SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Formation Adaptation in Obstacle-Cluttered Environments via MPC-Based Trajectory Planning

Yuda CHEN[1] & Zhongkui LI[1*]

[1]*Department of Mechanics and Engineering Science, College of Engineering, Peking University, Beijing* 100871*, China*

Appendix A More Details in Problem Statement

Appendix A.1 Collision Avoidance

For safety, assume that each agent $i \in \mathcal{N}$ occupies a ball of radius $r_a > 0$, i.e., $\mathcal{B}^i(t) = \{p \in \mathbb{R}^2 \mid \|p - p^i(t)\|_2 \leqslant r_a\}$. Thus, any pair of two agents $i, j \in \mathcal{N}$ collide with each other if $\|p^i - p^j\|_2 < 2r_a$. As in our previous work [1], the inter-agent collision avoidance is encoded via the modified buffered Voronoi cell (MBVC) as enforcing the following constraint:

$$a_k^{ij\mathrm{T}} p_k^i \geqslant b_k^{ij},\ j \neq i,\ k \in \mathcal{K}, \tag{A1}$$

where $a_k^{ij} = \frac{\overline{p}_k^i - \overline{p}_k^j}{\|\overline{p}_k^i - \overline{p}_k^j\|_2}$, $b_k^{ij} = a_k^{ij\mathrm{T}} \frac{\overline{p}_k^i + \overline{p}_k^j}{2} + \frac{r'_{\min}}{2}$, and $r'_{\min} = \sqrt{4{r_a}^2 + h^2 v_{\max}^2}$. Here, the predetermined trajectory is defined as $\overline{P}^i(t) = [\overline{p}_1^i(t), \overline{p}_2^i(t), \ldots, \overline{p}_K^i(t)]$,where $\overline{p}_k^i(t) = p_{k+1}^i(t-h), k \in \tilde{\mathcal{K}} := 1, 2, \ldots, K-1$ and we enforce that $\overline{p}_K^i(t) = p_K^i(t-h)$. It should be noted that, as also mentioned in [1], for agent i, the MBVC is not require to be established between itself and all the others. In real implementation, only some neighboring agent in a proximal range is required to be concerned and they are the one that this agent needs to communicate with.

The agent $i \in \mathcal{N}$ might collide with the static obstacles at time $t > 0$, if their occupied volumes intersect, i.e., $\mathcal{B}^i(t) \cap \mathcal{O} \neq \emptyset$ ought to be satisfied, where $\mathcal{O}$ is the volume occupied by a set of convex-shaped static obstacles, each of which is defined as the convex hull of a set of known vertices. Moreover, the collection of all the obstacles' indices is denoted as $\mathcal{M}$. In this work, we derive the safe zone from the agents' predetermined trajectories. Concretely, a safe zone is a convex polygon whose edge w.r.t. obstacle $m \in \mathcal{M}$ is derived by solving the following optimization:

$$\begin{aligned} \max_{a_m^{\mathrm{zone}}, b_m^{\mathrm{zone}}, \delta}\ & \delta, \\ \text{s.t. } & a_m^{\mathrm{zone}T} p \geqslant \delta + b_m^{\mathrm{zone}},\ p \in \mathcal{P}^{\mathrm{free}} \\ & a_m^{\mathrm{zone}T} p \leqslant b_m^{\mathrm{zone}},\ p \in \mathcal{P}_m^O, \\ & \left\|a_m^{\mathrm{zone}}\right\|_2 = 1, \end{aligned} \tag{A2}$$

where $\mathcal{P}^{\mathrm{free}} = \{\overline{p}_k^i | i \in \mathcal{N}, i \in \mathcal{K}\}$ and $\mathcal{P}_m^o$ behalves the vertices of obstacle m. The method which can handle the above non-convex optimization can be found in [2]. The details are omitted here for brevity.

In the above optimization, we try to form a separation plane between a set $\mathcal{P}^{\mathrm{free}}$ and the obstacle m. Note that many points in this set $\mathcal{P}^{\mathrm{free}}$ are interior points of the convex hull of this set, which can be omitted in calculations. The reason is obvious, since the optimization is encoded to separate two convex set, i.e., obstacle and $\mathcal{P}^{\mathrm{free}}$, and $\mathcal{P}^{\mathrm{free}}$ can be represented only by the vertex point of its convex hull. Moreover, the above optimization will not be applied to all obstacles. Specifically, the separating plane will be computed from the nearest obstacle to the farthest one. When turning on an obstacle, a separating plane will be built if the obstacle has a contact with the currently formed convex polyhedra. Otherwise, such an obstacle will be omitted. In addition, if the distance between this convex hull and the agent is too large, the obstacle will not be considered either.

Appendix A.2 Connectivity Maintenance

As the means to exchange information, any pair of agents $i, j \in \mathcal{N}$ can communicate with each other if the following two conditions hold: (i) their relative distance is bounded by the communication range $d_c > 0$, i.e., $\|p^i(t) - p^j(t)\|_2 \leqslant d_c$; (ii) the line-of-sight (LOS) between agents i and j cannot be obstructed by obstacles, i.e., $\mathbf{Line}(p^i(t),\, p^j(t)) \cap \mathcal{O} = \emptyset$, where $\mathbf{Line}(p^i(t),\, p^j(t))$ is the line segment from position $p^i(t)$ to position $p^j(t)$.

To preserve the connectivity of this swarm, the swarm during the motion should meet the following two demands: 1) the LOS of any pair of agents are collision-free; 2) the distance between any pair of agents should be smaller than the connection distance d_c. Fortunately, the swarm has been restricted in a collision-free convex polygon, i.e., the safe zone. Evidently, the LOS of any pairs must be included in this safe zone and collision-free.

As a result, we only need to limit the distance between two agents. The method is to enforce the agents' planned trajectories in a common sphere with diameter d_c as follows:

$$\left\|\overline{p}_k^i - p_c^{\mathrm{commu}}\right\|_2 \leqslant d_c,\ i \in \mathcal{N}, \tag{A3}$$

* Corresponding author (email: zhongkli@pku.edu.cn)

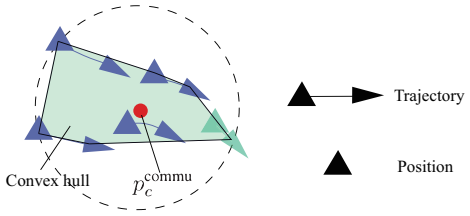

Figure A1 The scheme of communication sphere derivation. The green region is the convex hull of $\mathcal{P}^{\text{free}}$.

where p_c^{commu} is the center of this sphere. Concretely, p_c^{commu} will be chosen as the center of the minimum circle which can cover all trajectories of agents as depicted in Fig. A1. It can be calculated from the following optimization:

$$\begin{aligned} &\min_{d,\ p_c^{\text{commu}}} \ d, \\ &\text{s.t.,}\ \left\| p_c^{\text{commu}} - \overline{p}_k^i \right\|_2 \leqslant d, \end{aligned} \tag{A4}$$

where $i \in \mathcal{N}$ and d is the diameter to be minimized.

Appendix B More Details in Formation Adaptation

In this section, we will provide more details about formation adaptation.

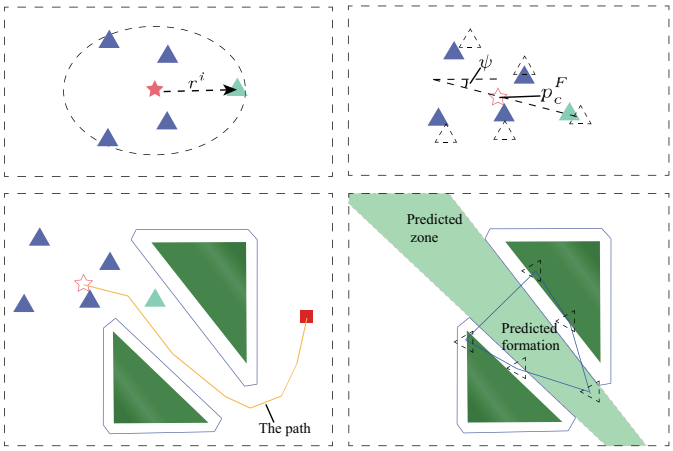

Figure B1 **Top-Left**: initialization. D **Top-Right**: pose estimation. **Bottom-Left**: path planning. **Bottom-Right**: predicted zone derivation.

Appendix B.1 Initialization

As shown in Fig. B1, given an expected formation, an ellipsoid is first calculated, which can cover these swarm and has the minimum area. The center of this ellipsoid is regarded as the center of this formation. The relative position between this center and agent i's position is denoted by r^i, $i \in \mathcal{N}$.

Appendix B.2 Pose Estimation

As shown in Fig. B1, the pose of formation is estimated. Concretely, to estimate the approximated yaw ψ and the central position p_c^F, the following optimization is formulated:

$$\begin{aligned} &\min_{\psi, p_c^F} \sum_{i=1}^{N} ||R^i(\psi) r^i + p_c^F - p_K^i||_2^2 \\ &\text{s.t.,}\ |\psi(\text{t} - \text{h}) - \psi| \leqslant \delta\psi_{\max}, \end{aligned} \tag{B1}$$

where $\psi(t-h)$ is the evaluated yaw in the last iteration and $\psi_{\max} \in \mathbb{R}$ is the yaw's maximum deviation. To handle this non-convex optimization, in real implementation, $\mathcal{R}(\psi)$ is replaced by

$$\mathcal{R}(\psi) = \mathcal{R}(\psi(t-h)) + \frac{\partial \mathcal{R}(\theta)}{\partial \theta}|_{\theta=\psi(t-h)}(\psi - \psi(t-h)). \tag{B2}$$

As a result, the final optimization is a quadratic program.

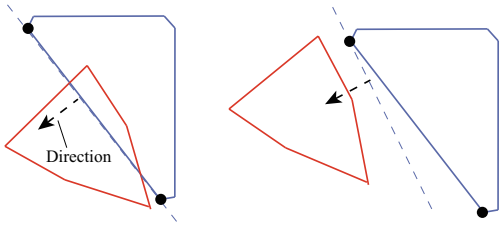

Figure B2 The scheme of GJK (left figure) and EPA (right figure).

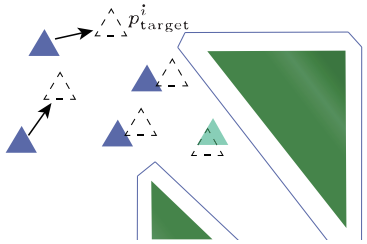

Figure B3 The scheme of formation optimization. The dashed triangle is the optimized target of each agent.

Appendix B.3 Path Planning

As shown in Fig. B1, a path is planned by the coordinator from p_c^F to p_{target}. To provide margin for making the group not got stuck by obstacles, when adapting path planning, the underlying obstacles will be inflated by r_F, where r_F is called the radius of this formation. The definition of r_F is defined as follows:

Definition 1. (Formation Radius) The expected formation is shrunk equally in every axis, until there exists a pair of agent i and j such that $\|p^i - p^j\|_2 < 2r_a$. In this shrunk formation, the minimum ellipsoid covering this swarm is obtained and r_F is chosen as the length of its minor axis.

After inflating obstacles, a path is planned via a RRT*-based method called ABIT* [3] , which indicates the motion tendency of the group. Specifically, the planned path is derived into a sequence of points as follows: $\Gamma = \{p_c^F, \cdots, p_{\text{target}}\}$.

Appendix B.4 Predicted Zone Derivation

To make the swarm adaptive to the distribution of obstacles, the underlying agents shall not only be restricted in the safe zone but also in a predicted zone, which can reflect the ongoing zone of the safety region. In the previous part, a path stemming from the current center to the target is planned, by which a possible formation in the future can be approximately derived as shown in Fig B1. Specifically, along this path, the center is chosen as the first turning point p_{tractive} and the yaw of the formation is the direction vertical to the angle bisector at this point. After obtaining this predicted formation, a spatial separation method called GJK-EPA, which follows from GJK (Gilbert-Johnson-Keerthi) [4] and EPA (Expanding Polytope Algorithm) [5], is introduced to split the space between the obstacles and this formation, as depicted in Fig B2. If the predicted formation has no contact with this obstacle, the GJK algorithm is activated to separate the space. Otherwise, the EPA algorithm is introduced. Concretely, GJK algorithm is utilized to find the maximum separation distance between two separated convex polygon. The core mechanism is to find a support direction which separates these two polygons in a iterative way. When the final separation distance is obtained, then the corresponding support direction will be chosen as the final separating direction in this work. On the other hand, if these two polygon are contacted, EPA algorithm is utilized to find the minimum penetration distance between them. Similar to the GJK method, it also adopts the support direction in a iterative way and the separating direction is the support direction with the minimum penetration distance. Once the aforementioned separating direction is obtained, the split plane is chosen as the one vertical to the direction and across the support point of this obstacle. It is worth noting that whether or not the predicted formation is collided, the formulated predicted zone is a collision-free convex polygon.

Appendix B.5 Formation Optimization

The formation optimization is realized by solving the following optimization:

$$\begin{aligned}
\min_{\overline{\psi}, p^i_{\text{target}}, w} \quad & ||p^0_{\text{target}} - p_{\text{tractive}}||_2^2 + \alpha_w w^2 \\
& + \sum_{i=1}^{N} ||\mathcal{R}(\overline{\psi}) r^i + p^0_{\text{target}} - p^i_{\text{target}}||_2^2, && \text{(B3a)} \\
\text{s.t.,} \quad & a_m^{\text{zone}\mathrm{T}} p^i_{\text{target}} + b_m^{\text{zone}} + w \geqslant 0, m \in \mathcal{M}, && \text{(B3b)} \\
& a_m^{\text{pre}\mathrm{T}} p^i_{\text{target}} + b_m^{\text{pre}} + w \geqslant 0, m \in \mathcal{M}, && \text{(B3c)} \\
& w \geqslant 0 && \text{(B3d)}
\end{aligned}$$

where p^i_{target} is the temporary target position of agent i; in particular, p^0_{target} is the one for the formation's center. Similarly as in equation (B2), the term $\mathcal{R}(\overline{\psi})$ is replaced by $\mathcal{R}(\overline{\psi}) = \mathcal{R}(\psi) + \frac{\partial \mathcal{R}(\theta)}{\partial \theta}|_{\theta=\psi}(\overline{\psi} - \psi)$.

In the cost function of the above optimization, the term $||p^0_{\text{target}} - p_{\text{tractive}}||_2^2$ is added to compel the group towards the tractive point p_{tractive} and $\sum_{i=1}^{N} ||\mathcal{R}(\overline{\psi}) r^i + p^0_{\text{target}} - p^i_{\text{target}}||_2^2$ is used to penalize the deformation from the pursued formation. Constraints (B3c) and (B3b) are utilized to restrict their temporary target p^i_{target} in a feasible space.

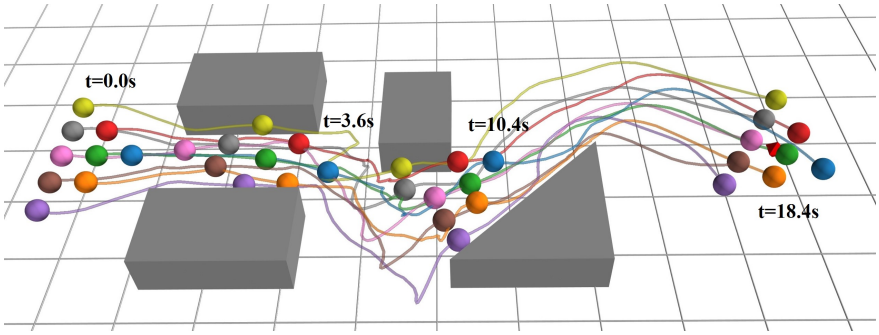

Figure D1 The trajectories of the agents in Case 1. The red cubic is the target position.

Appendix C Proof of Claim 1

Claim 1 is shown, if it can be proven that all the optimizations with strictly hard constraints such as (A2), (A4) and optimization (1) in the paper is recursively feasible, i.e., if in the last replanning, these optimizations are feasible, then in a new iteration they are feasible as well. As the trajectory derived from Optimization (1) is feasible, a collision-free and connectivity-preserving trajectory can be planned, and the proof is completed.

We will prove that via the proposed method, if in the last time $t-h$ feasible trajectories can be planned by all the agents, then the new feasible trajectories of all the agents can be derived in time t. Here, a feasible trajectory means that the planned trajectory satisfies all the hard constraints in Optimization (1) in the paper. More specifically, we will prove that for agent $i \in \mathcal{N}$, $x_k^i(t) = x_{k+1}^i(t-h)$, $u_{k-1}^i(t) = u_k^i(t-h)$, $k \in \mathcal{K}$ is a feasible solution for Optimization (1) (in the paper) in time t, where we enforce that $x_K^i(t) = x_K^i(t-h)$ and $u_{K-1}^i(t) = \mathbf{0}_d$. In the sequel, we set that $x_k^i(t) = x_{k+1}^i(t-h)$ and $u_{k-1}^i(t) = u_k^i(t-h)$.

If in time $t-h$ feasible trajectories can be planned, then by definition $x_k^i(t-h)$ and $u_{k-1}^i(t-h)$ satisfy all the constraints in Optimization 1. Then, we can easily derive that the following constraints can be satisfied:

$$\begin{aligned}
&x_k^i = \mathbf{A}x_{k-1}^i(t) + \mathbf{B}u_{k-1}^i(t),\\
&\|\Theta_a u_{k-1}^i(t)\|_2 \leqslant a_{\max},\ k \in \mathcal{K},\\
&\|\Theta_v v_k^i(t)\|_2 \leqslant v_{\max},\ k \in \mathcal{K},\\
&v_K^i(t) = \mathbf{0}_d.
\end{aligned}$$

Meanwhile, as already shown in [1], the constraint

$$a_k^{ij\mathrm{T}} p_k^i(t) \geqslant b_k^{ij}(t),\ \forall j \neq i,\ k \in \mathcal{K},$$

is met as well. Therefore, only the following two constraints:

$$a_m^{\mathrm{zone}}(t)^T p_k^i(t) \geqslant b_m^{\mathrm{zone}}(t), m \in \mathcal{M}, k \in \mathcal{K}, \tag{C1}$$

and

$$\left\|\overline{p}_k^i(t) - p_c^{\mathrm{commu}}(t)\right\|_2 \leqslant d_c, k \in \mathcal{K}, \tag{C2}$$

remain to be concerned. Since $a_m^{\mathrm{zone}}(t-h)^T p_k^i(t-h) \geqslant b_m^{\mathrm{zone}}(t-h), m \in \mathcal{M}, k \in \mathcal{K}$ holds, it is clear that the convex hull of $p_k^i(t-h)$, $k \in \mathcal{K}$, $i \in \mathcal{N}$ is restricted in a collision-free convex polygon. This indicates that this convex hull must be collision-free. Then, according to the well-known separating hyperplane theorem, by adopting Optimization (A2), the separated plane must exist. This means that a safe zone can be derived. Moreover, evidently $p_k^i(t-h)$, $k \in \mathcal{K}$, $i \in \mathcal{N}$ is included in this new safe zone as this constraints is inherently considered in Optimization (A2). Thus, the constraint (C1) can be satisfied. In a similar way, we can easily prove that the constraint (C2) is satisfied as well. In summary, we have proven that if in the last time $t-h$ feasible trajectories can be planned by all the agents, then the new feasible trajectories of all the agents can be derived in time t.

If the agents are initially collision-free and connected, it is evident that the initial safe and communication sphere can be derived and the initial resolution $x_k^i(0) = x^i(0)$, $u_{k-1}^i = 0$, $k \in \mathcal{K}$ satisfy all the constraints. Since the initial optimizations as well as these successive ones are feasible in a recursive way, the enforced constraints are satisfied during the entire motion. By satisfying these constraints, the agents are clearly collision-free and connected. Thus, the proof is completed.

It should be mentioned that the above proof is inspired by our previous work [2]. Detail the differences are the constraints (C2) and (C1) which are newly introduced ones.

Appendix D Numerical Simulations

The algorithm is implemented in Python3 programming language. All tests are performed on a computer with Intel Core i5 @2.4GHz 16GB RAM and the distributed execution of trajectory planning is simulated using multiprocessing. CVXOPT [6] and CVXPY [7] are used for solving the trajectory optimization and the formation optimization problems, respectively, and OMPL [8] for ABIT* path planning. In real implementations, we restrict the runtime of ABIT* within 0.05s. Moreover, to mimic the quadrotors implemented in later hardware experiments, the maximum acceleration and velocity of the underlying agents are set as 1.0m/s^2 and 1.0m/s, respectively. The communication range is determined as $d_c = 4.0$m. The replanning interval for MPC is set to the sampling time $h = 0.2$s. Additionally, the safety radius of an agent is set to $r_a = 0.3$m.

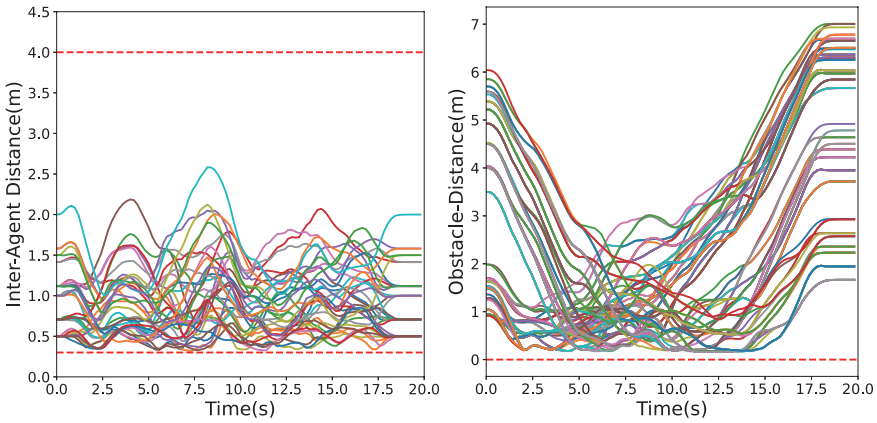

Figure D2 **Left:** the distance between each pair of agents. **Right:** the distance between LOS of each pair of agents and obstacles.

Appendix D.1 Case 1

In this case, a triangle-like formation is required to go through a corridor constituted by four obstacles. The map ranges from $(0.0\text{m}, 0.0\text{m})$ to $(12.0\text{m}, 4.0\text{m})$. The result can be found in Fig. D1, and the makespan of this task is 18.4s. During the motion process, at time $t = 3.6$s, the formation shrinks its width to enter the narrow passage between two rectangle. Meanwhile, the formation begins to turn right to avoid the upcoming obstacle. Then, the formation turns left to pass through the passage between the triangle and rectangle obstacles at time $t = 10.0$s. Thereafter, the swarm turns right and moves straight to the final target. In this process, we can find that the swarm can properly adapt its formation and choose a tailored attitude to pass through several passages. As shown in Fig. D2, since the inter-distance between agents are larger than $2r_a = 0.3$m, the inter-agent collisions are avoided. Furthermore, the connectivity can be maintained as the inter-agent distance is restricted within $d_c = 4.0$m and the distance between LOS of any pair of agents and obstacles is larger than 0.3m.

Appendix D.2 Case 2

This case is the one demonstrated in Fig. 1 of the paper. The details of the computation time cost is provided here. From Fig. D3, the time costs for calculating the safe zone and the connection sphere are less than 20ms, which can be neglected. And the runtime of formation adaptation ranges from 30ms to 140ms and the average time cost is 54ms, which can be evaluated online. Evidently, almost 90% of the runtime is taken by the trajectory planning of the agents. Fortunately, in real implementation, this computation can be executed in parallel by each agent, meaning that it can be decreased dramatically. The above analysis shows that the proposed method has a high performance handling a large-scale swarm.

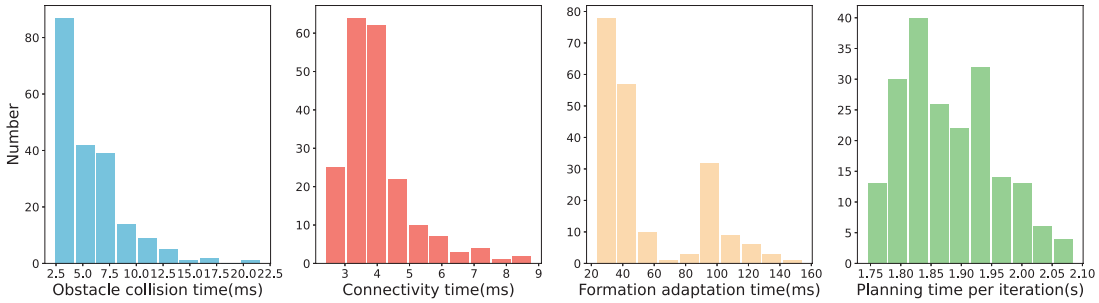

Figure D3 From left to right is the runtime of different parts in a replanning. They are the timespan taken by optimizations (A2) and (A4), formation adaptation, and the overall replanning.

Appendix E Experimental Validations

Appendix E.1 Hardware Platform

The layout of our laboratory is depicted in Fig. E1. Our experimental equipment consists of the following three parts:

• The Optitrack system is a set of real-time local positioning systems with precision of +/- 0.2mm even across large tracking areas. The Optitrack system consists of ten high-precision cameras with native frame rate of 120FPS and a Motive software installed on a Windows 10 personal computer to provide data integration function. Through proper initialization and calibration, we can use the Optitrack system to achieve high-accuracy real-time positioning of dynamic targets.

• The Crazyflie is a nano-quadrotor, which only weighs 27g and is less than 15cm long. It is worth noting that, in real implementation, to consider the inter-agent aerodynamic influence, the safety region of each quadrotor is enlarged and represented by a circle of radius $r_a = 0.17$m. Moreover, its maximum velocity and acceleration is set as 1.0m/s and 1.0m/s^2 respectively. It is equipped with low-latency and long-range radio as well as Bluetooth LE, which allow us to steer it with Python scripts and a remote controller. The onboard Li-Po battery can achieve about 5 minutes flying. Due to its small size and light weight, we can use a lot of them to carry out indoor experiments in a small space. Furthermore, during flying, each Crazyflie receives its own position information captured by the Optitrack in addition to the motion command set out by the workstation. Specifically, the motion commend is the planned trajectory encoded by 7_{th}-order polynomial. After receiving state and position information, Kalman filter

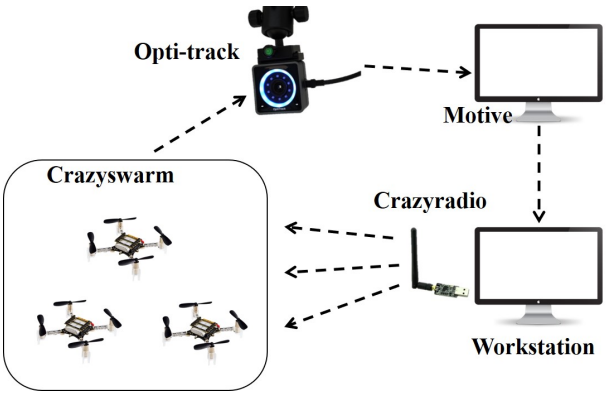

Figure E1 The architecture of the hardware platform.

Figure E2 **Left**: the testing ground. **Right**: the computers for Motive (left) and the workstation (right).

is used to estimate the current velocity and position and, then, a feedback controller is used to track the updated trajectory based on the method in [10]. In the following experiments, to balance the dynamic performance and computation cost, the sampling time h is set to 0.2s and the horizon length is chosen as $K = 8$.

• Robot Operating System (ROS) is used to construct the experimental platform. The flight control experimental system is elucidated in Fig E1. The Optitrack captures the Crazyflies' position information and sends it to the a computer, where Motive is installed. The Motive software integrates the data and broadcasts the information to the workstation in ROS. Python scripts are run on the workstation utilizing position information of Crazyflies broadcast by Motive. Finally, control commands for Crazyflies are broadcast from the workstation accordingly via the Crazyradio, which is a long range open USB radio dongle and capable of 2.6-GHz radio communication.

Appendix E.2 More Results

The measurements of the real fly states captured by Optitrack are illustrated in Fig. E3, from which it can be observed that inter-agent collision and LOS collision do not happen in any of these three scenarios. This indicates not only the safety of the underlying nano-quadrotors but also the effectiveness of the proposed method.

References

1 Chen Y, Guo M, Li Z. Deadlock resolution and recursive feasibility in MPC-based multi-robot trajectory generation. arXiv preprint, 2022, 2202.06071.
2 Chen Y, Guo M, Wang C et al. Multi-agent trajectory planning with feasibility guarantee and deadlock resolution: an obstacle-dense environment. IEEE Robotics and Automation Letters, 2023, 8:2197-2204.
3 Strub M, Gammell D. Advanced BIT (ABIT): sampling-based planning with advanced graph-search techniques. IEEE International Conference on Robotics and Automation, Pairs, 2020. 130-136.
4 Gilbert E, Johnson D, Keerthi S. A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE Journal on Robotics and Automation, 1988, 4:139-203.
5 Strub M, Gammell D. Proximity queries and penetration depth computation on 3d game objects. Game developers conference, San Jose, 2001.
6 Martin A, Joachim D, Lieven V. CVXOPT. In: http://cvxopt.org/, Website.
7 Steven D, Stephen B. CVXPY: A Python-embedded modeling language for convex optimization. Journal of Machine Learning Research, 2016, 17:1-5.
8 Ioan A, Mark M, Lydia E. The open motion planning library. IEEE Robotics Automation and Magazine, 2012, 19:72-82.
9 Preiss J, Hönig W, Sukhatme G, et al. Crazyswarm: a large nano-quadcopter swarm. IEEE International Conference on Robotics and Automation, Singapore, 2017, 3299-3304.
10 Mellinger D, Kumar V.Minimum snap trajectory generation and control for quadrotors. IEEE International Conference on Robotics and Automation, Shanghai, 2011, 520-2525.

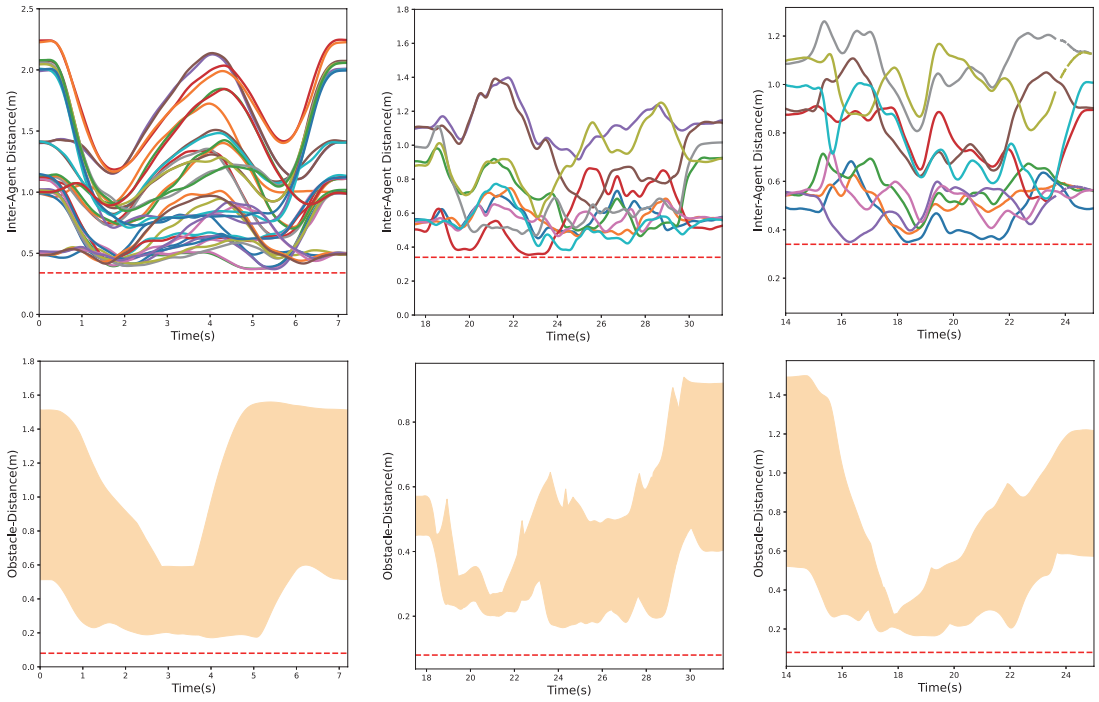

Figure E3 From left to right are the results of inter-agent distance (top) and LOS-obstacle distance (bottom) for Scenarios 1, 2, 3 in the paper, respectively. The red line denotes the minimum inter-agent distance, i.e., $2r_a$.