Safety-assured High-speed Navigation for MAVs

Yunfan Ren[1†], Fangcheng Zhu[1†], Guozheng Lu[1], Yixi Cai [1], Longji Yin[1], Fanze Kong[1], Jiarong Lin[1], Nan Chen[1], Fu Zhang[1*]

[†] These two authors contributed equally to this work,

[1]Department of Mechanical Engineering, The University of Hong Kong,

The University of Hong Kong, Pokfulam, Hong Kong, China

[*]Fu Zhang; E-mail: fuzhang@hku.hk.

Micro air vehicles (MAVs) capable of performing high-speed navigation in unknown environments can benefit numerous applications, such as search and rescue or disaster relief, where reaching the target location safely and timely is crucial. However, achieving autonomous, safe, and high-speed MAV navigation faces systematic challenges, necessitating reduced vehicle weight and size for high-speed maneuvering, strong sensing capability for detecting obstacles at distance, and advanced planning and control algorithms maximizing flight speed while ensuring obstacle avoidance (i.e., safety). In this article, we propose a Safety-assUred high-sPeed aErial Robot (SUPER), which has a compact size of 280 mm wheelbase with a thrust-to-weight ratio over 5.0, allowing agile flights in cluttered environments. SUPER employs a lightweight 3D (three-dimensional) LiDAR (light detection and ranging) sensor to provide accurate, long-range obstacle detection. To enable high-speed flights while ensuring safety, we propose an efficient planning framework that plans two trajectories in each re-plan cycle, a trajectory in known-free spaces to ensure safety and a trajectory in both known-free and unknown spaces to boost the flight speed. Compared with the baseline method, the planning framework reduces the failure rate by 35.9 times while flying at a higher speed and taking only half of the planning time. The complete system, SUPER, is validated in real-world environments, achieving safe autonomous flights with speeds exceeding 20 m/s in the wild, avoiding thin tree branches, and navigating through narrow spaces that are not capable by state-of-the-art commercial drones. We demonstrate the applicability of SUPER in various MAV applications, including target tracking, autonomous exploration, and waypoint navigation. SUPER represents a milestone in autonomous MAV navigation, propelling high-speed and robust autonomous systems from the laboratory to real-world applications.

One-Sentence Summary: Achieving safe autonomous MAV flights at speeds over 20 m/s in cluttered unknown environments, with fully onboard sensing and computing devices.

INTRODUCTION

Birds' flight capabilities have long captivated humans with their ability to navigate at high speeds in cluttered environments with remarkably low failure rates. Similarly, micro air vehicles (MAVs), among the most agile machines created by humans (*1*), hold the potential to achieve bird-like high-speed agile flights. The rapid and safe arrival of MAVs at their destinations is a critical factor for successfully deploying MAVs in practical applications. Being fast implies that an MAV is able to reach designated locations timely, enabling rapid response in time-critical missions like search and rescue (*2*) or disaster relief (*3, 4*). Being safe means the MAV could detect and avoid obstacles on the way without a collision failure. In this work, we explore how to endow MAVs with bird-like capabilities, relying solely on onboard sensing and computational units, to achieve safe and high-speed flights in unknown environments.

Achieving safety-assured, high-speed flights in unknown environments is a complex task that necessitates a holistic system design. Firstly, to execute aggressive maneuvers that avoid newly-detected obstacles during high-speed flights, the MAV must possess a high agility, characterized by its compact size and high thrust-to-weight ratio (TWR). Secondly, the MAV must have a long detection range in order to provide sufficient reaction time for avoiding obstacles at high speeds. Moreover, such detection ability should be achieved with sensors of compact size and lightweight in order not to degrade the MAV's agility. Lastly, the MAV has to carefully balance the flight speed with safety in its trajectory planning. Flight speed and safety are two mutually conflicting factors, and their balance has to be achieved efficiently with a limited computation power onboard the MAV.

Several existing works have focused on addressing the challenges in autonomous drone racing applications (*5–9*) and have achieved remarkable results in terms of flight speed and agility. Song *et al.* (*5*) employed reinforcement learning (RL) techniques to achieve flight speeds exceeding 30 m/s on a human-made racing track, utilizing external computing for control and a motion capture system for state feedback. Foehn *et al.* (*6*) proposed a strategy based on progress optimization to offline compute a time-optimal trajectory and track the trajectory with an onboard computer, while Romero *et al.* proposed a sample-based online planning approach (*7*) and utilized model predictive contouring control (*8*) to track the trajectory, achieving high-speed racing flights over 18 m/s. Kaufmann *et al.* (*9*), following a similar RL approach, combined onboard sensing and computation to surpass the performance of champion-level human pilots in racing missions. Despite achieving remarkable flight speed and agility, these methods depend on external sensing (i.e., motion capture system) (*5–8*), external computation during actual flights (*5*), or offline training (*5, 6, 9*). Moreover, these methods assume a fixed, known environment (i.e., the racing track) (*5–9*), where the control policy is exhaustively trained for the best performance. Their applicability to unknown environments is not clear.

Autonomous flights in unknown environments utilizing only onboard sensing and computation have been extensively investigated in the literature. The first set of approaches prioritizes flight speed (*10–17*). Escobar-Alvarez *et al.* adopted an expansion rate (ER)-based method to achieve high-speed flight ranging from 6 to 19 m/s in a relatively open area (*17*), utilizing visual sensors and a single-line time-of-flight (TOF) LiDAR. Loquercio *et al.* (*16*) employed imitation learning to enable autonomous flight in challenging conditions, using an end-to-end approach and achieving a maximum speed exceeding 10 m/s. Another approach is the Bubble planner proposed by Ren *et al.* (*13*), which utilized receding horizon corridors for trajectory optimiza-

tion, resulting in a maximum speed of 13.7 m/s in complex environments. An inherent issue with these works is their exclusive focus on flight speed at the expense of safety guarantees. In order to maximize flight speed, existing works (*10–15, 17*) all assume that the unknown regions caused by occlusions or limited sensor field of view (FOV) are free of obstacles during trajectory planning. Such an optimistic trajectory poses a risk of colliding with hidden obstacles in the unknown region, particularly in cluttered environments with frequent occlusions since high-speed flights require sufficient space to decelerate. Similarly, the learning-based approach in (*16*) does not incorporate safety considerations into its control policy, resulting in a low overall success rate (approximately 50% to 80%) when the flight speed exceeds 10 m/s. These limitations in success rate and the absence of safety guarantees obviously restrict the deployment of these methods (*10–16*) in real-world applications.

Alternatively, there are approaches that prioritize flight safety. The first stream of methods focuses on enhancing flight safety in a safety-aware manner (*18–20*). Quan *et al.* (*18*) and Wang *et al.* (*19*) achieve safety awareness by actively slowing down the MAV when approaching unknown spaces, while Zhou *et al.* (*20*) maximize the visibility to unknown spaces on the trajectory. While these methods could enhance flight safety, they compromise flight speed. Field experiments reported in safety-aware methods (*18–20*) indicate that their designs are rather conservative and cannot fully utilize the MAV agility, leading to a flight speed of no more than 5 m/s. Moreover, these methods rely on various heuristic strategies for safety-awareness and lack a rigorous safety guarantee. Another category of methods adopts safety-assured strategies that confine the trajectory completely within known-free spaces (*21, 22*). While offering a guaranteed level of safety, the safety-assured methods (*21,22*) tend to be even more conservative in flight speed, especially in cluttered environments. Such conservatism is partially overcome in Faster by Tordesillas *et al.* (*23*), which plans an additional trajectory in both unknown and known-free space alongside the trajectory in the known-free space. This two-trajectory planning strategy has significantly improved the flight speed, achieving a fast flight of over 8 m/s, several times faster than its prior work (*21, 22*) without losing the guarantee of safety. However, Faster suffers from high computation delay due to the use of a computationally expensive occupancy grid map (OGM) for identifying known-free spaces and a mixed integer quadratic programming (MIQP) for trajectory optimization. The high computation delay severely limits the flight speed and success rate.

In addition to the planning strategy, the sensing capability onboard the MAV plays another crucial role in achieving high-speed flights. Existing works on autonomous MAV navigation (*14–16, 18–21, 23*) often employ vision sensors (e.g., RGB(D) cameras) for localization and obstacle perception. Vision sensors suffer from several limitations, such as limited sensing range (typically 3 to 5 meters), low dynamic range (*24*), and susceptibility to strong motion blur. These factors severely restrict the attainable flight speeds and level of safety. Moreover, the performance of vision-based systems is significantly affected by lighting conditions, thereby imposing additional constraints on their practicality in real-world applications where insufficient illumination or large illumination variations are present.

Proposed Systematic Solution

In this work, we present a Safety-assUred high-sPeed aErial Robot (SUPER) to fulfill the task of safe, high-speed flights in unknown environments (Fig. 1 and Movie S1). The primary sens-

ing modality of SUPER is a three-dimensional (3D) light detection and ranging (LiDAR) sensor delivering 70 m sensing range at centimeter-level accuracy, all within a compact form factor and lightweight configuration (*25*). With all sensing, computing, and other necessary components onboard (see the system breakdown in Supplementary Section S1), SUPER achieves an impressive thrust-to-weight ratio (TWR) exceeding 5.0 (versus 0.87 for an F35 fighter aircraft at gross weight (*26*)) and a compact size of 280 mm wheelbase (versus 380 mm for that of DJI Mavic 3 (*27*)). Consequently, SUPER exhibits exceptional agility with fully onboard computing and sensing capability.

To achieve high-speed flights while ensuring safety, we adopt the two-trajectory planning strategy proposed in (*28*) and plan two trajectories in each re-planning cycle: one in known-free space to ensure safety, and the other in both known-free and unknown space to boost the flight speed. We improve the computation efficiency of the two-trajectory strategy by one order of magnitude by meticulously designing the planning and mapping modules. For mapping, we propose a method to distinguish known-free spaces directly on LiDAR points, which fundamentally enables the use of a more efficient point cloud map in the two-trajectory strategy. For planning, we use the differentiable trajectory parameterization MINCO (*29*) to enhance the trajectory optimization efficiency, as well as to optimize the switching time between the two trajectories. The considerably improved computation efficiency and optimally-determined switching time have greatly improved the attainable flight speeds and success rate as demonstrated in the benchmark comparison. Finally, the point cloud map effectively retains measurements on thin objects and represents the environments at a centimeter level accuracy, allowing the UAV to avoid thin obstacles and navigate through tight spaces.

We demonstrated the effectiveness of the complete system, SUPER, in terms of flight speed and safety in real-world environments. SUPER achieved a speed of 20 m/s in the unknown wild (Fig. 1B1) and a 100% success rate in eight different tests. Moreover, SUPER outperformed the most advanced commercial MAVs regarding flight safety against small obstacles and adaptability to environments of high clutter and low lighting conditions. Particularly, SUPER was able to detect and avoid thin objects in the wild, such as power lines and tree branches (Fig. 1B2) and navigate through extremely cluttered environments (Fig. 1B3), even at night (Fig. 1B4). The robustness of SUPER against small objects, cluttered environments, and lighting variations significantly expands its operation envelope in both natural and artificial environments, enabling round-the-clock operations spanning both daytime and nighttime. As a result, SUPER represents a significant milestone in transitioning high-speed autonomous navigation from laboratory settings to real-world applications.

RESULTS

We have demonstrated the safe and high-speed flight capabilities of SUPER through extensive simulations and real-world experiments. In over a thousand simulated flights, the planning module of SUPER reduced the failure rate by 35.9$\sim$95.8 times compared to the state-of-the-art baseline methods while achieving notably higher average flight speeds. The performance of SUPER in real-world scenarios has also been validated through a substantial number of field tests. In all experiments, SUPER is tasked to reach a goal position with fully onboard perception, planning, and control. The line connecting the goal and the MAV's current position is usually not collision-free, and the environment is completely unknown prior to the flight,

requiring SUPER to detect and avoid those obstacles in the environment in an online fashion. Video recordings of some flights are supplied in Movie S1.

Safe High-speed Navigation in Unknown Environments

We tested the flight speed of SUPER in an unknown forest environment (see Fig. 2 and Movie S2). The environment covered an area of around $280 \times 90\ \mathrm{m}^2$ and featured trees of varying thicknesses and diverse natural vegetation (Fig. 2A1). We conducted eight experiments at different times of a day, creating a wide range of lighting conditions from normal bright day to completely dark night (Fig. 2A2$\sim$A5). We also set different maximum speed constraints across the eight experiments, ranging from $5\,\mathrm{m/s}$ to $20\,\mathrm{m/s}$. In all experiments, SUPER started from position $\mathbf{p}_s$ and was assigned to reach four waypoints $\mathbf{p}_1 \sim \mathbf{p}_4$ sequentially, where $\mathbf{p}_4$ coincides with $\mathbf{p}_s$ (Fig. 2B1). The four waypoints guided the MAV to experience areas of different obstacle densities (Fig. 2B2$\sim$B4).

SUPER succeeded in all eight experiments, achieving a 100% success rate. Fig. 2C illustrates the speed distribution in the eight experiments along with the maximum speed constraints. SUPER reached a maximum flight speed of 20 m/s in Test 6-8, which represents, to the best of our knowledge, the highest flight speed achieved by an autonomous MAV relying on onboard sensing and computations in unknown environments. Moreover, Tests 6$\sim$8 were conducted during daytime, dusk, and nighttime, respectively, with the same maximum speed limit of 20 m/s. SUPER consistently showcased exceptional performance in these three experiments, demonstrating the robustness of our system to variations in lighting conditions. Besides high-speed flights, SUPER was able to adapt to different speeds and consistently met the speed constraints in all experiments. Fig. 2D presents the distribution of position tracking errors in the eight experiments. SUPER exhibited a maximum tracking error of 0.3 m and an average tracking error of 0.13 m, even at a speed of 20 m/s, showcasing exceptional tracking precision and ensuring stable performance during high-speed flights.

Navigation in Cluttered Environments.

To validate SUPER's navigation capability in complex environments, we commanded it to track a person jogging in a dense woods environment (Fig. 3A). The person passed two woods areas sequentially: the first area had a relatively sparse distribution of trees (Fig. 3C), and the second one presented a higher density where the tracked person had to lower her body to pass through (Fig. 3D). We compared SUPER's tracking and navigation performance with a state-of-the-art commercial product, the DJI Mavic 3 (*27*), by conducting two separate experiments. In both experiments, the tracked person followed roughly the same route and speed. Considering that Mavic 3 has a larger size than SUPER (0.31 m versus 0.21 m in radius), we increased SUPER's size to 0.32 m by adding four sticks on each arm to ensure a fair comparison (Fig. 3B). For target detection and tracking, Mavic 3 employed a visual recognition technique that is not available in SUPER. To work around this problem, the target wore a high-reflectivity vest, which can be easily detected in LiDAR point clouds based on the point reflectivity measurements. Both systems demonstrated successful detection and tracking of the target throughout the entire task, ensuring a fair comparison for their navigation capabilities.

The tracking trajectories of both MAVs are presented in Fig. 3A. In the beginning stage, where the target is in open areas, Mavic 3 and SUPER can achieve a comparable tracking performance. After the target entered the first woods area, Mavic 3 failed to track the target and

stopped in front of the woods (position A in Fig. 3A and Fig. 3C1). A likely reason is that the visual navigation employed by Mavic 3 had a limited map resolution and accuracy (typically tens of centimeters), which necessitated more conservative collision-free trajectories to enhance safety. Subsequently, the target exited the first woods area and ran towards the second one. Mavic 3 successfully resumed the tracking by finding a path detouring the woods area (the orange line between position A and B in Fig. 3A). However, after the target entered the second woods area with a higher density, Mavic 3 failed the tracking mission completely and disengaged from the automatic tracking mode (Fig. 3D1) due to the same reason mentioned above. In contrast, SUPER planned its trajectory directly on LiDAR point clouds with centimeter-level accuracy, enabling it to navigate through small corridors in cluttered environments. As a consequence, SUPER exhibited smooth and consistent target tracking throughout the entire mission, successfully navigating both woods areas without a stop encountered by Mavic 3 (Fig. 3 C2 and D2). The tracking processes of Mavic 3 and SUPER in this experiment are shown in Movie S3.

Avoiding Thin Objects

Thin objects, such as power lines and tree branches, present significant challenges for MAVs during field missions due to their abundance in the wild and difficulties in detection. To validate SUPER's ability to avoid small objects, we conducted a series of quantitative experiments and compared its performance with DJI Mavic 3 (*27*). As shown in Fig. 4, we selected four thin wires of different diameters: 30 mm, 20 mm, 11 mm, and 2.5 mm (Fig. 4A). The wires were hung between two trees approximately 3 m apart. To test the ability to avoid the thin wires, the MAV started at a position 4 m away on one side of the wire, with the target position 4 m away on the opposite side. Since Mavic 3 does not accept waypoint commands, we used joysticks to command it to fly from the start position to the target. To test if Mavic 3 is able to detect and avoid the thin wires on the flight path, we switched on its Advanced Pilot Assistance System (APAS) during the flight. The flight speed was approximately 2-3 m/s, which was also set as the speed constraint for SUPER for a fair comparison.

The experimental results are shown in Fig. 4. In the case of the 30 mm wire, both SUPER and Mavic 3 successfully avoided the wire and reached the target position (Fig. 4B1 and C1). However, in the experiments with the thinner wires with diameters of 20 mm, 11 mm, and 2.5 mm, Mavic 3 failed to detect and avoid them. Consequently, it crashed on the wire and caused the wire to swing (Fig. 4B2-B4). The Mavic 3's failure to detect thin wires may be attributed to challenges in accurately computing disparities and generating depth images for navigation with their vision-based navigation system. In contrast, SUPER reached its target position successfully in all four experiments, avoiding even the thinnest 2.5 mm wire (Fig. 4C1$\sim$C4). The exceptional obstacle avoidance capability is attributed to both the LiDAR sensor and our planning module: the LiDAR sensor provides point measurements on these thin objects (Fig. 4D) and the point-could map in our planning module effectively retains these points for trajectory planning. The robustness of SUPER when faced with small objects further strengthens its safety for real-world missions. A video illustration of this experiment is shown in Movie S4.

Evaluation of Safety, Success Rate, and Efficiency

We evaluated the performance of SUPER's planning module through a series of controlled experiments conducted in simulation. The testing environments consisted of a set of randomly generated 3D forests-like scenarios at a fixed size of $110\ \mathrm{m} \times 20\ \mathrm{m}$ (Fig. 5A). These scenarios

encompassed six distinct obstacle densities ranging from 3.1 to 6.5 in terms of the traversability (*30*). Traversability is a metric describing the obstacle density normalized by the robot's size r ($r = 0.2$ m in our simulation), where a higher traversability corresponds to a less challenging obstacle avoidance task. For each scenario, the positions and inclination angles of the obstacles were randomly generated. To ensure diversity, we used different random seeds to generate ten different maps for each density, resulting in a total of 60 distinct maps for evaluation. We benchmarked the performance of SUPER with three state-of-the-art baseline planners: Bubble planner (*13*), an optimistic planning strategy; Raptor, a safety-aware method (*20*); and Faster, a safety-assured method (*23*). The implementation details of these four planners are summarized in Table S1. Each planner was evaluated 18 times in each map, with maximum velocity varying from $1\,\mathrm{m/s}$ to $18\,\mathrm{m/s}$. The maximum acceleration was fixed at $20\,\mathrm{m/s^2}$ for all experiments. In each experiment, a planner is given a goal 100 m away from the start position, and the experiment terminated at any of the three conditions: (i) *Succeed*: the MAV reached the goal without any collision and satisfied all the kinodyanmic constraints (velocity and acceleration); (ii) *Collision*: the MAV collided with obstacles; (iii) *Unfinished*: in any re-plan cycle before reaching the goal, the planner failed to return a trajectory in $30\,\mathrm{s}$.

We evaluated the safety of all benchmarked methods in Fig. 5B in terms of the safe rate, which is the ratio of both *Succeed* and *Unfinished* cases over all experiments. Across all 1080 experiments with varying flight speed and obstacle density, SUPER had no collision or infeasible trajectory, achieving a perfect safe rate. Faster (*23*) is a safety-assured method in principle, due to the planning of a backup trajectory in each re-plan cycle similar to SUPER. However, we found that its backup trajectory was not constrained in the map region, where the unknown and occupied information are available, causing collisions when the map was not updated in time. The collision rate of Faster reaches 12.78%, leading to an overall safe rate of 87.22%. Bubble (*13*) achieved a safe rate of 68.80%, the remaining 31.20% *Collision* cases were caused by its optimistic planning strategy that treats unknown spaces as free. Regarding Raptor (*20*), it suffers from a collision rate of 38.33%, which is primarily due to the planned trajectory frequently violating the kinodynamic constraints during high-speed flights. These trajectories are challenging for the simulated MAV to track, resulting in collisions with obstacles. Consequently, the overall safe rate is merely 61.67%. To sum up, SUPER outperformed other benchmarked methods by achieving safe flights in all experiments, highlighting its exceptional safety-assured property.

Besides safety, whether the MAV achieved the task (i.e., reaching the goal) and at which speed it achieved the task are also important. We analyze the success rate, the ratio of *Succeed* over all tests, and the flight speed of all benchmarked methods in Fig. 5D. SUPER achieved the highest average flight speed across almost all obstacle densities, meanwhile maintaining a nearly perfect success rate of 99.63% (Fig. 5B). In contrast, Faster (*23*) had considerably lower average speeds due to the short mapping range. Moreover, Faster formulated the trajectory optimization as a mixed-integer quadratic programming (MIQP), which, on the one hand, had difficulties in finding high-speed trajectories that require intricate temporal optimization and, on the other hand, was time-consuming (Fig. 5E). The high computation time limited the planning horizon of both exploratory and backup trajectories. Since both trajectories terminate at zero speed at each re-plan, the short planning horizon caused low-speed trajectories. Frequent timeouts in the MIQP also forced the MAV to execute backup trajectories (Fig. 5C) more often,

which further reduces the overall flight speeds. Bubble (*13*), an optimistic planning strategy, and Raptor (*20*), a safety-aware planning method, had occasionally higher flight speeds than SUPER at certain obstacle densities, but at the cost of a considerably lower success rate. In sum, SUPER reduced the mission failure rate by 35.9 to 95.8 times compared to the other benchmarked methods while flying at higher or comparable speeds.

We analyze the time consumption of all benchmarked methods as shown in Fig. 5E. Each method has its computation time breaking into the mapping and trajectory planning, which further breaks into the front end and back end. SUPER and Bubble exhibited minimal time overhead for mapping (1$\sim$5 ms) due to their direct planning on point clouds, which can be obtained from LiDAR raw measurements with minimal processing (e.g., rigid transformation). Faster employed an occupancy grid map (OGM), which has to enumerate a large number of map grids traversed by all points contained in a LiDAR scan, leading to a mapping time of up to 47 ms. Raptor employed a Euclidean signed distance field (ESDF) map, which is built from OGM by further calculating the distance information of each grid (*31*), hence consuming an even longer time of 69 ms. For trajectory planning, SUPER, Bubble, and Raptor parameterized the trajectories by polynomials or B-splines, which are differentiable to the trajectory parameters and can be solved efficiently. In contrast, the MIQP problem solved in Faster for trajectory optimization is a non-differentiable (i.e., mixed-integer) and hard to solve, taking a computation time of up to 41 ms. Consequently, Bubble and SUPER took the least total computation time (8-44 ms for Bubble and 10$\sim$47 ms for SUPER). The slightly higher time consumption of SUPER when compared to Bubble is due to the planning of an extra backup trajectory. Raptor took 67$\sim$83 ms, and Faster took 67$\sim$91 ms, both are considerably higher than SUPER.

Applications

We demonstrated the use of SUPER in three representative applications: object tracking, autonomous exploration of unknown spaces, and waypoint navigation in changing environments (Movie S5). In object tracking, we commanded SUPER to track an object by repeatedly assigning its local planning goal as a position behind the object. We conducted extensive experiments to evaluate SUPER's performance in tracking both a person and a car, as depicted in Fig. 6A$\sim$D. In all experiments, the target object was attached with a sheet of high reflectivity, enabling reliable detection within the LiDAR point cloud based on reflectivity measurements. The target was then tracked using a simple constant velocity mode. The results show that SUPER successfully tracked the target and avoided all obstacles in all experiments, regardless of variations in target speeds (ranging from 1 to 8 m/s), different indoor and outdoor environments, light conditions of daytime and nighttime, and different scene scales. In the application of autonomous exploration, the MAV was commanded to explore an unknown space. To fulfill this task, a global planner named Bubble Explorer (*32*) first generated a set of waypoints online. These waypoints were expected to observe the maximum unknown spaces at the minimum cost (i.e., the total flight distance). Then, SUPER was used as a local planner to reach these waypoints while ensuring obstacle avoidance. In the experiment, SUPER navigated to all the waypoints without a collision failure and explored the entire 3000 m^2 environment within 120 seconds. Fig. 6E1$\sim$E3 present snapshots of the scene and the mapping results after the exploration. Finally, we tested the flight capability of SUPER in changing environments (Fig. 6F). A typical problem for planning methods based on point clouds is that they falsely view spaces passed by

moving objects as being occupied due to the points collected on the moving object. SUPER implemented a time-decaying point cloud map (Materials and Methods) and can hence avoid such a problem. In the experiment, the MAV starting from $\mathbf{p}_s$ was commanded to fly through a sequence of waypoints (i.e., $\mathbf{p}_1 \sim \mathbf{p}_{11}$, see Fig. 6F1), meanwhile, two persons were moving randomly in the same area (Fig. 6F2). SUPER successfully arrived at all waypoints by utilizing the free spaces left by the moving objects without a collision with the moving object or static obstacles in the environments. These results underscored the superior navigation capabilities of SUPER in complex unknown environments.

DISCUSSION

Autonomous Aerial Robots

SUPER is designed for high-speed navigation in unknown environments by relying solely on its onboard sensing and computation. Agility and onboard computation are, therefore, two important factors. SUPER has a size of 280 mm and a thrust-to-weight ratio (TWR) of 5.0, which characterizes its high agility. The onboard computer of SUPER is an Intel NUC 12 equipped with a 12-core 4.7 GHz CPU, providing sufficient computation power for efficient, online optimization of both exploratory trajectories, attaining high-speed flights and backup trajectories ensuring safety. The impressive agility and onboard computation power have enabled SUPER to achieve high-speed flights over 20 m/s in unknown cluttered environments without a collision failure. The safety-assured and high-speed flying capability of SUPER places itself in a unique position among existing autonomous aerial robots (Fig. 1A). Agilicous (*33*) exhibits a high TWR over 5.0 and a size similar to SUPER, but its imitation learning-based planning (*16*) achieves a lower flight speed of 10 m/s and lacks a safety guarantee, yielding a low success rate in high-speed flights (around 60% at 10 m/s). Moreover, the onboard computing unit, an Nvidia TX2 (*34*) with a 4-core 1.9 GHz CPU, has limited computational power, which restricts the potential application of Agilicous for more computationally demanding tasks, such as autonomous exploration. On the other hand, ZJU Swarm (*35*) concentrates on designing small-size and lightweight autonomous MAVs for swarm applications, achieving an impressive size of 110 mm wheelbase with a weight less than 300 g. However, this design has a low TWR (only 2.4) and limited onboard computation resources due to the weight constraints, leading to a maximum speed of merely 3 m/s in autonomous flights (*35*). Being a representative commercial MAV, the DJI Mavic 3 (*27*) has a significantly larger wheelbase of 380 mm compared to SUPER, and its relatively low TWR (only 2.3) further restricts its ability to perform high-speed flights in cluttered environments. Mavic 3 has an onboard Advanced Pilot Assistance System (APAS), which can detect and avoid potential obstacles on the flight path. The maximum flight speed supported by APAS of Mavic 3 is 15 m/s (*27*), which is still lower than the speed attained by SUPER in autonomous flights.

LiDAR-based MAV Navigation

SUPER employed a 3D LiDAR sensor to achieve its high-speed navigation. Compared to cameras that have been widely used in academic research (*14–16, 18–21, 23*) and commercial MAVs (*27, 36*), LiDAR sensors provide key advantages that are crucial to enhance the flight speed, safety, and efficiency of MAV navigation. Firstly, LiDAR sensors provide direct, accurate (centimeter level), and long-range (tens of to hundreds of meters) depth measurements. The

long-range measurements allow the MAV to perceive and avoid obstacles far ahead, hence enabling a high-speed flight, and the accurate measurements allow the MAV to fly through small spaces in cluttered environments. In contrast, visual approaches estimate depths based on disparity and triangulation (*37*), which is computationally demanding, and the accuracy decreases quadratically with the measuring distance. The state-of-the-art visual sensors, such as Intel Realsense (*38*), have an effective measuring range of merely 3$\sim$5 m. Secondly, LiDAR sensors measure 3D points at an extremely high frequency, from hundreds of thousands to millions of Hertz. Employing such high-frequency measurements would allow us to estimate extremely fast MAV motions (*39*), where traditional RGB or RGB-D cameras could suffer from severe motion blur due to the long exposure time and low dynamic range. Thirdly, LiDAR sensors use time of flight (ToF) for depth measuring, enabling them to detect thin objects with diameters even below 1 cm. In contrast, visual-based (including RGB, RGB-D and event cameras-based) approaches prove inadequate in detecting typical thin objects like powerlines, due to the challenges of accurately computing disparities for such objects. Finally, due to their active measuring mechanism, LiDAR sensors exhibit reliable performance even in low-light conditions, such as nighttime, surpassing visual sensors' capabilities in these scenarios.

Using 3D LiDAR sensors for MAV navigation is not new. For example, several works (*10, 11, 40–42*) employed the VLP-16 on multirotor UAVs, and other work (*29, 43–45*) utilized the Ouster OS1-128 (*46*). A crucial problem among these works lies in their LiDAR sensors. Existing LiDAR sensors are bulky, heavy, and costly: the prior two factors necessitate a UAV of large size (wheelbase of 330-1133 mm) and low thrust-to-weight ratio (e.g., typically around 2.0 (*11, 29, 40, 41, 43–45, 47, 48*)), which considerably restricted the UAV agility. The high cost of LiDAR sensors has also prevented the wider adoption of them in UAV navigation, not even to speak of the high-speed agile flights that pose a much higher risk to the system. The reported flight speeds of existing LiDAR-based UAVs were merely 2$\sim$5 m/s (*40,41,44*). Zhang *et al.* (*10, 11, 42*) achieved a flight speed of 10 m/s with a LiDAR-based UAV, but their UAV was large (i.e., wheelbase of 1133 mm), heavy (over 13 kg) and were only demonstrated in a static, sparse woods environment (e.g., the distance between obstacles was over 10 m) with very smooth flights. Our SUPER is enabled by the rapid recent advancements in LiDAR technology, which have enabled the commercialization and mass production of solid-state LiDARs with orders of magnitude lower cost and weight. SUPER used a Livox MID360 LiDAR sensor (*25*), which features a 265 g weight, $65 \times 65 \times 60$ mm ($L \times W \times H$) size, and a power consumption of around 6.3 W. Although the LiDAR sensor is still larger, heavier, and more power consuming than typical depth camera sensors (e.g., the RealSense D435i (*49*) weighs only 72 g with a $90 \times 25 \times 25$ mm size and 1.6 W power consumption), given a compact design of airframe, SUPER has achieved a wheelbase of only 280 mm, which is even smaller than state-of-the-art visual-based MAVs such as the DJI Mavic 3 (380 mm wheelbase). Furthermore, SUPER achieves an impressive thrust-to-weight ratio exceeding 5.0, surpassing the typical ratio of around 2.0 for existing LiDAR-based UAVs. The compact size and high thrust-to-weight ratio of SUPER lead to unprecedented agility when compared to previous LiDAR-based UAVs.

While LiDAR sensors provide a superior sensing ability over depth sensors in terms of range, resolution, and accuracy, leveraging the LiDAR measurements to achieve safe, high-speed flights is still challenging, requiring a carefully-designed map that truthfully represents the environments as measured by the LiDAR sensor. One mainstream approach in existing au-

tonomous systems (*12, 14, 15, 20, 23, 40*) utilizes occupancy grid maps (*50, 51*) or Euclidean signed distance field (ESDF) maps (*31*). However, these maps are computationally demanding due to the ray-casting or distance field update processes. They are also incapable of mapping small thin objects due to the limited spatial resolution in the ray-casting process (see Fig. S5). Another choice is the point cloud map, such as (*10, 11, 13, 41, 42, 52*), which is much more efficient to maintain and can effectively retain point measurements on small thin objects, due to the avoidance of ray-casting. However, one inherent issue with a point cloud map is the difficulty in distinguishing known-free and unknown space. Therefore, existing point cloud-based planning methods (*10, 11, 13, 41, 42, 52*) often treat the unknown area as free, which cannot ensure avoidance of occluded obstacles and have exhibited low success rates in cluttered environments. We proposed a method (i.e., Theorem 1) that distinguishes known-free spaces directly from the point cloud map, allowing SUPER to plan a backup trajectory ensuring safe flights. Consequently, SUPER has achieved unprecedented flight speeds and success rates than previous LiDAR-based navigation methods (*13, 42*) (see Supplementary Section S2 for a benchmark comparison). The accurate and high-resolution point map also enabled SUPER to avoid thin obstacles in the wild and navigate through tight spaces safely. Finally, outdated points caused by moving objects or LiDAR false measurements present another problem for point cloud maps. We addressed this problem by a spatio-temporal sliding mechanism, which has enabled SUPER to utilize the areas traversed by moving objects.

Safety-assured High-speed Trajectory Planning

SUPER achieved high-speed and safe planning by calculating two trajectories at each time: an exploratory trajectory in both known-free and unknown spaces and a backup trajectory in known-free spaces only. This two-trajectory strategy was first proposed in Faster (*23, 28*). SUPER adopted such a two-trajectory strategy but had completely different designs for sensing, mapping, and planning, which are optimized for higher flight speeds, success rates, and computation efficiency. Ultimately, SUPER achieved flight speeds exceeding 20 m/s in the unknown wild using only its onboard sensing and computing devices.

From sensing, Faster (*23*) employed a depth camera (*38*) for obstacle detection. The depth camera has a measuring range of merely 3$\sim$5 m, which limits the attainable flight speeds of the system. Moreover, Faster used an external motion capture system for state estimation, restricting its application in real-world environments that are often uninstrumented. In contrast, SUPER utilized a lightweight LiDAR sensor (*25*), which offers up to 70 m depth measurements enabling detection and avoidance of obstacles at a distance. The long-range detection capability constituted the base for high-speed flights achieved in this work. SUPER further employed the LiDAR measurements for state estimation, using our in-house developed LiDAR-inertial odometry, FAST-LIO2 (*39*). Consequently, the system relies completely on onboard devices and is adaptable to unknown environments without any prior instrumentation (e.g., GNSS-denied environments).

A key requirement for the two-trajectory strategy used in Faster (*23*) and SUPER is to distinguish known free spaces of the environment. Faster (*23*) utilizes occupancy grid maps (OGMs) to maintain such information. However, updating OGM is rather computationally expensive due to the ray-casting operations (*50*). To constrain the overall computation time for real-time operation, Faster updates only 5 meters of the OGM, attaining an average computation

time of 47 ms. The limited mapping distance further constrains the planning horizon, leading to short trajectories that have rather low flight speeds. In contrast, SUPER distinguishes known-free spaces directly on LiDAR point clouds (Materials and Methods), eliminating the time-consuming OGMs while retaining the full range of LiDAR points up to 70 m. The resultant map updating time is merely 1$\sim$5 ms.

For planning, Faster formulates the optimization of both exploratory and backup trajectories into mixed-integer quadratic programming (MIQP) problems. MIQP is non-differentiable and often hard to solve, leading to high computation time, occasional constraints violation, and local minimum trajectories with limited speeds. The high computation time, on one hand, causes frequent timeouts that trigger the MAV to switch to backup trajectories of lower speeds, and on the other hand, limits the planning horizon that also constrains the flight speeds. In contrast, SUPER parameterizes the trajectory as multi-stage polynomials (Materials and Methods) following (29), whose spatio-temporal parameters can be efficiently optimized using gradient methods. SUPER's trajectory optimization required only half the computation time of Faster while adhering to all constraints. Another key difference between Faster and SUPER is the determination of switching time between backup and exploratory trajectories. Faster determines the switching time heuristically prior to the backup trajectory optimization, based on a wild assumption that the backup trajectory would decelerate along the exploratory trajectory. In contrast, SUPER optimizes the switching time alongside with the backup trajectory optimization. The optimized switching time in SUPER is usually much larger than the heuristically determined one in Faster, reducing the flight time on the backup trajectory and the chance of switching to it (Fig. S6).

Limitations and Future Directions

At the hardware level, the emergence of smaller and lighter LiDAR sensors with longer sensing ranges and denser point clouds may open up new opportunities for autonomous aerial systems. While the LiDAR adopted by SUPER is much smaller and lighter than previous ones, it remains significantly larger and heavier than visual sensors, limiting the potential for further miniaturization of the MAV platform. This restricts the MAV's ability to operate in more confined spaces (e.g., narrow corridors with widths less than 0.5 m). Adopting lighter LiDAR sensors could also enhance the thrust-to-weight ratio of the drone, thereby improving its agility further. Additionally, the sensing range and point cloud density of LiDAR sensors can be further improved. Longer sensing ranges and denser point clouds can provide more detailed environmental information, which could enable SUPER to avoid thinner objects at higher speeds.

The aerodynamic design of the drone can also be further optimized. The current SUPER system only considers aerodynamic drag effects at the software level, by identifying drag coefficients and compensating it in the control module. However, with careful aerodynamics reducing air resistance, the drone's flight speed and stability could be increased even further.

At the algorithmic level, while SUPER has demonstrated the ability to handle dynamic obstacles, the current implementation does not actively detect moving objects nor predict their movement, limiting its flight capability in highly dynamic environments. By actively identifying the moving objects (e.g., (*53*)) and predicting their motion, the planning module could generate smoother, safer trajectories, enhancing flight safety and reducing energy consumption. Additionally, further parallelization of the SUPER software implementation and leveraging hardware

accelerators, such as graphics processing units (GPUs), could enhance the system's efficiency, reduce latency, and further improve flight speed.

Building upon the improvements in both hardware and software, we could further explore the deployment and application of SUPER in real-world applications, including autonomous exploration, logistics transportation, and integration within multi-agent swarm systems.

MATERIALS AND METHODS

Agile LiDAR-based MAV platform

The design objective of SUPER is to achieve a high thrust-to-weight ratio (e.g., exceeding 5.0) while minimizing its size. Following a thorough investigation, we have finalized the design of SUPER, as depicted in Fig. 7A. The platform possesses a takeoff weight of $1.5\,\mathrm{kg}$ and a compact wheelbase measuring only $280\,\mathrm{mm}$. For the motors, we have selected the T-Motor F90 type (*54*), with each motor capable of generating a maximum thrust of $23.6\,\mathrm{N}$. This configuration results in a maximum total thrust of 94.4 N and an estimated ideal thrust-to-weight ratio of approximately 6.3. Due to factors like airflow interference and blockage, the actual thrust-to-weight ratio is slightly above 5.0, enabling highly agile maneuvers. For computation, the MAV is equipped with a flight control unit (FCU) running PX4 Autopilot (*55*), and an Intel NUC onboard computer that features a 12-core 4.7 GHz i7-1270P central processing unit (CPU). For sensing, the MAV equipped a lightweight 3D LiDAR (*25*) enabling object detection at ranges exceeding $70\,\mathrm{m}$ with a rapid point rate of 200,000 Hz. The MAV also equipped an IMU on the FCU, providing acceleration and angular rate measurements at a frequency of $200\,\mathrm{Hz}$. The LiDAR point clouds and IMU measurements are input to the navigation software, which consists of a LiDAR-based perception module performing state estimation and obstacle mapping, a planning module for generating safety-assured and high-speed trajectories, and a trajectory tracking controller tracking the planned trajectories, all running on the onboard computer (Fig. 7). The outputs of the trajectory tracking controller are the desired angular velocity (consisting of pitch, roll, and yaw rate) and thrust acceleration, which are tracked by the lower-level controllers running on the FCU.

LiDAR-based perception

The perception module (Fig. 7B) consists of two main parts: state estimation and mapping. The state estimation part is based on our in-house designed LiDAR-inertial odometry, FAST-LIO2 (*39*), which provides low-latency 9-DOF state estimation (position, velocity, and attitude) at a frequency of 200 Hz and with a latency of less than 1 ms. It also produces world-frame registered point clouds at a scan rate of 50 Hz, with around 4,000 points per scan, resulting in a total of about 200,000 points per second at a latency of less than 10 ms.

For the mapping module, we propose a point cloud-based spatio-temporal sliding point cloud map to represent the occupied spaces of the environment. To prevent increasingly densified points over time accumulation, we downsample the points in the map using a uniform grid data structure. This structure divides the space into uniform grids with a resolution of 0.05 to 0.3 meters and retains only the center point of each occupied grid cell. Also, to prevent over-large maps that are not necessary for local planning, we only maintain a local map of size 50$\sim$100 m around the MAV using a zero-copy map sliding strategy following our previous work (*51*). Furthermore, a well-known problem of a point cloud map is that LiDAR points belonging to

both the static environment and moving objects in past locations are considered as occupied straightly, which will falsely classify the space ever traveled by moving objects as occupied. To address this issue, a temporal sliding strategy is used to cull points outside a temporal window t_d. Specifically, within each grid cell, we track the time t_{hit} when it was last hit by a LiDAR point. If at current time t_c, the $\Delta t = t_c - t_{\text{hit}}$ is less than a threshold t_d, the grid cell is considered occupied; otherwise, it is considered free.

The point-cloud map is constructed as follows. Upon the arrival of a new LiDAR scan, we locate the grid cells hit by new points and update their t_{hit} as the timestamp of the new point in it. On the other hand, to reduce the latency for map updates, we employ a lazy-update strategy for the outdated map grids, which are culled only when they are queried by the trajectory planning module. The resultant computational complexity of map update depends solely on the number of LiDAR points in a scan while being independent of the sensing range. Consequently, the proposed mapping module can effectively leverage the extended sensing range of LiDAR to accomplish long-range updates. In contrast, volume mapping methods such as occupancy grid maps (OGMs) (*50*) or Euclidean signed distance field (ESDF)-based maps (*31*) have to update all grid cells in the measuring range, leading to a significant high computation load and short mapping range, typically around 5$\sim$10 m.

Safety-assured high-speed trajectory planning

Framework overview

To achieve safe and high-speed trajectory planning, we employ a two-trajectory planning strategy where two trajectories are planned at each re-plan cycle (Fig. 7C): (i) exploratory trajectory $\mathcal{T}_e : \mathbf{p}_c \rightarrow \mathbf{p}_g$, which expands from the MAV current position $\mathbf{p}_c$ to a goal position $\mathbf{p}_g$. With an aim to achieve high-speed flight, the exploratory trajectory treats unknown space as free and is planned in both known-free and unknown spaces. (ii) backup trajectory $\mathcal{T}_b : \mathbf{p}_s \rightarrow \mathbf{p}_l$, which begins from a position $\mathbf{p}_s$ on the exploratory trajectory at time t_s and ends at a position $\mathbf{p}_l$ with zero speed. With an aim to ensure safe flights, the backup trajectory and the segment of the exploratory trajectory from $\mathbf{p}_c$ to $\mathbf{p}_s$ must remain in known-free spaces. In cases where the whole exploratory trajectory is already in the known-free spaces (e.g., when the MAV is in close proximity to the goal), no backup trajectory is needed to be planned. Finally, the committed trajectory $\mathcal{T}_c : \mathbf{p}_c \rightarrow \mathbf{p}_s \rightarrow \mathbf{p}_l$ consists of the exploratory trajectory from $\mathbf{p}_c$ to $\mathbf{p}_s$ and the backup trajectory from $\mathbf{p}_s$ to $\mathbf{p}_l$. The committed trajectory is then transmitted to the control module for execution.

The two trajectories above are re-planned at a constant frequency of 10 Hz. At each replan, we determine the goal position $\mathbf{p}_g$ used for exploratory trajectory planning from the goal position $\mathbf{G}$ specified externally, such as the next waypoint to pursue in a waypoint navigation task and the next viewpoint in an exploration task. Specifically, if the distance between the current MAV position $\mathbf{p}_c$ and the goal position $\mathbf{G}$ exceeds the planning horizon H, we project $\mathbf{G}$ onto a sphere centered at $\mathbf{p}_c$ with a radius of H to obtain $\mathbf{p}_g$; otherwise, $\mathbf{p}_g$ is set to $\mathbf{G}$. A re-plan is considered successful if both the exploratory and backup trajectories are generated successfully, or if an exploratory trajectory that remains entirely within known-free spaces is generated. Upon a successful re-plan, a new committed trajectory is formed and updated to the control module. Otherwise, the re-plan is considered to fail, and the MAV would execute the last committed trajectory. Since the last committed trajectory remains in the known-free space, executing it

would ensure collision avoidance and guarantee flight safety, even at planning failure.

Both the exploratory and backup trajectories are planned via corridor-constrained trajectory optimization. The method consists of generating a flight corridor and then optimizing a smooth trajectory within the corridor (Fig. 8A). The flight corridor represents collision-free spaces from the start position to the goal position by a series of convex shapes, such as cubes (*56*), balls (*13, 41*), and polyhedra (*12, 23, 40*). In our two-trajectory planning framework, two polyhedra corridors need to be generated: the exploratory corridor, which encompasses both known-free and unknown spaces, and the backup corridor, which encompasses only known-free spaces. Then, the exploratory and backup trajectories are planned by optimizing two smooth trajectories in the respective flight corridors. In the following sections, we will present how to generate exploratory and backup corridors directly on LiDAR point clouds and how to optimize smooth trajectories within the respective corridors.

Flight corridor generation in configuration space

We represent both exploratory and backup corridors as a set of overlapping polyhedra and extract them directly from LiDAR point cloud inputs to enhance the computation efficiency. For the exploratory corridor, the objective is to find a set of overlapping polyhedra that connects the start points $\mathbf{p}_s$ to the goal position $\mathbf{p}_g$. To achieve this, we first employ an A* path search algorithm (*57*) to find a collision-free path $\mathcal{G}$ connecting $\mathbf{p}_c$ and $\mathbf{p}_g$ (Fig. 8B1), the resulting path consists of a set of discrete positions with a step equal to the map resolution (e.g., 0.1 m). Then, we search for a series of line segments along the path $\mathcal{G}$, by repeatedly finding the furthest visible point from the end of the last line segment. Specifically, starting from $\mathbf{p}_c$, the first point of $\mathcal{G}$, we find the furthest point $\mathbf{s}_1$ that is visible from $\mathbf{p}_c$ and form a line segment $(\mathbf{p}_c, \mathbf{s}_1)$. Then, starting from $\mathbf{s}_1$, we find its furthest visible point $\mathbf{s}_2$ and form the line segment $(\mathbf{s}_1, \mathbf{s}_2)$. This process is repeated until the goal position $\mathbf{p}_g$ is visible (Fig. 8B2). Finally, each of the found line segments is utilized as a seed for convex decomposition, which extracts a polyhedra that contains the seed but not any obstacle points input to it (Fig. 8B3).

The convex decomposition has to be performed in the robot configuration space to consider the UAV size. Existing methods, such as FIRI (*58*), RILS (*40*), or their predecessor IRIS (*59*), address the robot size through point inflation or plane shrinkage. We introduce a new method for the convex decomposition in configuration space, termed Configuration-space Iterative Regional Inflation (CIRI). CIRI inherits the iterative optimization process and the efficient SOCP-based Maximum Inscribed Ellipsoid Volume (MIEV) algorithm from the prior work FIRI (*58*). It distinguishes itself from previous works (*40, 58, 59*) by directly operating on input point clouds and extracting polyhedra in the MAV configuration space without prior map inflation (*12, 23, 60*) or polyhedra shrinking (*40, 61*). Additionally, CIRI employs a plane adjustment strategy to ensure that the polyhedron generated in the configuration space also contains the line seed. Compared with point inflation and plane shrinkage strategies used in existing methods (*58, 62*), the polyhedron obtained by CIRI achieves a larger volume in less computation time, meanwhile ensuring a complete containment of line seed in the configuration space (see Fig. S8). Details of the CIRI method are provided in Supplementary Materials.

Backup corridor generation

Leveraging CIRI, we can efficiently generate exploratory corridors directly from LiDAR point clouds. However, generating backup corridors, which indicate known-free spaces, from point clouds is challenging due to the lack of information regarding known and unknown areas. To

overcome this limitation, we introduce a method that unifies the generation of exploratory and backup corridors within the same CIRI framework, relying solely on point cloud data and eliminating the need for ray cast-based occupancy mapping. Our approach leverages the dense and high-precision depth measurements obtained from LiDAR sensors, and is fundamentally based on the theorem below:

Theorem 1. *Let $\mathcal{D}$ be an omni-directional and infinitely-dense depth image captured at the position $\mathbf{p}_c \in \mathbb{R}^3$. If a set $\mathcal{S} \subseteq \mathbb{R}^3$ satisfies the following conditions, it must lie in the known-free space:*

(a) $\mathcal{S}$ is a convex set;

(b) $\mathcal{S}$ contains no point from $\mathcal{D}$;

(c) $\mathbf{p}_c \subseteq \mathcal{S}$.

Proof. See Supplementary Materials Section S6. □

Theorem 1 guarantees that a polyhedra $\mathcal{S}$ extracted by CIRI will lie in the known-free space and can hence serve as the backup corridor if the input point cloud makes a sufficiently dense depth image and the input seed contains the current position $\mathbf{p}_c$. This is because the polyhedra $\mathcal{S}$ extracted by CIRI (i) is convex, thus satisfying (a); (ii) does not contain any points from the input point cloud (hence depth image $\mathcal{D}$), thus satisfying requirement (b); and (iii) contains the seed and hence the position $\mathbf{p}_c$, fulfilling requirement (c).

Based on Theorem 1, the detailed process is illustrated in Fig. 8C. For the seed, we select the line segment that starts from the MAV's current position, $\mathbf{p}_c$, and ends at the furthest point $\mathbf{s}$ on the exploratory trajectory $\mathcal{T}_e$ that is visible from $\mathbf{p}_c$ (Fig. 8C1). The purpose is such that the backup corridor, which is extracted by the CIRI algorithm and should contain the seed $(\mathbf{p}_c, \mathbf{s})$, is aligned with the exploratory trajectory and can hence contain a substantial portion of the exploratory trajectory. Meanwhile, considering that practical LiDAR sensors are not infinitely dense, to make a sufficiently dense depth image, we accumulate recent LiDAR scans (e.g., 1-2 seconds) and input them to CIRI, along with the line seed $(\mathbf{p}_c, \mathbf{s})$, to obtain a convex polytope $\mathcal{S}$ (Fig. 8C2). The area within the generated polytope $\mathcal{S}$ is guaranteed to be known free according to Theorem 1. In case the LiDAR has a limited field of view (FOV), the intersection between the polytope $\mathcal{S}$ obtained by CIRI and a convex subset of the LiDAR FOV region can be used (Fig. 8C3). This intersection forms another polyhedron within the known-free space. Moreover, the first polytope of the exploratory corridor must also be intersected with the MAV's FOV to ensure that the initial portion of the exploratory trajectory lies in known-free space, thereby enhancing the feasibility of the backup trajectory optimization. A detailed method for selecting the convex subset of the FOV and an analysis of the FOV's impact are provided in the Supplementary Materials.

Trajectory optimization

Given the exploratory and backup corridors generated in the previous section, we consider how to optimize smooth trajectories within them. For exploratory trajectory generation, optimizing high-speed trajectories is challenging since it requires not only spatial optimization of the trajectory shape but also temporal optimization for time allocation. The objective is

to find a smooth trajectory $\mathbf{p}(t) : [0, t_M] \to \mathbb{R}^3$ that minimizes the flight duration t_M and control efforts represented by the s-th order derivative $\|\mathbf{p}^{(s)}\|^2$ ($s = 4$ in this work), while satisfying all kinodynamic and collision avoidance constraints. The dynamic constraints encompass maximum speed and acceleration limitations, and collision avoidance is achieved through positional constraints enforced by flight corridors. Considering the composition of the flight corridors, we parameterize the trajectory as a multi-stage polynomial as proposed in (*29*). Specifically, given the exploratory corridor $\mathcal{C}_e$ composed of M consecutively overlapped polyhedra $\mathcal{P}_i$, both the trajectory $\mathbf{p}(t)$ and the flight duration t_M are divided into M segments, where the i-th ($i = 1, 2, \cdots, M$) segment $\mathbf{p}_i(t)$ has a duration T_i and is contained in the polyhedra $\mathcal{P}_i$ (Fig. 8A). Two consecutive trajectory segments $\mathbf{p}_i(t)$ and $\mathbf{p}_{i+1}(t)$ connects at position $\mathbf{q}_i \in \mathbb{R}^3$ and are continuous up to the $(s-1)$-th order derivative. Then, given the initial state $\mathbf{s}_0 = \mathbf{p}^{(0:s-1)}(0)$, which is the current MAV state, and the terminal state $\mathbf{s}_f = \mathbf{p}^{(0:s-1)}(t_M)$, which is the front-end path end position with zero derivatives, the i-th polynomial trajectory can be parameterized by the intermediate point $\mathbf{Q} = (\mathbf{q}_1, \ldots, \mathbf{q}_{M-1})$ and time allocation $\mathbf{T} = (T_1, \ldots, T_M)$ as follows:

$$\mathbf{p}_i(t) = \mathbf{C}_i(\mathbf{Q}, \mathbf{T}, \mathbf{s}_0, \mathbf{s}_f)\beta(t), t \in [0, T_i], i = 1, 2, \cdots, M, \tag{1}$$

where $\beta(t) = [1, t, \ldots, t^{2s-1}]^{\mathrm{T}}$ is the basis function and $\mathbf{C}_i$ is the coefficient matrix of the polynomial that can be obtained from $\mathbf{Q}$ and $\mathbf{T}$ by solving a linear system (*29*). Moreover, $\mathbf{C}_i$ (and hence the trajectory $\mathbf{p}_i(t)$) is differentiable to both $\mathbf{Q}$ and $\mathbf{T}$ with the partial differentiation derived in (*29*).

With the parameterization in (1), the trajectory optimization is then formulated as:

$$\min_{\mathbf{Q},\mathbf{T}} \int_0^{t_M} \|\mathbf{p}^{(s)}(t)\|_2^2 \, \mathrm{d}t + \rho_T t_M + \rho_u \mathcal{U}, \tag{2a}$$

$$\text{s.t.} \quad \mathbf{p}(t) = \mathbf{C}(\mathbf{Q}, \mathbf{T}, \mathbf{s}_0, \mathbf{s}_f)\beta(t), \tag{2b}$$

$$t_M = \sum_{i=1}^{M} T_i, \tag{2c}$$

$$\|\dot{\mathbf{p}}\|_2^2 \le v_{\max}^2, \|\ddot{\mathbf{p}}\|_2^2 \le a_{\max}^2, \tag{2d}$$

$$\mathbf{p}_i(t) \in \mathcal{P}_i, \forall i \in [1, M], t \in [0, T_i], \tag{2e}$$

where $\rho_T > 0$ is the weight used to penalize the total flight time to achieve high flight speed , and $\rho_u > 0$ is the penalty weight for the soft penalty term. Constraint (2d) guarantees compliance with kinodynamic constraints, while constraint (2e) constrains the position trajectory within the flight corridor. The term $\mathcal{U}$ in the objective function is defined as

$$\mathcal{U} = \sum_{i=1}^{M} \mathcal{L}_\mu \left(\|\mathbf{q}_i - \mathbf{a}_i\|_2^2 \right), \tag{3}$$

where $\mathbf{a}_i$ is the center location of the overlapping area between $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$. This term is used to prevent the trajectory from being too proximate to the boundaries of the flight corridor, which is often where the obstacles are located. Maintaining a distance from obstacles would enhance the visibility of the onboard sensor, and hence allows to generate larger known-free flight corridors for backup trajectory planning. $\mathcal{L}_\mu$ is a C^2 continuous barrier function as illustrated in Fig. S9. we choose μ to a larger value (e.g., 0.5) to make the penalty softer. This choice ensures that the

intermediate point $\mathbf{q}_i$ is attracted towards $\mathbf{a}_i$ while allowing for some flexibility and avoiding excessive constraints on the optimization problem.

To solve the optimization problem in (2), we adopt the method in (*29*), which reformulates the constrained problem into an unconstrained one. Thanks to the differentiability of the trajectory (and hence the objective and constraints in (2)) to the optimization variables $\mathbf{Q}$ and $\mathbf{T}$, the unconstrained optimization problem can be solved efficiently using a gradient-based method, the L-BFGS solver (*63*). The solved exploratory trajectory is denoted as $\mathcal{T}_e(t)$.

For the backup trajectory, it should start from an initial state $\mathbf{s}_0$ that lies on the exploratory trajectory $\mathcal{T}_e(t)$ at time t_s (i.e., $\mathbf{s}_0 = \mathcal{T}_e(t_s)$) and end at a terminal position $\mathbf{p}_l$ in the known-free space with zero speed and acceleration (i.e., $\mathbf{s}_f = (\mathbf{p}_l, \mathbf{0}, \cdots)$). To maximize the average flight speed, it is desirable to minimize the proportion of executing the backup trajectory, as it is typically a deceleration trajectory due to the zero termination speed. That is being said, the switching time t_s from the exploratory trajectory should be as late as possible so that the MAV can spend more time on the high-speed exploratory trajectory. In case of re-planning failure or timeout, the late switching time also allows more attempts for the planning module to re-plan new trajectories to minimize the chance of executing the backup trajectory. On the other hand, t_s cannot be arbitrarily large. Too large t_s will cause the starting point of the backup trajectory $\mathbf{p}_s$ to be too close to the boundary of the backup corridor, which may render the backup trajectory optimization problem infeasible (Fig. 8D).

To determine the best switching time t_s, we optimize it along with the backup trajectory $\mathcal{T}_b(t)$. We adopt the same formulation as in the exploratory trajectory optimization (2) but replace the corridor constraints by the backup corridor $\mathcal{C}_b$ to ensure the whole backup trajectory lies in the known-free space. We also replace the cost item $\mathcal{U}$ as $\mathcal{U} = -t_s$ to maximize the switching time t_s. Besides modification of the objective function and constraints, two additional optimization variables are added, the switching time t_s and the terminal position $\mathbf{p}_l$. The terminal position $\mathbf{p}_l$ replaces the goal position $\mathbf{p}_g$ in (2) and should be determined within the optimization since the MAV could rest at any position in the known-free space. Moreover, since the backup trajectory lies within the backup corridor, the start point $\mathbf{p}_s$ should lie in the backup corridor as well, necessitating an additional constraint $t_c < t_s < t_o$, where t_c is the current time and t_o is the time when the exploratory trajectory exits the backup corridor, respectively (see Fig. 8D). To eliminate the inequality constraints in the optimization, we parameterize t_s as another variable $\eta \in \mathbb{R}$ as follows:

$$t_s = (t_o - t_c)\frac{1}{1 + e^{-\eta}} + t_c, \tag{4}$$

where η is completely unconstrained (see Fig. S10). As a result, the complete optimization variables are $(\mathbf{Q}, \mathbf{T}, \mathbf{p}_l, t_s)$. Combining the fact that the trajectory $\mathbf{p}_i(t)$ is differentiable to the $\mathbf{Q}, \mathbf{T}$, the initial state $\mathbf{s}_0$, and the terminal state $\mathbf{s}_f$ containing $\mathbf{p}_l$ (as proved in (*29*)) and the fact that $\mathbf{s}_0 = \mathcal{T}_e(t_s)$ is differentiable to t_s (and hence η), the whole optimization problem containing the objective and constraints is differentiable and can hence be solved by the gradient method, the L-BFGS solver (*63*), again.

On-manifold model predictive controller

The position trajectory $\mathbf{p}(t)$ generated by the planning module is then transmitted to the control module for tracking (Fig. 7D). The position trajectory is first transformed into a state trajectory

$\mathbf{s}(t)$ based on the differential flatness of the MAV subject to drag effects (*64*). Then, to accurately track this state trajectory at high speeds, we implement an on-manifold model predictive control (OMMPC) developed in our previous work (*65*). The method linearizes the quadrotor system by mapping its states from the state manifold to the local coordinates of each point along the trajectory $\mathbf{s}(t)$, leading to a linearized system that is minimally-parameterized meanwhile avoiding kinematic singularities, allowing the MAV to perform aggressive maneuvers with large attitude angles, such as banked turns at sharp corners. The linearized system is a linear time-varying system, which is then controlled by a usual model predictive controller (MPC). To enhance the control accuracy, we further consider the rotor drag compensation in the control. Detailed derivations of the OMMPC with rotor drag compensation and a quantitative analysis of the compensation effectiveness, are provided in Supplementary Materials and Fig. S13.

SUPPLEMENTARY MATERIALS

Supplementary Text
Fig. S1. System breakdown of SUPER.
Fig. S2. Scenario and trajectory of the endurance test.
Fig. S3. Illustration of Falco with and without gimbal structure.
Fig. S4. Evaluation of success rate and efficiency.
Fig. S5. Visual example of thin wire detection on OGM.
Fig. S6. Evaluation of planning module of Faster and SUPER.
Fig. S7. Illustration of the proposed flight corridor generation algorithm (CIRI).
Fig. S8. Comparison of convex decomposition in configuration space.
Fig. S9. The plot of the barrier function $\mathcal{L}_\mu$.
Fig. S10. The plot of the mapping from η to t_s.
Fig. S11. Comparison of tracking performance with different drone models of MPC.
Table S1. The implementation details of the four benchmarking methods.
Movie S1. Overview of SUPER.
Movie S2. Safe High-speed Navigation in Unknown Environments.
Movie S3. Navigation in Cluttered Environments.
Movie S4. Avoiding Thin Objects.
Movie S5. Applications.

References

1. J. Verbeke, J. D. Schutter, Experimental maneuverability and agility quantification for rotary unmanned aerial vehicle, *International Journal of Micro Air Vehicles* **10**, 3–11 (2018).

2. B. Mishra, D. Garg, P. Narang, V. Mishra, Drone-surveillance for search and rescue in natural disaster, *Computer Communications* **156**, 1–10 (2020).

3. S. M. S. M. Daud, *et al.*, Applications of drone in disaster management: A scoping review, *Science & Justice* **62**, 30–42 (2022).

4. B. Rabta, C. Wankmüller, G. Reiner, A drone fleet model for last-mile distribution in disaster relief operations, *International Journal of Disaster Risk Reduction* **28**, 107–112 (2018).

5. Y. Song, A. Romero, M. Müller, V. Koltun, D. Scaramuzza, Reaching the limit in autonomous racing: Optimal control versus reinforcement learning, *Science Robotics* **8**, eadg1462 (2023).

6. P. Foehn, A. Romero, D. Scaramuzza, Time-optimal planning for quadrotor waypoint flight, *Science Robotics* **6**, eabh1221 (2021).

7. A. Romero, R. Penicka, D. Scaramuzza, Time-optimal online replanning for agile quadrotor flight, *IEEE Robotics and Automation Letters* **7**, 7730–7737 (2022).

8. A. Romero, S. Sun, P. Foehn, D. Scaramuzza, Model predictive contouring control for time-optimal quadrotor flight, *IEEE Transactions on Robotics* **38**, 3340–3356 (2022).

9. E. Kaufmann, *et al.*, Champion-level drone racing using deep reinforcement learning, *Nature* **620**, 982–987 (2023).

10. J. Zhang, R. G. Chadha, V. Velivela, S. Singh, P-cal: Pre-computed alternative lanes for aggressive aerial collision avoidance, *The 12th International Conference on Field and Service Robotics (FSR)* (2019).

11. J. Zhang, R. G. Chadha, V. Velivela, S. Singh, P-cap: Pre-computed alternative paths to enable aggressive aerial maneuvers in cluttered environments, *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), pp. 8456–8463.

12. Y. Ren, S. Liang, F. Zhu, G. Lu, F. Zhang, Online whole-body motion planning for quadrotor using multi-resolution search, *2023 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2023), pp. 1594–1600.

13. Y. Ren, *et al.*, Bubble planner: Planning high-speed smooth quadrotor trajectories using receding corridors, *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2022), pp. 6332–6339.

14. X. Zhou, Z. Wang, H. Ye, C. Xu, F. Gao, Ego-planner: An esdf-free gradient-based local planner for quadrotors, *IEEE Robotics and Automation Letters* **6**, 478–485 (2020).

15. H. Ye, *et al.*, Tgk-planner: An efficient topology guided kinodynamic planner for autonomous quadrotors, *IEEE Robotics and Automation Letters* **6**, 494–501 (2020).

16. A. Loquercio, *et al.*, Learning high-speed flight in the wild, *Science Robotics* **6**, eabg5810 (2021).

17. H. D. Escobar-Alvarez, *et al.*, R-advance: rapid adaptive prediction for vision-based autonomous navigation, control, and evasion, *Journal of Field Robotics* **35**, 91–100 (2018).

18. L. Quan, Z. Zhang, X. Zhong, C. Xu, F. Gao, Eva-planner: Environmental adaptive quadrotor planning, *2021 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2021), pp. 398–404.

19. L. Wang, Y. Guo, Speed adaptive robot trajectory generation based on derivative property of b-spline curve, *IEEE Robotics and Automation Letters* **8**, 1905-1911 (2023).

20. B. Zhou, J. Pan, F. Gao, S. Shen, Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight, *IEEE Transactions on Robotics* **37**, 1992–2009 (2021).

21. H. Oleynikova, Z. Taylor, R. Siegwart, J. Nieto, Safe local exploration for replanning in cluttered unknown environments for microaerial vehicles, *IEEE Robotics and Automation Letters* **3**, 1474–1481 (2018).

22. S. Liu, M. Watterson, S. Tang, V. Kumar, High speed navigation for quadrotors with limited onboard sensing, *2016 IEEE international conference on robotics and automation (ICRA)* (IEEE, 2016), pp. 1484–1491.

23. J. Tordesillas, B. T. Lopez, M. Everett, J. P. How, Faster: Fast and safe trajectory planner for navigation in unknown environments, *IEEE Transactions on Robotics* **38**, 922–938 (2021).

24. E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, D. Scaramuzza, The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam, *The International Journal of Robotics Research* **36**, 142–149 (2017).

25. Livox Technology Company Limited, *Livox Mid-360 User Manual* (2023). Https://www.livoxtech.com/mid-360.

26. Lockheed Martin F-35 Lightning II stealth multirole combat aircraft, `https://en.wikipedia.org/wiki/Lockheed_Martin_F-35_Lightning_II` (2024).

27. DJI Mavic 3, `https://www.dji.com/mavic-3` (2024).

28. J. Tordesillas, B. T. Lopez, J. P. How, Faster: Fast and safe trajectory planner for flights in unknown environments, *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (IEEE, 2019), pp. 1934–1940.

29. Z. Wang, X. Zhou, C. Xu, F. Gao, Geometrically constrained trajectory optimization for multicopters, *IEEE Transactions on Robotics* **38**, 3259-3278 (2022).

30. C. Nous, R. Meertens, C. De Wagter, G. De Croon, Performance evaluation in obstacle avoidance, *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2016), pp. 3614–3619.

31. L. Han, F. Gao, B. Zhou, S. Shen, Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots, *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2019), pp. 4423–4430.

32. B. Tang, *et al.*, Bubble explorer: Fast uav exploration in large-scale and cluttered 3d-environments using occlusion-free spheres, *arXiv preprint arXiv:2304.00852* (2023).

33. P. Foehn, *et al.*, Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight, *Science robotics* **7**, eabl6259 (2022).

34. Nvidia Jetson TX2 embedded AI computing device, `https://developer.nvidia.com/embedded/jetson-tx2` (2024).

35. X. Zhou, *et al.*, Swarm of micro flying robots in the wild, *Science Robotics* **7**, eabm5954 (2022).

36. Skydio 2 Plus Enterprise, An intelligent flying robot powered by AI, https://www.skydio.com/skydio-2-plus-enterprise (2024).

37. A. Harmat, M. Trentini, I. Sharf, Multi-camera tracking and mapping for unmanned aerial vehicles in unstructured environments, *Journal of Intelligent & Robotic Systems* **78**, 291–317 (2015).

38. Intel Realsense Depth Camera D435i, https://www.intelrealsense.com/depth-camera-d435i (2024).

39. W. Xu, Y. Cai, D. He, J. Lin, F. Zhang, Fast-lio2: Fast direct lidar-inertial odometry, *IEEE Transactions on Robotics* **38**, 2053-2073 (2022).

40. S. Liu, *et al.*, Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments, *IEEE Robotics and Automation Letters* **2**, 1688–1695 (2017).

41. F. Gao, W. Wu, W. Gao, S. Shen, Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments, *Journal of Field Robotics* **36**, 710–733 (2019).

42. J. Zhang, C. Hu, R. G. Chadha, S. Singh, Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation, *Journal of Field Robotics* **37**, 1300–1313 (2020).

43. D. Cheng, *et al.*, Treescope: An agricultural robotics dataset for lidar-based mapping of trees in forests and orchards (2023).

44. P. De Petris, *et al.*, Rmf-owl: A collision-tolerant flying robot for autonomous subterranean exploration, *2022 International Conference on Unmanned Aircraft Systems (ICUAS)* (IEEE, 2022), pp. 536–543.

45. Y. Cai, *et al.*, Occupancy grid mapping without ray-casting for high-resolution lidar sensors, *IEEE Transactions on Robotics* pp. 1–20 (2023).

46. Ouster OS1-128 sensor, https://ouster.com/products/scanning-lidar/os1-sensor (2024).

47. DJI Matrice 300 RTK, https://enterprise.dji.com/zh-tw/matrice-300 (2024).

48. LiDAR USA surveyor 32, https://www.lidarusa.com/ (2024).

49. Intel, Product Family D400 Series Datasheet, https://www.intelrealsense.com/download/21345/?tmstv=1697035582 (2024).

50. A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, Octomap: An efficient probabilistic 3d mapping framework based on octrees, *Autonomous robots* **34**, 189–206 (2013).

51. Y. Ren, Y. Cai, F. Zhu, S. Liang, F. Zhang, Rog-map: An efficient robocentric occupancy grid map for large-scene and high-resolution lidar-based motion planning, *arXiv preprint arXiv:2302.14819* (2023).

52. F. Kong, W. Xu, Y. Cai, F. Zhang, Avoiding dynamic small obstacles with onboard sensing and computation on aerial robots, *IEEE Robotics and Automation Letters* **6**, 7869–7876 (2021).

53. H. Wu, Y. Li, W. Xu, F. Kong, F. Zhang, Moving event detection from lidar point streams, *Nature Communications* **15**, 345 (2024).

54. T-MOTOR F90 series racing drone motor specifications, `https://store.tmotor.com/goods-1064-F90.html` (2024).

55. PX4: Open source autopilot for drone developers, `https://px4.io` (2024).

56. J. Chen, K. Su, S. Shen, Real-time safe trajectory generation for quadrotor flight in cluttered environments, *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (IEEE, 2015), pp. 1678–1685.

57. P. Hart, N. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Transactions on Systems Science and Cybernetics* **4**, 100–107 (1968).

58. Q. Wang, Z. Wang, C. Xu, F. Gao, Fast iterative region inflation for computing large 2-d/3-d convex regions of obstacle-free space, *arXiv preprint arXiv:2403.02977* (2024).

59. R. Deits, R. Tedrake, Computing large convex regions of obstacle-free space through semidefinite programming, *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics* (Springer, 2015), pp. 109–124.

60. L. Yin, F. Zhu, Y. Ren, F. Kong, F. Zhang, Decentralized swarm trajectory generation for lidar-based aerial tracking in cluttered environments, *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2023), pp. 9285–9292.

61. J. Ji, N. Pan, C. Xu, F. Gao, Elastic tracker: A spatio-temporal trajectory planner for flexible aerial tracking, *2022 International Conference on Robotics and Automation (ICRA)* (IEEE, 2022), pp. 47–53.

62. S. Liu, N. Atanasov, K. Mohta, V. Kumar, Search-based motion planning for quadrotors using linear quadratic minimum time control, *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (IEEE, 2017), pp. 2872–2879.

63. LBFGS-Lite: A header-only L-BFGS unconstrained optimizer, `https://github.com/ZJU-FAST-Lab/LBFGS-Lite3` (2024).

64. M. Faessler, A. Franchi, D. Scaramuzza, Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories, *IEEE Robotics and Automation Letters* **3**, 620–626 (2017).

65. G. Lu, W. Xu, F. Zhang, On-manifold model predictive control for trajectory tracking on robotic systems, *IEEE Transactions on Industrial Electronics* pp. 1–10 (2022).

66. A. Koubâa, *et al.*, *Robot Operating System (ROS).*, vol. 1 (Springer, 2017).

67. W. Liu, Y. Ren, F. Zhang, Integrated planning and control for quadrotor navigation in presence of suddenly appearing objects and disturbances, *IEEE Robotics and Automation Letters* pp. 1–8 (2023).

68. C. Toumieh, A. Lambert, Voxel-grid based convex decomposition of 3d space for safe corridor generation, *Journal of Intelligent & Robotic Systems* **105**, 87 (2022).

69. X. Zhong, *et al.*, Generating large convex polytopes directly on point clouds, *arXiv preprint arXiv:2010.08744* (2020).

70. C. D. Toth, J. O'Rourke, J. E. Goodman, *Handbook of discrete and computational geometry* (CRC press, 2017).

71. D. Eberly, Distance from a point to an ellipse, an ellipsoid, or a hyperellipsoid (2013).

72. Z. Wang, A geometrical approach to multicopter motion planning, Phd thesis, Zhejiang University, Hangzhou, China (2022).

73. Y. Cui, D. Sun, K.-C. Toh, On the r-superlinear convergence of the kkt residuals generated by the augmented lagrangian method for convex composite conic programming, *Mathematical Programming* **178**, 381–415 (2019).

74. William.Wu, mockamap, `https://github.com/HKUST-Aerial-Robotics/mockamap`.

75. PX4 software in the loop simulation with Gazebo, `https://docs.px4.io/v1.12/en/simulation/gazebo.html` (2024).

Acknowledgments

Funding

This work was supported by the Hong Kong Research Grants Council (RGC) General Research Fund (GRF) 17204523 and a DJI donation.

Author contributions

The research was initiated by Yunfan Ren and Fu Zhang. Yunfan Ren was responsible for the design and manufacture of the UAV prototype, as well as the implementation of the planning modules for the UAV, with assistance from Guozheng Lu and Longji Yin. Fangcheng Zhu and Yixi Cai were responsible for implementing the perception model, while Guozheng Lu and Yunfan Ren implemented the control module with the assistance of Nan Chen. The experiments were designed by Yunfan Ren, Fangcheng Zhu and conducted by Yunfan Ren, Fangcheng Zhu,

Guozheng Lu, Nan Chen, and Fanze Kong. Yunfan Ren and Jiarong Lin performed all data analyses. The manuscript was written by Yunfan Ren and Fu Zhang, with input and advice from all authors. Fu Zhang provided funding for the research and supervised the project.

Competing interests

The authors declare that they have no competing interests.

Data and materials availability

All data are available in the main text or the supplementary materials.

Figures and Tables

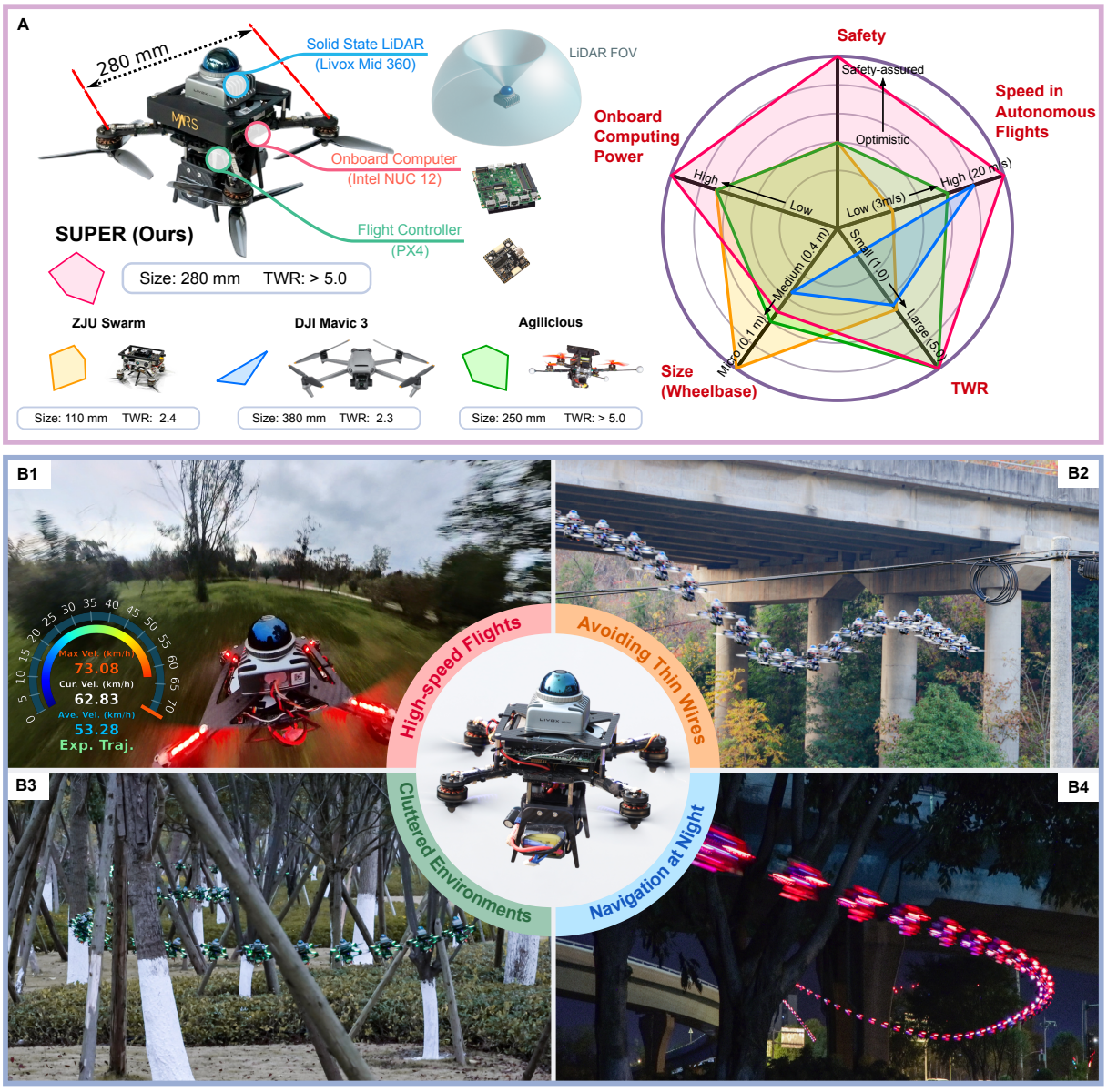

Fig. 1. Overview of the proposed autonomous aerial system (A) A radar chart comparing SUPER with other state-of-the-art autonomous aerial robots, including the palm-sized autonomous MAV (ZJU Swarm) (*35*), the open-sourced agile quadrotor platform (Agilicious) (*33*), and a representative commercial MAV DJI Mavic 3 (*27*). For Agilicious, we obtained the data from its best performance in onboard autonomous flights as reported in (*16*). For DJI Mavic 3, we obtained its size, TWR and the maximum flight speed allowed by its Advanced Pilot Assistance System (APAS) from the official manufacturer's website (*27*). The onboard computation power and safety strategy are not publicly disclosed and hence are not shown. **(B)** Demonstration of SUPER in real-world flights. (B1) High-speed flights in the wild; (B2) Avoidance of thin electrical wires; (B3) Navigation through cluttered environments; (B4) Flights at night.

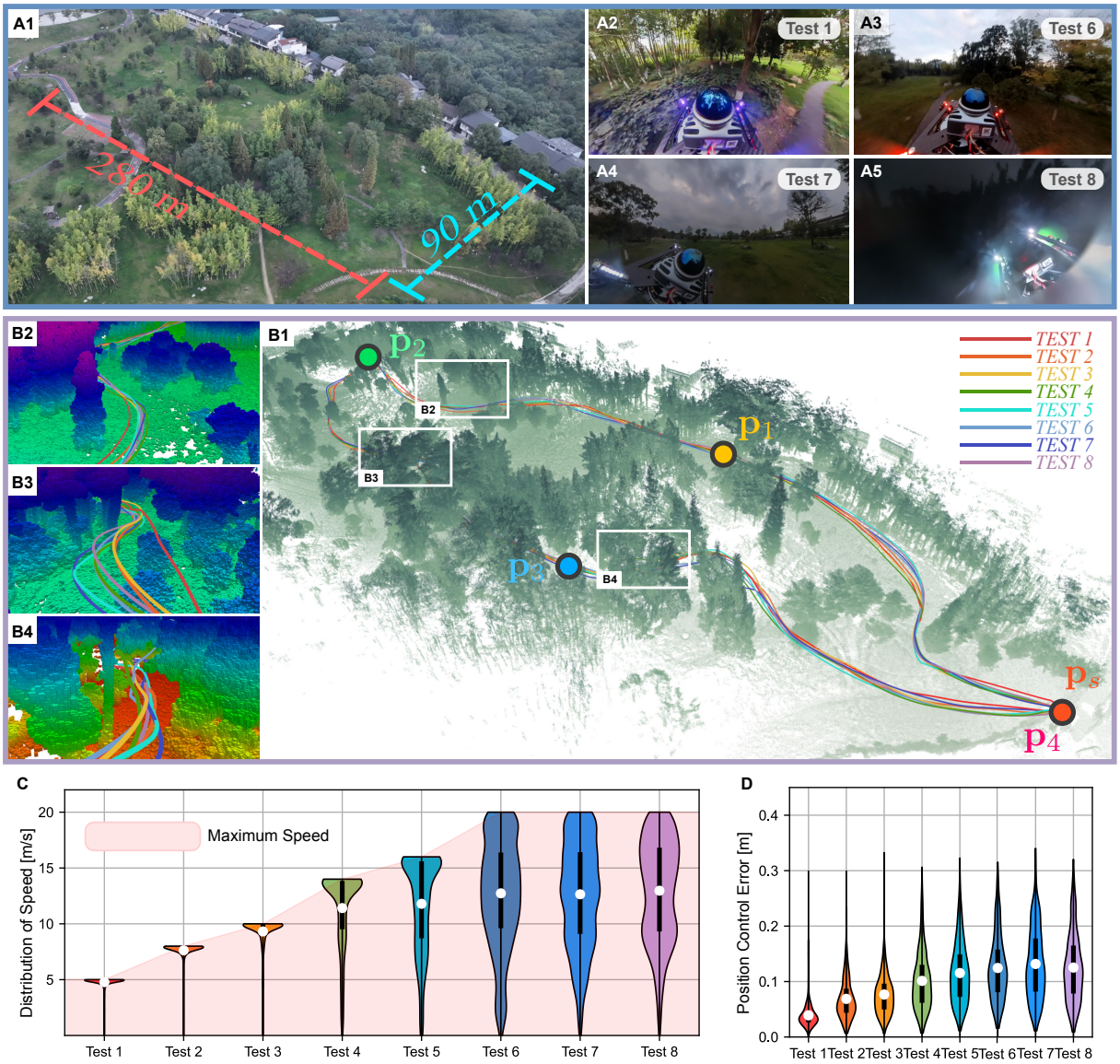

Fig. 2. Safe high-speed navigation in unknown environments (**A**) The bird-view of the experiment environment. (**A2-A5**) Different lighting conditions in eight experiment tests. Specifically, (**A2**) was captured during Test 1 while (**A3-A5**) were captured in Test 6-8, respectively. (**B1**) The executed trajectories during the eight tests are shown in curves with different colors. In all eight tests, the MAV flew through the waypoints $\mathbf{p}_s, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ sequentially and stopped at position $\mathbf{p}_4$. Overlaid with the trajectories is the point cloud map, which is synthesized offline with point measurements collected during the flight. (**B2-B4**) Three detailed views of the environment on flight paths. (**C-D**) Violin plot illustrating the speed distribution and position tracking error in the eight tests, respectively, with the mean value indicated by the white point.

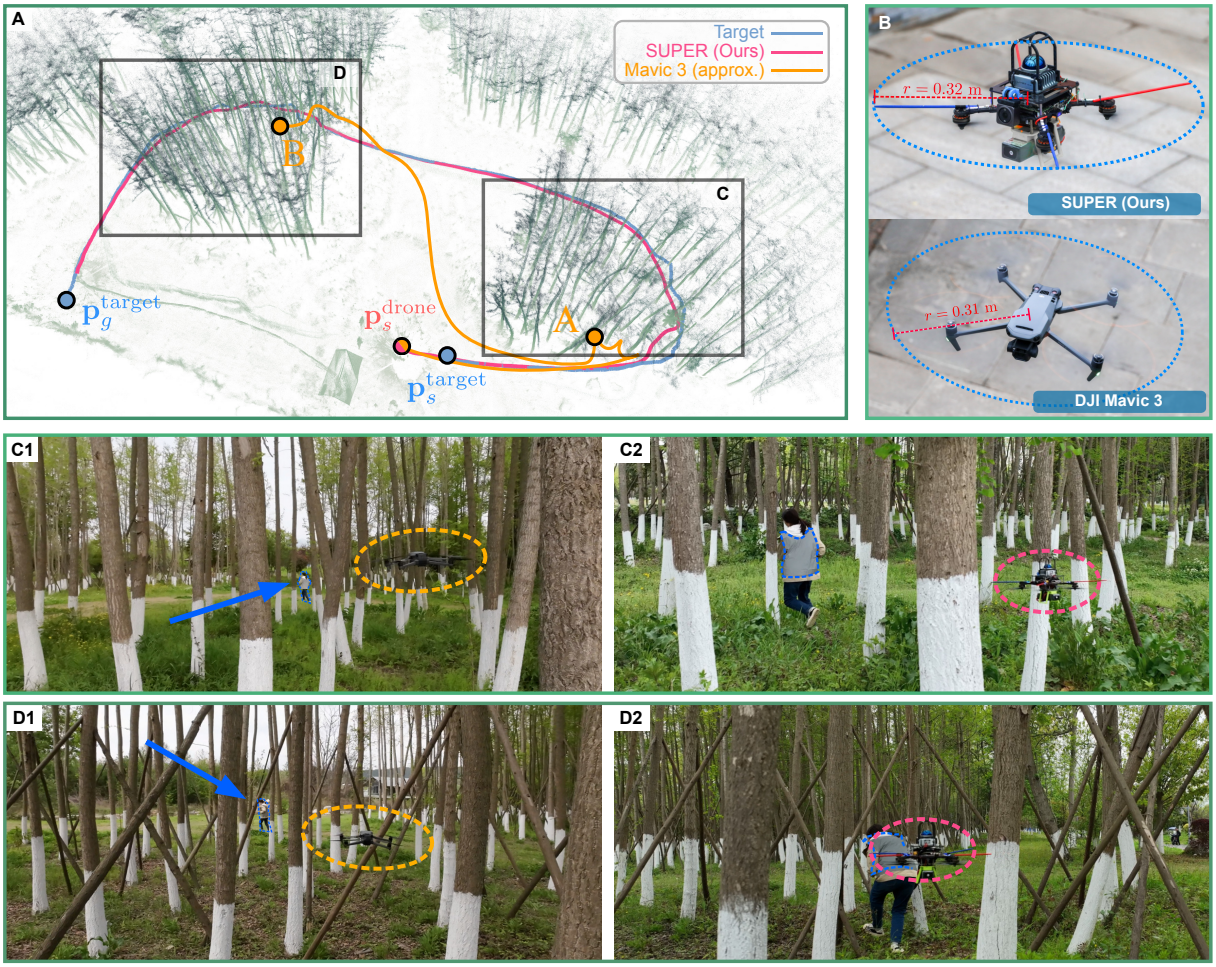

Fig. 3. Navigation in cluttered environments and avoid thin objects. **(A)** Trajectories of the target, Mavic 3 and SUPER during the target tracking experiment. After starting from position $\mathbf{p}_s^{\text{target}}$, the target passed sequentially through two woods areas and ended at position $\mathbf{p}_g^{\text{target}}$. Mavic 3 tracked the target closely until the target entered the first woods area, where Mavic 3 failed to find a collision-free path to track the object and stopped at position A. Once the target exited the first woods area, Mavic 3 resumed the tracking by finding a path detouring the woods. However, Mavic 3 failed the tracking again and stopped at position B when the target entered the second woods area. In contrast, SUPER tracked the target well without a stop throughout the whole process. The point cloud map is synthesized offline with point measurements collected online during the flight of SUPER. **(B)** SUPER with extended size and DJI Mavic 3 (*27*). **(C1, C2)** Tracking of the target when she entered the first woods areas. Mavic 3 failed to track the target due to the tight spaces, while SUPER could track the object closely. **(D1, D2)** Tracking of the target when she entered the second woods areas. Mavic 3 failed the tracking again while SUPER is able to fly through the tight space where the person being tracked even has to lower her body to pass through.

Fig. 4. Avoid thin objects. **(A1-A4)** Four thin wires of varying diameters were used in the experiments. **(B1-B4)** The time-lapse images capturing the flights of DJI Mavic3. It successfully avoided the thin wire of 30 mm diameter but failed to avoid wires with smaller diameters. The collision of Mavic 3 caused the wire to swing, leading to wire-like artifacts in the time-lapse images despite only one actual wire being present in each experiment. **(C1-C4)** The time-lapse images capturing the flights of SUPER. It successfully avoided all four types of thin wires. **(D)** The point cloud view of SUPER when facing a 2.5 mm thin wire. The wire was clearly seen in the current scan measurements (white points) and more evident in the accumulated point cloud (colored points).

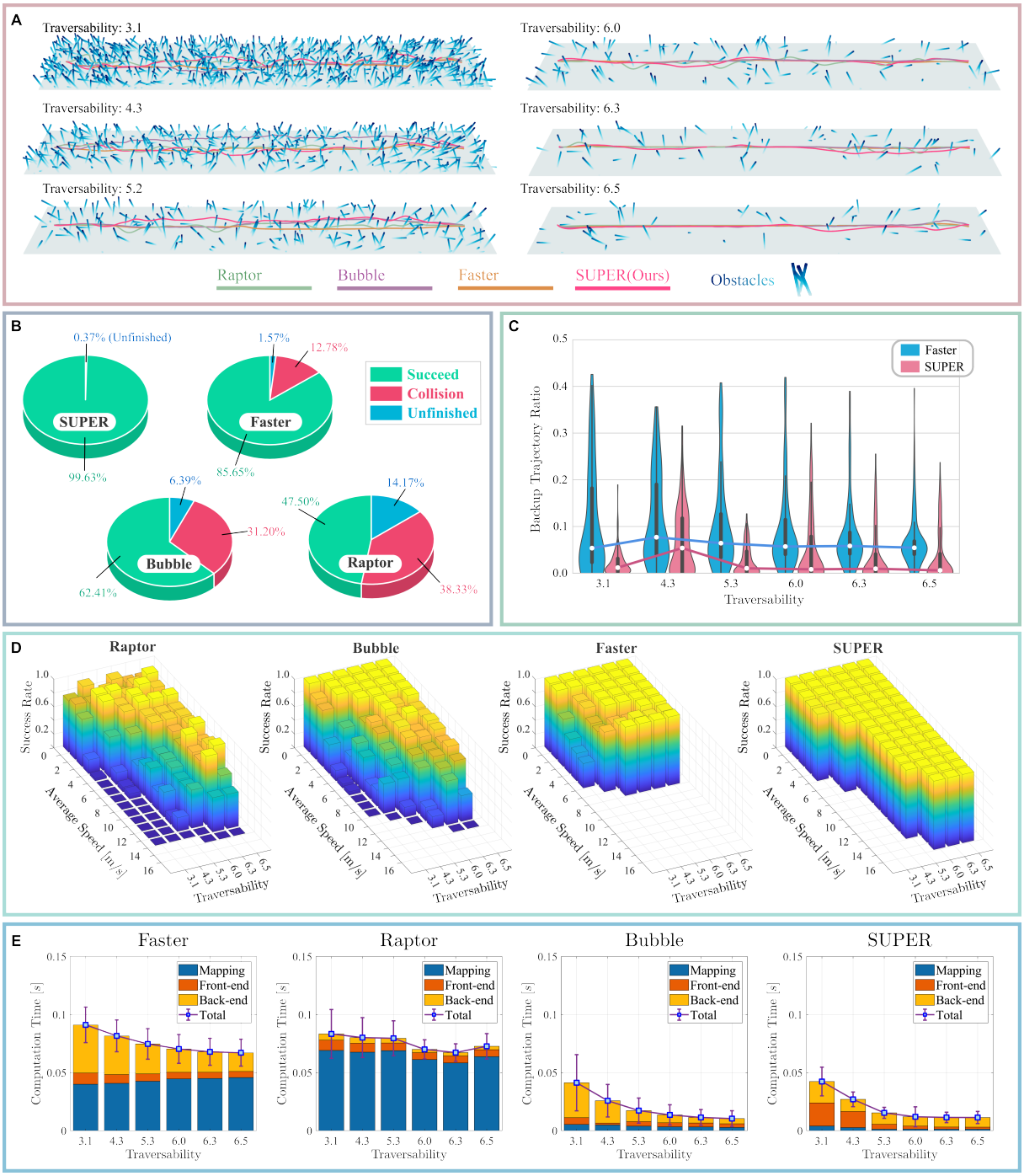

Fig. 5. Evaluation of safety, success rate, and efficiency (**A**) The benchmarking environments with varying traversability. (**B**) Flight results of the benchmarked methods in 1080 experiments. (**C**) The distribution of the ratio of switching to backup trajectories for SUPER and Faster at different obstacle densities. The white points indicate the mean value. (**D**) The success rate of the benchmarked methods at different obstacle densities and flight speeds. Empty columns indicate that the corresponding combination of speed and density was not achieved. (**E**) Time consumption of the benchmarked methods with the circle and error bars indicating the mean value and the standard deviation of the total computation time, respectively.

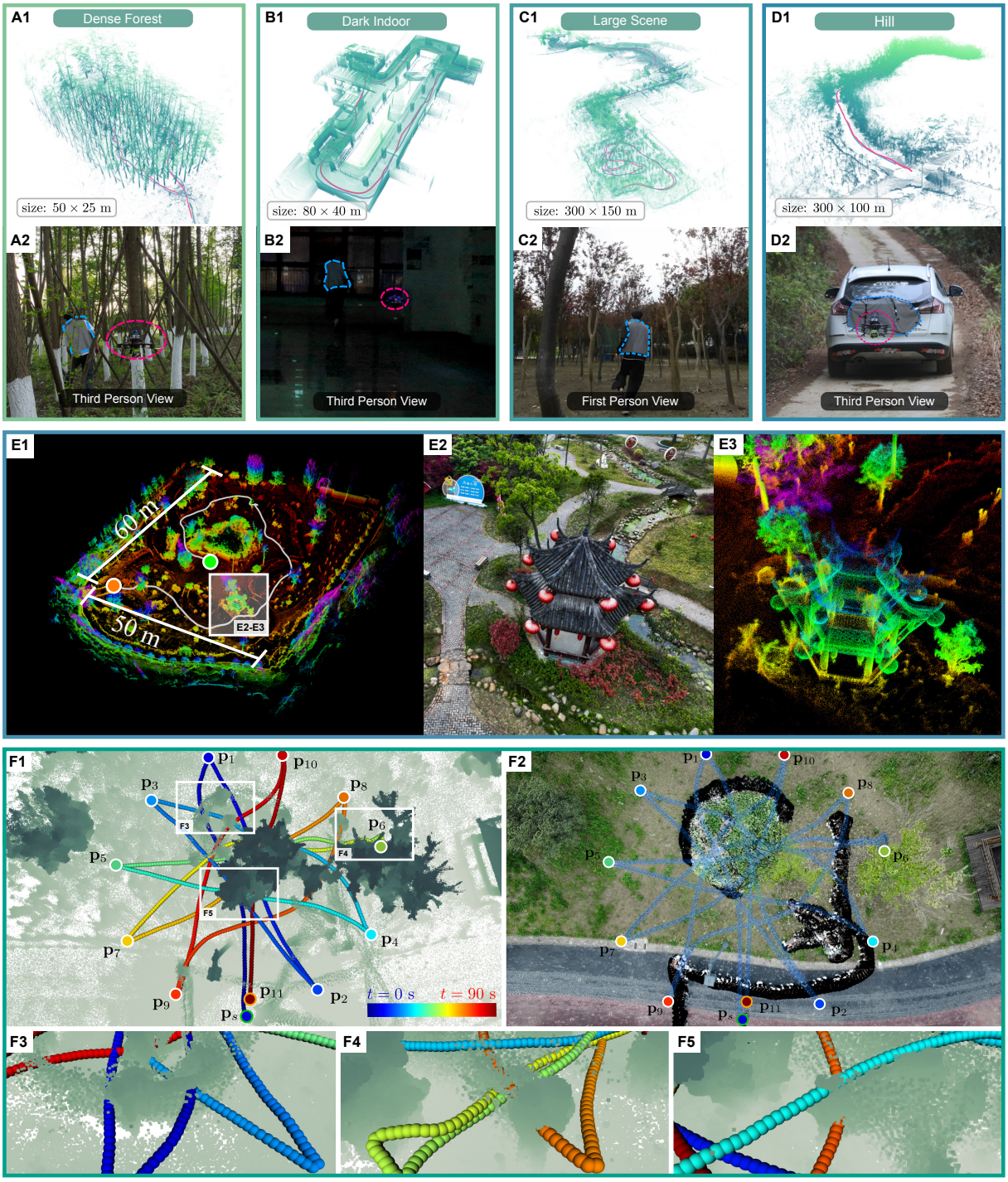

Fig. 6. Additional applications of SUPER. *Object Tracking:* (**A-D**) Tracking of a person running in a dense forest, a dark indoor, a large scene, and a car moving on a hilly village road. The upper pictures show the point cloud maps, which were synthesized from online LiDAR measurements after the flights were completed, with trajectories of the target and the MAV being colored in blue and red, respectively. The lower pictures show snapshots in either third or first-person view during the experiments. *Autonomous Exploration:* (**E1**) Exploration results in a 50 m$\times$ 60 m area with the executed trajectory of SUPER shown in the white path. (**E2-E3**) Snapshot of the scene and the corresponding reconstruction results, respectively. *Waypoints Navigation:* **(F1)** The MAV starting rom $\mathbf{p}_s$ visited 11 waypoints $\mathbf{p}_1 \sim \mathbf{p}_{11}$ sequentially. The MAV trajectory is color-coded based on time. **(F2)** Time-lapse image composition capturing the changes in the environments and the flight trajectories of the MAV. The black artifacts are caused by the two persons moving randomly in the area, and the MAV trajectory is highlighted in blue. (**F3-F5**) The MAV flight trajectories at locations traveled by moving persons. SUPER is able to utilize the spaces previously traversed by moving objects.

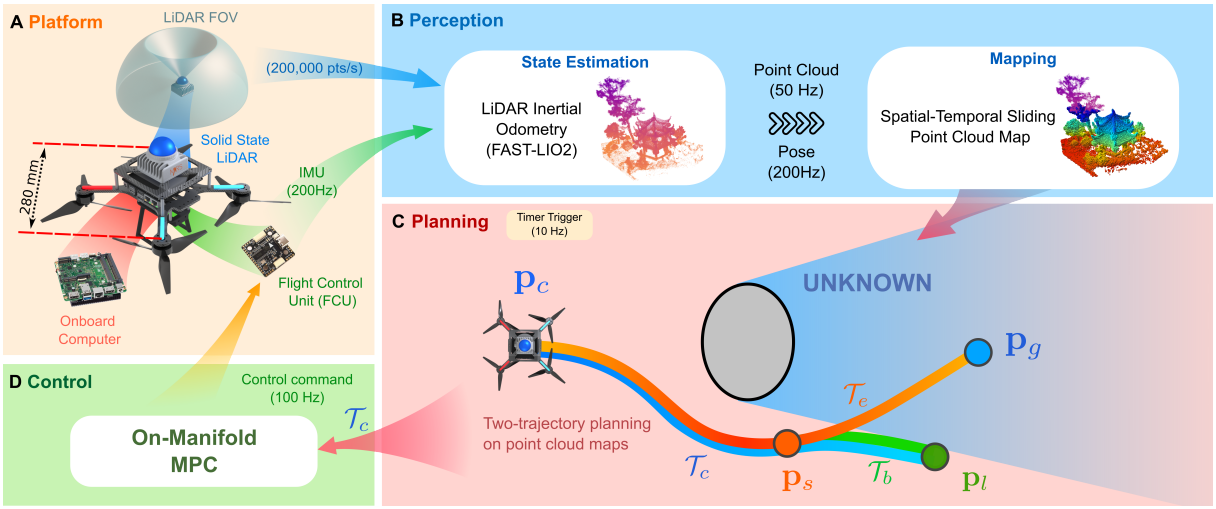

Fig. 7. System overview of SUPER. (**A**) The hardware configuration of SUPER. (**B**) The perception module comprises two components: state estimation and mapping. (**C**) The planning module is a safety-assured planner based on point cloud maps. The planned trajectory includes an exploratory trajectory $\mathcal{T}_e$, which spans both known-free and unknown space, and a backup trajectory $\mathcal{T}_b$, which starts from a state on $\mathcal{T}_e$ and lies entirely within known-free space. Part of $\mathcal{T}_e$ and the entirety of $\mathcal{T}_b$ form the committed trajectory $\mathcal{T}_c$, which is then executed by the MAV. (**D**) The control module is an accurate and efficient on-manifold model predictive controller.

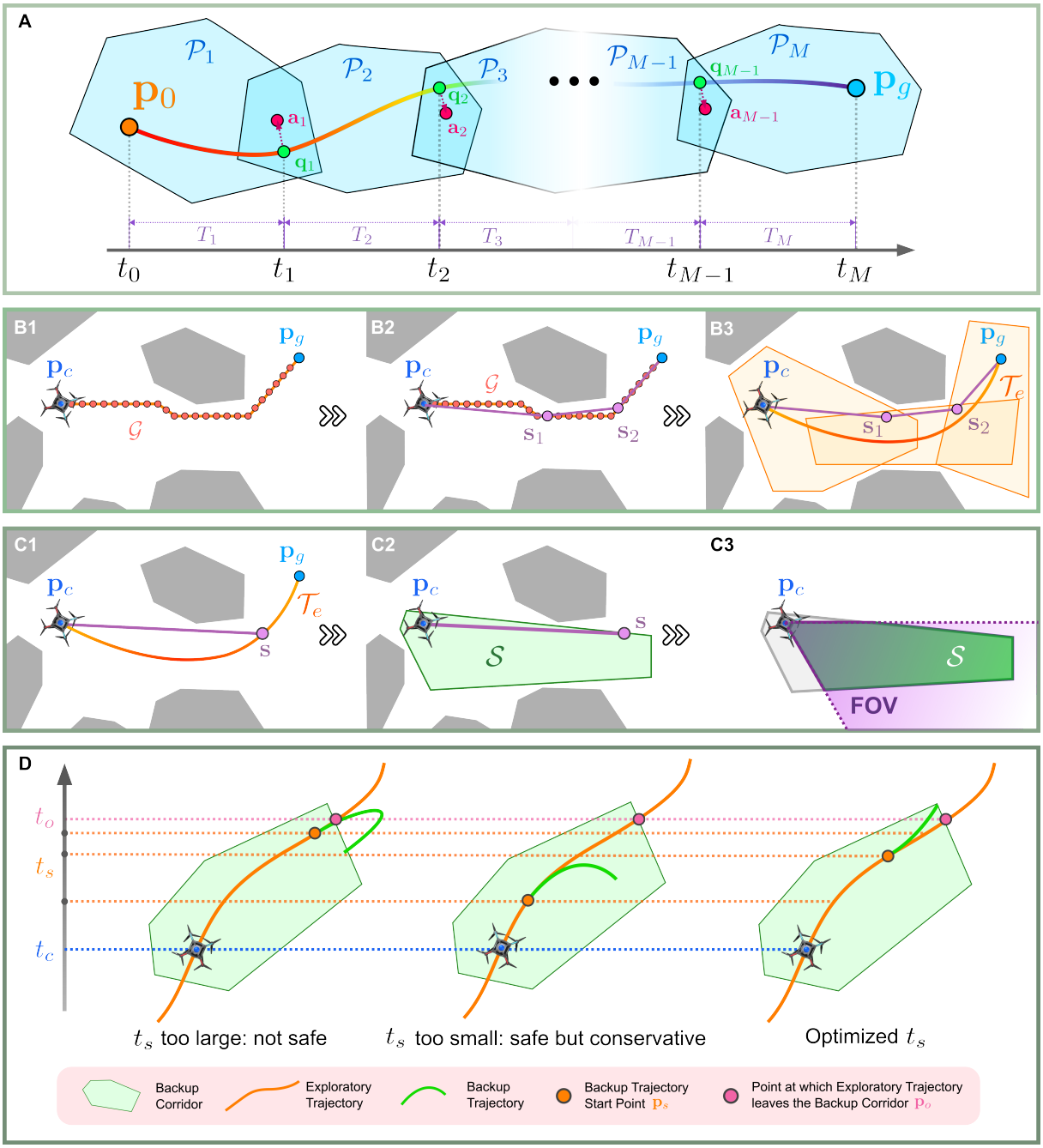

Fig. 8. Two-trajectory planning strategy. (**A**) Illustration of corridor-based trajectory generation method which involves generating a corridor, represented by a series of overlapping polyhedra $\mathcal{P}_i$, and then optimizing a smooth trajectory within the corridor. The trajectory, which begins at $\mathbf{p}_0$ and terminates at $\mathbf{p}_g$, is parameterized by a multi-stage polynomial trajectory through the intermediate position $\mathbf{q}_i$ connecting consecutive trajectory segments and the time duration T_i for each segment. $\mathbf{a}_i$ denotes the centers of the overlapping regions between two adjacent polyhedra. (**B1-B3**) The exploratory trajectory generation process includes an A*-based path search (B1), line seeds generation (B2), and convex decomposition followed by trajectory optimization (B3). (**C1-C3**) The backup corridor generation process involves searching for a seed on the exploratory trajectory (C1), convex decomposition (C2), and optionally cutting the corridor based on the LiDAR field of view (C3). (**D**) The effect of switching times t_s and the corresponding switching position $\mathbf{p}_s$ between the exploratory and backup trajectories. The switching time t_s is optimized within the constraint $t_c < t_s < t_o$.

Supplementary Text

Section S1. System breakdown of SUPER

This section provides a detailed breakdown of the SUPER system with respect to weights, power consumption, and CPU usage (Fig. S1). As illustrated in Fig. S1A, the total hardware weight of SUPER is 1,512 g, comprising the airframe (510 g), battery (432 g), onboard computer (266 g), LiDAR sensor (255 g), and other avionics components (52 g). The power consumption of SUPER was measured during a typical flight mission in a park (see Fig. S2A), with fully onboard sensing, planning, and control on and an average speed of 2.84 m/s. In flight mission, SUPER is commanded to continuously traverse the waypoints following: $\mathbf{p}_1 \rightarrow \mathbf{p}_2 \rightarrow \mathbf{p}_3 \rightarrow \mathbf{p}_4 \rightarrow \mathbf{p}_1 \rightarrow \mathbf{p}_5 \rightarrow \mathbf{p}_3 \rightarrow \mathbf{p}_4 \rightarrow \mathbf{p}_1 \rightarrow \mathbf{p}_6 \rightarrow \mathbf{p}_3 \rightarrow \mathbf{p}_4$ as shown in Fig. S2B. With a 3300 mAh battery, the drone can perform autonomous navigation among a given sequence of waypoints until the battery is out. The total measured flight time is approximately 11 minutes and 24 seconds with a total flight distance of 1,942.56 m (seeFig. S2). The total average power consumption is around 432 W, with 87.9% (379.7 W) consumed by the actuators. The onboard computer is the second-largest power consumer, drawing around 43.3 W. The LiDAR sensor and other avionics components consume 6.3 W and 3.2 W, respectively.

The CPU usage of the onboard computer was also measured during the flight mission. The average CPU usage is around 13.9%, with a peak of 19.7%. The state estimation module is the primary consumer, taking up 11.1% on average. The three main software components, state estimation, planning, and control, are running in parallel as separate processes and communicate with each other through the publisher-subscriber mechanism of the Robot Operating System (ROS) (*66*). Furthermore, the computation within each module is further parallelized using multiple threads to improve efficiency: (i) For the state estimation module, the nearest neighboring search during scan registration is parallelized with 16 threads. (ii) For the planning module, the updating process of the spatio-temporal sliding map and the trajectory planning (including both frontend path search and backend trajectory optimization) are running in two threads. (iii) For the control module, the model predictive controller (MPC) is running in a single thread.

Section S2. Benchmark comparison on LiDAR-based navigation methods

To compare the performance of the planning module of SUPER and state-of-the-art LiDAR-based navigation system Falco (*42*), Bubble planner (*13*) and IPC (*67*), this section provides a detailed benchmark comparison in simulation. The setup of the simulation test is the same as in the main results "Evaluation of Safety, Success Rate, and Efficiency" of the manuscript. We have tested two versions: (i) *Falco w/o Gimbal*: The official open-source implementation of Falco, which does not include a gimbal structure as reported in their real-world experiments. When the drone accelerates or decelerates, the LiDAR field of view (FOV) and the corresponding motion primitives will be aligned with the UAV's body frame (see Fig. S3A). (ii) *Falco with Gimbal*: We modified the official Falco version by adding a gimbal structure in the simulation to maintain the LiDAR sensor's horizontal orientation independent of the drone's attitude (see Fig. S3B). The gimbal structure is also used in their real-world experiments to achieve high-speed flights.

The results are shown in Fig. S4, where the result of *Bubble* is directly drawn from the main results. As can be seen, the *Falco with Gimbal* approach achieves an overall success rate of

57.04% and a maximum attainable average speed exceeding 10 m/s, which is in agreement with the performance presented in the original paper (*42*). In contrast, the *Falco w/o Gimbal* configuration, the default open-source Falco implementation, exhibits a lower overall success rate of only 40.28% and a maximum average speed below 6 m/s, even in the sparsest environment. Regarding *IPC*, it achieves a high success rate of 75%, outperforming both versions of Falco and the Bubble planner. While it shows a relatively lower success rate in low traversability environments due to its optimistic planning strategy, the success rate increases significantly as traversability improves. However, in terms of maximum achievable speed, *IPC* falls behind both *Bubble* and *SUPER*. These results demonstrate that *SUPER* achieves a significantly higher success rate compared to *Falco*, *Bubble* and *IPC*, while attaining much higher average flight speeds. The comparison of computation times is illustrated in Fig. S4C. *Falco*, with its offline trajectory library and map-less planning framework, requires less computation time than both *Bubble* and *SUPER*. For *IPC*, it generates only one or two polyhedra in each replan cycle, leading to lower computation costs on the frontend. Additionally, its MPC-based planning and control framework takes less than 1 ms for the backend, resulting in much lower overall computational costs.

Section S3. Implementation Details of Different Baseline Methods

We provide a concise comparison of the implementation details of different baseline methods in the "Evaluation of Safety, Success Rate, and Efficiency" section of the main text. Table S1 presents the comparison of these methods across four aspects. Firstly, the methods are categorized into three types based on their safety strategies: optimistic, safety-aware, and safety-assured. Bubble (*13*) is an optimistic method that treats unknown spaces as free. Raptor (*20*) is a safety-aware method that actively plans trajectories to enhance the visibility of unknown spaces but lacks safety guarantees. Faster (*23*) and our proposed SUPER are both safety-assured methods with theoretical safety guarantees. Secondly, the mapping approaches differ among the methods. Raptor employs an Euclidean signed distance field (ESDF)-based map (*31*), while Faster uses an occupancy grid map (OGM). In contrast, Bubble and SUPER adopt a point cloud-based method, with Bubble additionally utilizing a KD-Tree map for corridor generation. Generally, the point cloud-based mapping approach in Bubble and SUPER exhibits significantly lower computation time compared to Faster and Raptor. All four baselines adopt a frontend and backend style in trajectory generation, with different methods employed for each. Bubble, Faster, and SUPER utilize flight corridor-based frontends, while Raptor employs topological path search. For the backend, Faster uses a mixed-integer quadratic programming (MIQP)-based method, and Raptor uses path-guided trajectory optimization. However, both Faster's MIQP-based method and Raptor's path-guided optimization only optimize the spatial properties of the trajectory and have limited freedom to find feasible trajectories during high-speed flight, which necessitates temporal optimization. In contrast, both Bubble and SUPER incorporate spatio-temporal trajectory optimization based on MINCO (*29*), demonstrating superior performance in high-speed flights, as verified in the main text.

Section S4. Evaluation of the Planning Module of Faster and SUPER

We evaluated the planning module of Faster (*23*) and SUPER in a simulation, using a challenging indoor corner environment where an obstacle is concealed behind the corner, as depicted in Fig. S6A1. Additionally, we conducted an ablation study on SUPER, specifically SUPER

w/o OT, which is a variant of SUPER that disables switching time optimization. In each test, the drone commenced at $\mathbf{p_s} = [0, 0, 1.2]^{\mathrm{T}}$, with the goal set to $\mathbf{p_g} = [12.5, 7.5, 1.2]^{\mathrm{T}}$. All three methods achieved the goal safely. Both Faster and SUPER w/o OT switched to a backup trajectory twice when the drone detected the obstacle behind the corner. However, due to the computationally demanding MIQP-based trajectory optimization employed by Faster, it failed to plan a new trajectory transition from the backup trajectory and execute the backup trajectory till a stop. Consequently, the execution time of the backup trajectory was significantly longer for Faster compared to SUPER w/o OT, resulting in a total time of 6.2 s for Faster, considerably exceeding the 4.9 s recorded for SUPER w/o OT. Furthermore, the full version of SUPER, incorporating the proposed backup trajectory switching time optimization approach, did not switch to the backup trajectory and avoid unnecessary braking maneuvers. SUPER achieved the shortest arrival time of 4.8 s, showcasing the high-speed capability of the proposed planning module.

Additionally, we conducted a controlled, quantitative analysis of the backup trajectory generation between Faster and SUPER. As illustrated in Fig. S6C, both planners were provided with the same exploratory trajectory and backup corridor. The resulting backup trajectories and switching times t_s are depicted by the green curve and orange circle, respectively. Faster employed a heuristic approach to determine the switch time (*23*), setting the switching time t_s to be 0.45 s later than the drone's current state. In comparison, SUPER simultaneously optimized the switch time and backup trajectory, aiming to maximize t_s while ensuring the backup trajectory remained within the backup corridor. This optimization process yielded a switching time of $t_s = 0.84$ s for SUPER. Consequently, SUPER delayed the switching time by 86% compared to Faster, allowing more time for the planning module to re-plan new exploratory and backup trajectories. As a result, the portion of executing the backup trajectory was reduced, as shown in Fig. S6B, enhancing the overall flight speed and smoothness.

To verify the performance of our planning module when integrated into the complete system, we reproduced the aforementioned simulation experiments in a real-world setting, as depicted in Fig. S6D-F. The drone started at $\mathbf{p_s} = [0, 0, 1.2]^{\mathrm{T}}$ and was assigned a goal at $\mathbf{p_g} = [12.5, -7.5, 1.2]^{\mathrm{T}}$, with a maximum speed limit of $\mathbf{v}_{\text{max}} = 8$ m/s, mirroring the simulation setup. As shown in Fig. S6E, at time $t_1 = 2.1$ s, the obstacle was occluded by the wall, and the generated exploratory trajectory $\mathcal{T}_e^1$ collided with the obstacle. However, the backup trajectory $\mathcal{T}_b^1$ was confined within the backup corridor, ensuring that in the worst-case scenario, the drone could safely come to a stop without colliding with any obstacle. At $t_2 = 2.2$ s, the drone perceived the obstacle for the first time (Fig. S6D2) and re-planned a new exploratory trajectory $\mathcal{T}_e^2$ to avoid it. Consequently, the initial backup trajectory $\mathcal{T}_b^1$ was not executed, and a new backup trajectory was generated instead. The velocity profile is shown in Fig. S6F, demonstrating that the drone did not execute any backup trajectory and successfully reached the goal at a high average speed, highlighting the high-speed and safe properties of SUPER.

Section S5. Convex Decomposition in Configuration Space

Convex decomposition is a technique used to identify a convex shape around the given seed (e.g., a point or a line segment) that can effectively separate a convex region of space from a set of obstacles. In robotic navigation missions, performing convex decomposition in the configuration space and plan trajectories in the decomposed spaces (*12, 13, 23, 40*) is a popular way to efficiently achieve collision avoidance. Currently, two mainstream methods exist for

generating free polyhedra in the configuration space (C-space). (i) Point Inflation (*12, 23, 60*): The first method involves point inflation and performs convex decomposition on the inflated point cloud (Fig. S7A1). However, the point inflation exponentially increases the number of point clouds and negatively impacts computational efficiency. Additionally, the inflated obstacle point clouds often fail to represent the configuration space accurately, leading to small and sharp polyhedra (see Fig. S7A1) that may over-constrained the trajectory optimization problems. (ii) Plane Shrinkage (*40,61*): The second method generates the convex shape directly on the original point cloud and subsequently shrinks it by the robot's radius to obtain the free polyhedron in C-space (Fig. S7A2). However, this approach focuses on maximizing the polyhedron before shrinking, which can lead to the resultant polyhedron being small and sharp after the shrinking process. Additionally, there is a risk that the resulting shrunken polyhedron may not encompass the seed, thereby limiting its applicability in scenarios where it is crucial for the seed to reside within the generated polyhedron.

To address those issues, we propose the Configuration-space Iterative Regional Inflation (CIRI) algorithm. Unlike previous approaches (*40, 58, 59, 68, 69*) that represent obstacles as points, we model obstacle points as spheres with a radius equal to the robot's radius. This approach allows us to generate free polyhedra in the C-space directly, offering several advantages. Firstly, our algorithm operates directly on the original point clouds without the need for inflation. This approach reduces the number of points involved, resulting in a substantial improvement in computational efficiency. Secondly, the accurate modeling of obstacles in the configuration space enables our method to find polyhedra with larger sizes compared with the shrinkage-based or inflation-based methods. Additionally, we propose a plane adjustment method that ensures the inclusion of the seed in the generated polyhedra, which is helpful when generating backup corridors.

The proposed CIRI builds upon the iterative process of the FIRI framework (*58*) (see Algorithm 1), offering improved computational efficiency and generating larger polyhedra. The algorithm takes a line segment seed $\mathbf{S} : \{\mathbf{s}_a, \mathbf{s}_b\}$ and the obstacle point cloud $\mathcal{O}$ as input, and begins with the initialization of the ellipsoid as a unit ball (line 2) and proceeds iteratively until the ellipsoid's volume converges. The iterative process comprises two steps: (i) uniformly inflating the ellipsoid to tangentially intersect the obstacle point cloud, generating a set of hyperplanes that separate a convex region from the obstacles while ensuring the inclusion of the seed (line 7); and (ii) employing a second-order cone programming (SOCP)-based optimization to identify the maximum-volume ellipsoid within the polyhedron formed by the obstacle-free half-spaces defined by those hyperplanes (line 8). Through iterative refinement, the hyperplanes and the ellipsoid are adjusted to progressively expand the volume of the inscribed ellipsoid, resulting in a larger polyhedron and an increased region of obstacle-free space.

Ellipsoid-based Convex Decomposition

In this step, our objective is to determine a free polyhedron comprising a collection of separating hyperplanes. These hyperplanes ensure that the given seed is devoid of obstacles. Each hyperplane is denoted by $\mathcal{H}(A, b)$, where $A\mathbf{x} \leq b, \mathbf{x} \in \mathbb{R}^3$ defines a half-space. The polyhedron $\mathcal{P}$ is defined by intersection of K half-spaces: $\mathcal{P} : \{\mathcal{H}_1, \ldots, \mathcal{H}_k\}$. To represent the ellipsoid, we use the image of the unit ball, which is known as the Löwner-John ellipsoid (*70*), defined as follows:

$$\mathbf{E}(\mathbf{C}, \mathbf{d}) = \{\mathbf{C}\tilde{\mathbf{x}} + \mathbf{d} | \ \|\tilde{\mathbf{x}}\| \leq 1\}, \tag{S1}$$

Algorithm 1: Configuration-space Iterative Regional Inflation (CIRI)

Notation: Identity matrix: $\mathbf{I} \in \mathbb{R}^{3\times3}$; Ellipsoid $\mathbf{E}$ centered at $\mathbf{d}$ with the shape matrix $\mathbf{C}$; Converge threshold: ε
Input : Obstacle point cloud: $\mathcal{O}$; Robot's radius: r; Seed: $\mathbf{S}$
Output : Polyhedron: $\mathcal{P}$

1 **Function** `CIRI` $(\mathcal{O}, r, \mathbf{S})$
2 $\quad \mathbf{E}.\mathbf{C} = \mathbf{I}$;
3 $\quad \mathbf{E}.\mathbf{d} = (\mathbf{s}_a + \mathbf{s}_b)/2$;
4 $\quad$ converge = **False**;
5 $\quad$ **while *not* *converge*** **do**
6 $\quad\quad \mathbf{C}_{last}$ = $\mathbf{E}.\mathbf{C}$;
7 $\quad\quad \mathcal{P}$ = `EllipsoidCvxDecomp` $(\mathcal{O}, r, \mathbf{S}, \mathbf{E})$;
8 $\quad\quad \mathbf{E}$ = `MaximumInscribeVolumeEllipsoid` $(\mathcal{P})$;
9 $\quad\quad$ **if** $[\mathbf{det}(\mathbf{E}.\mathbf{C}) - \mathbf{det}(\mathbf{C}_{last})]\,/\mathbf{det}(\mathbf{C}_{last}) < \varepsilon$ **then**
10 $\quad\quad\quad$ converge = **True**;
11 $\quad\quad$ **end**
12 $\quad$ **end**
13 $\quad$ `return` $(\mathcal{P})$;
14 **End Function**

where $\mathbf{C} \in \mathbb{R}^{3\times3}$ is a diagonal shape matrix defined by $\mathbf{C} = \mathbf{diag}(r_x, r_y, r_z)$ and $\mathbf{d} \in \mathbb{R}^3$ is the center of the ellipsoid. We also define its uniform expansion as:

$$\mathbf{E}_\alpha(\mathbf{C}, \mathbf{d}) = \{\mathbf{C}\tilde{\mathbf{x}} + \mathbf{d} |\ \|\tilde{\mathbf{x}}\| \le \alpha\}, \tag{S2}$$

where $\alpha \ge 1$ is the expansion factor. We use the same definition to describe a sphere, where $\mathbf{C} = \mathbf{diag}(r, r, r)$ and $\mathbf{d}$ is the sphere's center.

The input of the proposed method consists of the obstacle point cloud $\mathcal{O} \in \mathbb{R}^{3\times N}$ with N points, the ellipsoid in the world frame $\mathbf{E}^{\mathcal{W}}$, the seed $\mathbf{S}$ and the robot's radius r. The process is outlined in Algorithm 2 and visualized in Fig. S7C.

At first, the activating obstacle point clouds $\mathcal{O}_a$ are initialized with the full obstacle point clouds $\mathcal{O}$ (line 2). Then, we continued to generate hyperplanes to separate $\mathcal{O}_a$ until it became empty. In each iteration, we choose the nearest obstacle sphere, denoted as $\mathbf{E}_s^{\mathcal{W}}$, to the center of the ellipsoid $\mathbf{E}$ (line 4) and try to find the separating hyperplanes between the $\mathbf{E}_s^{\mathcal{W}}$ and the ellipsoid. The key to finding these hyperplanes involves locating the planes that intersect the obstacles and are tangent to uniformly expanded ellipsoid $\mathbf{E}_\alpha^{\mathcal{W}}$. Building upon the concept presented in (*59*), we simplify the problem by transforming it to the ellipsoid frame $\mathcal{E}$ and form it into a single least-distance programming problem, thus avoiding searching over different expansion factors α. In (S1), we define the ellipsoid in world frame $\mathcal{W}$ as the image of the unit ball in the ellipsoid frame $\mathcal{E}$. Thus, we construct the inverse map and transform the problem to the ellipsoid frame $\mathcal{E}$ as shown in Fig. S7B. The intersecting point $\mathbf{p}_t^{\mathcal{W}}$'s image $\mathbf{p}_t^{\mathcal{E}}$ is the closest point to the origin of the $\mathcal{E}$ frame on $\mathbf{E}_s^{\mathcal{E}}$. Now, the problem is turned into a minimum distance optimization problem.

$$\begin{aligned} &\mathbf{p}_t^{\mathcal{E}} = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_2, \\ &\text{s.t.} \quad \mathbf{x} \in \mathbf{E}_s^{\mathcal{E}}. \end{aligned} \tag{S3}$$

Algorithm 2: Ellipsoid-based Convex Decomposition

Params : Active obstacle point cloud $\mathcal{O}_a$;
Input : Obstacle point cloud: $\mathcal{O}$; Robot's radius: r;
Seed: $\mathbf{S}$; Ellipsoid: $\mathbf{E}^{\mathcal{W}}$
Output : Polyhedron: $\mathcal{P}$

1 **Function** `GeneratePolytope` $(\mathcal{O}, r, \mathbf{S}, \mathbf{E})$
2 $\mathcal{O}_a = \mathcal{O}$;
3 **while not** $\mathcal{O}_a$.`empty`() **do**
4 $\mathbf{E}_s^{\mathcal{W}}$= `FindCloestObstacle` $(\mathbf{E}^{\mathcal{W}}, \mathcal{O}_a)$;
5 $\mathcal{H}$ = `ComputeSeparatingPlanes` $(\mathbf{E}_s^{\mathcal{W}})$;
6 **foreach** $\mathbf{s} \in \mathbf{S}$ **do**
7 $\mathcal{H}$ = `CheckAndAdjustPlane` $(\mathcal{H}, \mathbf{s}, \mathbf{E}_s^{\mathcal{W}})$;
8 **end**
9 $\mathcal{O}_a = \mathcal{O}_a \setminus (\mathcal{H} \cap \mathcal{O}_a)$;
10 $\mathcal{P}$.`AddPlane`$(\mathcal{H})$;
11 **end**
12 `return` $(\mathcal{P})$;
13 **End Function**

This problem can be solved efficiently by finding the root of a 6-th-degree polynomial (*71*). We implement it with C++, and on an Intel i7 CPU, the typical solving time is less than $0.5\,\mathrm{us}$. After finding the tangent point $\mathbf{p}_t^{\mathcal{E}}$, we can find the half space $\mathcal{H}^{\mathcal{E}}(A^{\mathcal{E}}, b^{\mathcal{E}}) = (\mathbf{p}_t^{\mathcal{E}}/\|\mathbf{p}_t^{\mathcal{E}}\|, -\|\mathbf{p}_t^{\mathcal{E}}\|)$ and the expansion factor $\alpha = \|\mathbf{p}_t^{\mathcal{E}}\|$. Then, we can transform them back to get the $\mathcal{H}^{\mathcal{W}}$ in the world frame:

$$\mathcal{H}^{\mathcal{W}}(\mathbf{C}\mathbf{A}^{\mathcal{E}}, b^{\mathcal{E}} + [\mathbf{C}\mathbf{A}^{\mathcal{E}}] \cdot \mathbf{d}), \tag{S4}$$

where $\mathbf{C}, \mathbf{d}$ is the shape matrix and the center of $\mathbf{E}$ respectively.

The process above finds a separating hyperplane between the ellipsoid and the obstacle spheres. However, it is possible that the seed does not lie in the half-space defined by the generated hyperplanes. To address this issue, we propose a plane adjustment method to ensure that the generated polyhedron includes the seed (line 6). As depicted in Fig. S7D, we first verify if both $\mathbf{s}_a$ and $\mathbf{s}_b$ lie within the half-space defined by $\mathcal{H}$. Without loss of generality, if the seed point $\mathbf{s}_a$ is located outside $\mathcal{H}$, we adjust $\mathcal{H}$ to a new hyperplane $\mathcal{H}_{adj}$ that passes through point $\mathbf{s}_a$, which is tangent to the obstacle sphere, and points towards another seed point $\mathbf{s}_b$. This adjustment ensures that the new hyperplane $\mathcal{H}_{adj}$ contains both $\mathbf{s}_a$ and $\mathbf{s}_b$. We then set the hyperplane generated in this round as the new $\mathcal{H}$, denoted as $\mathcal{H}_{adj}$. After the plane adjustment process (if necessary), all obstacle points outside $\mathcal{H}$ are excluded from $\mathcal{O}_a$, and $\mathcal{H}$ is added to the polyhedron $\mathcal{P}$. This process is repeated until all obstacle points are excluded.

Maximum Volume Inscribed Ellipsoid

Given the polyhedron (i.e., a set of hyperplanes), this step involves updating the ellipsoid by determining the maximum volume inscribed ellipsoid (MVIE) within the given polyhedron. Assume the polyhedron consists of K hyperplanes $\mathcal{P} = \{\mathcal{H}_1, \ldots, \mathcal{H}_K\}$ and the ellipsoid is define by (S1), the optimization problem can be formulated as:

$$\begin{aligned} \max_{\mathbf{C},\mathbf{d}} \quad & \det(\mathbf{C}), \\ \text{s.t.} \quad & \sup(A_i\mathbf{C}\tilde{\mathbf{x}}) + A_i\mathbf{d} \leq b_i, \forall i = 1, \ldots, K, \\ & \mathbf{C} \succeq 0, \end{aligned} \tag{S5}$$

where $\mathbf{C}$ and $\mathbf{d}$ are the shape matrix and the center of the ellipsoid, respectively. A_i and b_i are the parameters of the plane equation form of the hyperplane $\mathcal{H}(A, b) : A\mathbf{x} \leq b$. The optimization problem (S5) is a well-studied convex optimization problem in computational geometry. In this paper, we utilize the formulation proposed in (*72*) to convert (S5) as a Second-Order Cone Program (SOCP) and can be efficiently solved following the Conic Augmented Lagrangian method (*73*).

Comparison on Convex Decomposition

We conducted a comparative analysis of convex decomposition methods in three distinct environments characterized by different obstacle shapes: Pillars (Fig. S8A1), Forest (Fig. S8A2), and Perlin (Fig. S8A3) (*74*). For each obstacle type, we randomly generated 300 scenarios with varying densities of obstacles. To quantify the obstacle density of a scenario, we used the obstacle filling rate (OFR), which is calculated as the ratio of the total volume of obstacles to the total volume of the space. The 900 scenarios over different obstacle type and densities lead to an OFR ranging from 0.0005 to 0.6. For each scenario, we sample a point cloud map with a spatial resolution of 0.1 m, along with a random line seed that was ensured to be at least 0.2 m (i.e., the robot radius) away from the nearest obstacles to ensure the existence of a feasible polyhedron in the configuration space. The point cloud map and line seeds are then fed to different convex decomposition methods.

We compared three approaches that perform convex decomposition in the configuration space (see Fig. S7A): (i) Plane Shrinkage: The polyhedron is first generated on the original point cloud and then shrink each hyperplane by the robot's radius. (ii) Point Inflation: The input point clouds are inflated by adding more points within cubes centered at each input point. The cube length is equal to the robot's radius. Convex decomposition is then performed on the inflated point cloud, and the generated polyhedron is guaranteed in the configuration space. (iii) CIRI (Ours): The proposed Configuration-space Iterative Regional Inflation method, which models each input point as a sphere of radius equal to the robot radius and then performs convex decomposition on the point spheres. For (i) and (ii), we implement the baseline methods based on two state-of-the-art convex decomposition implementations: FIRI (*58*) and RILS (*40*), resulting in FIRI-Shrinkage, FIRI-Inflation, RILS-Shrinkage and RILS-Inflation, respectively.

We evaluated each method in terms of seed containment rate (i.e., the ratio of polyhedra that contains the line seed), computation time, and volume of the resulting polyhedra (see Fig. S8B). Considering the seed containment rate (Fig. S8B1), CIRI and the point inflation (either FIRI-inflation or RILS-inflation) achieved 100% seed containment rate in all tests. In contrast, the plane shrinkage-based methods (either FIRI-shrinkage or RILS-shrinkage) exhibited a significantly lower seed containment rate, as they can only ensure the seed is within the polyhedron before the shrinking process but not after. Regarding the volume of the generated polyhedra (Fig. S8B2), CIRI consistently produced polyhedra with the largest average volume compared to the other four baselines, thanks to its precise modeling of the configuration space. In terms of computational time (Fig. S8B3), CIRI and the two shrinkage-based baselines, FIRI-shrinkage and RILS-shrinkage, had significantly lower computation times compared to the two inflation-

based baselines. This is because the point inflation introduces significantly more points in the input point cloud, which substantially increases the computation time. Compared to the shrinkage-based methods, CIRI has slightly higher computation time due to the need to solve the tangent plane between a sphere and an ellipsoid, while the other two methods solve the point-ellipsoid tangent plane, which is computationally easier.

Since plane shrinkage-based methods, including FIRI-Shrinkage and RILS-Shrinkage, do not guarantee the containment of line seed, they are not suitable for SUPER. Comparing among the seed-containment-assured methods, CIRI can generate polyhedra with larger volumes while requiring significantly lower computation time than point inflation methods (FIRI-inflation and RILS-inflation). In conclusion, CIRI effectively enhances corridor quality within less computational time, thereby further improving the overall performance of SUPER.

Section S6. Proof of Theorem 1

Proof. Assume that $\mathcal{S}$ is not entirely contained within the known-free space. This implies the existence of a point $\mathbf{p}$ with an occupancy state that is either occupied or unknown. In the first case, if $\mathbf{p}$ is in the occupied space, it must be captured in $\mathcal{D}$, which contradicts condition (b). In the second case, if $\mathbf{p}$ is in the unknown space, its depth must be greater than any point $\mathbf{d} \in \mathcal{D}$ with the same bearing direction. For the point $\mathbf{d}$ lies on the line connecting $\mathbf{p}$ and $\mathbf{p}_c$ (i.e., the bearing direction), the convexity of $\mathcal{S}$ implies that $\mathbf{d} \in \mathcal{S}$. This once again contradicts condition (b). Hence, in both cases, we arrive at a contradiction to condition (b), leading to the conclusion that $\mathcal{S}$ must be entirely contained within the known-free space. □

Section S7. Trajectory Planning with Limited LiDAR FOV

When considering the LiDAR sensor's limited field of view (FOV), the backup corridor must intersect with the FOV to ensure that the intersecting space lies entirely within the known-free area. One most common type of LiDAR sensor has a 360-degree horizontal FOV and a limited vertical FOV, as shown in Fig. S11A1. Assuming the vertical FOV angle is $\theta < \pi$ (Fig. S11A2), the total FOV forms a non-convex shape. To ensure that the intersected backup corridor is convex, we must select a convex subset of the FOV and intersect it with the convex polytope generated by CIRI with this subset. The largest convex subset of the 360-degree horizontal FOV is the intersection of two half-spaces defined by hyperplanes that are tangent to the FOV's upper and lower bounds, as shown in Fig. S11A1 and A2. The FOV has infinitely many such convex subsets, each uniquely determined by a one degree-of-freedom (DOF) heading direction, $\mathbf{h}$, as illustrated in Fig. S11B1. To ensure that the resultant backup corridor contains the largest possible exploratory trajectory, we align $\mathbf{h}$ with the tangent direction of the exploratory trajectory (Fig. S11B2).

To investigate the influence of the vertical FOV angle θ on planning performance, we conducted a series of experiments in a simulated environment with vertical FOV angles ranging from 30 degrees to 180 degrees (i.e., omni-directional FOV). The testing environment is similar to that used in the benchmark comparisons in the main manuscript. We selected two types of simulation environments with different traversability scores: one with a traversability of 3.1, indicating a dense environment, and another with a traversability of 6.5, representing a relatively sparse environment. For each traversability setting, we tested 10 maps, running 10 trials per map without limiting the maximum velocity. For each trial, we recorded the average speed, the ratio of executing the backup trajectory, and the final outcome (i.e., success or failure). The

results are presented in Fig. S12.

Considering the success rate, SUPER achieved a 100% success rate in both environments, regardless of lower or higher traversability (Fig. S12A1, B1), highlighting its safety-assured properties. In the denser environment, the average speed increased as the vertical FOV became larger (Fig. S12A2), which is due to the fact that a larger FOV enables generation of larger volume backup corridors, reducing the need to execute the backup trajectory (Fig. S12A3), resulting in a higher average speed. In the sparser environment, the average flight speed also showed a similar trend, though the influence of FOV size was less significant (Fig. S12B2). This is because, in environments with fewer obstacles, even a small FOV is sufficient for planning safe, high-speed trajectories, and the ratio of executing the backup trajectory is less affected by the FOV size (Fig. S12B3).

Section S8. On-manifold MPC with Rotor Drag Compensation

In our prior work (*65*), we have proposed an on-manifold model predictive control (OMMPC) framework for robotic systems and demonstrated indoor quadrotor aerobatics. However, the aerodynamic effects were neglected in the previous work, which significantly affects the tracking accuracy when the vehicle performs high-speed flights in outdoor environments. To overcome this limitation, we extend our previous quadrotor model by incorporating rotor drag, which is the main aerodynamic effect causing tracking error (*64*). We derive the resultant OMMPC and implement it for all flight experiments in this paper, demonstrating sufficiently high tracking accuracy. The effectiveness of this improvement is also verified through comparative experiments.

Quadrotor dynamics

The quadrotor coordinate directions X, Y, and Z are defined as forward, right, and down, respectively. The quadrotor dynamics involving rotor drag (*64*) are formulated as follows:

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v}, \\ \dot{\mathbf{v}} &= \mathbf{g} - a_T \mathbf{R}\mathbf{e}_3 - \frac{1}{m}\mathbf{R}\mathbf{D}\mathbf{R}^{\mathrm{T}}\mathbf{v}, \\ \dot{\mathbf{R}} &= \mathbf{R}\lfloor \boldsymbol{\omega}^{\mathcal{B}} \rfloor, \end{aligned} \tag{S6}$$

where m is the mass of the drone, $\mathbf{g}$ represents the gravity vector with a fixed length of 9.81 m/s^2, and a_T denotes the scalar thrust acceleration. The position and velocity in the inertial frame are denoted by $\mathbf{p} \in \mathbb{R}^3$ and $\mathbf{v} \in \mathbb{R}^3$, respectively. The attitude of the airframe is represented by $\mathbf{R} \in SO(3)$, and $\boldsymbol{\omega}$ denotes angular velocity in the body frame. The operation $\lfloor \cdot \rfloor$ converts a vector into a skew-symmetric matrix. The operation $\lfloor \cdot \rfloor$ converts a vector into a skew-symmetric matrix. $\mathbf{D}$ denotes the rotor drag coefficient matrix, which is parameterized as $\mathbf{D} = \mathbf{diag}(d_h, d_h, d_v)$.

As discussed in (*64*), the above quadrotor dynamic system with rotor drag is differentially flat. The flat output vector is typically selected as $\sigma = [\mathbf{p}, \psi]^{\mathrm{T}}$, where ψ represents the yaw angle. This differential flatness property allows us to plan the trajectory of the flat output $\sigma(t)$ instead of the entire state trajectory $\mathbf{s}(t)$ of the quadrotor UAV.

System linearization

As previously described in our previous work (*65*), OMMPC takes the quadrotor system into a compound manifold, and defines the system states and inputs as follows:

$$\begin{aligned} &\mathcal{M} = \mathbb{R}^3 \times \mathbb{R}^3 \times SO(3), \quad \dim(\mathcal{M}) = 9, \\ &\mathbf{x} = \begin{pmatrix} \mathbf{p}^{\mathcal{I}} & \mathbf{v}^{\mathcal{I}} & \mathbf{R} \end{pmatrix} \in \mathcal{M}, \quad \mathbf{u} = \begin{bmatrix} a_T & \boldsymbol{\omega}^{\mathcal{B}} \end{bmatrix} \in \mathbb{R}^4, \end{aligned} \tag{S7}$$

and the error between measured state $\mathbf{x}$ and reference state $\mathbf{x}_d$, lying on the manifold $\mathcal{M}$, is mapped into the local homeomorphic space (an open set in Euclidean space) around each point $\mathbf{x}_d$ using local coordinates. Particularly, the state error is defined as follows:

$$\begin{aligned} \delta\mathbf{x} &\triangleq \mathbf{x}_d \boxminus \mathbf{x} = \begin{bmatrix} \delta\mathbf{p}^{\mathrm{T}} & \delta\mathbf{v}^{\mathrm{T}} & \delta\mathbf{R}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^9, \\ \delta\mathbf{p} &\triangleq \mathbf{p}_d \boxminus \mathbf{p} = \mathbf{p}_d - \mathbf{p} \in \mathbb{R}^3, \\ \delta\mathbf{v} &\triangleq \mathbf{v}_d \boxminus \mathbf{v} = \mathbf{v}_d - \mathbf{v} \in \mathbb{R}^3, \\ \delta\mathbf{R} &\triangleq \mathbf{R}_d \boxminus \mathbf{R} = \mathrm{Log}(\mathbf{R}^{\mathrm{T}}\mathbf{R}_d) \in \mathbb{R}^3, \end{aligned} \tag{S8}$$

where $\boxminus$ is an encapsulated operator implemented to the mapping $\boxminus : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^n$ as mentioned in (*65*), and the input error is also defined as

$$\begin{aligned} \delta\mathbf{u} &\triangleq \mathbf{u}_d - \mathbf{u} = \begin{bmatrix} \delta a_T & \delta\boldsymbol{\omega}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^4, \\ \delta a_T &\triangleq a_{T_d} - a_T \in \mathbb{R}, \\ \delta\boldsymbol{\omega} &\triangleq \boldsymbol{\omega}_d - \boldsymbol{\omega} \in \mathbb{R}^3, \end{aligned} \tag{S9}$$

and the original quadrotor system can be equivalently expressed by an error system where the vehicle states are mapped to the local coordinates at each point along the reference trajectory. Follow the derivation in (*65*), the resultant linearized error system is given by

$$\delta\mathbf{x}_{k+1} = \mathbf{F}_{\mathbf{x}_k}\delta\mathbf{x}_k + \mathbf{F}_{\mathbf{u}_k}\delta\mathbf{u}_k, \tag{S10}$$

where

$$\begin{aligned} \mathbf{F}_{\mathbf{x}_k} &= \begin{bmatrix} \mathbf{I}_3 & \mathbf{I}_3\Delta t & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 - \frac{\Delta t}{m}\mathbf{R}_k^d\mathbf{D}\mathbf{R}_k^{d\mathrm{T}} & \frac{\Delta t}{m}\mathbf{R}_k^d\left(2\mathbf{I}_3 - \mathbf{D}\right)\left\lfloor a_T m\mathbf{e}_3 + \mathbf{D}\mathbf{R}_k^{d\mathrm{T}}\mathbf{v} - \mathbf{R}_k^{d\mathrm{T}}\mathbf{v}\right\rfloor \\ \mathbf{0} & \mathbf{0} & \mathrm{Exp}\left(-\boldsymbol{\omega}_k^d\Delta t\right) \end{bmatrix}, \\ \mathbf{F}_{\mathbf{u}_k} &= \Delta t \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ -\mathbf{R}_k^d\mathbf{e}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}\left(\boldsymbol{\omega}_k^d\Delta t\right)^{\mathrm{T}} \end{bmatrix}, \end{aligned} \tag{S11}$$

where

$$\mathbf{A}(\boldsymbol{\theta}) = \mathbf{I}_3 + \left(\frac{1-\cos\|\boldsymbol{\theta}\|}{\|\boldsymbol{\theta}\|}\right)\frac{\lfloor\boldsymbol{\theta}\rfloor}{\|\boldsymbol{\theta}\|} + \left(1 - \frac{\sin\|\boldsymbol{\theta}\|}{\|\boldsymbol{\theta}\|}\right)\frac{\lfloor\boldsymbol{\theta}\rfloor^2}{\|\boldsymbol{\theta}\|^2}. \tag{S12}$$

Therefore, the OMMPC can be given by the current error and reference trajectory, leading to an efficient QP solution:

$$\begin{aligned} \min_{\delta\mathbf{u}_k} &\sum_{k=0}^{N-1}\left(\|\delta\mathbf{x}_k\|_{\mathbf{Q}}^2 + \|\delta\mathbf{u}_k\|_{\mathbf{R}}^2\right) + \|\delta\mathbf{x}_N\|_{\mathbf{P}}^2, \\ \text{s.t.} \quad &\delta\mathbf{x}_{k+1} = \mathbf{F}_{\mathbf{x}_k}\delta\mathbf{x}_k + \mathbf{F}_{\mathbf{u}_k}\delta\mathbf{u}_k, \\ &\delta\mathbf{u}_k \in \delta\mathbb{U}_k, \quad k = 0, \cdots, N-1, \end{aligned} \tag{S13}$$

where N is the predictive horizon, and positive-definite diagonal matrices $\mathbf{Q}, \mathbf{R}, \mathbf{P}$ denote the penalty of the stage state, stage input, and terminal state, respectively. $\mathbb{U}_k = \{\delta\mathbf{u} \in \mathbb{R}^4 | \mathbf{u}_{\min} - \mathbf{u}_k^d \le \delta\mathbf{u} \le \mathbf{u}_{\max} - \mathbf{u}_k^d\}$ denotes the constraints for the input error.

Experiment Results

To quantitatively assess the effectiveness of the proposed OMMPC incorporating the rotor-drag model, we conducted a comparative analysis with the common quadrotor model (*65*) using the PX4 SITL software with the Gazebo simulator (*75*). The simulator provides realistic dynamics and air resistance simulations. We generated an offline polynomial trajectory on the flat space denoted as $\sigma(t)$ with a maximum velocity of 15 m/s. The trajectory consisted of a straight line starting from $\mathbf{p}_0 = [0, 0, 0]^{\mathrm{T}}$ and ending at $\mathbf{p}_g = [120, 0, 0]^{\mathrm{T}}$ along the x-axis. Initially, we generated the desired state trajectory using differential flatness under the common quadrotor model and tracked the trajectory using the quadrotor OMMPC proposed in (*65*). As shown in Fig. S13A, the UAV exhibited significant attitude tracking errors, and the position tracking error was notably influenced by the velocity direction, reaching a maximum of 0.4 m. In contrast, by employing the quadrotor model with rotor drag described in equation (S6), the OMMPC effectively predicted and compensated for the drag, resulting in improved attitude-tracking performance and a reduced position-tracking error of 0.05 m.

Supplementary Figures and Tables

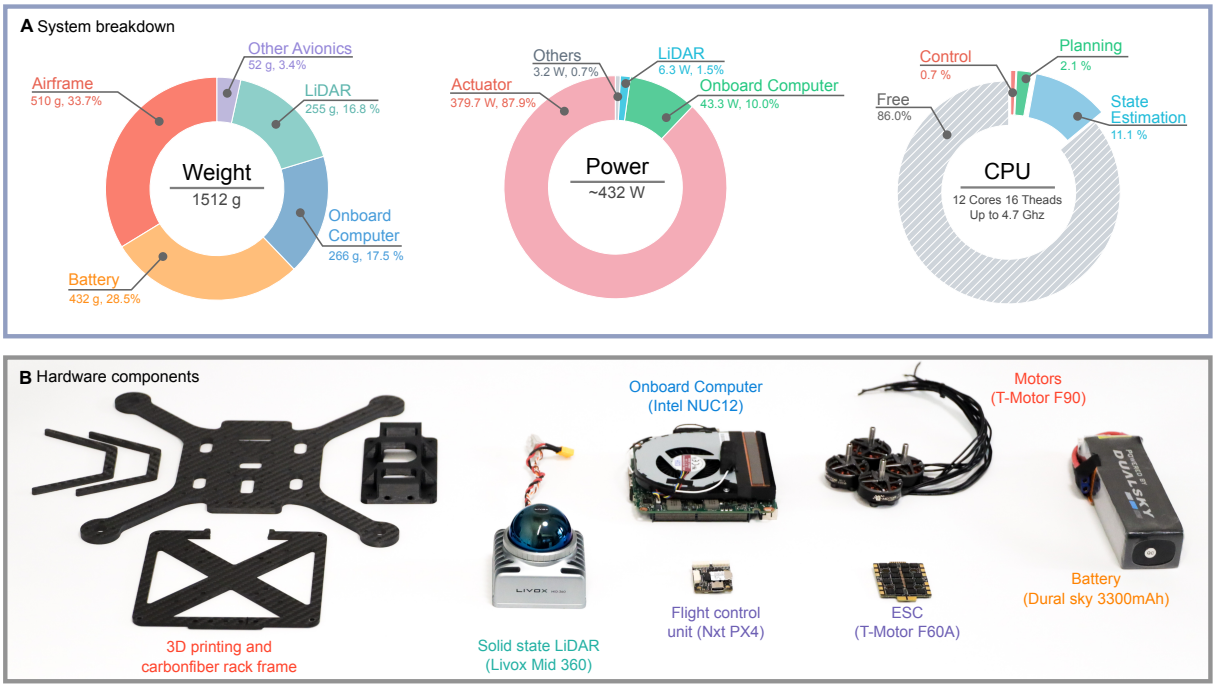

Fig. S1. System breakdown of SUPER (**A**) The weight, power, and CPU usage breakdown of SUPER. (**B**) The hardware components of SUPER.

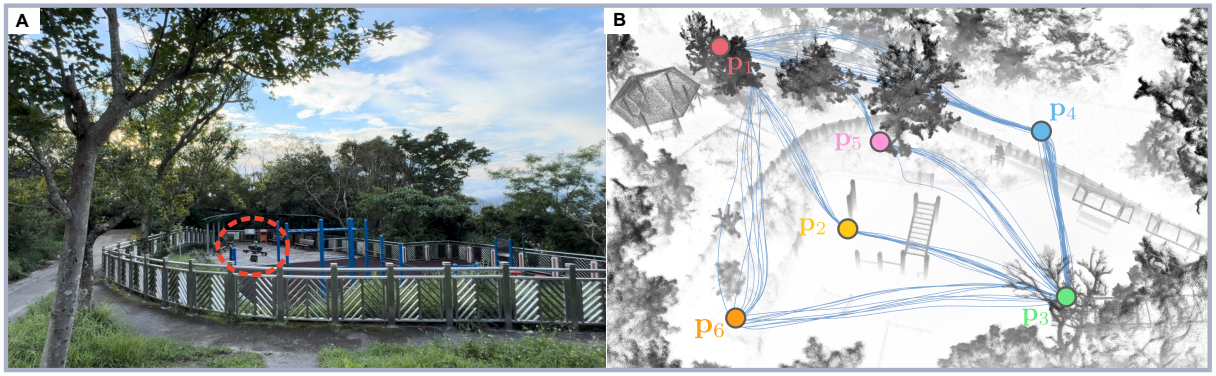

Fig. S2. Scenario and trajectory of the endurance test (**A**) The testing scenario. (**B**) The point cloud view and flying trajectory of the drone during the endurance test.

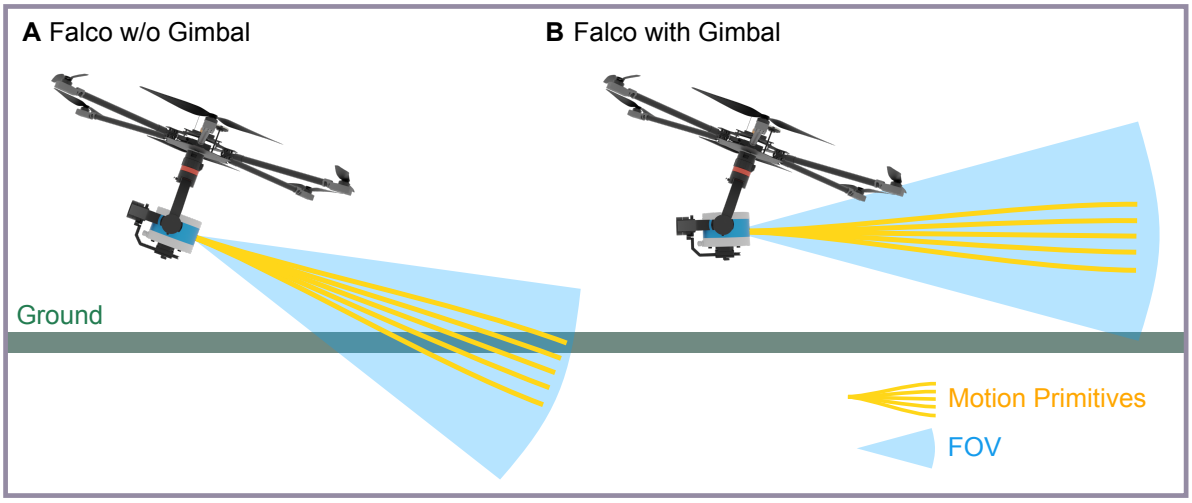

Fig. S3. Illustration of Falco with and without gimbal structure (**A**) The original version of Falco's open-sourced implementation lacked a gimbal structure. In this configuration, when the multirotor accelerates, the motion primitives are constrained to point downwards, prohibiting the drone from flying forward. (**B**) The modified version of Falco with a gimbal structure. The LiDAR sensor maintains a horizontal orientation independent of the drone's attitude.

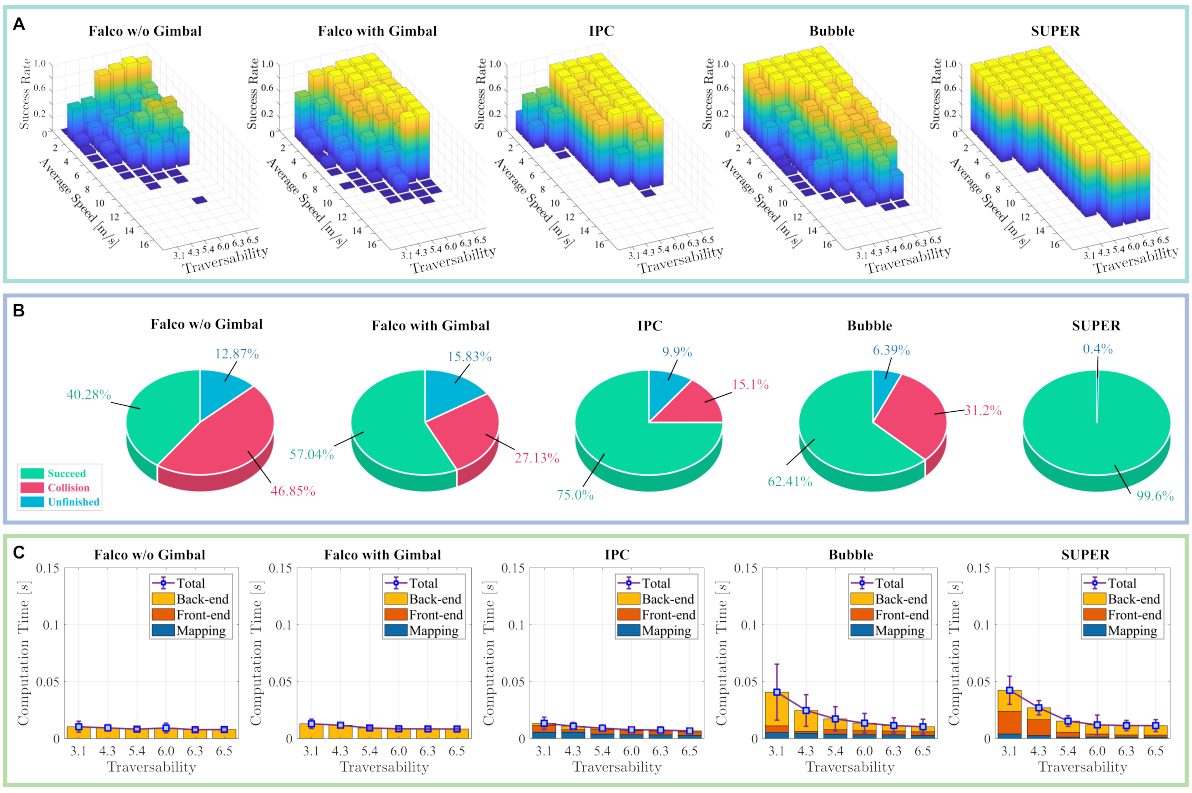

Fig. S4. Evaluation of success rate, and efficiency (**A**) The success rate of the benchmarked methods at different obstacle densities (measured as traversibility) and flight speeds. Empty columns indicate that the corresponding combination of speed and density was not achieved. (**B**) Flight results of the benchmarked methods in 1080 experiments. (**C**) Time consumption of the benchmarked methods with the circle and error bars indicating the mean value and the standard deviation of the total computation time, respectively.

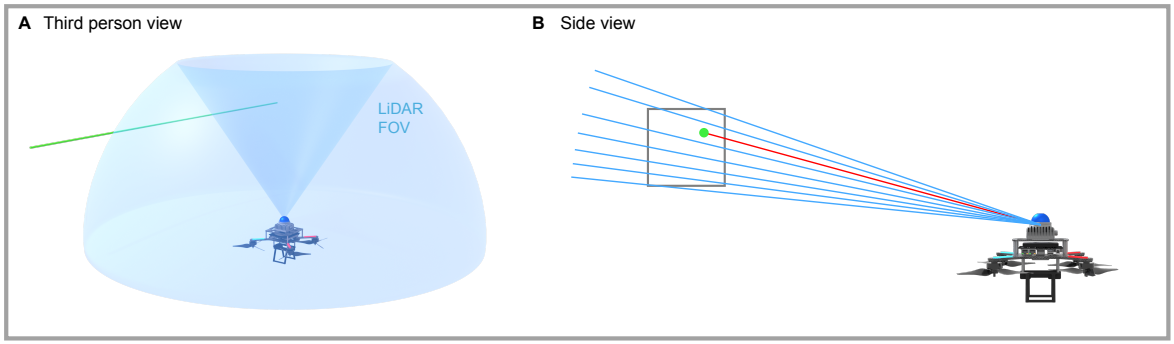

Fig. S5. Visual example of thin wire detection on OGM. (**A**) Third-person view of the MAV facing a thin wire (shown in green). (**B**) Side view of the detection process. The gray box represents the cell in the occupancy grid map (OGM) that contains the thin wire. The blue lines indicate the LiDAR beams that pass through the cell without hitting the wire, while the red line indicates the beam that hits the wire. Since most of the beams do not detect the wire, the cell is incorrectly classified as free space.

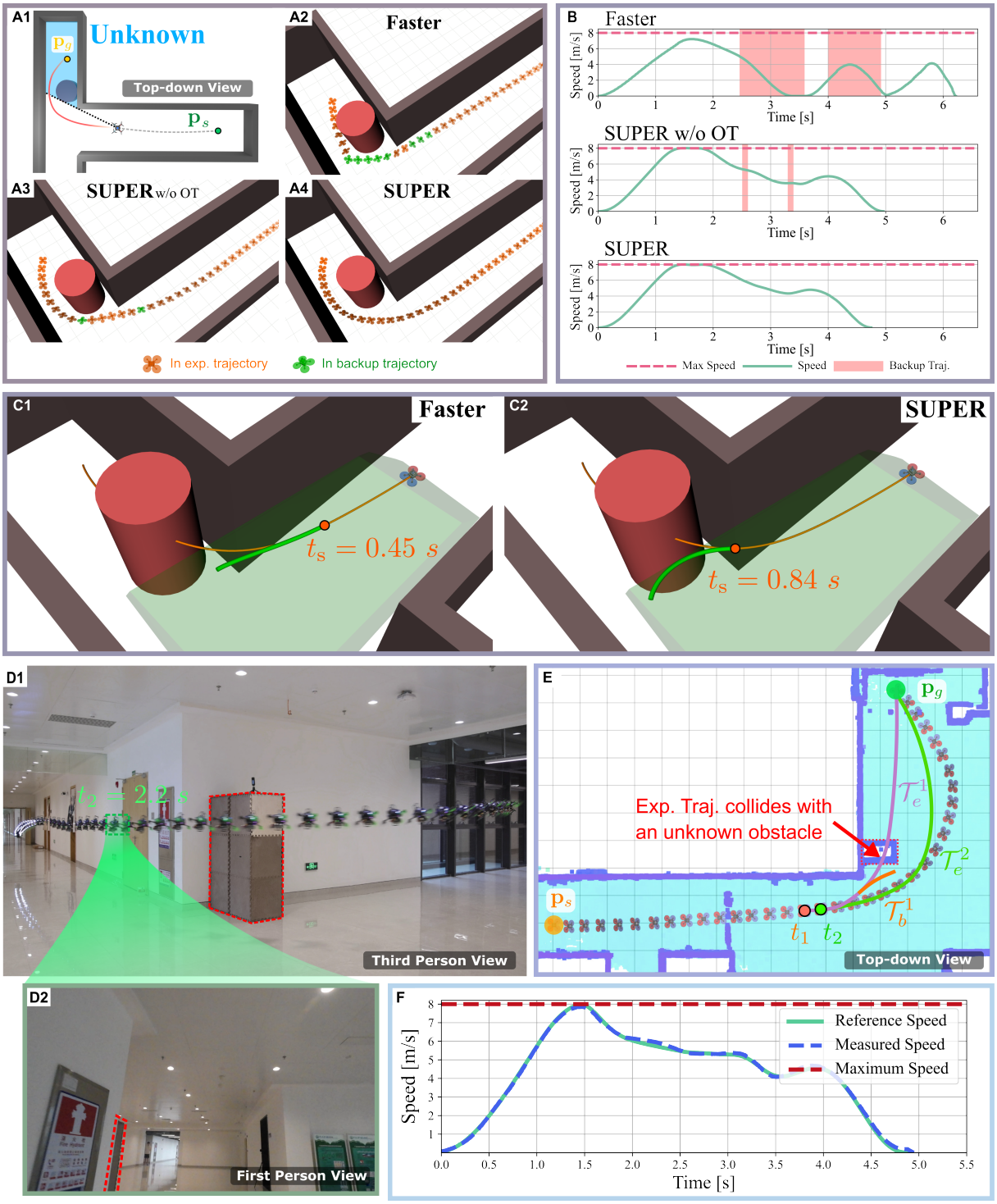

Fig. S6. Evaluation of planning module of Faster and SUPER (**A1**) A top-down view of the simulation environment, where a red cylinder is hidden behind the corner. (**A2-A4**) Comparison of the executed trajectories for the three methods: Faster (*23*), the proposed SUPER, and SUPER without switching time optimization (i.e., SUPER w/o OT). (**B**) Velocity profile and execution of backup trajectories in the simulation. (**C**) Switching time t_s of Faster and SUPER given the same exploratory trajectory and backup corridor. (**D1**) Real-world experiment of SUPER, with the hidden obstacle highlighted by a red dashed box. (**D2**) First-person view from the onboard camera at t_2. (**E**) Planned trajectories at time t_1 and t_2. $\mathcal{T}_e^1$ and $\mathcal{T}_b^1$ are the exploratory and backup trajectories at t_1, respectively, and $\mathcal{T}_e^2$ is the exploratory trajectory at t_2. The backup trajectory at t_2 is not displayed. (**F**) Velocity profile in the experiment.

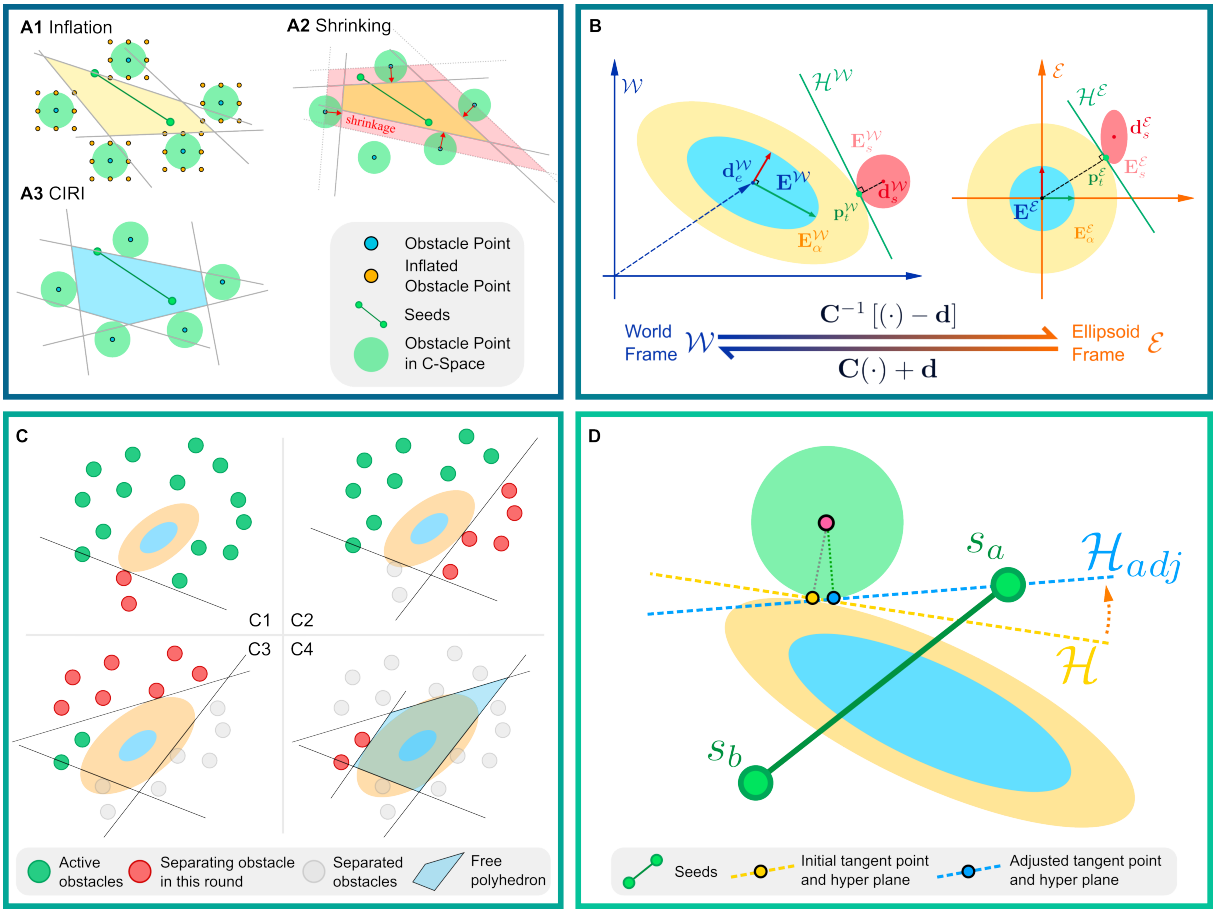

Fig. S7. Illustration of the proposed flight corridor generation algorithm (CIRI). (**A**) Comparison of three different methods for generating a safe flight corridor in configuration space. (**B**) Example illustrating the transformation between the ellipsoid frame $\mathcal{E}$ and the world frame $\mathcal{W}$. (**C**) Visual depiction of the process of convex decomposition in the C-space. (**D**) Example demonstrating the plane adjustment strategy to ensure that the given seeds lie within the generated polyhedron.

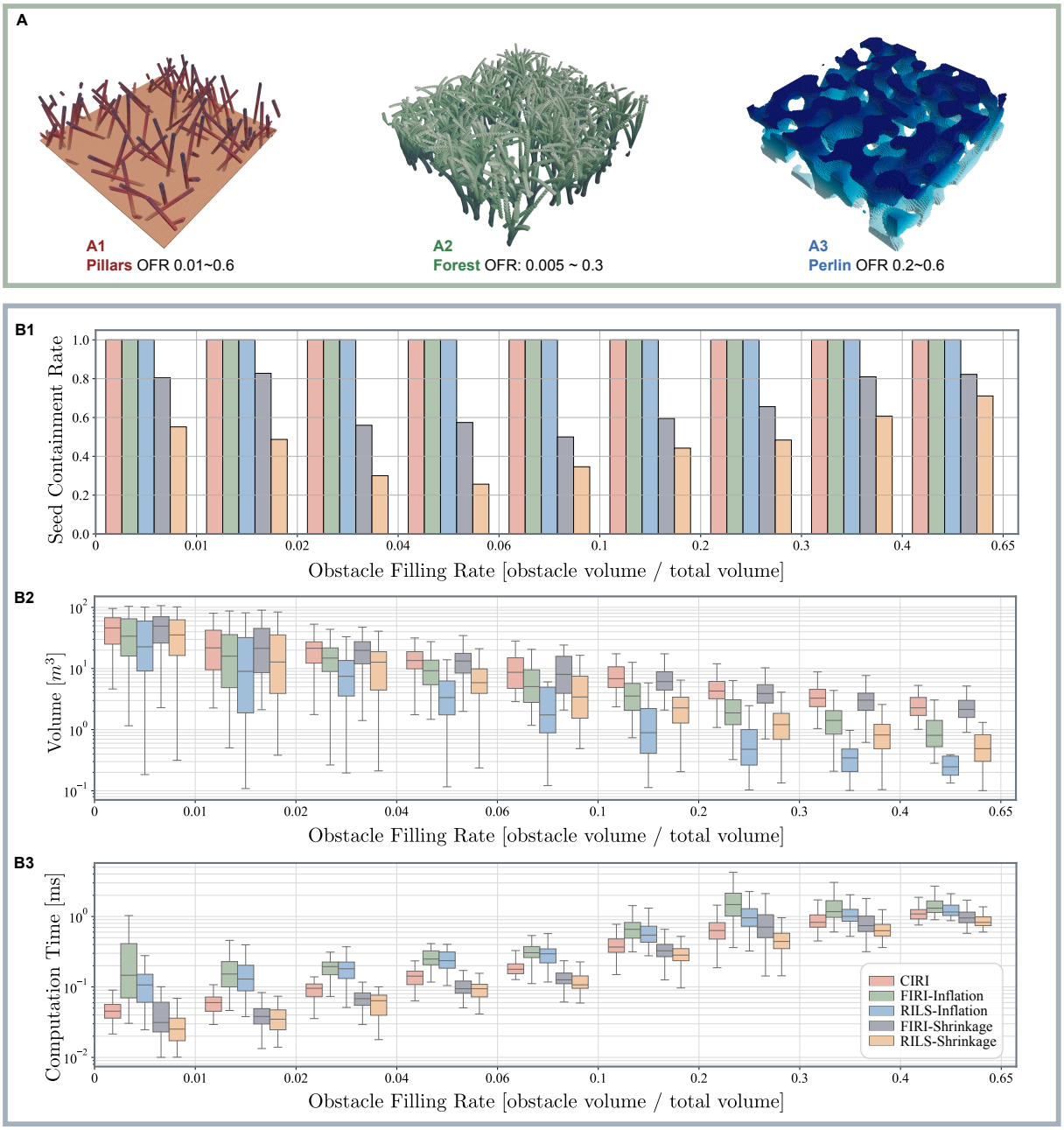

Fig. S8. Comparison of convex decomposition in configuration space. (**A1-A3**) Example benchmark environments with obstacle filling rates (OFR) ranging from 0.005 to 0.6. (**B1**) Seed containment rate of generated polyhedra by different methods across OFR. (**B2**) Average polyhedron volumes generated by different methods across OFR. (**B3**) Computation times for different methods across OFR.

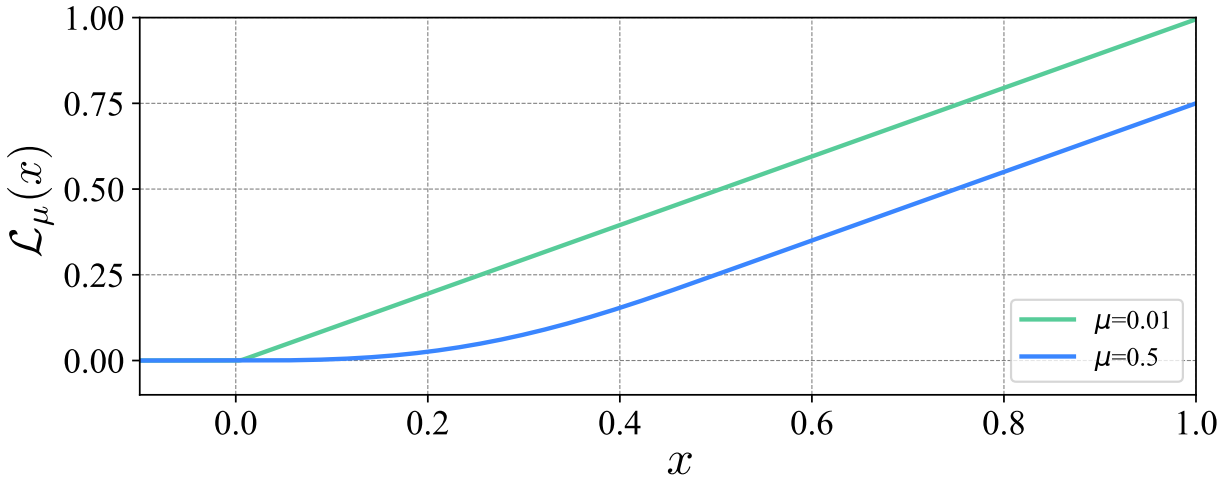

Fig. S9. The plot of the barrier function $\mathcal{L}_\mu$. The plot illustrates the function $\mathcal{L}_\mu$. As the parameter μ decreases from $\mu = 0.5$ to $\mu = 0.01$, the barrier function becomes progressively harder, with a sharper shape.

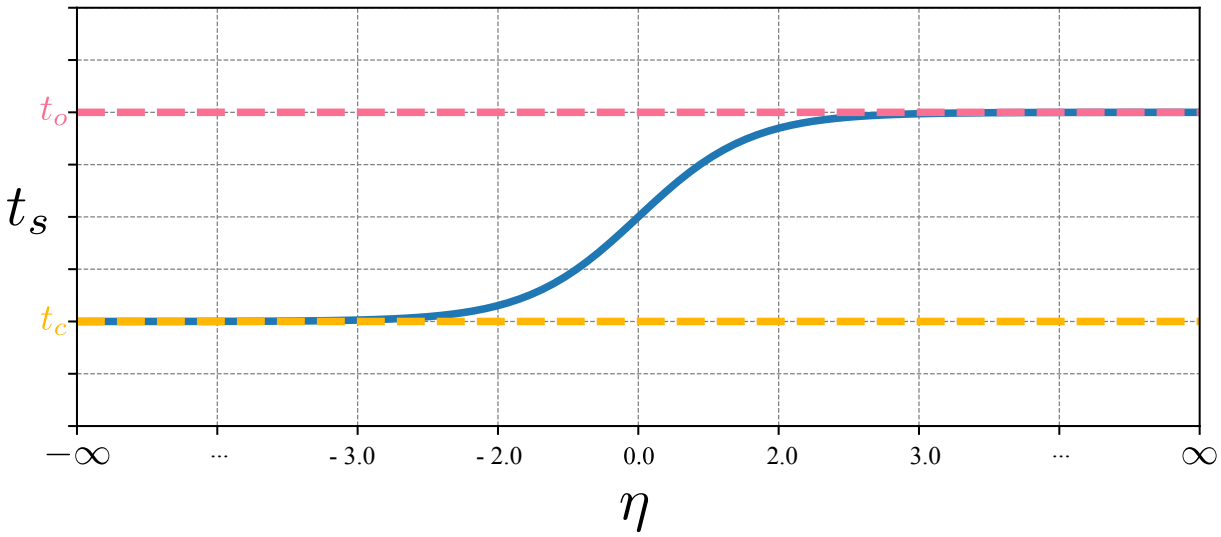

Fig. S10. The plot of the mapping from η to t_s. The plot illustrates a sigmoid-like function that maps $\eta \in \mathbb{R}$ to the interval (t_c, t_o).

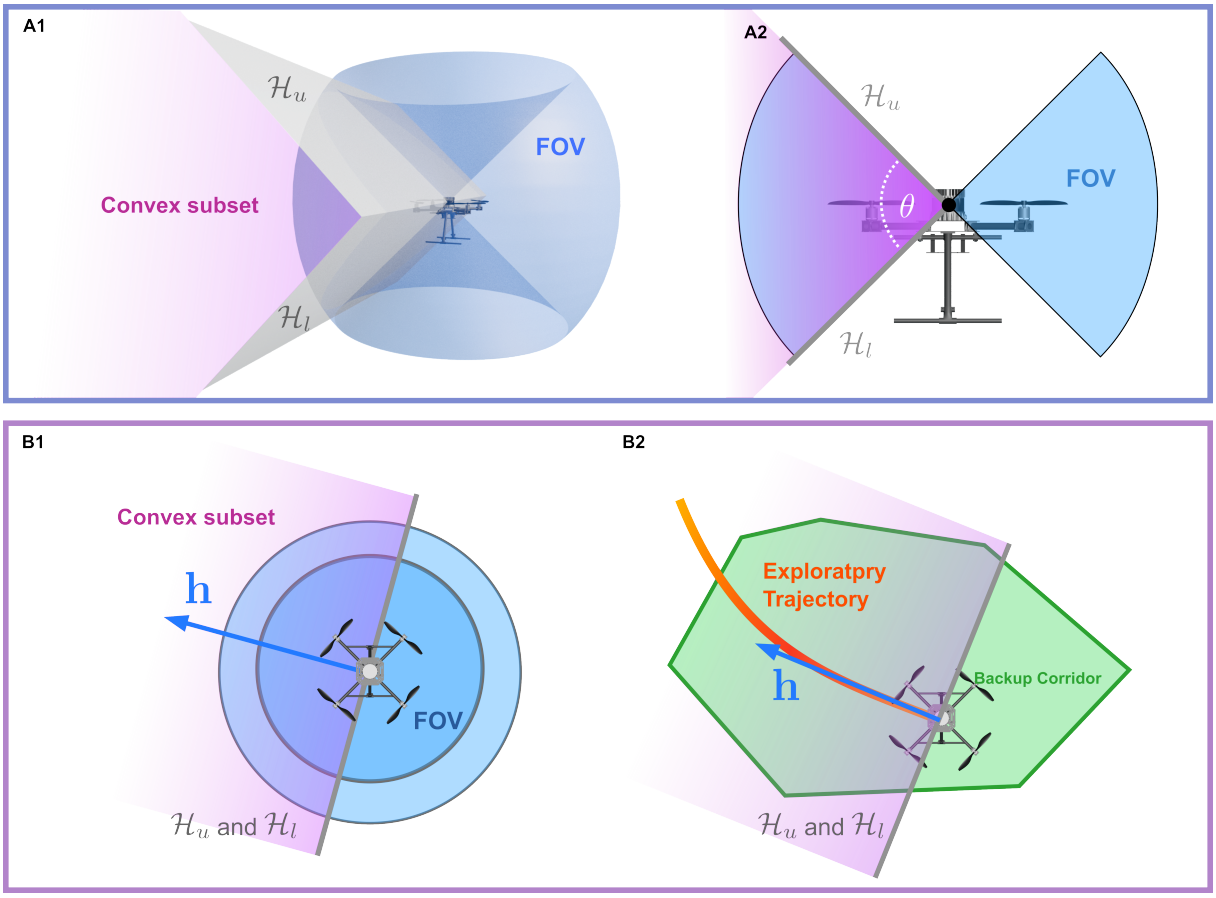

Fig. S11. Backup corridor generation of limited FOV (**A1**) The largest convex subset of a non-convex FOV is formed by the intersection of two half-spaces defined by the hyperplanes tangent to the FOV's upper boundary $\mathcal{H}_u$ and lower boundary $\mathcal{H}_l$. (**A2**) Side view of the hyperplanes $\mathcal{H}_u$ and $\mathcal{H}_l$ along with the vertical FOV angle, θ. (**B1**) Top-down view showing the convex subset and the two hyperplanes, which are uniquely determined by the heading direction $\mathbf{h}$. (**B2**) The heading direction is aligned with the exploratory trajectory to ensure that the generated backup corridor contains the initial portion of the exploratory trajectory.

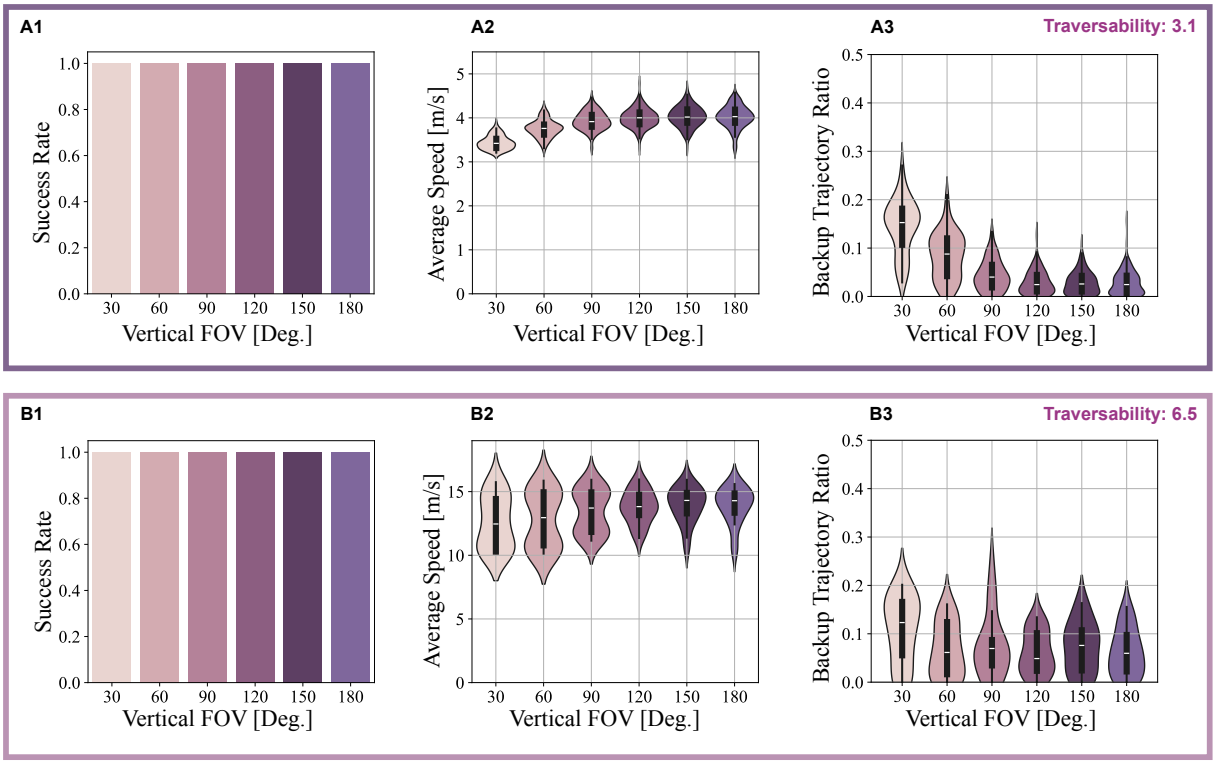

Fig. S12. **Benchmark comparison with different vertical FOV** (**A**) Benchmarking results on simulation maps with a traversability score of 3.1. (**B**) Benchmarking results on simulation maps with a traversability score of 6.5.

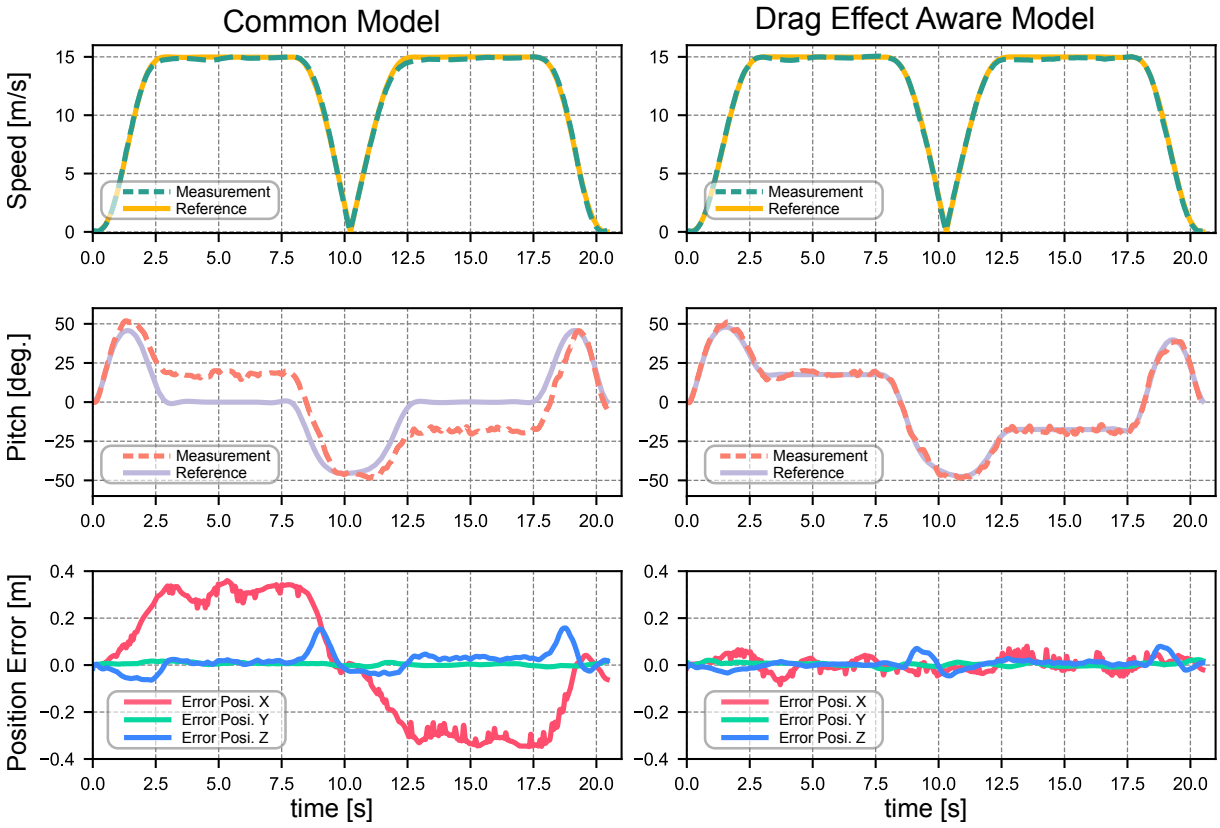

Fig. S13. Comparison of tracking performance with different drone models of MPC. The plot of speed, pitch angle, and position tracking error in each axis.

Table S1. Implementation details of different methods

	Type	Mapping	Frontend	Backend
Bubble (*13*)	Optimistic	Point Cloud + Kd-Tree	Sphere-shaped Corridor	Spatio-temporal Trajectory Optimization (STTP)
Raptor (*20*)	Safety-aware	ESDF	Topological Path Searching	Path Guided Trajectory Optimization (PGO)
Faster (*23*)	Safety-assured	OGM	Polyhedron-shaped Corridor	Mixed-Integer Qudratic Program (MIQP)
SUPER (ours)	Safety-assured	Point Cloud	Polyhedron-shaped Corridor	Spatio-temporal Trajectory Optimization (STTP)