IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Agile Trajectory Planning and Large Obstacle Avoidance for Formation Flight using a Virtual Core

Jingsen Zhang, Biao Hou, *Senior Member, IEEE,* and Rui Huang

***Abstract*—Current methods for formation flight primarily focus on maintaining formations, often neglecting the swarm's agility. Furthermore, most of these approaches fail to leverage global information from the swarm for obstacle avoidance, making them incapable of generating efficient and safe trajectories in large obstacle scenarios. To address these limitations, this letter proposes a novel swarm trajectory planning framework that utilizes a virtual core to control the swarm. We employ virtual core penalties and dynamic maximum speed allocation to strike a balance between swarm flexibility and formation keeping, allowing the drones to avoid obstacles more smoothly and safely while maintaining formation stability. For large obstacle avoidance, we design a collaborative large obstacle boundary search strategy and a global swarm planning method to enable the rapid and safe generation of drone trajectories . To validate the performance of the proposed method, we develop a comprehensive set of experimental scenarios that include both simulations and real-world environments. The experimental results confirm the effectiveness of our approach.**

***Index Terms*—Formation flight, trajectory planning, aerial swarms, large obstacles, virtual core.**

I. Introduction

In recent years, drones have attracted increasing attention from researchers and have been extensively utilized in precision agriculture [1], area surveillance [2], among others. In certain scenarios, such as search and rescue missions in forest environments, single drones often face limitations including insufficient payload capacity and restricted field of view for reconnaissance during task execution. These constraints necessitate formation flight with obstacle avoidance among multiple drones to collaboratively accomplish the task.

There have been many existing works on swarm formation flight, which can be categorized into several approaches, including leader-follower [3], artificial potential fields [4], [5], virtual structures [6], [7], consensus-based algorithms [8], and optimization-based methods [9], [10]. Chen et al. [4] propose a robust leader-follower control scheme that effectively reduces the impact of external disturbances on the swarm system through a robust controller. Pan et al. [11] propose a method based on the artificial potential field, which effectively mitigates the issue of drones getting trapped in local minima during formation flight by improving the artificial potential function. Lagunas-Avila et al. [12] integrate the artificial potential field method with the leader-follower approach to achieve obstacle avoidance and formation flight for drones. Liu et al. [13] addresses the formation control problem for drone swarm systems under switching topologies by proposing a robust control framework composed of position and attitude controllers. However, the aforementioned algorithms are unable to generate smooth and continuous motion trajectories for drones and lack comprehensive and effective real-world validation.

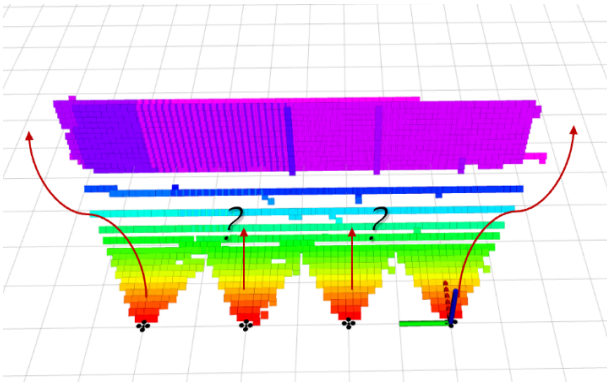

Fig. 1. Illustration of the Influence of large obstacles on drone swarm. The visibility of the two drones in the middle is severely obstructed, increasing the difficulty and risk of path planning. The drones on either side will navigate in opposite directions, disrupting the formation of the swarm.

EGO-swarm [14] is recognized as a seminal work in the field of optimization-based methods, which for the first time achieved swarm navigation in real-world dense obstacle environments. This breakthrough has greatly accelerated the development of optimization-based swarm planning methods, and our work also builds upon this foundation. However, EGO-swarm does not provide a satisfactory solution for formation keeping. In addressing formation control, Quan et al. [15] introduce a differentiable graph-theory-based cost function to measure the similarity of formations, achieving formation flight in dense obstacle environments. But the formation constraints of this method limit the flexibility of the drones. Peng et al. [16] propose a perception-shared method that allows each drone to utilize the global perspective of the swarm. Yet, this method relies on the exchange of map information across the entire swarm, which significantly increases the communication and computational burdens.

In addition to the limitations mentioned before, the formation constraints proposed by most methods have a significant impact on the flexibility of the drones. Furthermore, none of the current algorithms have the capability of large obstacles

Manuscript received: January 24, 2025; Revised: March 22, 2025; Accepted: June 19, 2025.

This paper was recommended for publication by Editor Olivier Stasse upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported in part by the National Natural Science Foundation of China under Grant 62171347, 62101405, 62371373; 111 Project.

All authors are with School of Artificial Intelligence, Xidian University, Xi'an 71000, China 22171110630@stu.xidian.edu.cn, houbiao@mail.xidian.edu.cn, 22171214683@stu.xidian.edu.cn.

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

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

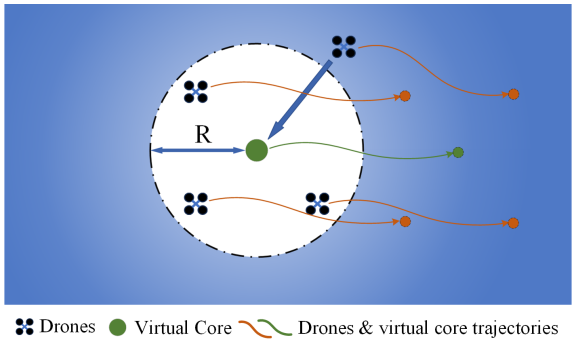

Fig. 2. Illustration of the virtual core penalty effect. The intensity of the blue color represents the strength of the virtual core's attraction.

avoidance. As shown in Fig. 1, large obstacles will severely obstruct the drones' FOV and disrupt the consistency of their decision-making. Worse still, the drones are unable to find safe paths under the formation constraints, leading to collisions with obstacles.

To address the gap, we propose a novel distributed drone swarm formation flight framework. We employ a non-physical virtual core to control swarm and generate virtual core trajectory to assist in drone path planning. To maintain cohesion, we apply penalties to drones that deviate too far from the virtual core, without affecting those within the threshold distance, ensuring their flexibility. In addition, we use dynamic maximum speed allocation to further adjust the drones' formation positions without compromising obstacle avoidance performance. During flight, each drone will detect whether the virtual core trajectory encounters any large obstacles. When a large obstacle is detected, all drones will collaborate to search for the boundary points of the obstacle. Utilizing these points, the drones are guided to re-plan their own trajectories, achieving safe and consistent large obstacle avoidance without the need for map exchange. To the best of our knowledge, our method is the first to be used for large-scale obstacle avoidance in formations.

The contributions of this work are summarized as follows:

1) We propose a novel formation flight framework that achieve formation flight through virtual core penalty and dynamic maximum speed allocation, while ensuring the flexibility of the swarm.
2) Without the need for map exchange, we equip the swarm with the capability to detect and avoid large obstacles, ensuring the consistency and safety of the drones' trajectories.
3) We validate the proposed method in both simulated and real-world experimental scenarios. Comparative analysis with other algorithms [14], [15] demonstrates the effectiveness of the proposed method.

II. Agile Swarm Formation Flight

A. Virtual Core Trajectory

Consider a swarm composed of N drones indexed by $i \in \{1,2,...,N\}$, and the position of each drone at time t is denoted as $\mathbf{p}_i(t) \in \mathbb{R}^3$. The goal for each drone is to move from its starting position $\mathbf{p}_i(t_s)$ to its end position $\mathbf{p}_i(t_e)$. The virtual core, being non-physical, symbolizes the central motion of the swarm, and its position at time t is denoted as $\mathbf{p}_v(t)$. The start position $\mathbf{p}_v(t_e)$ and end position $\mathbf{p}_v(t_e)$ of the virtual core are calculated using the following formula:

$$\mathbf{p}_v(t_s) = \frac{1}{N}\sum_i^n \mathbf{p}_i(t_s), i = \{1,2,...,N\} \tag{1}$$

$$\mathbf{p}_v(t_e) = \frac{1}{N}\sum_i^n \mathbf{p}_i(t_e), i = \{1,2,...,N\} \tag{2}$$

Following the approach of [14], the trajectory of the virtual core is parameterized by a uniform B-spline curve $\Phi_\mathbf{c}$. A uniform B-spline is piecewise polynomial uniquely determined by its degree p_b, a set of N_c+1 control points $\{Q_0, Q_1, ..., Q_{N_c}\}$ and knot vector $[t_0, t_1, ..., t_M]$, in which $Q_i \in \mathbb{R}^3$, $t_m \in \mathbb{R}$ and $M = N + p_b + 1$. The decision variables are control points Q of the B-spline curve in a reduced space of differentially flat outputs. Since the virtual core is not a physical entity, the trajectory of it does not consider obstacle collisions. The optimization problem is then formulated as follows:

$$\min_{\mathbf{Q}} J_{core} = \lambda_s J_s + \lambda_d J_d \tag{3}$$

where J_s is the smoothness penalty and J_d is the feasibility penalty. J_s is formulated as the time integral over square derivatives of the trajectory. J_d is ensured by restricting the higher order derivatives of the trajectory on every single dimension. λ_s, λ_d are weights for each penalty terms. The virtual core trajectory can be generated by any drone and influences the global flight of the swarm. Thanks to the virtual core's non-physical nature, it does not need to account for most obstacles (except for large ones), so the trajectory does not require frequent re-planning, thereby ensuring the stability of the swarm. The virtual core trajectory will be published to all drones once the initialization is complete.

B. Virtual Core Penalty

The virtual core begin to move simultaneously with the drones, and it generates a clustering area with a radius of R. We construct the virtual core penalty function J_v using the following formula:

$$J_v = \int_{t=t_s}^{t_e} \begin{cases} (d_i(t) - R)^2 \\ 0 \end{cases} dt, \begin{matrix} d_i(t) > R \\ d_i(t) \le R \end{matrix} \tag{4}$$

$$d_i(t) = \|\mathbf{p}_i(t) - \mathbf{p}_v(t)\| \tag{5}$$

where $||\cdot||$ denotes the Euclidean norm. When the distance between a drone and the virtual core exceeds the threshold R, a virtual core penalty is applied, realigning the drone's trajectory back within the virtual core's constraints. The radius R must greater than $\xi_{\max} = \max(||\mathbf{p}_i(t_s) - \mathbf{p}_v(t_s)||), i = 1,2,...,N$, which represents the maximum distance between the virtual core and the drones in the initial formation; otherwise, the planning will fall into chaos upon initialization. Beyond this threshold, a higher value of R enhances the swarm's flexibility but compromises its formation-keeping capability, allowing adaptive reconfiguration based on mission requirements. Furthermore, the virtual core's constraint boundary can be

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

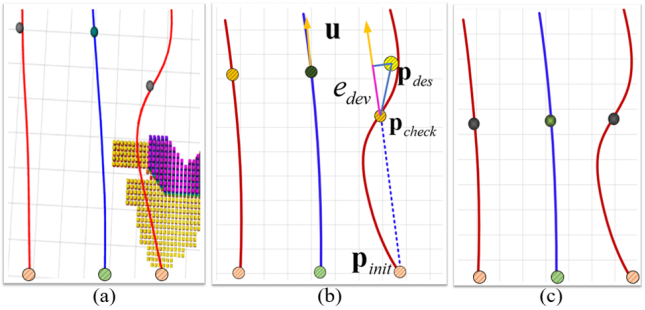

Fig. 3. (a) The position of a drone at t_{check} is behind the virtual core. (b) Illustration of the computational process of dynamic maximum speed allocation. (c) After using the dynamic maximum speed allocation method, the positions of the virtual core and the drones at t_{core}.

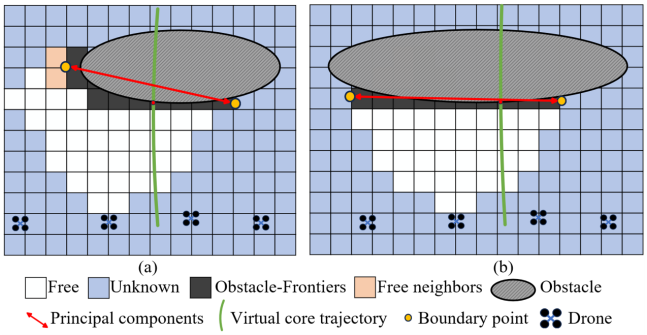

Fig. 4. Illustration of large obstacle detection. (a)The second drone detects a collision on the virtual core trajectory and finds free neighbors around the collision point. (b)The second drone is unable to find any free neighbors.

designed into arbitrary geometries (e.g., rectangular or triangular configurations) through modifications to the governing constraint equations.

The trajectories of drones are also represented by uniform B-spline curve. In addition to smoothness penalty J_s and feasibility penalty J_d, the optimization terms of drones J_{drone} are increased by obstacle Collision penalty J_c, reciprocal collision penalty J_w, and virtual core penalty J_v:

$$\min_{\mathbf{Q}} J_{drone} = \lambda_s J_s + \lambda_d J_d + \lambda_c J_c + \lambda_w J_w + \lambda_v J_v \tag{6}$$

J_c pushes control points away from obstacles, J_w prevents collisions between drones by checking for spatiotemporal intersections in their trajectories [14], and J_v constrains control points within the vicinity of the virtual core. Each drone completes its trajectory planning under the combined influence of these constraints.

C. Dynamic Maximum Speed Allocation

The virtual core allows drones to maintain cohesion, but it does not provide the capability to further refine the formation positions of drones within the constraint range. During flight, drones may gradually fall behind the virtual core due to the influence of obstacles (since the virtual core trajectory does not account for them), and this deviation can accumulate over time. To ensure that drones are capable of handling such situations, we design the dynamic maximum speed allocation algorithm, as illustrated in Fig. 3.

First, we have the drones periodically check if there is a relative position deviation between them and the virtual core at the future check time $t+T_d$. Based on the initial formation and the position of the virtual core at check time, we can obtain the desired position $\mathbf{p}_{i,des}$ of drone i at that time. Then, we define the position deviation error $e_{i,dev}$ of drone i using the following formula:

$$e_{i,dev} = ||\mathbf{u} \cdot (\mathbf{p}_{i,des}(t+T_d) - \mathbf{p}_i(t+T_d))|| \tag{7}$$

where $\mathbf{u}$ is the unit vector of the virtual core's motion direction at check time. The physical meaning of $e_{i,dev}$ is the projection of the distance between the actual and desired positions of the drone onto $\mathbf{u}$. The projection angle is denoted as θ. We use $e_{i,dev}$ to measure the displacement of the drone along the swarm motion direction, without considering the displacement caused by obstacle avoidance on the plane normal. This will inevitably result in some errors in formation maintenance, but it will enhance the drone's flexibility and safety during obstacle avoidance.

Since the planning of the virtual core trajectory does not consider obstacles, the trajectory exhibits a state that is approximately linear. By leveraging this property, we can approximate the straight line between $\mathbf{p}_i(t)$ and $\mathbf{p}_i(t+T_d)$ as a virtual core trajectory and compute the time required $T_{i,core}$ to traverse this trajectory:

$$T_{i,core} = \|\mathbf{p}_i(t) - \mathbf{p}_i(t+T_d)\| / \mathrm{v}_{c,\mathrm{arg}} \tag{8}$$

where $\mathrm{v}_{c,\mathrm{arg}}$ represents the average speeds of the virtual core during T_d. Then we obtain the desired maximum speed $v^{des}_{\max}$ using the following formula:

$$v^{des}_{i,\max} = \begin{cases} \min(\sigma \mathrm{v}_{i,\max}, \mu_{\max}) & e_{i,dev} > \tau, \theta \le \pi/2 \\ \max(\sigma \mathrm{v}_{i,\max}, \mu_{\min}) & e_{i,dev} > \tau, \theta > \pi/2 \\ \mathrm{v}_{\max} & others \end{cases} \tag{9}$$

$$\sigma = (L/T_{i,core})/\mathrm{v}_{i,\mathrm{arg}} \tag{10}$$

where $\mathrm{v}_{i,\max}$ is the original maximum speed of drone i, $\mu_{\max}$ and $\mu_{\min}$ are the upper and lower limits of speed, σ represents the degree of change of $\mathrm{v}_{i,\max}$, $\mathrm{v}_{i,\mathrm{arg}}$ represent the average speeds of drone i, L is the length of the drone's trajectory, τ is the threshold constant. We use θ to determine whether the drone is positioned in front of or behind the virtual core, thereby determining whether its maximum speed needs to be increased or decreased.

Since the trajectories of drones are represented using a uniform B-spline, each knot span has identical value of Δt. After obtaining $v^{des}_{i,\max}$, we recalculate knot span by $\Delta t' = \Delta t(v_{i,\max}/v^{des}_{i,\max})$. Finally, we use $v^{des}_{i,\max}$ and $\Delta t'$ to re-plan the control points, resulting in the desired trajectory Φ^{des}_i.

III. Global Swarm Planning For Large Obstacles Avoidance

Most formation flight algorithms perform well when facing small obstacles. However, when an obstacle is large enough, two issues arise:

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Algorithm 1 Large Obstacle Detection

Notation: Obstacle-Frontiers OFs, Endpoints $\mathbf{p}_{\mathbf{e}_1}, \mathbf{p}_{\mathbf{e}_2}$, Virtual Core Trajectory $\Phi_{\mathbf{c}}$, detection interval Δs
Initialize:
$t \leftarrow$CurrentGlobalTime()
$\mathbf{p}_v(t) \leftarrow$GetDetectPoint($t$,$\Phi_{\mathbf{c}}$)
while *(!ReachEndPoint*($\mathbf{p}_v(t)$)) **do**
  if *IsOccupiedGrids*($\mathbf{p}_v(t)$) **then**
    $OFs \leftarrow$*GetObstacleFrontiers*($\mathbf{p}_v(t)$)
    if *ExistFreeNeighbors(OFs)* **then**
      $t_{jump} \leftarrow$ *JumpObstacle()*
      $t \leftarrow t_{jump}$
    else
      $\mathbf{p}_{\mathbf{e}_1}, \mathbf{p}_{\mathbf{e}_2} \leftarrow$ *GetEndPoints(OFs)*
      if $||\mathbf{p}_{\mathbf{e}_1} - \mathbf{p}_{\mathbf{e}_2}|| > \delta$ **then**
        return Large Obstacle
  $t \leftarrow t + \Delta s$
  $\mathbf{p}_v(t) \leftarrow$GetDetectPoint($t$,$\Phi_{\mathbf{c}}$)

- Vision Limitations: Large obstacles can obstruct the FOV of most drones in the swarm, preventing them from quickly finding a collision-free path.
- Decision-making Consistency: Large obstacles can significantly affect the consistency of drones' trajectories, thereby severely disrupting the swarm formation.

In response to these two challenges, we propose a global swarm planning approach for large obstacle avoidance.

A. Large Obstacle Detection

Since the virtual core trajectory reflects the movement of the swarm, we have All drones detect whether the virtual core trajectory intersects with any large obstacles, as shown in Fig. 4 and Alg. 1. The grids of a map have three states: unknown, occupied, and free. Starting from the current time, drones will evaluate each point on the virtual core trajectory at intervals of Δs to ascertain whether it crosses any occupied grids (*IsOccupiedGrids()*). When a collision is detected by a drone, it will follow the method for obtaining frontiers described in [17] to get obstacle-frontiers (OFs) around the collision point (*GetObstacleFrontiers()*). OFs are defined as occupied grids that are adjacent to either free or unknown grids. Next, we use Principal Component Analysis (PCA) to extract the principal direction of the OFs and check if there are free grids along the direction. We refer to these special free grids as free neighbors. If $ExistFreeNeighbors(OFs)$, it indicates that there exists a navigable areas around the obstacle. The drone ill skips the currently occupied connected regions *JumpCurrentObstacle()* and detects the remaining virtual core trajectory. On the contrary, it indicates that the current obstacle poses a potential risk of disrupting the swarm's formation flight. $GetEndPoints(OFs)$ derives two endpoints, $\mathbf{p}_{\mathbf{e}_1}$ and $\mathbf{p}_{\mathbf{e}_2}$, along the principal component direction of the OFs, and then determines whether the current obstacle is large by comparing the distance between them against the threshold δ. When $||\mathbf{p}_{\mathbf{e}_1} - \mathbf{p}_{\mathbf{e}_2}|| > \delta$, the obstacle is determined to be a large obstacle, and the process of collaborative large obstacle boundary search is initiated.

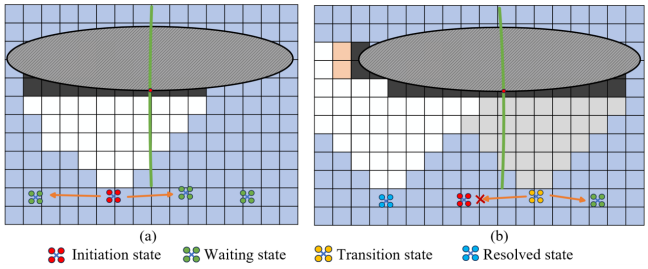

Fig. 5. Illustration of large obstacle boundary search. (a) The second drone sets itself to the initiation state and sends activation messages to the surrounding drones. (b) The first drone sets itself to the resolved state after finding free neighbors, while the third drone attempts to activate other drones and sets itself to the transition state.

B. Collaborative Large Obstacle Boundary Search

We utilize a finite state machine to implement the search process, as shown in Fig. 5. The states are divided into waiting state, initiation state, transition state, and resolved state. At first, all drones are in the waiting state. When a drone initiates the large obstacle boundary search process, it changes its own state to the initiation state, and then sends the activation message that contains endpoints and initiation time to the nearest drone along the principal component directions. The drones that receive the activation message will change their status to the transition state and record the initiation time. After that, they will generate OFs around the endpoint in their own map and search for free neighbors. If the drone still cannot find free neighbors, it will continue to send an activation message to other drones along the new principal directions.

Once a drone detects free neighbors, it marks the center point of the free neighbors as the boundary point $\mathbf{p}_b$. Then, it will change its status to resolved state and complete the re-planning of the virtual core trajectory. After that, the drone will send the new virtual core trajectory and $\mathbf{p}_b$ to all other drones and reset its status to waiting state. All drones that receive the message will use $\mathbf{p}_b$ and the new virtual core trajectory to re-plan their paths and change their status to waiting state after the re-planning is finished. The re-planning process for the virtual core and the drones will be detailed in Section III-C.

In certain cases, drones that are not in a waiting state may receive activation information again. We use the initiation time to resolve conflicts. Drones will compare their stored initiation time with the initiation time in the activation message. If the initiation time in the received message is earlier than their stored time, drones will update their stored initiation time and respond to the message. Otherwise, it will reject the activation message.

C. Global Swarm Planning

After obtaining the boundary points of the obstacle, we first re-plan the virtual core trajectory. The vector from the collision

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

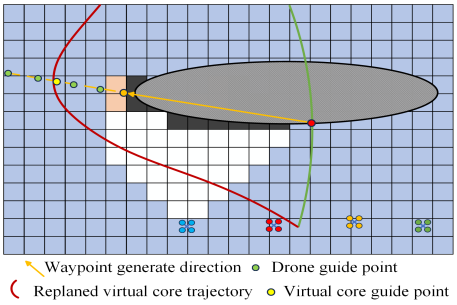

Fig. 6. Illustration of swarm trajectory planning. The drone in the resolved state generates a guide point and completes the trajectory planning for other drones and the virtual core.

point to the boundary point $\mathbf{p}_b$ is denoted as $\mathbf{v}_d$. Based on the direction of $\mathbf{v}_d$, a virtual core guide point $\mathbf{g}_c$ is placed at a distance l_c from $\mathbf{p}_b$. As for the value of parameter l_c, if it is set conservatively, the virtual core trajectory will be closer to the obstacle, increasing the risk of collision for the swarm. Conversely, if it is set to a larger value, the re-planned trajectory of the swarm may fall into unknown regions more. We set l_c equal to the virtual core penalty range R to ensure balance. Next, we introduce point $\mathbf{g}_c$ as an intermediate point between the current position and the goal position of the virtual core, generating a new virtual core trajectory.

After completing the replanning of the virtual core trajectory, we refer to the method described in [18] to solve the ALignment and task ASsignment (ALAS) problem to generate drone guide points $\mathbf{g}_i, i = 1,2,...,N$ around $\mathbf{g}_c$. Let the moment corresponding to the virtual core at the $\mathbf{g}_c$ be denoted as t_c. Based on the template of initial formation shape, the desired formation position of each drone around $\mathbf{g}_c$ is $\mathbf{p}_{i,des}(t_c)$. Then, the positions of drone guide points $\mathbf{g}_i$ can be written as:

$$\mathbf{g}_i = s \cdot \mathbf{p}_{i,des}(t_c) + \mathbf{d} \tag{11}$$

where $s \in \mathbb{R}$ is the scaling factor and $\mathbf{d} \in \mathbb{R}^3$ denotes the translation factor. The task assignment problem is formulated as:

$$\min_{\varphi} \sum_{i}^{n} ||\mathbf{p}_i(t_c) - (s^* \cdot \mathbf{p}_{\varphi(i),des}(t_c) + \mathbf{d}^*)||^2 \tag{12}$$

where $\varphi \in S_N$ is the assignment map of the formation task and S_N is the symmetric group of all permutations from the set $\{1,2,...,N\}$ to itself.The formation alignment problem is formulated as:

$$\min_{s,\mathbf{d}} \sum_{i}^{n} ||\mathbf{p}_i(t_c) - (s \cdot \mathbf{p}_{\varphi^*(i),des}(t_c) + \mathbf{d})||^2 \tag{13}$$

The work in [18] demonstrates that the two problems can be optimized in a decoupled manner. Therefore, we can first solve problem (13) to obtain the optimal φ^*. Subsequently, based on φ^*, we solve problem (14) to determine the best s^* and $\mathbf{d}^*$.Finally, the guide points of the drones surrounding the virtual core can be represented as:

$$\mathbf{g}_i = s^* \cdot \mathbf{q}_{\varphi^*(i),,des}(t_c) + \mathbf{d}^* \tag{14}$$

We insert the guide point $\mathbf{g}_i$ as an intermediate point between the current position and the goal position of drone i, and then generate new trajectories.

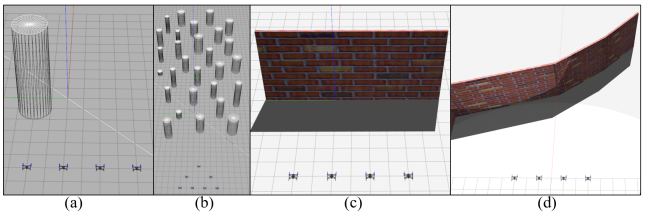

Fig. 7. Simulation experiment maps. (a) is the formation maintenance experiment map. (b) is the forest environment map. (c) and (d) are two kinds of large obstacle environment map.

If the obstacle is so large that all drones fail to find the free neighbors, we will initialize the farthest endpoint from the collision point as the guide point. At the same time, we will use dynamic maximum speed allocation to slow down the swarm's flight speed, as this planning is not globally optimal and is in close proximity to the large obstacle. During the exploration process, drones will continuously search for guide points using the updated map.

IV. Simulation And Experiment

A. Simulation Scenarios and Parameters

To validate the performance of the proposed method, we design three simulation experiments: the formation maintenance experiment, the forest environment fight experiment, and the wall obstacle environment fight experiment, as shown in Fig. 7. In all experiments, the constraint range R of the virtual core is set to $\xi_{\max} + 0.5m$, the weights $\lambda_s, \lambda_d, \lambda_c, \lambda_w, \lambda_v$ are set to 1, 0.1, 0.5, 0.5, 0.5, the large obstacle environment threshold δ is set to 4 m, the deviation error threshold τ for the dynamic maximum speed allocation method is set to 0.5 m, and T_d is set to 3s.

B. Formation Maintenance Experiment

In this experiment, four drones are sequentially spaced at 2 m intervals. We first set the maximum speeds of the four drones to 2 m/s, 1.5 m/s, 0.75 m/s and 0.5 m/s, and the maximum speed of the virtual core is set to 1.0 m/s. After the drones flight for two seconds along the positive x-axis, the dynamic maximum speed allocation algorithm is executed. The changes in position deviation error $e_{i,dev}$, speed, and flight trajectories during the flight are shown in Fig. 9. It can be observed that after the dynamic maximum speed allocation algorithm is executed, the drones flying too fast have their speeds limited, while the drones flying slower are continuously accelerated. The position deviation error of the drones converge to within 0.2 m after a few oscillations. In contrast, without the dynamic maximum speed allocation adjustment, $e_{i,dev}$ of drones gradually increased over time.

Next, we introduce an obstacle in front of the leftmost drone and set the speed of all drones to 1.5 m/s, while keeping the virtual core's speed at 1 m/s. The obstacle is a 2m x 2m cylinder, and we use it to observe the effect of the virtual core penalty on the drone. As shown in Fig. 10, under the influence of the virtual core penalty, the drone chooses the side closer to the virtual core when avoiding the obstacle and

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

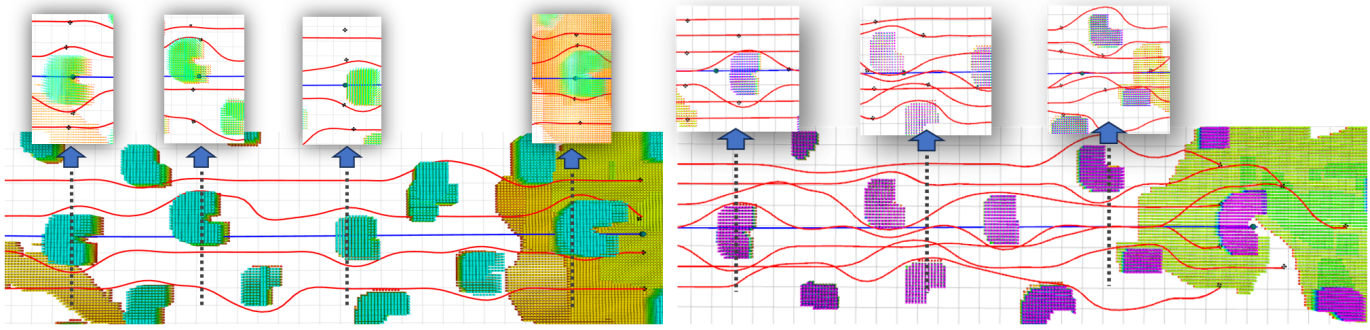

Fig. 8. The left figure illustrates trajectory plots of four drones arranged in a linear formation, while the right figure displays trajectory plots of seven drones arranged in a triangular formation. During the flight, the drones maintain a good linear formation around the virtual core.

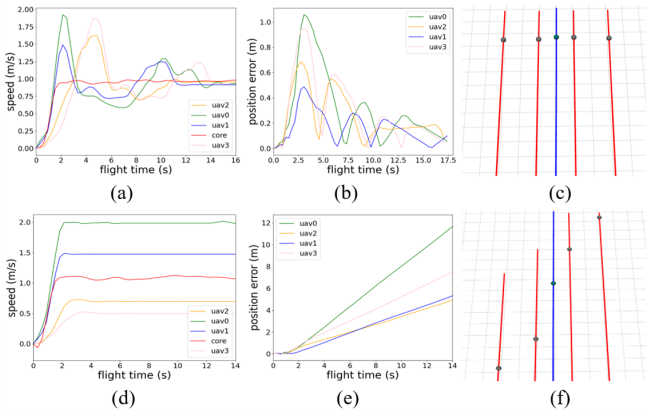

Fig. 9. (a-c) are flight results with the proposed algorithm. (d-f) are flight results without the proposed algorithm.

do not exhibit significant position deviation errors with the help of the dynamic maximum speed allocation. When the method is not executed, the drone chooses a planning direction that diverged from the swarm and significantly lags behind the other drones.

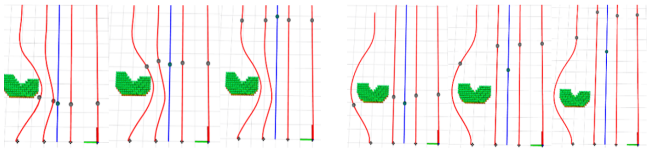

Fig. 10. Left figures are flight trajectories with the proposed algorithm. Right figures are flight trajectories without the proposed algorithm.

C. Forest Environment Fight Experiment

Next, we test the robustness of the proposed algorithm in a dense forest environment and compare it with EGO-swarm [14] and EGO-formation [15]. We first linearly positioned the four drones as described in the previous section, and conduct multiple experiments under different speed conditions, recording the trajectory time, success rate, trajectory length, and overall position deviation $\bar{e}_{dev}$ for each algorithm. The overall position deviation reflects the degree of formation maintenance of the swarm:

$$\bar{e}_{dev} = \frac{1}{NM}\sum_{i}^{N}\sum_{t_n=t_s}^{t_e} e_{i,dev}(t_n), n = 1,2,...,M \qquad (15)$$

TABLE I
FORMATION FLIGHT RESULTS IN FOREST ENVIRONMENT.

Drone Speed	method	Traj. Len(m)	Succ. Rate(%)	$\bar{e}_{dev}$(m)	Traj. Time(s)
1.0m/s	Zhou [14]	42.54	**100**	2.49	49.32
	Quan [15]	41.31	86.67	**0.27**	51.49
	Ours	**37.05**	96.67	0.35	**38.53**
1.5m/s	Zhou [14]	44.21	**96.67**	3.57	33.58
	Quan [15]	42.58	73.33	**0.54**	37.62
	Ours	**37.32**	90	0.62	**26.93**
2.0m/s	Zhou [14]	46.79	**93.33**	4.21	27.65
	Quan [15]	44.31	56.67	1.72	29.84
	Ours	**39.84**	86.67	**0.95**	**21.35**

where t_n represents the sampling moments between t_s and t_e. The left figure in Fig. 8 shows the flight trajectories of one execution of the proposed algorithm, while Table I presents the comparative results.

EGO-swarm, due to the absence of any formation constraints, possesses the highest degree of flexibility. It only occasionally experiences planning failures when irresolvable conflicts arise between inter-swarm collision avoidance and obstacle avoidance, which explains its superior success rate as demonstrated in Table 1. In contrast, EGO-formation introduces a scaling and rotational invariance formation metric matrix that imposes additional constraints on drone path planning, thereby exacerbating conflict probabilities and ultimately degrading success rate performance. The virtual core constraint we employ imposes a lesser impact on the drones, thereby achieving a success rate that closely approaches that of EGO-swarm. In terms of $\bar{e}_{dev}$, the performances of EGO-swarm and EGO-formation are inversely correlated with their success rates, whereas our algorithm remains close to the better-performing one. Notably, under the 2 m/s speed condition, the limited flexibility of EGO-formation impacts its formation keeping performance, causing its $\bar{e}_{dev}$ to lag behind the proposed algorithm. The result clearly demonstrates that our algorithm effectively balances formation constraints with the flexibility of the swarm. At the same time, the proposed algorithm achieved optimal results in both the trajectory length and the trajectory time. Especially in terms of trajectory time, dynamic maximum speed allocation method gives our algorithm an absolute advantage in terms of execution time

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

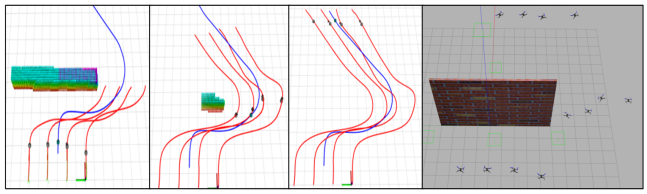

Fig. 11. Flight trajectories and drones' positions in gazebo of the proposed method in large obstacle environment.

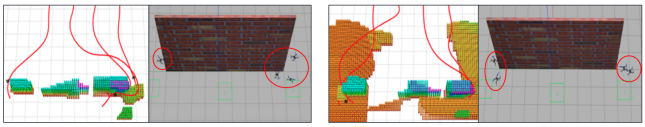

Fig. 12. Left figures are results of EGO-swarm. Right figures are results of EGO-formation.

compared to other algorithms. This also means that we can truly control the flight time and speed of the swarm by controlling the virtual core speed.

To further validate the generality of our proposed algorithm, we conducted flight experiments with seven aircraft in the same scenario. The initial formation of the swarm is shown in Fig. 7(b), with the drone speed set to 1.5 m/s. Under these experimental conditions, the success rate of the proposed algorithm reaches 86.67%, and the overall position deviation $\bar{e}_{dev}$ is 0.87 m. The experimental performance exhibits minimal degradation compared to the 4-drone configuration, demonstrating the algorithm's scalability with respect to the number of drones. The trajectories of the drones are shown in the right figure of Fig. 8, from which it can be seen that the swarm maintains a good formation while traversing the obstacles.

TABLE II
FLIGHT RESULTS IN LARGE OBSTACLE ENVIRONMENT.

Drone Speed	Obstacle Size(m)	Solution Time(ms)	Succ. Rate(%)	$\bar{e}_{dev}$(m)	Traj. Time(s)
1.0m/s	8	0.84	100	0.63	22.32
	10	0.88	93.33	0.85	23.59
	12	0.91	86.67	1.47	25.17
1.5m/s	8	0.85	96.67	0.91	15.10
	10	0.87	90	1.23	15.97
	12	0.93	83.33	1.94	17.03
2.0m/s	8	0.87	86.67	1.23	11.29
	10	0.84	73.33	1.48	11.73
	12	0.92	56.67	2.25	13.45

D. Large Obstacle Environment Flight Experiment

In this part of the experiment, we construct walls of 8 m, 10 m, and 12 m in width placed in front of the drones, while other testing conditions remained largely consistent with those in the forest environment experiments. Due to their inability to handle large obstacles, both EGO-swarm and EGO-formation exhibit chaotic path planning near the obstacles (as shown in Fig. 12) and fail to pass any experimental trials. The experimental data of our algorithm are recorded in Table II, where the solution time represents the duration from detecting a large obstacle to identifying its edge and completing the drone path planning. Fig. 11 illustrates the drone trajectories during one experimental trial.

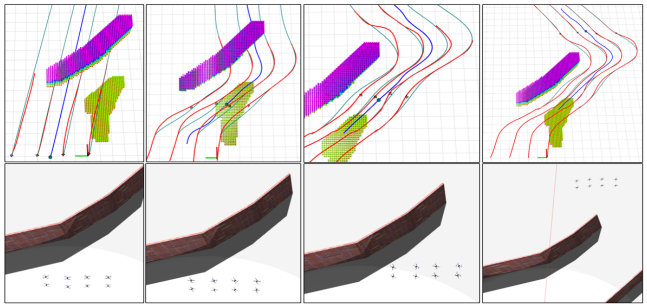

Fig. 13. Flight trajectories and drones' positions in gazebo of the proposed method in large obstacle environment.

As shown in Table II, the proposed algorithm achieves relatively satisfactory results under most conditions. However, as the width of the wall increases, the algorithm's success rate exhibits a noticeable decline. This is because the impact of obstacles on swarm formation and navigation intensifies as their size increases, causing the trajectories planned for edge drones to occasionally deviate from the virtual core constraint boundary during path planning. Additionally, by setting R=$\xi_{\max}+0.5$ m, we provide only 0.5m of free space for edge drones, which further amplifies the difficulty of achieving successful trajectory planning.

When confronted with larger obstacles, increasing R to trade for greater swarm flexibility would enhance the performance of our algorithm. To validate this view, we design a 17-meter-long wall scenario, increase R to $\xi_{\max}+1.0$ m, and set the drone's speed to 1.5 m/s. Fig. 13 demonstrates that in this scenario, the wall's excessive size prevents the swarm from detecting the obstacle boundary through a single collaborative search operation. The virtual core executes two replanning iterations before successfully navigating the swarm around the obstacle. The success rate of the experiment reaches 90%, demonstrating that our algorithm's enhanced flexibility significantly improves large obstacle handling capabilities. However, when the obstacle environment becomes highly complex (e.g., blind alley scenarios), the drone swarm struggles to achieve effective obstacle avoidance. This indicates that our algorithm's large-scale obstacle avoidance capability does not equate to autonomous maze navigation and exit discovery. Related experiment can be further examined in the supporting video material.

E. Real-world Experiments

Real-world experiments encompass two parts: indoor environment with large obstacles and outdoor forest environment. All drones utilized in real-world experiments are amov-p230, with Intel-NUC-12 as the onboard computers and Intel RealSense D435 as the cameras. In the outdoor forest environment, we utilize VINS [19] as the location algorithm, and simultaneously integrate UWB modules to mitigate drift in swarm localization [14]. The speeds of the drones and the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

virtual core are all set to 1.2 m/s. The experimental results are illustrated in Fig. 14.

In the indoor large obstacle experimental environment, for safety considerations, we employ a motion capture system to provide the drones with accurate position and attitude information. We construct a 6 m obstacle using foam boards, with one end connected to the wall. Initially, three drones are positioned directly facing the obstacle. The speeds of the three drones and the virtual core are all initialized to 0.5 m/s. The experimental results are illustrated in Fig. 15.

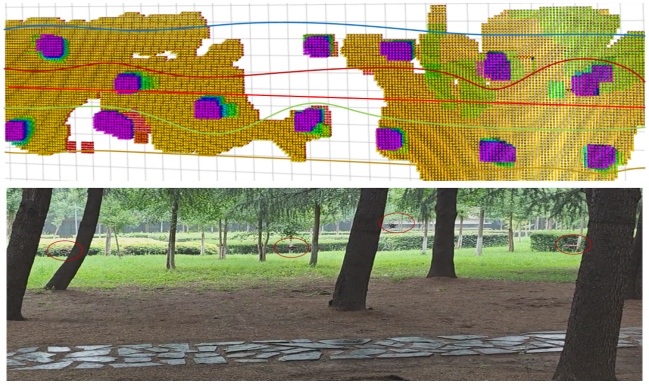

Fig. 14. Four drones real-world forest experiment. The bright red path indicates the virtual core trajectory.

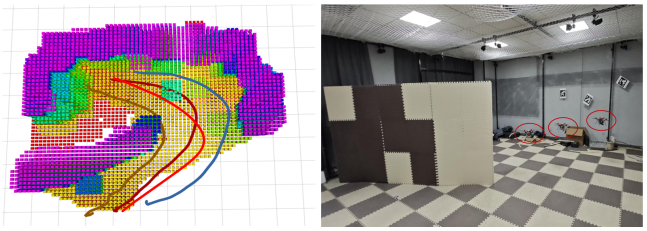

Fig. 15. Three drones real-world large obstacle experiment. The bright red path indicates the virtual core trajectory.

V. Conclusions

This paper proposes a virtual core-based swarm formation flight framework. It aims to balance the relationship between formation maintenance and swarm flexibility, and to endow the swarm with the ability to avoid large obstacles. By utilizing the virtual core, we can ensure smooth and safe swarm flight while achieving control over the swarm's flight trajectory and flight time. Simulation comparisons and real-world experiments validate the effectiveness of the proposed method.

The current method does not take into account the yaw of the drones, which leads to an overlap in the FOV between drones and limits the maximum size of obstacles that the swarm can detect. At the same time, the proposed algorithm is not yet capable of handling complex obstacles (such as blind alley and mazes). In the future, we plan to work on these issues and attempt to improve the planning success rate when facing large obstacles at high speeds.

References

[1] P. K. Singh and A. Sharma, "An intelligent wsn-uav-based iot framework for precision agriculture application," *Comput. Elect. Eng*., vol. 100, p. 107912, May 2022.

[2] J. Fan, L. Lei, S. Cai, G. Shen, P. Cao and L. Zhang, "Area Surveillance With Low Detection Probability Using UAV Swarms," *IEEE Trans. Veh. Technol*., vol. 73, no. 2, pp. 1736-1752, Feb. 2024.

[3] A. Dang and J. Horn, "Formation control of leader-following UAVs to track a moving target in a dynamic environment," *J. Autom. Control Eng*., vol. 3, no. 1, pp. 1–8, 2015.

[4] J. Sun, J. Tang, and S. Lao, "Collision avoidance for cooperative UAVs with optimized artificial potential field algorithm," *IEEE Access*, vol. 5, pp. 18382–18390, 2017.

[5] B.-S. Chen, C.-P. Wang, and M.-Y. Lee, "Stochastic robust team tracking control of multi-uav networked system under wiener and poisson random fluctuations," *IEEE Trans. Cybern*., vol. 51, no. 12, p. 5786–5799, Dec. 2021.

[6] D. Zhou, Z. Wang, and M. Schwager, "Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 916–923, 2018.

[7] Z. Sun and Y. Xia, "Receding horizon tracking control of unicycle type robots based on virtual structure," *Int. J. Robust Nonlin. Control*, vol. 26, no. 17, pp. 3900–3918, 2016.

[8] Y. Zhang, W. Feng, G. Shi, F. Jiang, M. Chowdhury, and S. H. Ling, "UAV swarm mission planning in dynamic environment using consensus-based bundle algorithm," *Sensors*, vol. 20, no. 8, p. 2307, 2020.

[9] C. E. Luis and A. P. Schoellig, "Trajectory generation for multiagent point-to-point transitions via distributed model predictive control," *IEEE Robot. Autom. Lett*., vol. 4, no. 2, pp. 375–382, 2019.

[10] S. H. Arul and D. Manocha, "DCAD: Decentralized collision avoidance with dynamics constraints for agile quadrotor swarms," *IEEE Robot. Autom. Lett*., vol. 5, no. 2, pp. 1191–1198, Apr. 2020.

[11] Z. Pan, C. Zhang, Y. Xia, H. Xiong, and X. Shao, "An improved artificial potential field method for path planning and formation control of the multi-uav systems," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 69, no. 3, pp. 1129–1133, 2022.

[12] J. Lagunas-Avila, R. Castro-Linares, and J. Alvarez-Gallegos, "Obstacle avoidance in leader-follower formation using artificial potential field algorithm," in *2021 18th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, 2021, pp. 1–6.

[13] H. Liu, Y. Wang, F. L. Lewis, and K. P. Valavanis, "Robust formation tracking control for multiple quadrotors subject to switching topolo- gies," *IEEE Trans. Control Network Syst*., vol. 7, no. 3, pp. 1319–1329, 2020.

[14] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, "EGO-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments," in *Proc. Int. Conf. Robot. Autom*., 2021, pp. 4101–4107.

[15] L. Quan, L. Yin, C. Xu, and F. Gao, "Distributed swarm trajectory optimization for formation flight in dense environments," in *Proc. Int. Conf. Robot. Autom*., 2022, pp. 4979–4985.

[16] P. Peng, W. Dong, G. Chen, and X. Zhu, "Obstacle avoidance of resilient UAV swarm formation with active sensing system in the dense environment," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*., 2022, pp. 10529–10535.

[17] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning," *IEEE Robot. Automat. Lett*., vol. 6, no. 2, pp. 779–786, Apr. 2021.

[18] L. Quan et al., "Robust and efficient trajectory planning for formation flight in dense environments," *IEEE Trans. Robot*., vol. 39, no. 6, pp. 4785–4804, Dec. 2023.

[19] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot*., vol. 34, no. 4, pp. 1004–1020, Aug. 2018.