Distributed Predictive Drone Swarms in Cluttered Environments

Enrica Soria, *Student Member, IEEE* Fabrizio Schiano, *Member, IEEE*, and Dario Floreano, *Senior Member, IEEE*

***Abstract*—Recent works in aerial robotics show that the self-organized and cohesive flight of swarms can emerge from the exchange of purely local information between neighboring agents. However, most of the current swarm models are not capable of flight in densely cluttered environments. Predictive models have the potential to incorporate safe collision avoidance capabilities and give the agents the ability to anticipate and synchronize their trajectories in real-time. Here, we propose a distributed predictive swarm model that generates self-organized, safe, and cohesive trajectories by solving an optimization problem in real-time. In simulation, we show that our method is scalable to large numbers of agents and suitable for deployment in different environments, specifically a forest and a funnel-like environment. Furthermore, our results show that the agents are capable of collision-free flight with noisy sensor measurements for a noise level of up to 70% of the magnitude of the agent safety distance. Real-world experiments with a swarm of up to 16 quadrotors in an indoor artificial environment validate our method. Supplementary Materials can be found at https://doi.org/10.5281/zenodo.5245214.**

***Index Terms*—Swarm Robotics, Aerial Systems: Perception and Autonomy, Multi-robot systems**

I. Introduction

THE synergistic flight of multiple aerial robots can enable many real-world applications in industries such as mapping, agriculture, search and rescue, and construction [1]–[5]. However, to bring drone swarms from research laboratories to the real world, they should be capable of integrating the environment safely [6]–[8].

Natural collectives, such as fish or birds, show that coordinated navigation can be achieved by decentralized decision-making [9]–[12]. Indeed, their motion can be explained with a set of simple rules based on local exchange of information, such as repulsion that steers an agent away from its neighbors, cohesion that models attraction to the group, migration that orients its motion in a preferred direction, and additionally repulsion from obstacles that makes it avoid collisions with the environment [13]. Decentralized control presents a key advantage for the swarm compared to centralized control. While the latter relies on a central computing node, the former is shared among all agents, thereby improving robustness against one individual's failure [2], [7], [14], [15]. Additionally, in decentralized control strategies, each agent's decisions only depend on a limited number of neighbors and are therefore independent of the swarm size. This aspect allows the simultaneous deployment of possibly tens or hundreds of agents with the same hardware computational resources.

Manuscript received: June 11, 2021; Revised: August 26, 2021; Accepted: September 28, 2021.

This paper was recommended for publication by Editor M. Ani Hsieh upon evaluation of the Associate Editor and Reviewers' comments.

The authors are with the Laboratory of Intelligent Systems (LIS), École Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland. https://lis.epfl.ch

Corresponding author: enrica.soria@epfl.ch

Digital Object Identifier (DOI): see top of this page.

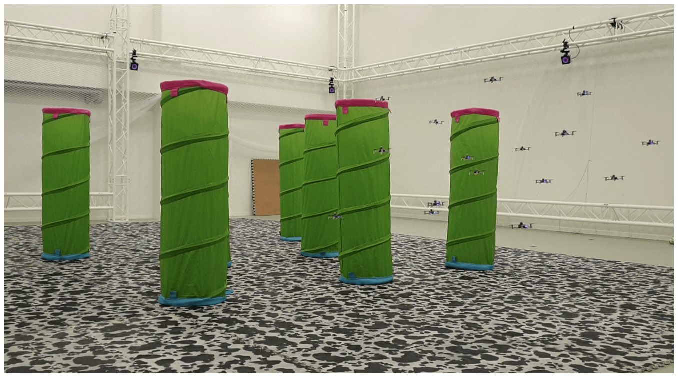
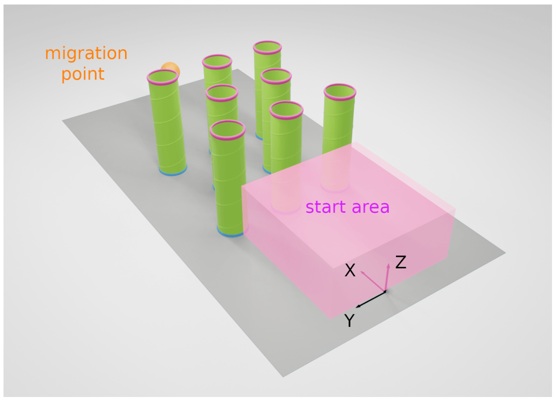


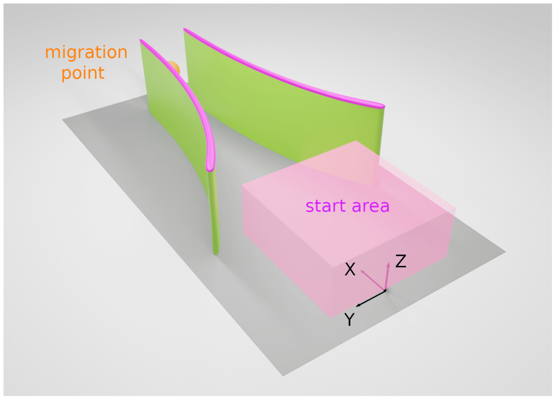
Figure 1: Self-organized predictive swarm flying in an artificial forest. Picture of a swarm of 16 drones flying in a forest-like environment in our indoor experimental facility. A supplementary video of our experiments is found at https://youtu.be/1Vg1jdw2Ruk.

A variety of swarm models based on the above rules have been successfully implemented on aerial robots by reproducing the swarm rules with artificial potential fields [15]–[18]. However, reliable obstacle avoidance capabilities are hard to obtain for real drone swarms, and they are neglected by these works. In other work, obstacle avoidance is modeled with artificial repulsive potentials located around the obstacles [14], [19]. This approach does not explicitly consider the physical limitations of the robots and, in certain settings, these limitations can result in infeasible control inputs that lead to unexpected collisions. In practice, this issue is solved by finding parameter values (i.e., preferred speed, cohesion, repulsion, and other coefficients) adapted to the environment and swarm configurations (i.e., size, number, and density of the obstacles) [14], [20].

Recent works suggest that predictive controllers can improve the safety of aerial swarms by predicting and optimizing the agents' future behavior in an iterative process [21]–[23]. Model Predictive Control (MPC) computes the control action of a system as the solution to an optimization problem that explicitly accounts for the robot dynamics and actuation constraints. Moreover, the computations can be shared among all agents according to a Distributed MPC (DMPC) formulation. With DMPC, every robot solves an optimal problem locally and then communicates its solution to the others to allow global coordination. Following this control scheme, multiple drones can reliably avoid reciprocal collisions when



(a) Forest-like environment



(b) Funnel-like environment

Figure 2: Modelling of the environment. On the left, 2a shows the modeling of the forest-like environment filled with cylindrical obstacles. On the right, 2b shows the funnel-like environment made of two curved surfaces that gradually reduce the flight volume towards the migration point. In both environments, the allowed flight workspace is set to $[8.5, 8.5, 1.1]\ m^3$ and the migration point is in $\boldsymbol{p}_{\text{mig}} = [7.5, 0, 0.6]\ m$). The swarm flies from the start region (pink cube) in the foreground of the scene towards the migration point (orange sphere) in the background.

assigned intersecting trajectories [21]. This approach is extended in [22], where the authors present a DMPC motion planner that allows the real-time and collision-free trajectory generation for swarms of up to 20 drones with a single off-board computer. Their on-demand Collision Avoidance (CA) method reduces the computation and travel time compared to Buffered Voronoi Cells (BVC) CA methods [24], [25]. These works show the remarkable potential of modern optimization-based motion planners for solving CA problems of collective systems, although they are designed for individual point-to-point transitions and do not generate self-organized cohesive flight similar to biological swarms.

In our previous work [20], we showed that the collective behavior of biological swarms could be reproduced with an NMPC (Nonlinear MPC) model. The results indicated improved safety and flight synchronization at different obstacle densities, inter-agent distances, and speeds compared to purely reactive approaches based on artificial potential fields [14]. However, the centralized nature of this model allowed the real-time control of only five drones and prevented it from scaling to a large number of drones. Also, the non-convex formulation of the optimization problem required using a computationally expensive scheme based on SQP (Sequential Quadratic Programming) [26].

Here, we present a novel and scalable DMPC swarm model that allows a safe and cohesive flight of aerial swarms in cluttered environments (Fig. 1). We show its scalability in the swarm size and its robustness to noise by systematically analyzing the swarm performance at different agents number and noise levels. The swarm performance is studied and compared for two different environments: a forest and funnel-like environment. We also compare the performance of the presented DMPC swarm model with different reciprocal CA methods, i.e., BVC, on-demand, and continuous CA. Finally, we validate the proposed algorithm in real-world experiments with up to 16 palm-sized quadrotors.

II. Methods

We consider a swarm composed of N agents labeled by $i \in \mathcal{V} = \{1, \dots, N\}$ and a set of M static obstacles labeled by $m \in \mathcal{M} = \{1, \dots, M\}$ (Fig. 2). The swarm can be modeled with a directed sensing graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertex set $\mathcal{V}$ represents the agents, and the edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ contains the pairs of agents (i, j) for which agent i can sense agent j. The state of the i-th agent is represented by $\boldsymbol{x}_i = (\boldsymbol{p}_i, \boldsymbol{v}_i) \in \mathbb{R}^6$ and is made of its position $\boldsymbol{p}_i \in \mathbb{R}^3$ and velocity $\boldsymbol{v}_i \in \mathbb{R}^3$. In the following, k denotes the index of a discrete time step with duration dt. In our DMPC scheme, at every step k, each agent i computes its neighborhood $\mathcal{N}_i^k = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$ according to the topological metric, which is defined as the set of the n nearest neighbors of agent i at time k [27]. If we indicate with $|\cdot|$ the cardinality of a set, then $|\mathcal{N}_i^k| = n$ is fixed for all instants k, although the neighbors in the set may vary at different k. Then, the agents determine their optimal trajectory w.r.t. their neighbors by solving a constrained optimization problem over a fixed time window called the *prediction horizon* and denoted as $T_P = P\ dt$, $P \in \mathbb{N}^+$ (Fig. 3a). The optimization problem aims at minimizing a cost function, which encodes the swarm rules, under constraints that include the dynamic limitations of the agent and the trajectory smoothness. The swarm rules comprise the migration, which steers the agents towards a common goal, the regulation of the inter-agent distance, which consists of cohesion and agents' reciprocal avoidance, and obstacle avoidance, which steers the agents away from obstacles. Additionally, the control effort rule minimizes the energy spent on maneuvering.

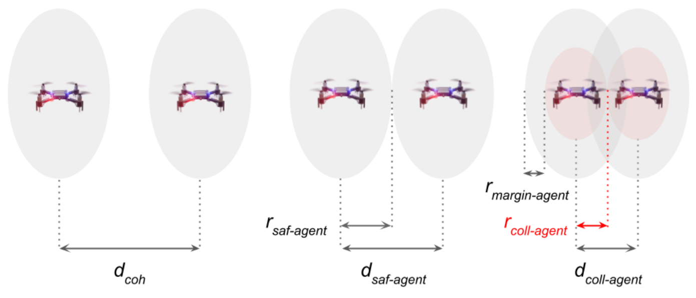
A. Model of a flying agent

Every agent of the swarm obeys a discrete linear system:

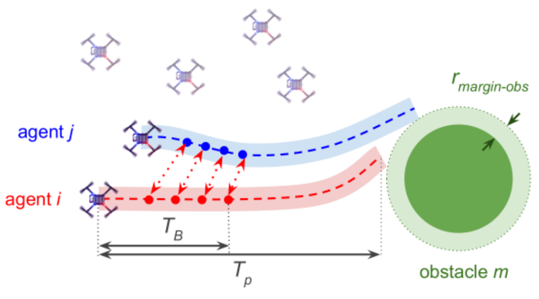
$$\boldsymbol{x}_i(k+1) = \boldsymbol{A}_i \boldsymbol{x}_i(k) + \boldsymbol{B}_i \boldsymbol{u}_i(k) \tag{1}$$

where $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ are constant matrices modeling the dynamics of the Crazyflie 2.1 quadrotor with an underlying position controller. To account for the dynamic feasibility, we limit the position commands by the dimensions of the environment and the acceleration by constant vectors. Hence, it holds $\boldsymbol{p}_{\text{min}} \leq \boldsymbol{u}_i(k) \leq \boldsymbol{p}_{\text{max}}$ and $\boldsymbol{a}_{\text{min}} \leq \boldsymbol{a}_i(k) \leq \boldsymbol{a}_{\text{max}}$.

To quantify the effects of noisy sensor measurements on the flight performance, we model the noise on the agents'



(a) Inter-agent distances



(b) Agent's predicted trajectory

Figure 3: Schematic overview of the method. In 3a, inter-agent distance parameters: on the left, the cohesion distance $d_{\text{coh}} = 1.30\ m$ (i.e., the maximum allowed distance between neighboring drones), in the middle, the safety distance $d_{\text{saf-agent}} = 0.30\ m$, and on the right, the collision distance $d_{\text{coll-agent}} = 0.14\ m$. In 3b, illustration of the predicted trajectory for the focal agent i. At every time step k, the focal agent i determines the neighbor set $\mathcal{N}_i^k$, then computes the distance to the migration point, the relative distances to its neighbors, and potential obstacle collisions over the horizon $T_P = 3.0\ s$. Agents' cohesion and reciprocal avoidance with the proposed continuous CA method are enforced over $T_B = 1.8\ s$. Trajectory replanning happens every $0.2\ s$.

positions with a random normal distribution with zero average and equal standard deviation in the three dimensions, σ_p.

B. Trajectory parameterization

We model the agents' trajectories with l Bezier curves in $\mathbb{R}^3$ of duration T_l, in the same spirit as [22], [28]. This parameterization allows us to define continuous input trajectories to the drones from a finite set of control points. A 3-dimensional Bezier curve of order d is uniquely characterized by a set of $d+1$ control points $\tilde{\mathcal{U}} = \{\tilde{\boldsymbol{u}}_0, ..., \tilde{\boldsymbol{u}}_d\} \in \mathbb{R}^{3(d+1)}$ and the trajectory of agent i is defined by $l(d+1)$ control points $\tilde{\boldsymbol{U}}_i \in \mathbb{R}^{3l(d+1)}$. In the following, $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$ are considered as function of the new unknown $\tilde{\boldsymbol{U}}_i$. At each time k, we can rewrite the dynamic feasibility conditions described above as:

$$\boldsymbol{A}_{\text{dyn},i}\tilde{\boldsymbol{U}}_i^k \leq \boldsymbol{b}_{\text{dyn},i} \tag{2}$$

To obtain smooth trajectories, we impose continuity requirements $\mathcal{C}^2$ on the input curve. Samples of the Bezier curves and their derivatives are obtained by linear combinations of the control points. Hence, at each time k, the continuity conditions translate in linear equality constraints of the form:

$$\boldsymbol{A}_{\text{cont},i}\tilde{\boldsymbol{U}}_i^k = \boldsymbol{b}_{\text{cont},i} \tag{3}$$

C. Migration

A *migration point* $\boldsymbol{p}_{\text{mig}} \in \mathbb{R}^3$ known by all agents orients the motion of the swarm in a common direction. The advantage of choosing a migration point over a migration velocity as in [20] is that, even if the agents' path is deviated by some obstacles, the agents will correct their direction to reach the expected destination. We let $(\cdot)(k+\pi|k)$ represent the predicted value of $(\cdot)(k+\pi)$ with the information available at time step k. Then, the migration term is defined as:

$$J_{\text{mig},i}^k = \sum_{\pi=1}^{P} q_{\text{mig}} \|\boldsymbol{p}_i(k+\pi|k) - \boldsymbol{p}_{\text{mig}}\|_2^2 \tag{4}$$

where q_{mig} is the weight of the migration behavior.

D. Agents' reciprocal avoidance

Safety among agents is ensured by requiring neighboring couples of agents to fly at a distance larger than a *safety inter-agent distance* $d_{\text{saf-agent}}$ (Fig. 3a). To model the down-wash effect of the quadrotors, 2-norm distances between agents are scaled according to a weight matrix E, with positive diagonal elements $E_{xx} = E_{yy} = 1$ and $E_{zz} < 1$ that defines an ellipsoidal distance $\|\cdot\|_E$. In this paper, we explore three methods for Collision Avoidance (CA): (i) BVC [24], [25], (ii) on-demand [22], and (iii) continuous CA, the method that we propose. All methods are based on the principle of imposing hyperplane constraints that limit the available space over which the agent optimizes its future trajectory to avoid collisions with its neighbors. In the following, we describe each of them.

BVC CA. In the BVC method, each agent i is forced to stay within its Buffered Voronoi Cell $\mathcal{V}_i$ for a time T_l corresponding to the first Bezier curve of its input trajectory. Let $d_{ij} = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|_E$ be the 2-norm scaled distance between agents i and j, then the Buffered Voronoi Cell of agent i is:

$$\mathcal{V}_i = \left\{ \frac{(\boldsymbol{p}_i - \boldsymbol{p}_j)^T E^{-2} (\boldsymbol{p} - \boldsymbol{p}_i)}{d_{ij}} \geq \frac{d_{\text{saf-agent}} - d_{ij}}{2},\ \forall j \in \mathcal{N}_i \right\} \tag{5}$$

Condition 5 translates into a linear constraint per each of the $(d+1)$ control points of the Bezier curve. For more details on the BVC method we refer to [22], [24], [25].

On-demand CA. This method is based on an event-triggered strategy. It assumes communicative agents that share with their neighbors their predicted actions (i.e., $\boldsymbol{u}_i(k+\pi|k)$, $\pi \in \{0, ..., P-1\}$), and imposes constraints only if potential collisions are detected. On-demand CA only constrains one sample of the agents' trajectory, corresponding to the time of the first detected collision. If agent i detects the first collision at time $k_{\text{coll},i}$, then avoidance constraints are enforced with all its neighbors at that time. Based on the results in [22], we write the constraints in the input space as:

$$\|\boldsymbol{u}_i(k_{\text{coll},i}|k) - \boldsymbol{u}_j(k_{\text{coll},i}|k)\|_E \geq d_{\text{saf-agent}} \tag{6}$$

which results in collision-free position commands. Then, we linearize them with a first-order Taylor expansion. For more details on the on-demand method we refer to [22].

Continuous CA. In this method, the constraints have the same formula as in the on-demand method, but they are enforced over an entire trajectory segment rather than a single sample. We define the *braking time* $T_\text{B} = B\ dt \leq T_P$ as the minimum time required by an agent that flies at maximum speed to brake until zero velocity. Continuous CA constraints for agent i are enforced with the neighbors $j \in \mathcal{N}_i^k$ over the horizon T_B. Hence, neighboring agents need to share their predicted actions over the braking horizon (i.e., $\boldsymbol{u}_i(k+\pi|k),\ \pi \in \{0, ..., B-1\}$). As for on-demand CA, these constraints are written in the input space. They are:

$$\|\boldsymbol{u}_i(k+\pi|k) - \boldsymbol{u}_j(k+\pi|k)\|_E \geq d_\text{saf-agent} \tag{7}$$

with $\pi \in \{0, ..., B-1\}$. Because of the larger number of constraints, continuous CA leads to more conservative maneuvers than on-demand CA. Also in this case, we approximate the constraints with a first-order Taylor expansion.

For all methods, we introduce a set of slack variables $\boldsymbol{\mathcal{E}}_i^k$ that relax the hyperplane constraints and make it more likely for the optimization problem to find a viable path. Each variable $\epsilon_{ij} \geq 0$ indicates the amount by which the avoidance constraint between agent i and neighbor j is violated. For example, for continuous CA, the relaxed constraints are:

$$\|\boldsymbol{u}_i(k+\pi|k) - \boldsymbol{u}_j(k+\pi|k)\|_E \geq d_\text{saf-agent} - \epsilon_{ij}(k+p|k) \tag{8}$$

More generally, for all CA methods, we indicate the set of linear reciprocal CA constraints for agent i at time k as:

$$\boldsymbol{A}^k_{\text{saf-agent},i}[(\tilde{\boldsymbol{U}}_i^k)^T, (\boldsymbol{\mathcal{E}}_i^k)^T]^T \leq \boldsymbol{b}^k_{\text{saf-agent},i} \tag{9}$$
$$-\boldsymbol{\mathcal{E}}_i^k \leq 0 \tag{10}$$

The cost associated with the violation of these constraints includes linear and quadratic terms in ϵ_{ij} with constant weights l_saf and q_saf, respectively. For example, for continuous CA, the total violation cost is:

$$J^k_{\text{saf-agent},i} = \sum_{j\in\mathcal{N}_i}\sum_{\pi=0}^{B-1}\left(l_\text{saf}\epsilon_{ij}(k+\pi|k) + q_\text{saf}\epsilon^2_{ij}(k+\pi|k)\right) \tag{11}$$

E. Agents' cohesion

The swarm behavior of cohesion requires neighboring couples of drones to stay closer than the *cohesion distance* d_coh. We introduce slack variables δ_{ij} for agent i and neighbors $j \in \mathcal{N}_i^k$ and we formulate the cohesion constraint over the horizon T_B in the input space as:

$$\|\boldsymbol{u}_i(k+\pi|k) - \boldsymbol{u}_j(k+\pi|k)\|_E \leq d_\text{coh} + \delta_{ij}(k+\pi|k) \tag{12}$$

If we indicate with $\boldsymbol{\Delta}_i^k$ the set of slack variables for agent i at time k, then the set of cohesion constraints approximated by a first-order Taylor expansion is denoted as:

$$\boldsymbol{A}^k_{\text{coh},i}[(\tilde{\boldsymbol{U}}_i^k)^T, (\boldsymbol{\Delta}_i^k)^T]^T \leq \boldsymbol{b}^k_{\text{coh},i} \tag{13}$$
$$-\boldsymbol{\Delta}_i^k \leq 0 \tag{14}$$

The cost associated with the violation of the constraints is:

$$J^k_{\text{coh},i} = \sum_{j\in\mathcal{N}_i}\sum_{\pi=0}^{B-1}\left(l_\text{coh}\delta_{ij}(k+\pi|k) + q_\text{coh}\delta^2_{ij}(k+\pi|k)\right) \tag{15}$$

where l_coh and q_coh are constant weights.

F. Obstacle avoidance

The obstacle avoidance behavior is produced by requiring the drones to fly at a *safety distance* $d_\text{saf-obs}$ from the obstacles. To keep this behavior local, each agent only considers the first obstacle on the collision course with its predicted trajectory. In this article, we consider convex obstacles that we model with 3D ellipsoids with arbitrary axes dimensions. If a drone i detects its first collision with obstacle m at instant $k_{\text{coll},i}$ it adds an anti-collision constraint with it to its optimization problem. We introduce the slack variable $\zeta_{im} \geq 0$ referred to agent i and obstacle m and consider the following constraint:

$$\|\boldsymbol{u}_i(k_{\text{coll},i}|k) - \boldsymbol{p}_m\|_E \geq d_\text{saf-obs} - \zeta_{im}(k_{\text{coll},i}|k) \tag{16}$$

As done before, we indicate with $\boldsymbol{\mathcal{Z}}_i^k$ the obstacle avoidance slack variables for agent i at time k, and write the first-order approximation of the above constraint as:

$$\boldsymbol{A}^k_{\text{saf-obs},i}[(\tilde{\boldsymbol{U}}_i^k)^T, (\boldsymbol{\mathcal{Z}}_i^k)^T]^T \leq \boldsymbol{b}^k_{\text{saf-obs},i} \tag{17}$$
$$-\boldsymbol{\mathcal{Z}}_i^k \leq 0 \tag{18}$$

The cost associated with the violation of the constraint is:

$$J^k_{\text{saf-obs},i} = l_\text{saf}\zeta_{im}(k_{\text{coll},i}|k) + q_\text{saf}\zeta^2_{im}(k_{\text{coll},i}|k) \tag{19}$$

G. Control effort

The *control effort* is responsible for minimizing the energy required by the control commands. It is defined by a weighted sum of the second squared derivative of the input commands that penalizes acceleration and deceleration of an agent:

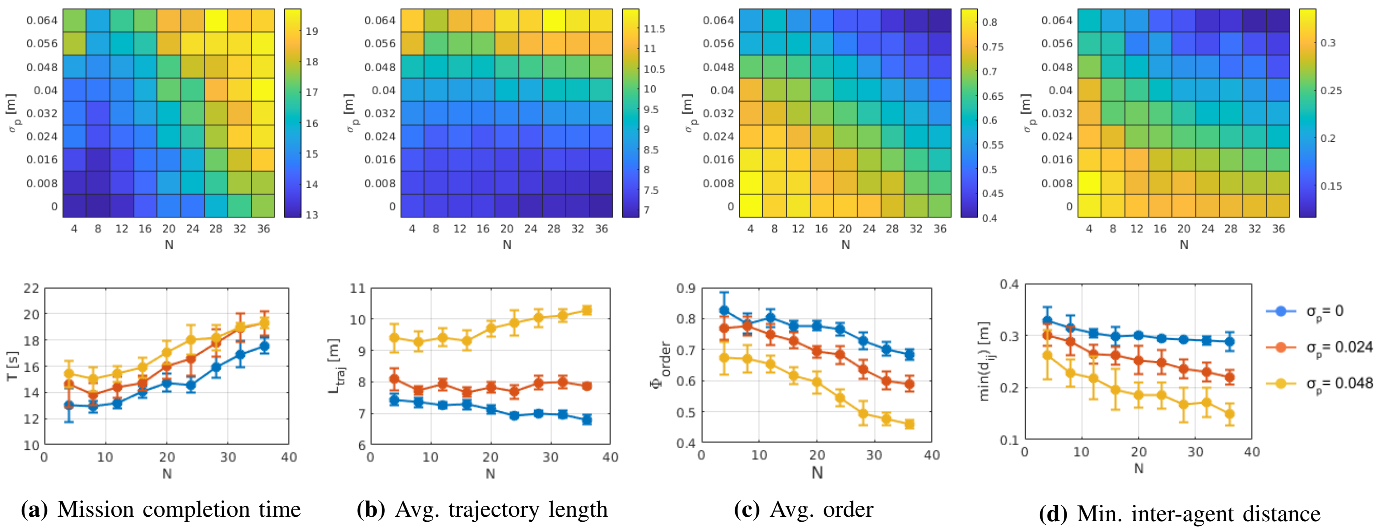
$$J^k_{\text{effort},i} = \sum_{\pi=0}^{P-1} q_\text{effort}\left\|\frac{d^2}{dt^2}\boldsymbol{u}_i(k+\pi|k)\right\|_2^2 \tag{20}$$

where q_effort is the weight of the control effort rule.

H. Desired trajectory

To calculate the desired trajectory at each time step k, every drone i solves the following QP problem, which includes all the above cost terms and constraints:

$$\begin{aligned}
\min_{\tilde{\boldsymbol{U}}_i^k, \boldsymbol{\mathcal{E}}_i^k, \boldsymbol{\Delta}_i^k, \boldsymbol{\mathcal{Z}}_i^k} \quad & J^k_{\text{mig},i} + J^k_{\text{saf-agent},i} + J^k_{\text{coh},i} + J^k_{\text{saf-obs},i} + J^k_{\text{effort},i} \\
\text{subject to:} \quad & \\
& \boldsymbol{A}_{\text{dyn},i}\tilde{\boldsymbol{U}}_i^k \leq \boldsymbol{b}_{\text{dyn},i} \\
& \boldsymbol{A}_{\text{cont},i}\tilde{\boldsymbol{U}}_i^k = \boldsymbol{b}_{\text{cont},i} \\
& \boldsymbol{A}^k_{\text{saf-agent},i}[(\tilde{\boldsymbol{U}}_i^k)^T, (\boldsymbol{\mathcal{E}}_i^k)^T]^T \leq \boldsymbol{b}^k_{\text{saf-agent},i} \\
& \boldsymbol{A}^k_{\text{coh},i}[(\tilde{\boldsymbol{U}}_i^k)^T, (\boldsymbol{\Delta}_i^k)^T]^T \leq \boldsymbol{b}^k_{\text{coh},i} \\
& \boldsymbol{A}^k_{\text{saf-obs},i}[(\tilde{\boldsymbol{U}}_i^k)^T, (\boldsymbol{\mathcal{Z}}_i^k)^T]^T \leq \boldsymbol{b}^k_{\text{saf-obs},i} \\
& -\boldsymbol{\mathcal{E}}_i^k \leq 0 \\
& -\boldsymbol{\Delta}_i^k \leq 0 \\
& -\boldsymbol{\mathcal{Z}}_i^k \leq 0
\end{aligned} \tag{21}$$



(a) Mission completion time (b) Avg. trajectory length (c) Avg. order (d) Min. inter-agent distance

Figure 4: Swarm performance in forest-like environment. On top, simulation results for swarms with different agent numbers and noise levels. For each parameter combination (i.e., one bin in the heatmaps), the results show the average of 10 random simulations. Specifically, 4a reports the mission completion time T, 4b reports the average trajectory length L_{traj}, 4c reports the average order Φ_{order}, and 4d reports the minimum inter-agent distance $\min(d_{ij})$. At the bottom, aggregate average and standard deviation of the same metrics for three different noise levels: in blue $\sigma_p = 0\ m$, in orange $\sigma_p = 0.024\ m$, and in yellow $\sigma_p = 0.048\ m$. Collisions between agents happen at noise levels $\sigma_p \geq 0.056\ m$. Instead, collisions with the obstacles already happen at $\sigma_p \geq 0.040\ m$.

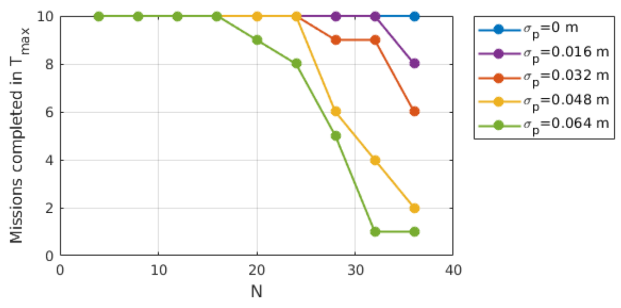
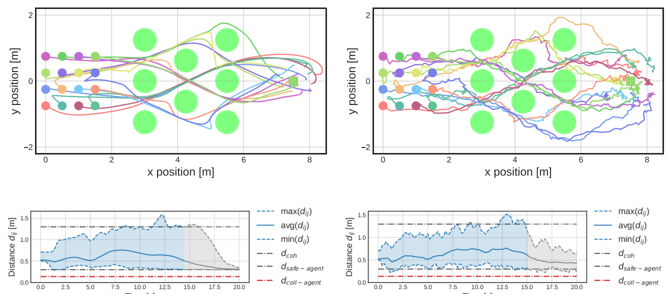


Figure 5: Number of completed missions in time $t \leq T_{\max}$. Number of completed missions (out of 10 random simulations) for swarms of different agent numbers N at different noise levels σ_p, flying in the forest-like environment. We consider the mission completed only if the swarm gets to the migration point before a maximum time of $T_{\max} = 20\ s$. For a noise level $\sigma_p = 0\ m$, the swarm could complete all missions.

I. Swarm performance metrics

We assess the performance of the swarm's flight according to six different metrics. The mission completion time T measures the time that the swarm requires to complete a mission. A mission is completed if the swarm average position reaches the migration point up to a tolerance distance d_{tol} and if, at the same time, all the drones are within the distance d_{coh} from their neighbors. The trajectory length L_{traj} measures the average of the agents' flown distances until they complete the mission. The minimum and the maximum inter-agent distances, $\min(d_{ij})$ and $\max(d_{ij})$, measure the minimum and the maximum distance among neighboring couples of drones over the mission. The minimum distance to the obstacles $\min(d_{im})$ measures the minimum distance between all agents and all obstacles. Finally, the order Φ_{order} measures the average correlation of the agents' directed movements. It is often used to quantify the synchronization of the agents' flight [10], [14],



(a) Noise level $\boldsymbol{\sigma_p} = 0\ m$ (b) Noise level $\boldsymbol{\sigma_p} = 0.048\ m$

Figure 6: Swarm trajectories at different noise levels. Simulated trajectories and inter-agent distances for a swarm of 16 drones in a forest-like environment at two noise levels ($\boldsymbol{\sigma_p} = 0$ and $0.048\ m$). Although we report no collisions in both cases, the trajectories are visibly smoother at zero noise level (6a) than in the presence of noise (6b). The bottom row shows the envelope of the distance between neighboring drones d_{ij}. The plots are grayed out from the mission completion time T until the simulation end time $T_{\max}$.

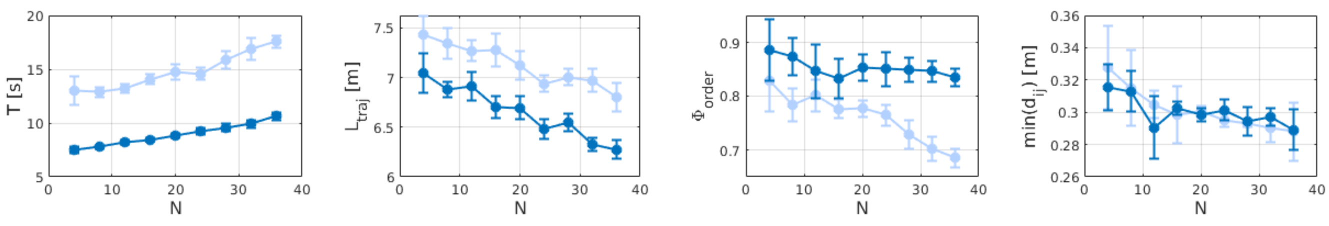
and in formula, it is written as:

$$\Phi_{\text{order}} = \sum_{k\in\{1,\dots,K\}} \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i^k} \frac{\boldsymbol{v}_i(k)\boldsymbol{v}_j(k)}{KNn\|\boldsymbol{v}_i(k)\|\|\boldsymbol{v}_j(k)\|} \tag{22}$$

where $K = \min(\lceil T/dt \rceil, \lceil T_{\max}/dt \rceil)$ and $\lceil \cdot \rceil$ is the ceiling function.

III. Simulation results

We implemented our swarm model in MATLAB 2020b, and executed our simulations on a computer equipped with Intel Core i7-8750H CPU with 12 cores and 16 GB of RAM. For the solution of the optimization problem, we used the active set algorithm [29]. The parameter values used in simulation experiments are detailed in the Supplementary Materials.



(a) Mission completion time **(b)** Avg. trajectory length **(c)** Avg. order **(d)** Min. inter-agent distance

Figure 7: Swarm performance comparison in the funnel and forest-like environments. Aggregate results (average and standard deviation of 10 random simulations) of the swarm performance in the funnel-like (in blue) and forest-like (in light blue) environments at zero noise level and for different agent numbers N. For all agent numbers, we report zero agent-agent collisions.

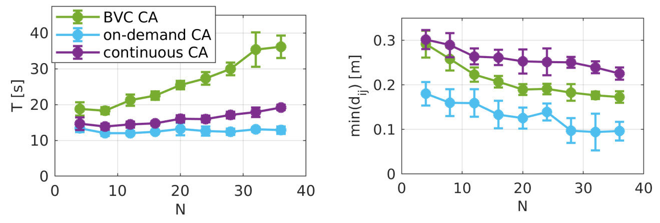
A. Scalability in the agent number and noise robustness

We run swarm simulations for 9 swarm sizes $N \in \{4, 8, 12, 16, 20, 24, 28, 32, 36\}$ and 9 noise levels, $\sigma_p \in \{0, 0.008, 0.016, 0.024, 0.032, 0.040, 0.048, 0.056, 0.064\}$ m in the forest-like environment (Fig. 2a). For every configuration, we run 10 random simulations and averaged the results. In the computation of the swarm performance, we only considered the missions that the swarm could complete within $T_{\text{max}} = 20\ s$ (Fig. 5).

The results show that the coordination of a large swarm requires a longer time than a small swarm. Analogously, an increase in the noise level requires extra time for the swarm to reach the migration point (Fig. 4a). The increased mission time in the presence of noise can be explained by an increment in the agents' trajectory lengths (Fig. 4b). Instead, for swarms of large sizes, the trajectory lengths seem almost stationary for a given noise level (Fig. 4b), necessarily implying a lower speed. The average swarm order remarkably reduces when the swarm size and noise level increase (Fig. 4c). Specifically, when passing from zero noise level to noise level $\sigma_p = 0.048\ m$ the order decreases of about 21 to 32% depending on the swarm size ($\Phi_{\text{order}} \approx 0.83$ to 0.68 for $N = 4$ to 36 at $\sigma_p = 0\ m$, while $\Phi_{\text{order}} \approx 0.67$ to 0.46 for $N = 4$ to 36 at $\sigma_p = 0.048\ m$). The reduction of order in the presence of noise, indicating a decrease in the correlation of the agents' movements, is due to the agents' need of adjusting their directions in order to avoid collisions (Fig. 6b). This tendency increases with the agent number. With no noise, the minimum inter-agent distance is approximately steady around the safety value ($\min(d_{ij}) \approx 0.3\ m$) independently on the swarm size (Fig. 4d). Instead, in the presence of noise, the minimum distance decreases when the swarm size increases (Fig. 4d). On average, collisions between drones happen from a noise level $\sigma_p = 0.056\ m$, which is approximately 70% of the magnitude of the safety margin $r_{\text{margin-agent}}$ and 80% of agent collision radius $r_{\text{coll-agent}}$. Finally, the minimum distance to obstacles showed the same trend as the minimum inter-agent distance. We did not observe a significant trend for the maximum inter-agent distance, which slightly fluctuates above the cohesion distance ($\max(d_{ij}) = 1.40 \pm 0.09\ m$ on average).

B. Adaptability to different environments

To evaluate the swarm flight quality in different environments, we simulate our swarm model in a funnel-like environment (Fig. 2b) for the same swarm sizes as in the forest-like environment and compare the results.



(a) Mission completion time **(b)** Min. inter-agent distance

Figure 8: Comparison of collision avoidance methods. Comparison of the average performance metrics for different CA methods: BVC, on-demand, and continuous CA. The shown data is the average over 10 random trials for each swarm size.

In the funnel-like environment, the swarm flies overall shorter trajectories to get to the migration point compared to the forest-like environment (Fig. 7b). As a result, the mission times are generally shorter (Fig. 7a). The mission completion time and the average trajectory length present the same trends in both environments: while the first increases with the agent number, the second slightly decreases. The order is almost steady across the swarm sizes (Fig. 7c) since all agents fly consistently along the x-direction. Instead, in the forest-like environment, the swarm is less ordered due to the obstacle avoidance maneuvers, and this effect is amplified at large swarm sizes. Finally, we report zero collisions and the maximum inter-agent distance stayed below the cohesion distance for all agent numbers. However, in the funnel environment we notice a phenomenon of swarm compression characterized by a decrease of the average inter-agent distance towards the end of the funnel and due to the reduced volume.

C. Comparison of collision avoidance methods

We compared the performance of the optimization-based swarm model with the three reciprocal CA methods. In order to test the algorithms as the agent density increases, we run 10 random simulations in the forest-like environment for all the swarm sizes. For brevity and clarity, we present here the results at a fixed noise level (i.e., $\sigma_p = 0.024\ m$), although similar trends can be observed for other noise levels too.

Being the least conservative, on-demand CA leads to the fastest mission completion times for all swarm sizes (20% and 49% of average time reduction over all swarm sizes compared to the continuous and BVC methods, respectively (Fig. 8a). However, continuous CA outperforms the other two methods in terms of safety. This evidence is supported by the plot of the minimum inter-agent distance (Fig. 8b). For continuous

Solve time	Swarm size								
	4	8	12	16	20	24	28	32	36
Avg $[ms]$	9.3	9.3	9.4	9.8	10.7	11.3	12.1	12.3	16.1
Std $[ms]$	2.6	1.7	2.0	2.4	3.0	3.4	4.4	4.9	8.4

Table I: Empirical runtime per agent for different swarm sizes. Empirical runtime results (average and standard deviation) for the algorithm implementing continuous CA method. The shown data is the average over 10 random trials.

CA, the minimum inter-agent distance stays close to the safety value (ranging from 0.30 to 0.23 m for swarms of 4 to 36 agents). Instead, for the other two methods it is significantly lower (ranging from 0.29 to 0.18 m and from 0.18 to 0.10 m for swarms of 4 and 36 agents and for the BVC and the on-demand methods, respectively). Indeed, with the BVC and on-demand methods we recorded occasional collisions.

D. Computational complexity

We formally analyze the computational complexity of the presented algorithm with the three reciprocal CA methods in terms of computation time per agent to build constraints and solve the QP. In our notation, $|\mathcal{N}_i^k|$ indicates the number of considered neighbors for agent i at time step k, which is constant and equal to n. The number of reciprocal CA constraints for each algorithm scales linearly with the number of neighbors, i.e., $O(n)$. The BVC method adds $(d+1)n$ slack variables for relaxing the reciprocal avoidance constraints, where d is the Bezier curve order. The on-demand method adds only n slack variables and hence has the lowest complexity, while our method adds Bn new decision variables. Since in our method the number of variables increases with B, increasing the agents' maximum speed increases the computational complexity. Solving a standard QP problem has complexity $O(\nu^3)$, where ν is the number of decision variables, which is linear in the planning horizon P and the number of neighboring robots n. Hence, for each robot, the total computational complexity is $O(\nu^3)$.

Numerical values for the runtime depend on the used hardware and solver capabilities. Here, we report the empirical runtime of the algorithm running on the ground station with continuous CA in the forest-like environment and for different swarm sizes (Table I). We expect the runtime per agent to be independent of the swarm size since we chose agents' neighborhood to have a fixed cardinality (i.e., n). Instead, we notice that it increases as we add more agents to the swarm. We explain this fact with an increase in the swarm density, which implies more complex QPs to be solved for each agent, due to its closer proximity to the obstacles, neighbors, and environment boundaries.

E. Hardware experiments

We implemented the swarm model in Python, where we used numba[1] to speed up the mathematical computations, OSQP to solve the QP [30], and the Crazyswarm interface to communicate with the Crazyflie 2.1 drones [31]. An Optitrack motion capture system acquired the drone positions and streamed them to the ground station. Then, the ground station computed the optimal predicted trajectories online and for all drones in parallel and broadcast them to the swarm via two radio links. In our experiments, local communication between drones is mimicked by the ground station exchanging information between threads. Instead, full on-board implementation would require direct communication between neighboring robots. Moreover, drones should embed sensors to estimate their relative positions to obstacles and more powerful computers to solve the optimization problems on-board. The model parameters are the same as above, except for the migration point ($\boldsymbol{p}_{\text{mig}} = [6, 0, 1]$ m) to fit our experimental room.

[1]https://numba.pydata.org/

Metric	Simulation		Hardware	
	8 agents	16 agents	8 agents	16 agents
T $[s]$	9.2	9.0	8.9	8.7
L_{traj} $[m]$	5.88	5.47	5.93	5.49
Φ_{order} $[-]$	0.93	0.88	0.88	0.85
$\min(d_{ij})$ $[m]$	0.31	0.26	0.26	0.20
$\max(d_{ij})$ $[m]$	1.31	1.26	1.31	1.45
$\min(d_{im})$ $[m]$	0.22	0.18	0.09	0.08

Table II: Comparison of simulation and hardware swarm performance. Comparison of the performance metrics of two swarms of 8 and 16 drones between simulation and hardware experiments.

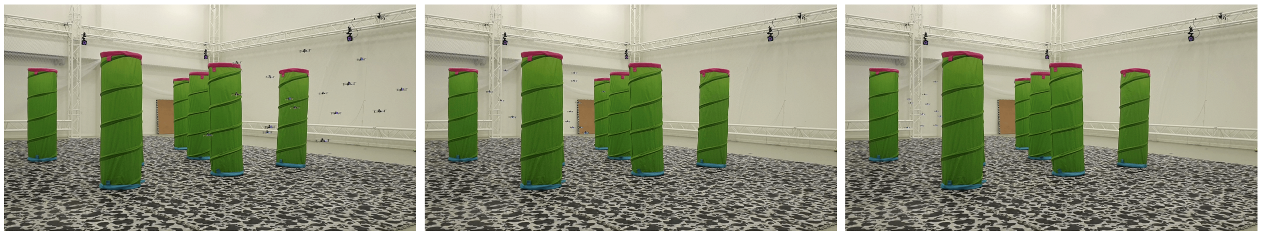
The swarms of 8 and 16 drones reach the migration point in comparable times and cover comparable distances (see Table II). However, the trajectory lengths in hardware experiments are slightly longer than in simulation because of the small positional errors in real-world experiments. The average order decreases when passing from 8 to 16 drones ($\Phi_{\text{order}} = 0.88$ and 0.85, respectively). This result follows the simulation experiments ($\Phi_{\text{order}} = 0.93$ and 0.88 for 8 and 16 agents, respectively) and confirms the above statistic results in the forest-like environment: larger swarms decrease their order to insure collision avoidance in cluttered environments. The minimum inter-agent distance remains above the collision threshold in all cases. However, it decreases when increasing the swarm size ($\min(d_{ij}) = 0.26$ and 0.20 m for the swarms of 8 and 16 drones, respectively). Finally, the agents do not collide with obstacles.

IV. Conclusion

In this article, we described a distributed MPC model for aerial swarms that results in self-organized, safe, and cohesive flight in cluttered environments. The proposed DMPC algorithm with the continuous collision avoidance method generates collection-free trajectories even in the presence of sensor noise at levels up to 70% of the magnitude of the agent safety margin distance. We validated the proposed algorithm in two types of simulated environments with obstacles, and on 16 palm-sized drones flying in a real forest-like indoor environment.

While this work paves the way for large and safe decentralized aerial drone swarms, future work should focus on the challenges for a full on-board implementation. Work in this direction will address communication issues for large multi-agent systems such as communication delays, packets losses, interference, and synchronization.

Additionally, future work will explore swarm navigation in non-convex configuration space. In the presence of concave obstacles, the current obstacle avoidance strategy may lead to deadlocks due to the presence of local minima. Hence,



(a) Time $t = 0\ s$ **(b)** Time $t = 4\ s$ **(c)** Time $t = 8\ s$

Figure 9: Experimental results with 16 drones in a forest-like environment. Snapshots at three instants (from left to right, $t = 0,\ 4,\ 8\ s$) of a swarm of 16 drones flying from the start area (in the foreground) to the migration point (in the background).

the integration of alternative techniques based on topological planning as in [32] should be investigated to solve this issue.

Acknowledgments

We gratefully acknowledge Y. Lian, J. Yuning, and Prof. C. Jones for the helpful discussions. We thank M. Pfister, A. De Bortoli and F. Schilling for their support technical advice. This work was supported by the Swiss National Science Foundation (SNSF) with grant number 200020_188457 and the European Union's Horizon 2020 programme under grant agreement ID: 871479 AERIAL-CORE.

References

[1] S. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A Survey on Aerial Swarm Robotics," *IEEE Trans. Robot. (T-RO)*, vol. 34, no. 4, pp. 837–855, 2018.
[2] K. N. McGuire, C. D. Wagter, K. Tuyls, H. J. Kappen, and G. C. H. E. de Croon, "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment," *Sci. Robot*, vol. 4, no. 35, p. eaaw9710, 2019.
[3] T. Stirling, J. Roberts, J.-C. Zufferey, and D. Floreano, "Indoor navigation with a swarm of flying robots," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2012, pp. 4641–4647.
[4] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M. Hehn, M. W. Mueller, J. S. Willmann, F. Gramazio, M. Kohler, and R. D'Andrea, "The Flight Assembled Architecture installation: Cooperative construction with flying machines," *IEEE Control Syst.*, vol. 34, no. 4, pp. 46–64, 2014.
[5] A. Tagliabue, M. Kamel, R. Siegwart, and J. Nieto, "Robust collaborative object transportation using multiple MAVs," *Int. J. Robot. Res.*, vol. 38, no. 9, pp. 1020–1044, 2019.
[6] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, 2015.
[7] E. R. Hunt and S. Hauert, "A checklist for safe robot swarms," *Nat. Mach. Intell.*, vol. 2, no. 8, pp. 420–422, 2020.
[8] M. Coppola, K. N. McGuire, C. De Wagter, and G. C. H. E. de Croon, "A Survey on Swarming With Micro Air Vehicles: Fundamental Challenges and Constraints," *Front. Robot. AI*, vol. 7, 2020.
[9] G. Dell'Ariccia, G. Dell'Omo, D. P. Wolfer, and H.-P. Lipp, "Flock flying improves pigeons' homing: GPS track analysis of individual flyers versus small groups," *Anim. Behav.*, vol. 76, no. 4, pp. 1165–1172, 2008.
[10] M. Nagy, Z. Ákos, D. Biro, and T. Vicsek, "Hierarchical group dynamics in pigeon flocks," *Nature*, vol. 464, no. 7290, pp. 890–893, 2010.
[11] I. Couzin, "Collective minds," *Nature*, vol. 445, no. 7129, pp. 715–715, 2007.
[12] M. Yomosa, T. Mizuguchi, G. Vásárhelyi, and M. Nagy, "Coordinated Behaviour in Pigeon Flocks," *PLOS ONE*, vol. 10, no. 10, p. e0140558, 2015.
[13] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," *Annual Conf. Comp. Graph Interactive Technol. (SIGGRAPH)*, vol. 21, pp. 25–43, 1987.
[14] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, "Optimized flocking of autonomous drones in confined environments," *Science Robot.*, 2018.
[15] F. Schilling, F. Schiano, and D. Floreano, "Vision-Based Drone Flocking in Outdoor Environments," *IEEE Robot. Autom. Lett. (RA-L)*, vol. 6, no. 2, pp. 2954–2961, 2021.
[16] S. Hauert, S. Leven, M. Varga, F. Ruini, A. Cangelosi, J.-C. Zufferey, and D. Floreano, "Reynolds Flocking in Reality with Fixed-Wing Robots: Communication Range vs. Maximum Turning Rate," in *IEEE Int. Conf. Intel. Rob. Sys. (IROS)*, 2011.
[17] G. Vasarhelyi, C. Viragh, G. Somorjai, N. Tarcai, T. Szorenyi, T. Nepusz, and T. Vicsek, "Outdoor flocking and formation flight with autonomous aerial robots," in *IEEE Int. Conf. Intel. Rob. Sys. (IROS)*, 2014, pp. 3866–3873.
[18] F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, "Learning Vision-based Flight in Drone Swarms by Imitation," *IEEE Robot. Autom. Lett. (RA-L)*, vol. 4, pp. 4523–4530, 2019.
[19] P. Petracek, V. Walter, T. Baca, and M. Saska, "Bio-inspired compact swarms of unmanned aerial vehicles without communication and external localization," *Bioinspir. Biomim.*, 2020.
[20] E. Soria, F. Schiano, and D. Floreano, "Predictive Control of Aerial Swarms in Cluttered Environments," *Nat. Mach. Intell.*, 2021.
[21] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, "Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control," in *IEEE Int. Conf. Intel. Rob. Sys. (IROS)*, 2017, pp. 236–243.
[22] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online Trajectory Generation With Distributed Model Predictive Control for Multi-Robot Motion Planning," *IEEE Robot. Autom. Lett. (RA-L)*, vol. 5, no. 2, pp. 604–611, 2020.
[23] H. Cheng, Q. Zhu, Z. Liu, T. Xu, and L. Lin, "Decentralized navigation of multiple agents based on ORCA and model predictive control," in *IEEE Int. Conf. Intel. Rob. Sys. (IROS)*, 2017, pp. 3446–3451.
[24] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, Online Collision Avoidance for Dynamic Vehicles Using Buffered Voronoi Cells," *IEEE Robot. Autom. Lett. (RA-L)*, vol. 2, no. 2, pp. 1047–1054, 2017.
[25] B. Şenbaşlar, W. Hönig, and N. Ayanian, "Robust Trajectory Execution for Multi-robot Teams Using Distributed Real-time Replanning," in *Distr. Auton. Rob. Sys.* Springer International, 2019, pp. 167–181.
[26] V. Kungurtsev and M. Diehl, "Sequential quadratic programming methods for parametric nonlinear optimization," *Computational Optimization and Applications*, vol. 59, no. 3, pp. 475–509, 2014.
[27] M. Ballerini, N. Cabibbo, R. Candelier, A. Cavagna, E. Cisbani, I. Giardina, V. Lecomte, A. Orlandi, G. Parisi, A. Procaccini, M. Viale, and V. Zdravkovic, "Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study," *Proc. Natl. Acad. Sci. USA (PNAS)*, vol. 105, no. 4, pp. 1232–1237, 2008.
[28] R. Van Parys and G. Pipeleers, "Online distributed motion planning for multi-vehicle systems," in *2016 European Control Conference (ECC)*. IEEE, 2016, pp. 1580–1585.
[29] L. Hei, J. Nocedal, and R. A. Waltz, "A numerical study of active-set and interior-point methods for bound constrained optimization," in *Modeling, Simulation and Optimization of Complex Processes*. Springer Berlin Heidelberg, 2008, pp. 273–292.
[30] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.
[31] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2017, pp. 3299–3304.
[32] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, "EGO-Swarm: A Fully Autonomous and Decentralized Quadrotor Swarm System in Cluttered Environments," *arXiv*, 2020.