which accounts for rigid body dynamics. This includes the agents independent rotational capabilities alongside translational motion, offering a more realistic representation of their behavior in the environment. Functional costs and obstacle avoidance constraints are adapted to also consider the agent orientation. In particular, obstacle avoidance constraints incorporate payload vertices, too. Moreover, potential repulsive field function and follower control effort function are added to the previous agent functional cost formulation (Section 2.2). The state machine implementing the variable prediction horizon for this framework is described, with the aim of enhancing computational efficiency. Lastly, considering that in Chapter 2 agents are assumed to know the exact map of the environment and do not possess realistic sensing capabilities, a perception range for obstacle sensing is introduced to achieve a more realistic and effective algorithm.
The non-linearity of this new framework arises from the definition of the set of vertices $\mathcal{V}^*$, where each $\mathbf{v}_j^* \in \mathcal{V}^*$ is referenced to agent $* = L, F$ body frame, centered in the agent center of mass. This means that, in a system where agents can rotate, a rototranslation of the vertices is required to express their position in world frame.

One of the principal aims of this work is to develop a control solution that enables each agent in the MRS to rotate independently of the payload. This capability enhances the transportation system flexibility in challenging environments, where precise obstacle avoidance is critical and often pushed to its limits. Consequently, the following algorithmic changes and improvements of the algorithm are tested with a rectangular leader agent tasked with navigating through a narrow space between two obstacles, which is designed to be smaller than the biggest side of the leader shape, to force the robot to rotate (Figure 3.1). This experiment is inspired by robots with an elongated shape commonly employed in robotic transportation, such as OmniAGV© [12].

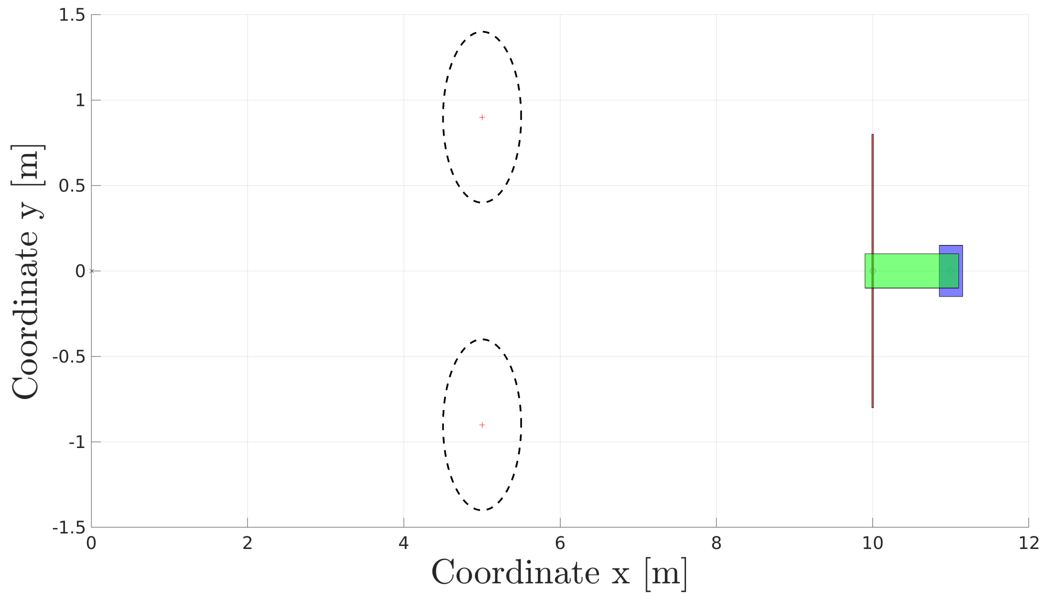


Figure 3.1: Test setup for rotation capabilities of the system. The MRS, on the right, is composed by the leader (in red), the follower (in blue) and the load (in green), while the obstacles are the two dotted line circles on the left. The biggest side of the leader is $1.6\,m$ long, while the passage between obstacles is $0.8\,m$ long, $o_1 = \begin{bmatrix}5 & 0.9\end{bmatrix}^T m$, $o_2 = \begin{bmatrix}5 & -0.9\end{bmatrix}^T m$, $r_1 = r_2 = 0.5\,m$.

3.1 Application scenario assumptions

The assumptions made regarding the application scenario for the 3-DOF NLMPC framework are consistent with those reported in Section 2.1. There is only one difference regarding the end-effector which can be considered as a revolute joint, allowing agents independent rotation with respect to the payload.

3.2 Rigid body agents dynamic model

The enhanced dynamic model of both the UGVs composing the considered MRS is described here, and it closely follows the model presented in Section 2.1.1.
Agent $* = L,F$ state is now

$$\mathbf{x}^*(t) = \begin{bmatrix} x^*(t) & y^*(t) & \theta^*(t) & v_x^*(t) & v_y^*(t) & \omega^*(t) \end{bmatrix}^T \in \mathbb{R}^6 \tag{3.1}$$

referenced to the world frame, such that heading angle and angular velocity are incorporated. t is a generic time step. The position of the agent in the plane is:

$$\mathbf{p}^*(t) = \begin{bmatrix} x^*(t) & y^*(t) \end{bmatrix}^T \in \mathbb{R}^2 \tag{3.2}$$

The control input of the $*$ agent is

$$\mathbf{u}^*(t) = \begin{bmatrix} u_x^*(t) & u_y^*(t) & u_\theta^*(t) \end{bmatrix}^T \in \mathbb{R}^3 \tag{3.3}$$

referenced to the world frame, and it is composed of robot linear ($u_x^*(t)$ and $u_y^*(t)$) and angular accelerations ($u_\theta^*(t)$).
The continuous-time dynamic model changes into:

$$\dot{\mathbf{x}}^*(t) = A_c\mathbf{x}^*(t) + B_c\mathbf{u}^*(t) \tag{3.4}$$

$$\mathbf{y}^*(t) = \mathbf{x}^*(t) \tag{3.5}$$

where matrices are defined as:

$$A_c = \begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix} \in \mathbb{R}^{6\times6} \qquad B_c = \begin{bmatrix} 0_{3\times3} \\ I_{3\times3} \end{bmatrix} \in \mathbb{R}^{6\times3} \tag{3.6}$$

The time discretization of the continuous-time model is exactly the same as in (2.7):

$$\mathbf{x}^*(s+1) = A\mathbf{x}^*(s) + B\mathbf{u}^*(s) \tag{3.7}$$

$$\mathbf{y}^*(s) = \mathbf{x}^*(s) \tag{3.8}$$

where $s = \lfloor \frac{t}{T_s} \rfloor$ is the discrete time step, T_s is the sample time, with t being the continuous time and $s \in \mathbb{Z}$. A and B are the discrete-time model state and input matrices. State and control input during the prediction horizon are defined with the same notation introduced in Section 2.2 for the 2-DOF LMPC model, namely:

$$\mathbf{u}_k^* = \begin{bmatrix}\mathbf{u}^*(0) & \dots & \mathbf{u}^*(N-1)\end{bmatrix}^T \in \mathbb{R}^{mN}, k = 0\dots(N-1) \tag{3.9}$$

$$\mathbf{x}_h^* = \begin{bmatrix}\mathbf{x}^*(1) & \dots & \mathbf{x}^*(N)\end{bmatrix}^T \in \mathbb{R}^{nN}, h = 1\dots N \tag{3.10}$$

3.3 Non-Linear Model Predictive Control formulation

Leader and follower have the same task to accomplish described in Section 2.2, therefore they have the same functional costs. However, to improve the performance of the system, potential repulsive field function and follower control effort function are added. Moreover, non-linear constraints are defined to account for agents rotation and to avoid load vertices collision, taking as a reference the geometrical approach explained in Section 2.2.4.

The rototranslation expressing vertices in world frame is formulated as:

$$\begin{aligned}\mathbf{v}_{j,\mathcal{F}_w}^*(t) &= \mathbf{p}^*(t) + R(\theta^*(t))\mathbf{v}_j^* \\ &= \begin{bmatrix}x^*(t)\\ y^*(t)\end{bmatrix} + \begin{bmatrix}cos(\theta^*(t)) & -sin(\theta^*(t))\\ sin(\theta^*(t)) & cos(\theta^*(t))\end{bmatrix}\begin{bmatrix}v_{j_x}^*\\ v_{j_y}^*\end{bmatrix}\end{aligned} \tag{3.11}$$

where $R(\phi) \in SO(2)$ is the transformation matrix that performs a rotation of ϕ angle in the 2-D plane, $\mathbf{p}^*(t) \in \mathbb{R}^2$ is the position of the $* = L,F$ agent and $\theta^*(t) \in \mathbb{R}$ is the $* = L, F$ agent heading angle.

3.3.1 Potential repulsive field

A potential repulsive field function is introduced to force agents to keep a certain distance to obstacles, with the purpose of improving obstacle avoidance performance and robustness and to make this algorithm more suitable for a real application. Specifically, this function optimizes the distance between each vertex of the agents and the obstacles.
This potential repulsive field function is similar to the one used in [7] and is:

$$J_3^* = \sum_{i=1}^{M}\sum_{j=1}^{L^*}\sum_{h=1}^{N} C_{pot} exp\left(-\lambda\left[\left\|\mathbf{p}^*(h) + R(\theta^*(h))\begin{bmatrix}v_{x_j}^*\\ v_{y_j}^*\end{bmatrix} - \begin{bmatrix}o_{x_i}\\ o_{y_i}\end{bmatrix}\right\| - r_i\right]\right) \tag{3.12}$$

$C_{pot} \in \mathbb{R}$ is a weight variable. The higher C_{pot} is, the greater the penalization of the functional cost is when vertices are too near an obstacle. $\lambda \in \mathbb{R}$ determines the decay rate of the cost, such that a higher value of λ ensures that only obstacles in close proximity to the agent significantly affect the cost. This is valid for agents vertices only, $* = L, F$. This approach does not apply to load vertices because it would simply increase the cost without offering any advantage.

For clarity, J_3^* will be manipulated to explicitly show it depends on $\mathbf{u}_k^F$. Superscripts are omitted to simplify the notation.
Firstly we need to extract $\mathbf{u}_k$ from the state $\mathbf{x}_h$, using the equation $\mathbf{x}_h = \bar{T}\mathbf{x}(0) + \bar{S}\mathbf{u}_k$ (2.16). Then $\mathbf{p}(h)$ and $\theta(h)$ need to be selected, so, respectively, selection matrices $F(h)$ and $G(h)$ are defined. These are time-variant matrices that depend on the step h of the prediction horizon. In fact, in order to select the correct variables from the state vector, these matrices will have ones only in the position corresponding to the correct variable within $\mathbf{x}_h$. For example, if $h = 1$, it is:

$$F(1) = \begin{bmatrix} 1 & 0 & 0 & \dots \\ 0 & 1 & 0 & \dots \end{bmatrix} \in \mathbb{R}^{2\times nN} \qquad G(1) = \begin{bmatrix} 0 & 0 & 1 & 0 & \dots \end{bmatrix} \in \mathbb{R}^{1\times nN} \tag{3.13}$$

$F(1)$ selects $\mathbf{p}(1)$ and $G(1)$ selects $\theta(1)$. Thanks to these matrices, functional cost J_3 can be decomposed in h parts, each of them referred to time step h of the prediction horizon:

$$J_3(h) = \sum_{i=1}^{M}\sum_{j=1}^{L} C_{pot} exp\Bigg(-\lambda\Bigg[\Bigg\|F(h)(\bar{T}\mathbf{x}(0) + \bar{S}\mathbf{u}_k) + R\left(G(h)(\bar{T}\mathbf{x}(0) + \bar{S}\mathbf{u}_k)\right)\begin{bmatrix} v_{x_j} \\ v_{y_j} \end{bmatrix} - \begin{bmatrix} o_{x_i} \\ o_{y_i} \end{bmatrix}\Bigg\| - r_i\Bigg]\Bigg) \tag{3.14}$$

such that the total functional cost is:

$$J_3 = \sum_{h=1}^{N} J_3(h) \tag{3.15}$$

So (3.15) is the function added to J^L (2.20), J^L_{rec} (2.35) and J^F (2.28) to incorporate the repulsive field into agents functional cost, that change into:

$$J^L = J_1^L + J_3^L \tag{3.16}$$

$$J^L_{rec} = J^L_{rec,1} + J_3^L \tag{3.17}$$

$$J^F = CJ_1^F + J_2^F + J_3^F \tag{3.18}$$

Gradient of the follower functional cost is used during the optimization process to accelerate computation. The expression for this gradient is presented here.
Using the definition of $R(\phi)$ we can derive:

$$J_3(h) = \sum_{i=1}^{M}\sum_{j=1}^{L} C_{pot}exp\Bigg(-\lambda\Bigg[\Bigg|\Bigg|F(h)(\bar{T}\mathbf{x}(0)+\bar{S}\mathbf{u}_k)+ \begin{bmatrix} cos\left(G(h)(\bar{T}\mathbf{x}(0)+\bar{S}\mathbf{u}_k)\right) & -sin\left(G(h)(\bar{T}\mathbf{x}(0)+\bar{S}\mathbf{u}_k)\right) \\ sin\left(G(h)(\bar{T}\mathbf{x}(0)+\bar{S}\mathbf{u}_k)\right) & cos\left(G(h)(\bar{T}\mathbf{x}(0)+\bar{S}\mathbf{u}_k)\right) \end{bmatrix} \begin{bmatrix} v_{x_j} \\ v_{y_j} \end{bmatrix} - \begin{bmatrix} o_{x_i} \\ o_{y_i} \end{bmatrix}\Bigg|\Bigg| - r_i\Bigg]\Bigg) \tag{3.19}$$

It is useful to define vector D:

$$D(h) := F(h)(\bar{T}\mathbf{x}(0)+\bar{S}\mathbf{u}_k)+ + \begin{bmatrix} cos\left(G(h)(\bar{T}\mathbf{x}(0)+\bar{S}\mathbf{u}_k)\right)v_{x_j} - sin\left(G(h)(\bar{T}\mathbf{x}(0)+\bar{S}\mathbf{u}_k)\right)v_{y_j} \\ sin\left(G(h)(\bar{T}\mathbf{x}(0)+\bar{S}\mathbf{u}_k)\right)v_{x_j} + cos\left(G(h)(\bar{T}\mathbf{x}(0)+\bar{S}\mathbf{u}_k)\right)v_{y_j} \end{bmatrix} - \begin{bmatrix} o_{x_i} \\ o_{y_i} \end{bmatrix} \tag{3.20}$$

so that $J_3(h)$ equation is simplified:

$$J_3(h) = \sum_{i=1}^{M}\sum_{j=1}^{L} C_{pot}exp\left(-\lambda\left[||D(h)|| - r_i\right]\right) \tag{3.21}$$

The gradient of $J_3(h)$ with respect to $\mathbf{u}_k$ is:

$$g_3(h) = \sum_{i=1}^{M}\sum_{j=1}^{L} -\lambda C_{pot}exp(-\lambda[||D(h)|| - r_i])\Bigg(\bar{S}^T(F(h))^T+ \bar{S}^T(G(h))^T \begin{bmatrix} -sin\Big(G(h)(\bar{T}\mathbf{x}(0)+\bar{S}\mathbf{u}_k)\Big)v_{x_j} - cos\Big(G(h)(\bar{T}\mathbf{x}(0)+\bar{S}\mathbf{u}_k)\Big)v_{y_j} \\ cos\Big(G(h)(\bar{T}\mathbf{x}(0)+\bar{S}\mathbf{u}_k)\Big)v_{x_j} - sin\Big(G(h)(\bar{T}\mathbf{x}(0)+\bar{S}\mathbf{u}_k)\Big)v_{y_j} \end{bmatrix}^T\Bigg)\frac{D(h)}{||D(h)||} \tag{3.22}$$

The total potential repulsive field gradient for both $* = L,F$ agent is:

$$g_3^* = \sum_{h=1}^{N} g_3^*(h) \tag{3.23}$$

3.3.2 Follower control effort function

During the initial tests of the algorithm, follower agent was easily induced to move or rotate way more than necessary, making the control effort to be higher than expected. For this reason, a control effort functional cost is introduced for this agent to avoid useless accelerations:

$$J_4^F = (\mathbf{u}_k^F)^T R_{eff} \mathbf{u}_k^F \tag{3.24}$$

$R_F \in \mathbb{R}^{m \times m}$ is the follower control effort weight, that aims to reduce the energy spent to move the follower, while $R_{eff} = I_N \otimes R_F \in \mathbb{R}^{mN \times mN}$ is the matrix used to apply the control effort weight to all inputs throughout the prediction horizon, $\mathbf{u}_k^F$.
The follower functional cost J^F finally becomes:

$$J^F = CJ_1^F + J_2^F + J_3^F + J_4^F \tag{3.25}$$

The corresponding gradient of equation (3.24) is:

$$g_4^F = 2R_{eff} \mathbf{u}_k^F \tag{3.26}$$

Accordingly, gradient of the follower functional cost is:

$$g^F = Cg_1^F + g_2^F + g_3^F + g_4^F \tag{3.27}$$

3.3.3 Angular velocity and acceleration constraints

The enhanced rigid body dynamic model of both the UGVs enabling rotation of the agent requires a new matrix inequality definition for linear constraints, introducing angular velocity and acceleration limits. Moreover, with respect to equation (2.40), obstacle avoidance constraints are excluded, since they turned into non-linear constraints and have to be expressed separately. This new formulation is:

$$G' \mathbf{u}_k^* \leq W' + S' \mathbf{x}^*(0) \tag{3.28}$$

Expanding matrices we obtain:

$$\begin{bmatrix} G'_{in} \\ A'_{vel} \bar{S} \end{bmatrix} \mathbf{u}_k \leq \begin{bmatrix} B_{in} \\ B_{vel\_constr} \end{bmatrix} + \begin{bmatrix} 0_{2vN \times n} \\ -(A'_{vel} \bar{T}) \end{bmatrix} \mathbf{x}(0) \tag{3.29}$$

Heading angle is added to the model, so it is also necessary to apply boundaries to angular acceleration and velocity, along with linear accelerations and velocities already treated in Section

2.2.4. G', W' and S' have different dimensions with respect to G, W and S according to agent state and input dimension. In particular G'_{in} and A'_{vel} are hereafter defined.

Angular acceleration constraints

Acceleration constraints need to include $u^*_\theta(k)$, such that $u^*_\theta(k) \in [-u^*_{\theta,max}, u^*_{\theta,max}]$. To reach this results, G_{in} of the 2-DOF LMPC (2.51) needs to change into:

$$G'_{in} = I_N \otimes U_{nl} = \begin{bmatrix} U_{nl} & \dots & 0_{2m\times m} \\ \vdots & \ddots & \vdots \\ 0_{2m\times m} & \dots & U_{nl} \end{bmatrix} \in \mathbb{R}^{2mN\times mN} \tag{3.30}$$

where U_{nl} is the matrix selecting each acceleration input, once for the positive and once for the negative bound, within $\mathbf{u}^*(k)$ (see Appendix .1), which is defined as:

$$U_{nl} := \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix} \tag{3.31}$$

Angular velocity constraints

The same concept as angular acceleration is implemented for angular velocities, where $\omega^*(h) \in [-\omega^*_{max}, \omega^*_{max}]$. A_{vel} of the 2-DOF LMPC (2.54) dimension changes and it becomes:

$$A'_{vel} = I_N \otimes V_{nl} = \begin{bmatrix} V_{nl} & \dots & 0_{2v\times n} \\ \vdots & \ddots & \vdots \\ 0_{2v\times n} & \dots & V_{nl} \end{bmatrix} \in \mathbb{R}^{2vN\times nN} \tag{3.32}$$

where V_{nl} is the matrix selecting each velocity, once for the positive and once for the negative bound, within $\mathbf{x}^*(h)$ (see Appendix .1), defined as follows:

$$V_{nl} := \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{3.33}$$

3.3.4 Non-linear constraints

Geometrical approach linear constraints analyzed in Section 2.2.4 are used as reference to build non-linear obstacle avoidance constraints. Additionally, load obstacle avoidance constraints are introduced using the same approach to account for the possibility of load vertices protruding from the leader or follower robot's shape. Load vertices in world frame are expressed as:

$$\begin{aligned} \mathbf{v}_{j,\mathcal{F}_w}^{load}(t) &= \mathbf{p}^F(t) + R(\theta^{load}(t))\mathbf{v}_j^{load} \\ &= \begin{bmatrix} x^F(t) \\ y^F(t) \end{bmatrix} + \begin{bmatrix} cos(\theta^{load}(t)) & -sin(\theta^{load}(t)) \\ sin(\theta^{load}(t)) & cos(\theta^{load}(t)) \end{bmatrix} \begin{bmatrix} v_{j_x}^{load} \\ v_{j_y}^{load} \end{bmatrix} \end{aligned} \tag{3.34}$$

Notice that load vertices $\mathbf{v}_j^{load} \in \mathbb{R}^2$ are referenced to follower body frame. As a matter of fact,

$$\theta^{load}(t) = atan2\big((y^F(t) - y^L(t)), (x^F(t) - x^L(t))\big) \tag{3.35}$$

is the rotation angle of the load in a generic step t, that depends on the relative position between leader and follower. It is $\theta^{load} = 0$ when both agents center lays on the same y coordinate and follower stands on leader right.
Since in Section 3.3.4 load vertices will be added to obstacle avoidance constraints and considering that load vertices position in world frame depends on both leader and follower position, the leader optimization problem must be solved first. Then follower optimization problem must be computed incorporating the obstacle avoidance constraints for the load.

The idea is to use the same inequality used for linear constraints (2.48) and introducing the rotation of the vertices. This formulation is valid for each agent, i-th obstacle, j-th vertex and h time step.

$$(\mathbf{vec_1})^T(\mathbf{vec_2}) \geq 0 \tag{3.36}$$

$$\mathbf{vec_1} := \begin{bmatrix} x^*(0) \\ y^*(0) \end{bmatrix} - \begin{bmatrix} q^*_{i_x} \\ q^*_{i_y} \end{bmatrix} \tag{3.37}$$

$$\mathbf{vec_2} := \left(\begin{bmatrix} x^*(h) \\ y^*(h) \end{bmatrix} + \begin{bmatrix} cos(\theta^*(h)) & -sin(\theta^*(h)) \\ sin(\theta^*(h)) & cos(\theta^*(h)) \end{bmatrix} \begin{bmatrix} v^*_{j_x} \\ v^*_{j_y} \end{bmatrix} \right) - \begin{bmatrix} q^*_{i_x} \\ q^*_{i_y} \end{bmatrix} \tag{3.38}$$

By expliciting and then rearranging signs and inequality sign:

$$\left(\begin{bmatrix} x^*(0) \\ y^*(0) \end{bmatrix} - \begin{bmatrix} q^*_{i_x} \\ q^*_{i_y} \end{bmatrix} \right)^T \left(\left(\begin{bmatrix} x^*(h) \\ y^*(h) \end{bmatrix} + \begin{bmatrix} cos(\theta^*(h)) & -sin(\theta^*(h)) \\ sin(\theta^*(h)) & cos(\theta^*(h)) \end{bmatrix} \begin{bmatrix} v^*_{j_x} \\ v^*_{j_y} \end{bmatrix} \right) - \begin{bmatrix} q^*_{i_x} \\ q^*_{i_y} \end{bmatrix} \right) \geq 0$$

$$-\left(\left(\begin{bmatrix} q^*_{i_x} \\ q^*_{i_y} \end{bmatrix} - \begin{bmatrix} x^*(0) \\ y^*(0) \end{bmatrix} \right)^T \left(\begin{bmatrix} q^*_{i_x} \\ q^*_{i_y} \end{bmatrix} - \left(\begin{bmatrix} x^*(h) \\ y^*(h) \end{bmatrix} + \begin{bmatrix} cos(\theta^*(h)) & -sin(\theta^*(h)) \\ sin(\theta^*(h)) & cos(\theta^*(h)) \end{bmatrix} \begin{bmatrix} v^*_{j_x} \\ v^*_{j_y} \end{bmatrix} \right) \right) \right) \leq 0$$

$$* = L, F. \qquad h = 1 \dots N. \qquad i = 1 \dots M. \qquad j = 1 \dots L^*.$$

The process slightly changes for load vertices, since they are referenced to follower body frame, so $\mathbf{p}^F(h)$ is used:

$$(\mathbf{vec_1}^{load})^T(\mathbf{vec_2}^{load}) \geq 0 \tag{3.39}$$

$$\mathbf{vec_1}^{load} := \left(\begin{bmatrix} x^F(0) \\ y^F(0) \end{bmatrix} + \begin{bmatrix} cos(\theta^{load}(0)) & -sin(\theta^{load}(0)) \\ sin(\theta^{load}(0)) & cos(\theta^{load}(0)) \end{bmatrix} \begin{bmatrix} x^{load} \\ y^{load} \end{bmatrix} \right) - \begin{bmatrix} q^{load}_{i_x} \\ q^{load}_{i_y} \end{bmatrix} \tag{3.40}$$

$$\mathbf{vec_2}^{load} := \left(\begin{bmatrix} x^F(h) \\ y^F(h) \end{bmatrix} + \begin{bmatrix} cos(\theta^{load}(h)) & -sin(\theta^{load}(h)) \\ sin(\theta^{load}(h)) & cos(\theta^{load}(h)) \end{bmatrix} \begin{bmatrix} v^{load}_{j_x} \\ v^{load}_{j_y} \end{bmatrix} \right) - \begin{bmatrix} q^{load}_{i_x} \\ q^{load}_{i_y} \end{bmatrix} \tag{3.41}$$

By expliciting and then rearranging signs and inequality sign:

$$\begin{gathered} -\left(\left(\begin{bmatrix} q^{load}_{i_x} \\ q^{load}_{i_y} \end{bmatrix} - \left(\begin{bmatrix} x^F(0) \\ y^F(0) \end{bmatrix} + \begin{bmatrix} cos(\theta^{load}(0)) & -sin(\theta^{load}(0)) \\ sin(\theta^{load}(0)) & cos(\theta^{load}(0)) \end{bmatrix} \begin{bmatrix} x^{load} \\ y^{load} \end{bmatrix} \right) \right)^T \right. \\ \left. \left(\begin{bmatrix} q^{load}_{i_x} \\ q^{load}_{i_y} \end{bmatrix} - \left(\begin{bmatrix} x^F(h) \\ y^F(h) \end{bmatrix} + \begin{bmatrix} cos(\theta^{load}(h)) & -sin(\theta^{load}(h)) \\ sin(\theta^{load}(h)) & cos(\theta^{load}(h)) \end{bmatrix} \begin{bmatrix} v^{load}_{j_x} \\ v^{load}_{j_y} \end{bmatrix} \right) \right) \right) \leq 0 \\ h = 1 \dots N. \qquad i = 1 \dots M. \qquad j = 1 \dots L^{load}. \end{gathered} \tag{3.42}$$

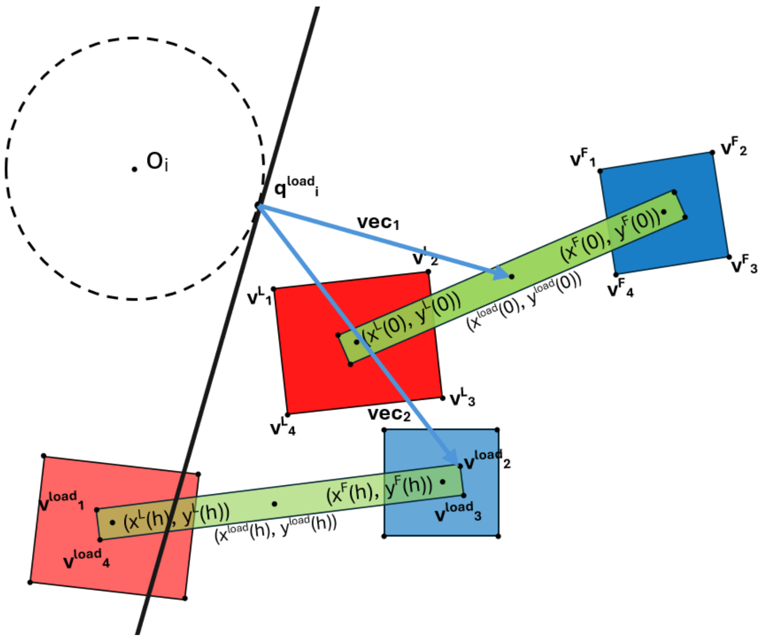
In equation (3.42)

$$\mathbf{x}^{load} = \begin{bmatrix} x^{load} & y^{load} \end{bmatrix}^T \in \mathbb{R}^2 \tag{3.43}$$

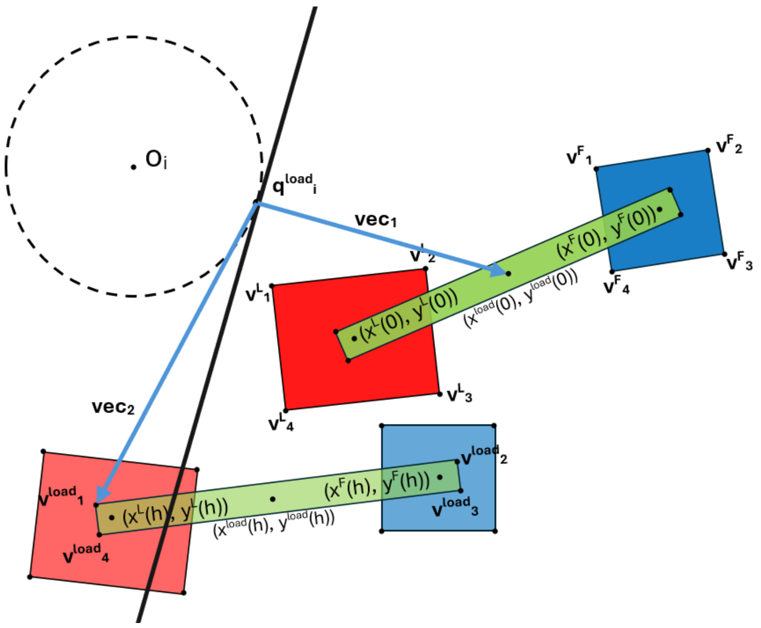
are the coordinates of the center of mass of the load defined as a fixed position in follower body

frame and $\theta^{load} \in \mathbb{R}$ is payload orientation. Therefore, $\mathbf{q}_i^{load}$ is the vector representing the point on the surface of the i-th obstacle in world frame that is the closest to the center of the load at step 0 of the current prediction horizon. As a consequence, the same geometrical meaning is valid for load vertices. They need to lie inside load safe area over the prediction horizon, which is built by the intersection of half-planes tangent to the i-th obstacle at $\mathbf{q}_i^{load}$ and normal to the vector $\mathbf{vec_1}$ connecting $\mathbf{q}_i^{load}$ and the load center at step 0 (Figure 3.2).
The total amount of non-linear obstacle avoidance constraints is:

- NML^L for leader, where L^L is the number of leader vertices;
- $NM(L^F + L^{load})$ for follower, where L^F and L^{load} are the number of follower and load vertices, respectively.



(a) Constraint satisfied, because both vectors lay inside the safe area.



(b) Constraint violated, because $\mathbf{vec_2}$ lays outside the safe area.

Figure 3.2: Examples of non-linear obstacle avoidance load constraints. The system in bright colors is the system at time step 0 of the current prediction horizon, while the system in faded colors represents the system at a generic step h of the current prediction horizon. $\mathbf{vec_1}$ and $\mathbf{vec_2}$ for load vertices $\mathbf{v}_1^{load}$ and $\mathbf{v}_2^{load}$, as an example, are drawn in light blue.

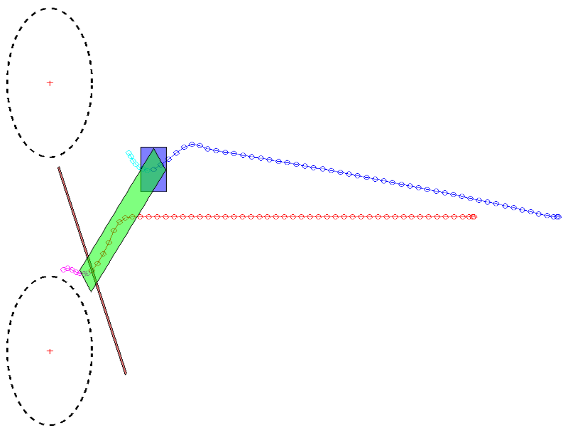
3.3.5 Qualitative discussion about non-linear obstacle avoidance constraints

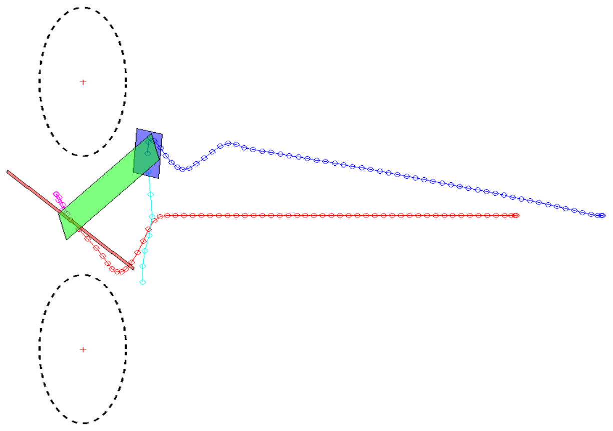
This section discusses the non-linear obstacle avoidance strategy implemented in Section 3.3.4, highlighting the significant enhancement in mobility of the MRS in cluttered environments. Moreover, different critical situations and environments are depicted and a deeper analysis of these constraints is provided to fully characterize this algorithm. Also, a few possible solutions to the emerged problems are proposed.

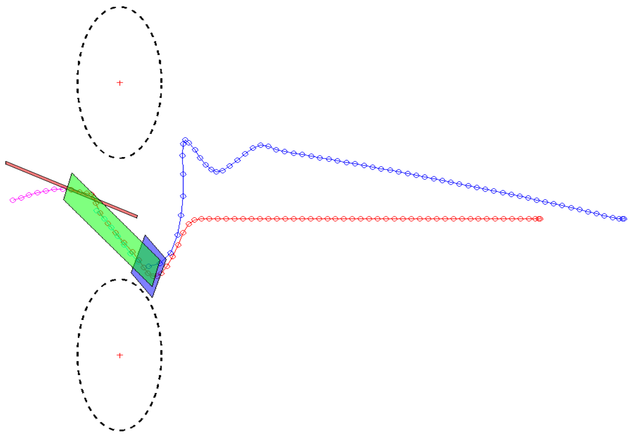
Narrow passage between static obstacles

To prove the effectiveness of the algorithm, qualitative tests are conducted using the setup shown in Figure 3.1. The purpose of these tests is simply to show that the MRS is capable of reaching the goal. However, these results strongly depend on MPC parameters, such as C_{pot}, and environment setup, where the dimension of the passage is crucial. The MPC parameters used in these tests are the optimal ones described in Section 4.1.
Figure 3.3 shows a sequence of images representing the simulation of this test, demonstrating that the MRS successfully navigates through the passage without any collision.

Despite its advantages, the proposed navigation solution has some limitations. As previously mentioned, the performance and the chance of success strongly depends on the set of MPC parameters and environment setup. For instance, if the corridor is too narrow and obstacles have a relatively small radius, the MRS may choose to bypass the corridor and navigate around the obstacles. This behaviour primarily depends on the C_{pot} weight, which establishes the distance each agent has to guarantee for obstacle avoidance.


(a) Instant 1


(b) Instant 2


(c) Instant 3

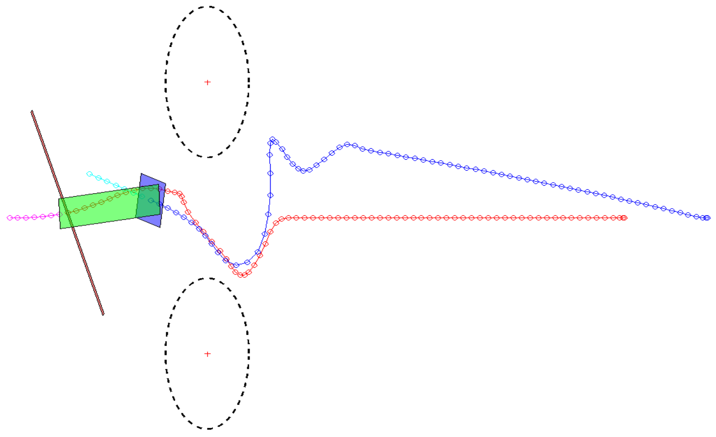

(d) Instant 4

Figure 3.3: Sequence that shows the effectiveness of agent rotation to navigate through narrow passages. Red and blue lines with circles are respectively leader and follower navigated path, while magenta and light blue lines with circles are respectively leader and follower predicted path.

Dynamic obstacles

Dynamic obstacles tests are important to assess whether the proposed navigation solution is effective and robust also in chaotic and dynamic environments, since the idea is to use this strategy in industrial context such as warehouse or other spaces shared with humans.
The presence of moving obstacles in the environment is considered in this navigation framework. However, since obstacles are considered fixed during the optimization process and half-planes that compose the safe areas are static during the prediction horizon, the algorithm is unreliable in presence of dynamic obstacles. Specifically, if an agent is near a moving obstacle, the optimization may lead the robot to plan a trajectory that avoids the obstacle based on its actual position, but this could lead to a collision if the object is moving toward the predicted position of the agent. For instance, a simple setup with two dynamic obstacles is shown in Figure 3.4.

At 7.4 s a collision clearly occurs (Figure 3.5) between the follower agent and one of the obstacles. For this reason, geometrical obstacle avoidance constraints approach designed in Sections 2.2.4 and 3.3.4 cause the algorithm to lack in robustness in presence of dynamic obstacles. One possible solution is to assume that moving obstacles have a sufficiently slow dynamics, making it reasonable to approximate them as static obstacles over the prediction horizon. A different solution might be the introduction of a sort of object behaviour prediction algorithm that can compute a prediction of the obstacle path, so that constraints are capable to deal with moving obstacles. However, this approach would increase significantly the computational effort.

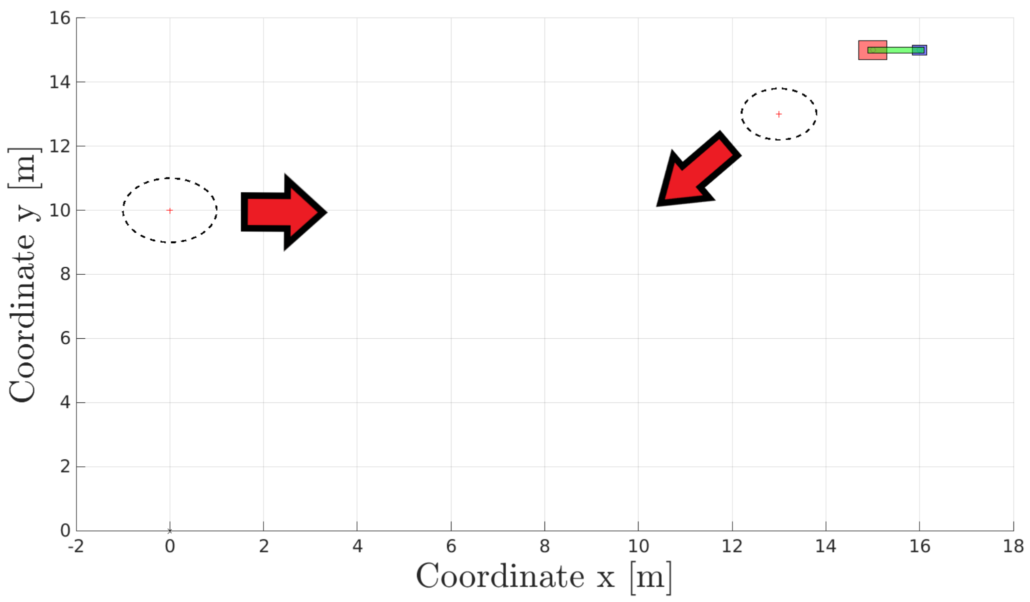


Figure 3.4: *Dynamic obstacles* environment. Two moving obstacles are placed in the environment. Red arrows indicate the rectilinear direction in which each obstacle moves. This frame shows the initial pose of these obstacles.

Other challenging conditions

In particular cases, the constraints described in equations (3.3.4) and (3.42) may be too strict. The fact that the half-planes are static during the prediction horizon could lead to an infeasible path, for example in particularly narrow L-shaped corridors (Figure 3.6), especially when the load is a long bar like the one considered in this work. An infeasible path, in this case, could cause the system to become stuck at a certain position without the possibility to move. Therefore, testing the system performance in specific conditions to evaluate the behaviour of these constraints could be a goal for future work.

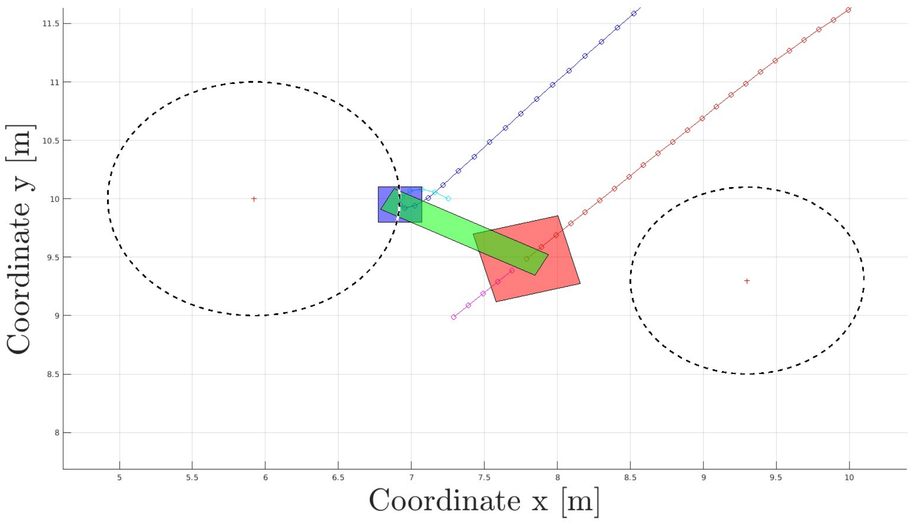


Figure 3.5: Follower collision with a dynamic obstacle.

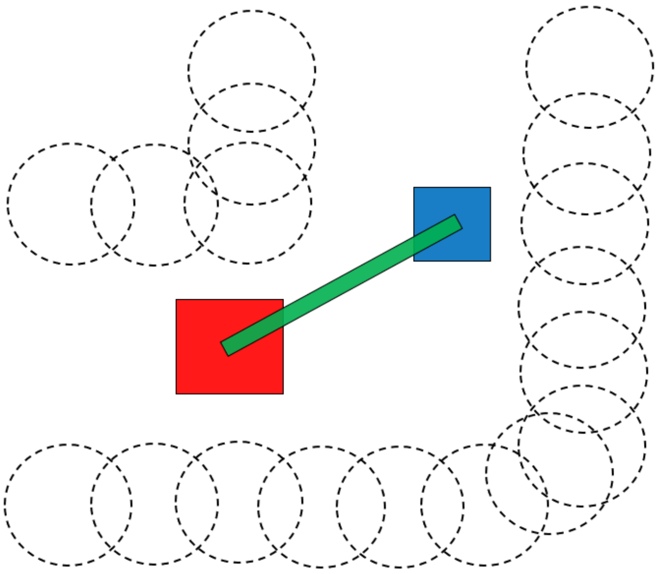


Figure 3.6: L-shaped corridor that might be challenging for this obstacle avoidance constraints approach.

An interesting feature that could be implemented may also be a leader-follower collision-avoidance, designed to handle MRS where the leader and follower agents have unconventional designs that could collide while navigating.

A possible solution to most of these challenging scenarios is the implementation of time-variant half-planes based constraints. This would significantly improve the flexibility of the

MRS and, on the other hand, increase drastically the computational cost.

3.4 State machine for variable prediction horizon

The aim of this thesis is to develop an algorithm that is suitable for real applications, operated in real-time and that is as efficient as possible. For this reason, new features are incorporated to reduce computational effort, with the aim of keeping the computational time for each step below the sample time of the system, namely $T_s = 0.1\ s$.

In the literature, all MPC solutions adopt a fixed prediction horizon length, which guarantees an easy-to-manage algorithm but often sacrifices performance in certain situations. For instance, in the context exposed in this thesis, when the MRS is far from obstacles, the obstacle avoidance constraints become unnecessary and can negatively impact computational effort. This is because a large number of non-linear constraints can be computationally expensive. At the same time, a longer prediction horizon is beneficial for planning the optimal path, primarily to guarantee a feasible follower path. On the other hand, when the MRS is near obstacles it needs to avoid collisions and maintain a bounded formation error with an appropriate algorithm execution time, so a shorter prediction horizon is preferable to meet these needs efficiently. A shorter horizon decreases the number of hard and soft constraints evaluated during each MPC iteration, as these constraints scale proportionally with the length of prediction horizon.
For this reason, a state machine composed of two states is implemented, each corresponding to the conditions described. The MRS state is selected according to the distance between leader or follower to their nearest obstacle surface. N_{long} is the prediction horizon used when system is far from obstacles, N_{short} is the shorter prediction horizon used to perform obstacle avoidance. The transition between these states is triggered by a distance r_d^* which is defined as a distance from the $* = L,F$ agent center. This trigger distance value represents the distance an agent, traveling at full speed, would cover before coming to a stop under maximum deceleration, plus the agent maximum radius r_{max}^*, adjusted by a safety factor $C_{rd} \in \mathbb{R}$. Formally, this is:

$$r_d^* = C_{r_d}\left(r_{max}^* + \frac{1}{2}\frac{(v_{max}^*)^2}{a_{max}^*}\right) \tag{3.44}$$

The trigger distance r_d^* can be interpreted as the radius of the obstacle-avoidance area (OA area), as reported in Figure 3.7. When at least one obstacle is inside the OA area of either the leader or follower, the N_{short} state is triggered, while if neither agent detects obstacles within their OA area, the N_{long} state is applied.

3.4.1 Perception range

As discussed in Section 1.1, a decentralized MPC is more scalable than a centralized MPC, but still has its limits. Numerical results presented in Section 4.2 show that a large amount of obstacles significantly reduce the performance of the algorithm, causing the computational time of the single step to exceed the system sample time T_s. In addition, computing obstacle avoidance for obstacles that are far from the MRS, considering their relative velocity, wastes computational power. To address this issue, a perception range is introduced to simulate the MRS ability to detect nearby obstacles through its sensors.
Perception range r_{pr} is defined relative to the center of the MRS, located in the load center for simplicity. It is defined as the maximum vertex-MRS center distance in the MRS, incremented by the safety factor $C_{pr} \in \mathbb{R}$:

$$r_{pr} = C_{pr} \left(\frac{d_{L,F}}{2} + max(r^L_{max}, r^F_{max}) \right) \tag{3.45}$$

So, r_{pr} can be interpreted as the radius of the perception area (Figure 3.7), that has to be a equal or wider region than the OA areas of both robots.

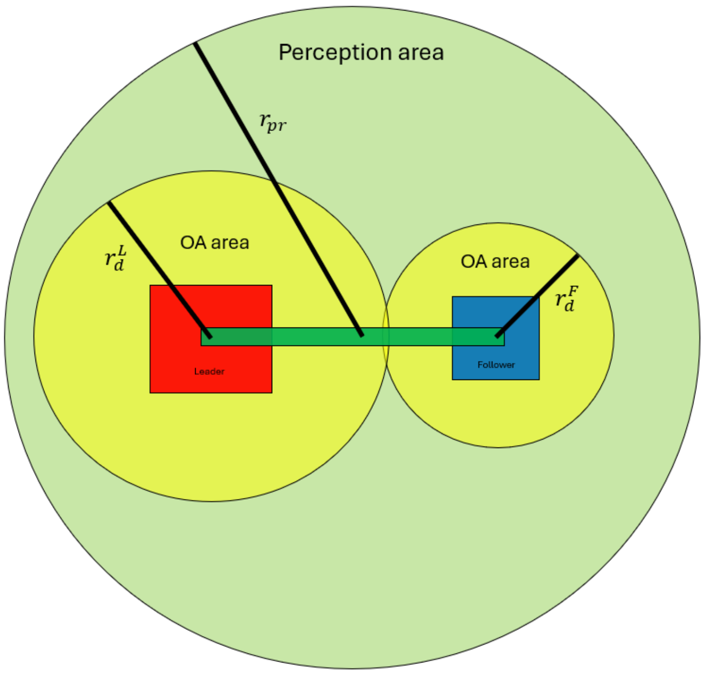


Figure 3.7: Perception area (in green) and OA areas (in yellow) of the MRS.

3.5 Discussion and remarks

This novel 3-DOF NLMPC algorithm is developed to enhance 2-DOF LMPC framework performance regarding computational cost, flexibility in cluttered environments and obstacle avoidance effectiveness. Consequently, hereafter improvements are listed:

1. UGVs dynamic model is extended to a 3-DOF, incorporating independent agents rotations with respect to the payload, so that robots can agilely move in more narrow spaces and cluttered environments, compared with the 2-DOF LMPC framework.

2. Potential repuslive field function ensures an higher safety of the framework, incrementing the distance between agents and obstacles during navigation.

3. Follower control effort functional cost reduces energy spent moving this agent and unnecessary accelerations. Obviously, this functional cost member needs to be weighted correctly to avoid interfering with proper follower task accomplishment.

4. Load vertices are incorporated to the definition of obstacle avoidance hard constraints, to guarantee collision avoidance even when payload has a particular shape.

5. Perception range emulating a sensing system is implemented, realizing a more realistic framework and reducing the number of obstacles sensed in each algorithm iteration. This feature causes a decreased computational effort.

6. Lastly, the state machine providing a variable prediction horizon framework is designed. When obstacles are far from the system, obstacle avoidance is not performed and a wider prediction horizon is used. On the other hand, when the system is approaching obstacles, a shorter prediction horizon is applied to perform obstacle avoidance, causing reduced execution time due to the decreased number of hard and soft constraints to be computed for obstacle avoidance.

Cons about the geometrical approach implemented in both 2-DOF LMPC and 3-DOF NLMPC frameworks are reported in Section 3.3.5.

Chapter 4

Non-Linear Model Predictive Control Framework for a 3 degrees of freedom Multi-Robot System: assessment and validation

In this chapter, MATLAB® simulations are performed to assess and validate the proposed transportation framework. Firstly, optimal parameters for 3-DOF NLMPC framework are selected through a detailed analysis. Then fixed and variable prediction horizon configurations are compared to evaluate the differences regarding both computational effort and formation error. Also, a comparison between 2-DOF LMPC and 3-DOF NLMPC frameworks is reported to prove the effectiveness of the improvements developed. The influence of the perception range is demonstrated, and finally, a Monte Carlo simulations campaign is performed to assess the robustness to disturbances of the 3-DOF NLPMC framework.

To simplify the simulations, it is important to note that input delays due to computation are not considered in this analysis. Specifically, the system remains frozen until the necessary inputs are computed and applied to the robots.

Simulations results can significantly change depending on the values assigned to certain variables. Some of them are the selected MPC parameters (as discussed in Section 4.1), while others are considered system's constants, such as velocity and acceleration boundaries for agents, agents dimensions, load dimension and so on. The constants used in the simulations are summarized below:

- Leader dynamic parameters:
  - maximum linear velocity: $v^L_{max} = 1\ m/s$.
  - maximum linear acceleration: $u^L_{max} = 3\ m/s^2$.
  - maximum angular velocity: $\omega^L_{max} = 1.57\ rad/s$.
  - maximum angular acceleration: $u^L_{\theta,max} = 10\ rad/s^2$.
- Follower dynamic parameters:
  - maximum linear velocity: $v^F_{max} = 1.5\ m/s$.
  - maximum linear acceleration: $u^F_{max} = 5\ m/s^2$.
  - maximum angular velocity: $\omega^F_{max} = 3.14\ rad/s$.
  - maximum angular acceleration: $u^F_{\theta,max} = 20\ rad/s^2$.

  Note that follower dynamic parameters are higher than leader's, in order to have a follower that can easily accomplish its task and efficiently follow the leader.
- MRS dimensional parameters:
  - Leader: square robot $0.6\ m \times 0.6\ m$, resulting in $r^L_{max} = 0.424\ m$.
  - Follower: square robot $0.3\ m \times 0.3\ m$, resulting in $r^F_{max} = 0.212\ m$.
  - Payload: rectangular load $1.0\ m \times 0.2\ m$, resulting in $d_{L,F} = 1\ m$.
- Distance threshold between the leader center and the final goal that determines if the simulation ended successfully: $\delta = 0.1\ m$.
- Steps of the prediction horizon, starting from the first prediction, where formation error is checked to decide if recovery policy is needed: $k_{loose\_grip} = 3$.
- Formation error that triggers recovery policy: $\epsilon_{loose\_grip} = 0.01\ m$.

In Section 4.1 a fixed prediction horizon $N = 10$ is used to study optimal parameters. Instead, C_{r_d}, defining OA area extension, requires a variable prediction horizon to be tested. $N_{long} = 15$, $N_{short} = 5$ is the chosen state machine configuration for the analysis of this parameter. In Section 4.2 the impact of prediction horizon is analyzed.
These MATLAB® simulations run on a laptop with Ubuntu 22.04, an Intel™ Core i7-1165G7 @ 2.80GHz $\times$ 8 processor, a RAM memory of 16.0 GB.

4.1 Optimal MPC parameters analysis

To achieve optimal performance for this algorithm, it is crucial to determine the optimal MPC parameters. Given the system's complexity and the presence of multiple functional costs, there are several MPC parameters to consider:

1. $W \in \mathbb{R}^{n\times n}$: leader state weight during prediction horizon. (eq. (2.19))
2. $R_L \in \mathbb{R}^{m\times m}$: leader control effort weight during prediction horizon. (eq. (2.19))
3. $Z \in \mathbb{R}^{n\times n}$: leader state weight on the last prediction horizon step. (eq. (2.19))
4. $R_F \in \mathbb{R}^{m\times m}$: follower control effort weight during prediction horizon. (eq. (3.24))
5. $C \in \mathbb{R}$: squared formation error weight. (eq. (2.28))
6. $\beta \in (0, 1] \subset \mathbb{R}$: decaying factor of the squared formation error during prediction horizon in J_1^F. (eq. (2.26))
7. $\lambda \in \mathbb{R}$: the parameter of the potential repulsive field that affects the functional cost weight of each obstacle according to their distance to the agent considered. (eq. (3.12))
8. $C_{pot} \in \mathbb{R}$: potential repulsive field weight. (eq. (3.12))
9. $C_{r_d} \in \mathbb{R}$: safety parameter that regulates the dimension of the OA area, where obstacle avoidance is performed. (eq. (3.44))
10. $C_{pr} \in \mathbb{R}$: safety parameter that regulates the dimension of the perception area. (eq. (3.45))

λ, C_{pot} and C_{r_d} may have different values for each agent, but they are considered equal for leader and follower in this thesis.
Given the large number of parameters involved, the simulations to select optimal parameters are divided into blocks aiming to analyze together parameters that have mutual effects on the same variables or functional costs. The blocks are designed as follows:

1. Leader navigation parameters. $\{W,\ R_L,\ Z\}$
2. Follower navigation parameter. $\{R_F\}$
3. Formation error parameters. $\{C,\ \beta\}$
4. Potential repulsive field parameters. $\{C_{pot},\ \lambda\}$
5. Potential repulsive field dimension. $\{C_{r_d}\}$

6. Perception range dimension. $\{C_{pr}\}$

The simulations are structured in a way that each block of parameters is tested in a controlled manner to evaluate their impact on the algorithm's behavior, while the other parameters are fixed. These fixed parameters for the simulations are reported in the first row of each block table, namely Tables 4.1, 4.3, 4.5, 4.7, 4.9 and 4.11.
Specifically designed environments, to cover every aspect of the tested algorithm, are used for these simulations to understand the generic behaviour of the algorithm. Each one of them has a specific purpose in terms of testing the impact of the parameters on the path planning and decision-making process of the MRS. The considered environments include:

- *No obstacles* environment. Obstacle avoidance is not required in this environment, so it is mostly useful to analyze how quickly the leader trajectory converges to the goal and how the follower responds. Obviously, it is unnecessary to test potential field parameters in this environment.
  Leader initial state is $\begin{bmatrix} 20 & 10 & 0 & 0 & 0 & 0 \end{bmatrix}^T m$ (Figure 4.1).

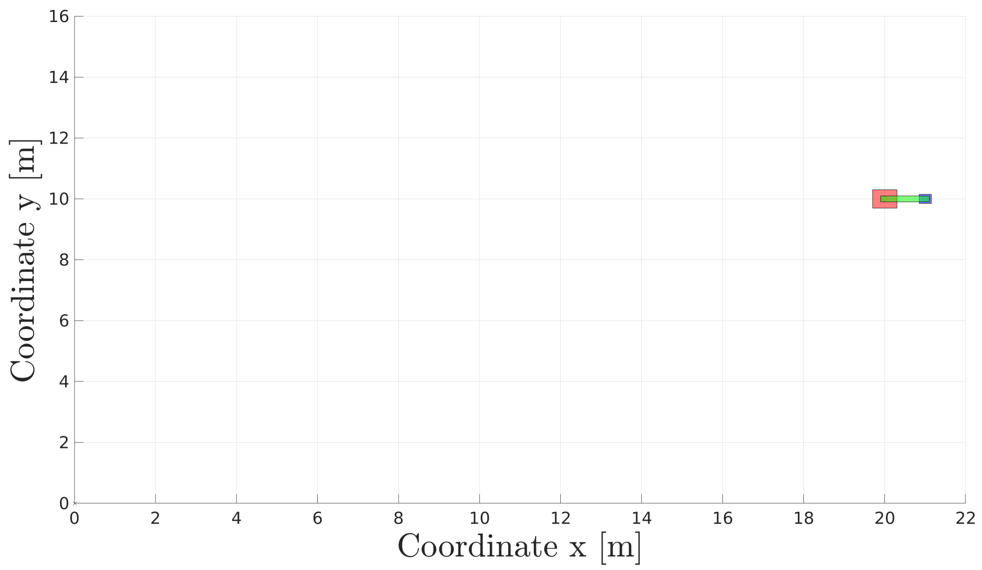


Figure 4.1: *No obstacles* environment.

- *One obstacle* environment. In order to assess how the algorithm responds to a potential singularity, the obstacle is placed along the ideal leader path.
  The static obstacle is placed at $\begin{bmatrix} 5 & 5 \end{bmatrix}^T m$, its radius is $r_1 = 3\ m$. The leader initial position is $\begin{bmatrix} 10 & 10 & 0 & 0 & 0 & 0 \end{bmatrix}^T m$ (Figure 4.2).

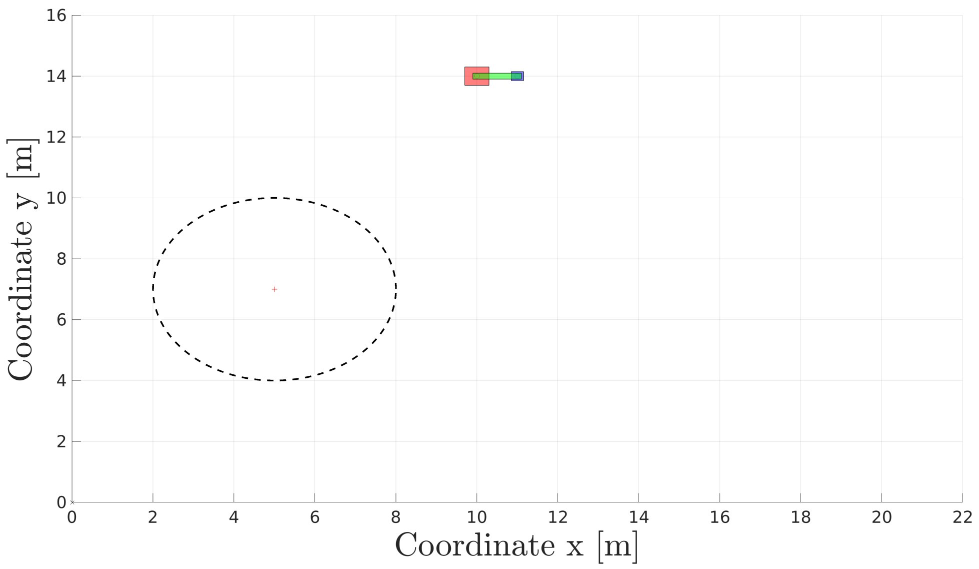


Figure 4.2: *One obstacle* environment.

- *Corridor* environment. This scenario involves multiple static obstacles arranged to create a narrow passage, providing an ideal scenario to assess obstacle avoidance performance. Additionally, the recovery policy may be activated.

  Obstacles placement and radii are:

Obstacle	Center position $[m]$	Radius $[m]$
$\mathbf{o_1}$	$\begin{bmatrix}7 & 7\end{bmatrix}^T$	0.8
$\mathbf{o_2}$	$\begin{bmatrix}10 & 7\end{bmatrix}^T$	0.8
$\mathbf{o_3}$	$\begin{bmatrix}7 & 6\end{bmatrix}^T$	0.8
$\mathbf{o_4}$	$\begin{bmatrix}10 & 6\end{bmatrix}^T$	0.8
$\mathbf{o_5}$	$\begin{bmatrix}7 & 5\end{bmatrix}^T$	0.7
$\mathbf{o_6}$	$\begin{bmatrix}10 & 5\end{bmatrix}^T$	0.8
$\mathbf{o_7}$	$\begin{bmatrix}6.5 & 4.5\end{bmatrix}^T$	0.8
$\mathbf{o_8}$	$\begin{bmatrix}7 & 8\end{bmatrix}^T$	0.8
$\mathbf{o_9}$	$\begin{bmatrix}9.5 & 4\end{bmatrix}^T$	0.7

  The leader initial position is $\begin{bmatrix}10 & 10 & 0 & 0 & 0 & 0\end{bmatrix}^T$ m (Figure 4.3).

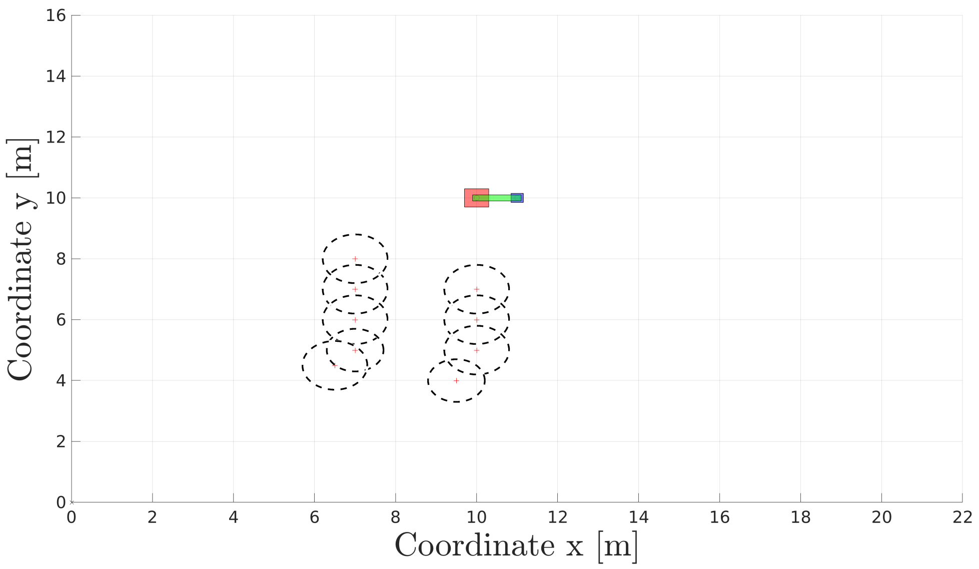


Figure 4.3: *Corridor* environment.

- *Obstacle near goal* environment. It tests the ability of the MRS to reach the goal in environments where obstacles are placed near the target, providing optimal choice of potential repulsive field parameters. For this reason, it is employed for potential repulsive field parameters block only. This obstacle is placed in $\begin{bmatrix} -2 & 0 \end{bmatrix}^T$ m, and its radius is $r_1 = 1\ m$ (Figure 4.4).

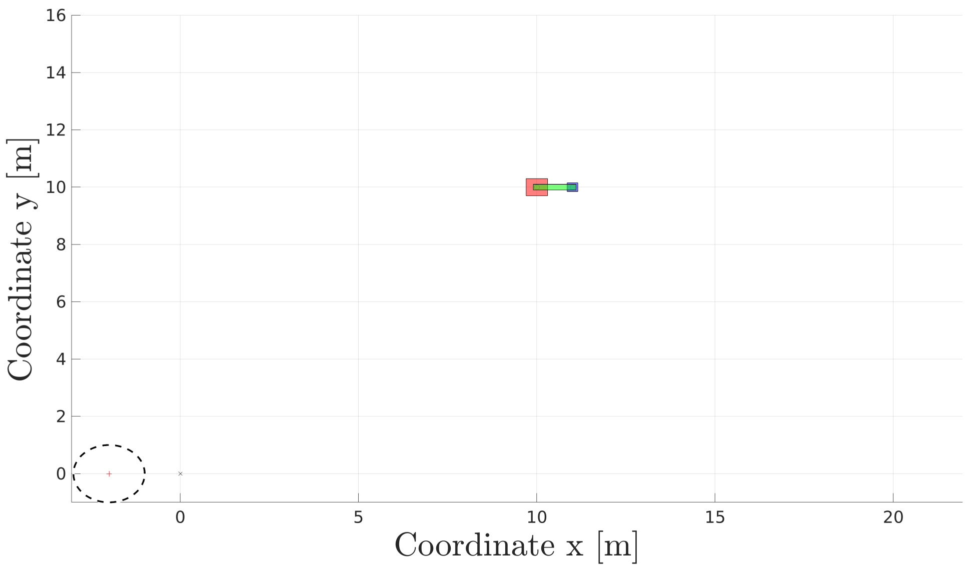


Figure 4.4: *Obstacle near goal* environment.

The criteria for selecting the best configuration of parameters are different, depending on the nature of the parameters. For example, simulations involving formation error parameters show a noticeable impact on formation error, whereas in other simulations, the differences are negligible. For this reason, only relevant data are presented.
Optimal set of parameters chosen for each block is highlighted in Tables 4.1, 4.3, 4.5, 4.7, 4.9 and 4.11.

4.1.1 Leader navigation parameters

Table 4.1: Leader navigation parameters tested.

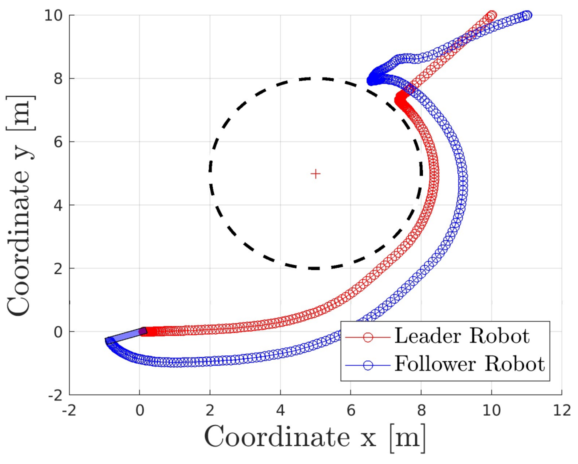
Set	W	R_L	Z
1	$diag([1, 1, 1, 1, 1, 1])$	$diag([0.9, 0.9, 0.9])$	$diag([1, 1, 1, 1, 1, 1])$
2	$diag([10, 10, 10, 10, 10, 10])$	$diag([1.5, 1.5, 10])$	$diag([10, 10, 10, 10, 10, 10])$
3	$diag([1, 1, 10, 1, 1, 10])$	$diag([1, 1, 10])$	$diag([100, 100, 100, 100, 100, 100])$
4	$diag([1, 1, 10, 1, 1, 10])$	$diag([0.9, 0.9, 0.9])$	$diag([500, 500, 100, 100, 100, 100])$

Total execution time and trajectory planned in these simulations are the significant results analyzed to choose the best set of parameters. *No obstacles* environment results for these simulations are similar, therefore they are not significant.

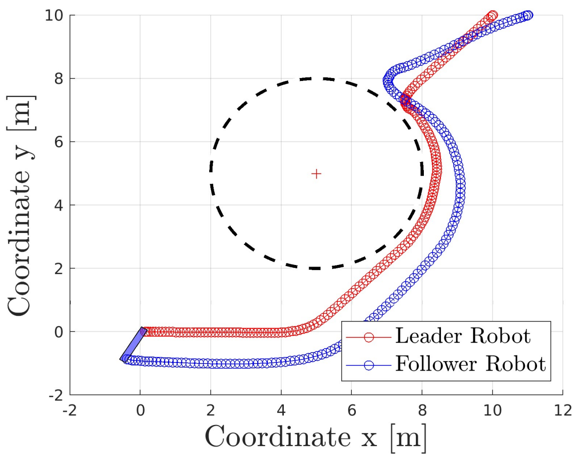
Table 4.2: Simulations data for leader navigation parameters.

Set	*One obstacle* total execution time [s]	*Corridor* total execution time [s]
1	44.126	85.054
2	46.041	101.681
3	29.709	83.881
4	**28.715**	**79.875**

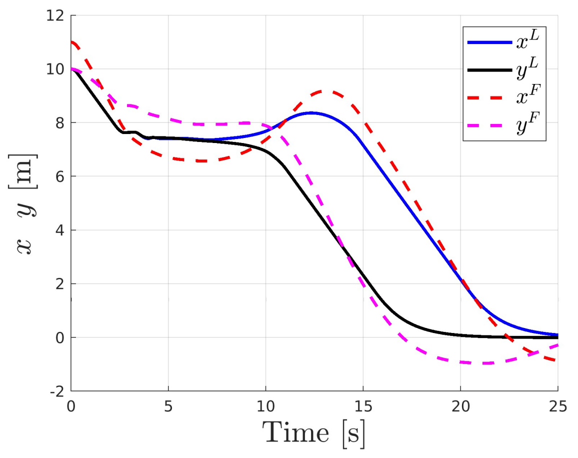
The trajectory difference between set couples $(1, 2)$ and $(3, 4)$ in *one obstacle* environment is significant. As shown in Figure 4.5, during the simulations with sets 1 and 2, the MRS remains stationary from $t = 4\ s$ to $t = 6.5\ s$, causing the increased execution time in the first two rows in Table 4.2, while with set 3 and 4 the MRS is able to plan a smoother path.
Collision avoidance is successfully guaranteed for all the simulation of these sets.



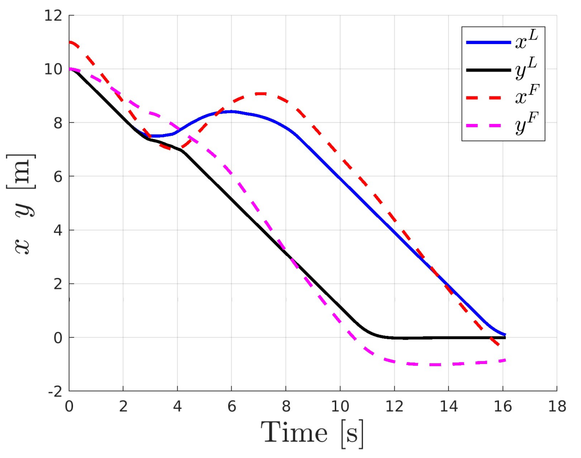
(a) Leader and follower planned trajectory, set 2



(b) Leader and follower planned trajectory, set 4



(c) Leader and follower x and y coordinates, set 2



(d) Leader and follower x and y coordinates, set 4

Figure 4.5: Trajectory and position coordinates comparison for leader navigation parameters in *one obstacle* environment.

4.1.2 Follower navigation parameter

Table 4.3: Follower parameters tested.

Set	R_F
1	$diag([5, 5, 10])$
2	$diag([10, 10, 20])$
3	$diag([1, 1, 5])$
4	$diag([1, 1, 1])$
5	$diag([0.5, 0.5, 0.5])$

In this case, only total execution time of the simulation is relevant. Table 4.4 shows that set 4 is generally the best one, even if set 5 performs significantly faster in environments with more obstacles. For this reason, a trade-off between performance and unnecessary movements is considered and set 4 is chosen.
Anyways, the trend shows that a low control effort weight improves the algorithm computational performance.
Collision is always avoided successfully and no significant formation error difference between sets is reported.

Table 4.4: Total execution time in follower parameters simulations.

Set	*No obstacles* [s]	*One obstacle* [s]	*Corridor* [s]
1	15.963	28.582	78.789
2	17.640	55.334	102.373
3	13.893	25.365	73.409
4	**13.033**	**22.977**	75.418
5	15.351	24.527	**55.168**

4.1.3 Formation error parameters

Table 4.5: Formation error parameters tested.

Set	C	β
1	5000	0.95
2	5000	1
3	500	0.97
4	50	1
5	5000	0.90

To evaluate formation error parameters performance, maximum and mean formation error are considered, as well as the total execution time (Table 4.6). Sets 3 and 4 are excluded because the MRS could not reach the goal in a reasonable number of steps, while other sets assure the MRS to reach the target avoiding obstacles.

Table 4.6: Formation error parameters simulations data.

Set	Environment	Max $\lvert fe_{L,F} \rvert$ [mm]	Mean $\lvert fe_{L,F} \rvert$ [mm]	Tot execution time [s]
1	*No obstacles*	7.343	0.247	15.567
2	*No obstacles*	**6.971**	**0.228**	**14.724**
5	*No obstacles*	7.735	0.268	15.432
1	*One obstacle*	7.341	1.085	29.992
2	*One obstacle*	**6.972**	**1.015**	30.943
5	*One obstacle*	13.405	1.241	**29.431**
1	*Corridor*	9.222	**1.253**	78.445
2	*Corridor*	10.915	1.823	161.258
5	*Corridor*	**7.725**	1.555	**77.411**

In environments where few obstacles are placed, set 2 has the best overall performance, but in *corridor* environment its behaviour is by far the worst. Thus, the combination of these results prove that set 1 is the optimal choice.

4.1.4 Potential repulsive field parameters

Table 4.7: Potential repulsive field parameters tested.

Set	C_{pot}	λ
1	1	1
2	7	1
3	7	3
4	15	3
5	15	10

Obstacle near goal environment is essential for potential repulsive field parameter optimal choice. Certain combinations of repulsive potential field parameters and obstacle proximity to the goal might prevent the MRS from reaching the target, causing it to stop at a distance determined by the repulsion parameters. However, the choice made in Table 4.1 of using an high terminal position weight on matrix Z mitigates the impact of this issue.

The key metric analyzed here is the minimum distance $d_{a,o}$ between an agent and an obstacle during the simulation. For *obstacle near goal* simulations the primary focus is on whether the

navigation successfully reaches the target (Table 4.8).
Since a reasonable safety margin for obstacle avoidance is desirable, set 5 is the optimal choice.

Table 4.8: Potential repulsive field simulation data.

Set	*One obstacle* $d_{a,o}$ [m]	*Corridor* $d_{a,o}$ [m]	*Obstacle near goal* success
1	0.000972	0.072410	**Yes**
2	0.023683	0.058267	No
3	0.048787	0.166017	**Yes**
4	**0.069657**	**0.302202**	No
5	0.069441	0.158751	**Yes**

All simulations of these sets achieve the same formation error performance.

4.1.5 Potential repulsive field dimension

Table 4.9: Potential repulsive field dimension parameters tested.

Set	C_{r_d}
1	2
2	3
3	4
4	5

Potential repulsive field dimension is relevant for variable prediction horizon algorithm only and does not have a direct impact on performance. However, a larger OA area increases the number of steps in which obstacle avoidance is active. The impact of this parameter become evident when combined with a prediction horizon length N analysis, as discussed in Section 4.2. So, in this situation, C_{r_d}, namely the safety factor regulating OA area radius, is chosen considering a reasonable distance at which obstacle avoidance becomes necessary, considering the ratio between this dimension and the maximum radius of the agent, for both agents.

Table 4.10: Potential repulsive field dimension simulation data.

Set	Leader r_d [m]	Leader r_d/r_{max}	Follower r_d [m]	Follower r_d/r_{max}
1	1.267	2.714	0.917	3.928
2	1.900	4.071	1.375	5.893
3	2.533	5.428	1.833	7.857
4	3.167	6.786	2.292	9.821

Given these robot dimensions and values of v_{max} and u_{max}, set 2 seems to ensure the correct safety margin for obstacle avoidance actuation (Table 4.10). As a matter of fact, collisions are avoided successfully. Formation error performance is not relevant in these simulations.

4.1.6 Perception range dimension

Table 4.11: Perception range dimension parameter tested.

Set	C_{pr}
1	4
2	5
3	6
4	7

These tests are conducted exclusively in the *corridor* environment to evaluate how many obstacles the system detects at each instant. The goal is to select the set with the most reasonable trade-off between safety and computational effort, as the computational load increases significantly when more obstacles are sensed.

Table 4.12: Perception range dimension simulation data.

Set	r_{pr} [m]	Tot execution time [s]
1	3.867	**65.457**
2	4.833	74.972
3	5.800	68.130
4	6.767	71.617

Table 4.12 shows that set 1 is the fastest configuration and presents a sufficient perception range distance, considering the path navigated (Figure 4.6). Figure 4.7 supports this intuition, showing that the number of obstacles sensed is reasonable for this setup. In general, all simulations reach the target correctly.

Table 4.13 groups the MPC optimal parameters chosen in this section.

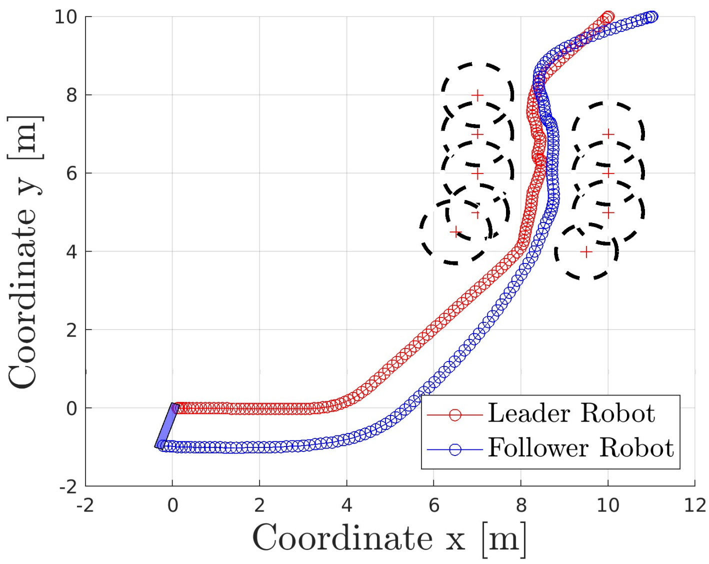
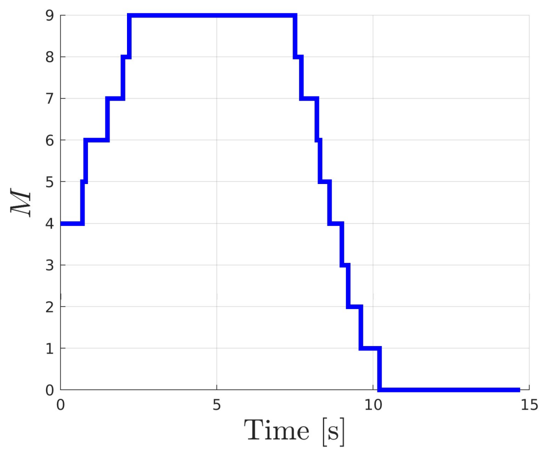


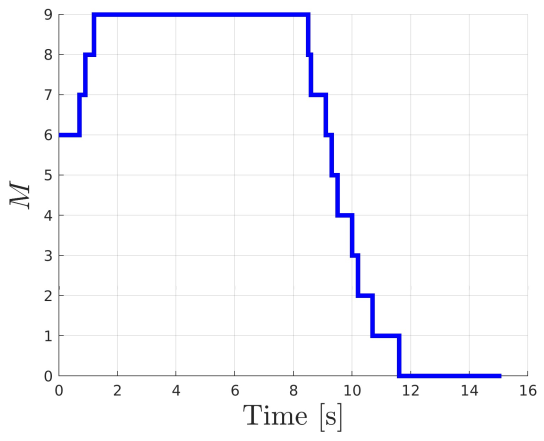
Figure 4.6: MRS trajectory navigated in *corridor* environment.

Table 4.13: Optimal parameters chosen in Section 4.1. These are set for all simulations in next sections.

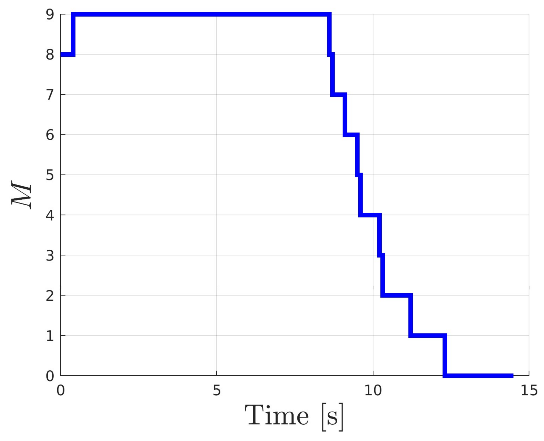
Parameter	Value
W	$diag([1, 1, 10, 1, 1, 10])$
R_L	$diag([0.9, 0.9, 0.9])$
Z	$diag([500, 500, 100, 100, 100, 100])$
R_F	$diag([1, 1, 1])$
C	5000
β	0.95
C_{pot}	15
λ	10
C_{r_d}	3
C_{pr}	4



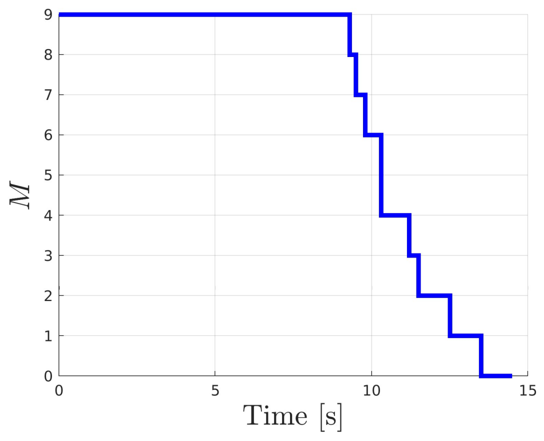
(a) Set 1



(b) Set 2



(c) Set 3



(d) Set 4

Figure 4.7: Number of obstacles M sensed at each step of the algorithm by the system in each simulation set.

4.2 Fixed vs variable prediction horizon

The aim of this section is to study fixed and variable prediction horizon performance to highlight the differences. The environments designed for these simulations are the same tested in [6]:

- *No obstacles*, the same used in Section 4.1 (Figure 4.1).
- *Two obstacles* (Figure 4.8), where

Obstacle	Center position $[m]$	Radius $[m]$
$\mathbf{o_1}$	$\begin{bmatrix}3 & 12\end{bmatrix}^T$	3
$\mathbf{o_2}$	$\begin{bmatrix}6 & 5\end{bmatrix}^T$	3

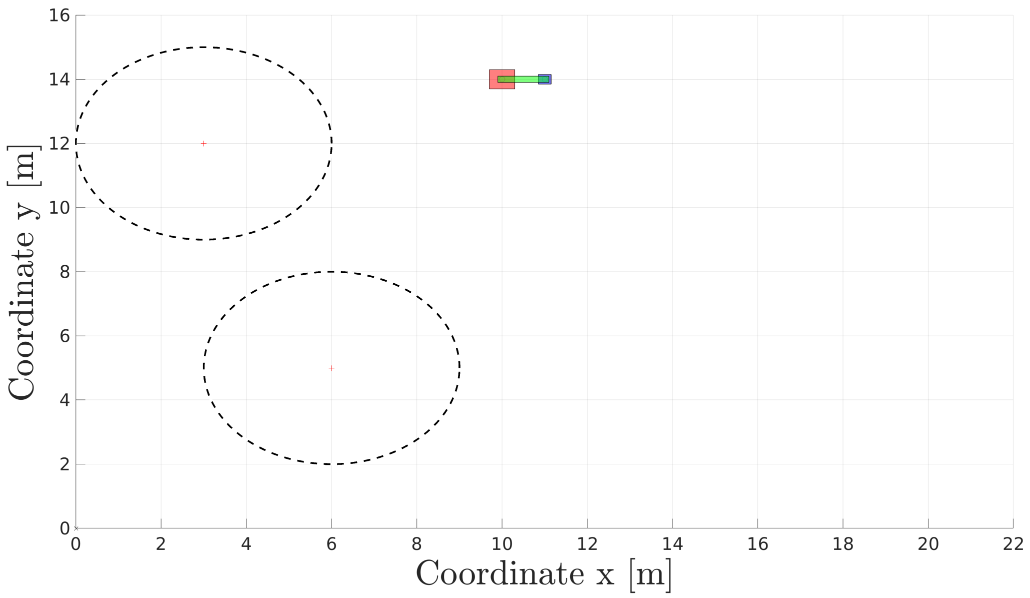


Figure 4.8: *Two obstacles* environment.

- *Three obstacles* (Figure 4.9), where

Obstacle	Center position $[m]$	Radius $[m]$
$\mathbf{o_1}$	$\begin{bmatrix}5 & 11\end{bmatrix}^T$	3
$\mathbf{o_2}$	$\begin{bmatrix}7 & 4\end{bmatrix}^T$	3
$\mathbf{o_3}$	$\begin{bmatrix}1 & 3\end{bmatrix}^T$	1

- *Valzer*, where 24 obstacles are placed to compose multiple corridors and narrow passages. An additional dynamic obstacle is implemented, that moves at $x = 12\ m$ with a $v_y = -1\ \frac{m}{s}$ (Figure 4.10).

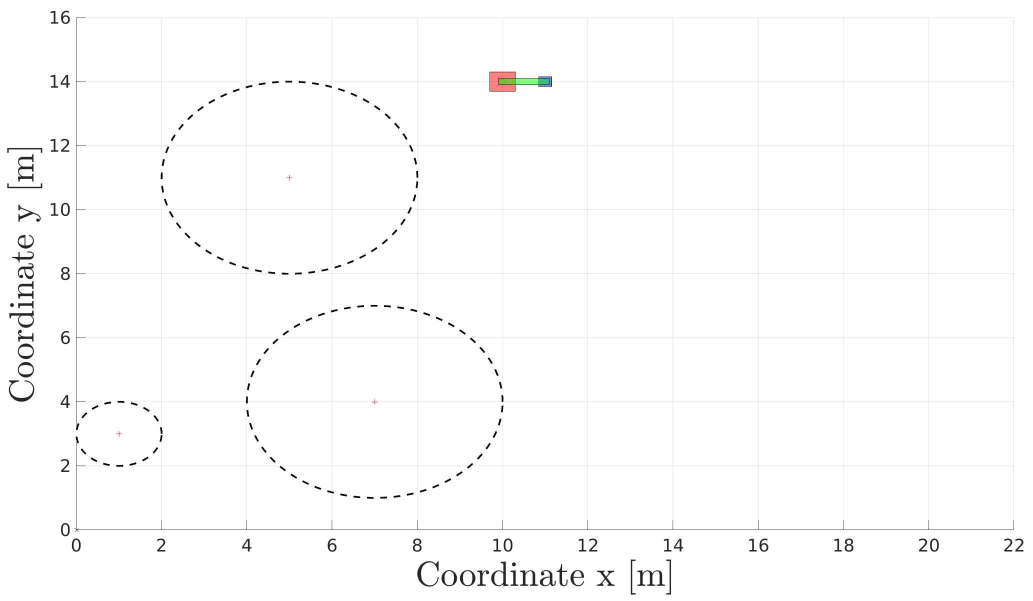


Figure 4.9: *Three obstacles* environment.

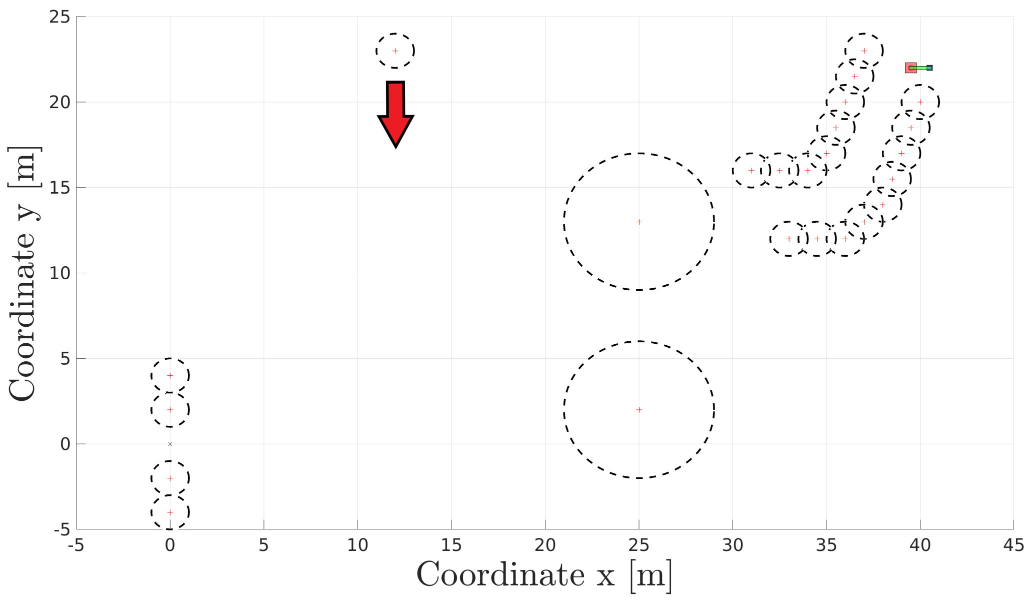


Figure 4.10: *Valzer* environment. The red arrow indicate the moving direction of the only dynamic obstacle in this environment. This image depicts the initial conditions of the environment.

Seven prediction horizon settings are tested, four with a fixed and three with a variable prediction horizon. In [6], only a $N = 20$ fixed horizon was tested.

- Fixed horizon, $N = 20$ (F20).
- Fixed horizon, $N = 15$ (F15).
- Fixed horizon, $N = 10$ (F10).
- Fixed horizon, $N = 5$ (F5).
- Variable horizon, $N_{long} = 20$, $N_{short} = 10$ (V20_10).
- Variable horizon, $N_{long} = 20$, $N_{short} = 5$ (V20_5).
- Variable horizon, $N_{long} = 15$, $N_{short} = 5$ (V15_5).

These values are selected to facilitate comparisons and anticipate potential real-time implementation. Each one of these configurations is tested in the four reported environments. In this section, the system detects all obstacles in the environment. Perception range is introduced later in Section 4.3. "ET" denotes the execution time, while $fe_{L,F}(t) = ||\mathbf{p^L}(t) - \mathbf{p^F}(t)|| - d_{L,F}$ is the formation error.
Tables 4.14, 4.15, 4.16, 4.17 present the results, where execution time is the primary difference between prediction horizon settings. In all the tests, the target is successfully reached.

Table 4.14: *No obstacles* environment fixed and variable prediction horizon data. Best result for each row is highlighted in green, while the worst is highlighted in red.

	F20	F15	F10	F5	V20_10	V20_5	V15_5
Max ET [s]	**0.598**	0.085	0.030	**0.009**	0.262	0.204	0.109
Min ET [s]	0.017	0.010	0.006	**0.003**	**0.021**	0.020	0.011
Mean ET [s]	0.034	0.016	0.009	**0.004**	**0.037**	0.033	0.019
Mean ET N_{long} [s]	–	–	–	–	**0.037**	0.033	**0.019**
Mean ET N_{short} [s]	–	–	–	–	–	–	–
Total ET [s]	7.117	3.372	1.805	**0.868**	**7.852**	6.934	3.909
Max $\|fe_{L,F}\|$ [mm]	3.158	**3.154**	3.155	**3.315**	3.158	3.158	**3.154**
Mean $\|fe_{L,F}\|$ [mm]	**0.068**	0.073	0.087	**0.250**	**0.068**	**0.068**	0.073

Formation error is generally below 10 mm, but in cluttered environments a peak formation error of up to 15 mm is possible. Different prediction horizon settings present similar formation error values on the same environment. In general, F20 selection results in the highest maximum and mean formation error. F5 selection results in the worst performance in *no obstacles* and *three obstacles* environments, likely due to the shorter prediction horizon used throughout the simulation, but performs better in other conditions. For this reason, it is difficult to find a pattern that explains the relation between formation error and length of prediction horizon.

Table 4.15: *Two obstacles* environment fixed and variable prediction horizon data. Best result for each row is highlighted in green, while the worst is highlighted in red.

	F20	F15	F10	F5	V20_10	V20_5	V15_5
Max ET [s]	**3.325**	1.535	0.392	**0.051**	0.375	0.220	0.100
Min ET [s]	**0.245**	0.147	0.080	**0.011**	0.025	0.013	0.015
Mean ET [s]	**1.002**	0.472	0.147	0.029	0.084	0.037	**0.025**
Mean ET N_{long} [s]	–	–	–	–	0.043	**0.044**	**0.022**
Mean ET N_{short} [s]	–	–	–	–	**0.135**	**0.028**	**0.028**
Total ET [s]	**159.2**	71.76	21.17	4.112	12.67	5.527	**3.678**
Max $\|fe_{L,F}\|$ [mm]	**11.553**	6.814	6.561	3.315	7.122	3.159	**3.154**
Mean $\|fe_{L,F}\|$ [mm]	**0.710**	0.522	0.390	0.378	0.314	**0.178**	0.202

Table 4.16: *Three obstacles* environment fixed and variable prediction horizon data. Best result for each row is highlighted in green, while the worst is highlighted in red.

	F20	F15	F10	F5	V20_10	V20_5	V15_5
Max ET [s]	**4.299**	2.223	0.556	**0.092**	0.652	0.205	0.095
Min ET [s]	0.206	**0.254**	0.067	0.022	0.034	0.024	**0.014**
Mean ET [s]	**1.676**	0.786	0.225	0.041	0.198	0.046	**0.040**
Mean ET N_{long} [s]	–	–	–	–	0.068	**0.082**	**0.038**
Mean ET N_{short} [s]	–	–	–	–	**0.212**	0.043	**0.040**
Total ET [s]	**310.0**	137.5	35.98	6.139	31.41	6.975	**5.936**
Max $\|fe_{L,F}\|$ [mm]	**8.949**	8.635	7.032	7.681	7.979	**5.600**	6.775
Mean $\|fe_{L,F}\|$ [mm]	1.210	**1.050**	1.139	**1.285**	1.189	1.163	1.242

Table 4.17: *Valzer* environment fixed and variable prediction horizon data. Best result for each row is highlighted in green, while the worst is highlighted in red.

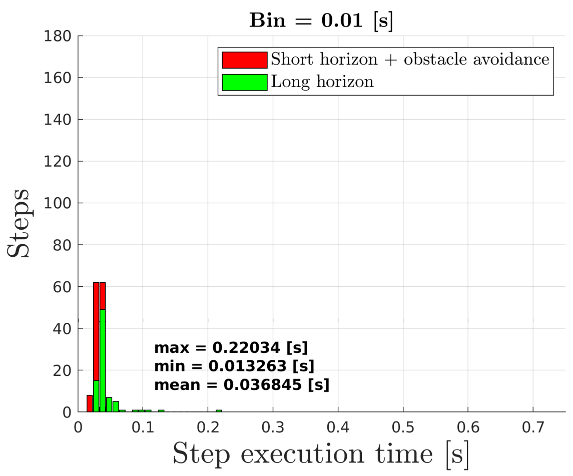
	F20	F15	F10	F5	V20_10	V20_5	V15_5
Max ET [s]	**1383**	288.6	3.218	0.745	3.215	0.744	**0.743**
Min ET [s]	**2.560**	1.531	0.559	0.054	0.016	0.017	**0.009**
Mean ET [s]	**17.16**	4.981	1.381	0.237	0.895	0.166	**0.161**
Mean ET N_{long} [s]	–	–	–	–	**0.029**	0.028	**0.016**
Mean ET N_{short} [s]	–	–	–	–	**1.471**	**0.259**	**0.259**
Total ET [s]	**8407**	2247	588.4	99.97	381.4	70.07	**67.83**
Max $\|fe_{L,F}\|$ [mm]	**15.341**	9.972	10.349	**9.708**	10.349	**9.708**	**9.708**
Mean $\|fe_{L,F}\|$ [mm]	**1.102**	0.763	0.496	0.710	**0.492**	0.682	0.677

Execution time significantly increases with the number of obstacles in the environment and the length of the prediction horizon. For instance, F20 selection in *valzer* environment leads to an unacceptable mean execution time, since sampling time of the system is $T_s = 0.1\ s$. A

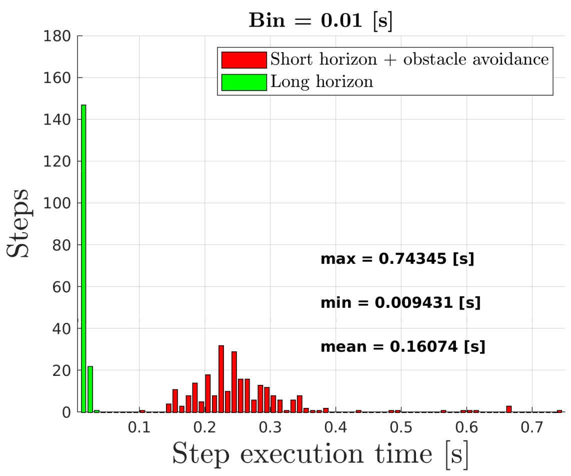
good result would be $Mean\_ET < T_s$, but none of these settings, with optimal parameters selected in Section 4.1, can guarantee this performance for all four environments. Actually, the only environment that exceeds this execution time limit is *valzer*, due to the high number of obstacles. In any other environment, at least a few settings have good results, such as F5, V20_5 and V15_5.
However, for a hard real-time algorithm, it is necessary for all steps to have $ET < T_s$, and these results indicate that none of the settings meet the condition $Max\_ET < Ts$ in all environments. Nevertheless, the incorporation of the perception range is expected to improve the computational performance (Section 4.3).
Here, time distribution of each MPC iteration execution time of setting V20_5 in *two obstacles* environment is represented in Figure 4.11a. When parameter N_{long} is high, there is much more possibility that long horizon state time steps take a longer time to compute than T_s, probably due to the number of hard constraints computed or formation error minimization, even if obstacle avoidance is not performed.
In general, the first step of the algorithm tends to be much more computationally expensive than subsequent steps, likely due to the intrinsic convergence challenges of the optimization problem. As a result, the execution time for the first step can vary significantly each time the simulation runs.
Moreover, in particularly cluttered environments with multiple obstacles, such as a corridor, a single step can be computationally intensive, increasing the maximum execution time value. For instance, in Figure 4.11b none of the N_{long} state time steps exceed T_s, while obstacle avoidance is clearly expensive and increases both maximum and mean execution time values.

Only one fixed prediction horizon configuration, *valzer* environment excluded, has an acceptable behaviour, with this parameters and environments setup, namely F5. V15_5 works well in *two obstacles* and *three obstacles* environments, but exceeds T_s in its first step only in *no obstacles* environment, as shown in Figure 4.11c.

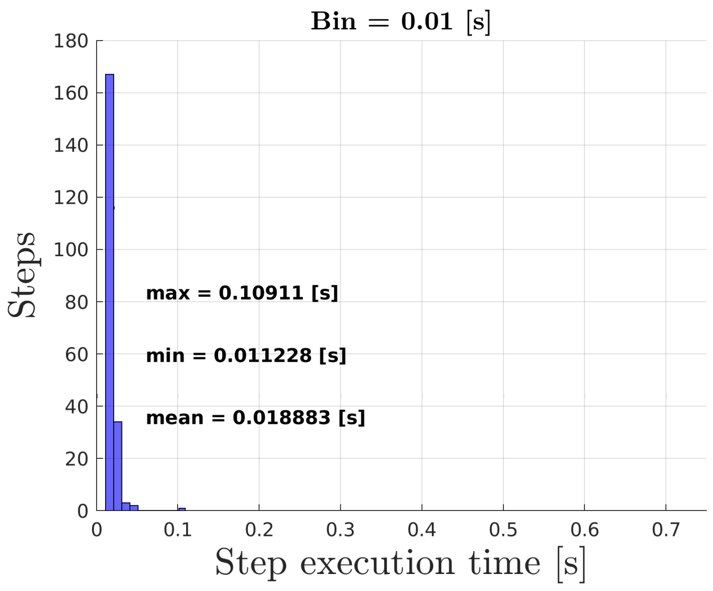
These results prove how a short fixed prediction horizon generally has good formation error performance and faster computation times. In contrast, a variable prediction horizon provides a broader prediction window, which can be crucial in more dynamic environments, given the correct combination of N_{long} and N_{short}. Also, this solution achieves a slightly better formation error performance.



(a) V20_5 in *two obstacles* environment.



(b) V15_5 in *valzer* environment.



(c) V15_5 in *no obstacles* environment.

Figure 4.11: Distribution of each MPC iteration execution time of various configurations in different environments. Histograms bin width is $0.01\ s$.

Regarding obstacle avoidance, the effectiveness is the same in all configurations, since no collision occurs. Table 4.18 shows the minimum agent-obstacle distance for each setting. This variable clearly depends on N, for fixed horizon, and N_{short}, for variable horizon, since in some simulations where $N = N_{short}$ the agents keep the same minimum distance. However, no specific trend is observed in the results.

Table 4.18: Minimum agent-obstacle distance. Highest distance for each environment, which means more safety, are highlighted in green, while smallest distances are highlighted in red.

Setting	*Two obs* [m]	*Three obs* [m]	*Valzer* [m]
F20	0.436	0.145	0.236
F15	0.581	**0.695**	0.109
F10	**0.154**	0.331	**0.032**
F5	0.589	**0.002**	**0.567**
V20_10	**0.154**	0.075	**0.032**
V20_5	**0.634**	0.005	**0.567**
V15_5	0.627	0.003	**0.567**

4.3 Perception range

A properly sized perception range can improve drastically the performance of this algorithm by reducing computational effort in environments with many obstacles. This is because it limits the number of obstacles sensed by the system, resulting in fewer collision avoidance hard and soft constraints. Tables 4.19, 4.20 and 4.21 demonstrate the improvement brought by the perception range introduction in environments with obstacles. Only settings that performed well without perception range are considered. Obviously, *no obstacles* environment simulations are not repeated, since perception range does not effect their results.

Table 4.19: *Two obstacles* environment with perception range. Best result for each row is highlighted in green, while the worst is highlighted in red.

	F5	V20_5	V15_5
Max ET [s]	**0.038**	**0.237**	0.110
Min ET [s]	**0.004**	0.009	**0.010**
Mean ET [s]	**0.019**	**0.036**	0.025
Mean ET N_{long} [s]	–	**0.044**	**0.023**
Mean ET N_{short} [s]	–	0.027	0.027
Total ET [s]	**2.690**	**5.422**	3.683
Max $\lvert fe_{L,F}\rvert$ [mm]	**3.315**	3.159	**3.154**
Mean $\lvert fe_{L,F}\rvert$ [mm]	**0.374**	**0.177**	0.201

Similar to simulations in Section 4.2, formation error values are consistent across the settings, but we can observe that variable prediction horizon settings have a slightly better performance, considering all environments. F5 mean formation error is between 1.05 and 3.68 times bigger than mean formation error of V20_5 and between 1.05 and 3.42 times bigger than mean formation error of V15_5. No particular difference is reported between formation error performance with and without perception range.

Table 4.20: *Three obstacles* environment with perception range. Best result for each row is highlighted in green, while the worst is highlighted in red.

	F5	V20_5	V15_5
Max ET [s]	**0.070**	**0.205**	0.100
Min ET [s]	**0.012**	**0.014**	**0.014**
Mean ET [s]	**0.028**	**0.035**	0.030
Mean ET N_{long} [s]	–	**0.082**	**0.042**
Mean ET N_{short} [s]	–	**0.031**	**0.029**
Total ET [s]	**4.113**	**5.255**	4.534
Max $\|fe_{L,F}\|$ [mm]	5.212	**5.809**	**5.111**
Mean $\|fe_{L,F}\|$ [mm]	**1.281**	1.173	**1.152**

Table 4.21: *Valzer* environment with perception range. Best result for each row is highlighted in green, while the worst is highlighted in red.

	F5	V20_5	V15_5
Max ET [s]	**0.303**	0.311	**0.319**
Min ET [s]	**0.003**	**0.009**	**0.009**
Mean ET [s]	**0.045**	**0.054**	0.050
Mean ET N_{long} [s]	–	**0.029**	**0.016**
Mean ET N_{short} [s]	–	**0.071**	**0.073**
Total ET [s]	**19.11**	**22.70**	21.03
Max $\|fe_{L,F}\|$ [mm]	7.073	7.073	7.073
Mean $\|fe_{L,F}\|$ [mm]	**0.631**	0.601	**0.597**

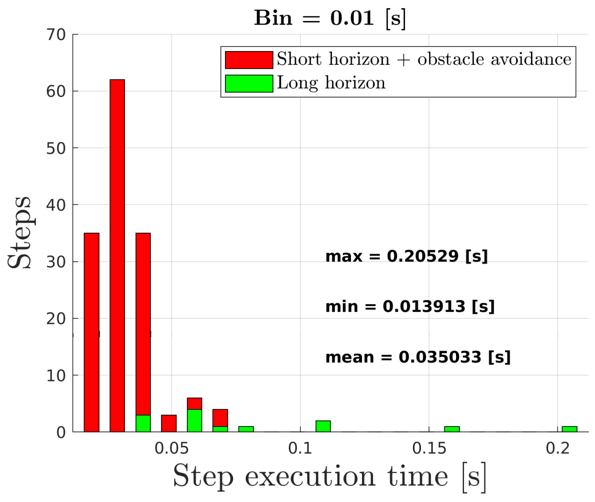
Data demonstrate that the perception range highly decreases computational effort when the environment presents lots of obstacles spread over a large area, such as *valzer* environment, while it is less effective for more bounded environments, where there are few obstacles permanently inside the perception range of the system, such as *two obstacles* environment.
Table 4.22 shows that F5 achieves the biggest improvement, considering that this setting always performs obstacle avoidance, while variable prediction horizon setups almost have no advantage in *two obstacles* environment, for the reason explained before, but exhibit substantial improvement in *valzer* environment.

Thanks to perception range, each configuration now has a mean execution time compatible with T_s, but the maximum execution time issue is still affecting this algorithm. Mean execution times for both N_{long} and N_{short} states in variable configuration are below T_s, suggesting that the problem stems from a few computationally intensive steps that exceed the sample time.

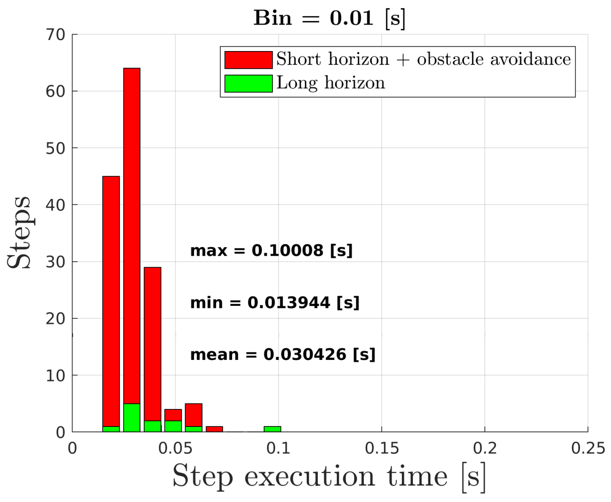
Table 4.22: Mean execution time reduction for each setting provided by perception range.

Setting	*Two obs*	*Three obs*	*Valzer*
F5	34.57%	32.56%	80.64%
V20_5	1.90%	24.66%	67.61%
V15_5	0.00%	23.63%	69.00%

Figure 4.12 shows that this problem is caused by a too high N_{long} value for V20_5 and a peak execution time in the first step of the optimization problem for V15_5 in *three obstacles* environment. Since *Two obstacles* environment behaves similarly, the explanation remains the same.



(a) V20_5.



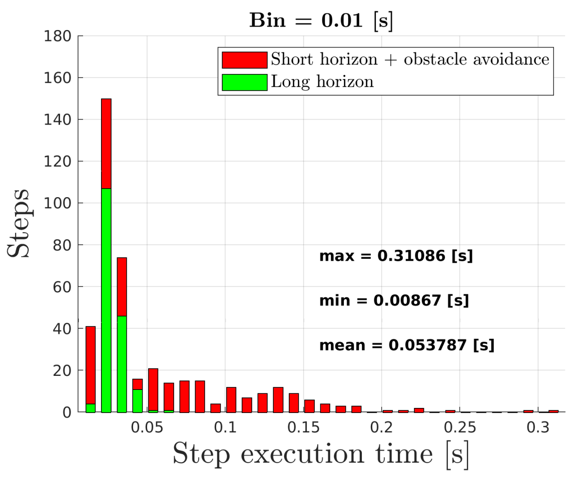
(b) V15_5.

Figure 4.12: Distribution of each MPC iteration execution time of variable configurations in *three obstacles* environment.

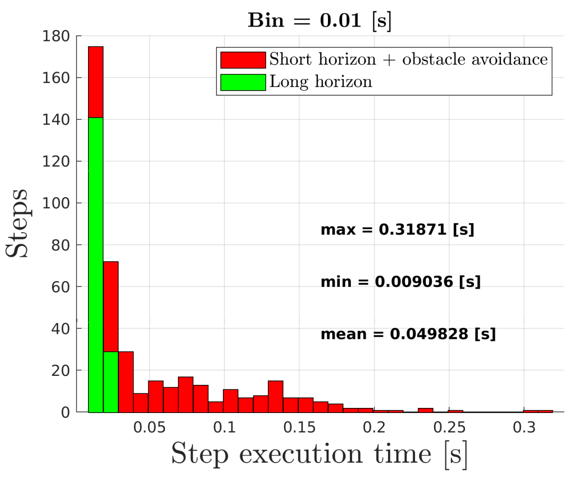
Valzer environment execution time data, shown in Figure 4.13, prove that, even though mean execution time while performing obstacle avoidance is below T_s, many steps require $ET > T_s$, in all three configurations. This suggests that, with these sets of N_{long} and N_{short}, this algorithm may be ineffective for hard real-time applications with the chosen rate, particularly when a large number of obstacles are sensed simultaneously (Figure 4.14).
The potential solutions are:

- increase T_s, so that all steps are computed within the sample time.
- reduce N, N_{long} and N_{short} until all steps are computed within the desired time, yet considering the trade-off with formation error performance and the possible impact of computationally critical steps caused by a too short prediction horizon.

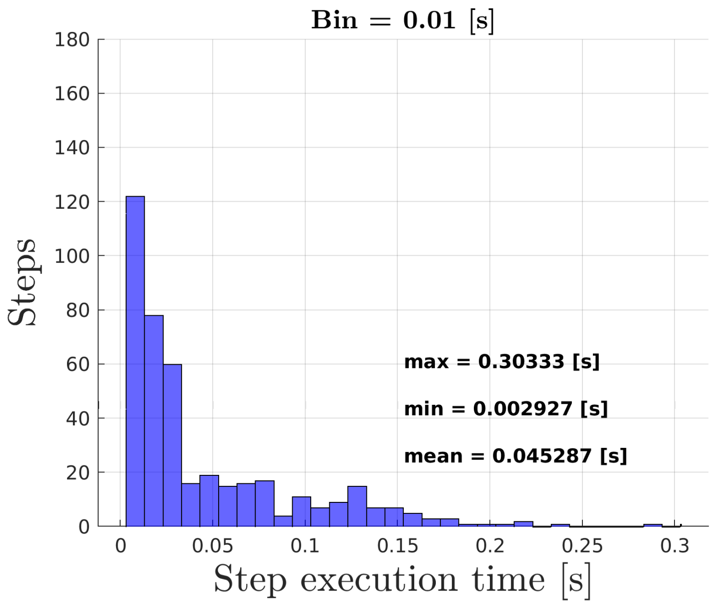
In conclusion, execution time of variable prediction horizon configurations is slightly higher. Nevertheless, we expect that increasing sample time to ensure the correct real-time behaviour of V15_5 should not have a significant impact on 3-DOF NLMPC framework's performance with respect to F5. This holds because in *valzer* environment the difference between peak execution time of F5 and V15_5 configurations is negligible, and in other environments the sample time increase required by V15_5 selection is not significant. Moreover, formation error in variable prediction horizon configurations is lower. Thus, the configuration V15_5 is selected as the best overall performance.



(a) V20_5 selection.

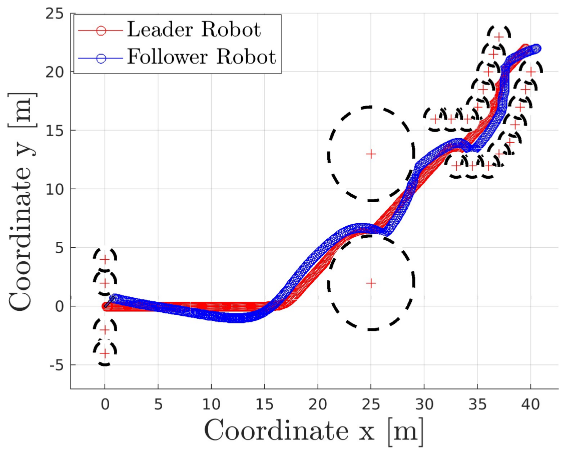


(b) V15_5 selection.

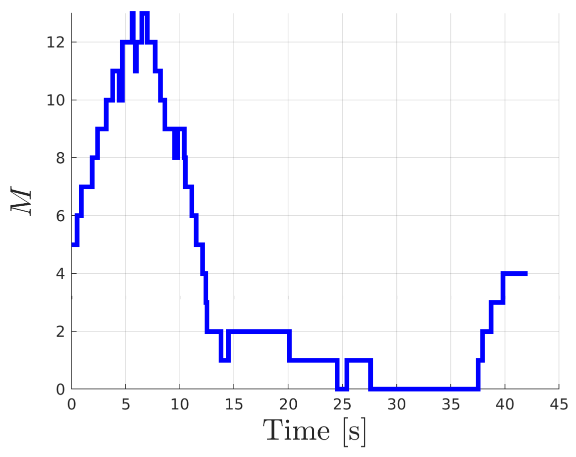


(c) F5 selection.

Figure 4.13: Distribution of each MPC iteration execution time in *valzer* environment.



(a) Trajectory navigated.



(b) Number of obstacles M sensed.

Figure 4.14: Trajectory navigated by the MRS in *valzer* environment and number of obstacles sensed.

4.4 2-DOF LMPC and 3-DOF NLMPC frameworks comparison

The best comparison between 2-DOF LMPC and 3-DOF NLMPC frameworks should focus on both formation error and computational time performance. Both algorithms use the same optimal parameters (Section 4.1). 2-DOF LMPC algorithm is run by selecting $N = 20$ and it is compared with 2 configurations of 3-DOF NLMPC framework: F20 configuration, which uses an equivalent MPC prediction horizon length, and V15_5 configuration, the best overall performance configuration for the 3-DOF NLMPC algorithm. Optimal perception range is used by the 3-DOF NLMPC during these simulations.
Table 4.23 show how the 2-DOF LMPC algorithm has better formation error performance, both in terms of maximum and mean values. The explanation could be that follower functional cost in 2-DOF LMPC framework is primarily focused on formation error optimization, while in 3-DOF NLMPC framework follower functional cost incorporates potential repulsive field function, which also leads to an optimization of the distance to obstacles. Figure 4.15 displays a formation error comparison in *valzer* environment. At the same time Table 4.24 proves that 2-DOF LMPC is far less efficient than 3-DOF NLMPC V15_5 configuration in all environments. So, the 3-DOF NLMPC framework presents a decreased formation error performance compensated by a great improvement on computational effort.
3-DOF NLMPC is preferred to the 2-DOF LMPC considering the enhanced obstacle avoidance performance and flexibility demonstrated and the reduced computational effort. In fact, 2-DOF LMPC presents an excessive execution time that cannot be solved by an increased sam-

ple time because it would lead to an insufficient path update frequency, causing the algorithm to be unreliable. Instead, formation error can be reduced acting on manipulator, mounted on the robots base, and on end-effector. The solutions are either controlling the manipulator mounted on agents base to move the end-effector or designing a passive end-effector that can deal with formation error without damaging the payload.

Table 4.23: Formation error performance comparison between 2-DOF LMPC and 3-DOF NLMPC algorithms in different environments. Best results for each environment are highlighted in green, while worst values are highlighted in red.

Environment	Configuration	Max $\lvert fe_{L,F}\rvert$ [mm]	Mean $\lvert fe_{L,F}\rvert$ [mm]
No obstacles	2-DOF F20	**0.787**	**0.075**
	3-DOF F20	**3.158**	**0.068**
	3-DOF V15_5	3.154	0.073
Two obstacles	2-DOF F20	**0.633**	**0.094**
	3-DOF F20	**12.293**	**0.897**
	3-DOF V15_5	3.154	0.201
Three obstacles	2-DOF F20	**0.985**	**0.187**
	3-DOF F20	**12.582**	**1.571**
	3-DOF V15_5	5.111	1.152
Valzer	2-DOF F20	**1.352**	**0.128**
	3-DOF F20	**11.687**	**0.971**
	3-DOF V15_5	7.073	0.597

Table 4.24: Computational time comparison between 2-DOF LMPC and 3-DOF NLMPC algorithms in different environments.

Environment	Configuration	Max ET [s]	Mean ET [s]	Tot ET [s]
No obstacles	2-DOF F20	**0.816**	**0.071**	**15.095**
	3-DOF F20	0.579	0.034	7.102
	3-DOF V15_5	**0.089**	**0.019**	**3.884**
Two obstacles	2-DOF F20	0.830	0.110	16.930
	3-DOF F20	**3.561**	**0.644**	**102.402**
	3-DOF V15_5	**0.110**	**0.025**	**3.683**
Three obstacles	2-DOF F20	0.869	0.150	24.466
	3-DOF F20	**4.933**	**1.189**	**221.095**
	3-DOF V15_5	**0.100**	**0.030**	**4.534**
Valzer	2-DOF F20	1.013	0.271	117.809
	3-DOF F20	**90.752**	**2.834**	**1362.964**
	3-DOF V15_5	**0.319**	**0.050**	**21.028**

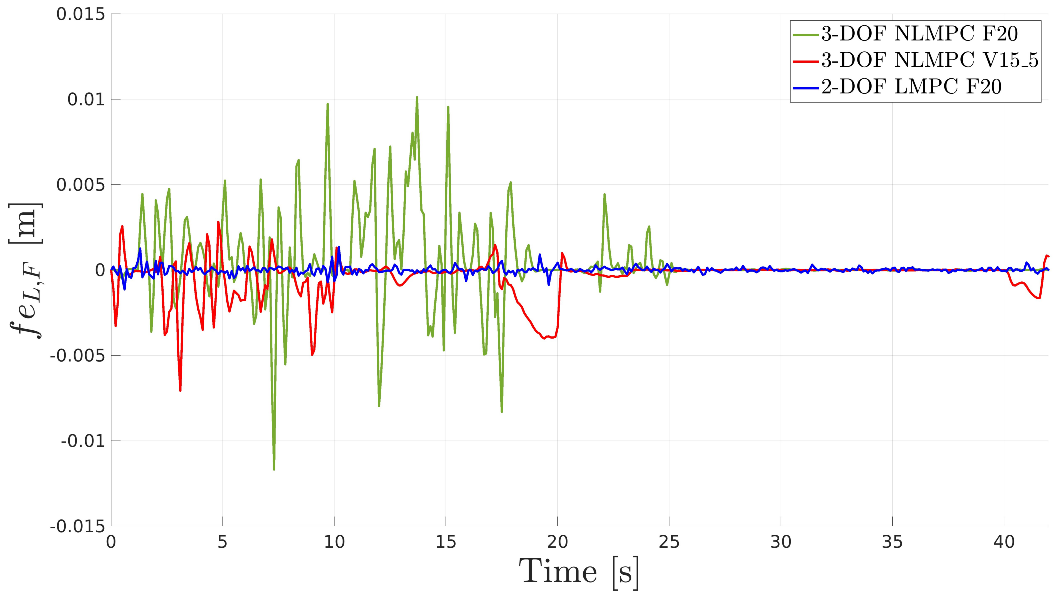


Figure 4.15: Formation error comparison between 2-DOF LMPC and 3-DOF NLMPC algorithms in *valzer* scenario.

4.5 Monte Carlo analysis

In this section, the assumptions made in Section 2.1 regarding ideal low-level controllers and measurement system are relaxed. These are now considered non-ideal, where the measurement system is affected by additive Gaussian noise, $\mathbf{n}(t)$, which represents both process noise (due to low-level controller inaccuracies) and measurement noise. The control scheme is shown in Figure 4.16.

Additive Gaussian noise $\mathbf{n}(t) \in \mathbb{R}^n$ only affects the measurements of the position of agents center, $\mathbf{p^L}(t)$ and $\mathbf{p^F}(t)$. Its mean value is 0 and its covariance is σ^2:

$$\mathbf{n}(t) = \begin{bmatrix} n(t) & n(t) & 0 & 0 & 0 & 0 \end{bmatrix}^T \qquad n(t) \sim \mathcal{N}(0, \sigma^2) \tag{4.1}$$

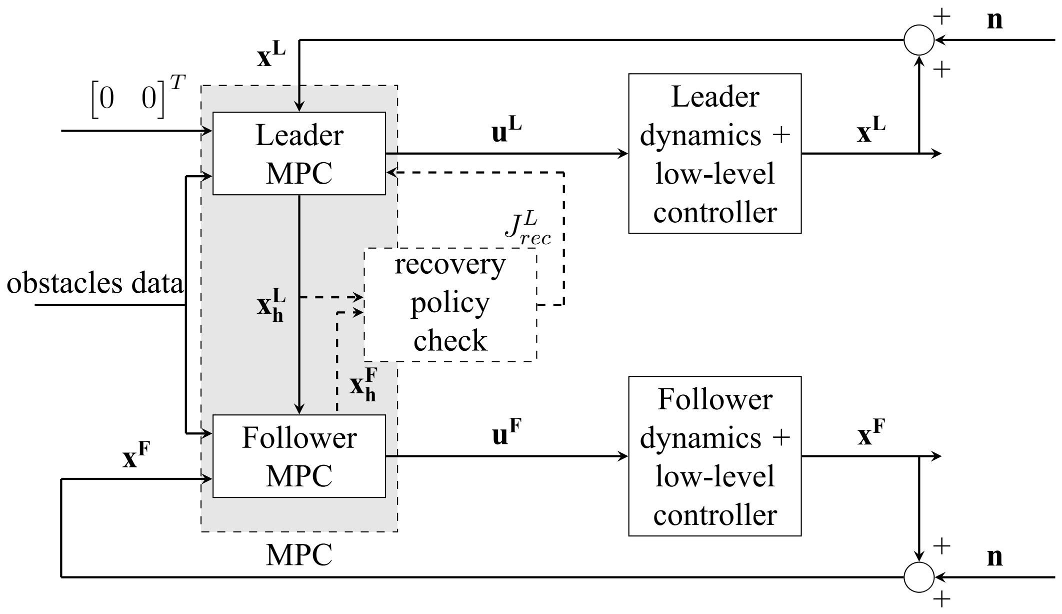


Figure 4.16: Control scheme with measurement noise inside the control loop.

To evaluate a realistic covariance σ^2, indoor localization systems accuracy is considered. For instance, Ultra-WideBand (UWB) is one of the most used systems and presents an accuracy of few centimeters, with particularly narrow environments and additional technologies, or between $10 - 15\ cm$ in more common applications in environments with a maximum extension of $50\ m$, similar to the environments dimensions considered in this thesis ([13] and [14]). It is also considered a second indoor localization system with an assumed accuracy between $5\ mm$ and $2\ cm$, such as Motion Capture.

The Monte Carlo analysis is conducted in all four environments introduced in Section 4.2, with the configuration 3-DOF NLMPC V15_5, repeated $N_{sims} = 25$ times for each combination of noise and environment. The success rate of the simulations is clearly the most important result to study, as it takes into consideration both the fact that leader reaches the goal within a certain tolerance δ and the fact that collisions are successfully avoided. In addition, average and standard deviation values of variables already considered in previous sections are reported to depict the performance of the algorithm in presence of disturbances.

4.5.1 UWB accuracy simulations

A set of three covariance values, chosen such that $n(t)$ lays inside the accuracy bounds with a 99.7% probability, is tested to show the robustness to disturbances of the system. (Table 4.25)

Table 4.25: First set of additive Gaussian noise tested.

Noise	Accuracy [cm]	σ^2 $[m^2]$
N6	6	0.02^2
N10	10	0.033^2
N15	15	0.05^2

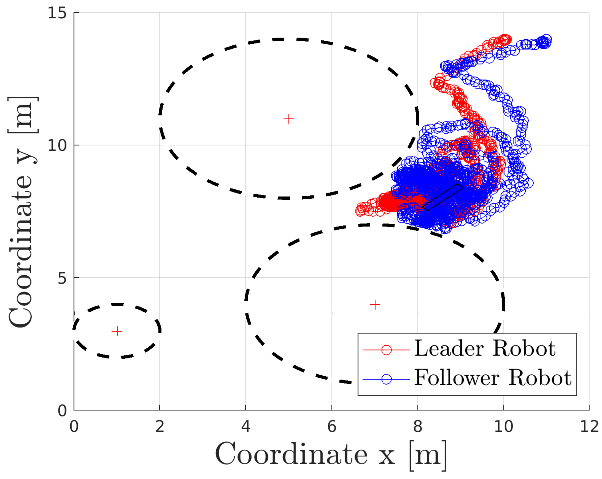
The MPC variable ϵ_{loose_grip} is the recovery policy formation error trigger value that determines when the formation error becomes critical and recovery policy needs to be actuated. Since noise is affecting agents position, formation error will also be noisy and with the same magnitude as accuracy. Depending on the ϵ_{loose_grip} value, recovery policy will be executed with a different rate.

Constant $\epsilon_{loose\_grip} = 0.01\ m$

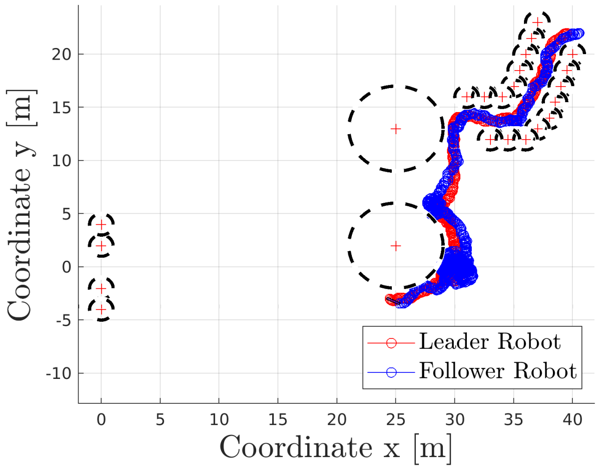
Table 4.26 shows that simpler environments, without obstacles or with large passages, consistently allow the system to converge to the goal without failure. In *three obstacles* environment, a particularly narrow passage does not allow the MRS to reach the goal, since the agents seems to be unable to satisfy constraints. It is also clear that if covariance increases, the success rate decreases significantly. In *valzer* environment there are no passages as narrow as in *three obstacles*, resulting in a higher success rate.

Table 4.26: Percentage of simulations where the goal is reached, collision avoidance rate and success rate, which is the combination of goal reached and collisions avoided successfully, with $\epsilon_{loose\_grip} = 0.01\ m$.

Noise	Environment	Goal reached	Collision avoided	Success rate
N6	*No obstacles*	100%	-	100%
	Two obstacles	100%	100%	100%
	Three obstacles	100%	100%	100%
	Valzer	100%	100%	100%
N10	*No obstacles*	100%	-	100%
	Two obstacles	100%	100%	100%
	Three obstacles	72%	92%	72%
	Valzer	100%	96%	96%
N15	*No obstacles*	100%	-	100%
	Two obstacles	100%	100%	100%
	Three obstacles	40%	60%	32%
	Valzer	96%	80%	80%



(a) *Three obstacles* environment failure.



(b) *Valzer* environment failure.

Figure 4.17: Examples of algorithm failure in presence of noise, where the MRS cannot reach the goal.

An analysis on mean and peak formation error values is conducted, so the average and standard deviation values among N_{sims} simulations of these parameters are reported in Tables 4.27 and 4.28. Formation error is bounded in all simulations, but its magnitude scales linearly with noise covariance. The maximum formation error is generally about twice the related accuracy, while the mean formation error is typically around half of the accuracy (Table 4.27). Figure 4.18 show the formation error comparison between different noise configurations. Moreover, both maximum and mean formation error are similar across all four environments, since the noise is significantly higher, at least ten times greater, than algorithm formation error performance without disturbance (Section 4.3).

Table 4.27: Maximum and mean $fe_{L,F}$ average values among N_{sims} simulations for each noise setting in different environments with $\epsilon_{loose\_grip} = 0.01\ m$. Highest value is highlighted in red and the lowest in green for each noise configuration.

Noise	Environment	Max $\lvert fe_{L,F}\rvert$ [m]	Mean $\lvert fe_{L,F}\rvert$ [m]
N6	*No obstacles*	0.101960	**0.026493**
	Two obstacles	**0.098156**	0.027092
	Three obstacles	0.107750	0.027390
	Valzer	**0.113644**	**0.028212**
N10	*No obstacles*	**0.179787**	**0.047795**
	Two obstacles	0.179812	0.047056
	Three obstacles	**0.199276**	**0.046895**
	Valzer	0.191798	0.047338
N15	*No obstacles*	0.307955	0.075279
	Two obstacles	**0.303027**	**0.073747**
	Three obstacles	0.337092	0.075050
	Valzer	**0.345605**	**0.076812**

Table 4.28: Maximum and mean $fe_{L,F}$ standard deviation values among N_{sims} simulations for each noise setting in different environments with $\epsilon_{loose\_grip} = 0.01\ m$.

Noise	Environment	Max $\lvert fe_{L,F}\rvert$ [m]	Mean $\lvert fe_{L,F}\rvert$ [m]
N6	*No obstacles*	0.011241	0.001538
	Two obstacles	0.013421	0.001883
	Three obstacles	0.016644	0.001815
	Valzer	0.010239	0.001306
N10	*No obstacles*	0.022123	0.003339
	Two obstacles	0.029875	0.003201
	Three obstacles	0.024222	0.002477
	Valzer	0.029118	0.001719
N15	*No obstacles*	0.032893	0.004726
	Two obstacles	0.044856	0.004502
	Three obstacles	0.050534	0.002779
	Valzer	0.047834	0.004627

Formation error values reported in Table 4.27 are almost always above recovery policy trigger limit $\epsilon_{loose\_grip} = 0.01\ m$, so inevitably recovery policy is adopted very often in these simulations. This is the reason why some simulations do not reach the goal, because leader recovery policy functional cost J^L_{rec} causes the leader to slow down and plan a more feasible path for the follower, while only J^L has the task to guide the leader to the goal.

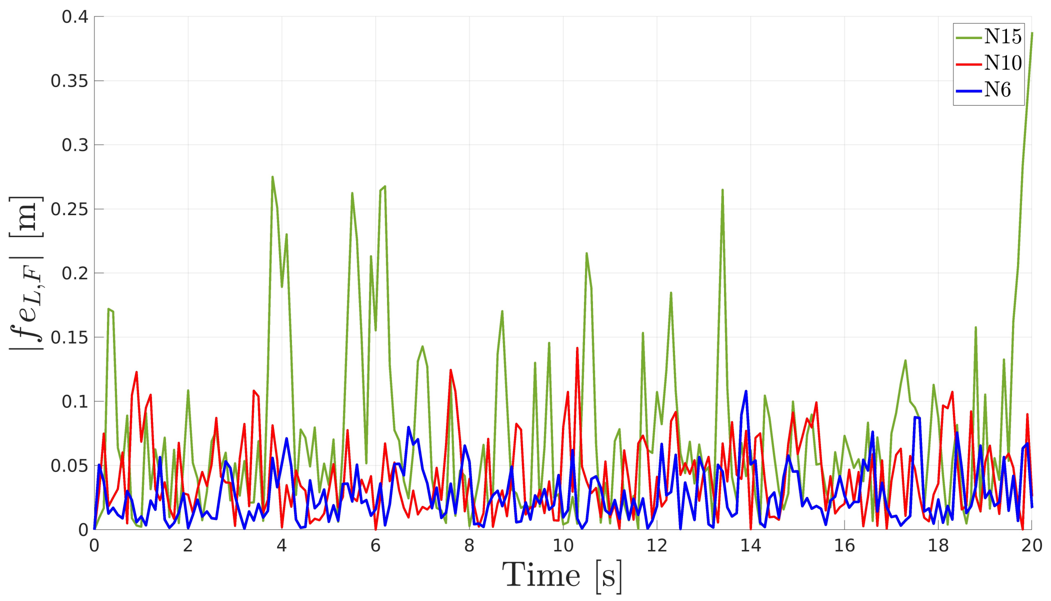


Figure 4.18: Formation error comparison between one simulation for each noise setting in *three obstacles* environment and $\epsilon_{loose\_grip} = 0.01\ m$.

ϵ_{loose_grip} same magnitude as accuracy

Setting ϵ_{loose_grip} at a value comparable with formation error peaks in presence of noise, $\epsilon_{loose\_grip} = 0.06,\ 0.1,\ 0.15\ m$ respectively, reduces the number of steps where recovery policy is actuated. This adjustment acknowledges that a certain magnitude of formation error under these conditions is unavoidable. This causes all simulations to reach the goal, but also leads to an higher number of collisions, probably because leader does not slow down as frequently as with a lower ϵ_{loose_grip}, so agents velocities are higher and collision is tougher to avoid. As a consequence, success rate is equal to collision avoidance rate, in these simulations (Table 4.29). To understand the relationship between increased collision rate and agents velocities, leader and follower mean navigation velocity average values among N_{sims} simulations in some noise configurations, with both ϵ_{loose_grip} values, are compared in Tables 4.30, 4.31.
Increased mean agents velocities also have the side effect of causing velocity or acceleration constraints violations. This happens, for instance, when an agent tries to avoid an unexpected collision scenario caused by the noise. As a matter of fact, all constraints violations occur when the MRS is near obstacles. In Figure 4.19, the violation occurs at $17.9\ s$, that corresponds to a leader position of $(23.8; 6.0)\ m$, extremely near to an obstacle (see *valzer* environment at Figure 4.10).

Table 4.29: Percentage of successful simulations with variable ϵ_{loose_grip}.

Noise	Environment	Success rate
N6	*No obstacles*	100%
	Two obstacles	76%
	Three obstacles	52%
	Valzer	92%
N10	*No obstacles*	100%
	Two obstacles	60%
	Three obstacles	40%
	Valzer	96%
N15	*No obstacles*	100%
	Two obstacles	52%
	Three obstacles	48%
	Valzer	80%

Table 4.30: Mean navigation agents velocities average value among N_{sims} simulations for noise configuration N6 with constant and variable ϵ_{loose_grip}.

Noise	ϵ_{loose_grip} [m]	Leader velocity [m/s]	Follower velocity [m/s]
No obstacles	0.01	1.144	1.172
	0.06	1.144	1.179
Two obstacles	0.01	1.158	1.281
	0.06	1.200	1.347
Three obstacles	0.01	0.885	0.975
	0.06	1.218	1.419
Valzer	0.01	1.135	1.165
	0.06	1.165	1.222

Average and standard deviation values among N_{sims} simulations of mean and peak formation error are reported in Tables 4.32 and 4.33.
Table 4.29 shows that an higher ϵ_{loose_grip} reduces the overall success rate, but the inconvenience of a MRS not moving towards the goal is removed. This trade-off highlights a key aspect of balancing recovery policy actuation with success rate. Moreover, when comparing the effect of different ϵ_{loose_grip} on the framework, formation error performance shown in Tables 4.27 and 4.32 remains almost unchanged. However, it is slightly better when recovery policy is more frequently activated. Thus, in presence of noise, ϵ_{loose_grip} does not affect formation error significantly.

Table 4.31: Mean navigation agents velocities average value among N_{sims} simulations for noise configuration N15 with constant and variable ϵ_{loose_grip}.

Noise	ϵ_{loose_grip} [m]	Leader velocity [m/s]	Follower velocity [m/s]
No obstacles	0.01	0.892	1.061
	0.15	1.121	1.243
Two obstacles	0.01	0.856	1.037
	0.15	1.167	1.274
Three obstacles	0.01	0.612	0.838
	0.15	1.191	1.276
Valzer	0.01	1.028	1.144
	0.15	1.182	1.246

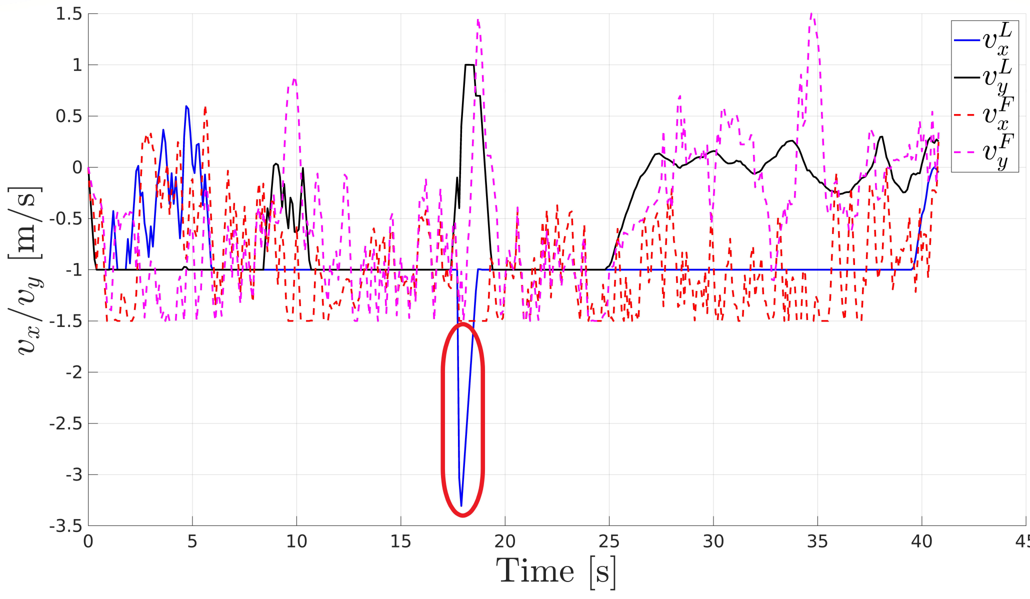


Figure 4.19: Leader velocity constraint violation in *valzer* environment with $\sigma^2 = 0.05^2\ m^2$ and $\epsilon_{loose\_grip} = 0.15\ m$. Leader velocity in x-axis has a spike at around $17.9\ s$ (highlighted with the red sign) and exceeds the limit of $v_{max}^L = 1\ m/s$ imposed at the beginning of Chapter 4.

Recovery policy increases computational effort, regardless of the environment, probably due to the fact that iterations where recovery policy is actuated perform twice the inputs computation, with respect to standard iterations, as it is explained in Section 2.2.3).
Tables 4.38 and 4.39 show an example of how mean and total execution time increases with higher noise covariance when ϵ_{loose_grip} is small with respect to the noise, because of the raising actuation of recovery policy. Instead, when ϵ_{loose_grip} is the same magnitude as noise, the difference between noise configurations mean execution time is mitigated. In general, execution time decreases when noise is smaller.
On the other hand, maximum execution time has a different behaviour (Tables 4.40 and 4.41).

In fact, when ϵ_{loose_grip} is high, computational time peaks become bigger, probably due to high velocities of the agents that lead the disturbed system to a configuration where the optimal path is harder to compute.

Table 4.32: Maximum and mean $fe_{L,F}$ average values among N_{sims} simulations for each noise setting in different environments with variable ϵ_{loose_grip}. Highest value is highlighted in red and the lowest in green for each noise configuration.

Noise	Environment	Max $\lvert fe_{L,F} \rvert$ [m]	Mean $\lvert fe_{L,F} \rvert$ [m]
N6	*No obstacles*	0.101482	0.027253
	Two obstacles	**0.097734**	**0.026998**
	Three obstacles	0.101521	0.027429
	Valzer	**0.114048**	**0.027885**
N10	*No obstacles*	0.183457	0.048233
	Two obstacles	**0.175444**	**0.048435**
	Three obstacles	0.178431	**0.047678**
	Valzer	**0.202803**	0.047786
N15	*No obstacles*	0.309866	0.077616
	Two obstacles	**0.278231**	**0.076377**
	Three obstacles	0.294648	0.077192
	Valzer	**0.357100**	**0.078387**

Table 4.33: Maximum and mean $fe_{L,F}$ standard deviation values among N_{sims} simulations for each noise setting in different environments with variable ϵ_{loose_grip}.

Noise	Environment	Max $\lvert fe_{L,F} \rvert$ [m]	Mean $\lvert fe_{L,F} \rvert$ [m]
N6	*No obstacles*	0.009877	0.001631
	Two obstacles	0.014704	0.002164
	Three obstacles	0.016778	0.001929
	Valzer	0.012673	0.001231
N10	*No obstacles*	0.034089	0.003273
	Two obstacles	0.028390	0.003810
	Three obstacles	0.031699	0.003356
	Valzer	0.033805	0.001808
N15	*No obstacles*	0.065969	0.006785
	Two obstacles	0.048341	0.007043
	Three obstacles	0.051394	0.008133
	Valzer	0.097592	0.005777

4.5.2 Motion Capture

A different set of covariance σ^2 is here tested, such that accuracy is between $5\ mm$ and $2\ cm$ (Table 4.34).

Table 4.34: Second set of additive Gaussian noise tested.

Noise	Accuracy [cm]	$\sigma^2\ [m^2]$
N05	0.5	0.00167^2
N1	1	0.00333^2
N2	2	0.00667^2

With ϵ_{loose_grip} set as $0.01\ m$, all simulations successfully reach the target (Table 4.35). The success rate improves significantly compared to previous scenarios (Section 4.5.1) due to the smaller noise magnitude.

For settings N1 and N2 success rate is smaller than setting N6, in the most challenging environments, even though ϵ_{loose_grip} is the same (Table 4.26). Considering that configurations with lower noise magnitude should behave better, this is an unexpected result. This happens because ϵ_{loose_grip} is now the same magnitude as these N1 and N2 settings covariance, so recovery policy is actuated less frequently with respect to N6, causing the effects explained previously. Despite that, algorithm simulation with setting N05 has a perfect success rate in all environments, probably because this noise is so small that it does not affect the efficiency of the algorithm.

Table 4.35: Percentage of successful simulations of the second set of disturbances with $\epsilon_{loose\_grip} = 0.01\ m$.

Noise	Environment	Success rate
N05	*No obstacles*	100%
	Two obstacles	100%
	Three obstacles	100%
	Valzer	100%
N1	*No obstacles*	100%
	Two obstacles	100%
	Three obstacles	96%
	Valzer	96%
N2	*No obstacles*	100%
	Two obstacles	100%
	Three obstacles	84%
	Valzer	96%

Table 4.36: Maximum and mean $fe_{L,F}$ average values among N_{sims} simulations for each noise setting in different environments with $\epsilon_{loose\_grip} = 0.01\ m$.

Noise	Environment	Max $\|fe_{L,F}\|$ [m]	Mean $\|fe_{L,F}\|$ [m]
N05	*No obstacles*	0.008746	0.002309
	Two obstacles	**0.008340**	**0.002288**
	Three obstacles	0.009966	**0.002661**
	Valzer	**0.010927**	0.002524
N1	*No obstacles*	0.016704	**0.004642**
	Two obstacles	**0.016424**	0.004678
	Three obstacles	0.017360	**0.004765**
	Valzer	**0.019248**	0.004713
N2	*No obstacles*	**0.033185**	**0.009006**
	Two obstacles	0.033482	0.009336
	Three obstacles	0.035160	0.009321
	Valzer	**0.036753**	**0.009356**

Table 4.37: Maximum and mean $fe_{L,F}$ standard deviation values among N_{sims} simulations for each noise setting in different environments with $\epsilon_{loose\_grip} = 0.01\ m$.

Noise	Environment	Max $\|fe_{L,F}\|$ [m]	Mean $\|fe_{L,F}\|$ [m]
N05	*No obstacles*	0.001198	0.000147
	Two obstacles	0.001425	0.000155
	Three obstacles	0.001743	0.000191
	Valzer	0.001704	0.000154
N1	*No obstacles*	0.001472	0.000356
	Two obstacles	0.002282	0.000451
	Three obstacles	0.002247	0.000369
	Valzer	0.001928	0.000221
N2	*No obstacles*	0.004352	0.000547
	Two obstacles	0.004726	0.000588
	Three obstacles	0.004759	0.000820
	Valzer	0.003426	0.000441

Average and standard deviation values among N_{sims} simulations of mean and peak formation error are reported in Tables 4.36 and 4.37.
The same observations made for execution times of previous settings are valid for this new set. (Tables 4.38 and 4.40)

ϵ_{loose_grip} might be decreased to aim for 100% success rate in each environment with noise configurations (N05, N1, N2), losing computational performance. For example $\epsilon_{loose\_grip} = 0.005\ m$ makes the simulations reach a better success rate (Table 4.42), but presents the problems exposed before about a too small ϵ_{loose_grip} value.

Table 4.38: *No obstacles* environment execution times with disturbance and $\epsilon_{loose\_grip} = 0.01\,m$. Best result for average values in each column is highlighted in green, while the worst is highlighted in red.

Noise		Max ET [s]	Mean ET [s]	Tot ET [s]
N05	Average	0.117	**0.021**	**4.515**
	Standard deviation	0.081785	0.001444	0.301695
N1	Average	**0.090**	0.024	4.946
	Standard deviation	0.004644	0.000363	0.080425
N2	Average	0.093	0.031	6.375
	Standard deviation	0.008111	0.001075	0.219525
N6	Average	0.136	0.055	11.67
	Standard deviation	0.072076	0.002537	1.233502
N10	Average	0.127	0.066	15.21
	Standard deviation	0.007215	0.003493	2.760342
N15	Average	**0.162**	**0.075**	**27.62**
	Standard deviation	0.070868	0.003115	6.536876

Table 4.39: *No obstacles* environment execution times with disturbance and variable ϵ_{loose_grip}. Best result for average values in each column is highlighted in green, while the worst is highlighted in red.

Noise		Max ET [s]	Mean ET [s]	Tot ET [s]
N6	Average	**0.117**	**0.035**	**7.257**
	Standard deviation	0.075168	0.001245	0.273934
N10	Average	**0.093**	0.038	8.175
	Standard deviation	0.009451	0.000906	0.402068
N15	Average	0.098	**0.039**	**8.366**
	Standard deviation	0.008421	0.001063	0.728493

This Monte Carlo simulations campaign proves that this 3-DOF NLMPC framework is robust to disturbance below $10\ cm$, depending on MPC parameters and settings chosen, such as ϵ_{loose_grip}, and depending on how obstacles are placed. In fact, adjusting MPC parameters can allow the system to reach the target successfully. Too narrow passages tend to be a critical point in terms of collision avoidance and ability to reach the target for this 3-DOF NLMPC algorithm.

Table 4.40: *Valzer* environment execution times with disturbance and $\epsilon_{loose\_grip} = 0.01\ m$. Best result for average values in each column is highlighted in green, while the worst is highlighted in red.

Noise		Max ET [s]	Mean ET [s]	Tot ET [s]
N05	Average	**0.315**	**0.051**	**21.46**
	Standard deviation	0.057635	0.001064	0.463230
N1	Average	0.347	0.054	22.61
	Standard deviation	0.073248	0.001309	0.556421
N2	Average	0.414	0.067	28.48
	Standard deviation	0.032908	0.002132	1.022726
N6	Average	0.392	0.102	46.25
	Standard deviation	0.021056	0.002991	1.929293
N10	Average	0.413	**0.116**	55.17
	Standard deviation	0.027148	0.003009	3.446492
N15	Average	**0.465**	0.114	**64.70**
	Standard deviation	0.314906	0.008419	7.647056

Table 4.41: *Valzer* environment execution times with disturbance and variable ϵ_{loose_grip}. Best result for average values in each column is highlighted in green, while the worst is highlighted in red.

Noise		Max ET [s]	Mean ET [s]	Tot ET [s]
N6	Average	**0.399**	**0.063**	**26.60**
	Standard deviation	0.293805	0.001874	0.908621
N10	Average	0.540	0.066	27.84
	Standard deviation	0.298196	0.003110	1.321838
N15	Average	**0.814**	**0.069**	**29.11**
	Standard deviation	0.356279	0.003403	1.584019

Table 4.42: Percentage of successful simulations of the second set of disturbances with $\epsilon_{loose\_grip} = 0.005\ m$.

Noise	Environment	Success rate
N05	*No obstacles*	100%
	Two obstacles	100%
	Three obstacles	100%
	Valzer	100%
N1	*No obstacles*	100%
	Two obstacles	100%
	Three obstacles	96%
	Valzer	100%
N2	*No obstacles*	100%
	Two obstacles	100%
	Three obstacles	100%
	Valzer	100%

Chapter 5

Real-time simulations

In this chapter, the 3-DOF NLMPC framework, configured with the V15_5 prediction horizon selection and optimal parameters, is tested in almost real-time environments, namely Simulink® and Gazebo, to evaluate its feasibility for real-world applications. The planned trajectory and the trend of the formation error are compared. The assumption made in MATLAB® simulations regarding the absence of delay in agent acceleration inputs due to computations is removed, since this algorithm needs to work in real-time. Measurement disturbance is not considered.

5.1 3-DOF NLMPC Simulink results

The 3-DOF NLMPC framework assessed in MATLAB®, and represented in Figure 2.2, is now tested in Simulink® to evaluate its behavior in a more time-realistic environment, where a constant MPC output delay due to computation is present. The delay is equal to the sample time $\Delta T = T_s = 0.1\ s$. As a matter of fact, the MPC operates at a sample rate of $f_s = \frac{1}{T_s}$, and each agent input is actuated at the end of each iteration. Tests are performed for all environments introduced in Section 4.2.
Since this delay can have effects similar to a disturbance, recovery policy has a decisive role in terms of effectiveness and formation error performance. After conducting several tests, $\epsilon_{loose\_grip} = 0.05\ m$ is set.

Simulation reported in Figure 5.4 demonstrate that the algorithm is significantly affected by this delay, leading the system to failure, considering multiple collisions and incompatible formation error performance with the problem considered.

5.2 3-DOF NLMPC for Trajectory Generation

Considering the 3-DOF NLMPC framework approach of applying agents acceleration directly computed by the MPC to the robots produced the unacceptable results discussed in Section 5.1, now the 3-DOF NLMPC framework is employed as a trajectory generator (3-DOF NLMPC 4TG), represented in Figure 5.1. The MPC output becomes the next predicted state for both agents and it is given as a reference to two low-level controllers, for example PID, with the task of following this trajectory. This approach is consistent with other MPC approaches studied in the literature [7], [1], [2]. The primary change to the control scheme shown in Figure 2.2 is the MPC output, which now corresponds to the next predicted agent state $\mathbf{x}^*_{ref}$, along with the inclusion of a PID structure responsible for controlling the trajectory of the robots (Figure 5.1). Since the objective is to follow a trajectory reference composed of

$$\mathbf{tr}^*_{ref} = \begin{bmatrix} x^*_{ref} & y^*_{ref} & \theta^*_{ref} \end{bmatrix}^T \tag{5.1}$$

but the MPC output is a predicted state

$$\mathbf{x}^*{}_{ref} = \begin{bmatrix} x^*_{ref} & y^*_{ref} & \theta^*_{ref} & v^*_{x_{ref}} & v^*_{y_{ref}} & \omega^*_{ref} \end{bmatrix}^T \tag{5.2}$$

two different PID structures are analyzed and compared: one with a simple position and heading angle control, one with both position-heading angle and linear-angular velocities, introducing a feedforward using reference velocities (Figures 5.2, 5.3).

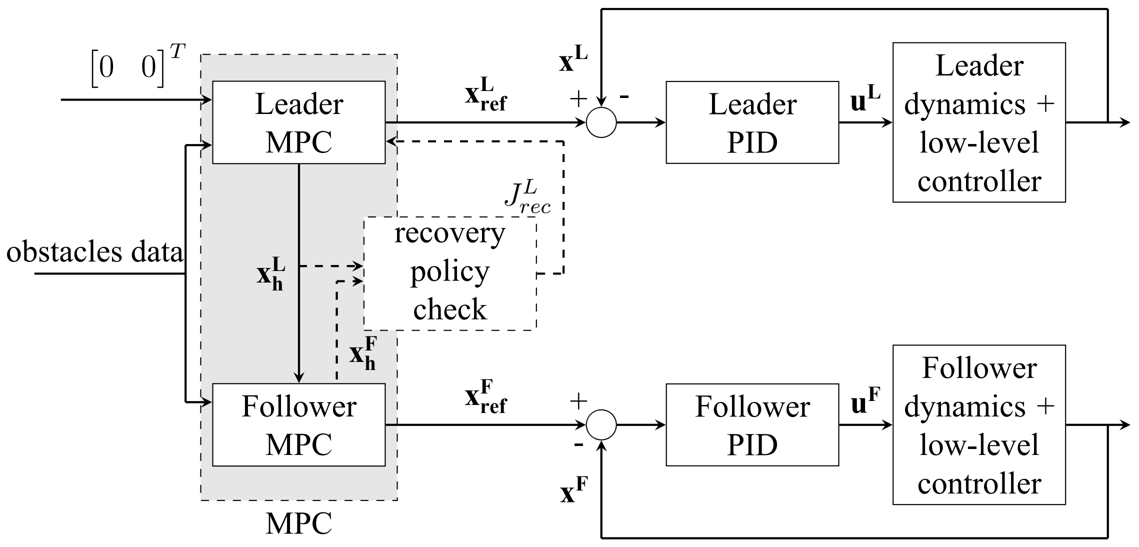


Figure 5.1: 3-DOF NLMPC 4TG framework scheme.

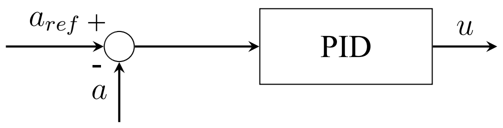


Figure 5.2: Example of position-heading PID (PH PID) structure, with a generic spatial variable to control and u generic output.

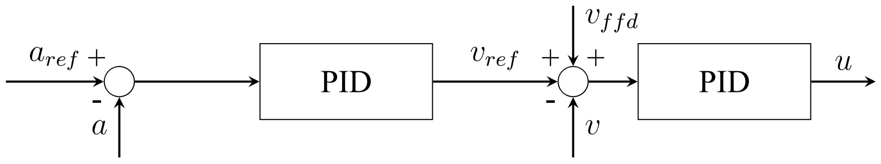


Figure 5.3: Example of position-heading and velocity PID (PH-V PID) structure, with a generic spatial variable to control, v generic velocity variable and u generic output.

With this new configuration, this 3-DOF NLMPC 4TG scheme can fulfill its tasks and transport the payload to the target without colliding with obstacles. $C_{pot} = 50$ is set to improve collision avoidance performance. Also, linear acceleration limits are increased ($u^L_{lim} = 5\ m/s^2$, $u^F_{lim} = 7\ m/s^2$) to avoid a too strict saturation, penalizing the tracking performance.
Formation error performance of each PID structure is reported in Figures 5.4 and 5.5 showing a significant improvement with respect to the previous 3-DOF NLMPC control scheme. This holds because the computational delay, in this case, only determines a time shift of the state reference and does not compromise the effectiveness of the MPC shown in Chapter 4. Therefore, the performance of the algorithm highly depends on PID tracking capability, which in this case proves to be sufficiently accurate. In particular, both PID structures assure a good formation error (Table 5.3), but with a different trend, that seems to be more stable using position-heading angle control only. Moreover, this latter seems to converge after an initial transient phase. Obviously, these results strongly depend on PID parameters[1]. PID configuration is reported in Tables 5.1 and 5.2.

Simulink® simulations demonstrated that a direct MPC control on the MRS is not possible in real-time, because of the computational delay on agents input. Consequently a 3-DOF NLMPC 4TG scheme is developed to solve this problem and effectively control the MRS with the framework studied in previous chapters, showing good formation error and tracking performance.

[1]PID filtered derivative is employed.

Table 5.1: Leader PID parameters used in 3-DOF NLMPC 4TG architecture.

	Leader PH PID			Leader PH-V PID		
	P	I	D	P	I	D
x	1600	50	20	900	10	42
y	1600	110	20	1000	0	0
θ	1250	1	43	3200	0.5	50
v_x	–	–	–	5	0	1
v_y	–	–	–	0.2	0.005	0.06
ω	–	–	–	1	0	0.01

Table 5.2: Follower PID parameters used in 3-DOF NLMPC 4TG architecture.

	Follower PH PID			Follower PH-V PID		
	P	I	D	P	I	D
x	1275	80	45	3000	0	0
y	1050	110	32	3000	0	0
θ	1250	1	43	1250	1	43
v_x	–	–	–	0.1	0	0.025
v_y	–	–	–	0.1	0	0.0009
ω	–	–	–	0.05	0.057	0.01

Table 5.3: PID structure formation error performance in all environments tested. Best value for each environment is highlighted.

Environment	PID structure	Max $\lvert fe_{L,F}\rvert$ [m]	Mean $\lvert fe_{L,F}\rvert$ [m]
No obstacles	PH	0.063	**0.007**
	PH-V	**0.057**	0.028
Two obstacles	PH	**0.063**	**0.013**
	PH-V	0.065	0.032
Three obstacles	PH	**0.063**	**0.013**
	PH-V	0.077	0.035
Valzer	PH	**0.067**	**0.008**
	PH-V	0.125	0.025

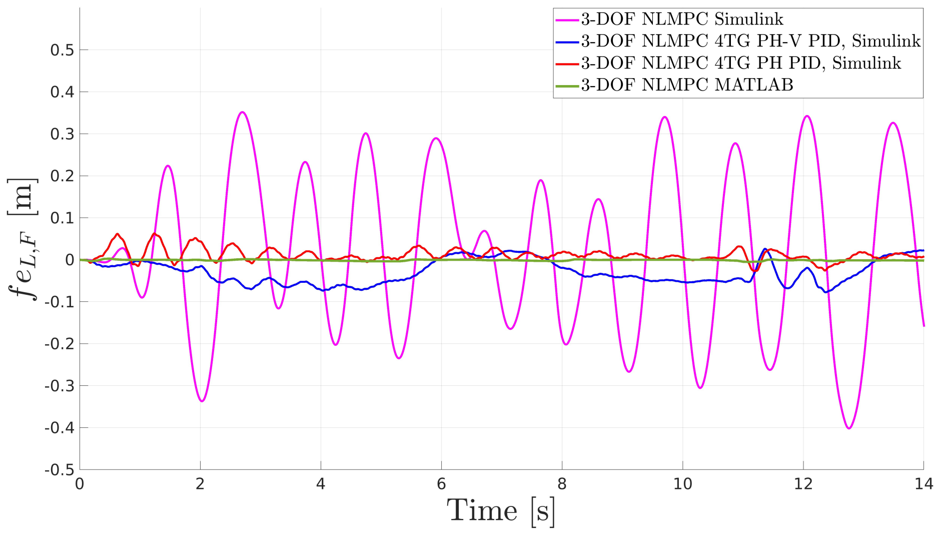
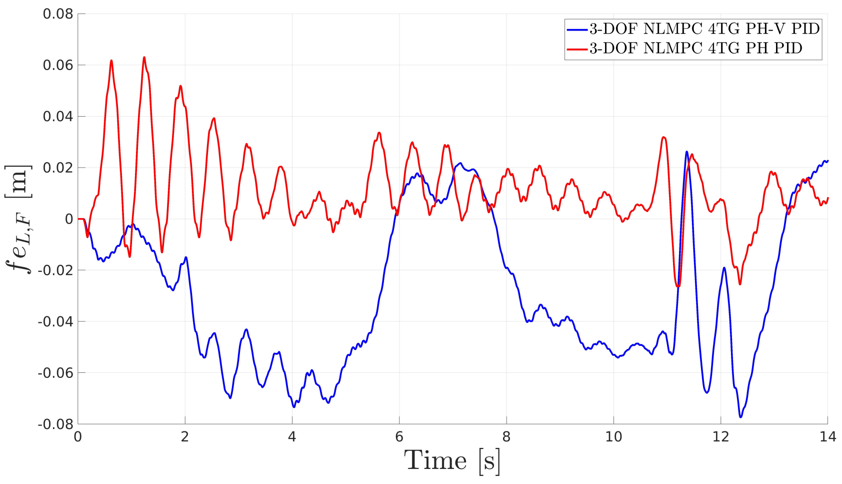
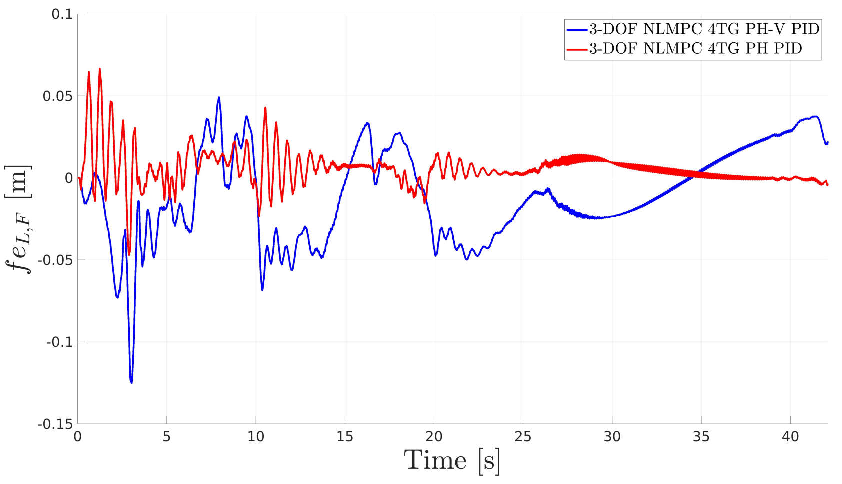


Figure 5.4: Formation error comparison between MATLAB® and Simulink® simulations in *Three obstacles* environment of the 3-DOF NLMPC and 3-DOF NLMPC 4TG algorithm.



(a) *Three obstacles* environment.



(b) *Valzer* environment.

Figure 5.5: Formation error comparison between PID structures of the 3-DOF NLMPC 4TG framework in different scenarios.

5.3 Gazebo experiments

Gazebo is a realistic simulation environment that can implement physical properties to the models inserted, such as introducing gravity, friction and collision dynamics. It is useful to evaluate the behaviour and the interaction of the tested model with a more realistic world, in presence of various kind of real disturbances. Thus, *three obstacles* environment is implemented in Gazebo[2] to assess the behaviour of the 3-DOF NLMPC 4TG algorithm, with V15_5 prediction horizon and PH PID controller, in a realistic environment, where real-world physics is taken into account. The algorithm runs on Simulink® and ROS2 topics are used to communicate state measurement, from Gazebo to Simulink®, and to exchange agent inputs, which are provided as force and torque applied to the robot base from the PID output. The computational delay investigated previously in Simulink® is present. Moreover, delays concerning the non-ideality of ROS2 topics are introduced. Measurement noise is not considered.

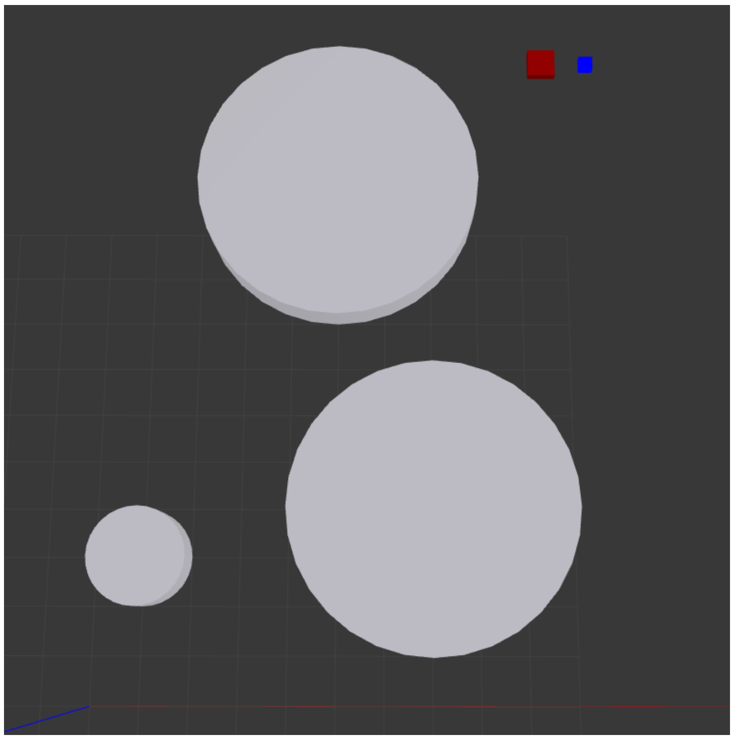
5.3.1 Simplified agents model and no friction

Agents are modeled as simple boxes and obstacles as fixed cylinders, with their dimensions specified in Chapter 4. No payload is placed to measure formation error and to avoid unexpected behaviour due to forces applied on the revolute joint in combination with the non-deformability of the payload (Figure 5.6). Friction is initially excluded.
MPC is configured with optimal parameters (Section 4.1) and $\epsilon_{loose\_grip} = 0.05\ m$, except for C_{pot} that needs to be higher to assure the reliability of the algorithm in these conditions and to successfully avoid obstacles, so $C_{pot} = 125$. Also, agents acceleration bounds introduced in Section 5.2 are used for the same reason.

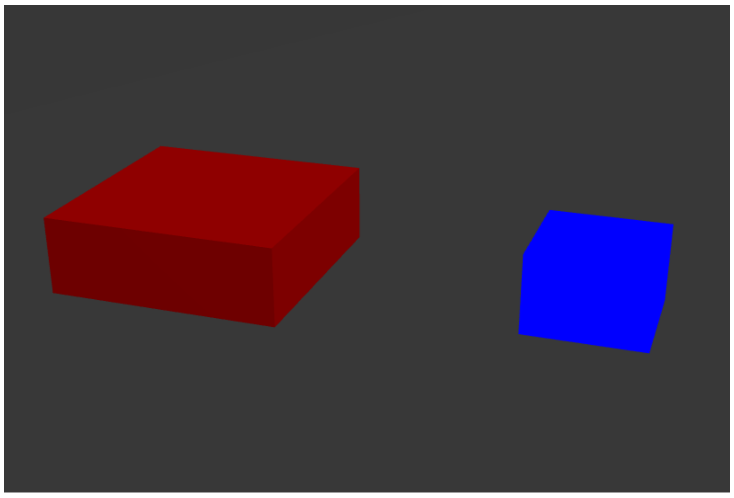
Gazebo simulations show that the 3-DOF NLMPC 4TG framework behaves similarly to Simulink® simulations discussed in Section 5.2, guaranteeing the same formation error performance, with a mean $|fe_{L,F}|$ of $7.3\ mm$, but with a more noisy trend (Figure 5.7). Despite that, observing the simulation in Gazebo GUI, it is noticeable that agents navigation is not really smooth and it appears as though the robots stop at each predicted position. It is not really clear if this is due to Gazebo computational performance or it is the actual algorithm behaviour. Accord-

[2] Algorithm effectiveness decreases when Gazebo Graphical User Interface (GUI) is active, so these simulations are realized using Gazebo server only (command gzserver). This performance drop is likely caused by the Gazebo environment, probably because it results to be computationally demanding for the device running both the MPC and the simulative environment. Moreover, each Gazebo simulation with the same exact setup can differ in some results, probably due to the intrinsic Gazebo uncertainties and its impact on device performance.

ingly, it might be caused by non-optimal PID gains, that may imply inefficient position tracking and a noisy navigation path. Another explanation can be inconsistent robots parameters, such as velocity and acceleration bounds, which could lead to jerky movement.



(a) *Three obstacles* environment setup.



(b) Simplified MRS.

Figure 5.6: Test setup for Gazebo simulations: it presents the MRS, composed of two simple boxes with no friction, where leader is in red and follower in blue, and the obstacles are modeled as gray fixed cylinders.

5.3.2 Experiments with wheels and friction

Now four symmetric spherical wheels are added to the two agents in the Gazebo model to carry out even more realistic experiments, even though spherical wheels are not typically used in common applications. This addition does not impact the overall performance, since the results remain consistent with those shown in Figure 5.7. An interesting experiment would involve studying the effect of friction applied to the wheels. This friction would act like a disturbance, more precisely like a process noise, since friction is not considered in agents dynamic model. Friction coefficient $\mu = 1$ is set for translational and rotational movements of the wheels. This value of the friction coefficient emulates the contact between rubber and concrete in dry conditions.

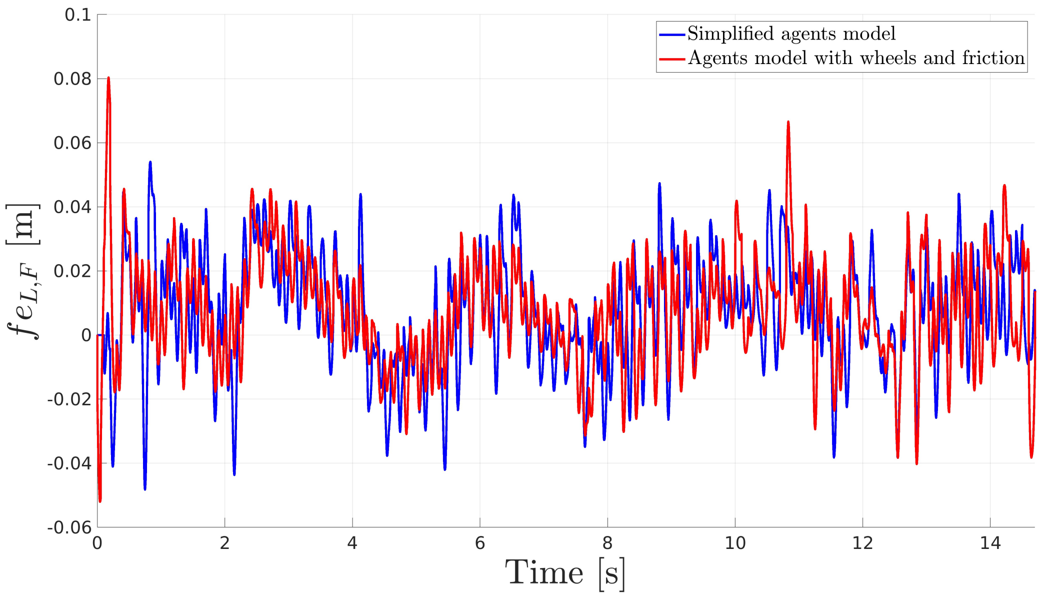


Figure 5.7: *Three obstacles* Gazebo simulation formation error graph, without Gazebo GUI running.

Figure 5.7 shows how the performance is not affected by the friction, meaning the tracking controller is effective. However, the fact that the spherical wheel model touches the ground at a single point may be a significant approximation. The mean formation error $|fe_{L,F}|$ is $7.8\ mm$. The only observable difference is the spike during the first few instants of the simulation shown in Figure 5.7, that is probably caused by a little delay on agents position tracking due to friction.

5.3.3 Experiments with the payload

Lastly, the payload is added to the model. It is designed as a box with the dimensions specified in Chapter 4 and a negligible weight with respect to robots so that it does not interfere with the wrench input of agents, applied to the base. Payload is attached with fixed joints to two cylinders that act like a revolute joint with respect to robots base (Figure 5.8).

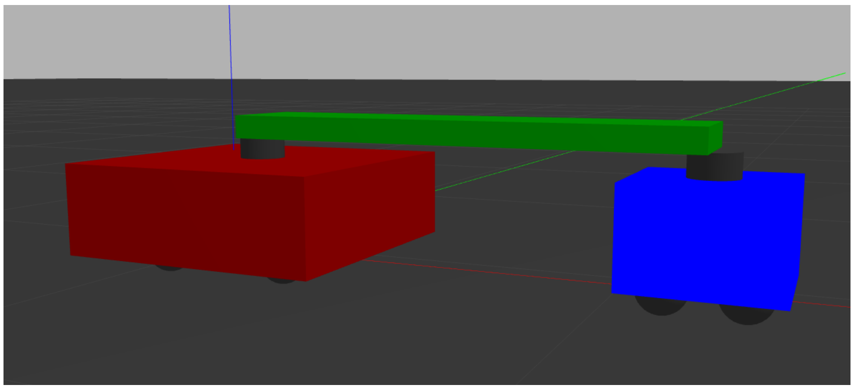


Figure 5.8: MRS with payload in Gazebo. Payload is the green box and is attached to two gray cylinders, that act as revolute joints with agents base.

Even this solution behaves really well, considering that the trajectory of the MRS remains consistent with previous Gazebo simulations (Figure 5.9). Since the payload is fixed to the cylinders, the expectations were to see a null formation error, but probably Gazebo has a certain tolerance for these kind of joints where mechanical loads are applied. Hence, a formation error is measured, but it is bounded below $4\ cm$ (Figure 5.10).

Gazebo simulations proved how this 3-DOF NLMPC 4TG framework has a great potential in real-world applications, showing good performance on cooperative transportation and a compatible formation error graph with the problem addressed.

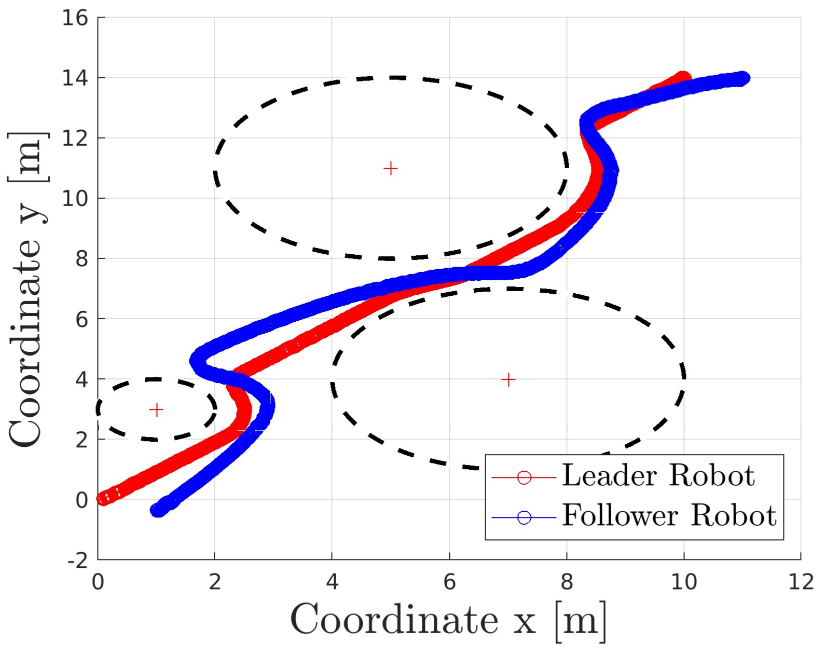


Figure 5.9: Trajectory of Gazebo simulations in *hree obstacles* environment.

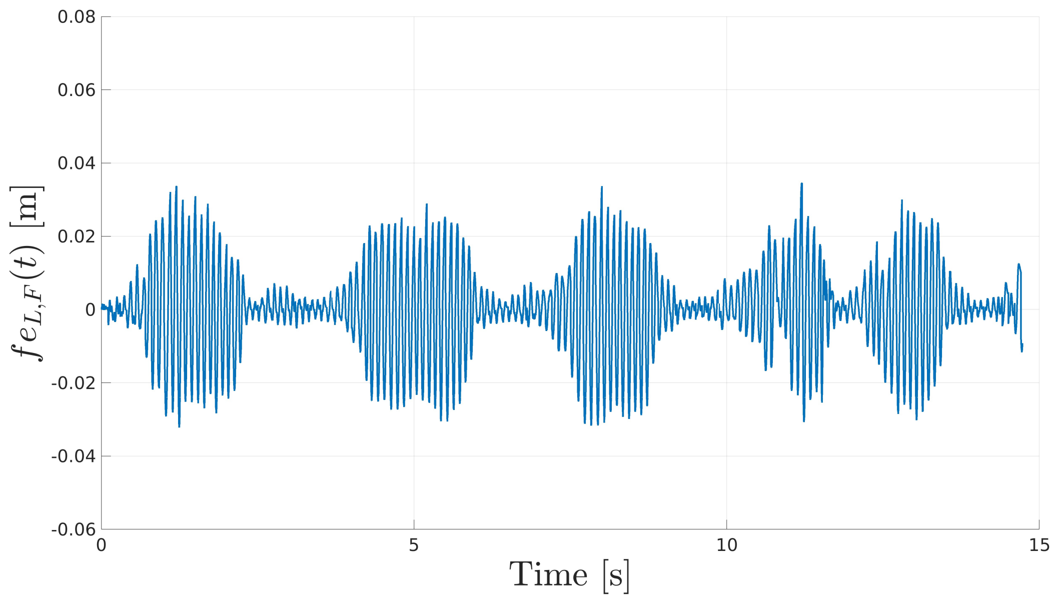


Figure 5.10: Formation error of the system with payload in Gazebo simulation.

Chapter 6

Conclusions

This thesis addressed the cooperative transportation topic proposing a feasibility-aware non-linear MPC framework for a MRS composed of two UGVs with 3-DOF, navigating in a planar space in presence of obstacles. The 3-DOF dynamic model of the agents ensured the capability of the MRS to become more flexible and efficient in its obstacle avoidance task, with respect to the previous work presented in [6]. Moreover, potential repulsive field and obstacle avoidance constraints for load vertices improved the safety of the system regarding collision avoidance. These changes introduced non-linearity to the 2-DOF LMPC framework designed in [6], which may have caused the increased formation error reported in Section 4.4. Although, the enhanced obstacle avoidance performance are considered more significant, since formation error can be reduced either controlling the manipulator mounted on agents base to move the end-effector or designing a passive end-effector that can deal with formation error without damaging the payload.

A state machine enabling the implementation of a variable prediction horizon MPC demonstrated that a variable prediction horizon MPC can reach similar computational performance of a small fixed prediction horizon MPC, but reporting a lower formation error. Moreover, sample rate of the MPC can be modified according to the execution time of the 3-DOF NLMPC framework. This novel approach based on a state machine is hardly treated in the literature.

Monte Carlo simulations campaign revealed that the system is robust to measurement disturbances relatable to Motion Caption, while it is more susceptible with disturbances similar to UWB accuracy.

Time-realistic environments, namely Simulink® and Gazebo, simulations demonstrated that a 3-DOF NLMPC architecture giving as an output the control input of the agents is ineffective due to computational delay. Consequently, the 3-DOF NLPMC framework is redesigned to compute agents next predicted state as an output, consistently with other MPC approaches in the literature [7], [1], [2], in combination with a PID structure tasked to track the trajectory.

This solution is far more effective and it is able to reach the target without colliding with obstacles and keeping the formation with a certain error in Gazebo experiments. If PID structure performance is considered insufficient, different tracking controllers can be investigated.
Various possible further steps to improve this algorithm can be implemented, such as revise obstacle avoidance constraints to correctly consider dynamic obstacles, to implement agent-collision avoidance and to achieve a reliable and efficient algorithm, trying more intensive tests. For instance, a few experiments are suggested in Section 3.3.5. Then, introducing chance constraints presented in [10] to smooth out the trajectory can be a valid improvement, even though this algorithm already reaches great performance from this point of view. Lastly, increasing system complexity by controlling more formations at the same time, and so implementing formation avoidance constraints, is an interesting feature [8].
Experiments with real robots are obviously the priority concerning further works, to assure the actual feasibility of the algorithm. A revolute joint is not recommended since the end-effector needs to have a certain level of mobility tolerance to allow formation error, as Gazebo simulations proved, otherwise the behaviour might be different than the one reported here.
All things considered, this 3-DOF NLMPC implementing a variable prediction horizon have a great potential in collaborative transportation, ensuring enhanced obstacle avoidance performance and a compatible formation error with respect to this application. It also guarantees a certain level of robustness to disturbances, consistent with indoor localization systems employed in common applications.

Appendices

.1 Linear constraints matrices

$$G = \begin{bmatrix} A_{bar}\bar{S} \\ G_{in} \\ A_{vel}\bar{S} \end{bmatrix} \in \mathbb{R}^{(NML+2(v+m)N)\times mN} \tag{1}$$

$$A_{constr} = \begin{bmatrix} q_{1_x} - x(0) & q_{1_y} - y(0) & 0 & 0 \\ q_{1_x} - x(0) & q_{1_y} - y(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ q_{2_x} - x(0) & q_{2_y} - y(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ q_{M_x} - x(0) & q_{M_y} - y(0) & 0 & 0 \end{bmatrix} \in \mathbb{R}^{ML\times n} \tag{2}$$

A_{constr} is composed of ML rows, where there are L identical rows for each obstacle.

$$A_{bar} = I_N \otimes A_{constr} =$$

$$= \begin{bmatrix} \begin{bmatrix} q_{1_x} - x(0) & q_{1_y} - y(0) & 0 & 0 \\ q_{1_x} - x(0) & q_{1_y} - y(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ q_{2_x} - x(0) & q_{2_y} - y(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ q_{M_x} - x(0) & q_{M_y} - y(0) & 0 & 0 \end{bmatrix} & 0_{ML\times n} & \cdots \\ \vdots & \ddots & \vdots \\ 0_{ML\times n} & \cdots & \begin{bmatrix} q_{1_x} - x(0) & q_{1_y} - y(0) & 0 & 0 \\ q_{1_x} - x(0) & q_{1_y} - y(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ q_{2_x} - x(0) & q_{2_y} - y(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ q_{M_x} - x(0) & q_{M_y} - y(0) & 0 & 0 \end{bmatrix} \end{bmatrix} \tag{3}$$

$$\in \mathbb{R}^{NML\times nN}$$

$$G_{in} = I_N \otimes \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} = I_N \otimes U_l = \begin{bmatrix} U_l & \cdots & 0_{2m\times m} \\ \vdots & \ddots & \vdots \\ 0_{2m\times m} & \cdots & U_l \end{bmatrix} \in \mathbb{R}^{2mN\times mN} \tag{4}$$

G_{in} is a block diagonal matrix selecting each acceleration input, once for the positive and once for the negative bound, within $\mathbf{u}_k^*$.

$$A_{vel} = I_N \otimes \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} = I_N \otimes V_l = \begin{bmatrix} V_l & \dots & 0_{2v \times n} \\ \vdots & \ddots & \vdots \\ 0_{2v \times n} & \dots & V_l \end{bmatrix} \in \mathbb{R}^{2vN \times nN} \tag{5}$$

A_{vel} is a block diagonal matrix selecting each agent velocity, once for the positive and once for the negative bound, within $\mathbf{x}_h^*$.

$$W = \begin{bmatrix} B_{bar} \\ B_{in} \\ B_{vel\_constr} \end{bmatrix} \in \mathbb{R}^{(NML+2(v+m)N) \times 1} \tag{6}$$

$$\begin{aligned} b_{constr} &= \begin{bmatrix} \begin{bmatrix} q_{1_x} - x(0) & q_{1_y} - y(0) \end{bmatrix} \begin{bmatrix} q_{1_x} - v_{1_x} \\ q_{1_y} - v_{1_y} \end{bmatrix} \\ \begin{bmatrix} q_{1_x} - x(0) & q_{1_y} - y(0) \end{bmatrix} \begin{bmatrix} q_{1_x} - v_{2_x} \\ q_{1_y} - v_{2_y} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} q_{2_x} - x(0) & q_{2_y} - y(0) \end{bmatrix} \begin{bmatrix} q_{2_x} - v_{1_x} \\ q_{2_y} - v_{1_y} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} q_{M_x} - x(0) & q_{M_y} - y(0) \end{bmatrix} \begin{bmatrix} q_{M_x} - v_{L_x} \\ q_{M_y} - v_{L_y} \end{bmatrix} \end{bmatrix} = \\ &= \begin{bmatrix} [q_{1_x} - x(0)][q_{1_x} - v_{1_x}] + [q_{1_y} - y(0)][q_{1_y} - v_{1_y}] \\ [q_{1_x} - x(0)][q_{1_x} - v_{2_x}] + [q_{1_y} - y(0)][q_{1_y} - v_{2_y}] \\ \vdots \\ [q_{2_x} - x(0)][q_{2_x} - v_{1_x}] + [q_{2_y} - y(0)][q_{2_y} - v_{1_y}] \\ \vdots \\ [q_{M_x} - x(0)][q_{M_x} - v_{L_x}] + [q_{M_y} - y(0)][q_{M_y} - v_{L_y}] \end{bmatrix} \in \mathbb{R}^{ML \times 1} \end{aligned} \tag{7}$$

$$B_{bar} = repmat(b_{constr}, N, 1) = \begin{bmatrix} [q_{1_x} - x(0)][q_{1_x} - v_{1_x}] + [q_{1_y} - y(0)][q_{1_y} - v_{1_y}] \\ \vdots \\ [q_{M_x} - x(0)][q_{M_x} - v_{L_x}] + [q_{M_y} - y(0)][q_{M_y} - v_{L_y}] \\ [q_{1_x} - x(0)][q_{1_x} - v_{1_x}] + [q_{1_y} - y(0)][q_{1_y} - v_{1_y}] \\ \vdots \\ [q_{M_x} - x(0)][q_{M_x} - v_{L_x}] + [q_{M_y} - y(0)][q_{M_y} - v_{L_y}] \\ \vdots \end{bmatrix} \in \mathbb{R}^{NML \times 1} \tag{8}$$

`repmat(vec, N, 1)` is the MATLAB® function that repeats the same matrix or vector `vec`, b_{constr} in this case, for N rows and 1 column.

$$B_{in} = \begin{bmatrix} \begin{bmatrix} u_{max} \\ u_{max} \\ u_{max} \\ u_{max} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} u_{max} \\ u_{max} \\ u_{max} \\ u_{max} \end{bmatrix} \\ \vdots \end{bmatrix} \in \mathbb{R}^{2mN \times 1} \qquad B_{vel\_constr} = \begin{bmatrix} \begin{bmatrix} v_{max} \\ v_{max} \\ v_{max} \\ v_{max} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} v_{max} \\ v_{max} \\ v_{max} \\ v_{max} \end{bmatrix} \\ \vdots \end{bmatrix} \in \mathbb{R}^{2vN \times 1} \tag{9}$$

$$S = \begin{bmatrix} -(A_{bar}\bar{T}) \\ 0_{2vN \times n} \\ -(A_{vel}\bar{T}) \end{bmatrix} \in \mathbb{R}^{(NML+2(v+m)N) \times n} \tag{10}$$

$$\begin{bmatrix} A_{bar}\bar{S} \\ G_{in} \\ A_{vel}\bar{S} \end{bmatrix} \mathbf{u_k} \leq \begin{bmatrix} B_{bar} \\ B_{in} \\ B_{vel_constr} \end{bmatrix} + \begin{bmatrix} -(A_{bar}\bar{T}) \\ 0_{2vN \times n} \\ -(A_{vel}\bar{T}) \end{bmatrix} \mathbf{x}(0) \tag{11}$$

Equation (2.40) is the expanded inequality representing the linear constraints. Checking the dimension we can see that: $\mathbb{R}^{(NML+2(v+m)N) \times mN} \times \mathbb{R}^{mN \times 1} \leq \mathbb{R}^{(NML+2(v+m)N) \times 1} + \mathbb{R}^{(NML+2(v+m)N) \times n} \times \mathbb{R}^{n \times 1}$, so $\mathbb{R}^{(NML+2(v+m)N) \times 1} \leq \mathbb{R}^{(NML+2(v+m)N) \times 1}$.

.2 *fmincon* algorithms comparison

In MATLAB® code, optimization problem is performed by `fmincon()` function, only for follower functional cost optimization in 2-DOF LMPC algorithm and for all functional costs in 3-DOF NLMPC algorithm. It has different options depending on the problem, such as the optimization algorithm it uses. The performance of the latter, in terms of computational speed, depends on the functional costs and constraints of the problem, so, to reach the best possible solution, an algorithm comparison is necessary.
Algorithms recommended by the `fmincon()` guide, for the kind of optimization problem treated in this thesis, are:

- "interior-point".
- "sqp".
- "active-set".

These tests are made using optimal parameters selected in Section 4.1 and considering the best configurations for each fixed and variable prediction horizon found in Section 4.3, also using perception range.
All tests reveal "sqp" is the best `fmincon()` algorithm, guaranteeing the best computational performance and usually the best results regarding formation error and agent-obstacle distance. A few examples are reported in Tables 1, 2, 3, 4, 5.

Table 1: `fmincon()` algorithms comparison with F5 prediction horizon in *no obstacles* environment.

	"interior-point"	"sqp"	"active-set"
Max ET [s]	0.822	**0.056**	0.171
Mean ET [s]	0.017	**0.005**	0.007
Total ET [s]	3.399	**0.950**	1.368
Max $\lvert fe_{L,F}\rvert$ [mm]	3.300	3.315	**3.297**
Mean $\lvert fe_{L,F}\rvert$ [mm]	0.333	0.250	**0.246**

Table 2: `fmincon()` algorithms comparison with F5 prediction horizon in *three obstacles* environment.

	"interior-point"	"sqp"	"active-set"
Max ET [s]	0.284	**0.066**	0.122
Mean ET [s]	0.047	**0.027**	0.035
Total ET [s]	7.073	**3.986**	5.142
Max $\lvert fe_{L,F}\rvert$ [mm]	7.989	**5.212**	8.004
Mean $\lvert fe_{L,F}\rvert$ [mm]	1.423	**1.275**	1.413
Min obs distance [mm]	5.671	1.952	**5.969**

Table 3: `fmincon()` algorithms comparison with V15_5 prediction horizon in *no obstacles* environment.

	"interior-point"	"sqp"	"active-set"
Max ET [s]	0.125	**0.094**	0.185
Mean ET [s]	0.031	**0.018**	0.032
Total ET [s]	6.504	**3.816**	6.711
Max $\lvert fe_{L,F}\rvert$ [mm]	**2.694**	3.154	3.154
Mean $\lvert fe_{L,F}\rvert$ [mm]	0.099	0.073	**0.070**

Table 4: `fmincon()` algorithms execution time comparison with V15_5 prediction horizon in *valzer* environment .

	"interior-point"	"sqp"	"active-set"
Max ET [s]	2.393	**0.280**	0.677
Mean ET [s]	0.079	**0.044**	0.057
Mean ET N_{long} [s]	0.028	**0.015**	0.024
Mean ET N_{short} [s]	0.114	**0.064**	0.079
Total ET [s]	33.68	**18.76**	24.08

Table 5: `fmincon()` algorithms formation error and obstacle avoidance performance comparison with V15_5 prediction horizon in *valzer* environment.

	"interior-point"	"sqp"	"active-set"
Max $\lvert fe_{L,F}\rvert$ [mm]	8.602	**7.073**	8.957
Mean $\lvert fe_{L,F}\rvert$ [mm]	0.614	**0.596**	0.676
Min obs distance [cm]	18.6	**19.8**	16.3

Bibliography

[1] S. Sun and A. Franchi, “Nonlinear mpc for full-pose manipulation of a cable-suspended load using multiple uavs,” in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2023, pp. 969–975. DOI: 10.1109/ICUAS57906.2023.10156031.

[2] G. Li and G. Loianno, “Nonlinear model predictive control for cooperative transportation and manipulation of cable suspended payloads with multiple quadrotors,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 5034–5041. DOI: 10.1109/IROS55552.2023.10341785.

[3] A. Mohiuddin, T. Taha, Y. Zweiri, and D. Gan, “Dual-uav payload transportation using optimized velocity profiles via real-time dynamic programming,” *Drones*, vol. 7, no. 3, 2023, ISSN: 2504-446X. DOI: 10.3390/drones7030171. [Online]. Available: https://www.mdpi.com/2504-446X/7/3/171.

[4] D. K. D. Villa, A. S. Brandão, R. Carelli, and M. Sarcinelli-Filho, “Cooperative load transportation with two quadrotors using adaptive control,” *IEEE Access*, vol. 9, pp. 129 148–129 160, 2021. DOI: 10.1109/ACCESS.2021.3113466.

[5] J. Horyna, T. Baca, and M. Saska, “Autonomous collaborative transport of a beam-type payload by a pair of multi-rotor helicopters,” in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2021, pp. 1139–1147. DOI: 10.1109/ICUAS51884.2021.9476789.

[6] B. Elaamery, M. Pesavento, T. Aldovini, N. Lissandrini, G. Michieletto, and A. Cenedese, “Model predictive control for cooperative transportation with feasibility-aware policy,” *Robotics*, vol. 10, no. 3, 2021, ISSN: 2218-6581. DOI: 10.3390/robotics10030084. [Online]. Available: https://www.mdpi.com/2218-6581/10/3/84.

[7] N. Lissandrini, C. K. Verginis, P. Roque, A. Cenedese, and D. V. Dimarogonas, “Decentralized nonlinear mpc for robust cooperative manipulation by heterogeneous aerial-ground robots,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 1531–1536. DOI: 10.1109/IROS45743.2020.9341023.

[8] Y. Sirineni, R. Tallamraju, A. Rawat, and K. Karlapalem, “Decentralized collision avoidance and motion planning for multi-robot deformable payload transport systems,” in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2020, pp. 154–161. DOI: 10.1109/SSRR50563.2020.9292636.

[9] S. Satir, Y. F. Aktas, S. Atasoy, M. Ankarali, and E. Sahin, “Distributed model predictive formation control of robots with sampled trajectory sharing in cluttered environments,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 9309–9315. DOI: 10.1109/IROS55552.2023.10341414.

[10] T. Wakabayashi, Y. Nunoya, and S. Suzuki, “Dynamic obstacle avoidance of multi-rotor uav using chance constrained mpc,” in *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, 2021, pp. 412–417. DOI: 10.23919/ICCAS52745.2021.9649942.

[11] M. Bujarbaruah, Y. R. Stürz, C. Holda, K. H. Johansson, and F. Borrelli, “Learning environment constraints in collaborative robotics: A decentralized leader-follower approach,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1636–1641. DOI: 10.1109/IROS51168.2021.9636444.

[12] G. ”Indiveri, “”omnidirectional mobile robots”,” in *”Encyclopedia of Robotics”*, M. H. ”Ang, O. Khatib, and B. Siciliano, Eds. ”Berlin, Heidelberg”: ”Springer Berlin Heidelberg”, ”2020”, ”1–5”, ISBN: ”978-3-642-41610-1”. DOI: "10.1007/978-3-642-41610-1_47-1". [Online]. Available: "https://doi.org/10.1007/978-3-642-41610-1_47-1".

[13] T.-G. Sorescu, A.-V. Militaru, V.-M. Chiriac, C.-R. Comsa, and I.-E. Alecsandrescu, “Uwb indoor localization: Accuracy evaluation in a controlled environment,” in *2024 16th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2024, pp. 1–6. DOI: 10.1109/ECAI61503.2024.10607552.

[14] R. Bharadwaj, C. Parini, and A. Alomainy, “Indoor tracking of human movements using uwb technology for motion capture,” in *The 8th European Conference on Antennas and Propagation (EuCAP 2014)*, 2014, pp. 2097–2099. DOI: 10.1109/EuCAP.2014.6902221.