MPC-based distributed formation control of multiple quadcopters with obstacle avoidance and connectivity maintenance

Salim Vargas[a], Héctor M. Becerra[a,*], Jean-Bernard Hayet[a]

[a]*Centro de Investigación en Matemáticas (CIMAT), A.C., Jalisco S/N, Col. Valenciana, 36023, Guanajuato, Gto., México*

Abstract

In this work, a distributed model predictive control (MPC) scheme based on consensus theory is proposed for the formation control of a group of quadcopters. The MPC scheme provides velocities for the quadcopters, which are considered as holonomic agents modeled at kinematic level. We propose soft and hard constraints for the MPC problem to address collision and obstacle avoidance as well as to maintain the connectivity of the communication topology during the motion of the agents to reach the desired formation. The contributions of this work are the following: First, we propose an integrated solution for the three tasks, including connectivity maintenance, which is uncommon in existing approaches, in addition to dynamic formation control and collision/obstacle avoidance. Second, we show that using both soft and hard constraints in the MPC problem gives better performance than using only one of the two. Third, unlike most MPC-based schemes from the literature, the effectiveness of our approach is validated through real experiments for a group of quadcopters.

Keywords: Formation control; consensus; multiple quadcopters; distributed model predictive control; obstacle avoidance; connectivity maintenance.

*Corresponding author
Email address: `hector.becerra@cimat.mx` (Héctor M. Becerra)

1. Introduction

In the last few years, control of multi-agent systems (MASs) and, more particularly, formation control have raised an increasing interest in the robotics community. One of the reasons is that swarms of cheap robotic agents typically introduce much more flexibility and robustness in the realization of essential robotic tasks, such as exploration [1], mapping [2] and tracking [3], with particular applications in search and rescue [4], load transportation [5] and vehicle guidance [6], to give a few examples.

In this work, we propose a new method for solving the problem of formation control with quadcopters, i.e., to drive autonomously the different agents of a distributed swarm of quadcopters in a predefined configuration of relative positions. We assume that the perception and communication of each quadcopter is limited to the one of its neighbors, and that the environment is populated with static obstacles. Hence, it is important that the control laws allowing the swarm to reach the formation objective also allow to avoid collisions, either among the quadcopters of the swarm or between quadcopters and static obstacles, while connectivity is maintained.

Formation control is one of the most actively studied topics within the control of MASs. This problem generally aims at driving multiple agents simultaneously to achieve some prescribed constraints on their states [7]. The formation control of multiple unmanned aerial vehicles (MUAVs) has been addressed in two forms. With *centralized approaches* [8, 9], the computation of the controllers of all the agents takes place in the same station or all the variables are measured with respect to a global coordinate system, sometimes relying on a broadcast transmission of data. The other form to address the formation control is using a *distributed approach* [10, 11], where the computation of the controllers takes place in a different station for each agent and where the approach relies on relative information between agents.

When several agents move from arbitrary initial positions to reach a formation, collisions may happen among them, and a collision avoidance strategy becomes essential for MUAVs formation control [12]. Besides, it is also important that the agents can reach their goal even if there are static obstacles in the environment. Hence, both collision and obstacle avoidance must be handled [13, 14]. One popular approach to deal with static obstacles has been the use of artificial potential fields (APFs). For instance in [15, 13], APFs have been used in centralized schemes where a map of the environment must be previously known. The formation flight among obstacles has also been addressed using classical approaches such as the reduced visibility graph optimizing the overall traveled distance [16]. The three aforementioned works describe centralized schemes requiring global measurements. There also exist distributed approaches to achieve formation of UAVs while avoiding obstacles. In [17], Voronoi partitions are combined with APFs to accomplish both goals. Consensus theory [18] is a backbone for the design of distributed formation control schemes. In [19, 20] formation of UAVs is tackled as a consensus problem and the collision/obstacle avoidance capability is reached by using a null space behavioral approach. All the aforementioned works are continuous-time schemes, which have poor flexibility to deal with additional constraints required in real situations.

A control technique that provides improved flexibility to consider additional constraints is Model Predictive Control (MPC) [21]. In the problem of formation control for quadcopters, each agent is subject to spatial constraints due to collision/obstacle avoidance and physical constraints, as limits on the control inputs. MPC is able to consider explicitly and simultaneously different control specifications in the form of constraints while also optimizing a global performance criterion. Thus, the constraints of the formation control of quadcopters can be introduced as constraints in the optimization problem formulated in an MPC problem. Several MPC-based schemes have been proposed for the formation control of UAVs. In [22], a centralized approach for global motion planning and control in cluttered, GPS-denied straitened environments is presented. In [23], a couple of centralized and decentralized control architectures are presented, with the trajectory of each agent obtained from the one of the leader in both cases. In [24], a leader–follower quadcopter formation flight control is presented for only one follower. An MPC controller generates a collision-free state reference and a robust feedback linearization controller is in charge of tracking that trajectory. In [25], a decentralized algorithm is proposed for the deployment and reconfiguration of a MAS in a convex bounded polygonal area, considering several outgoing agents. In [26], a distributed MPC scheme is presented for UAVs with nonholonomic dynamics, with the formation expressed in a virtual reference coordinate system. Avoidance of fixed obstacles and inter-vehicle collision avoidance are guaranteed by cost penalties. In [27], a decentralized scheme based on a triangular mesh is proposed for UAVs with nonholonomic dynamics. In the presence of obstacles, MPC is used to form independent sub-swarms, optimizing the formation spreading to avoid collisions. Note that although some of the schemes described above are decentralized, none of them exploits the advantages of the consensus theory to formulate distributed formation controllers.

One way to address MPC-based formation of UAVs is by pre-generating reference trajectories for each agent, while a preview controller ensures that the tracking error is minimized. For instance, in [28], the vehicles' trajectories are parameterized as polynomial B-splines and constraints on the relative agents positions are enforced to yield the formation. The consensus theory has also been used to formulate MPC-based formation controllers using pre-defined trajectories. In [29], a distributed consensus-based MPC scheme for formation of quadcopters is formulated within a leader-follower approach. The leader tracks the desired reference and generates the desired trajectories for the followers, without taking into account obstacles or collisions between agents. In [30], a cooperative formation control strategy is proposed for a group of quadcopters using decentralized MPC and consensus-based control. Static reference trajectories are pre-generated for each quadcopter based on the theoretical motion toward consensus and collisions are avoided using only the vertical motion.

Consensus-based formation controllers through MPC have also been proposed without relying on pre-computed trajectories. In [31], a scheme for formation control with collision avoidance for general linear systems has been described; the minimization of a consensus error is introduced in the MPC and the optimal reciprocal collision avoidance (ORCA) method is used. In [32], an MPC-based approach for consensus in first-order MAS is presented. The reference for the MPC is the local consensus error at each iteration. The decentralized protocol guarantees consensus for a switching communication topology when some conditions over the connectivity graphs are held. In [33], a global motion planning strategy using constrained optimization and consensus is shown to solve several high level tasks, including formation control. In [14], the authors propose a formation control scheme for multiple linear second-order agents. It handles collision and obstacle avoidance using distributed optimization.

Recent works have also tried to maintain the connectivity in the formation problem in addition of considering collision/obstacle avoidance. In [34], second-order MASs subject to both velocity and input constraints are considered, and a topology-based connectivity maintenance strategy is introduced. A switching nominal controller is combined with barrier function-based constraints to formulate a quadratic programming problem which modifies the nominal controller when necessary. In [35], a formulation of MPC including binary decision variables is proposed for the problem of maneuvering multiple agents that must visit a number of target sets, while enforcing connectivity constraints and avoiding obstacles as well as inter-agent collisions. Two mixed-integer linear programming formulations are presented considering a trade-off between optimality and scalability. In [36], a navigation controller for a general class of Lagrangian second order MASs in a bounded workspace with static obstacles is presented. The proposed scheme guarantees that the initially connected agents remain always connected using a decentralized MPC.

As it can be seen, a lot of theoretical work has been done for formation control of UAVs, using control schemes in continuous time and optimization-based techniques, in particular with MPC. Nevertheless, very few works include realistic simulations or experiments with real quadcopters. In this work, the formation control of a group of quadcopters is addressed via a distributed MPC scheme. It minimizes a position consensus error at each time step (dynamic formation control). We introduce virtual quadcopters that are related to the position of the real quadcopters by the desired displacements, such that if the virtual agents achieve consensus, the real ones reach the desired formation, as proposed in [26]. We propose soft and hard constraints in the MPC problem to address collision and obstacle avoidance as well as to maintain the same communication topology during the motion of the quadcopters. Maintaining the communication topology is important, for instance, in applications such as load transportation and vehicle guidance. The MPC scheme acts at kinematic level for the quadcopters position and we assume that a low level controller can execute the velocities commanded by the MPC and drive the vehicle to the adequate position. The main contributions of our work are the following:

- We formulate a distributed MPC-based formation control algorithm that considers the maintenance of connectivity, in addition to obstacle avoidance, as one of the objectives of the formation control task. To compute the velocities of the quadcopters, only relative information from neighboring agents and local measurements is required. From a continuous-time two-task scheme considering formation and obstacle avoidance, it is not straightforward to add the connectivity maintenance task without switching among dedicated controllers. That is why most of the existing approaches deal with two tasks [9, 13, 29, 30, 33, 20]. The MPC method facilitates the integration of the three tasks in a single control scheme.

- For both constraints (connectivity maintenance and collision/obstacle avoidance), we show that using soft and hard constraints in the MPC optimization problem gives better performance than using only one of the two. We propose the use of the average distance to obstacles, reducing the number of hard constraints in the optimization problem, unlike other works [23, 30, 14]. Together with a soft constraint in the form of a potential field, the formation is effectively achieved while avoiding collisions. This is a simple method in comparison with existing approaches as the ORCA method [31], hyperplane separation [28] or planning methods [33, 22].

- We evaluate the performance of our approach in both realistic simulations and real experiments for a group of quadcopters. To our knowledge, four works for formation control with evasion based on MPC have been proposed, but only one exploits consensus theory. Besides, [24] is limited to one follower, [22, 23] track precomputed references derived from a predefined leader trajectory, and [33] is more a motion planning technique than a controller that uses consensus to find an agreement on the free space.

To the authors knowledge, the closest works to ours on formation control with collision avoidance and connectivity maintenance are [34, 35, 36]; however, they are theoretical works, not focused on quadcopters and their approach to preserve connectivity differs from ours. Ours is simpler, since it treats connectivity maintenance similarly as collision avoidance. The proposed control scheme aims to maintain the same initial communication topology by using a single control scheme. This differs from [34], where the goal is to preserve a minimum cost spanning tree by modifying a switching nominal controller when necessary. The work in [35] is a task allocation and trajectory planning scheme, more than a formation controller, since the agents reach a fixed terminal target set, but without a predefined order. Another difference is that our algorithm uses local connectivity between pairs of quadcopters to preserve the initial topology, while the aforementioned work uses global information of the topology. In contrast with [36], we do not use predefined trajectories that must be provided by a high level planner. Since the consensus error is introduced in the MPC objective function, our approach adapts its global position according to the spatial constraints given by obstacles in the environment and to connectivity constraints.

The paper is organized as follows: Section 2 formulates the formation control problem for UAVs. In Section 3, the proposed distributed control scheme is detailed, including constraints for collision avoidance and connectivity maintenance. Section 4 analyzes the performance of our approach through simulations and real experiments. Finally, concluding remarks are presented in Section 5.

2. Problem formulation

Given a networked MAS, *reaching a consensus* means evolving towards an agreement state for all the agents, which should be achieved only by sharing local information among connected agents [18].

Let us consider a group of n quadcopters with position state $\boldsymbol{x}_i \in \mathbb{R}^3$ at time $\tau \in \mathbb{R}$ for $i = 1, \ldots, n$. A quadcopter is a nonlinear underactuated system in which four flat outputs (three position coordinates and yaw angle) can be independently controlled [37]. In the literature, feedback linearization has been used to achieve decoupled control for each position coordinate in a quadcopter formation control scheme [24]. The feedback linearization can be formulated in terms of velocities, accelerations or thrusts as control inputs; in the first case, using a low-level velocity controller, the quadcopter dynamics for each position coordinate can be modeled as a decoupled single-integrator [20]. In this work, we consider the modeling of the quadcopters position $\boldsymbol{x}_i = [x_i, y_i, z_i]^T$ through a single-integrator for each coordinate and velocities as control inputs $\boldsymbol{u}_i = [u_{x_i}, u_{y_i}, u_{z_i}]^T$ as follows

$$\dot{\boldsymbol{x}}_i(\tau) = \boldsymbol{u}_i(\tau) \in \mathbb{R}^3 \qquad \forall i \in \{1, 2, \ldots, n\}. \tag{1}$$

A consensus protocol for the quadcopters position must provide velocities for each quadcopter i such that the agreement in the position of all quadcopters in the network is accomplished. Since the consensus in position is not practical for real quadcopters, we define a desired configuration for the group of quadcopters to achieve a formation of their three dimensional position[1] by giving n displacement vectors $\boldsymbol{d}_i \in \mathbb{R}^3$ with respect to an arbitrary origin. As shown in [38] for generic agents, *virtual quadcopters* can be defined so that achieving *consensus in position* between those virtual quadcopters is equivalent to reaching the *formation* among the real quadcopters. The real quadcopters positions

[1] Note that the formation is defined only in terms of 3D positions, and that the final orientation of the quadcopters is arbitrary.

are denoted by $\boldsymbol{x}_i \in \mathbb{R}^3$ and the virtual agents positions by $\boldsymbol{z}_i \in \mathbb{R}^3$, so that

$$\boldsymbol{z}_i = \boldsymbol{x}_i - \boldsymbol{d}_i. \tag{2}$$

Since the vectors $\boldsymbol{d}_i$ are constant, then the real and virtual quadcopters of Eq. 2 share the same dynamics, driven by a velocity control input $\boldsymbol{u}_i \in \mathbb{R}^3$

$$\dot{\boldsymbol{z}}_i = \dot{\boldsymbol{x}}_i = \boldsymbol{u}_i. \tag{3}$$

In order to achieve consensus with asymptotic convergence between the virtual quadcopters, a consensus law is defined as follows

$$\boldsymbol{u}_i(\tau) = \sum_{j \in \mathcal{N}_i} (\boldsymbol{z}_j(\tau) - \boldsymbol{z}_i(\tau)), \tag{4}$$

where $\mathcal{N}_i$ is the set of neighbors of agent i. Denoting the consensus value, i.e. the common position, reached by the virtual quadcopters as $\boldsymbol{\alpha}_z$, the final state of any real quadcopter i will satisfy

$$\boldsymbol{x}_i - \boldsymbol{d}_i = \boldsymbol{\alpha}_z. \tag{5}$$

Since $\boldsymbol{x}_i = \boldsymbol{d}_i + \boldsymbol{\alpha}_z$, then, when consensus is achieved for the virtual quadcopters $\boldsymbol{z}_i$, the real ones $\boldsymbol{x}_i$ reach the shape defined by the vectors $\boldsymbol{d}_i$, up to an overall 3D translation given by the consensus value $\boldsymbol{\alpha}_z$.

In the framework of formation control of quadcopters based on consensus, we will consider the following assumptions:

1. For each position coordinate, the dynamics of the quadcopters, expressed in the state vector $\boldsymbol{x}_i(\tau) \in \mathbb{R}^3$, is modeled as a single-integrator (Eq. 1) since they are holonomic agents. The model is discretized by considering a constant sampling period $T \in \mathbb{R}^+$ as

$$\boldsymbol{x}_i(k+1) = \boldsymbol{x}_i(k) + T\boldsymbol{u}_i(k), \tag{6}$$

   where $\boldsymbol{u}_i(k) \in \mathbb{R}^3$ is the vector of velocity control inputs.

2. A low-level velocity controller with good performance is available. It could be, for instance, a feedback linearizing controller as in [24].

3. At initialization, the communication graph for the group of quadcopters is directed and has a spanning tree, which is accomplished by undirected connected topologies (see, for instance, [18, 39] for the details).

4. Obstacles are convex. They can be spherical (strongly convex) or generic convex polygons. In the last case, a sphere can be defined around the nearest obstacle point from the quadcopter (that point changes with the quadcopter's motion) and the evasion is similarly treated as a spherical obstacle.

We define the problem of interest as follows.

Definition 2.1 (Problem statement)**.** Consider a set $\mathcal{N}$ of n quadcopters with an associated communication graph $\mathcal{G}$ that satisfies Assumption 3. In the environment there exists a set $\mathcal{L}$ of l fixed obstacles. Given displacement vectors $\boldsymbol{d}_i$ defining the positions of the quadcopters within a desired formation, the problem consists in computing input velocities for each quadcopter $\boldsymbol{u}_i(k)$ to achieve: 1) Consensus of the virtual quadcopters' positions: $\lim_{k\to\infty} ||\boldsymbol{z}_i(k) - \boldsymbol{z}_j(k)|| = 0, \forall i, j \in \mathcal{N}, i \neq j$ and $\dot{\boldsymbol{z}}_i = 0, \forall i \in \mathcal{N}$; equivalently, the real quadcopters achieve the desired formation. 2) Avoidance of inter-quadcopter collision and of collision with obstacles: $||\boldsymbol{x}_i(k) - \boldsymbol{x}_j(k)|| \geq D, \forall i, j \in \mathcal{N}, i \neq j$ and $||\boldsymbol{x}_i(k) - \boldsymbol{x}_o^l|| \geq D, \forall l \in \mathcal{L}$ where $\boldsymbol{x}_o^l$ are the obstacles positions and D a safety distance. 3) Maintenance of connectivity between neighboring quadcopters: $||\boldsymbol{x}_i(k) - \boldsymbol{x}_j(k)|| \leq E, \forall i \in \mathcal{N}, \forall j \in \mathcal{N}_i$ with E a maximal distance for ensuring communication.

Notice that only relative quadcopter-quadcopter and quadcopter-obstacle information is required to tackle the defined problem. We will present the formulation of the proposed solution considering that the relative information can be computed using global coordinates; however, that information can be measured if the quadcopters are equipped with adequate onboard sensors.

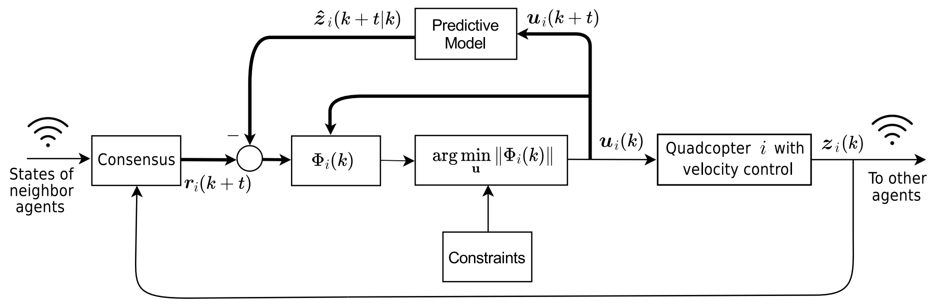


Figure 1: Model Predictive Control block diagram for each quadcopter i.

3. Proposed control scheme

In Fig. 1, we depict the Model Predictive Control (MPC) scheme proposed for each quadcopter. Each quadcopter i receives the position of its neighbors according to the communication topology and, using its position, computes a reference $\boldsymbol{r}_i(k)$ obtained from a consensus law. This reference must be followed by the virtual quadcopter position $\boldsymbol{z}_i(k)$ along a given horizon of size H_p. In order to determine the control input (quadcopter translational velocities), a cost function $\Phi_i(k)$ is minimized at each iteration k; this cost function is expressed in terms of the control input $\boldsymbol{u}_i(k)$ itself and of the consensus error computed from the reference $\boldsymbol{r}_i(k)$ and the predicted states $\hat{\boldsymbol{z}}(k+t)$ in the horizon window. The latter are given by a predictive model of the system along the prediction horizon. The thicker lines in Fig. 1 correspond to information obtained over the time horizon. $\Phi_i(k)$ is optimized while handling constraints that can be spatial constraints due to obstacles, connectivity constraints, or velocity limits, etc. The result of the optimization problem provides optimal inputs $\boldsymbol{u}_i^*(k+t|k)$ for all the timesteps t along a control horizon of size $H_u \leq H_p$, but only the control input of the present iteration $\boldsymbol{u}_i^*(k|k)$ is applied to the system. This process is then repeated at $k+1$ and subsequent timesteps.

3.1. MPC formulation

We propose the generic MPC cost function of Eq. 7 for the quadcopter i, which penalizes, on the one hand, the quadratic error between the predicted positions of the virtual quadcopter and a reference (to be defined later) and, on the other hand, the norm of the control inputs.

$$\Phi_i(k) = \lambda_i \sum_{t=H_w}^{H_p} ||\hat{\boldsymbol{z}}_i(k+t|k) - \boldsymbol{r}_i(k+t)||^2 + \gamma_i \sum_{t=1}^{H_u} ||\boldsymbol{u}_i(k+t-1|k)||^2. \quad (7)$$

The notation $\hat{\boldsymbol{a}}(p|q)$ in Eq. 7 refers to the *predicted* value of $\boldsymbol{a}$ at timestep p, with the last observation of $\boldsymbol{a}$ done at $q < p$. This prediction is obtained from the following predictive model of the quadcopter tridimensional position:

$$\hat{\boldsymbol{z}}_i(k+t|k) = \boldsymbol{A}\hat{\boldsymbol{z}}_i(k+t-1|k) + \boldsymbol{B}\boldsymbol{u}'_i(k+t-1), \quad (8)$$

where $\boldsymbol{u}'_i(k+t-1) \triangleq \boldsymbol{u}_i(k+t-1)$ for $1 \leq t \leq H_u$ and $\boldsymbol{u}'_i(k+t-1) \triangleq \boldsymbol{u}_i(k+H_u-1)$ for $H_u \leq t \leq H_p$. The dynamic system is defined by $\boldsymbol{A} = \boldsymbol{I}_3$ and $\boldsymbol{B} = T\boldsymbol{I}_3$, where $\boldsymbol{I}_3$ is the identity matrix of size 3 and T is the sampling control period.

The value H_p is the prediction horizon, i.e., the length of the window of predicted values for the quadcopter's position. The value $H_u \leq H_p$ is the control horizon, i.e., the number of future steps of control inputs to be optimized. Finally, $1 \leq H_w \leq H_p$ is the first timestep in the prediction horizon to be penalized. This is used because we may not want to start penalizing the error coming from the reference right since the current state.

At each iteration, we compute an optimal sequence of future velocity vectors

$$\boldsymbol{U}_i^* \triangleq \left[\boldsymbol{u}_i^*(k|k)^T, \boldsymbol{u}_i^*(k+1|k)^T, \cdots, \boldsymbol{u}_i^*(k+H_u-1|k)^T\right]^T \in \mathbb{R}^{3H_u}, \quad (9)$$

by solving the following optimization problem

$$\boldsymbol{U}_i^* = \underset{\boldsymbol{U}_i}{\arg\min}\ \Phi_i(k). \quad (10)$$

The velocity vector actually applied to quadcopter i, at each k, is the *first* term of the sequence in Eq. 9, i.e., $\boldsymbol{u}_i^*(k|k)$, discarding the rest of the sequence.

3.2. Distributed control

To build a distributed architecture, the control laws are computed in different processors for each quadcopter. We propose that the quadcopters share not only their current positions $\boldsymbol{x}_j(k)$ but also their (first) optimal velocity vector $\boldsymbol{u}_j^*(k|k)$ computed with neighboring quadcopters. This way, each quadcopter knows the movements to be performed by its neighbors in the near future. We consider virtual quadcopters as in Eq. 2 and aim at controlling them to achieve consensus, and consequently the real quadcopters will reach the desired formation.

In order to formulate the MPC problem, we propose a *dynamic reference* $\boldsymbol{r}_i(k+t)$ that considers the predicted consensus values computed from the current positions of the virtual quadcopters and from their predicted positions. Other approaches, as for instance in [30], use a *static* reference trajectory by applying a consensus law beforehand at the starting configuration. Using a predefined reference reduces slightly the computation time because $\boldsymbol{r}_i(k+t)$ does not have to be updated but it also reduces the flexibility of the control scheme. Our *dynamic reference* approach has the advantage to be able to continuously adapt the control inputs to the current situation and allows temporary variations, e.g. due to unforeseen obstacles.

Thus, our reference for the MPC cost function of quadcopter i is given by

$$\boldsymbol{r}_i(k+t) = \frac{1}{|\mathcal{N}_i(k)|+1} \sum_{j\in\mathcal{N}_i(k)\cup\{i\}} \hat{\boldsymbol{z}}_j(k+t|k), \tag{11}$$

and it should be clear that the errors to the reference are proportional to the consensus error

$$\hat{\boldsymbol{z}}_i(k+t|k) - \boldsymbol{r}_i(k+t) = \frac{1}{|\mathcal{N}_i(k)|+1} \sum_{j\in\mathcal{N}_i(k)\cup\{i\}} \hat{\boldsymbol{z}}_i(k+t|k) - \hat{\boldsymbol{z}}_j(k+t|k). \tag{12}$$

Hence, the objective function from Eq. 7 applied on the virtual quadcopters stands as a sum of predicted consensus errors and of control inputs. Quadcopter i considers that its neighbor $j \in \mathcal{N}_i$ will apply the same control input as the one that it has computed through its own MPC and that it has communicated to him along all the prediction horizon, that is

$$\begin{bmatrix} \hat{\boldsymbol{z}}_j(k+1|k) \\ \hat{\boldsymbol{z}}_j(k+2|k) \\ \vdots \\ \hat{\boldsymbol{z}}_j(k+H_p|k) \end{bmatrix} = \boldsymbol{P}_X \boldsymbol{z}_j(k) + \boldsymbol{P}_U \begin{bmatrix} \boldsymbol{u}_j^*(k|k) \\ \boldsymbol{u}_j^*(k|k) \\ \vdots \\ \boldsymbol{u}_j^*(k|k) \end{bmatrix}, \tag{13}$$

where $\boldsymbol{u}_j^*(k|k) \in \mathbb{R}^3$ is the communicated control input from j to i and $\boldsymbol{P}_X \in \mathbb{R}^{3H_p\times 3}$, $\boldsymbol{P}_U \in \mathbb{R}^{3H_p\times 3H_u}$ are defined in Appendix A as Eq. A.4 according to the dynamics in Eq. 8.

The reference $\boldsymbol{r}_i(k+t)$ depends not only on the predicted states $\hat{\boldsymbol{z}}_j(k+t|k)$ of the neighbors of quadcopter i, but also on the predicted positions of the quadcopter i itself. To solve the MPC problem of quadcopter i and take into account information along the prediction horizon, the reference must be introduced as an extended vector containing predicted positions from the current timestep to timestep H_p, as follows

$$\boldsymbol{R}_i(k+1) \triangleq \frac{1}{|\mathcal{N}_i(k)|+1} \begin{bmatrix} \sum\limits_{j\in\mathcal{N}_i(k)} \hat{\boldsymbol{z}}_j(k+1|k) \\ \sum\limits_{j\in\mathcal{N}_i(k)} \hat{\boldsymbol{z}}_j(k+2|k) \\ \vdots \\ \sum\limits_{j\in\mathcal{N}_i(k)} \hat{\boldsymbol{z}}_j(k+H_p|k) \end{bmatrix} + \frac{1}{|\mathcal{N}_i(k)|+1} \begin{bmatrix} \hat{\boldsymbol{z}}_i(k+1|k) \\ \hat{\boldsymbol{z}}_i(k+2|k) \\ \vdots \\ \hat{\boldsymbol{z}}_i(k+H_p|k) \end{bmatrix}. \tag{14}$$

Note that the second term of Eq. 14 depends on the future positions of the controlled quadcopter i, hence on the control inputs that are going to be optimized locally; when introducing $\boldsymbol{R}_i(k+1)$ in the optimization problem, this linear dependency is made explicit. Thus, the extended reference vector $\boldsymbol{R}_i(k+1)$ can be written in terms of the input vector along the control horizon

$$\boldsymbol{U}_i(k) \triangleq \left[\boldsymbol{u}_i(k|k)^T, \boldsymbol{u}_i(k+1|k)^T, \dots, \boldsymbol{u}_i(k+H_u-1|k)^T\right]^T. \tag{15}$$

By using the dynamics of the predicted position given in Eq. 8, the extended reference vector is written as a linear function of the input vector $\boldsymbol{U}_i(k)$

$$\boldsymbol{R}_i(k+1) = \boldsymbol{Q}_i(k) + \boldsymbol{S}_i(k)\boldsymbol{U}_i(k), \tag{16}$$

where

$$\boldsymbol{Q}_i(k) \triangleq \frac{1}{|\mathcal{N}_i(k)|+1}\begin{bmatrix} \sum\limits_{j\in\mathcal{N}_i(k)} \hat{\boldsymbol{z}}_j(k+1|k) \\ \sum\limits_{j\in\mathcal{N}_i(k)} \hat{\boldsymbol{z}}_j(k+2|k) \\ \vdots \\ \sum\limits_{j\in\mathcal{N}_i(k)} \hat{\boldsymbol{z}}_j(k+H_p|k) \end{bmatrix} + \frac{1}{|\mathcal{N}_i(k)|+1}\boldsymbol{P}_X\boldsymbol{z}_i(k) \in \mathbb{R}^{3H_p},$$

$$\boldsymbol{S}_i(k) \triangleq \frac{1}{|\mathcal{N}_i(k)|+1}\boldsymbol{P}_U \in \mathbb{R}^{3H_p\times 3H_u}. \tag{17}$$

This formulation allows each quadcopter to solve the optimization problem as a quadratic programming problem. Note that by writing the cost function of Eq. 7 in a standard quadratic form, we can find the analytical solution for the optimal control inputs as detailed in Appendix A. Besides, based on the results in [32], we can guarantee the stability of the closed-loop system to achieve consensus of the virtual quadcopters and therefore, the real quadcopters achieve the desired formation. The solution of the MPC problem, formulated up to now, solves the first point of the problem Definition 2.1; however, the tasks related to collision/obstacle avoidance and connectivity maintenance will be addressed in the next sections by including constraints to the optimization problem.

3.3. Collision and obstacle avoidance

Minimizing Eq. 7 with Eq. 11 as a reference along a prediction horizon will achieve a formation of the group of quadcopters with the shape defined by the vectors $\boldsymbol{d}_i$, but without guarantee about the absence of collisions between quadcopters or with obstacles in the environment. Thus, we add constraints in the optimization problem to ensure that the quadcopters do not collide and that they do not get close one to another less than a security distance $D > 0$.

We consider two types of constraints: *hard constraints*, which are constraints in the mathematical sense, i.e. they must by satisfied by a solution of the optimization problem for this solution to be considered as feasible, and *soft constraints*, which are a relaxed version of hard constraints, and can be violated. Soft constraints are implemented as a penalty in the objective function, where, typically, the greater the amount by which the constraint is violated, the greater is the penalty. This penalty makes the solver prefer solutions satisfying the constraint. In our formulation, the hard constraints are introduced in the optimization problem as linear inequalities that the quadcopters positions must satisfy (and, because they are linear, they can be coped with by QP methods), while the soft constraints are implemented as potential fields, added to the cost function of Eq. 7. Both types of constraints complement each other, both acting in favor of avoiding collisions between quadcopters and fixed obstacles as well as keeping connectivity. However, their range of impact on the solver is different: hard constraints are activated when the currently explored solutions are close to be non-feasible; soft constraints influence the search for a solution at a wider range.

We apply the following evasion strategies to other quadcopters from the group and to fixed obstacles located in the environment. In the latter case, we consider Assumption 4 of Section 2 and also assume that each quadcopter knows the position of the obstacles by measuring its relative distance to them, such that inter-quadcopter and quadcopter-obstacle evasions are handled similarly.

3.3.1. Hard constraints

For a quadcopter i, hard constraints for collision avoidance with another quadcopter $j \in \mathcal{N}_i$ along the prediction horizon are expressed in the form

$$||\hat{\boldsymbol{x}}_i(k+t|k) - \boldsymbol{y}_j(k+t|k)||^2 \geq D^2, \tag{18}$$

where $\boldsymbol{y}_j(k+t|k) \triangleq \hat{\boldsymbol{x}}_j(k+t|k)$, with $\hat{\boldsymbol{x}}_j(k+t|k)$ the predicted state of quadcopter j by quadcopter i along the prediction horizon, as given by Eq. 8 for the real quadcopters (using Eq. 2). For the case of a fixed obstacle in the environment whose position is $\boldsymbol{x}_o^l$, $\boldsymbol{y}_j(k+t|k) \triangleq \boldsymbol{x}_o^l$.

To handle a reduced number of constraints in the optimization problem, we consider the *average* distance between the quadcopter i and a neighbor j along the prediction horizon. This value is given by

$$\delta_{ij}^2 = \frac{1}{H_p}||\hat{\boldsymbol{X}}_i(k+1) - \boldsymbol{Y}_j(k+1)||^2, \tag{19}$$

where $\hat{\boldsymbol{X}}_i(k+1)$ stacks the predicted states for quadcopter i on the horizon

$$\hat{\boldsymbol{X}}_i(k+1) \triangleq \left[\hat{\boldsymbol{x}}_i(k+1|k)^T, \hat{\boldsymbol{x}}_i(k+2|k)^T, \cdots, \hat{\boldsymbol{x}}_i(k+H_p|k)^T\right]^T, \tag{20}$$

and $\boldsymbol{Y}_j(k+1)$ is defined as

$$\boldsymbol{Y}_j(k+1) \triangleq \begin{bmatrix} \hat{\boldsymbol{x}}_j(k+1|k) \\ \hat{\boldsymbol{x}}_j(k+2|k) \\ \vdots \\ \hat{\boldsymbol{x}}_j(k+H_p|k) \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{z}}_j(k+1|k) \\ \hat{\boldsymbol{z}}_j(k+2|k) \\ \vdots \\ \hat{\boldsymbol{z}}_j(k+H_p|k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{d}_j \\ \boldsymbol{d}_j \\ \vdots \\ \boldsymbol{d}_j \end{bmatrix}, \tag{21}$$

with $\hat{\boldsymbol{x}}_j$ the state of the quadcopter j predicted by quadcopter i.

Then, we consider just one hard constraint for each neighbor j, given by

$$\delta_{ij}^2 \geq D^2. \tag{22}$$

There may be up to $|\mathcal{N}_i|$ constraints of the type of inequality of Eq. 22 added to the optimization problem solved by the quadcopter i. This option to manage the collision avoidance is computationally cheaper than handling one constraint per timestep ($H_p|\mathcal{N}_i|$ constraints would be needed in that case); however, it does not ensure that all the predicted positions are away from the neighbors by more than D, as only the average is. We can rewrite the average distance of Eq. 19 explicitly in terms of the current state $\boldsymbol{x}_i(k)$ and of the control vector $\boldsymbol{U}_i(k)$, that stacks the future controls, as defined in Eq. A.1, and using Eq. A.3

$$\begin{aligned} \delta_{ij}^2(\boldsymbol{U}_i(k)) &= \frac{1}{H_p}||\boldsymbol{P}_X\boldsymbol{x}_i(k) + \boldsymbol{P}_U\boldsymbol{U}_i(k) - \boldsymbol{Y}_j(k+1)||^2 \\ &= \frac{1}{H_p}\Big[\boldsymbol{U}_i(k)^T\boldsymbol{P}_U^T\boldsymbol{P}_U\boldsymbol{U}_i(k) + 2\left(\boldsymbol{P}_X\boldsymbol{x}_i(k) - \boldsymbol{Y}_j(k+1)\right)^T\boldsymbol{P}_U\boldsymbol{U}_i(k) \\ &\qquad + ||\boldsymbol{P}_X\boldsymbol{x}_i(k) - \boldsymbol{Y}_j(k+1)||^2\Big]. \end{aligned} \tag{23}$$

Eq. 10 is optimized as a quadratic programming problem for $\boldsymbol{U}_i(k)$, and constraints in this type of formulation should be linear in $\boldsymbol{U}_i(k)$. As seen in Eq. 23, $\delta_{ij}^2(\boldsymbol{U}_i(k))$ is nonlinear in $\boldsymbol{U}_i(k)$ (it is quadratic). Hence, we linearize $\delta_{ij}^2(\boldsymbol{U}_i)$ around $\boldsymbol{U}_i = \boldsymbol{0}$. Let $\bar{\delta}_{ij}^2$ be this linear approximation defined as

$$\begin{aligned} \bar{\delta}_{ij}^2(\boldsymbol{U}_i(k)) &= \delta_{ij}^2(\boldsymbol{0}) + \nabla_{\boldsymbol{U}_i}\delta_{ij}^2(\boldsymbol{0}) \cdot (\boldsymbol{U}_i(k) - \boldsymbol{0}) \qquad (24) \\ &= \frac{||\boldsymbol{P}_X\boldsymbol{x}_i(k) - \boldsymbol{Y}_j(k+1)||^2}{H_p} + \frac{2}{H_p}\left(\boldsymbol{P}_X\boldsymbol{x}_i(k) - \boldsymbol{Y}_j(k+1)\right)^T\boldsymbol{P}_U\boldsymbol{U}_i(k). \end{aligned}$$

Then, the constraint of Eq. 22 is substituted by

$$\bar{\delta}_{ij}^{2} \geq D^{2}. \tag{25}$$

3.3.2. Soft constraints

Hard constraints for collision avoidance have the advantage of providing guarantees about whether the solution will exhibit collisions or not. However, they tend to be active (and, hence, modify the course of the quadcopter) only close to the obstacle (when the distance is potentially violating the constraint in the prediction horizon). That is why we add a penalty term to the cost function to be optimized in order to consider these potential collisions even at farther distances. This penalty term takes the form of a potential field and is given by

$$\rho_{ij} = \frac{k_d}{\left(\delta_{ij} - D\right)^2}, \tag{26}$$

where $k_d > 0$ is a constant associated with the smoothness of evasion and δ_{ij} is the average distance defined in Eq. 19. This penalty term in Eq. 26 becomes smaller as the quadcopters are farther, and increases as they approach one to the other or to the fixed obstacle. It is clearly not quadratic in $\boldsymbol{U}_i(k)$, hence, as we did above, we quadratize ρ_{ij} for keeping a quadratic programming formulation. By removing the 0th-order term that does not affect the optimum $\boldsymbol{U}_i^*(k)$, we can show that we only need to add the following linear penalty term to the cost

$$-k_\rho \left(\boldsymbol{P}_X \boldsymbol{x}_i(k) - \boldsymbol{Y}_j(k+1)\right)^T \boldsymbol{P}_U \boldsymbol{U}_i(k), \tag{27}$$

where $k_\rho = \dfrac{2k_d}{H_p \delta_{ij}(\mathbf{0}) \left(\delta_{ij}(\mathbf{0}) - D\right)^3}$, and to add to the Hessian matrix of the quadratic form, the following matrix

$$k_\rho \left[\frac{1}{H_p \delta_{ij}(\mathbf{0})} \left(\frac{1}{\delta_{ij}(\mathbf{0})} + \frac{3}{\delta_{ij}(\mathbf{0}) - D} \right) \right.$$
$$\left. \boldsymbol{P}_U^T \left(\boldsymbol{P}_X \boldsymbol{x}_i(k) - \boldsymbol{Y}_j(k+1)\right) \left(\boldsymbol{P}_X \boldsymbol{x}_i(k) - \boldsymbol{Y}_j(k+1)\right)^T \boldsymbol{P}_U - \boldsymbol{P}_U^T \boldsymbol{P}_U \right]. \tag{28}$$

3.4. Connectivity maintenance

The connectivity between quadcopters, that is defined through the communication graph G, usually depends on the distance between them. Hence, in order to preserve the initial connectivity between neighboring quadcopters, the distance between them should be top-limited by a constant E, which corresponds to the maximum distance for communication to operate satisfactorily. We rewrite the hard and soft constraints that we have used for obstacle avoidance in order to satisfy connectivity maintenance.

With the average distance between a quadcopter i and its neighbor $j \in \mathcal{N}_i$ along the prediction horizon δ_{ij}, given by Eq. 19, we write connectivity *hard constraints* as linear constraints in terms of $\boldsymbol{U}_i(k)$, as in Eq. 25. Considering the linear approximation of the average distance $\bar{\delta}_{ij}^2$ from Eq. 24, we use

$$\bar{\delta}_{ij}^2 \leq E^2. \tag{29}$$

Besides, a *soft constraint* is added as a penalty

$$\sigma_{ij} = \frac{k_c}{\left(\delta_{ij} - E\right)^2}, \tag{30}$$

where $k_c > 0$ and δ_{ij} is the average distance defined in Eq. 19. This penalty term becomes larger as the quadcopters are farther and close to the maximum distance E. For being used in the quadratic programming optimization, this term is quadratized, so that we add a linear penalty term to the cost function

$$-k_\sigma \left(\boldsymbol{P}_X \boldsymbol{x}_i(k) - \boldsymbol{Y}_j(k+1)\right)^T \boldsymbol{P}_U \boldsymbol{U}_i(k), \tag{31}$$

where $k_\sigma = \dfrac{2k_c}{H_p \delta_{ij}(\mathbf{0}) \left(\delta_{ij}(\mathbf{0}) - E\right)^3}$, and a quadratic term with a contribution of the following matrix to the Hessian

$$k_\sigma \Bigg[\frac{1}{H_p \delta_{ij}(\mathbf{0})} \left(\frac{1}{\delta_{ij}(\mathbf{0})} + \frac{3}{\delta_{ij}(\mathbf{0}) - E} \right) \boldsymbol{P}_U^T \left(\boldsymbol{P}_X \boldsymbol{x}_i(k) - \boldsymbol{Y}_j(k+1)\right) \left(\boldsymbol{P}_X \boldsymbol{x}_i(k) - \boldsymbol{Y}_j(k+1)\right)^T \boldsymbol{P}_U - \boldsymbol{P}_U^T \boldsymbol{P}_U \Bigg]. \tag{32}$$

Using both types of soft constraints, for obstacle avoidance (Eq. 26) and connectivity maintenance (Eq. 30), results for any pair (i, j) of neighboring quadcopters in summing both penalty terms, giving a potential field with two asymptotes for the average distance δ_{ij}. Avoidance of fixed obstacles is treated similarly once the relative position between quadcopters and obstacles is computed or measured. The Fig. 2 illustrates the individual and summed penalty terms. Note that, for the optimization problem designed with the proposed hard constraints (Eq. 25 for obstacle avoidance and Eq. 29 for connectivity maintenance) to find a solution, a necessary condition is the feasible set to be non-empty. Also note that the feasible set induced by our relaxed constraints on averaged distances includes the feasible set induced by the point-wise constraints. Moreover, using the corresponding penalty functions based on the averaged distances has a softening effect on the constraints that increases the feasibility of solution for the optimization problem, as analyzed in [40].

As mentioned in Section 3.2 and in Appendix A, stability of the formation control task alone is guaranteed; however, the introduction of hard and soft constraints requires a more extensive theoretical analysis to demonstrate the stability of the proposed MPC. Moreover, the linearization of the constraints might introduce some stability issues. In this work, the main goal was to present an effective implementation of an MPC-based formation control with obstacle avoidance and connectivity maintenance for a group of real quadcopters, and our achievements are detailed in the next section, first showing a preliminary evaluation in simulations. We left as future work a theoretical stability analysis that could be based on results of the literature on constrained MPC [41].

4. Simulation results and real experiments

This section presents an evaluation of the proposed approach in simulations and experiments with real quadcopters. To prove the concept, we consider strongly convex obstacles; however, as mentioned in the last assumption of Section 2, the obstacle avoidance strategy is also valid for unknown polygonal

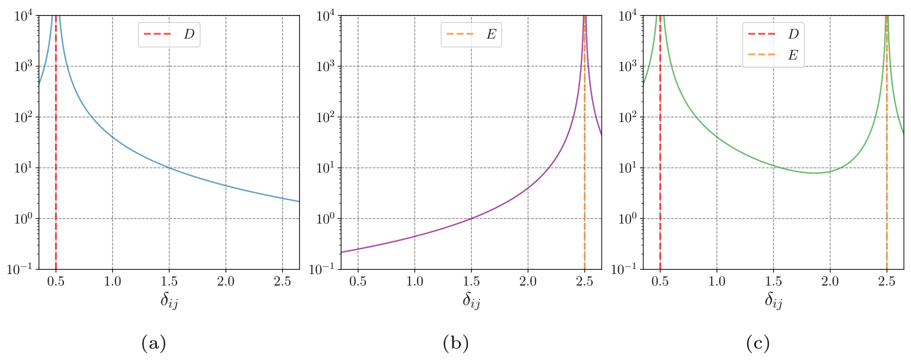


Figure 2: Shape of the soft constraints. (a) Obstacle avoidance (ρ_{ij}). (b) Connectivity maintenance (σ_{ij}). (c) Sum of both ($\rho_{ij} + \sigma_{ij}$).

obstacles, similarly as in [20]. We made a practical consideration for all the results that will be presented. The evasion between quadcopters is handled in the $x - y$ plane, considering the other quadcopters as obstacles modeled as infinite vertical cylinders. This is to exclude avoidance of two quadcopters in the z-axis, which may cause in practice important disturbances for the quadcopter who is at the bottom, due to air turbulence [42].

4.1. Comparison of control options

In this section, we present initial simulations in `Python` to evaluate three aspects of the proposed control scheme: control architecture, type of constraints (hard vs. soft) and reference trajectory (dynamic vs. fixed). In all the cases, some Gaussian noise is added to the position of each neighbor quadcopter, which gives certain randomness to the chosen evasion path.

4.1.1. Control architecture

We consider a variant of the proposed distributed architecture that we call *decentralized* control architecture. Actually, both architectures are decentralized, but the distributed one takes into account the knowledge of control inputs from other quadcopters. In the decentralized case, the control inputs for each quadcopter are computed separately but they are not sent to the other quadcopters, only the current state is shared. Thus, the reference value $\boldsymbol{r}_i$ is the same for all the prediction horizon at each iteration, because each robot does not have more information about what will occur to the neighbors. Then, to compute Eq. 11, the following is taken instead of Eq. 13:

$$\left[\hat{\boldsymbol{z}}_j(k+1|k)^T, \hat{\boldsymbol{z}}_j(k+2|k)^T, \cdots, \hat{\boldsymbol{z}}_j(k+H_p|k)^T\right]^T = \boldsymbol{P}_X \boldsymbol{z}_j(k).$$

Besides, for collision avoidance between quadcopters in the decentralized architecture, $\boldsymbol{y}_j(k+t|k) = \boldsymbol{x}_j(k)$ in Eq. 18. Thus, since quadcopter i only knows the position of its neighbors at iteration k and does not know in what direction they are going to move, the neighboring quadcopters are considered as *static* obstacles along the prediction horizon, i.e.

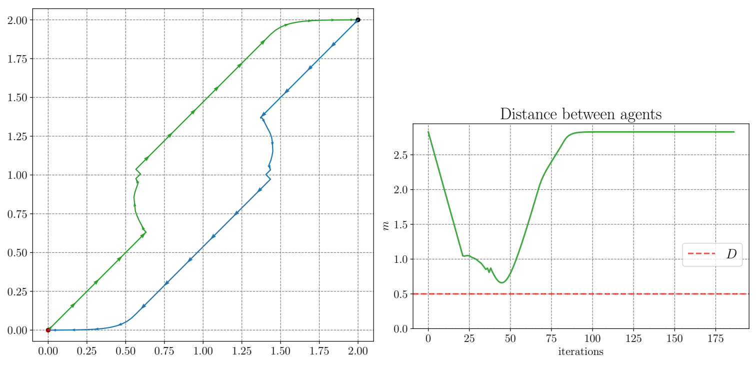
$$\boldsymbol{Y}_j(k+1) \triangleq \left[\boldsymbol{x}_j(k)^T, \boldsymbol{x}_j(k)^T, \cdots, \boldsymbol{x}_j(k)^T\right]^T,$$

in all the expressions of Section 3.3 and 3.4.

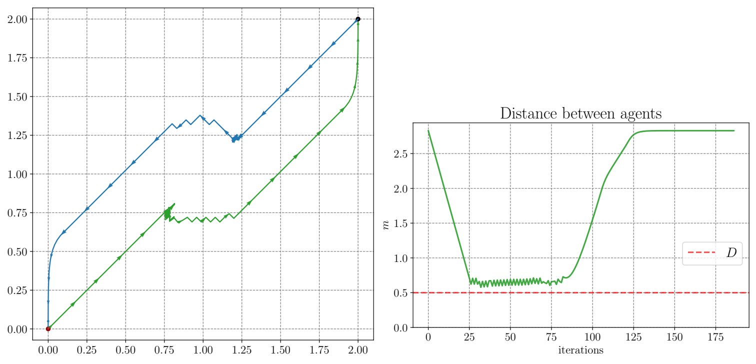
We evaluated both distributed and decentralized methods, in order to highlight the benefits of the distributed case. For that purpose, we present results of an experiment where two quadcopters start on two opposite corners of a square, at the same height. Then, each quadcopter tries to reach the opposite corner, making them going one towards the other.

Fig. 3 shows a comparison between the distributed and decentralized control architectures. In both cases, we use hard constraints on the average distance according to Eq. 25 and soft constraints as in Eq. 27. The use of both types of constraints simultaneously is discussed in the next section. In the distributed architecture, the evasion maneuver occurs earlier than in the decentralized case because it can anticipate the motion of the other quadcopter by knowing its control inputs. Besides, the trajectories of the quadcopters in the distributed architecture are smoother in comparison to the ones generated in the decentralized case. For the latter, the distance between quadcopters shows oscillations during the evasion maneuvers and they endure for a large range of iterations.

Regarding the computed control inputs, Fig. 4 presents, in the top, the computed controls for the distributed architecture, and, in the bottom, the



(a) Distributed architecture.



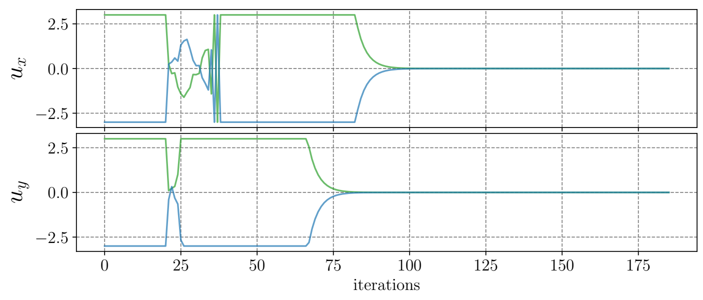
(b) Decentralized architecture.

Figure 3: Comparison of the proposed distributed control architecture versus a decentralized architecture. In both cases, hard and soft constraints are used for collision avoidance.

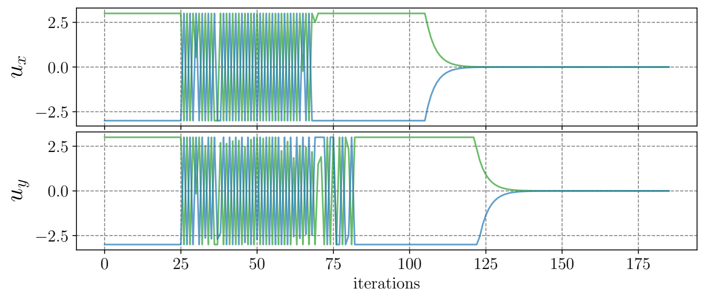
controls for the decentralized one. In both cases, the range of allowed control inputs is set as $[-3, 3]$. It can be seen that the computed controls oscillate drastically less in the distributed architecture. Also, it can be verified that the evasion effect in the controls starts earlier for the distributed case. This is caused by the information that each quadcopter has in advance regarding the actions to be taken by its neighbors.

4.1.2. Type of constraints

In this section, we evaluate the different options to define the constraints in the MPC, in particular hard constraints on the average distance according Eq. 25, soft constraints in the linear form of Eq. 27 and both types of constraints



(a) Distributed architecture.



(b) Decentralized architecture.

Figure 4: Computed controls for the evasion experiments of Fig. 3.

acting simultaneously. We use the same experiment as in the previous section, with only two quadcopters.

Fig. 5 shows a top view of an experiment. On the left is the evasion using only average hard constraints of the form of Eq. 25; the evasion maneuver is performed around $(0.62, 0.62)$ for the green agent and around $(1.37, 1.37)$ for the blue one. The evolution of the distance between quadcopters shows a smooth evolution and has a minimum value of 0.54 around iteration 75, without violating the security distance $D = 0.5$. On the right, we present the result using only soft constraints of the form of Eq. 27, showing that the task is not completed in this case. Both quadcopters reach a symmetric position where they start oscillating permanently forwards and backwards. This issue is alleviated by combining both types of constraints in the proposed control scheme.

The case of using both types of constraints has been presented in Fig. 3, where the task is effectively finished; the evolution of the distance between

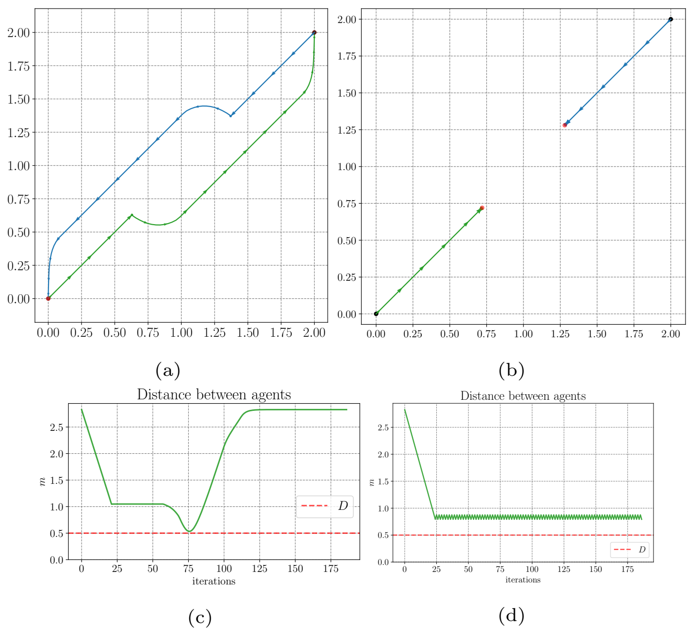


Figure 5: Decentralized evasion between two quadcopters. (a) and (c): using only the average hard constraints of the form (25). (b) and (d): using only the linear soft constraints of the form (27). The graphs (a) and (b) give the trajectories in the $x - y$ plane; the graphs (c) and (d) give the evolution of the distance between agents during the experiment.

quadcopters is smooth and has a minimum value of 0.65 around iteration 45, which is a better behavior than using only hard constraints. Even with the decentralized architecture, using both types of constraints solves the task effectively, although with a poorer performance in terms of trajectory smoothness.

To extend the analysis of the use of the constraints options, we evaluate the ratio of success of each evasion strategy for more quadcopters. We performed 1000 evasion experiments for two, three and four quadcopters; each agent is initially positioned at the corner of a square and attempts to reach the opposite corner. Each experiment was performed using either only average hard constraints, only soft constraints or both, for the distributed and decentralized architectures. The results are shown in Table 1. Note that a case is considered as successful when the quadcopters reach their reference and collision between them was never detected, i.e., the distance between the quadcopters was never lower than the security distance D.

N. quadcopters	Distributed			Decentralized		
	Avg. hard	Soft	Both	Avg. hard	Soft	Both
2	100%	0%	100%	100%	0.6%	100%
3	90.6%	71.4%	100%	0%	100%	100%
4	25.6%	12.9%	99.2%	0%	99.8%	100%

Table 1: Percentage of success for evasion maneuvers between different number of quadcopters using the three options of evasion strategies, in distributed and decentralized architectures.

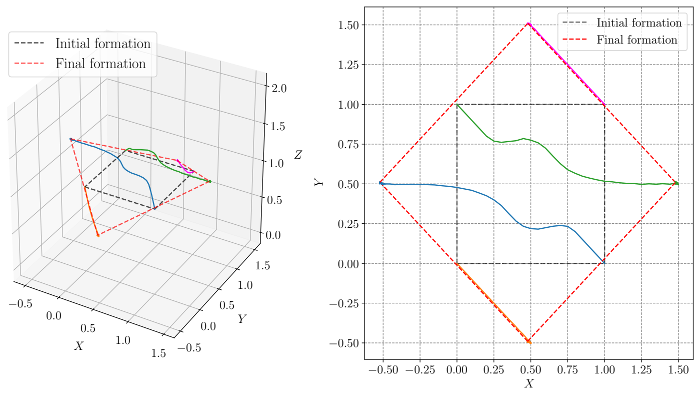
It is interesting to notice that neither the average hard constraints alone, nor the soft constraints alone have a good performance for all the evasion cases. However, the use of both types of constraints is effective for the tested evasion cases, in both distributed and decentralized control architectures. We conclude that the best avoidance strategy consists on using both types of constraints, soft and average hard. They allow to reach the reference and to reduce oscillations when the distance between the quadcopters is near the security distance D.

4.1.3. Reference trajectory

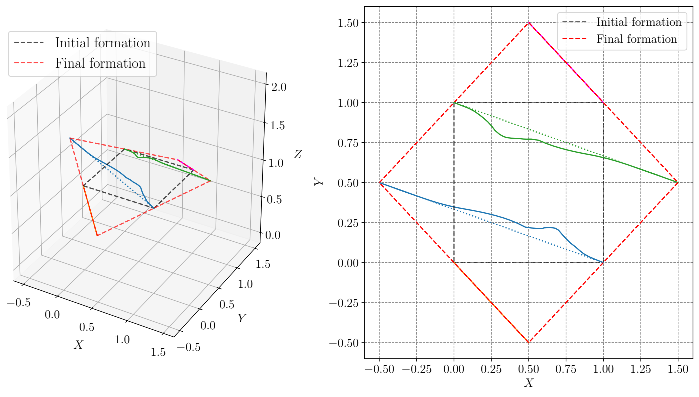
In this section, we compare two options to generate the reference $\boldsymbol{r}_i$ required in the cost function in Eq. 7. The reference can be generated in advance by using the theoretical value of consensus or can be dynamically specified by Eq. 11. Discrete reference trajectories $\boldsymbol{r}_i(k)$ for each virtual quadcopter $\boldsymbol{z}_i$ can be generated and predefined by using Euler integration with sampling period $T_r = 0.01s$ and the consensus protocol in Eq. 4. Knowing the initial conditions of the quadcopters, we define $\boldsymbol{r}_i(0) = \boldsymbol{z}_i(0) = \boldsymbol{x}_i(0) - \boldsymbol{d}_i$ for $i = 1, \ldots, n$, and the reference trajectory for each virtual quadcopter is described recursively by

$$\boldsymbol{r}_i(k+1) = \boldsymbol{r}_i(k) + T_r \sum_{j \in \mathcal{N}_i} (\boldsymbol{r}_j(k) - \boldsymbol{r}_i(k)). \tag{33}$$

Fig. 6 shows a comparison between using the dynamic reference or the predefined consensus trajectories of Eq. 33 in a decentralized architecture for 4 quadcopters. The experiment is designed in such a way that two quadcopters must avoid each other. The predefined trajectories consist on straight lines from the initial configuration to the target formation. However, when evasion constraints are added, the final trajectories differ from the reference ones, but the evasion between the blue and green quadcopters does not affect the trajectories of the other two. In the case of using the dynamic reference, the trajectories are longer and curvy, even for the quadcopters that do not need to evade.



(a) Dynamic reference.



(b) Predefined reference (dotted lines).

Figure 6: Decentralized formation experiment comparing the use of a dynamic or predefined reference in the MPC.

Although straighter trajectories are generated using the predefined reference, its main disadvantage arises with the presence of fixed obstacles in the environment. In this case, the achievement of the formation cannot be guaranteed because, if the final position of the predefined trajectories is obstructed by an obstacle, then the quadcopters will not be able to reach the desired configuration. The dynamic reference for the MPC has the advantage to adapt to different environments, even in the presence of fixed obstacles and even when the evasion maneuvers of one quadcopter change the dynamic consensus reference.

Given these observations, our proposed formation control strategy is summarized as an MPC using a dynamic reference given by the local consensus error between virtual quadcopters, both average hard constraints of the form of Eq. 25 and soft constraints of the form of Eq. 27 adapted for horizontal evasion, and both average hard constraints of the form of Eq. 29 and soft constraints of the form of Eq. 31 for connectivity maintenance, all of this using a distributed control architecture. Connectivity maintenance is evaluated in the next section.

4.2. Experiments in a dynamic simulator

The proposed control strategy has been implemented in the `Gazebo` simulator [43] using the `rotorS` package [44] for the formation of 4 quadcopters in 3D and with a fixed obstacle as a column located at $(0.5, 0.5)$. A fully connected undirected communication graph is considered and the proposed MPC scheme aims for maintaining this connectivity all the time. At each iteration k, our scheme computes the linear velocities $v_x(k)$, $v_y(k)$ and $v_z(k)$, that are then integrated into positions by using the Euler forward formulation with sampling period $T_r = 0.01s$. Then, the estimated position is passed as a reference to the position controller of the `rotorS` package. The sampling control period was set $T = 0.1s$, the weight $k_d = 10.0$ and the limits of control inputs $[-5.0, 5.0]$.

Important parameters in a MPC formulation are the prediction horizon H_p and the control horizon H_u. A large prediction horizon results in better performance; however, a larger prediction horizon also increases the computational load. In our case, we found experimentally a good compromise between performance and computational load by setting $H_p = 15$ and $H_u = 10$. These relatively large values are adequate for the possibly fast motion of the quadcopters; lower values generate jittering movements and larger values do not improve the performance significantly, hence it is not worth the increment in computational burden. This is shown at the end of the section.

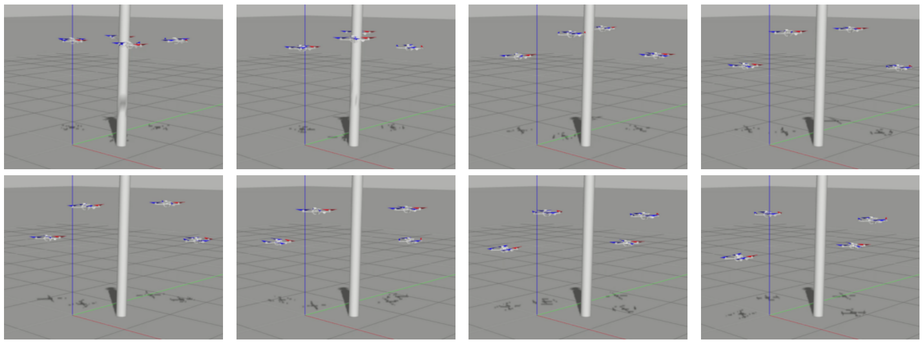
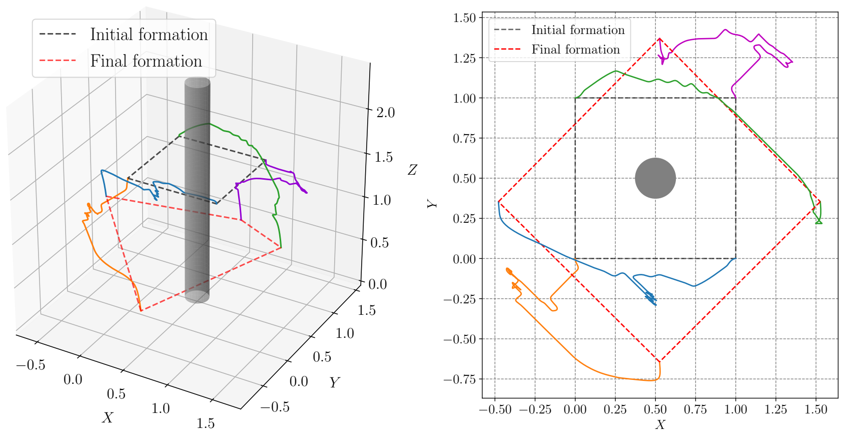


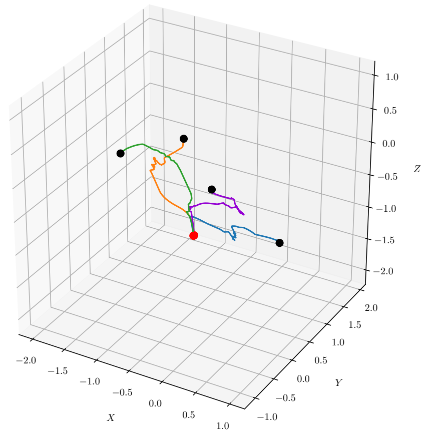
Figure 7: Evolution of the quadcopters for a formation experiment in `Gazebo` with connectivity constraints.

Some snapshots of the evolution of the quadcopters during the formation experiment can be seen in Fig. 7 and a video of the whole experiment can be found at `https://youtu.be/X8HgnmrU3e4`. The details of the quadcopters trajectories are presented in Fig. 8, in 3D and in the $x - y$ plane. Also, the same figure shows how the virtual quadcopters reach consensus. In this experiment one can observe that two quadcopters have to wait for the others two to move, and then they complete the formation. During their motion, the purple and orange quadcopters are in risk of separating more than the connectivity limit E from the others and the connectivity constraints make that these two agents wait for the others to move. Due to the connectivity constraints and given the waiting effect for some quadcopters to prevent disconnection, the desired formation is achieved in more iterations than an experiment without connectivity maintenance, but the task is effectively completed.

Fig. 9 shows the distance between the quadcopters and to the fixed obstacle. We observe that the distance between the more distant quadcopters, the orange and purple ones, gets close to but do not reach the connectivity limit E. Nevertheless, at some point, the distance between the green and purple quadcopters violates the evasion limit D. This can be avoided by increasing the gain k_d of



(a) 3D view (left) and upper view (right).



(b) Trajectories of the virtual quadcopters reaching consensus.

Figure 8: Trajectories generated by `Gazebo` for the formation experiment of Fig. 7 with connectivity maintenance.

the penalty evasion term. However, this may jeopardize the guarantee of no traversing the connectivity limit. Hence, a good trade-off must be found.

Fig. 10 presents the computed controls for this formation experiment with connectivity constraints. Due to these constraints, some oscillations are observed but the controls stay within the predefined limits.

To conclude this section, a parameter validation for the prediction horizon

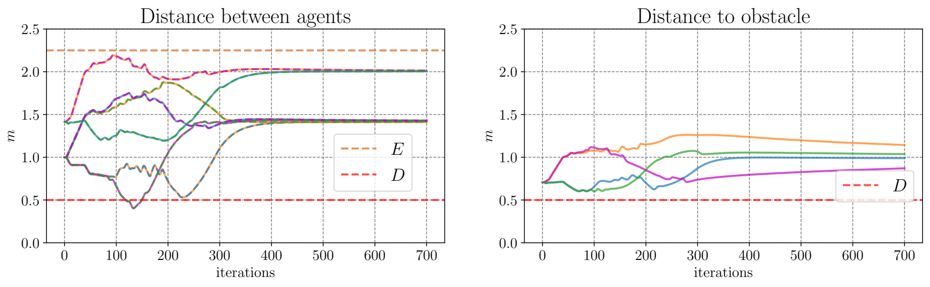


Figure 9: Distance between quadcopters and to the obstacle for the distributed formation control of Fig. 7.

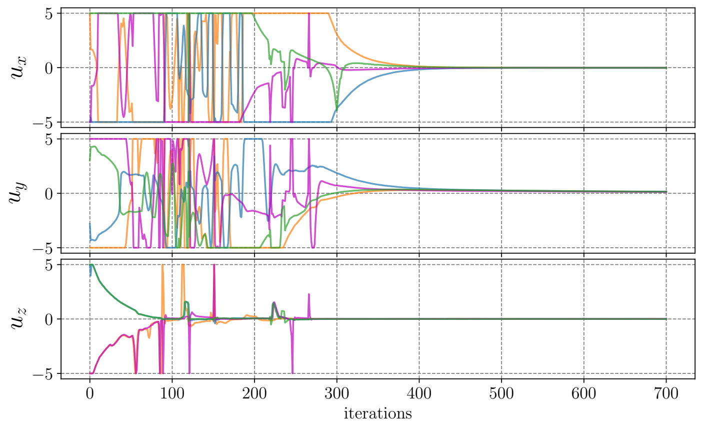
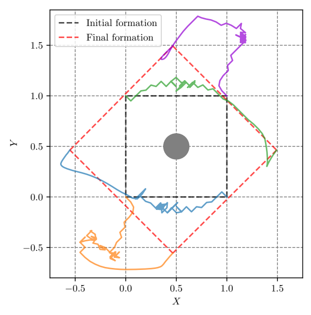
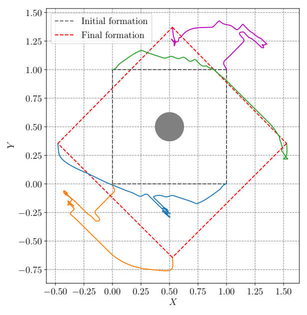


Figure 10: Computed controls for the formation experiment of Fig. 7.

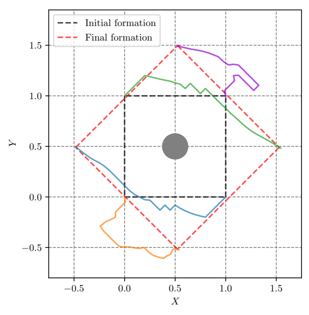
H_p is presented. Fig. 11 shows the trajectories followed by the quadcopters for prediction horizons 5, 15 and 25. The behavior with $H_p = 5$ is very reactive and the quadcopters present jittering movements. This undesired effect is reduced with $H_p = 15$ and the trajectories are almost smooth with $H_p = 25$, however, the cost to pay is an important increment in the time to solve the optimization problem, as shown in Fig. 12. Moreover, as H_p is larger, the optimization problem is more complex and the probability of not finding a feasible solution increases. Similar experiments to the one shown here with $H_p = 15$ were performed 100 times and in all the cases convergence to the desired formation was achieved, not so for $H_p = 25$, where only 35 executions were successful. Therefore, the intermediate value $H_p = 15$ was selected.



(a) $H_p = 5$.

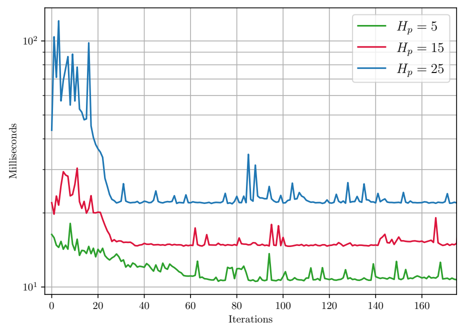


(b) $H_p = 15$, selected.

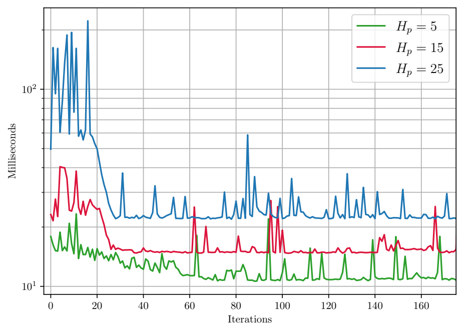


(c) $H_p = 25$.

Figure 11: Trajectories of the quadcopters for different prediction horizons.



(a) Mean value between the 4 quadcopters.



(b) Maximum value between the 4 quadcopters.

Figure 12: Times to solve the optimization problem for the cases of Fig. 11.

4.3. Experiments with real quadcopters

We ran a few formation experiments with *Bebop 2.0* quadcopters from *Parrot*®. We used an Optitrack motion capture with a dedicated computer running the Motive software so that the position of every quadcopter is known all along the experiment. Each quadcopter has reflecting markers that are tracked by the set of cameras of the motion capture system. These positions are read by each quadcopter, at each iteration, in order to know where its neighbors are.

The control velocities are calculated in a different computer for each quadcopter and then communicated to the aircraft via WiFi using the `bebop_autonomy` driver [45], so the computations are not performed on board. Each computer for control used a virtual reality peripheral network (VRPN) client for `ROS` to get the data at 100 Hz. The `bebop_autonomy` driver allows us to directly publish velocity commands to the *Bebop 2.0* quadcopter that uses its own low level control to transform the velocity commands into velocities for each rotor, and also to read the computed controls published by its neighbors.

Formation experiments with three quadcopters have been performed. In this case, one of the quadcopters stays at hovering along the experiment, acting as a leader, resulting on a communication graph given by the adjacency matrix

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix},$$

where the static quadcopter sends its states to the other two but does not receive any information.

The computers used to compute the velocities of the two active quadcopers had the following features: CPU Intel Core i7-4790 @3.6GHz$\times$8 and CPU Intel Core i7-3632QM @2.2GHz$\times$8, both with 16 GB of RAM and running Ubuntu 16.04 and `ROS` Kinetic. Each computer solved independently the optimization problem using the Operator Splitting for Quadratic Programs (OSQP) solver [46], taking as input the data from neighbors. In each computer and each iteration, the time to solve the optimization problem varied due to the changes in the constraints, but the time for each iteration was always inferior to $70ms$ for $H_p = 15$. Therefore, in order to make the sampling period uniform for all the quadcopters, the parameter T was set to 0.1 seconds.

In the experiments with real quadcopters, the limits of the control inputs were set as $[-0.03m/s, 0.03m/s]$. These low values were used in order to avoid an undesirable inertia effect that we observed when the quadcopters must change their direction or must stop according to the MPC. This is due to a poor performance of the low-level controller of the quadcopters. Besides, the parameter γ_i of the cost function in Eq. 7 that penalizes the size of the control inputs was set large between $[750, 1000]$ in order to reduce the occurrence of control inputs

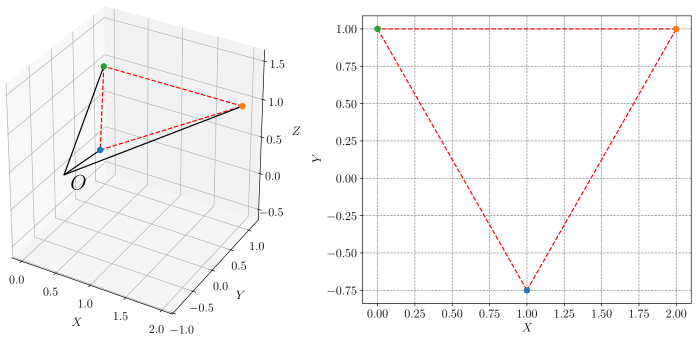


Figure 13: Target formation for experiments with three *Bebop 2.0* quadcopters.

saturation, resulting on less problems associated with the quadcopter inertia.

In addition, we use an adaptive value for λ_i, depending on the consensus error e_i for each coordinate at the current iteration, given by

$$\lambda_i(e_i) = \begin{cases} \lambda_{\max} - \dfrac{\lambda_{\max} - \lambda_{\min}}{e_r} e_i & e_i < e_r, \\ \lambda_{\min} & e_i \geq e_r, \end{cases} \tag{34}$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are bounds for λ_i and e_r is a predefined level of error. This function $\lambda_i(e_i)$ produces low values when the consensus error is high, i.e., when the quadcopters are far from the target formation. In this case, the computed controls focus in a slower and controlled movement. When the error starts to decrease and the computed controls decrease, the value of λ_i increases and penalizes more the consensus error, resulting in higher computed inputs that can lead the quadcopters to achieve the target formation more accurately.

Here we present the results for two experiments: a formation control experiment for three quadcopters without obstacle, and the same experiment but with a single column obstacle. The dimensions of the space where the quadcopters could navigate were of $4.75m \times 3m$ and the limits of this space were introduced as hard constraints for the optimization problem. Due to the lack of physical space, we did not perform experiments with connectivity constraints. Also, to ensure that the formation was achieved within the available space, one of the three quadcopters remains in hovering state along all the experiments, i.e., it is not controlled and is static, acting as an anchor for the formation. Fig. 13 shows the target formation used along the experiments.

Both experiments used the same parameters: The prediction horizon H_p is 15 and the control horizon H_u is 10. The evasion penalty gain is $k_d = 1$. The control penalty parameter is $\gamma_i = 1000$ and the consensus error penalty parameter was between $\lambda_{\min} = 20$ and $\lambda_{\max} = 100$, with a threshold $e_r = 3$.

4.3.1. Formation experiment without obstacles

Fig. 14 shows some snapshots of the evolution of a formation experiment with 3 *Bebop 2.0* quadcopters as seen from the top. Two quadcopters start at the right and the third one at the lower left. At the end of the experiment the three quadcopters reach the desired configuration presented in Fig. 13. A video of this experiment can be consulted at `https://youtu.be/5-GtJJBk8Zo`.

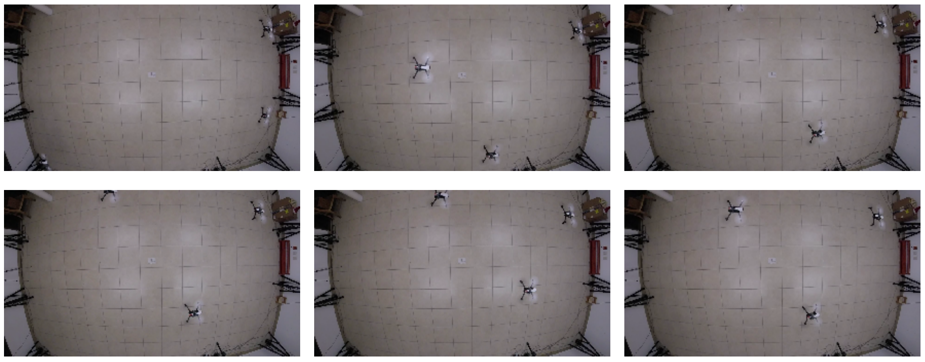


Figure 14: Evolution of a formation experiment with 3 real *Bebop 2.0* quadcopters.

The generated paths for the three quadcopters, along with the initial and final formation, can be seen in Fig. 15. The green quadcopter moves practically in straight line until it reaches the vicinity of its target position. The blue quadcopter tries to minimize its error with respect to the green one and that is why its path approaches to $(0, -1)$; but when the green quadcopter is closer, the blue one gets also closer to its target position.

Oscillations can be observed near the target position as the quadcopters try to converge to the desired formation. These oscillations might be reduced by

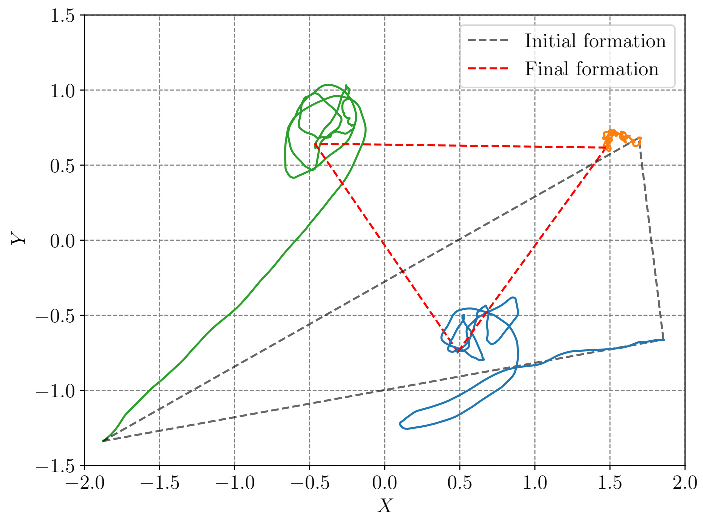


Figure 15: Paths generated by the formation experiment of Figure 14.

improving the accuracy of the low-level velocity controller. Finally, even though the orange quadcopter is not controlled, we can see some displacement due both to its inner hovering control and to the external disturbance caused by the turbulence of the other quadcopters. Nevertheless, the controlled quadcopters adjust to this movement, reaching their target position.

As can be seen in Fig. 15, there is no risk of collision along the experiment. Therefore in Fig. 16 we observe that the distance between quadcopters does not get close to the security distance.

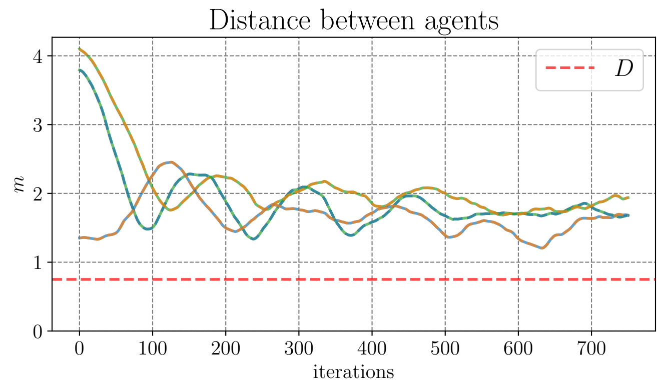


Figure 16: Distance between quadcopters in the formation experiment of Fig. 15.

Finally, the computed control inputs appear in Fig. 17. We observe very little saturation, because of the very high value of the control penalty parameter and the adaptive value of consensus error penalty parameter. The velocities oscillate but decrease in amplitude, allowing the system to converge. The computed control input of the orange quadcopter remains in 0 as is not controlled.

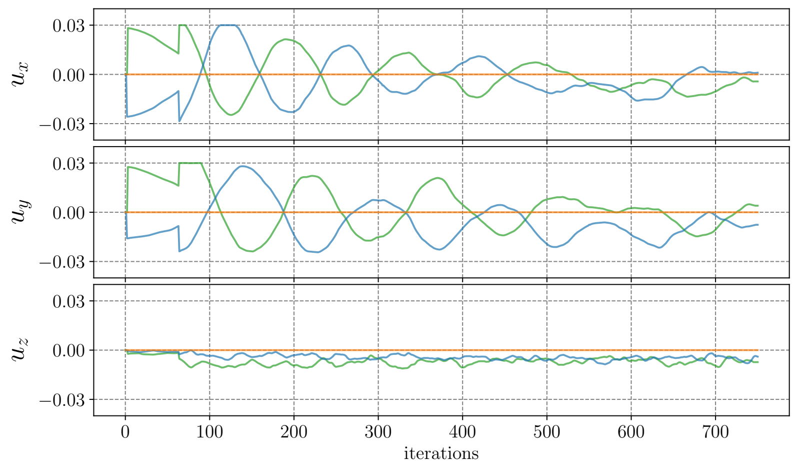


Figure 17: Computed control inputs for the formation experiment of Fig. 15.

To close the case of experiments without obstacles, we highlight that experiments with 5 quadcopters in a larger workspace were realized. One of these experiments is described in `https://youtu.be/DYwxDR39Rp0`, where the quadcopters are initially aligned and they effectively reach the desired formation. In the video, we also present the evolution in time of the important variables.

4.3.2. Formation experiment with an obstacle

Fig. 18 shows some snapshots of the evolution of a formation experiment with 3 *Bebop 2.0* quadcopters with an obstacle, seen from the top. The target formation is the same as in the previous experiment, given by Fig. 13. The initial configuration is very close to the previous one, with two quadcopters aligned at the right and the third one at the bottom left corner. The obstacle is the white column observed in the scene and is located at the right of the quadcopter in the left, with the purpose of avoiding a straight trajectory for this agent. This experiment can be seen at `https://youtu.be/qaPfFxFETvc`.

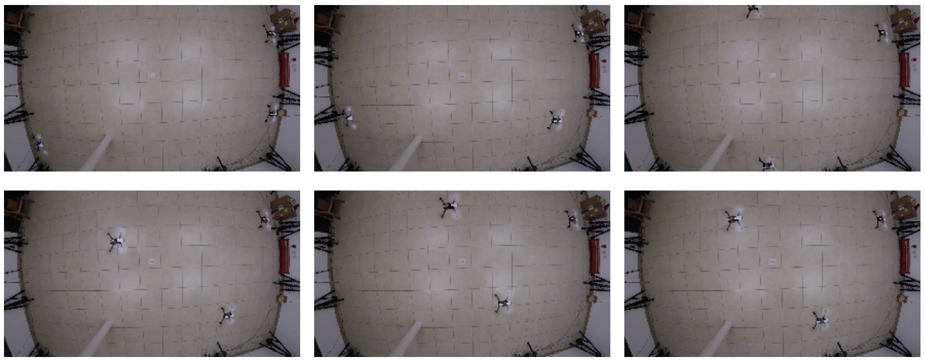


Figure 18: Evolution of a formation experiment with 3 real *Bebop 2.0* quadcopters with an obstacle.

As can be seen in Fig. 19, the green quadcopter does not follow a straight trajectory anymore, as it has to evade the obstacle. The trajectory of the blue quadcopter is very similar to the one of the previous experiment, trying to decrease the error relative to the green quadcopter first, and when this one is closer, converging to the desired formation. In this experiment, the movement of the noncontrolled orange quadcopter has a lower amplitude than in the previous experiment, but it can still be appreciated.

The distance between the quadcopters and to the obstacle along the experiment can be seen in Fig. 20. Initially, the distance between the green quadcopter and the obstacle is close the security distance, and following a straight trajectory would have led in passing this limit, that is why the green quadcopter moves along a curved trajectory far from the obstacle. The obstacle stops the movement of the blue quadcopter to the left around the iterations 100 and 150. Even though the gain for the evasion penalty term $k_d = 1$ is much lower than the control penalty parameter $\gamma_i = 1000$ or the consensus error penalty parameter whose lower value is 20, it is enough to avoid collisions.

Finally, Fig. 21 shows the computed controls. Some saturation can be observed during the first 100 iterations, and then again between iterations 100 and 150 for the blue quadcopter when there is a need to evade the obstacle and to

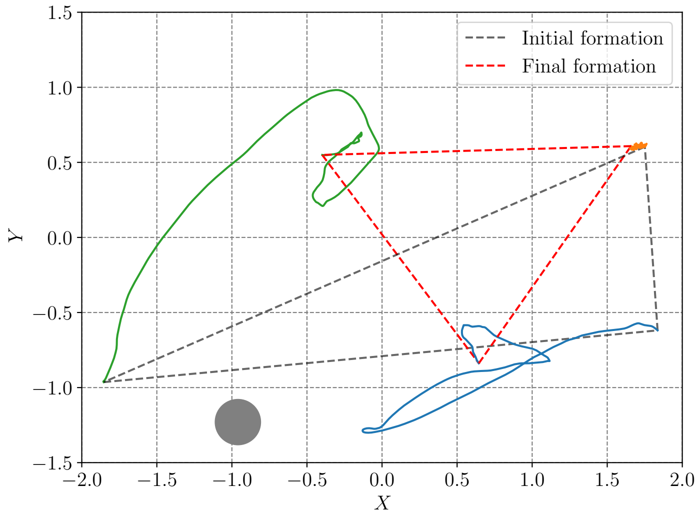


Figure 19: Paths generated by the formation experiment of Fig. 18.

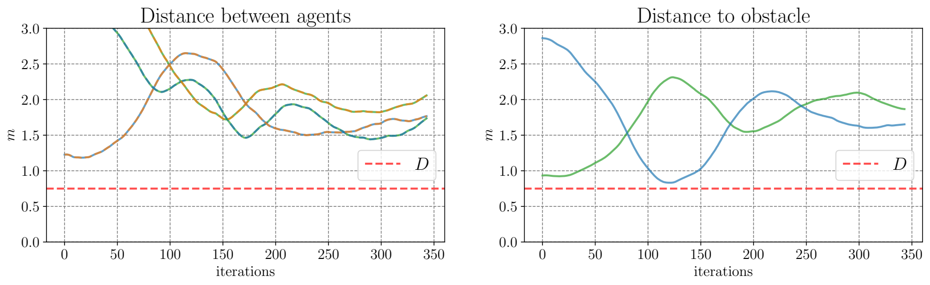


Figure 20: Distance between quadcopters and to the obstacle in the formation experiment of Fig. 19.

adapt to the movement of the green quadcopter. Again, the controls oscillate but with decreasing amplitude, leading to convergence to the formation.

5. Conclusions and future work

In this paper, we have addressed the formation control of a group of quadcopters via a distributed model predictive control scheme. A dynamic formation control was proposed, in which a position consensus error is minimized at each control cycle. The formation is achieved using a consensus strategy with virtual quadcopters defined such that if the virtual agents achieve consensus, the

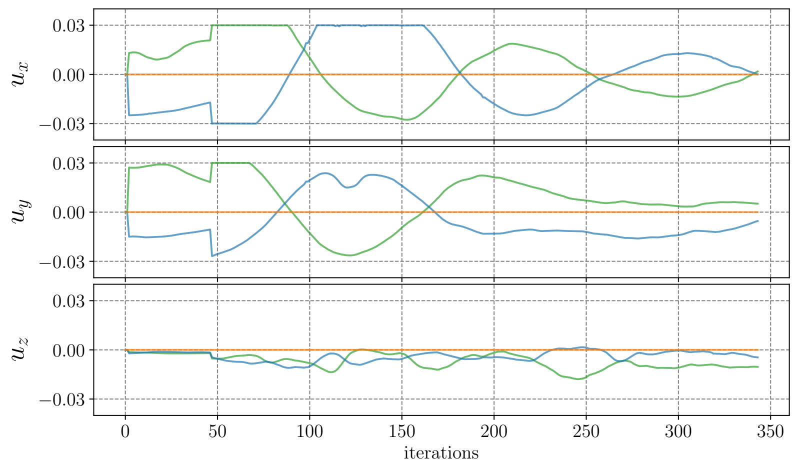


Figure 21: Computed control inputs for the formation experiment of Fig. 19.

real quadcopters reach the desired formation. We have integrated an effective solution for three different tasks of a system with multiple quadcopters: formation control, collision/obstacle avoidance and connectivity maintenance. We have introduced soft and hard constraints for the MPC problem to deal with these tasks and we have shown that using both kinds of constraints give better performance than using only one of the two. The advantages of the proposed approach are that only relative positions between neighboring quadcopters and local measurements to avoid obstacles are needed; hence, our approach might be implemented without using a global coordinate system nor a map of the environment. Moreover, only local connectivity between pairs of quadcopters is required to preserve the initial communication graph, without the need of global information about the topology. The approach is free of predefined trajectories, providing flexibility to achieve the goals of the three tasks without relying on a high level motion planner. Our kinematic control scheme was evaluated in realistic simulations and experiments with real quadcopters in indoors settings, showing its effectiveness to solve the MPC problem in a distributed fashion.

As future work, the proposed approach could be extended to the leader-following configuration, which might allow us to implement the approach for long-distance navigation of a group of quadcopters in formation in more general environments with complex obstacles while connectivity is guaranteed. In principle, one could ensure to reach the formation without using a common frame, by using the notion of rigidity and extracting measurements from onboard cameras only, as explored in some recent works [47, 48]. To achieve that, we need to leave the global localization system, equip the quadcopters with on-board sensors such as depth cameras and develop adequate algorithms to measure the relative information required by the distributed control scheme.

Acknowledgement

The authors would like to acknowledge the financial support of Intel Corporation for the development of this work and to Eng. Antonio Sánchez Medel for his technical support in the realization of the experiments.

Conflict of interest statement

Conflict of interest - none declared.

Appendix A. Formulation as a quadratic programming problem

For general dynamic systems, the optimization problem of Eq. 10 is hard to solve and it is not evident that a solution exists. The equations presented in this Appendix holds for each quadcopter i. In order to rewrite Eq. 10 as a quadratic programming problem, that guarantees the existence of a global minimum, the following vectors are defined

$$\hat{\boldsymbol{Z}}(k+1) \triangleq \begin{bmatrix} \hat{\boldsymbol{z}}(k+1|k) \\ \hat{\boldsymbol{z}}(k+2|k) \\ \vdots \\ \hat{\boldsymbol{z}}(k+H_p|k) \end{bmatrix}, \boldsymbol{R}(k+1) \triangleq \begin{bmatrix} \boldsymbol{r}(k+1|k) \\ \boldsymbol{r}(k+2|k) \\ \vdots \\ \boldsymbol{r}(k+H_p|k) \end{bmatrix}, \boldsymbol{U}(k) \triangleq \begin{bmatrix} \boldsymbol{u}(k|k) \\ \boldsymbol{u}(k+1|k) \\ \vdots \\ \boldsymbol{u}(k+H_u-1|k) \end{bmatrix}, \tag{A.1}$$

where the different numerical values over the horizon are stacked together. Then, having that $H_w = 1$, the objective function of Eq. 7 can be rewritten as

$$\Phi(k) = \left(\hat{\boldsymbol{Z}}(k+1) - \boldsymbol{R}(k+1)\right)^T \boldsymbol{\Lambda} \left(\hat{\boldsymbol{Z}}(k+1) - \boldsymbol{R}(k+1)\right) + \boldsymbol{U}(k)^T \boldsymbol{\Gamma} \boldsymbol{U}(k), \tag{A.2}$$

where $\boldsymbol{U}(k)$ is defined in Eq. 15 and the matrices $\boldsymbol{\Lambda}$ and $\boldsymbol{\Gamma}$ hold weights for the different terms of the objective function,

$$\boldsymbol{\Lambda} \triangleq \begin{bmatrix} \lambda & 0 & \cdots & 0 \\ 0 & \lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda \end{bmatrix}, \quad \boldsymbol{\Gamma} \triangleq \begin{bmatrix} \gamma & 0 & \cdots & 0 \\ 0 & \gamma & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \gamma \end{bmatrix}.$$

We use uniform weights along the horizon window, but the weights could be defined in a different manner (penalizing differently according to the position in the horizon window).

Given the model of the system defined in Eq. 8, it can be shown that the model can be extended along the prediction horizon as

$$\hat{\boldsymbol{Z}}(k+1) = \boldsymbol{P}_X \boldsymbol{z}(k) + \boldsymbol{P}_U \boldsymbol{U}(k), \tag{A.3}$$

where

$$\boldsymbol{P}_X \triangleq \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^{H_p} \end{bmatrix}, \quad \boldsymbol{P}_U \triangleq \begin{bmatrix} \boldsymbol{B} & 0 & \cdots & 0 \\ \boldsymbol{AB} & \boldsymbol{B} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}^{H_u-1}\boldsymbol{B} & \boldsymbol{A}^{H_u-2}\boldsymbol{B} & \cdots & \boldsymbol{B} \\ \boldsymbol{A}^{H_u}\boldsymbol{B} & \boldsymbol{A}^{H_u-1}\boldsymbol{B} & \cdots & \sum_{i=0}^{1} \boldsymbol{A}^i\boldsymbol{B} \\ \vdots & \vdots & & \vdots \\ \boldsymbol{A}^{H_p-1}\boldsymbol{B} & \boldsymbol{A}^{H_p-2}\boldsymbol{B} & \cdots & \sum_{i=0}^{H_p-H_u} \boldsymbol{A}^i\boldsymbol{B} \end{bmatrix}. \tag{A.4}$$

Let us first analyze the case where the reference trajectory is static, i.e., precomputed from the theoretical value of consensus as described in Section 4.1.3. Substituting Eq. A.3 in Eq. A.2, the objective cost function can be rewritten in the following standard form

$$\Phi(k) = \frac{1}{2}\boldsymbol{U}(k)^T\boldsymbol{H}(k)\boldsymbol{U}(k) + \boldsymbol{f}(k)^T\boldsymbol{U}(k) + \boldsymbol{g}(k), \tag{A.5}$$

where

$$\boldsymbol{H}(k) \triangleq 2\big(\boldsymbol{P}_U^T\boldsymbol{\Lambda}\boldsymbol{P}_U + \boldsymbol{\Gamma}\big), \tag{A.6}$$

$$\boldsymbol{f}(k) \triangleq 2\boldsymbol{P}_U^T\boldsymbol{\Lambda}\big(\boldsymbol{P}_X\boldsymbol{z}(k) - \boldsymbol{R}(k+1)\big), \tag{A.7}$$

$$\boldsymbol{g}(k) \triangleq \big(\boldsymbol{P}_X\boldsymbol{z}(k) - \boldsymbol{R}(k+1)\big)^T\boldsymbol{\Lambda}\big(\boldsymbol{P}_X\boldsymbol{z}(k) - \boldsymbol{R}(k+1)\big). \tag{A.8}$$

Now, let us consider the proposed distributed scheme in which the reference is a linear function of the control inputs given by $\boldsymbol{R}(k+1) = \boldsymbol{Q}(k) + \boldsymbol{S}(k)\boldsymbol{U}(k)$, (Eq. 16). In this case, the matrices in the standard formulation are as follows:

$$\boldsymbol{H}(k) \triangleq 2\left[\big(\boldsymbol{P}_U - \boldsymbol{S}(k)\big)^T\boldsymbol{\Lambda}\big(\boldsymbol{P}_U - \boldsymbol{S}(k)\big) + \boldsymbol{\Gamma}\right], \tag{A.9}$$

$$\boldsymbol{f}(k) \triangleq 2\big(\boldsymbol{P}_U - \boldsymbol{S}(k)\big)^T\boldsymbol{\Lambda}\big(\boldsymbol{P}_X\boldsymbol{z}(k) - \boldsymbol{Q}(k)\big), \tag{A.10}$$

$$\boldsymbol{g}(k) \triangleq \big(\boldsymbol{P}_X\boldsymbol{z}(k) - \boldsymbol{Q}(k)\big)^T\boldsymbol{\Lambda}\big(\boldsymbol{P}_X\boldsymbol{z}(k) - \boldsymbol{Q}(k)\big). \tag{A.11}$$

One can obtain the optimal control vector by making $\frac{\partial\Phi(k)}{\partial\boldsymbol{U}(k)} = 0$. The matrix in Eq. A.9 is positive definite and describes the quadratic part of the objective function in Eq. A.5, and the vector in Eq. A.10 describes the linear part. The term in Eq. A.11 is independent of $\boldsymbol{U}$ so it has no influence in the determination of the optimum $\boldsymbol{U}^*$. Then, the optimal control vector is

$$\boldsymbol{U}^* = -\Psi\big(\boldsymbol{P}_X\boldsymbol{z}(k) - \boldsymbol{Q}(k)\big), \tag{A.12}$$

with $\Psi \triangleq \left(\big(\boldsymbol{P}_U - \boldsymbol{S}(k)\big)^T\boldsymbol{\Lambda}\big(\boldsymbol{P}_U - \boldsymbol{S}(k)\big) + \boldsymbol{\Gamma}\right)^{-1}\big(\boldsymbol{P}_U - \boldsymbol{S}(k)\big)^T\boldsymbol{\Lambda}$. The first term of the sequence in Eq. A.12 is actually implemented as control law $\boldsymbol{u}^*(k|k)$ for the quadcopter i.

In Lemma 3.1 of [32], it is proven that the kind of control protocol of Eq. A.12 has an equivalent expression as an individual consensus control law $\boldsymbol{u}_i^*(k|k) = -\psi_i\left(\boldsymbol{z}_i(k) - \boldsymbol{r}_i(k)\right)$, with $\psi_i \in \mathbb{R}^+$ a control gain and $\boldsymbol{r}_i(k) = \frac{1}{|\mathcal{N}_i(k)|+1}\sum_{j\in\mathcal{N}_i(k)\cup\{i\}}\boldsymbol{z}_j(k)$. Using this equivalent control law, in Theorem 3.1 of [32], the stability of the closed-loop system to achieve consensus of the virtual quadcopters (equivalently the formation of the real quadcopters) is guaranteed for communication graphs having a spanning tree.

The cost function in Eq. A.5 is written as a quadratic form in $\boldsymbol{U}$, and this ensures that solving optimization problem of Eq. 10 is possible and that the optimum is unique, because $\boldsymbol{H}(k)$ is positive definite.

There exists a lot of solvers for this type of optimization problems, that are fast and reliable. This is important given the fact that the system we are using, a quadcopter, requires a fast computation of the control inputs.

Appendix A.1. Additional constraints

Model Predictive Control is very flexible when it is necessary to take constraints into account as they just need to be attached to the optimization problem, as depicted in the "Constraints" block of Fig. 1. Some of the most used ones are the ones related to the physical limitations in the control input signals, such as bounds in the values that can be applied to the system. These constraints can be expressed as

$$\boldsymbol{u}_{\min}(k) \leq \boldsymbol{u}(k) \leq \boldsymbol{u}_{\max}(k), \tag{A.13}$$

then the optimization problem in Eq. 10 becomes

$$\begin{aligned} \boldsymbol{u}^* = \quad & \underset{\boldsymbol{u}}{\arg\min}\ \Phi(k), \\ & \text{subject to} \quad \boldsymbol{M}\boldsymbol{u} \leq \boldsymbol{m} \end{aligned} \tag{A.14}$$

where

$$\boldsymbol{M} \triangleq \begin{bmatrix} \boldsymbol{I}_m \\ -\boldsymbol{I}_m \end{bmatrix} \quad \boldsymbol{m} \triangleq \begin{bmatrix} \boldsymbol{u}_{\max}(k) \\ -\boldsymbol{u}_{\min}(k) \end{bmatrix}. \tag{A.15}$$

Extending this to the quadratic form in Eq. A.5, the constraints become

$$\boldsymbol{U}(k) \leq \boldsymbol{U}_{\max}(k) \tag{A.16}$$

$$-\boldsymbol{U}(k) \leq -\boldsymbol{U}_{\min}(k) \tag{A.17}$$

where

$$\boldsymbol{U}_{\min}(k) \triangleq \begin{bmatrix} \boldsymbol{u}_{\min}(k) \\ \boldsymbol{u}_{\min}(k+1) \\ \vdots \\ \boldsymbol{u}_{\min}(k+H_u-1) \end{bmatrix}, \quad \boldsymbol{U}_{\max}(k) \triangleq \begin{bmatrix} \boldsymbol{u}_{\max}(k) \\ \boldsymbol{u}_{\max}(k+1) \\ \vdots \\ \boldsymbol{u}_{\max}(k+H_u-1) \end{bmatrix}.$$

References

[1] K. N. McGuire, Indoor swarm exploration with pocket drones, Ph.D. thesis, Delft University of Technology (11 2019).

[2] T. I. Zohdi, Multiple UAVs for mapping: A review of basic modeling, simulation, and applications, Annual Review of Environment and Resources 43 (1) (2018) 523–543.

[3] R. Tallamraju, E. Price, R. Ludwig, K. Karlapalem, H. H. Bülthoff, M. J. Black, A. Ahmad, Active perception based formation control for multiple aerial vehicles, IEEE Robotics and Automation Letters 4 (4) (2019) 4491–4498.

[4] E. T. Alotaibi, S. S. Alqefari, A. Koubaa, LSAR: multi-UAV collaboration for search and rescue missions, IEEE Access 7 (2019) 55817–55832.

[5] D. K. D. Villa, A. S. Brandao, M. Sarcinelli-Filho, A survey on load transportation using multirotor UAVs, J Intell Robot Syst 98 (2020) 267–296.

[6] O. S. Oubbati, A. Lakas, P. Lorenz, M. Atiquzzaman, A. Jamalipour, Leveraging communicating UAVs for emergency vehicle guidance in urban areas, IEEE Trans. on Emerging Topics in Computing 9 (2) (2019) 1070–1082.

[7] K. K. Oh, M. C. Park, H. S. Ahn, A survey of multi-agent formation control, Automatica 53 (2015) 424 – 440.

[8] C. Rosales, P. Leica, M. Sarcinelli-Filho, G. Scaglia, R. Carelli, 3D formation control of autonomous vehicles based on null-space, J Intell Robot Syst 84 (1-4) (2016) 453–467.

[9] E. L. de Angelis, F. Giulietti, G. Rossetti, Multirotor aircraft formation flight control with collision avoidance capability, Aerospace Science and Technology 77 (2018) 733–741.

[10] A. Joshi, A. Wala, M. Ludhiyani, D. Chakraborty, H. Chung, D. Manjunath, Outdoor cooperative flight using decentralized consensus algorithm and a guaranteed real-time communication protocol, Control Engineering Practice 88 (2019) 128–140.

[11] J. Wang, Z. Zhou, C. Wang, J. Shan, Multiple quadrotors formation flying control design and experimental verification, Unmanned Systems 7 (1) (2019) 47–54.

[12] S. Huang, R. S. H. Teo, K. K. Tan, Collision avoidance of multi unmanned aerial vehicles: A review, Annual Reviews in Control 48 (2019) 147–164.

[13] O. Cetin, G. Yilmaz, Real-time autonomous UAV formation flight with collision and obstacle avoidance in unknown environment, J Intell Robot Syst 84 (1-4) (2016) 415–433.

[14] L. Dai, Q. Cao, Y. Xia, Y. Gao, Distributed MPC for formation of multi-agent systems with collision avoidance and obstacle avoidance, Journal of the Franklin Institute 354 (4) (2017) 2068–2085.

[15] X. Liu, S. S. Ge, C.-H. Goh, Formation potential field for trajectory tracking control of multi-agents in constrained space, Int. Journal of Control 90 (10) (2017) 2137–2151.

[16] E. D'Amato, M. Mattei, I. Notaro, Bi-level flight path planning of UAV formations with collision avoidance, J Intell Robot Syst 93 (2019) 193–211.

[17] J. Hu, M. Wang, C. Zhao, P. Quan, C. D., Formation control and collision avoidance for multi-UAV systems based on voronoi partition, Science China Technol. Sci. 63 (2020) 65–72.

[18] R. Olfati-Saber, J. Fax, R. Murray, Consensus and cooperation in networked multi-agent systems, Proceedings of the IEEE 95 (2007) 215–233.

[19] X. Zhu, Y. Liang, M. Yan, A flexible collision avoidance strategy for the formation of multiple unmanned aerial vehicles, IEEE Access 7 (2019) 140743–140754.

[20] M. A. Trujillo, H. M. Becerra, D. Gomez-Gutierrez, J. Ruiz-Leon, Ramirez-Treviño, Hierarchical task-based formation control and collision avoidance of UAVs in finite time, European Journal of Control 60 (2021) 48–64.

[21] P. D. Christofides, R. Scattolini, D. M. de la Peña, J. Liu, Distributed model predictive control: A tutorial review and future research directions, Computers & Chemical Engineering 51 (2012) 21–41.

[22] M. Saska, D. Hert, T. Báca, V. Krátký, T. P. do Nascimento, Formation control of unmanned micro aerial vehicles for straitened environments, Auton. Robots 44 (6) (2020) 991–1008.

[23] L. Dubois, S. Suzuki, Formation control of multiple quadcopters using model predictive control, Advanced Robotics 32 (19) (2018) 1037–1046.

[24] W. Zhao, T. H. Go, Quadcopter formation flight control combining MPC and robust feedback linearization, Journal of the Franklin Institute 351 (3) (2014) 1335–1355.

[25] T. Chevet, C. Vlad, C. S. Maniu, Z. Y., Decentralized MPC for UAVs formation deployment and reconfiguration with multiple outgoing agents, J Intell Robot Syst 97 (2020) 155–170.

[26] Z. Chao, S. L. Zhou, L. Ming, W. G. Zhang, UAV formation flight based on nonlinear model predictive control, Mathematical Problems in Engineering 2012 (2012).

[27] S. R. Bassolillo, E. D'Amato, I. Notaro, L. Blasi, M. Mattei, Decentralized mesh-based model predictive control for swarms of UAVs, Sensors 20 (15) (2020).

[28] R. Van Parys, G. Pipeleers, Distributed MPC for multi-vehicle systems moving in formation, Robotics and Autonomous Systems 97 (2017) 144–152.

[29] C. W. Chang, J. K. Shiau, Quadrotor formation strategies based on distributed consensus and model predictive controls, Applied Sciences 8 (2018) 2246.

[30] Y. Kuriki, T. Namerikawa, Formation control with collision avoidance for a multi-UAV system using decentralized MPC and consensus-based control, SICE Journal of Control, Measurement, and System Integration 8 (4) (2015) 285–294.

[31] T. Murayama, Distributed model predictive consensus control for robotic swarm system, Artif. Life Robot. 23 (4) (2018) 628–635.

[32] Z. Cheng, M. C. Fan, H. T. Zhang, Distributed MPC based consensus for single-integrator multi-agent systems, ISA Transactions 58 (2015) 112–120.

[33] J. Alonso-Mora, E. Montijano, T. Nägeli, O. Hilliges, M. Schwager, D. Rus, Distributed multi-robot formation control in dynamic environments, Auton. Robots 43 (5) (2019) 1079–1100.

[34] J. Fu, G. Wen, X. Yu, Z. G. Wu, Distributed formation navigation of constrained second-order multiagent systems with collision avoidance and connectivity maintenance, IEEE Trans. on Cybernetics (2020) 1–14.

[35] R. J. M. Afonso, M. R. O. A. Maximo, R. K. H. Galvão, Task allocation and trajectory planning for multiple agents in the presence of obstacle and connectivity constraints with mixed-integer linear programming, Int. Journal of Robust and Nonlinear Control 30 (14) (2020) 5464–5491.

[36] A. Filotheou, A. Nikou, D. V. Dimarogonas, Robust decentralised navigation of multi-agent systems with collision avoidance and connectivity maintenance using model predictive controllers, Int. Journal of Control 93 (6) (2020) 1470–1484.

[37] D. Mellinger, V. Kumar, Minimum snap trajectory generation and control for quadrotors, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 2520–2525.

[38] J. Jin, N. R. Gans, Collision-free formation and heading consensus of nonholonomic robots as a pose regulation problem, Robotics Auton. Syst. 95 (2017) 25–36.

[39] R. Olfati-Saber, R. M. Murray, Consensus problems in networks of agents with switching topology and time-delays, IEEE Transactions on Automatic Control 49 (9) (2004) 1520–1533.

[40] M. N. Zeilinger, M. Morari, C. N. Jones, Soft constrained model predictive control with robust stability guarantees, IEEE Transactions on Automatic Control 59 (5) (2014) 1190–1202.

[41] D. Mayne, J. Rawlings, C. Rao, P. Scokaert, Constrained model predictive control: Stability and optimality, Automatica 36 (6) (2000) 789–814.

[42] S. Zhou, A. Shen, M. Wang, S. Peng, Z. Liu, Study on composing dense formations in a dynamic environment of multirotor UAVs by distributed control, Mathematical Problems in Engineering 2018 (2018).

[43] N. Koenig, A. Howard, Design and use paradigms for Gazebo, an open-source multi-robot simulator, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Vol. 3, 2004, pp. 2149–2154.

[44] F. Furrer, M. Burri, M. Achtelik, R. Siegwart, Robot Operating System (ROS): The Complete Reference (Volume 1), Springer International Publishing, Cham, 2016, Ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625.

[45] M. Monajjemi, bebop_autonomy - ROS driver for Parrot Bebop drone (quadcopter) 1.0 & 2.0, `http://bebop-autonomy.readthedocs.io/`, accessed: 2020-12-05 (2015).

[46] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, S. Boyd, OSQP: an operator splitting solver for quadratic programs, Mathematical Programming Computation 12 (4) (2020) 637–672.

[47] F. Schiano, P. Robuffo Giordano, Bearing rigidity maintenance for formations of quadrotor UAVs, in: IEEE Int. Conf. on Robotics and Automation, 2017, pp. 1467 – 1474.

[48] E. Montijano, E. Cristofalo, D. Zhou, M. Schwager, C. Sagüés, Vision-based distributed formation control without an external positioning system, IEEE Trans. on Robotics 32 (2) (2016) 339–351.