Predictive Control of Aerial Swarms in Cluttered Environments

Enrica Soria (enrica.soria@epfl.ch)
EPFL
Fabrizio Schiano
EPFL
Dario Floreano
Ecole Polytechnique Federale de Lausanne https://orcid.org/0000-0002-5330-4863

Article

Keywords: aerial swarms, cluttered environments, potential fields models, predictive model

Posted Date: September 29th, 2020

DOI: https://doi.org/10.21203/rs.3.rs-82503/v1

Version of Record: A version of this preprint was published at Nature Machine Intelligence on May 17th, 2021. See the published version at https://doi.org/10.1038/s42256-021-00341-y.

Predictive Control of Aerial Swarms in Cluttered Environments

Enrica Soria*, Fabrizio Schiano, and Dario Floreano[1]

Abstract. Classical models of aerial swarms often describe global coordinated motion as the combination of local interactions that happen at the individual level. Mathematically, these interactions are represented with Potential Fields. Despite their explanatory success, these models fail to guarantee rapid and safe collective motion when applied to aerial robotic swarms flying in cluttered environments of the real world, such as forests and urban areas. Moreover, these models necessitate a tight coupling with the deployment scenarios to induce consistent swarm behaviors. Here, we propose a predictive model that combines the local principles of potential field models with the knowledge of the agents' dynamics. We show that our approach improves the speed, order, and safety of the swarm, it is independent of the environment layout, and scalable in the swarm speed and inter-agent distance. Our model is validated with a swarm of five quadrotors that can successfully navigate in a real-world indoor environment populated with obstacles.

1 Introduction

From the fluid wavelike movements of starling flocks to the swift turning maneuvers of bee swarms, nature displays many examples of coordinated flight [1]–[7]. Recent progress in aerial robotics technologies led to the availability of smart drones at the price of smartphones [8], but the deployment of drone swarms that autonomously coordinate their local trajectories remains a challenge. Drone swarms can offer larger area coverage than a single drone for monitoring and exploration missions [9], [10], and they can collect multi-dimensional sensory data by flying a diverse set of sensors [11]. Autonomous aerial swarms can also enable functionalities that are beyond the capabilities of a single drone, such as cooperative transportation of large objects and aerial construction [12], [13]. Hundreds of drones have been deployed in aerial light shows by companies such as Intel [14], Ehang [15], and Verity Studios [16], but in those circumstances, every drone is individually controlled by a central computer to follow a precomputed trajectory. Instead, the coordinated, synchronized motion of biological swarms is a self-organized behavior that emerges from local information[4]–[6], [17]–[19], and can thus cope with unforeseen situations, such as flying through forests or in urban canyons.

Early work suggested that the collective motion of a biological swarm can be described by the combination of three behavioral rules that apply to each agent simultaneously [20]. These rules consist of (a) *cohesion,* which brings each agent closer to its neighbors, (b) *repulsion,* which drives each agent away from its neighbors to avoid collisions, and (c) *alignment,* which steers each agent towards the average heading of its neighbors. In goal-directed flight, alignment is replaced by *migration*, which steers each agent in a preferred migration direction [21], [22]. For

[1] The authors are from the Laboratory of Intelligent Systems (LIS), École Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland.
* Corresponding author: enrica.soria@epfl.ch

navigating environments with obstacles, the addition of a fourth rule, *collision avoidance*, is necessary to steer the agents around the obstacles [20], [23], [24]. Mathematically, these rules can be modeled by virtual forces exerted by the agents on their neighbors and are associated with Potential Fields (PFs), i.e., vector fields describing how forces act at various positions in space. PFs encode the desired behaviors of the swarm. They regulate the inter-agent distance among neighboring individuals similarly to a spring-mass system, adjust the velocity of the agents, steer them towards a common direction, and regulate their distance to obstacles [23].

The advantage of PF swarm models is that they are purely reactive, meaning that their decisions are solely based on the current sensory information and thus have low computational complexity [20], [23]. For this reason, PF models are convenient for the implementation on real robotic systems, either in free environments [21], [25], or in environments with convex obstacles [24]. In the latter case, collision avoidance is obtained by defining virtual repulsive agents (called shill agents) located along the obstacles' boundaries. However, these shill agents present the inconvenience of slowing down the swarm as it approaches the obstacles [20], [26]. This effect becomes prominent in environments with high obstacle densities, where PF swarms can significantly slow down. The slowdown can be attenuated by weakening the repulsion potentials, albeit at the expense of the swam safety, because some agents may collide. Moreover, to account for the idiosyncrasies of the real world, these models often include a significant number of parameters that have complex interdependencies [2], [24]. As a consequence, they often require the adoption of optimization techniques such as evolutionary algorithms, to identify a viable instantiation of the parameters, and each instantiation is specific to the swarm's preferred speed and inter-agent distance and to the environmental layout [21], [24], [27].

Here we propose a method to remove those difficulties that consists of endowing swarming agents with prediction-based control. Specifically, we show that aerial swarms with predictive control display faster flight while guaranteeing safe navigation in cluttered environments, they can adapt to diverse obstacle densities, and they are scalable to changes in the inter-agent distance and swarm's speed. It has been recently advocated that some form of predictive control, in the form of an internal model of the actions of their conspecifics, may also be leveraged by biological swarms where the apparent synchronization of coordinated maneuvers, such as a flock of starlings or a school of fish, cannot be explained by a purely reactive system [19]. Inspired by this hypothesis, the method proposed in this paper endows flying agents with a model of swarm behavior based on Nonlinear Model Predictive Control (NMPC).

Model Predictive Control (MPC) is a method that computes the control action of a system as the solution of a constrained optimization problem [28], [29]. MPC leverages a mathematical representation of the system to predict and optimize its future behavior in an iterative process. Differently from PF control, MPC can explicitly handle constraints, such as physical limitations (e.g., flight speed and acceleration ranges of a drone) [30]–[32], and environmental restrictions (e.g., no-flight zones) [32]–[34]. However, the recursive online solution of constrained optimization problems is associated with higher computational costs, and therefore the adoption of predictive controllers in robotics has spread only recently [35].

MPC has shown promising results in simulation on multi-vehicle systems. Examples include the stabilization of multiple agents in obstacle-free environments [36], [37], in the presence of obstacles [33], and the generation of collision-free trajectories for groups of robots with known target locations [38]–[40]. NMPC is a variant of MPC that can handle the nonlinearities of a system or its constraints [29]. This advantage comes at the cost of being more computationally demanding. In simulation, NMPC has been used to control leader-follower formations of drones without obstacles [41], and to control 2D quadrotor formations in the presence of convex obstacles [34].

Less work has been done on the use of MPC with multiple real drones, notably due to the difficulty of real-time implementation. Linear MPC has been used for trajectory planning in the presence of virtual obstacles in a leader-follower configuration, where a drone (the follower) has to keep a constant distance from a virtual agent (the leader), [42]. However, in leader-follower approaches, the leader has the extra knowledge of the group trajectory, which is either preprogrammed or provided by an external source. This aspect introduces an asymmetry in the agents' roles and adds a single point of failure in the swarm [43]. MPC has been used for the online generation of collision-free trajectories for a group of drones in environments with obstacles, where every drone is individually assigned an initial position and a target destination [32]. Instead, the model presented here is meant to coordinate the navigation of the swarm as a unique entity and guarantee internal order, in lieu of generating the trajectories separately. Concurrently, we avoid imposing a rigid formation or a fixed topology to the swarm, which may impact the freedom and fluidity of the agents' movements. Finally, NMPC has been shown to be capable of dealing with non-convex collision avoidance constraints in real multi-drone systems when the agents are assigned intersecting paths, although they were flying in empty environments [44].

In the proposed NMPC model, the objective to be optimized is made of three components inspired from PF swarm models: (a) *separation*, which drives the inter-agent distances to a preferred value, (b) *navigation*, which makes the agents' speed approximate a preferred value, and (c) *direction*, which steers the swarm along a preferred direction. A fourth rule, (d) *control effort* is added to minimize the agents' accelerations, thereby smoothing flight trajectories and increasing energy efficiency. Each drone regulates its flight based on the knowledge of its neighbors and its own state and predicts its own trajectory and those of its neighbors thanks to a linearized dynamical model. The proposed NMPC model integrates a set of constraints to ensure safety distances among drones and with obstacles. We compare our NMPC model to a PF model and show that predictive controllers can safely fly the swarm in cluttered environments while significantly increasing the flight speed and synchronization of the swarm. Also, we show that the performance of the proposed NMPC model is independent of the obstacle density and environmental layout, differently from PF models. Additionally, we test the scalability of the proposed model to variations of desired inter-agent distance and swarm speed. We perform systematic experiments in simulation and validate the results with a swarm of five palm-sized quadrotors.

2 Results

For the performance assessment of the swarm models, we set up a forest-like environment that consists of a rectangular flight region populated with cylindrical obstacles (Fig. 1a). At the experiment onset, we place five drones at random positions within a predefined start area on one side of the region (Fig. 1a, red zone) and let the swarm fly through the region along the migration direction (Fig. 1a, orange arrow). The mission is completed when all drones cross the arrival plane (Fig. 1a, orange plane) on the opposite side of the region.

We assess the quality of the aerial swarm's flight considering eight different metrics. The mission completion time T measures the time that the swarm requires to cross the region. The inter-agent distance error E_d measures the deviation of the distances that the drones maintain from each other from the preferred distance d_{ref}, and the inter-agent distance range R_d measures the range in which the inter-agent distances vary (defined by the minimum and maximum inter-agent distance over time). The speed error E_v measures the deviation of the agents' speeds from the preferred migration speed v_{ref}, and the speed range R_v measures the range in which the agents' speeds vary. E_d, R_d, E_v and R_v take values greater than or equal to 0 (ideal case). We determine the swarm's level of synchronization by calculating the directional correlation of the agents' movements, expressed by the so-called order Φ_{order}. Φ_{order} takes values between -1 (complete disorder) and 1 (perfect order). Finally, the agent-agent safety $\Phi_{\mathrm{agent-safety}}$ assesses the ability of the swarm's agents to avoid collisions among themselves, and the agent-obstacle safety $\Phi_{\mathrm{obs-safety}}$ assesses the ability of the agents to avoid collisions with the obstacles. $\Phi_{\mathrm{agent-safety}}$ and $\Phi_{\mathrm{obs-safety}}$ take values between 0 (complete unsafety) and 1 (perfect safety, i.e., zero collisions) (see Supplementary Table 1 for mathematical formulation). To evaluate the overall performance of the swarm during a mission, we compute the average and standard deviation of these metrics. For the instantaneous evaluation of the swarm over time, we additionally plot the inter-agent distance and speed, and the distance to obstacles, from which we can appreciate their respective errors and ranges, and the occurrence of collisions.

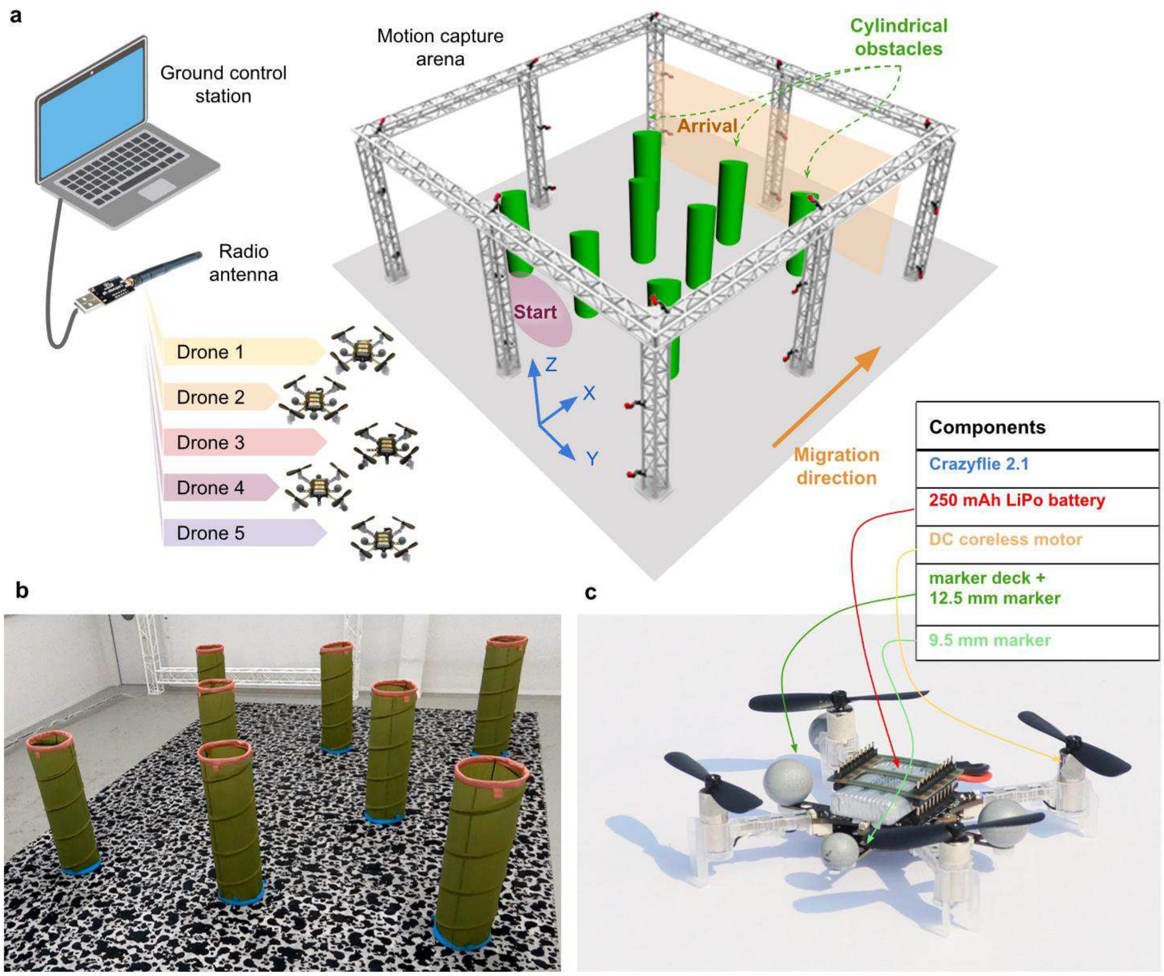

Fig. 1: **Experimental setup of drone swarm flight in cluttered environments. (a)** Illustration of the experimental setup and the environment configuration. A ground control station, equipped with a radio transmitter, computes and sends run-time control commands to the drones. The swarm flies in the 3D space of an indoor flying arena. The drones take off from initial random positions within a predefined start area (red zone). Drones swarm along the preferred migration direction (orange arrow). The mission is completed when all drones cross the arrival plane (Fig. 1a) on the opposite side of the region. **(b)** Indoor test environment populated with cylindrical obstacles. **(c)** Components of the drones used for the hardware experiments.

We extensively tested the proposed NMPC swarm model in simulation and compared it to a reactive PF model that has been recently described and validated on 30 real drones [24]. In addition to the repulsion and obstacle avoidance rules, the PF model includes a *friction* rule to reduce velocity oscillations. In order to ensure cohesive goal-directed flight in open environments, we added the rules of *cohesion* and *migration* to the PF model. As in previous work [21], [24], [27], we used evolutionary optimization to search the large parameter space of the PF swarm model, and favored swarms with highly ordered flight ($\Phi_{\text{order}} = 1$) and a low number of agent-agent and agent-obstacle collisions ($\Phi_{\text{agent-safety}} = 1, \Phi_{\text{obs-safety}} = 1$) (see

Supplementary Table 3). The purpose of the experimental comparison between NMPC swarming and PF swarming is to emphasize behavioral differences and performance advantages of the proposed NMPC swarm model. However, the choice of a swarm model for the deployment on physical drones should also consider computational resources, which are significantly larger for NMPC swarming.

Below we present three sets of simulation experiments: (i) we compare the performance metrics of the two models in the same environmental conditions, (ii) we investigate the adaptability of the PF and NMPC swarm models to environments with different obstacle density, and (iii) we study the scalability of the NMPC swarm model at different preferred speeds and inter-drone distances. Finally, we experimentally validate the NMPC swarm model with five palm-sized drones (Fig. 1c) flying through a room with cylindrical obstacles (Fig. 1b).

2.1 Comparison of PF and NMPC aerial swarms

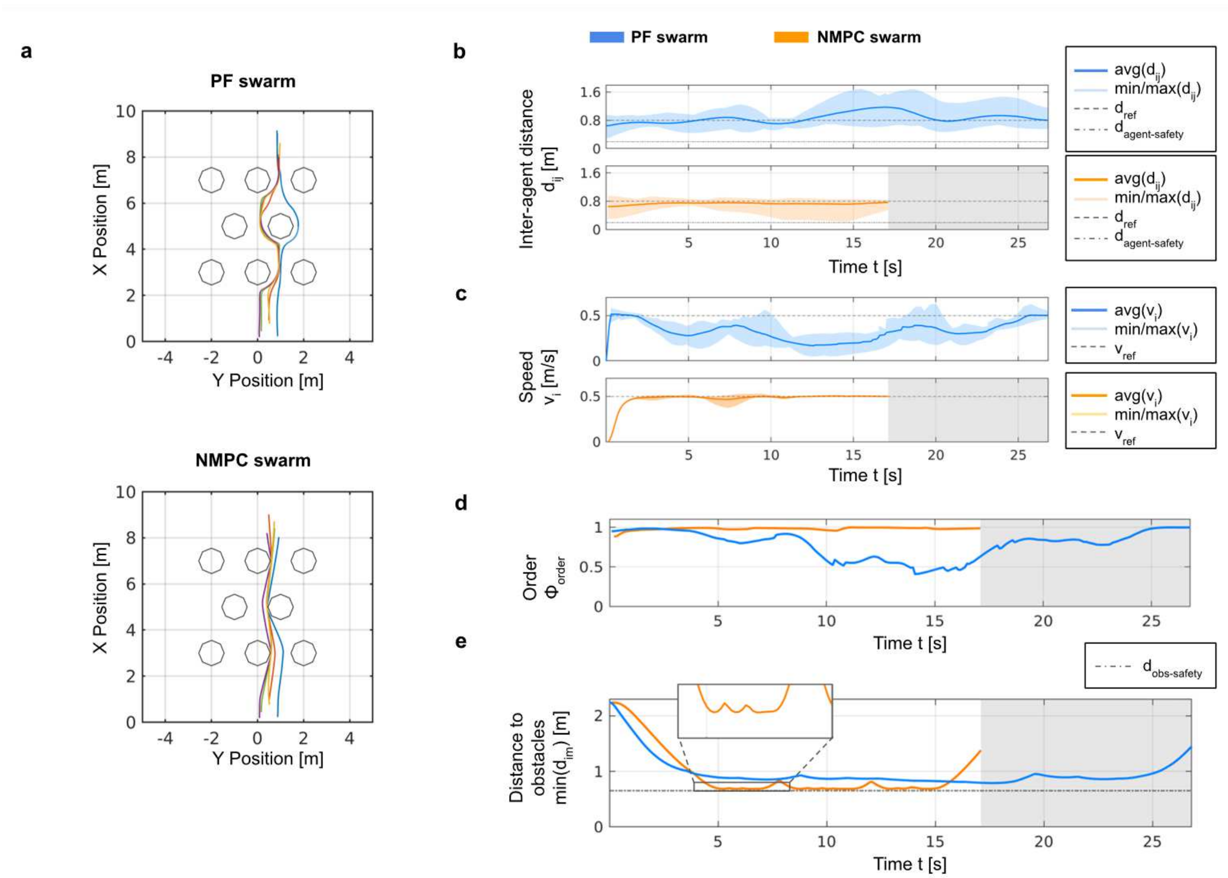

Fig. 2: **Comparison of the PF and NMPC aerial swarms in simulation experiments**. **(a)** Top views of the 3D trajectories of five drones flying in a cluttered environment with the PF (top) and the NMPC models (bottom) (see Supplementary Video 1). The circular objects on the map correspond to cylindrical obstacles. **(b)** Inter-agent distance average (solid line) and range (shaded region). The curve on top (blue) refers to the PF swarm, while the one at the bottom

(orange) refers to the NMPC swarm. **(c)** Swarm speed average (solid line) and range (shaded region). **(d)** Order metric. **(e)** Distance to obstacles, $\min(d_{im})$, expressed as the minimum distance between the swarm's agents and the set of obstacles.

Both PF and NMPC swarms navigated around the obstacles without collisions (Fig. 2e), but the NMPC swarm completed the mission 57% faster than the PF swarm. The reduced mission time is due to the ability of the NMPC swarm to track the preferred speed v_{ref} more consistently (E_v= 0.02 ± 0.02, R_v= 0.08$\pm$0.07) than PF swarm (E_v=0.39 $\pm$ 0.15, R_v= 0.47 ± 0.15) (Fig. 2c). The NMPC swarm also generated a smaller inter-agent distance error ($E_d = 0.11 \pm 0.02$) and range ($R_d = 0.55 \pm 0.18$) compared to the PF swarm ($E_d = 0.26 \pm 0.15$, R_d=0.90 $\pm$ 0.26) (Fig. 2b). The NMPC model generated almost perfectly ordered flight maneuvers throughout the entire flight ($\Phi_{\mathrm{order}} = 0.98 \pm 0.02$) while the PF model displayed lower and more variable order ($\Phi_{\mathrm{order}} = 0.78 \pm 0.17$) (Fig. 2d). Neither the NMPC nor the PF swarm presented agent-agent or agent-obstacle collisions ($\Phi_{\mathrm{agent-safety}} = 1 \pm 0, \Phi_{\mathrm{obs-safety}} = 1 \pm 0$) (Fig. 2e). While optimizing the swarm's objectives, the NMPC model reduced the minimum distance to obstacles of $0.03\ m$. In comparison, the PF swarm achieved a minimum distance to obstacles of 0.14 m. This difference is due to the fact that in the PF model the obstacles apply a repulsion force on the agents' in their proximity, while in the NMPC model there is no penalty for approaching the obstacles. As a consequence, when implementing the NMPC model on a real-world swarm, the user should carefully choose a safety margin.

2.2 Environments with different obstacle densities

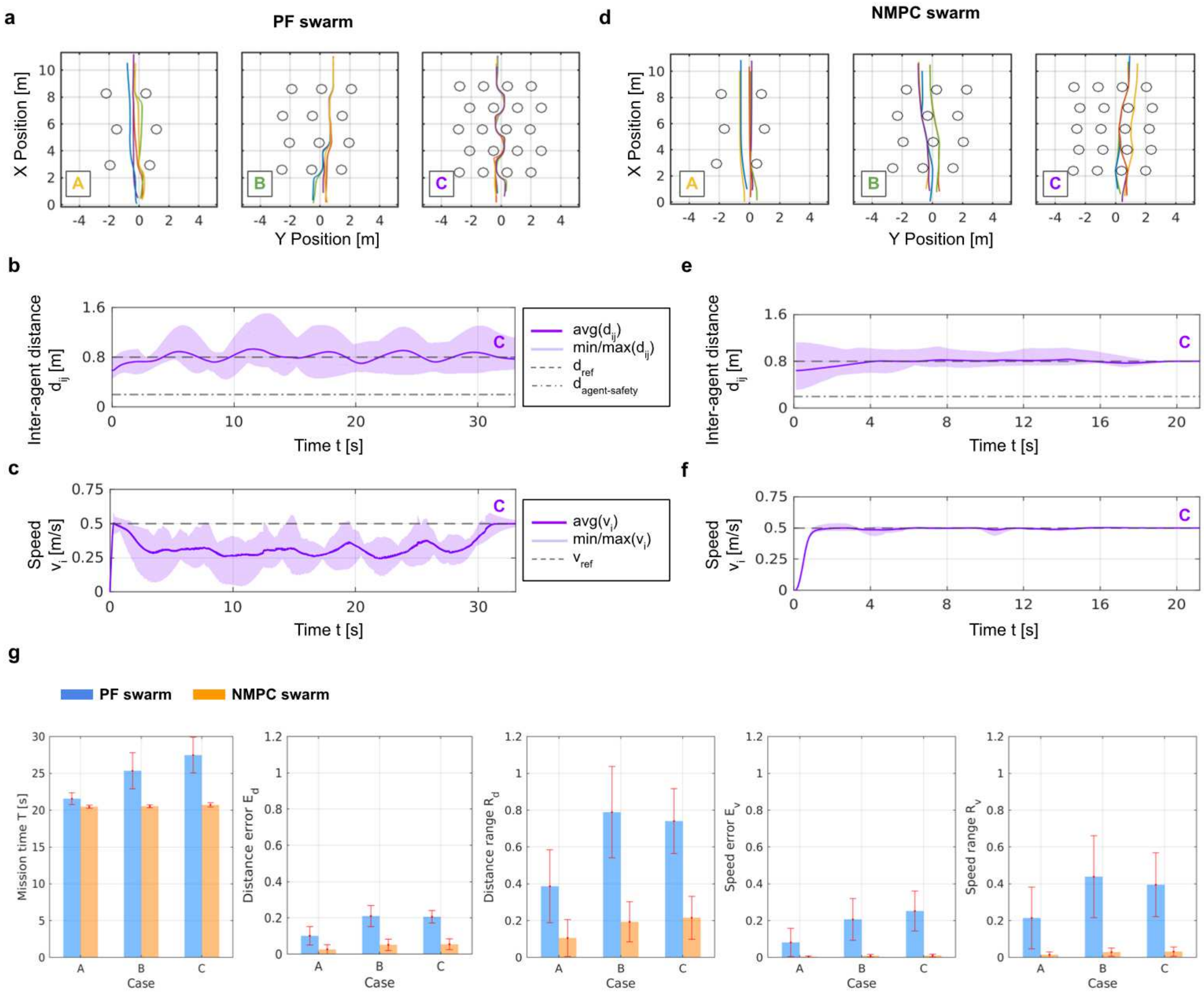

Fig. 3: **Comparison of the PF and NMPC swarm deployment in environments with different obstacle densities**. **(a, d)** Top views of the 3D simulated trajectories of the PF and the NMPC swarms in environments with three different obstacle densities. The density increases from left to right (*Case A*: 0.06, *B*: 0.12, and *C*: 0.20) (see Supplementary Video 1). **(b, c)** Inter-agent distance and speed of the PF swarm in *Case C*. **(e, f)** Inter-agent distance and speed of the NMPC swarm in *Case C*. **(g)** Aggregated results (average and standard deviation) of 10 stochastic simulations of the PF (blue) and NMPC (orange) swarm models in *Cases A, B,* and *C*. The represented metrics are the mission time T, the distance error E_d, the distance range R_d, the speed error E_v, and the speed range R_v (see Supplementary Table 1).

Parameter	Unit	Description	Value
d_{ref}	m	Preferred (or reference) value for the inter-agent distance	0.8
v_{ref}	m/s	Preferred (or reference) value for the swarm speed	0.5
$\boldsymbol{u}_{\mathrm{ref}}$	-	Preferred migration direction	(1 0 0)

L_{map}	m	Length of an edge of the square flight region (or map)	10
r_{obs}	m	Obstacles radius	0.35
ρ_{obs}	$-/m^2$	Obstacle density	Case A: 0.06 Case B: 0.12 Case C: 0.20

Table 1: **Swarm and environment configurations of the simulation experiments with different obstacle densities.** The same configurations are used for both the PF and the NMPC swarm models.

We tested the PF and the NMPC swarm models for three different obstacle densities (Case A: 0.06, B: 0.12, and C: 0.20) to quantify the impact on the swarms' performance. The obstacles occupy random positions on the map, but they have a homogenous distribution (Fig. 3a and 3d). The initial positions of the drones are random. For both swarm models, we show the evolution of the inter-agent distance and speed for the scenario with the highest obstacle density (Case C). The results show that the inter-agent distance error is smaller with NMPC swarms ($E_d = 0.11 \pm 0.02$) than with PF swarms ($E_d = 0.27 \pm 0.12$), and the inter-agent distance range is shorter for NMPC swarms ($R_d = 0.56 \pm 0.18$) than with PF swarms ($R_d = 0.90 \pm 0.26$,). The NMPC swarms tracked the preferred speed v_{ref} more precisely ($E_v = 0.03 \pm 0.02$) than the PF swarms ($E_v = 0.39 \pm 0.15$), and the speed range was shorter ($R_v = 0.08 \pm 0.07$ and 0.47 ± 0.15, respectively). The faster speed of NMPC swarms resulted in faster mission completion time than the PF swarms ($T = 21.5$ s and 34.1 s, respectively).

To assess the reproducibility of the results, we performed ten stochastic simulations for each of the three obstacle densities and for the two swarm models, and we report here aggregated performance results (Fig. 2g). While the speed error in the NMPC swarm is small and constant for all obstacle densities (Case A: $\mathrm{E_v} = 0.01 \pm 0.01$, B: 0.01 ± 0.01, C: 0.01 ± 0.01), it is larger and increases with larger obstacles densities in the PF swarm (Case A: $\mathrm{E_v} = 0.08 \pm 0.07$, B: 0.21 ± 0.11, C: 0.25 ± 0.11). As a consequence, the mission completion time of the PF swarm is increased when increasing the obstacle density (Case A: $T = 21.56 \pm 0.81$ s, *B*: 25.35 ± 2.46 s, *C*: 27.48 ± 2.43 s), while for the NMPC swarm it is shorter and it stays almost constant across the different densities (Case A: $T = 20.47 \pm 0.22$ s, B: 20.54 ± 0.21 s, C: 20.72 ± 0.28 s). Also the PF swarm's order deteriorates when increasing the obstacle density (Case A: $\Phi_{\text{order}} = 0.98 \pm 0.03$, B: 0.92 ± 0.08, C: 0.81 ± 0.08), while for the NMPC swarm it stays almost constant (Case A: $\Phi_{\text{order}} = 0.99 \pm 0.01$, B: 0.98 ± 0.02, C: 0.98 ± 0.02). While the NMPC swarm produces collision-free movements in all cases, for the PF swarm we observe some agent-obstacle collisions at high obstacle densities (Case A: $\Phi_{\text{obs-safety}} = 1 \pm 0$, B: $(99.98 \pm 0.06)\ 10^{-2}$, C: $(99.99 \pm 0.02)\ 10^{-2}$). The aggregated performance results are summarized in Supplementary Table 5.

2.3 Scalability to different inter-agent distances and speeds

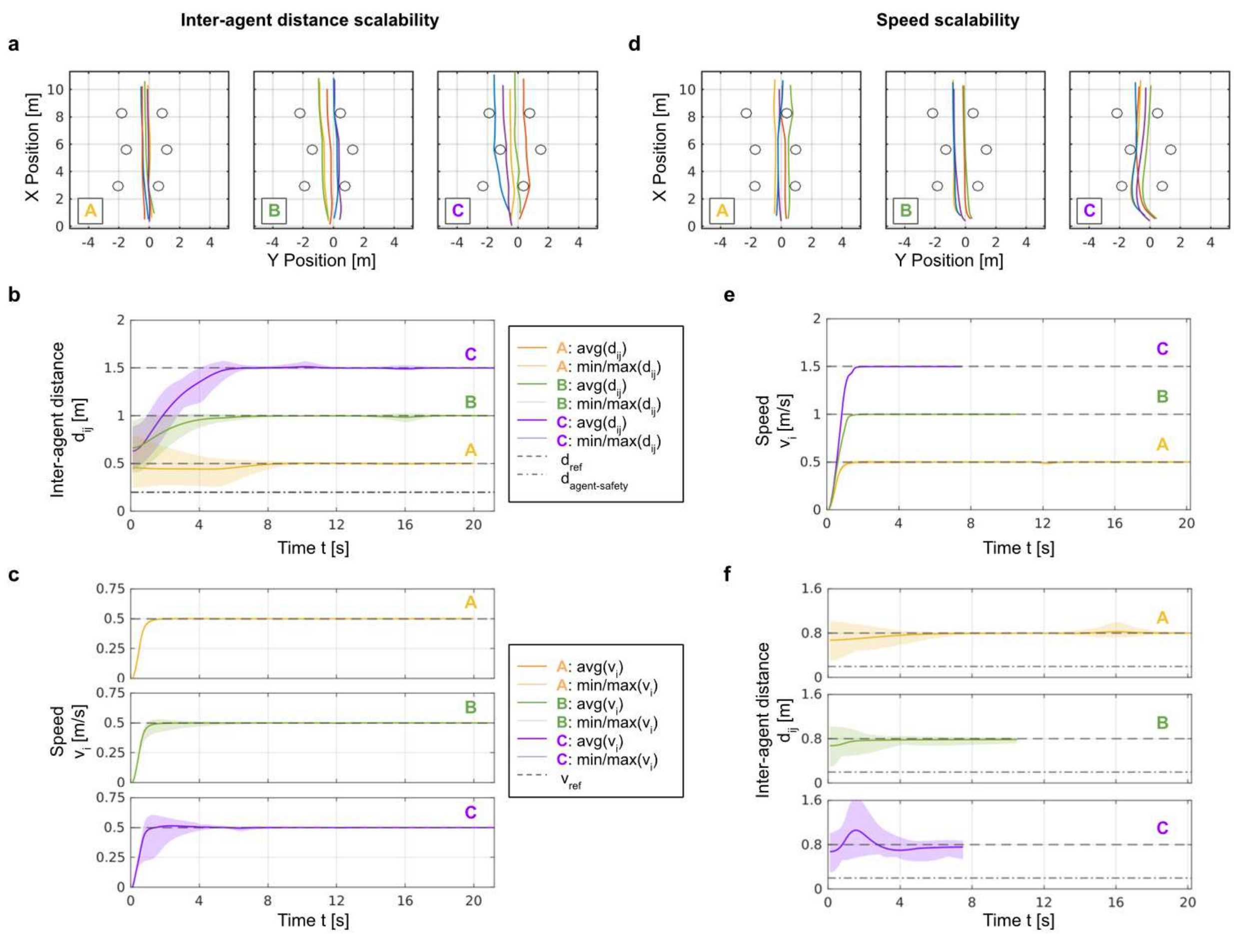

Fig. 4: **Scalability of the NMPC swarm in inter-agent distance and speed**. On the left, simulation results on the scalability of the NMPC swarm model in the inter-agent distance for three preferred distance values (Case A: $d_{\text{ref}} = 0.5m$, B: $1.0m$, and C: $1.5m$). On the right, simulation results on the scalability in the swarm speed for three preferred speed values (Case A: $v_{\text{ref}} = 0.5m/s$, B: $1.0m/s$, and C: $1.5m/s$). **(a, d)** Top views of the 3D trajectories of the swarm (see Supplementary Video 1). **(b, c)** Inter-agent distance and speed for the experiment on the inter-agent distance scalability. **(e, f)** Inter-agent distance and speed for the experiment on the speed scalability. The obstacle size and density are the same for the six cases.

We assess the scalability of the proposed NMPC model to different values of the preferred inter-agent distance (Case A: $d_{\text{ref}} = 0.5\ m$, B: $1.0\ m$, and C: $1.5\ m$, see Fig 4 a-c) and speed (Case A: $v_{\text{ref}} = 0.5\ m/s$, B: $1.0\ m/s$, and C: $1.5\ m/s$, see Fig 4 d-f) in the same environmental conditions. We analyze the swarm's inter-agent distance and speed and quantify their respective errors and ranges. The results show that at different inter-agent distance levels the swarm inter-agent distance converged to the preferred value with comparable errors (Case A: $E_d = 0.05 \pm 0.06$, B: 0.01 ± 0.02, C: 0.02 ± 0.03, see Fig. 4b). The swarm's speed error is almost zero in the three cases (see Fig. 4c), and it resulted in similar mission times (Case A: $\mathrm{T} = 20\ s$, B: $21\ s$, and C: $21.2\ s$). We did not observe collisions. Regarding the experiments on the scalability in speed, the speed error E_v was close to zero in the three cases (Fig. 4e). However, the variability of the inter-agent distance in Case C is higher ($R_d = 0.46 \pm 0.05$) than in Cases A ($R_d = 0.13 \pm 0.11$) and B ($R_d = 0.19 \pm 0.03$) (Fig. 4f). Indeed, when the agents turn around the obstacle in the middle of the scene, they rearrange and increase their distance. Also in these experiments, we did not observe collisions. Comparative results on the PF swarm are in Supplementary Fig. 1. Aggregate results of stochastic simulations for each of the preferred inter-agent distance and speed values, and for both the PF and the NMPC models are in Supplementary Fig. 2, and in Supplementary Tables 6 and 7.

2.4 Validation with real drones

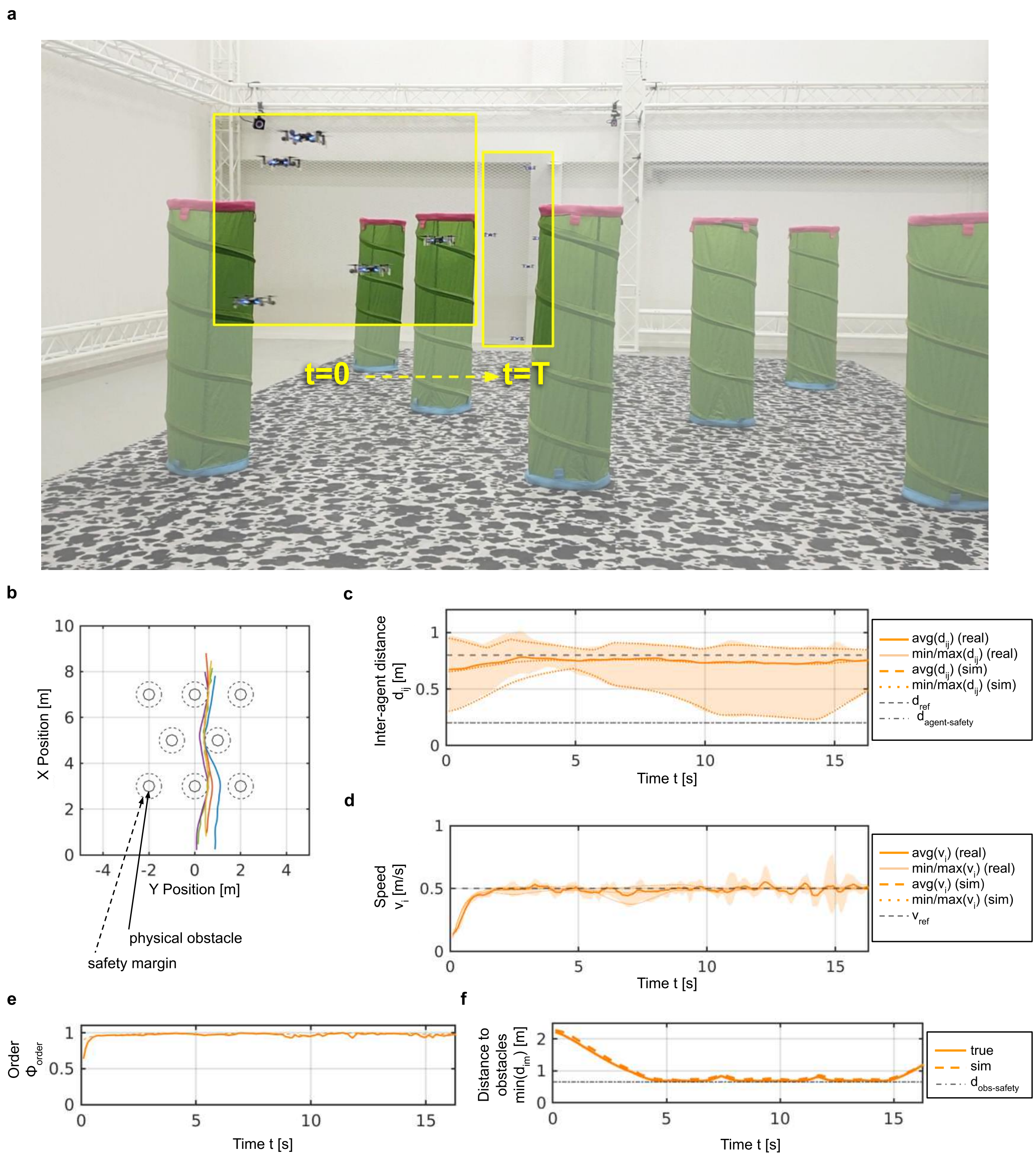

Fig. 5: **Real-world experiment of the NMPC swarm. (a)** The swarm, composed of five commercial palm-sized quadrotors, flies through cylindrical obstacles in a motion capture arena. The swarm crosses the region from the foreground ($t = 0\ s$) to the background ($t = T$), while maintaining cohesion and avoiding the obstacles (see Supplementary Video 2). **(b)** Top view of the trajectories of the drones. For the real-world deployment, we selected obstacles with a smaller radius ($r_{\text{obs}} = 0.30\ m$) than in simulation ($r_{\text{obs}} = 0.55\ m$), but we used the same safety distance for collision avoidance as in simulation ($d_{\text{obs-safety}} = 0.65\ m$), which introduces a

safety margin of $0.25\ m$ from the physical obstacles (see Supplementary Table 4). **(c)** Average inter-agent distance and range with the real swarm (solid line and shaded region, respectively) and the simulated swarm (dashed and dotted lines, respectively). **(d)** Average speed and range with the real swarm (solid line and shaded region, respectively) and the simulated swarm (dashed and dotted lines, respectively). **(e)** Swarm's order: real (solid line) and simulated (dashed line) swarm. **(f)** Swarm distance to obstacles. The offset in the real data (solid line) with respect to the simulated data (dashed line) is due to the safety margin.

We validated the NMPC swarm on five commercial quadrotors in an indoor motion capture arena where we reconstructed the environment described in Sec. 2.1 (Fig. 5a). We measured the real flight performance, and we compared them with the simulation performance. The real drones achieve the preferred inter-agent distance $d_{\mathrm{ref}} = 0.8\ m$ with an error (E_d= 0.12$\pm$0.02) comparable to the simulation error (E_d= 0.11$\pm$0.02) (Fig. 5c). However, the speed error is slightly higher ($E_v = 0.07 \pm 0.03$) than in simulation ($E_v = 0.02 \pm 0.02$) (Fig. 5d). The higher speed error in the real swarm can be explained by small communication delays and air turbulence due to the proximity of the drones to each other and obstacles. The order of the real swarm ($\Phi_{\mathrm{order}} = 0.97 \pm 0.04$) is comparable to the simulated swarm ($\Phi_{\mathrm{order}} = 0.98 \pm 0.02$) (Fig. 5e), and in both cases we did not observe collisions ($\Phi_{\mathrm{agent-safety}} = 1 \pm 0, \Phi_{\mathrm{obs-safety}} = 1 \pm 0$) (Fig. 5f).

3 Discussion

This article shows that a Nonlinear Model Predictive Control (NMPC) model achieves a faster and more synchronized flight in cluttered environments as compared to state-of-the-art models based on potential fields (PFs). NMPC swarms report no collisions in cluttered environment, they better attain and maintain target speeds, and they remain more ordered and cohesive. The benefits brought by predictive controllers to robotic aerial swarms confirm a parallel with biological systems, where individuals are thought to enhance their synchronization by future state projection [19].

In robotics, the advantages of the NMPC method are promising for applications that require navigation in crowded scenarios, such as the exploration of urban environments, collapsed buildings, or forests [45], [46]. Also vision-based swarms could benefit from all these features since the reliability of reciprocal visual detection of the drones strongly depends on their distance, and NMPC swarms showed that they can better maintain target inter-agent distances [22], [47]. Overall, predictive methods can improve the autonomy of swarm operations as well as the safety of the swarm and the environment, which are both essential elements to build public confidence in the use of swarms [48].

For our experiments, we relied on a central computing node that generates the motion of the agents at run time according to local interactions only. This assumption simplifies the implementation since it requires only one computer, acting as a ground control station, instead of several onboard computers that the agents would carry. However, the NMPC model requires a higher amount of computational resources than the PF model, and scale worse with the swarm size. It will be interesting to develop a decentralized NMPC model where the computational costs are independent of the number of agents. Work in this direction will allow to scale our approach to swarms of larger size.

Finally, our results motivate future works to address research questions in the design of robust swarm models in dynamic environments. Thanks to their recursive structure, MPC controllers offer a promising method to allow navigation in scenarios with moving obstacles. However, a generalization of the proposed model to dynamic environments would require theoretical and numerical investigation on the conditions for stability, as well as reliable estimation of the obstacles' motion [49].

4 Methods

In this work, we consider a swarm of N agents labeled by $i \in \{1, \dots, N\}$. The position, velocity, and control input of the i-th agent are denoted by $\boldsymbol{p}_i, \boldsymbol{v}_i, \boldsymbol{u}_i \in \mathbb{R}^3$, respectively. Let $d_{ij} = \| \boldsymbol{p}_j - \boldsymbol{p}_i \|$ represent the distance between the center of two agents i and j, where $\|\cdot\|$ denotes the Euclidean norm. We model the swarm with a directed sensing graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertex set $\mathcal{V} = \{1, \dots, N\}$ represents the agents, and the edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ contains the pairs of agents $(i, j) \in \mathcal{E}$ for which agent i can sense agent j. We denote as $\mathcal{N}_i = \{j \in \mathcal{V} | \, (i, j) \in \mathcal{E}\} \subset \mathcal{V}$ the set of neighbors of an agent i in $\mathcal{G}$, and $| \cdot |$ indicates the cardinality of a set. To keep the $|\mathcal{N}_i|$ constant, we define the neighbors set utilizing a topological distance, a reasonable hypothesis also for natural systems [7]. Therefore, the set $\mathcal{N}_i$ contains the $|\mathcal{N}_i|$ nearest neighbors of agent i. To reproduce a forest-like environment, we introduce M cylindrical obstacles labeled by $m \in \{1, \dots, M\}$. We denote as d_{im} the distance between an agent i and the symmetry axis of cylinder m. In our simulations, the dynamics of the agents is reproduced in discrete time. We let $\boldsymbol{p}_i(k)$, $\boldsymbol{v}_i(k)$, $\boldsymbol{u}_i(k) \in \mathbb{R}^3$ be the position, velocity, and control input of the i-th agent at the time $t(k) = k \; dt$, respectively.

4.1 PF swarm model

The PF model we present is inspired by a state-of-the-art model that allows drone swarm navigation in confined environments [24]. From the original model, we include the rule of *repulsion* to prevent inter-drone collisions, *friction* to reduce velocity oscillations, and *obstacle avoidance* to avoid collisions with obstacles. For the mathematical definition of these rules, we refer the reader to [24]. To ensure goal-directed flight in open environments, we added two rules: *migration* to provide a preferred velocity vector, and *cohesion* to keep agents together. We denote the migration velocity with $\boldsymbol{v}_{\mathrm{ref}} = v_{\mathrm{ref}} \boldsymbol{u}_{\mathrm{ref}}$, where v_{ref} is the preferred speed and $\boldsymbol{u}_{\mathrm{ref}}$ is the preferred direction. Then, the migration term, equal for every agent, corresponds to:

$$\boldsymbol{v}_{\mathrm{mig}} = v_{\mathrm{ref}} \boldsymbol{u}_{\mathrm{ref}} \tag{1}$$

If the repulsion is active when neighboring agents are closer than the preferred distance d_{ref} and push them further apart, the cohesion is active when they are father than d_{ref} to bring them closer. Repulsion and cohesion are inactive when two agents are precisely at the distance d_{ref}. The cohesion exerted on an agent i from a neighbor j is:

$$\boldsymbol{v}_{\mathrm{coh},ij} = \begin{cases} c_{\mathrm{coh}}(d_{ij} - d_{\mathrm{ref}}) \frac{\boldsymbol{p}_j - \boldsymbol{p}_i}{d_{ij}} & \text{if } d_{\mathrm{ij}} < d_{\mathrm{ref}} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where we choose the pairwise gain of cohesion equal to the repulsion gain $c_{\mathrm{coh}} = c_{\mathrm{rep}}$ and the cutoff for the minimum cohesion range equal to the repulsion range d_{ref}. The total cohesion effect calculated for agent i with respect to its neighbors is:

$$\boldsymbol{v}_{\mathrm{coh},i} = \sum_{j \in \mathcal{N}_i} \boldsymbol{v}_{\mathrm{coh},ij} \tag{3}$$

At any instant, the velocity for agent i resulting from the contributions above is:

$$\widetilde{\boldsymbol{v}}_i = \boldsymbol{v}_{\mathrm{mig}} + \boldsymbol{v}_{\mathrm{coh},i} + \boldsymbol{v}_{\mathrm{rep},i} + \boldsymbol{v}_{\mathrm{fric},i} + \sum_{s \in M_i} \boldsymbol{v}_{\mathrm{obstacle},is} \tag{4}$$

After summing the contributions, we apply a cutoff on the acceleration at $a_{\max}$ according to:

$$\boldsymbol{a}_i = \frac{\widetilde{\boldsymbol{a}}_i}{\|\widetilde{\boldsymbol{a}}_i\|} \min(\|\widetilde{\boldsymbol{a}}_i\|, a_{\max}) \tag{5}$$

where $\widetilde{\boldsymbol{a}}_i(k+1) = (\widetilde{\boldsymbol{v}}_i\,(k+1) - \widetilde{\boldsymbol{v}}_i(k))/dt$. Then, we apply a cutoff on the speed at $v_{\max}$, and get the velocity command $\boldsymbol{v}_i$ of the i-th agent:

$$\boldsymbol{v}_i = \frac{\widetilde{\boldsymbol{v}}_i}{\|\widetilde{\boldsymbol{v}}_i\|} \min(\|\widetilde{\boldsymbol{v}}_i\|, v_{\max}) \tag{6}$$

To search the large parameter space of the PF swarm model, we used evolutionary optimization for highest-order flight and lowest number of collisions. The evaluation of the swarm behavior is based on a single fitness that sums three independent values (Φ_{order}, $\Phi_{\mathrm{agent-safety}}$, and $\Phi_{\mathrm{obs-safety}}$) smaller or equal to 1 (ideal case). The fitness is determined in simulation where the swarm initialized with random positions in an environment where obstacles are randomly placed. The parameter values and their description are detailed in the Supplementary Materials.

4.2 Agents' dynamics

The NMPC swarm model supposes the availability of the agents' dynamic model. We assume that every drone of the swarm obeys a discrete linear system, given by:

$$\boldsymbol{x}_i(k+1) = A_i \boldsymbol{x}_i(k) + B_i \boldsymbol{u}_i(k) \tag{7}$$

where A_i and B_i are constant matrices. In this article, we consider the system to represent a quadrotor with an underlying acceleration controller. The input $\boldsymbol{u}_i$ is an acceleration command and the state $\boldsymbol{x}_i = [\boldsymbol{p}_i, \boldsymbol{v}_i] \in \mathbb{R}^6$ is a vector containing the position and velocity.

We assume that the velocities and acceleration inputs of the agents are bounded by constant vectors $\boldsymbol{v}_{\min}, \boldsymbol{v}_{\max}$ and $\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}$ respectively. This translates into the inequalities

$$\boldsymbol{v}_{\min} \leq \boldsymbol{v}_i(k) \leq \boldsymbol{v}_{\max} \tag{8}$$

$$\boldsymbol{u}_{\min} \leq \boldsymbol{u}_i(k) \leq \boldsymbol{u}_{\max} \tag{9}$$

Let $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_N] \in \mathbb{R}^{6N}$ the positions and velocities of the agents of the swarm, and $\boldsymbol{u} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots \boldsymbol{u}_N] \in \mathbb{R}^{3N}$. The system defining the motion of the swarm can be written as:

$$\boldsymbol{x}(k+1) = A\boldsymbol{x}(k) + B\boldsymbol{u}(k) \tag{10}$$

where A and B are block diagonal matrices with blocks $A_1, \ldots, A_N$ and $B_1, \ldots, B_N$, respectively.

4.3 NMPC swarm model

For our NMPC swarm model, we defined behavioral rules similar to those of the PF model. These rules are encoded as four terms of a cost function, including *separation*, *navigation*, *direction*, and *control effort*. At each time step, the evolution of the agents' movements is predicted over a constant time window, called the *prediction horizon*, with the dynamic model introduced in Sec. 4.2. These predictions are fed into the cost function, and the solution of the constrained optimization problem gives the control inputs for the swarm over the so-called *control horizon* (see Fig. 6). The prediction and control horizons are finite and shift forward at every time step. In the following, they will be denoted as $T_P = P\ dt$ and $T_C = C\ dt$ respectively, with $P \geq C$ and $P, C \in \mathbb{N}^+$.

We let $(\cdot)(k+l|k)$ represent the predicted value of $(\cdot)(k+l)$ with the information available at time $t(k)$ and $l \in \{0, \ldots, P\}$. We formulated a centralized version of the model[2], where the swarm rules are defined locally and every agent is only influenced by its neighbors. The *separation* term for agent i and time $t(k)$ is:

$$J_{\text{sep},i}(k) = \sum_{j \in \mathcal{N}_i} \sum_{l=1}^{P} \frac{w_{\text{sep}}}{|\mathcal{N}_i|} \left(\| \boldsymbol{p}_j(k+l|k) - \boldsymbol{p}_i(k+l|k) \|^2 - d_{\text{ref}}^2 \right)^2 \tag{11}$$

The *navigation* term is:

$$J_{\text{nav},i}(k) = \sum_{l=1}^{P} w_{\text{nav}} \left(\| \boldsymbol{v}_i(k+l|k) \|^2 - v_{\text{ref}}^2 \right)^2 \tag{12}$$

The *direction* term:

$$J_{\text{dir},i}(k) = \sum_{l=1}^{P} w_{\text{dir}} \left(1 - \frac{(\boldsymbol{v}_i(k+l|k) \cdot \boldsymbol{u}_{\text{ref}})^2}{\| \boldsymbol{v}_i(k+l|k) \|^2} \right)^2 \tag{13}$$

The combined action of the navigation (6) and direction (7) terms contribute to the so-called migration behavior of the swarm. The *control effort* is:

$$J_{u,i}(k) = \sum_{l=0}^{P-1} w_u \| \boldsymbol{u}_i(k+l|k) \|^2 \tag{14}$$

where w_{sep}, w_{nav}, w_{dir}, and w_u represent the constant weights associated with the cost function terms.

To prevent the agents from colliding with their neighbors or the obstacles, we associated with the cost function two sets of collision avoidance constraints:

$$d_{ij}(k+l|k)^2 \geq d_{\text{agent-safety}}{}^2 \quad i \in \{1, \ldots, N\}, j \in \mathcal{N}_i \tag{15}$$

$$d_{im}(k+l|k)^2 \geq d_{\text{obs-safety}}{}^2 \quad i \in \{1, \ldots, N\}, m \in \{1, \ldots, M\} \tag{16}$$

where $d_{\text{agent-safety}}$ is the safety distance between two agents' positions and $d_{\text{obs-safety}}$ is the safety distance that an agent should keep from the obstacle's position.

We let $\boldsymbol{X}(k) \in \mathbb{R}^{6NP}$ the stacked sequence of the predicted states $\boldsymbol{x}(k+l|k)$ over the horizon $l \in \{1, \ldots, P\}$ and $\boldsymbol{U}(k) \in \mathbb{R}^{3NP}$ the stacked sequence of the predicted control inputs $\boldsymbol{u}(p|k)$ over the horizon $l \in \{0, \ldots, P-1\}$. Then, the cost function and constraints define the following non-convex optimization problem:

[2] The cost function sums the contributions of every agent and the optimization process is run by a centralized software.

$$
\begin{aligned}
&\min_{\boldsymbol{X}(k),\boldsymbol{U}(k)} \quad \sum_{i=1}^{N} \left(J_{\mathrm{sep},i}(k) + J_{\mathrm{nav},i}(k) + J_{\mathrm{dir},i}(k) + J_{\mathrm{u},i}(k)\right) \\
&\text{subject to} \\
&\quad \boldsymbol{x}(k+l+1|k) = A\boldsymbol{x}(k+l|k) + B\boldsymbol{u}(k+l|k) \\
&\quad \boldsymbol{x}(k|k) = \boldsymbol{x}(k) \\
&\quad \boldsymbol{v}_{\min} \le \boldsymbol{v}_i(k+l|k) \le \boldsymbol{v}_{\max} \\
&\quad \boldsymbol{u}_{\min} \le \boldsymbol{u}_i(k+l|k) \le \boldsymbol{u}_{\max} \\
&\quad d_{ij}(k+l|k)^2 \ge d_{\text{agent-safety}}{}^2 \\
&\quad d_{im}(k+l|k)^2 \ge d_{\text{obs-safety}}{}^2
\end{aligned}
\tag{17}
$$

with $l \in \{1, \dots, P\}$, $i \in \{1, \dots, N\}$, $j \in \mathcal{N}_i$, and $m \in \{1, \dots, M\}$.

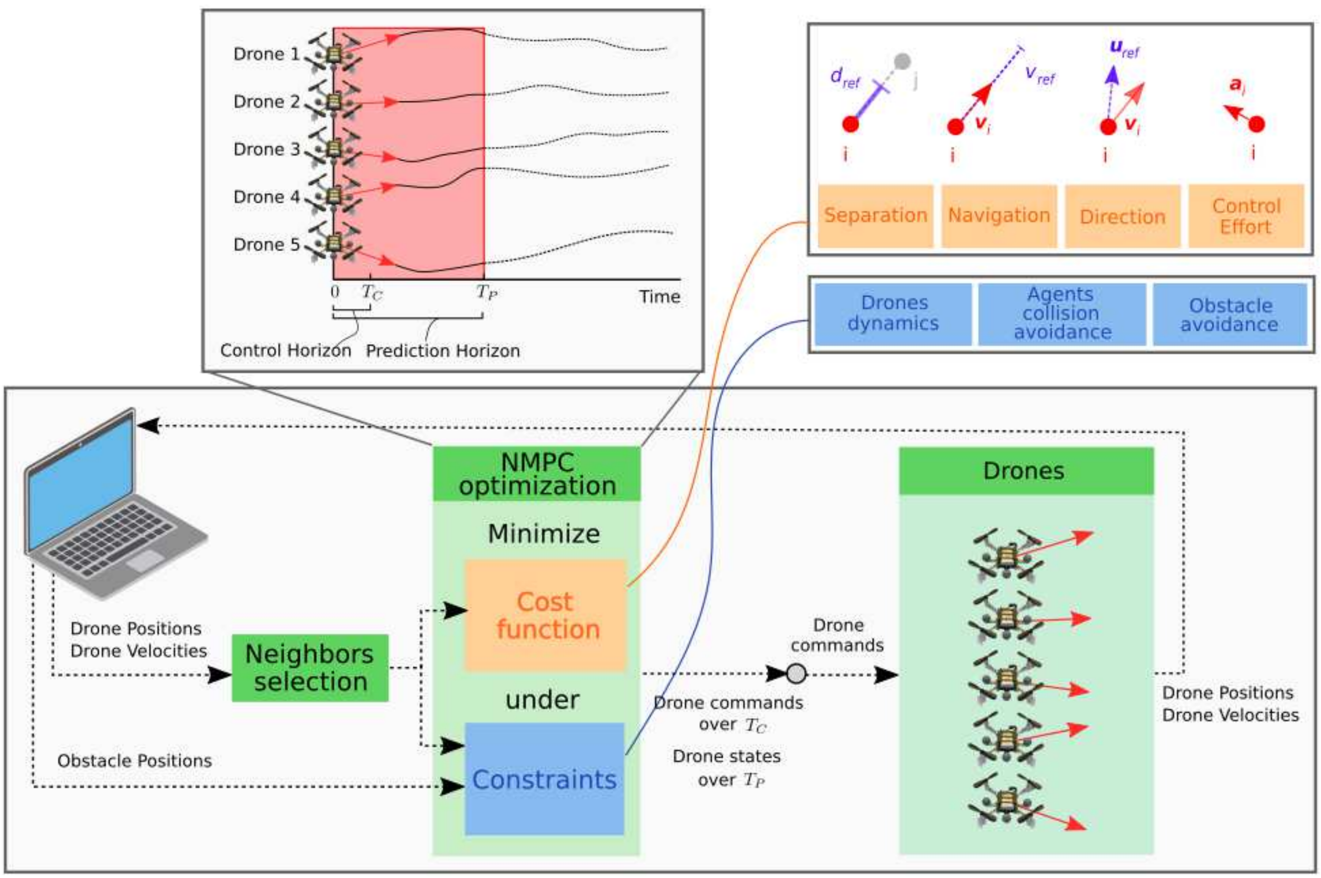

Fig. 6: **Predictive swarm algorithm workflow**. The proposed NMPC swarm algorithm optimizes four local rules: the *separation* incentivizes neighboring drones to stay at the preferred inter-agent distance d_{ref}, the *navigation* incentivizes constant migration speed v_{ref}, the *direction* drives the agents towards a preferred direction $\boldsymbol{u}_{\mathrm{ref}}$, and the *control effort* incentivizes small acceleration values. For each agent, the algorithm selects the nearest neighbors and feed their states into the optimization problem. The optimization problem, solved at discrete time instants, minimizes a cost function over the prediction horizon T_P and yields an optimal temporal sequence of control actions over the command horizon T_C. Only the first action is sent to the drones, which perform their motion accordingly. This procedure is repeatedly applied throughout the control process.

4.4 Simulation setup

We implemented our NMPC model in MATLAB with the help of acados [50], an open-source library for fast nonlinear optimal control. This software relies on C code generation for speeding up the computation in real-time applications. The system dynamics and the constraints of the problem are discretized by the library over the prediction horizon to obtain a structured Nonlinear Program (NLP). Then, the NLP is approximated through Sequential Quadratic Programming (SQP) that iteratively solves convex Quadratic Program (QP) sub-problems. After applying a condensing step, a linear algebra solver, IPOPT, based on the Interior Point (IP) method finds the solution of the sub-problems [51]. We run our simulations on a DELL Precision Tower with a 3.6 GHz Intel Core i7-7700 processor and 16 GB 2400 MHz RAM, where we set the maximum number of SQP to 7 and the maximum number of QP iterations to 7.

4.5 Drone experimental setup

In our experiments, we used five Bitcraze Crazyflie 2.1 quadrotors (Fig. 1c). Each quadrotor is equipped with a 3-axis accelerometer, a 3-axis gyroscope, a pressure sensor, and a marker deck for hosting passive reflective markers. The microcontroller is a STM32F4 running at 168MHz, on which both state estimation and low-level control are running. An OptiTrack motion capture system was used to track the position of the robots. All the acceleration commands for the drones were computed on a single computer with our NMPC model, integrated into position commands and broadcast to the swarm through a radiolink, alongside the estimated position of each drone. The estimated positions were used by the drones to perform the lower-level control loops and track the commands sent. The positions and velocities used by the swarm model were predicted with the agents' dynamic model. To guarantee the transferability of the NMPC swarm model to hardware experiments, we decreased the number of maximum SQP to 4. This was sufficient to compute converging solutions of the NLP in less than $0.1\ s$.

Data and materials availability

The data needed to reproduce the experiments are present in the paper or in the Supplementary Materials. The data collected during simulation and hardware experiments can be downloaded from http://doi.org/10.5281/zenodo.4018870.

Code availability

The code that supports the findings of this study are available from the corresponding author upon reasonable request.

References

[1] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective leadership and decision-

making in animal groups on the move", *Nature*, vol. 433, pp. 513-516, 2005

[2] H. Hildenbrandt, C. Carere, and C. K. Hemelrijk, "Self-organized aerial displays of thousands of starlings: a model", *Behav. Ecol.*, vol. 21, no. 6, pp. 1349–1359, 2010

[3] G. F. Young, L. Scardovi, A. Cavagna, I. Giardina, and N. E. Leonard, "Starling Flock Networks Manage Uncertainty in Consensus at Low Cost", *PLOS Comput. Biol.*, vol. 9, no. 1, p. e1002894, 2013

[4] 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

[5] M. Nagy, Z. Ákos, D. Biro, and T. Vicsek, "Hierarchical group dynamics in pigeon flocks," *Nature*, vol. 464, no. 7290, pp. 890–893, 2010

[6] M. Yomosa, T. Mizuguchi, G. Vásárhelyi, and M. Nagy, "Coordinated Behaviour in Pigeon Flocks," *PLOS ONE*, vol. 10, no. 10, p. e0140558, 2015

[7] M. Ballerini *et al.*, "Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study," *Proc. Natl. Acad. Sci.*, vol. 105, no. 4, pp. 1232–1237, 2008

[8] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, 2015

[9] T. Stirling, J. Roberts, J.-C. Zufferey, and D. Floreano, "Indoor navigation with a swarm of flying robots", *IEEE Int. Conf. Rob. Autom. (ICRA)*, pp. 4641–4647, 2012

[10] 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

[11] A. T. Erman, L. van Hoesel, P. Havinga, and J. Wu, "Enabling mobility in heterogeneous wireless sensor networks cooperating with UAVs for mission-critical management", *IEEE Wirel. Commun.*, vol. 15, no. 6, pp. 38–46, 2008

[12] 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

[13] F. Augugliaro *et al.*, "The Flight Assembled Architecture installation: Cooperative construction with flying machines", *IEEE Control Syst. Mag.*, vol. 34, no. 4, pp. 46–64, 2014

[14] "Intel Drone Light Show: Intel's 50th Anniversary", https://www.intel.com/content/www/us/en/technology-innovation/videos/drone-light-show-50th-anniversary-video.html, 21 Nov. 2019

[15] "EHang Egret's 1374 drones dancing over the City Wall of Xian, achieving a Guinness World Records title", https://www.ehang.com/news/365.html, 02 May 2018

[16] "The Globe and Mail: Mini-drone use on the rise to light up big concerts like Celine Dion and Drake", https://veritystudios.com/news/globe-and-mail-celine, 06 Dec. 2019

[17] D. Chen, X. Liu, B. Xu, and H.-T. Zhang, "Intermittence and connectivity of interactions in pigeon flock flights", *Sci. Rep.*, vol. 7, no. 1, 2017

[18] A. Berdahl, C. J. Torney, C. C. Ioannou, J. J. Faria, and I. D. Couzin, "Emergent sensing of complex environments by mobile animal groups", *Science*, vol. 339, no. 6119, pp. 574–576, 2013

[19] I. D. Couzin, "Synchronization: The Key to Effective Communication in Animal Collectives", *Trends Cogn. Sci.*, vol. 22, no. 10, pp. 844–846, 2018

[20] C. W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model", *Comput.*

Graph., vol. 21, pp. 25–43, 1987.

[21] S. Hauert *et al.*, “Reynolds Flocking in Reality with Fixed-Wing Robots: Communication Range vs. Maximum Turning Rate”, *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 5015-5020, 2011

[22] F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, “Learning Vision-based Flight in Drone Swarms by Imitation”, *IEEE Robot. Autom. Lett.*, vol. 4, pp. 4523–4530, 2019

[23] R. Olfati-Saber, “Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory,” *IEEE Trans. Autom. Control*, vol. 51, no. 3, pp. 401–420, 2006

[24] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, “Optimized flocking of autonomous drones in confined environments”, *Sci. Rob.*, vol. 3, no. 20, p. eaat3536 2018

[25] G. Vasarhelyi *et al.*, “Outdoor flocking and formation flight with autonomous aerial robots”, *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 3866–3873, 2014

[26] Y. Koren and J. Borenstein, “Potential field methods and their inherent limitations for mobile robot navigation”, *IEEE Int. Conf. Rob. Autom. (ICRA)*, pp. 1398–1404, 1991

[27] E. Soria, D. Floreano, and F. Schiano, “The influence of limited visual sensing on the Reynolds flocking algorithm”, *Int. Conf. Rob. Comp. (IRC)*, pp. 138–145, 2019

[28] F. Borrelli, A. Bemporad, and M. Morari, “Predictive Control for Linear and Hybrid Systems”, 1st ed. Cambridge University Press, 2017

[29] L. Grune and J. Pannek, “Nonlinear Model Predictive Control. Theory and Algorithms”, Springer London, 2011

[30] T. Baca, D. Hert, G. Loianno, M. Saska, and V. Kumar, “Model Predictive Trajectory Tracking and Collision Avoidance for Reliable Outdoor Deployment of Unmanned Aerial Vehicles”, *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 6753–6760, 2018

[31] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, “PAMPC: Perception-Aware Model Predictive Control for Quadrotors”, *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 1-8, 2018

[32] 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.*, vol. 5, no. 2, pp. 604–611, 2020

[33] T. Keviczky, F. Borrelli, K. Fregene, D. Godbole, and G. J. Balas, “Decentralized Receding Horizon Control and Coordination of Autonomous Vehicle Formations”, *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 1, pp. 19–33, 2008

[34] Ruben Van Parys, “Distributed MPC for multi-vehicle systems moving in formation”, *Robot. Auton. Syst.*, 2017

[35] U. Eren, A. Prach, B. B. Koçer, S. V. Raković, E. Kayacan, and B. Açıkmeşe, “Model Predictive Control in Aerospace Systems: Current State and Opportunities”, *J. Guid. Control Dyn.*, vol. 40, no. 7, pp. 1541–1566, 2017

[36] W. B. Dunbar and R. M. Murray, “Distributed receding horizon control for multi-vehicle formation stabilization”, *Automatica*, vol. 42, no. 4, pp. 549–558, 2006

[37] R. L. Raffard, C. J. Tomlin, and S. P. Boyd, “Distributed optimization for cooperative agents: application to formation flight”, *IEEE Conf. Decision Control (CDC)*, vol. 3, pp. 2453-2459 , 2004

[38] T. Schouwenaars, J. How, and E. Feron, “Decentralized Cooperative Trajectory Planning of Multiple Aircraft with Hard Safety Guarantees”, *AIAA Guid. Nav. and Cont. Conf.*, 2004

[39] A. Richards and J. How, “Implementation of Robust Decentralized Model Predictive Control”, *AIAA Guid. Nav. and Cont. Conf.*, 2005
[40] Y. Kuwata and J. P. How, “Robust Cooperative Decentralized Trajectory Optimization using Receding Horizon MILP,” in *American Cont. Conf.*, pp. 522–527, 2007
[41] I. Kagan Erunsal, A. Martinoli, and R. Ventura, “Decentralized Nonlinear Model Predictive Control for 3D Formation of Multirotor Micro Aerial Vehicles with Relative Sensing and Estimation”, *Int. Symp. on Multi-Robot and Multi-Agent Syst. (MRS)*, pp. 176–178, 2019
[42] W. Zhao and T. H. Go, “Quadcopter formation flight control combining MPC and robust feedback linearization”, *J. Frankl. Inst.*, vol. 351, no. 3, pp. 1335–1355, 2014
[43] W. Ren and N. Sorensen, “Distributed coordination architecture for multi-robot formation control”, *Robot. Auton. Syst.*, vol. 56, no. 4, pp. 324–333, 2008
[44] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, “Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control”, *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 236–243, 2017
[45] V. Kumar and N. Michael, “Opportunities and challenges with autonomous micro aerial vehicles,” *Int. J. Robot. Res.*, vol. 31, no. 11, pp. 1279–1291, 2012
[46] M. Petrlík, T. Báča, D. Heřt, M. Vrba, T. Krajník, and M. Saska, “A Robust UAV System for Operations in a Constrained Environment”, *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2169–2176, 2020
[47] K. R. Sapkota *et al.*, “Vision-based Unmanned Aerial Vehicle detection and tracking for sense and avoid systems,” *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 1556–1561, 2016
[48] E. R. Hunt and S. Hauert, “A checklist for safe robot swarms,” *Nat. Mach. Intell.*, vol. 2, no. 8, pp. 420–422, 2020
[49] B. Lindqvist, S. S. Mansouri, A. Agha-mohammadi, and G. Nikolakopoulos, “Nonlinear MPC for Collision Avoidance and Control of UAVs With Dynamic Obstacles,” *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6001–6008, 2020
[50] R. Verschueren *et al.*, “Towards a modular software package for embedded optimization”, *IFAC-Pap.*, vol. 51, no. 20, pp. 374–380, 2018
[51] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”, *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006

Acknowledgments

We thank Fabian Schilling and Anthony De Bortoli for their valuable contributions. This work was supported by the Swiss National Science Foundation with grant number 200020_188457.

Author contributions

All authors contributed to the conception of the project and were involved in the analysis of the results. E.S. has designed, implemented, and performed software and hardware experiments of the NMPC algorithm for the navigation of drone swarms in cluttered environments. All authors contributed to the writing of the manuscript.

Competing interests

The authors declare that they have no competing interests.

Figures

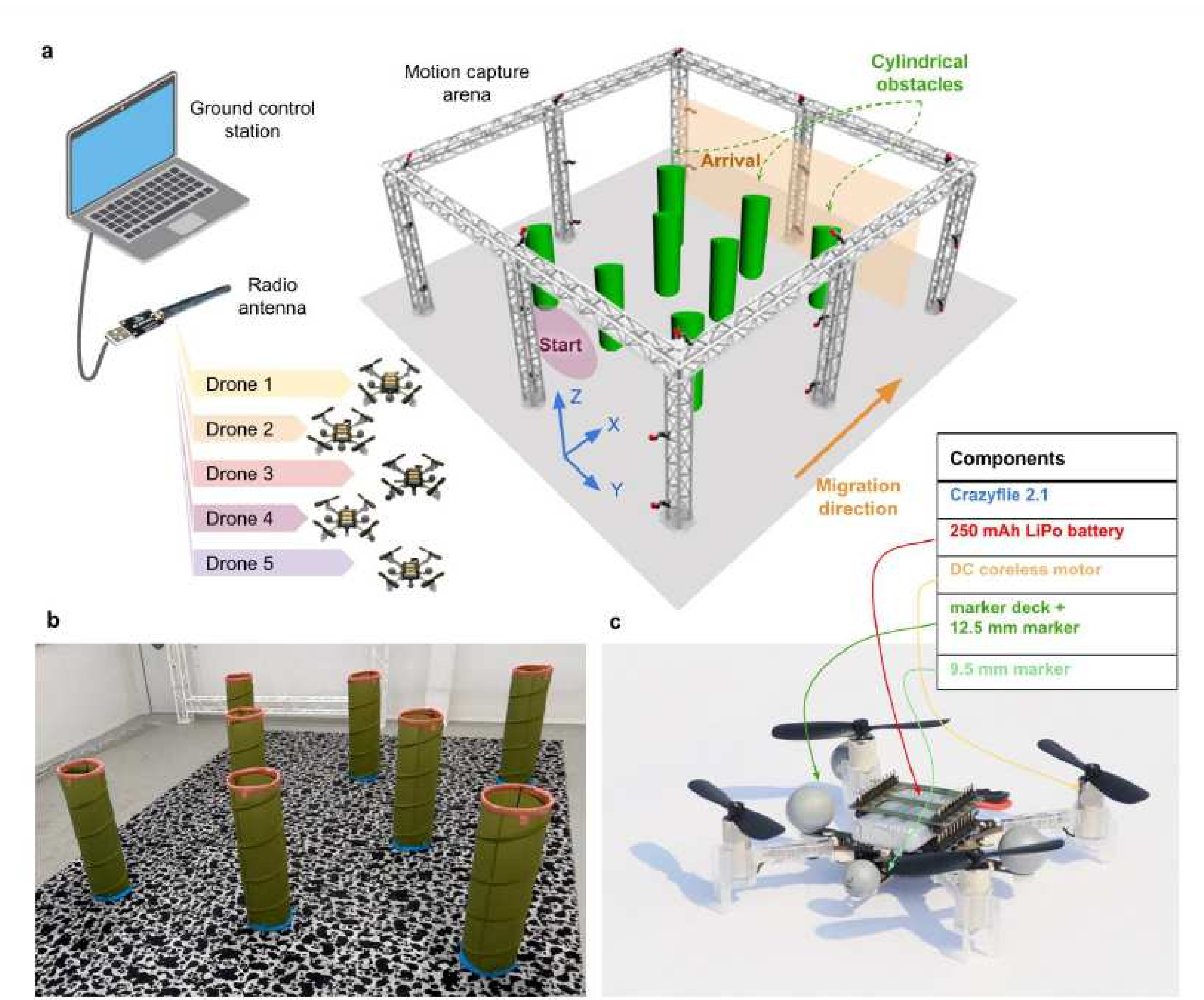

Figure 1

Experimental setup of drone swarm flight in cluttered environments. (a) Illustration of the experimental setup and the environment configuration. A ground control station, equipped with a radio transmitter, computes and sends run-time control commands to the drones. The swarm flies in the 3D space of an indoor flying arena. The drones take off from initial random positions within a predefined start area (red zone). Drones swarm along the preferred migration direction (orange arrow). The mission is completed when all drones cross the arrival plane (Fig. 1a) on the opposite side of the region. (b) Indoor test environment populated with cylindrical obstacles. (c) Components of the drones used for the hardware experiments.

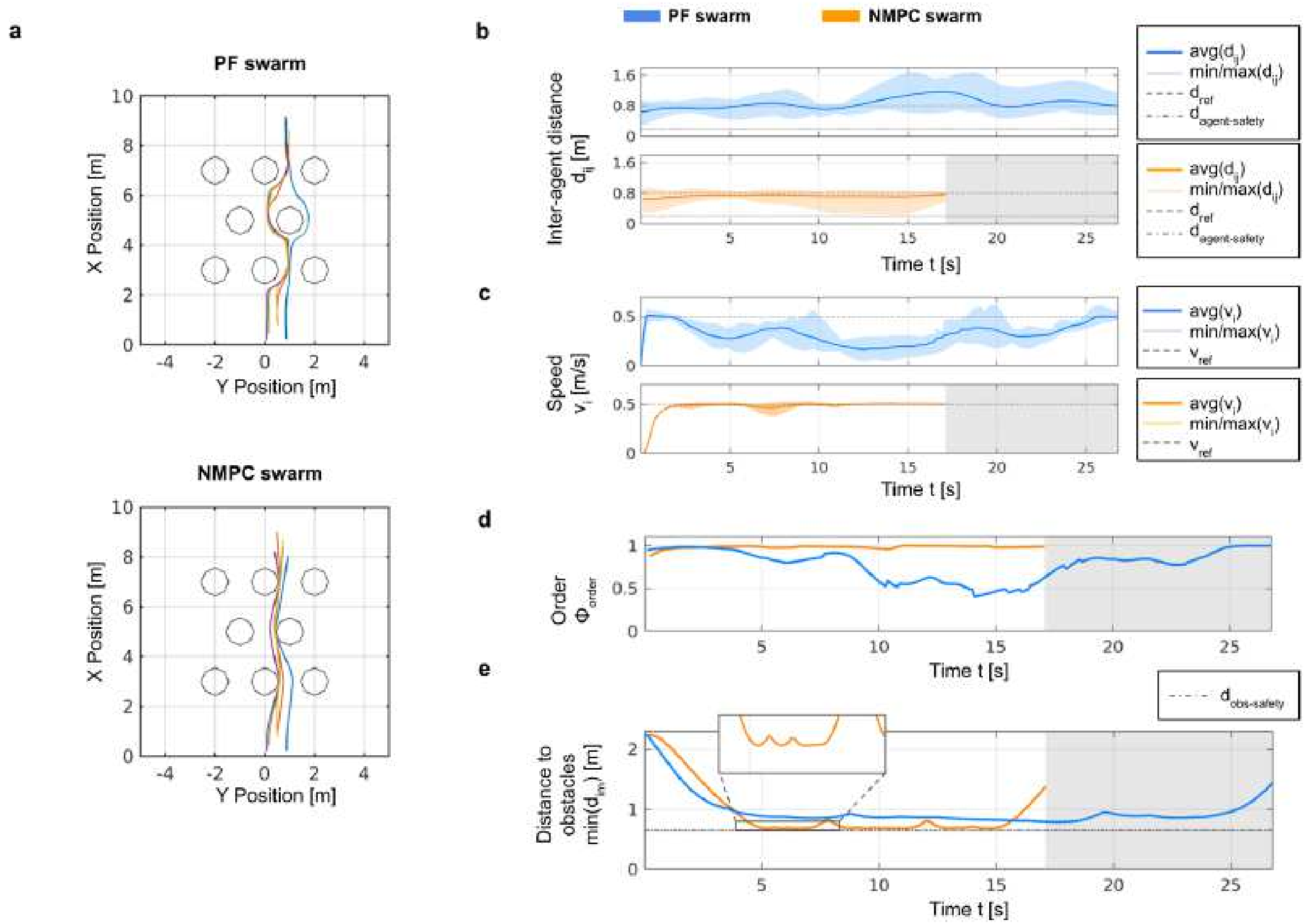

Figure 2

Comparison of the PF and NMPC aerial swarms in simulation experiments. (a) Top views of the 3D trajectories of five drones flying in a cluttered environment with the PF (top) and the NMPC models (bottom) (see Supplementary Video 1). The circular objects on the map correspond to cylindrical obstacles. (b) Inter-agent distance average (solid line) and range (shaded region). The curve on top (blue) refers to the PF swarm, while the one at the bottom (orange) refers to the NMPC swarm. (c) Swarm speed average (solid line) and range (shaded region). (d) Order metric. (e) Distance to obstacles, min(□□□), expressed as the minimum distance between the swarm's agents and the set of obstacles.

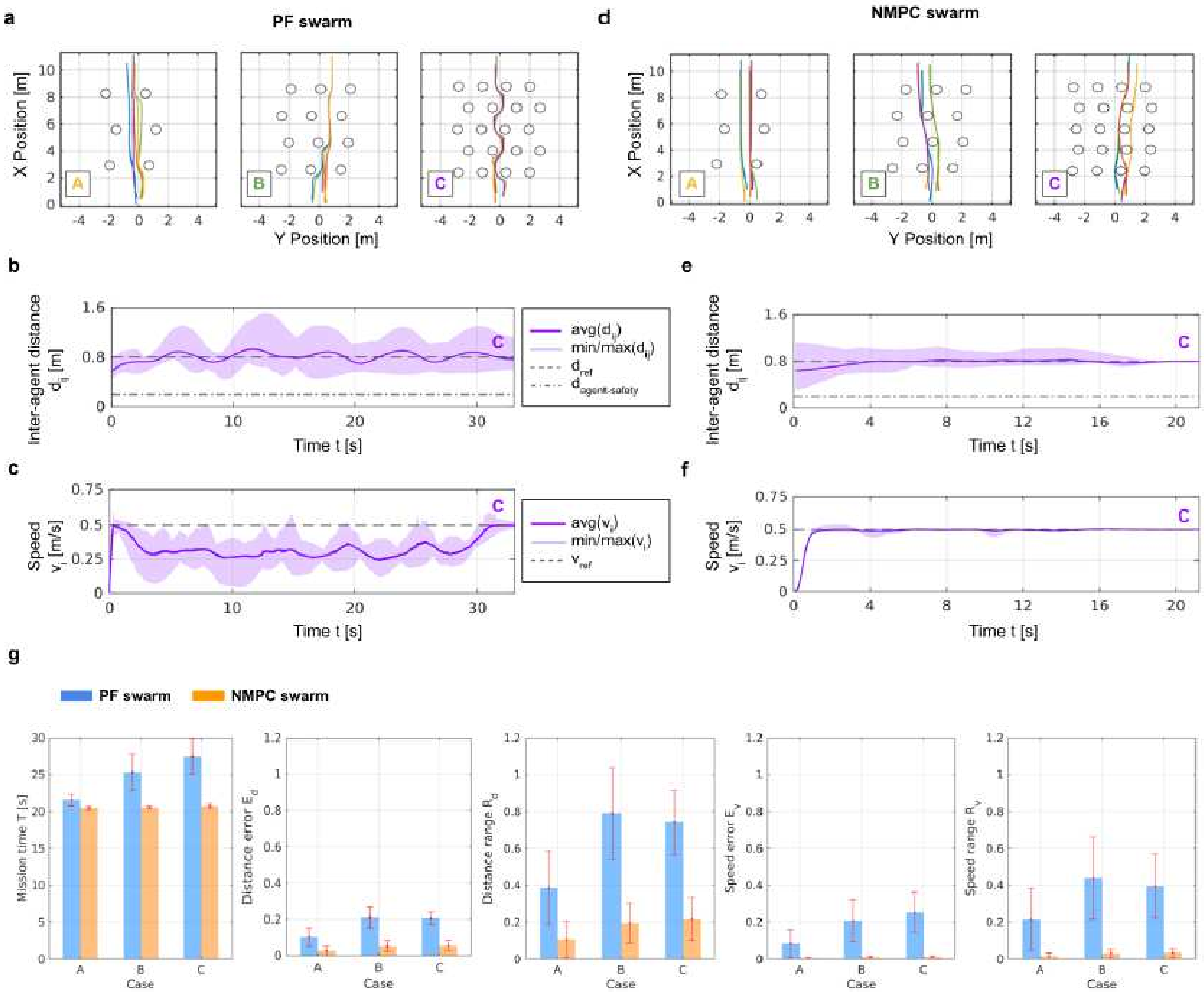

Figure 3

Comparison of the PF and NMPC swarm deployment in environments with different obstacle densities. (a, d) Top views of the 3D simulated trajectories of the PF and the NMPC swarms in environments with three different obstacle densities. The density increases from left to right (Case A: 0.06, B: 0.12, and C: 0.20) (see Supplementary Video 1). (b, c) Inter-agent distance and speed of the PF swarm in Case C. (e, f) Inter-agent distance and speed of the NMPC swarm in Case C. (g) Aggregated results (average and standard deviation) of 10 stochastic simulations of the PF (blue) and NMPC (orange) swarm models in Cases A, B, and C. The represented metrics are the mission time □, the distance error □□, the distance range □□, the speed error □□, and the speed range □□ (see Supplementary Table 1).

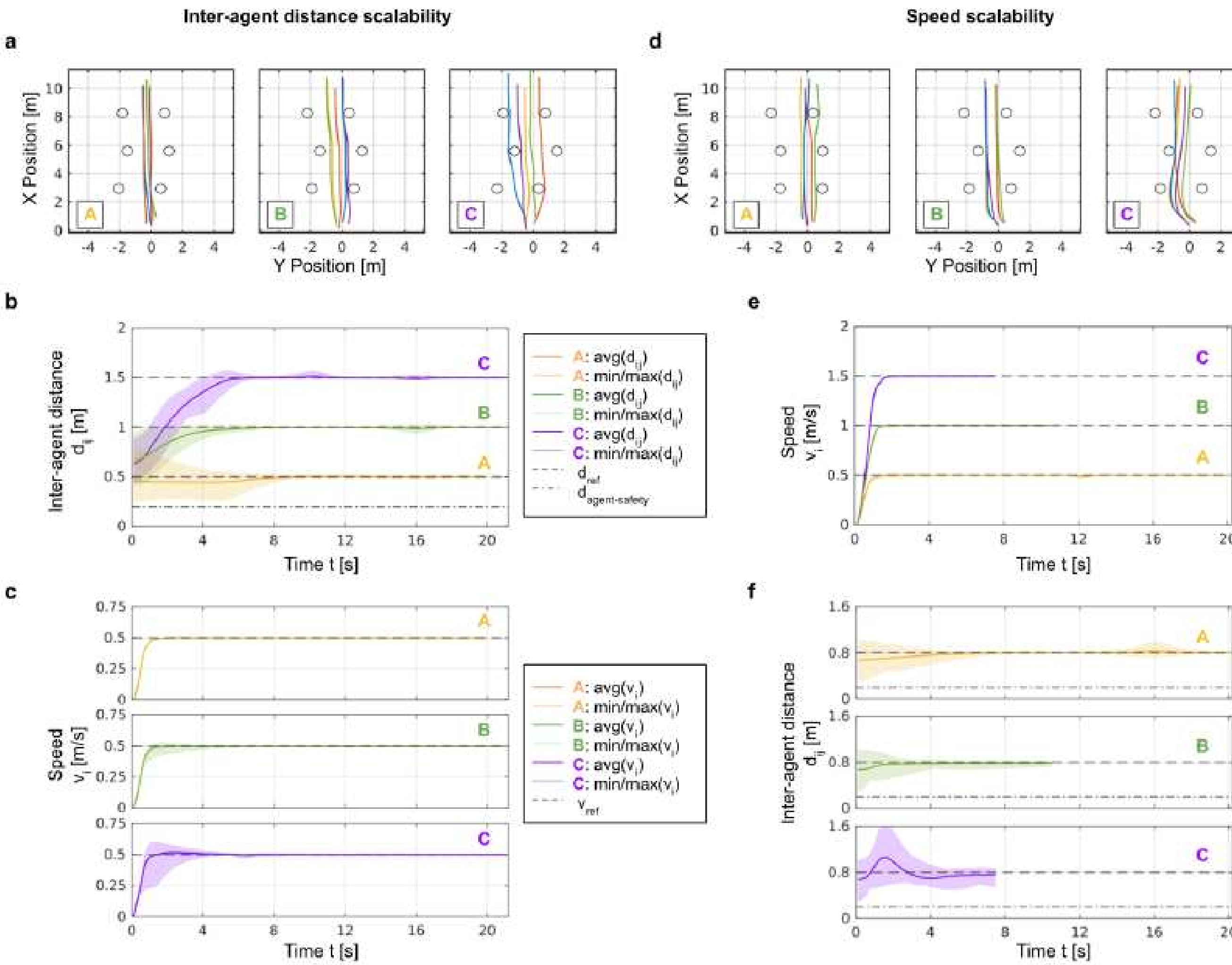

Figure 4

Scalability of the NMPC swarm in inter-agent distance and speed. On the left, simulation results on the scalability of the NMPC swarm model in the inter-agent distance for three preferred distance values (Case A: ref=0.5, B: 1.0, and C: 1.5). On the right, simulation results on the scalability in the swarm speed for three preferred speed values (Case A: ref= 0.5/, B: 1.0/, and C: 1.5/). (a, d) Top views of the 3D trajectories of the swarm (see Supplementary Video 1). (b, c) Inter-agent distance and speed for the experiment on the inter-agent distance scalability. (e, f) Inter-agent distance and speed for the experiment on the speed scalability. The obstacle size and density are the same for the six cases.

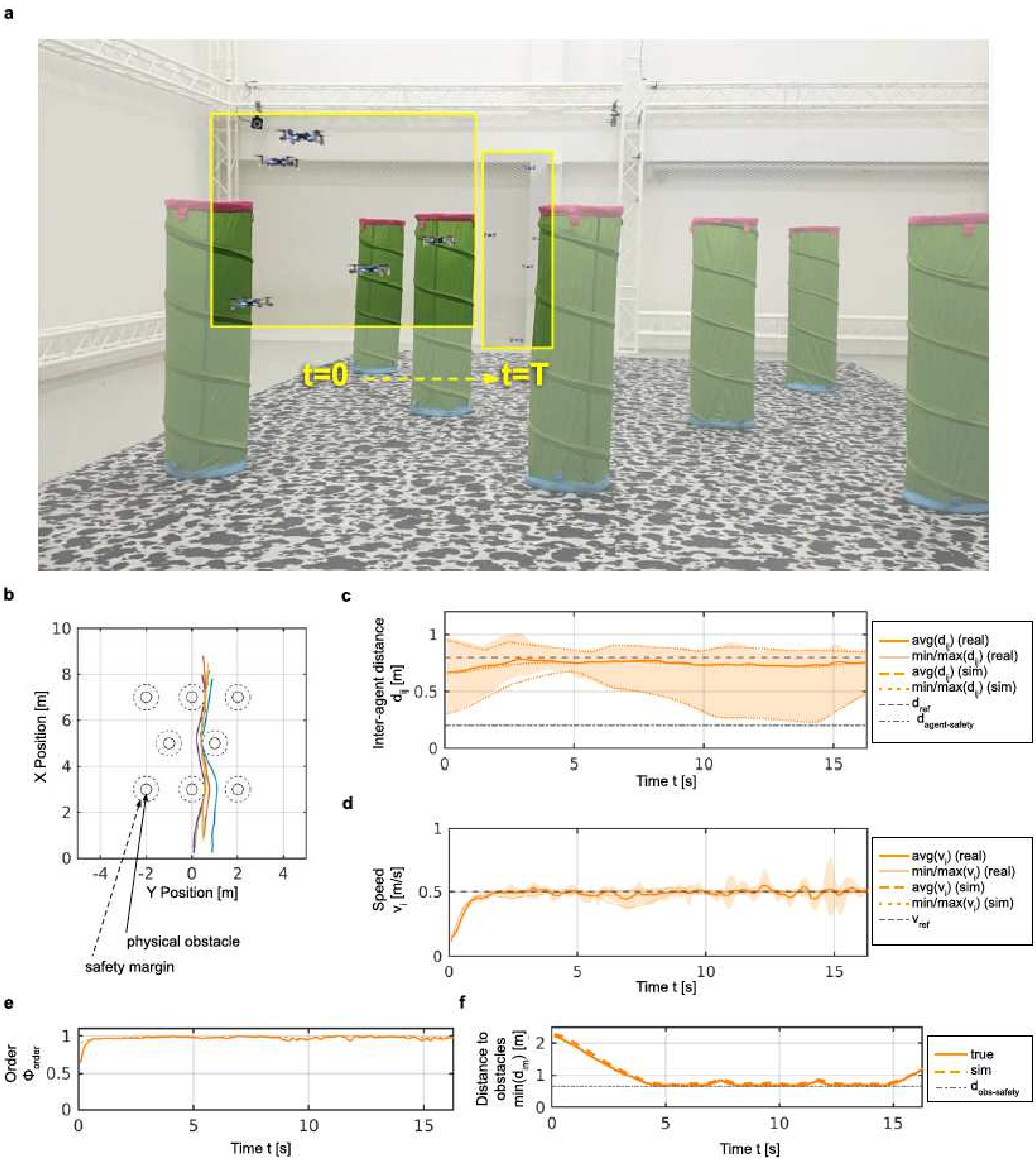

Figure 5

Real-world experiment of the NMPC swarm. (a) The swarm, composed of five commercial palm-sized quadrotors, flies through cylindrical obstacles in a motion capture arena. The swarm crosses the region from the foreground (t=0 s) to the background (t=T), while maintaining cohesion and avoiding the obstacles (see Supplementary Video 2). (b) Top view of the trajectories of the drones. For the real-world deployment, we selected obstacles with a smaller radius (R_{obs}= 0.30 m) than in simulation (R_{obs}=0.55 m), but we used the same safety distance for collision avoidance as in simulation (□obs−safety=0.65 □), which introduces a safety margin of 0.25 □ from the physical obstacles (see Supplementary Table 4). (c) Average inter-agent distance and range with the real swarm (solid line and shaded region, respectively) and the simulated swarm (dashed and dotted lines, respectively). (d) Average speed and range with the real swarm (solid line and shaded region, respectively) and the simulated swarm (dashed and dotted lines, respectively). (e) Swarm's order: real (solid line) and simulated (dashed line) swarm. (f) Swarm distance to obstacles. The offset in the real data (solid line) with respect to the simulated data (dashed line) is due to the safety margin.

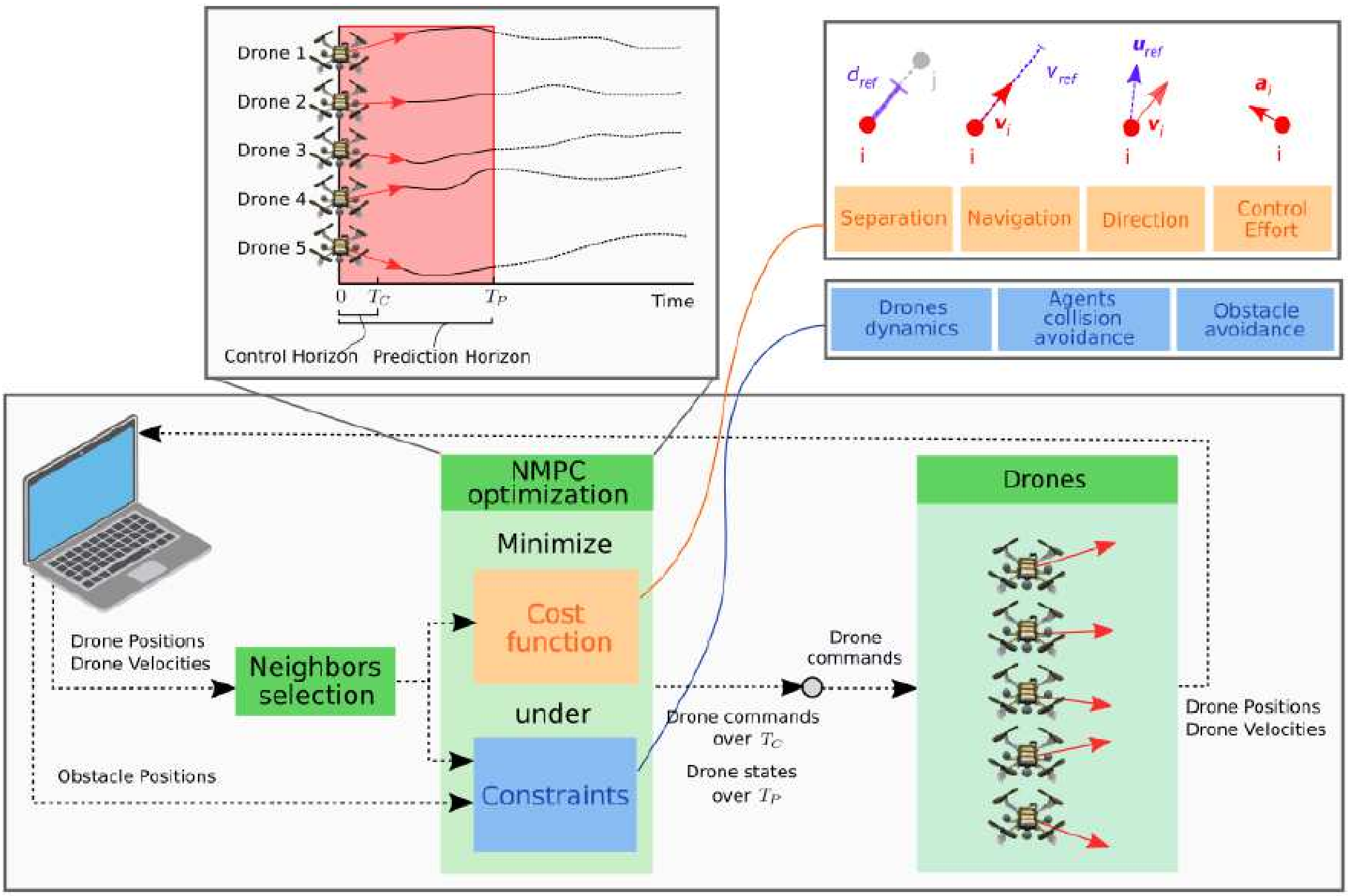

Figure 6

Predictive swarm algorithm workflow. The proposed NMPC swarm algorithm optimizes four local rules: the separation incentivizes neighboring drones to stay at the preferred inter-agent distance □ref, the navigation incentivizes constant migration speed □ref, the direction drives the agents towards a preferred direction □ref, and the control effort incentivizes small acceleration values. For each agent, the algorithm selects the nearest neighbors and feed their states into the optimization problem. The optimization problem, solved at discrete time instants, minimizes a cost function over the prediction horizon □□ and yields an optimal temporal sequence of control actions over the command horizon □□. Only the first action is sent to the drones, which perform their motion accordingly. This procedure is repeatedly applied throughout the control process.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- movies1.mp4
- movies2.mp4
- supplementarymaterials.pdf