2024 International Conference on Unmanned Aircraft Systems (ICUAS)
June 4-7, 2024 | Chania, Crete, Greece

Trajectory tracking with obstacle avoidance for autonomous UAV swarms based on distributed model predictive control

Despoina Vavelidou, Teo Protoulis, and Alex Alexandridis, *Senior Member, IEEE*

***Abstract*— Multi-agent quadcopter systems, a specialized class of unmanned aerial vehicles (UAVs), have become key players in various industries, showing potential for further growth. However, the inherently nonlinear and complex behavior of quadcopters, coupled with the need for collision and obstacle avoidance within the swarm, while simultaneously achieving collective goals, poses significant challenges for efficient control. Trajectory tracking - a particularly demanding task for swarms - requires a delicate balance between precise tracking and safe navigation. In this work, we address the challenge of trajectory tracking for multi-agent quadcopter configurations through a novel distributed model predictive control (DMPC) framework, with individual local controllers for each agent. For a balanced control scheme, we employ both PID controllers for angle regulation and linear adaptive MPC (LAMPC) controllers for three-dimensional position control. The local MPC schemes ensure safe trajectory tracking, without the need to use predetermined reference trajectories for each agent and predefined formation strategies. Simulation results of the proposed framework demonstrate advanced robustness and dynamic adaptation to unpredicted situations.**

I. Introduction

Unmanned aerial vehicles (UAVs) have received significant interest from both the industry and the academia world, thanks to their unmatched qualities. UAVs are used in a wide range of applications, including communication relay, agriculture, traffic control and surveillance. Among them, quadcopters, a subset of multicopters with four independently controlled rotors, stand out for their compact size, low cost, operational safety, and low energy consumption [1]. These characteristics, along with their relatively simpler mechanical design, make them an ideal vehicle for many different types of missions. However, despite their advantages, their inherently nonlinear and underactuated nature, employing only four inputs for controlling six degrees of freedom (6-DoF), exacerbated by external factors like wind disturbances, renders their control a challenging task [2]. Multiple approaches have been reported for quadcopter control, with proportional-integral-derivative (PID) controllers being the most commonly used for individual drones, due to their simplicity and low computational cost; their limitations, though, in handling physical operational system constraints narrow their applicability domain.

Although, individual quadcopters are capable of carrying out a wide range of missions, there are many tasks that are either challenging, or even impossible to achieve with a single quadcopter. The use of multiple quadcopters that cooperate in a collective manner, forming what is referred to as a quadcopter swarm, is then necessary to provide precision, adaptability, and flexibility when operating in complex environments. UAV and especially quadcopter swarms find application in multiple domains, such as search and rescue operations, precision agriculture or environmental monitoring [3]. However, the efficient operation of multi-agent quadcopter systems further escalates the difficulties in precisely controlling their dynamic flight behavior, as it poses challenges like obstacle and collision avoidance; these challenges are further augmented by the need to keep up with collective goals, such as trajectory tracking or formation maintenance. Extensive research has been conducted for the development of intelligent control strategies capable of addressing these challenges [4]. Among those, the use of model predictive control (MPC) is notably prevalent.

MPC utilizes a model of the controlled system to predict its future dynamic behavior. Based on these predictions, an optimization problem is recursively formulated and solved at each discrete time-step of the closed-loop operation; its solution is the optimal control sequence that drives the system state to the desired reference, while satisfying possible constraints imposed on the system states or/and inputs. Its great advantages, namely its preview capability, its ability to handle constraints, nonlinear dynamics and multiple-input multiple-output (MIMO) systems, and its optimal nature, have established MPC as one of the most commonly deployed control schemes in a variety of applications [5], [6]. Its influence extends not only to single UAV control [7], but also to UAV swarm control, exemplified by [8], where a distributed multiple model MPC (DMMMPC) controller is introduced for target tracking in formation flight; [9], that uses nonlinear MPC (NMPC) for trajectory tracking in unknown complex environments; [10], which proposes an MPC controller with a deterministic chaotic ant colony mobility model for area exploration; and [11], where a centralized control approach using NMPC is proposed for the full-pose manipulation of an object through cables.

Whether employing MPC or alternative strategies, the trajectory tracking task poses additional challenges in controlling multi-agent UAV configurations. There is a need to balance between closely following a desired trajectory, while simultaneously avoiding collisions among the agents, and responding efficiently in the presence of obstacles, without significantly deviating from the desired path. To achieve this, formation strategies are frequently proposed in the literature, often emphasizing on formation control instead of trajectory tracking. In [12], a distributed leader-follower control law is employed for achieving formation control, while

*D. Vavelidou, T. Protoulis, and A. Alexandridis are with the Department of Electrical and Electronic Engineering, University of West Attica, Ancient Olive Grove Campus, Thivon 250 & P. Ralli, 12244, Aigaleo, Greece (email: despoinavavelidou@gmail.com; tprotoulis@uniwa.gr; alexx@uniwa.gr).

in [13], the authors make use of graph theory to preserve the swarm formation. In addition, most works either strictly predetermine the agent trajectories by assigning them separate references [14], [15] or utilize trajectory parameterization techniques [16].

In this work, we introduce a distributed MPC (DMPC) framework for multi-agent quadcopter configurations, specifically tailored for trajectory tracking with collision and obstacle avoidance, where local controllers are integrated within each agent. The control scheme design follows a hybrid approach, employing linear adaptive MPC controllers for position control and PID controllers for angle regulation of each agent in a cascaded design. By harnessing the advantages of both methods, we establish a system that balances optimal performance with computational cost. A key contribution of this work is that each agent reference trajectory is not determined a-priori, but rather a single reference path is provided to all swarm members, and the local MPC schemes are formulated so as to dynamically determine the optimal trajectory to be followed by each agent, while making use of state constraints to ensure obstacle and collision avoidance. The latter property leads to efficient and safe navigation in the case of unpredictable situations, where the obstacle positions are unknown. By combining the aforementioned collective goals into a common control scheme following the distributed approach, we avoid the existence of a single point of failure associated with centralized control frameworks, thus increasing the robustness of the overall control framework.

The rest of this paper is organized as follows: In Section II, the quadcopter dynamics are presented. Section III introduces the proposed control framework, featuring the MPC controllers for position control along with the collision and obstacle avoidance strategies. Section IV presents the case studies, encompassing the experimental setup, as well as simulation results from the control scheme application. Finally, in Section V, concluding remarks and future research plans are stated.

II. QUADCOPTER DYNAMICS

The dynamic behavior of quadcopter vehicles is influenced by numerous parameters related to the aerodynamic and aeroelastic phenomena affecting their flight conditions, while also the propellers rotation further escalates the time-varying nonlinearity of the aforementioned phenomena. To limit this complexity, certain assumptions regarding the vehicle chassis need to be considered. The quadcopter body is assumed to be rigid and symmetrical; the propellers are rigid and the aerodynamic forces affect its vertical and horizontal movement in a proportional fashion to its velocities. Under these assumptions, the vehicle dynamics can be modelled as that of a rigid body with 6-DoF, i.e., 3 of them corresponding to the quadcopter position and 3 to its orientation.

To properly represent the vehicle equations of motion (EoMs), two main reference frames need to be considered. Let *{I}* be the earth fixed inertial frame adopting the North-East-Down (NED) coordinate system, and *{B}* be the body fixed reference frame that coincides with the vehicle center of gravity (CoG) adopting the identical coordinate system as *{I}*. Then, the quadrotor kinematic model is described by the following equations [17]:

$$\mathbf{v}_I = \boldsymbol{R} \cdot \mathbf{v}_B \tag{1a}$$

$$\boldsymbol{\omega}_I = \boldsymbol{T} \cdot \boldsymbol{\omega}_B \tag{1b}$$

where $\mathbf{v}_I = [\dot{x}\ \dot{y}\ \dot{z}]^T \in \mathbb{R}^3$ denote the linear velocities along the three unit axes corresponding to the position vector $[x\ y\ z]^T \in \mathbb{R}^3$ with respect to (w.r.t) frame *{I}*; $\mathbf{v}_B = [u\ v\ w]^T \in \mathbb{R}^3$ denote the linear velocities along the three unit axes w.r.t frame *{B}*; $\boldsymbol{\omega}_I = \left[\dot{\phi}\ \dot{\theta}\ \dot{\psi}\right]^T \in \mathbb{R}^3$ are the angular velocities corresponding to the common roll-pitch-yaw (ϕ, θ, ψ) angles w.r.t frame *{I}* and $\boldsymbol{\omega}_B = [p\ q\ r]^T \in \mathbb{R}^3$ are the angular velocities w.r.t frame *{B}*. The matrices $\boldsymbol{R}, \boldsymbol{T} \in SO(3)$ relate the inertial to body frame coordinates and angular velocities, respectively, and their form is given below:

$$\boldsymbol{R} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{2a}$$

$$\boldsymbol{T} = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix} \tag{2b}$$

where $c_i = \cos(i)$, $s_i = \sin(i)$, $t_i = \tan(i)$, $i = \phi, \theta, \psi$, with the angles in the range of $\psi \in [-\pi\ \pi]$ and $\phi, \theta \in [-\pi/2\ \ \pi/2]$ to comply with physical constraints.

The equations describing the total force $\boldsymbol{F} = \left[f_x\ f_y\ f_z\right] \in \mathbb{R}^3$ and torque $\boldsymbol{T} = \left[\tau_x\ \tau_y\ \tau_z\right] \in \mathbb{R}^3$ being applied to the quadrotor vehicle are derived by applying the Newton-Euler laws of motion and are provided below:

$$m \cdot (\boldsymbol{\omega}_B \times \mathbf{v}_B + \dot{\mathbf{v}}_B) = \boldsymbol{F} \tag{3a}$$

$$\boldsymbol{I} \cdot \dot{\boldsymbol{\omega}}_B + \boldsymbol{\omega}_B \times (\boldsymbol{I} \cdot \boldsymbol{\omega}_B) = \boldsymbol{T} \tag{3b}$$

where m is the quadrotor mass and $\boldsymbol{I} = diag(I_x, I_y, I_z) \in \mathbb{R}^{(3\times3)}$ is the diagonal inertia matrix. By considering the state vector $[x\ y\ z\ \phi\ \theta\ \psi\ u\ v\ w\ p\ q\ r] \in \mathbb{R}^{12}$, a 12-state dynamic model is derived from equations (1a-b), (3a-b). This model may be capable of precisely simulating the dynamic behavior of quadcopter vehicles, its complexity, though, renders it difficult to be used in designing appropriate automatic control frameworks. To remedy this situation, one can make the assumption $\left[\dot{\phi}\ \dot{\theta}\ \dot{\psi}\right]^T = [p\ q\ r]^T$ that holds for small angles [18] and derive a simpler quadrotor model w.r.t frame *{I}* that can be efficiently handled to develop control schemes. The following expression regarding the force acting on the quadcopter vehicle can be derived by properly applying the Newton's law:

$$m \cdot \dot{\mathbf{v}}_I = \boldsymbol{R} \cdot \boldsymbol{F} = mg \cdot \hat{\boldsymbol{z}}_I - f_t \cdot \boldsymbol{R} \cdot \hat{\boldsymbol{z}}_B \tag{4}$$

where g is the gravitational acceleration; $\hat{\boldsymbol{z}}_I$ is the unit vector expressed in the *{I}* frame z-axis; $\hat{\boldsymbol{z}}_B$ is the corresponding vector expressed in frame *{B}* and f_t is the total thrust produced by the 4 rotors. Then, by considering the small angles assumption, the following mathematical model describing the quadcopter motion in 3D-space is derived:

$$m \cdot \ddot{x} = -f_t \cdot \left(c_\phi c_\psi s_\theta + s_\phi s_\psi\right) \tag{5a}$$

$$m \cdot \ddot{y} = -f_t \cdot \left(c_\phi s_\psi s_\theta - c_\psi s_\phi\right) \tag{5b}$$

$$m \cdot \ddot{z} = m \cdot g - f_t \cdot c_\phi c_\theta \tag{5c}$$

$$I_x \cdot \ddot{\phi} = \left(I_y - I_z\right) \cdot \dot{\psi} \cdot \dot{\theta} + \tau_x \tag{5d}$$

$$I_y \cdot \ddot{\theta} = (I_z - I_x) \cdot \dot{\psi} \cdot \dot{\phi} + \tau_y \quad (5e)$$

$$I_z \cdot \ddot{\psi} = (I_x - I_y) \cdot \dot{\phi} \cdot \dot{\theta} + \tau_z \quad (5f)$$

where equations (5a-c) describe the position dynamics w.r.t frame $\{I\}$ and equations (5d-f) represent the orientation dynamics w.r.t frame $\{I\}$. In the rest of this paper, the dynamic model corresponding to equations (1a-b) and (3a-b) is considered to represent the "real" quadcopter vehicle, while equations (5a-f) are employed to form the predictive model used in the development of the overall control framework.

III. Proposed Control Framework

A. Altitude controller

The altitude controller objective is to determine the required thrust (f_t) needed to be produced by the four rotors of the vehicle so that the quadcopter successfully tracks a desired trajectory w.r.t frame $\{I\}$ z-axis, regardless of its initial position. In this work, to successfully accomplish the aforementioned objective, a linear adaptive model predictive controller (LAMPC) is designed that makes use of a simple affine in the control discrete-time predictive model that has been derived by discretizing equation (5c).

Equation (5c) relates the vehicle vertical acceleration in terms of frame $\{I\}$ with the total thrust (f_t) produced. By considering the z-direction state vector as $\boldsymbol{x}_z = [z\ \dot{z}]^T = \begin{bmatrix} x_{1,z} & x_{2,z} \end{bmatrix}^T$, the state-space formulation of equation (5c) is:

$$\dot{x}_{1,z} = x_{2,z} \quad (6a)$$

$$\dot{x}_{2,z} = g - \frac{f_t}{m} \cdot c_\phi c_\theta \quad (6b)$$

To derive the corresponding discrete-time representation, the forward Euler discretization method is applied and the resulting model is provided below:

$$\boldsymbol{x}_{z,k+1} = \boldsymbol{A} \cdot \boldsymbol{x}_{z,k} + \boldsymbol{B}_k \cdot f_{t,k} + \boldsymbol{G} = f_z(\boldsymbol{x}_{z,k}, f_{t,k}) \quad (7)$$

where k corresponds to the current discrete time instant; $\boldsymbol{x}_{z,k} \in \mathbb{R}^{\boldsymbol{2}}$ is the state vector containing the quadcopter altitude and its respective linear velocity at sampling instant k; $\boldsymbol{A} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}$ is the constant state transition matrix; $\boldsymbol{B}_k = \begin{bmatrix} 0 & -(T_s/m) \cdot (c_{\phi_k} \cdot c_{\theta_k}) \end{bmatrix}^T$ is the time-varying control influence vector; $\boldsymbol{G} = [0\ \ T_s \cdot g]^T$ is the gravitational vector and $f_{t,k}$ is the total thrust applied to the quadrotor at time instant k. Note that T_s is the controller sampling time. The control influence vector is adaptive in the sense that its elements are updated with the current angle measurements at each discrete timestep of the closed-loop operation. The discrete-time affine in the control model (7) is utilized as the altitude LAMPC controller predictive model.

Based on the integrated predictive model predictions, the LAMPC scheme formulates and solves at each discrete step an optimization problem, referred to as the open-loop Optimal Control Problem (OCP). For the setup considered in this paper, the OCP for the altitude controller at sampling instant k with initial state $\boldsymbol{x}_z(k) \in \mathbb{R}^{\boldsymbol{2}}$ is constructed as follows:

$$\min_{\boldsymbol{\Delta f}_t(\cdot)} J\big(\boldsymbol{x}_z(k), \boldsymbol{f}_t(\cdot)\big) \quad (8)$$

with

$$J\big(\boldsymbol{x}_z(k), \boldsymbol{f}_t(\cdot)\big) = \boldsymbol{e}_z^T \cdot \boldsymbol{Q}_1 \cdot \boldsymbol{e}_z + \dot{\boldsymbol{e}}_z^T \cdot \boldsymbol{Q}_2 \cdot \dot{\boldsymbol{e}}_z + \boldsymbol{\Delta f}_t(\cdot)^{\boldsymbol{T}} \cdot \boldsymbol{R} \cdot \boldsymbol{\Delta f}_t(\cdot) \quad (9)$$

subject to: $\forall i \in [0\ 1 \cdots\ H_{p_z} - 1]$

$$\hat{\boldsymbol{x}}_z(k+i+1) = f_z\big(\hat{\boldsymbol{x}}_z(k+i), \boldsymbol{f}_t(k+i)\big), \quad (10a)$$

$$\hat{\boldsymbol{x}}_z(k) = \boldsymbol{x}_z(k) \quad (10b)$$

$$f_t(k+i) = \Delta f_t(k) + f_t(k+i-1), \quad (10c)$$

$$\Delta f_t(k+j) = 0, j = H_{c_z} + 1, \dots, H_{p_z} \quad (10d)$$

$$\Delta f_t(k+i) \in \boldsymbol{\mathcal{U}}_1 \quad (10e)$$

$$f_t(k+i) \in \boldsymbol{\mathcal{U}}_2 \quad (10f)$$

$$e_z\big(H_{p_z}\big) = 0 \quad (10g)$$

$$\dot{e}_z\big(H_{p_z}\big) = 0 \quad (10h)$$

where H_{p_z}, H_{c_z} denote the prediction and control horizons for z-axis, respectively; $\hat{\boldsymbol{x}}_z(k) = \big[\hat{x}_{z,1}(k)\ \hat{x}_{z,2}(k)\big]^T$ is the prediction vector including the altitude and its corresponding velocity at time instant k, respectively; $\boldsymbol{Q}_1, \boldsymbol{Q}_2 \in \mathbb{R}^{(H_{p_z} \times H_{p_z})}$ and $\boldsymbol{R} \in \mathbb{R}^{(H_{c_z} \times H_{c_z})}$ are positive definite weighting matrices; $\boldsymbol{e}_z, \dot{\boldsymbol{e}}_z \in \mathbb{R}^{(H_{p_z} \times H_{p_z})}$ are the errors corresponding to the deviation of the predicted altitude and its respective velocity from the desired setpoints, defined as $e_z = \hat{x}_{z,1} - x_{z,1d}$, $\dot{e}_z = \hat{x}_{z,2} - x_{z,2d}$; $\boldsymbol{\Delta f}_t(\cdot) \in \mathbb{R}^{(H_{p_z} \times 1)}$ is the vector representing the manipulated variable, i.e., the total thrust ($\boldsymbol{f}_t$) deviation between two consecutive time-steps for the entire prediction horizon; $f_t(k+i)$ represents the manipulated variable value at prediction time-step $k+i$; $\Delta f_t(k+i)$ denotes the manipulated input deviation at time instant $k+i$ and $\boldsymbol{\mathcal{U}}_1, \boldsymbol{\mathcal{U}}_2$ are feasible convex sets imposing constraints on the manipulated variable values and its corresponding deviations. Equations (10g) and (10h) are terminal constraints included to enforce zero tracking error for the model predictions at the end of the prediction horizon.

The optimization problem design variable is chosen to be the deviation in total thrust between consecutive timesteps, so as to prevent sudden variations regarding the rotor speeds, and thus mitigating the risk of excessive wear on the motors. So, by optimizing the thrust variation, the MPC controller also accounts for preventing physical damage on the quadrotor mechanical parts.

B. X-Y Plane controller

Due to the inherent dynamics of the quadcopter system, x-axis position control is intricately linked to the pitch angle (θ), while y-axis position control is achieved by manipulating the roll angle (φ). This is also evident by examining equations (5a) and (5b). Specifically, manipulating the pitch angle influences the forward or backward tilt of the quadcopter, leading to adjustments in its position along the x-axis. Conversely, adjustments in the roll angle induce lateral tilting, affecting the quadcopter position along the y-axis. That interdependence means that we cannot directly control the x-y coordinates without accounting for the aforementioned angles and creating a direct relationship between them by introducing two virtual control variables. The latter are derived from equations (5a) and (5b) as follows:

$$u_x = c_\phi c_\psi s_\theta + s_\phi s_\psi \tag{11a}$$

$$u_y = c_\phi s_\psi s_\theta - c_\psi s_\phi \tag{11b}$$

The x-y plane controller aims to determine the optimal reference roll (φ_k) and pitch (θ_k) angles for each time instant k, which subsequently serve as references for the inner-loop attitude control discussed in Sec. III-C. Solving equations (11a), (11b) for roll and pitch assuming that the heading angle ψ_k is near 0, yields the expressions for the angle setpoints at time instant k:

$$\theta_{d_k} = \sin^{-1}\left(\frac{u_{x_k}}{c_{\varphi_k}}\right), c_{\varphi_k} \neq 0 \tag{12}$$

$$\varphi_{d_k} = \sin^{-1}(u_{y_k}) \tag{13}$$

where ϕ_k, θ_k must confine to the interval $(-\pi/2 \;\; \pi/2)$ for physical feasibility. This constraint ensures practical and safe operation by preventing the occurrence of extreme angles that could compromise performance or induce instability.

In order to achieve trajectory tracking for the x-y plane w.r.t frame *{I}* x-axis and *{I}* y-axis, we employ an LAMPC designed for both x and y directions. The term adaptive refers to the fact that the control vector is updated, as shown below, at each time instant according to the determined thrust value originating from the z-axis MPC. To obtain the $x-y$ prediction models, we follow the same methodology described in Section III-A. So, by discretizing equations (5a) and (5b) the resulting models are provided below:

$$\boldsymbol{x}_{x,k+1} = \boldsymbol{A}\cdot\boldsymbol{x}_{x,k} + \boldsymbol{B}_k\cdot u_{x,k} = f_x(\boldsymbol{x}_{x,k}, u_{x,k}) \tag{14}$$

$$\boldsymbol{x}_{y,k+1} = \boldsymbol{A}\cdot\boldsymbol{x}_{y,k} + \boldsymbol{B}_k\cdot u_{y,k} = f_y(\boldsymbol{x}_{y,k}, u_{y,k}) \tag{15}$$

where $\boldsymbol{x}_{x,k}, \boldsymbol{x}_{y,k} \in \mathbb{R}^{\boldsymbol{2}}$are the state vectors containing the quadcopter x, y position predictions along with their respective linear velocities at sampling instant k; u_{x_k}, u_{y_k} are the defined virtual control variables at time instant k and $\boldsymbol{B}_k = \begin{bmatrix}0 & -T_s\cdot f_{t,k}/m\end{bmatrix}^T$ is the time-varying control input vector. As it is evident from equations (14) and (15), the discretized prediction models are affine in the control as in the case of the corresponding z-axis prediction model.

The x-y LAMPC controller formulates and solves at each discrete time instant an OCP, which at sampling instant k with initial states $\boldsymbol{x}_x(k), \boldsymbol{x}_y(k) \in \mathbb{R}^{\boldsymbol{2}}$, is constructed as follows:

$$\min_{\boldsymbol{\Delta u}_x(\cdot), \boldsymbol{\Delta u}_y(\cdot)} J\left(\boldsymbol{x}_x(k), u_x(\cdot), \boldsymbol{x}_y(k), u_y(\cdot)\right) \tag{16}$$

with

$$\begin{aligned} &J\left(\boldsymbol{x}_x(k), u_x(\cdot), \boldsymbol{x}_y(k), u_y(\cdot)\right) \\ &= \boldsymbol{e}_x^T\cdot\boldsymbol{Q}_{1,x}\cdot\boldsymbol{e}_x + \dot{\boldsymbol{e}}_x^T\cdot\boldsymbol{Q}_{2,x}\cdot\dot{\boldsymbol{e}}_x + \boldsymbol{\Delta u}_x(\cdot)^{\boldsymbol{T}} \\ &\cdot\boldsymbol{R}_x\cdot\boldsymbol{\Delta u}_x(\cdot) + \boldsymbol{e}_y^T\cdot\boldsymbol{Q}_{1,y}\cdot\boldsymbol{e}_y + \dot{\boldsymbol{e}}_y^T\cdot\boldsymbol{Q}_{2,y}\cdot\dot{\boldsymbol{e}}_y \\ &+ \boldsymbol{\Delta u}_y(\cdot)^{\boldsymbol{T}}\cdot\boldsymbol{R}_y\cdot\boldsymbol{\Delta u}_y(\cdot) \end{aligned} \tag{17}$$

subject to: $\forall i \in \begin{bmatrix}0\; 1 \cdots H_{p_{x,y}} - 1\end{bmatrix}$

$$\hat{\boldsymbol{x}}_x(k+i+1) = f_x(\hat{\boldsymbol{x}}_x(k+i), u_x(k+i)), \tag{18a}$$

$$\hat{\boldsymbol{x}}_y(k+i+1) = f_y\left(\hat{\boldsymbol{x}}_y(k+i), u_y(k+i)\right), \tag{18b}$$

$$\hat{\boldsymbol{x}}_x(k) = \boldsymbol{x}_x(k) \tag{18c}$$

$$\hat{\boldsymbol{x}}_y(k) = \boldsymbol{x}_y(k) \tag{18d}$$

$$u_x(k+i) = \Delta u_x(k) + u_x(k+i-1), \tag{18e}$$

$$u_y(k+i) = \Delta u_y(k) + u_y(k+i-1), \tag{18f}$$

$$\Delta u_x(k+j) = 0, j = H_{c_x}+1, \dots, H_{p_x} \tag{18g}$$

$$\Delta u_y(k+j) = 0, j = H_{c_y}+1, \dots, H_{p_y} \tag{18h}$$

$$\Delta u_x(k+i) \in \boldsymbol{\mathcal{U}}_{1,x} \tag{18i}$$

$$\Delta u_y(k+i) \in \boldsymbol{\mathcal{U}}_{1,y} \tag{18j}$$

$$u_x(k+i) \in \boldsymbol{\mathcal{U}}_{2,x} \tag{18k}$$

$$u_y(k+i) \in \boldsymbol{\mathcal{U}}_{2,y} \tag{18l}$$

where H_{c_x}, H_{c_y} and H_{p_x}, H_{p_y} are the control and prediction horizons for x-axis and y-axis, respectively, $\hat{\boldsymbol{x}}_x(k) = \begin{bmatrix}\hat{x}_{x,1}(k)\; \hat{x}_{x,2}(k)\end{bmatrix}^T$is the prediction vector for x-control, including the position x and its corresponding velocity at time instant k, respectively; $\hat{\boldsymbol{x}}_y(k) = \begin{bmatrix}\hat{x}_{y,1}(k)\; \hat{x}_{y,2}(k)\end{bmatrix}^T$is the parallel prediction vector for y-control; $\boldsymbol{Q}_{1,x}, \boldsymbol{Q}_{2,x} \in \mathbb{R}^{(H_{p_x}\times H_{p_x})}$, $\boldsymbol{Q}_{1,y}, \boldsymbol{Q}_{2,y} \in \mathbb{R}^{\left(H_{p_y}\times H_{p_y}\right)}$ and $\boldsymbol{R}_x \in \mathbb{R}^{(H_{c_x}\times H_{c_x})}$, $\boldsymbol{R}_y \in \mathbb{R}^{\left(H_{c_y}\times H_{c_y}\right)}$ are positive definite weighting matrices; $\boldsymbol{e}_x, \dot{\boldsymbol{e}}_x \in \mathbb{R}^{(H_{p_x}\times H_{p_x})}$ $\boldsymbol{e}_y, \dot{\boldsymbol{e}}_y \in \mathbb{R}^{\left(H_{p_y}\times H_{p_y}\right)}$ are the errors corresponding to the deviation of the predicted positions x and y and their respective velocities from the desired setpoints, defined as $e_x = \hat{x}_{x,1} - x_{x,1d}$, $\dot{e}_x = \hat{x}_{x,2} - x_{x,2d}$ and $e_y = \hat{x}_{y,1} - x_{y,1d}$, $\dot{e}_y = \hat{x}_{y,2} - x_{y,2d}$ for x and y axes, respectively; $\boldsymbol{\Delta u}_x(\cdot) \in \mathbb{R}^{(H_{p_x}\times 1)}$, $\boldsymbol{\Delta u}_y(\cdot) \in \mathbb{R}^{\left(H_{p_y}\times 1\right)}$ are the vectors representing the virtual control variables, i.e., the virtual controls (u_x, u_y) deviation trajectory between two consecutive time-steps for the entire prediction horizon; $u_x(k+i)$, $u_y(k+i)$ represent the virtual control variable values at prediction time-step $k+i$; $\Delta u_x(k+i), \Delta u_y(k+i)$ denote the virtual control input deviation at time instant $k+i$ and $\boldsymbol{\mathcal{U}}_{1,x}, \boldsymbol{\mathcal{U}}_{2,x}, \boldsymbol{\mathcal{U}}_{1,y}, \boldsymbol{\mathcal{U}}_{2,y}$ are feasible convex sets imposing constraints on the virtual control variables values and their corresponding deviations for x and y axes, respectively. In contrast to the OCP formulation for altitude control (equations (8) to (10h)), the x-y plane OCP does not include a terminal constraint, as this would violate the need for collision avoidance among the agents of the swarm.

C. Attitude controller

The inner-loop controller of each vehicle is responsible to stabilize the orientation dynamics, i.e., the quadcopter roll, pitch and yaw angles, by determining the torque values τ_x, τ_y and τ_z, respectively. Stabilization of the quadrotor rotational dynamics is a well-studied subject and, in this work, we utilize three simple PID controllers, each one responsible for regulating one of the aforementioned angles. The total torque vector is computed based on the following expression:

$$\boldsymbol{T} = \boldsymbol{K}_P\cdot\boldsymbol{E} + \boldsymbol{K}_I\cdot\int_0^t \boldsymbol{E}(\boldsymbol{\tau})d\tau + \boldsymbol{K}_D\cdot\dot{\boldsymbol{E}} \tag{19}$$

where $\boldsymbol{K}_P, \boldsymbol{K}_I, \boldsymbol{K}_D \in \mathbb{R}^{(\boldsymbol{3}\times\boldsymbol{3})}$ are diagonal positive definite matrices denoting the proportional, integral and derivative gains, respectively. The vector $\boldsymbol{E} = \begin{bmatrix}e_\phi\; e_\theta\; e_\psi\end{bmatrix}^T$corresponds to the roll, pitch and yaw angle errors that are defined as: $e_\phi = \phi_d - \phi, e_\theta = \theta_d - \theta$ and $e_\psi = \psi_d - \psi$, where ϕ_d, θ_d and

ψ_d are the reference angle values. The setpoints for the roll and pitch angles are computed using equations (12), (13), while the setpoint for the yaw angle is provided externally.

Despite their simplicity, PID controllers have been widely used in stabilizing the rotational dynamics of quadcopter vehicles since they provide some important practical benefits. Their simple structure leads to control schemes of low computational complexity, suitable for real-time implementation, while the gain tuning procedure is relatively easy, especially compared to alternative and more complex controllers. Moreover, PID controllers inherently exhibit robustness properties, rendering them capable of satisfactory dealing with disturbances affecting the dynamic behaviour of the quadrotor vehicle during flight conditions [19]. Finally, it should be noted that in order to maintain compliance with the discrete-time nature of the developed MPC controllers, equation (19) has been discretized by utilizing the simple forward Euler discretization method. The proposed control scheme for each quadcopter, is depicted in Fig. 1.

D. *Collision and obstacle avoidance constraints*

Collision and obstacle avoidance constraints are crucial for swarm safety, preventing physical contact among agents, as well as with external obstacles. To formulate these constraints, we adopt a strategy based on collaborative inter-agent sharing of the positions predicted during the previous time-step. To be more specific, at each time-step the process begins with the execution of the z-axis LAMPC controller, as presented in section III-A. Subsequently, the resulting control input f_t is transmitted to the x-y plane LAMPC controller, explicitly for the calculation of the corresponding model predictions (14), (15). Concurrently, the predictions for x, y and z coordinates from the other agents at the previous timestep $(k-1)$ are communicated to the x-y plane LAMPC controller of the focal agent. This shared information, combined with the x-y-z predictions of the focal agent at that particular moment in discrete time k, with the z component information originating from the concluded z-axis LAMPC, and the x-y components arising from the ongoing execution of the x-y plane LAMPC, enable collective decision-making.

In order to avoid collisions between agents, a new constraint is incorporated into the OCP formulation of the x-y plane controller. This constraint ensures that the Euclidean distance between the local agent and the other swarm units, calculated using position estimations obtained from their predictive models, exceeds twice the agent radius (i.e., the agent diameter), through the prediction horizon, as follows:

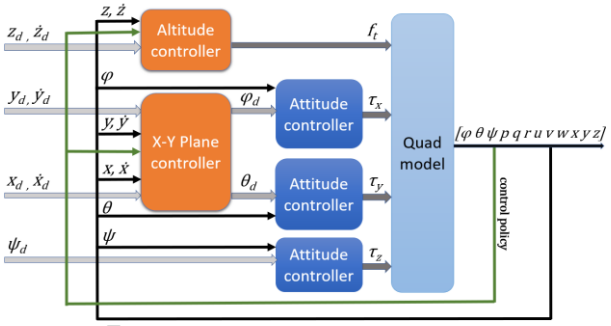

Fig. 1. Trajectory tracking control scheme for each quadcopter

$$\forall h \in \left\{1,2,\dots,H_{p_{x,y}}-1\right\}, \forall i \in \{1,2,\dots,n_{ag}\}, \\ \forall j \in \{1,2,\dots,n_{ag}\} \setminus \{i\}, \left\|d_{ag(i,j,h)}\right\| > 2 \cdot r_{ag} \quad (20)$$

where n_{ag} is the number of agents within the swarm; i represents the focal agent; j represents all the other agents; $d_{ag(i,j,h)}$ is the Euclidean distance between the centers of agent i and agent j for every step h, and r_{ag} is the agent radius, assuming that all quadcopters are homogenous and thus have the same physical characteristics. Assuming that the agents communicate between them their position (x,y,z) predictions, calculation of distance $d_{ag(i,j,h)}$ involves the position estimations of the predictive model used by agent i at the current timestep k, while for each of the other agents j, the corresponding position predictions obtained at the previous timestep $(k-1)$ are employed, as the current predictions are not yet available. The Euclidean distances are calculated for all the timesteps of the prediction horizon save the last one, as the corresponding prediction is not yet available to the focal agent from the rest of the swarm.

Apart from the collision avoidance constraint, an obstacle avoidance constraint is also introduced in the OCP formulation of the x-y plane controller, which ensures that for every timestep within the prediction horizon, the Euclidean distance between each agent of the swarm and the center of the obstacle remains greater than the summation of their respective radii. In this paper, we consider spherical obstacles of a known radius r_{obs} along the reference path. Then, the constraint for each timestep k can be expressed as follows:

$$\forall i \in \{1,2,\dots,n_{ag}\}, \forall h \in \left\{1,2,\dots,H_{p_{x,y}}\right\} \\ \left\|c_{obs(i,k)}\right\| < d_{thres} \Rightarrow \\ \left\|d_{obs(i,h)}\right\| > (r_{ag} + r_{obs}) \quad (21)$$

where $c_{obs(i,k)}$ is the distance between the current position of the agent i and the center of the obstacle at the current timestep k; $d_{obs(i,h)}$ is the Euclidean distance between each agent model predictions and the center of the obstacle, calculated for every step h of the prediction horizon; r_{obs} is the obstacle radius and d_{thres} is the threshold value for which the obstacle avoidance constraint is activated. This activation condition mirrors real-world scenarios where the agents become aware of the obstacle only within close range to it, through appropriate sensors capable of detecting obstacles.

IV. Case Studies

A. *Experimental setup*

Three identical quadcopters were simulated for the swarm $(n_{ag}=3)$, with their parameters presented in Table I. The experiments, which were conducted in MATLAB, aimed to evaluate the performance and efficacy of the proposed DMPC framework in achieving optimal obstacle and collision avoidance, while striving to minimize trajectory tracking errors to the greatest extent possible, given the shared reference trajectory for all swarm units. All simulations were conducted using the ode45 solver. It is noted that, in all cases, the reference yaw angle ψ_d was set to zero. Finally, the optimization problems were solved using a sequential quadratic programming (SQP) solver [20].

All tuning parameters were selected through a process of trial and error, in conjunction with literature suggestions. The tuning parameters for the LAMPC horizons are presented in Table II. Control horizons, influencing the OCP dimensions and thus the computational cost, were chosen relatively small, while the prediction horizon, equal for all dimensions for simplicity, was set sufficiently large to capture the system dynamics. Moreover, when tuning the horizons, we had to consider the actual time for actions (in $[sec]$); this is calculated as $H_c \cdot T_s$ for control actions, and $H_p \cdot T_s$ for predictions. Table III shows the weighting matrices, with the matrix $\boldsymbol{I}_m$ corresponding to the identity matrix of dimensions $(m \times m)$; Q_1 was chosen especially short, allowing flexibility for swarm agents in collision-free trajectory tracking. To accommodate the quadcopter high sensitivity and tendency for abrupt changes, narrow bound constraints were applied to controlled variables and the manipulated variables f_t, u_x and u_y, as specified in Table IV, while, for simplification, these were constrained symmetrically. Particular emphasis was placed upon the tuning for u_x and u_y, due to their direct association with the quadcopter angles, as evident in (11a), (11b). Their maximum values were meticulously chosen to produce small angles, preventing violations of the system physical constraints. This choice is considered adequate for the swarm scenario examined, involving a total of three agents. Finally, not much effort was placed upon PID gain tuning since extensive adjustments did not yield significant improvements. The same tuning parameters were used for all agents, since the quadcopter units were considered to be geometrically identical; the same tuning parameters were also used for both cases of obstacle and collision avoidance.

TABLE II. CONTROL AND PREDICTION HORIZONS

Description	Symbol	Value
Altitude control Control Horizon	H_{c_z}	3
X-Y Plane control Control Horizon	$H_{c_{x,y}}$	6
Altitude control Prediction Horizon	H_{p_z}	40
X-Y Plane control Prediction Horizon	$H_{p_{x,y}}$	40

TABLE III. WEIGHTING MATRICES

	$\boldsymbol{Q_1}$	$\boldsymbol{Q_2}$	$\boldsymbol{R}$
Altitude control	$2 \cdot \boldsymbol{I}_{H_{p_z}}$	$0.05 \cdot \boldsymbol{I}_{H_{p_z}}$	$0.01 \cdot \boldsymbol{I}_{H_{c_z}}$
X-Y Plane control	$1.5 \cdot \boldsymbol{I}_{H_{p_{x,y}}}$	$0.5 \cdot \boldsymbol{I}_{H_{p_{x,y}}}$	$0.01 \cdot \boldsymbol{I}_{H_{c_{x,y}}}$

TABLE IV. CONTROLLED AND MANIPULATED VARIABLES BOUND CONSTRAINTS

Variables	Constraint range
$\Delta f_t\ [N]$	$[-5\ 5]$
$\Delta u_x, \Delta u_y$	$[-0.05\ 0.05]$
$f_t\ [N]$	$[-20\ 20]$
u_x, u_y	$[-0.2\ 0.2]$

B. *Case 1: Collision avoidance*

In the first case, we tested how the three-agent quadcopter configuration achieves trajectory tracking while avoiding collisions among the agents, using the proposed DMPC framework. The starting locations of the quadcopters were $(-1,0,0)$, $(0,0,0)$ and $(1,0,0)[m]$ for the first, the second and the third agent, respectively, i.e., adjacent positions in the x-axis, in order to establish a safe starting point. The common reference trajectory was randomly selected as a spiral denoted by $x_d = 5\cos(0.5t)$, $y_d = 5\sin(0.5t)$ and $z_d = -1 - 0.1t$.

The trajectory tracking performance for the multi-agent quadcopter configuration over three periods (approximately 38 [sec] of simulation time) is illustrated in Fig. 2 as a 3D plot, while the separate trajectories in the (x, y, z) axes are presented in Fig. 3. Since all agents follow a common reference trajectory, deviations from the reference occur in the x-y plane, due to the imperative need to avoid collisions, as dictated by the collision avoidance constraint (20). In contrast, agents are permitted to share the same altitude, as the z-axis LAMPC lacks additional constraints. Moreover, we observe that the agents quickly converge towards a region around the reference trajectory from their initial positions, which underscores their agility in dynamically alignment.

Overall, the aforementioned figures illustrate the swarm success in the challenging trajectory tracking task, while avoiding collisions among the agents. This accomplishment is attributed to the intelligent adaptation facilitated by the proposed framework, which includes an effective communication strategy, a reliable integrated collision

TABLE I. CONSTANT PHYHSICAL QUADCOPTER PARAMETERS

Parameter	Value
Mass (m)	1 kg
Arm length (l)	0.24 m
Safe radius (r_{ag})	0.3 m
Gravitational acceleration (g)	9.81 m/s^2
Inertia moment about x-axis (I_x)	8e^{-3} kg m^2
Inertia moment about y-axis (I_y)	8e^{-3} kg m^2
Inertia moment about z-axis (I_z)	14.2e^{-3} kg m^2

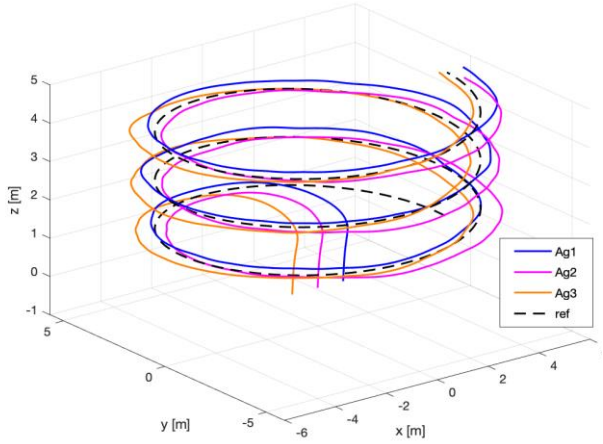

Fig. 2. Three-agent swarm 3D plot for Case 1

TABLE V. PERFORMANCE METRICS

Case	MSE for x-axis averaged for n_{ag} $[m^2]$	MSE for y-axis averaged for n_{ag} $[m^2]$	MSE for z-axis averaged for n_{ag} $[m^2]$	Mean Euclidean distance for agent 1 $[m]$	Mean Euclidean distance for agent 2 $[m]$	Mean Euclidean distance for agent 3 $[m]$
1	0.884	0.233	0.010	0.736	0.739	0.748
2	0.576	0.353	0.238	1.189	0.711	0.821

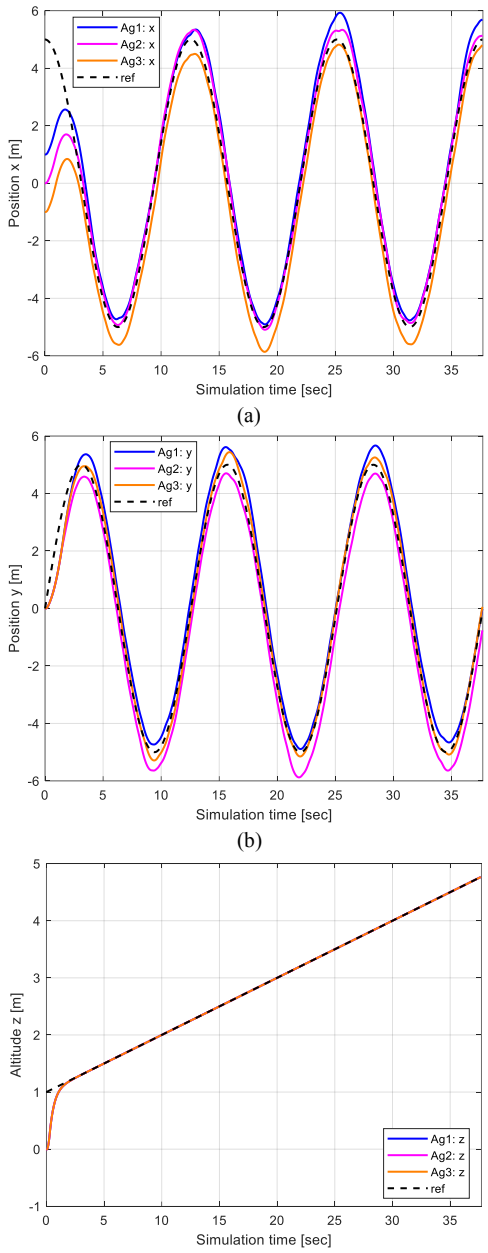

Fig. 3. Three-agent swarm graphs for trajectory tracking in (a) x-axis, (b) y-axis, (c) z-axis for Case 1

avoidance constraint and an absence of predetermined trajectories for each swarm unit or a specific formation strategy. The swarm performance is further supported by numerical indices calculated and presented in Table V, depicting the mean squared error (MSE) per axis values averaged for all agents, and the mean Euclidean distance per agent. The relatively low values achieved for these indices, signify success in trajectory tracking for the x, y, and z axes, as well as for the overall trajectory.

C. Case 2: Obstacle and collision avoidance

In the second case, we tested how the three-agent quadcopter configuration achieves trajectory tracking, while avoiding collisions with an obstacle appearing suddenly and among the agents. The starting locations of the quadcopters were $(2,2,0)$, $(2,3,0)$ and $(2,1,0)[m]$ for the first, the second and the third agent, respectively, i.e., adjacent positions in the y-axis. We tested the configuration for the linear reference trajectory denoted by $x_d = 2 + 1.5t$, $y_d = 2 + 0.7t$ and $z_d = -2 - 0.1t$ while the center of the obstacle was randomly positioned at $(9.5, 5.5, -2.3)[m]$. We assumed that the obstacle radius is equal to $r_{obs} = 1\ [m]$ and that the swarm is able to detect the obstacle at distance $d_{thres} = 3\ [m]$. The quantitative performance of the proposed framework for Case 2 is summarized in Table V, confirming the effectiveness of the proposed approach.

The 2D plot depicting the x-y plane trajectory tracking in presence of a sudden obstacle, featuring the obstacle as a cyan circle, is shown in Fig. 4, whereas the corresponding 3D plot is presented in Fig. 5, for a total of 500 timesteps k (or 10 [sec] of simulation time). Moreover, Fig. 6 depicts the z-axis trajectory tracking performance. In Figs. 4 and 5, the obstacle avoidance strategy is evident, as the quadcopters surpass the obstacle on either side, while trying to closely follow the desired reference. Specifically, the proposed framework allows for individualized dynamic decision-making among swarm units, which results in them not uniformly choosing the same side to bypass the obstacle. It should also be noted that, though agent trajectories may seem to intersect, actual collisions do not occur; the agents traverse the same spatial coordinates but at different time instances, so physical contact is successfully prevented. Finally, examining Fig. 6, it becomes apparent that the altitude z of the agents is effectively maintained at a commendable level, a result again attributed to the developed MPC controller.

In total, the aforementioned figures highlight the proposed framework ability to perform obstacle avoidance, free from predetermined reference trajectories and formation strategies. The swarm achieves trajectory tracking, while effectively avoiding suddenly appearing obstacles and ensuring collision avoidance among the agents. Overall, the control scheme demonstrates consistent performance across different scenarios, notably without requiring parameter re-tuning.

V. CONCLUSIONS

This work introduces a distributed model predictive control scheme for efficiently addressing the task of quadcopter multi-agent systems navigation. By sharing information among the agents regarding their future position predictions,

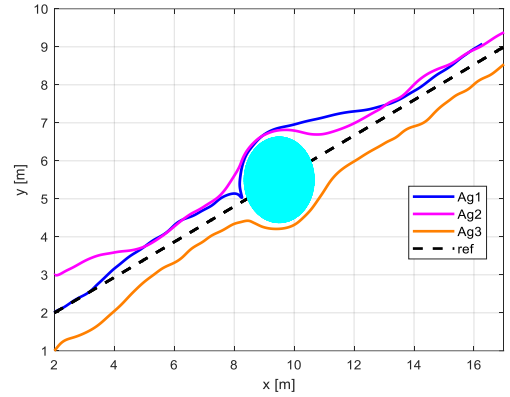

Fig. 4. Three-agent swarm x-y plot with obstacle for Case 2

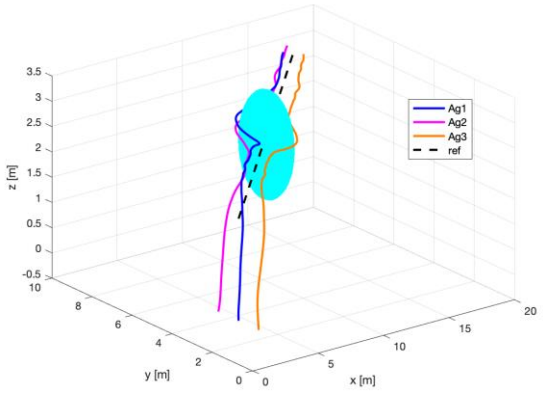

Fig. 5. Three-agent swarm 3-D plot with obstacle for Case 2

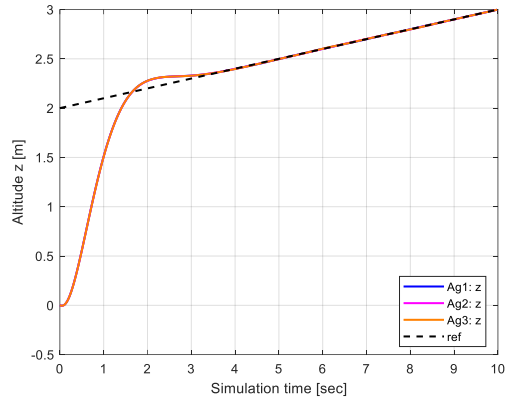

Fig. 6. Three-agent swarm z-axis trajectory tracking for Case 2

the local MPC controllers formulate and solve an optimization problem including constraints corresponding to collision and obstacle avoidance, while minimizing the deviation from a given reference trajectory. Since the reference trajectory is the same for all units and also because no formation control is imposed, the swarm agents are free to calculate their optimal trajectories online, a fact that allows them to deal with sudden and abrupt circumstances, like a-priori unknown obstacles. Future research plans include the extension of the proposed control scheme to address stability issues and the case where moving obstacles disturb the swarm smooth navigation, the evaluation of system performance with real vehicles under turbulent conditions, and the examination of larger swarms with limited vicinity-based information exchange.

ACKNOWLEDGMENT

The publication fees were covered by the University of West Attica.

REFERENCES

[1] S. Omari, M. D. Hua, G. Ducard, and T. Hamel, "Hardware and software architecture for nonlinear control of multirotor helicopters," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1724–1736, 2013.

[2] A. Kapnopoulos, C. Kazakidis, and A. Alexandridis, "Quadrotor trajectory tracking based on backstepping control and radial basis function neural networks," *Results in Control and Optimization*, vol. 14, p. 100335, 2024.

[3] S. J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A Survey on Aerial Swarm Robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[4] Y. Zhou, B. Rao, and W. Wang, "UAV swarm intelligence: Recent advances and future trends," *IEEE Access*, vol. 8, pp. 183856–183878, 2020.

[5] M. Papadimitrakis and A. Alexandridis, "Active vehicle suspension control using road preview model predictive control and radial basis function networks," *Applied Soft Computing*, vol. 120, p. 108646, 2022.

[6] M. Stogiannos, A. Alexandridis, and H. Sarimveis, "Model predictive control for systems with fast dynamics using inverse neural models," *ISA Transactions*, vol. 72, pp. 161–177, 2018.

[7] A. Kapnopoulos and A. Alexandridis, "A cooperative particle swarm optimization approach for tuning an MPC-based quadrotor trajectory tracking scheme," *Aerospace Science and Technology*, vol. 127, 2022.

[8] S. Wolfe, S. Givigi, and C. A. Rabbath, "Distributed Multiple Model MPC for Target Tracking UAVs," *2020 International Conference on Unmanned Aircraft Systems, ICUAS 2020*, pp. 123–130, 2020.

[9] L. F. Recalde, B. S. Guevara, V. Andaluz, J. Varela, J. Gimenez, and D. Gandolfo, "Nonlinear MPC for Multiple Quadrotors in Dynamic Environments," *2022 International Conference on Unmanned Aircraft Systems, ICUAS 2022*, pp. 1201–1209, 2022.

[10] M. Rosalie *et al.*, "Area exploration with a swarm of UAVs combining deterministic chaotic ant colony mobility with position MPC," *2017 International Conference on Unmanned Aircraft Systems, ICUAS 2017*, pp. 1392–1397, 2017.

[11] S. Sun and A. Franchi, "Nonlinear MPC for Full-Pose Manipulation of a Cable-Suspended Load using Multiple UAVs," *2023 International Conference on Unmanned Aircraft Systems, ICUAS 2023*, pp. 969–975, 2023.

[12] O. Mechali, J. Iqbal, J. Wang, X. Xie, and L. Xu, "Distributed Leader-Follower Formation Control of Quadrotors Swarm Subjected to Disturbances," *2021 IEEE International Conference on Mechatronics and Automation, ICMA 2021*, pp. 1442–1447, Aug. 2021.

[13] S. Zhou, X. Dong, Q. Li, and Z. Ren, "Time-varying formation tracking control for UAV-UGV heterogeneous swarm systems with switching directed topologies," *IEEE International Conference on Control and Automation, ICCA Proceedings*, pp. 1068–1073, 2020.

[14] E. Olcay, K. Gabrich, and B. Lohmann, "Optimal Control of a Swarming Multi-agent System through Guidance of a Leader-Agent," *IFAC-PapersOnLine*, vol. 52, no. 20, pp. 1–6, 2019.

[15] S. Vargas, H. M. Becerra, and J. B. Hayet, "MPC-based distributed formation control of multiple quadcopters with obstacle avoidance and connectivity maintenance," *Control Engineering Practice*, vol. 121, p. 105054, 2022.

[16] E. Soria, F. Schiano, and D. Floreano, "Distributed Predictive Drone Swarms in Cluttered Environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 73–80, 2022.

[17] F. Sabatino, "Quadrotor control: modeling, nonlinear control design, and simulation," *Master Thesis*, 2015.

[18] A. Das, K. Subbarao, and F. Lewis, "Dynamic inversion with zero-dynamics stabilisation for quadrotor control," *IET Control Theory and Applications*, vol. 3, no. 3, pp. 303–314, 2009.

[19] P. Pounds, R. Mahony, and P. Corke, "Modelling and control of a large quadrotor robot," *Control Engineering Practice*, vol. 18, no. 7, pp. 691–699, Jul. 2010.

[20] J. Nocedal, and S. J. Wright, "Numerical Optimization," *Springer Link*, 2006.