AUTONOMOUS VEHICLES

Swarm of micro flying robots in the wild

Xin Zhou[1,2], Xiangyong Wen[1,2], Zhepei Wang[1,2], Yuman Gao[1,2], Haojia Li[3], Qianhao Wang[1,2], Tiankai Yang[1,2], Haojian Lu[1], Yanjun Cao[2], Chao Xu[1,2]*, Fei Gao[1,2]*

Aerial robots are widely deployed, but highly cluttered environments such as dense forests remain inaccessible to drones and even more so to swarms of drones. In these scenarios, previously unknown surroundings and narrow corridors combined with requirements of swarm coordination can create challenges. To enable swarm navigation in the wild, we develop miniature but fully autonomous drones with a trajectory planner that can function in a timely and accurate manner based on limited information from onboard sensors. The planning problem satisfies various task requirements including flight efficiency, obstacle avoidance, and inter-robot collision avoidance, dynamical feasibility, swarm coordination, and so on, thus realizing an extensible planner. Furthermore, the proposed planner deforms trajectory shapes and adjusts time allocation synchronously based on spatial-temporal joint optimization. A high-quality trajectory thus can be obtained after exhaustively exploiting the solution space within only a few milliseconds, even in the most constrained environment. The planner is finally integrated into the developed palm-sized swarm platform with onboard perception, localization, and control. Benchmark comparisons validate the superior performance of the planner in trajectory quality and computing time. Various real-world field experiments demonstrate the extensibility of our system. Our approach evolves aerial robotics in three aspects: capability of cluttered environment navigation, extensibility to diverse task requirements, and coordination as a swarm without external facilities.

INTRODUCTION

Science fiction films present multirobot aerial systems as a symbol of future technology. In *Prometheus* (2012), astronauts release several micro-airborne units to explore an unknown alien ship before deciding which path to take. In *Ender's Game* (2013), drone swarms surround the spaceship, forming a shield from alien attack and later clear a way for humans to win the battle. In *Star Wars: Episode III* (2005) and *Blade Runner 2049* (2017), bustling yet orderly air traffic flows among skyscrapers are common in high-technology planets. The swarms' capability of navigation and coordination in these films has attracted and inspired numerous researchers. Here, we take a step forward such a future (Movie 1).

With recent developments in computation, sensing, and communication, aerial robots such as quadrotors have entered human life with extraordinary versatility, ranging from accurate to aggressive missions (*1*) and at a low price. Nikkei reports (*2*) that DJI's Mavic Air 2, one of the best-selling drones, which has obstacle avoidance, tracking, and a 10-km communication distance, is made of components only worth approximately $135. In addition, there are still an abundance of possibilities for drones in the market, with the value forecast to be $500 billion by 2028 (*3*).

For single-drone navigation, agile multicopter control systems are well developed (*4*). Furthermore, accurate localization using visual-inertial odometry (*5*, *6*) has matured, which has built up a reliable and efficient perception system with probabilistic mapping (*7*). For collision avoidance and other safety requirements, various methods from reaction-based, as seen in (*8*), to planning-based (*9*, *10*) are proposed. Such developments herald the dawn of aerial swarms in the wild that previously have only been imagined in science fiction.

Although single-drone autonomous navigation has been developed aggressively for both industrial (*11*, *12*) and academic practices (*13*, *14*), very rarely has comparable performance has been achieved by aerial swarm systems. Building on the development of individual drones with autonomy, here, we address the fundamental problems of how to navigate aerial swarms in cluttered wild environments autonomously, thus improving the applicability of swarms in a variety of real-world tasks. These tasks include the following: (i) providing disaster relief. In natural disasters like earthquakes and floods, a swarm of drones can search, guide, and deliver emergency supplies to trapped people (*15*). For example, in wildfires (*16*), agile multicopters can quickly collect information from a close view of the front line without the risk of human injury. (ii) Aiding biological studies (*17*). Thanks to reduced drone size and weight, researchers can inspect a confined area without direct human contact to minimize ecological footprints. (iii) Dense air traffic system or ready-to-deploy transportation for rovers and drones landing on Mars. In these scenarios, transporters randomly fly between dense buildings. Therefore, both external and inter-vehicle collision avoidance is crucial (*18*). (iv) Collaborative transportation. When the payload weight exceeds the capacity of a single drone, multidrone formation flight is required (*19*).

Common requirements of the above tasks can be categorized into four aspects abbreviated as TEEM (trajectory optimality, extensibility, economical computing, and miniature size). Trajectory optimality indicates the mission quality and flight time. An optimality-concerned trajectory planner will not stop at a feasible solution but continues to try to find a nearly optimal one among all solutions. Such efficiency is especially crucial in emergency and rescue scenarios, in which time is of essence. Extensibility refers to the diversity of available applications, so the system must be compatible with assorted task-specific objectives. Economical computing is important as well, which allows smaller embedded onboard computers, reduces the reaction time to environmental change and sudden events (*20*), and reserves as much available computing resource as possible for other user-defined tasks such as object detection and decision-making.

[1]State Key Laboratory of Industrial Control and Technology, Zhejiang University, Hangzhou, China. [2]Huzhou Institute of Zhejiang University, Huzhou, China. [3]Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hongkong, China.
*Corresponding author. Email: fgaoaa@zju.edu.cn (F.G.); cxu@zju.edu.cn (C.X.)

Movie 1. A comprehensive presentation of the proposed swarm. This video cover is a composite image of the aerial swarm in a bamboo forest.

Last, all such capabilities should be placed into the smallest container because weight and volume are directly related to the flight time and acceptable narrow space.

Unfortunately, achieving these four aspects together is internally contradictory. Higher optimality mostly comes from sophisticated modeling and more iterations or trials in the solution space, all of which are achieved at the cost of increased computing time. Higher extensibility requires the problem to be defined in a more general form at the sacrifice of potential problem-specific optimization that can improve optimality and reduce computing time. Then, between optimality and extensibility, as various user-defined objectives are imposed, the problem becomes increasingly complex, which makes it challenging to find a solution. Only satisfying some basic requirements such as safety and feasibility while minimizing time and maximizing smoothness for aerial swarms is already a difficult problem (*21*, *22*, *23*) and is even more difficult to achieve simultaneously on a miniature platform. That is why previous studies are unable to take the step from structured, human-made environments to the unforeseen wild.

In the real world, various aerial swarms have been deployed, including enormous impressive drone light shows presented by Intel (*24*), High Great (*25*), and CollMot (*26*). Nevertheless, behind large-scale and successful commercial usages, the swarm positioned by the Global Navigation Satellite System merely follows preprogrammed trajectories and therefore cannot be operated in unforeseen places with obstacles. Autonomous outdoor aerial flocking was presented in (*27*–*29*), where drones adjusted their motions according to others' states in real-time using simple reactive rules such as the potential field (PF) method. However, the lack of optimality consideration during the flight resulted in actions that are not sufficiently coherent. Such inconsistency further renders far-neighbor distance [>10 m on average reported in (*29*)] required for safety and thus unsuitability in cluttered environments. Moreover, although they successfully imitate the behaviors of large flocks of birds, accurately operating each individual is difficult because the parameters are tightly coupled with specific deployment scenarios and neighbor states. To achieve rapid and safe collective motions with dense obstacles, Soria *et al.* (*30*) incorporated model predictive control into the PF. Nevertheless, higher performance was achieved at the cost of heavy computation and a centralized organization that lacked the scalability for a large swarm size (*30*). In addition to navigation algorithms, the above solutions (*27*–*30*) still depend on global localization and a known environment, which prevent their applications in the wild. In contrast, McGuire *et al.* (*8*) mitigated these two requirements by proposing a reactive swarm gradient bug algorithm (SGBA) that used optical flow for localization and laser ranging sensors for obstacle detection. The SGBA is very lightweight while incorporating all the sensing, decision-making, and control into a 30-g CrazyFlie (*31*) but that attribute comes at the cost of low efficiency and extensibility and is therefore more suitable for structured places and simple tasks (spreading from and returning to home). The inaccurate localization and single-point obstacle sensing restricted by the hardware platform further aggravate such limitations. A qualitative comparison of (*29*, *8*, *30*) is presented in Fig. 1B. Accurate self-positioning with a single camera and inertial measurement unit (IMU) on a miniature swarm platform was first released by Loianno *et al.* (*32*), but only some trajectory planning in sparse known environments was demonstrated, and the swarm did not enter the wild. Furthermore, purely visual-inertial odometery (VIO)–based localization in their work may drift in long-range flights. To make a swarm more efficient and robust in dense environments, our previous work, EGO-Swarm (*33*, *34*), proposed an optimization-based method. Full-stack navigation solution of aerial swarms deployed in the forest is rare. One limitation of our previous work is that the potential for scalability lacks solid validation because we used only three drones. Moreover, the planner is unable to adjust the time profile, which would occasionally produce less optimal and even unsafe trajectories. These imperfections still leave the aerial swarm in the cluttered wild as an unsolved problem.

Observing how nature tackles this navigation challenge, two mainstream approaches have inspired robotics researchers. Insects perform short-term reactive actions, whereas birds prefer relatively long-term smooth maneuvers (*35*). This is because birds have a sharper sense of sight and movement, higher–degree-of-freedom motion systems, and more brain capacity compared with insects (*35*, *36*). These two approaches have also inspired two mainstream drone navigation methods in the literature: insects for reaction-based applications and birds for trajectory planning approaches. Among the two, the former approach contains extremely lightweight and efficient solutions in terms of computation and memory allowing for even lighter drones, whereas the latter shows higher optimality and flexibility. Accordingly, for higher task efficiency and extensibility in field environments, we choose the latter approach. Here, we address this TEEM contradiction by incorporating spatial-temporal optimization techniques for trajectory optimality and formulating trajectory planning as a multiobjective optimization under a goal-chasing framework for extensibility. Furthermore, the combination of the above two features renders fast convergence, therefore guaranteeing economical computing, which makes a miniature platform possible.

Proposed systematic solution

After investigating a variety of applications, we find that the key to TEEM is trajectory planning, which not only deforms trajectory shapes but also adjusts the time profiles to exhaustively exploit the solution space and squeeze the capability of drones. If only spatial deformation is performed (*33*, *37*), as compared in the "Benchmark comparisons" section, drones tend to circumnavigate to wait for others while passing through a narrow passage, which hinders later drones and results in inferior and even unsafe trajectories. Therefore,

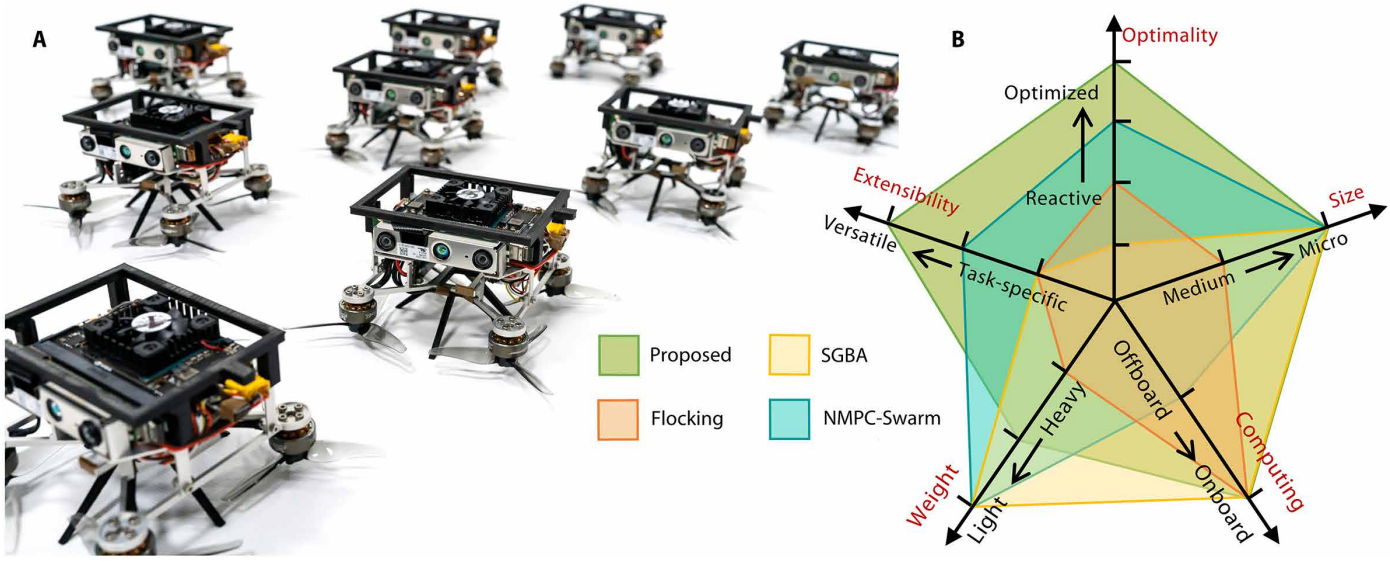


Fig. 1. Overview of the proposed aerial swarm. (**A**) Static closeup. (**B**) Comparison with swarm gradient bug algorithm (SGBA) (*8*), flocking (*29*), and nonlinear model predictive control (NMPC)–Swarm (*30*). The ticks of each axis from the graph center to the outward are as follows: Optimality: handcrafted rules, optimized rules, spatial optimization, and spatial-temporal optimization; size: arm-sized and palm-sized; computing: offboard and onboard; weight: below 100 g, above 100 g, and above 1 kg; extensibility: task specific, tasks with specific formulations, and tasks that can be analytically modeled of decision variables.

simultaneously planning the shape and time profile of a trajectory, also called spatial-temporal trajectory planning, is crucial for safe and efficient drone flights. Despite this, such joint optimization has a historically difficult problem for multicopters, because the spatial and temporal parameters determining the trajectory together are highly coupled (*38*, *39*), which, for example, results in ~40 min to compute a time-optimal trajectory (*1*). In the proposed approach, we achieve real-time spatial-temporal optimization by decoupling the spatial and temporal parameters in objective function computation and achieving a linear complexity mapping between the optimized variables and intermediate variables that represent a trajectory.

Under the trajectory-planning framework, the task-specific requirements of generating a trajectory can always be formulated as goals to reach; multiple objectives, such as shorter flight time, higher smoothness, and closeness to a given path; and constraints, such as collision avoidance and dynamical feasibility. For the first requirement, we build our planner under the goal-chasing scheme, which receives users' goals continuously and keeps chasing the latest one. For the second and third requirements, the nonconvexity among them makes the optimization problem difficult to solve. To achieve high compatibility, we adopt the constraint transcription method (*40*) that converts all objectives and constraints to weighted penalties. Specifically, penalties derived from constraints are assigned with weights orders of magnitude higher than other objectives. Note that here, terms "objectives" and "constraints" refer to task requirements, while "penalties" are their relative mathematical formulations forming the final cost function. The trajectory-planning problem can then be solved quickly by standard solvers leveraging sparse parametric optimization and constraint transcription. To simplify the situation, we provide detailed examples of adding task-specific objectives and constraints intuitively with preformulated general-purpose penalties (GPPs). GPPs consist of time minimization, smoothness maximization, collision avoidance, and dynamical feasibility, which are defined in Materials and Methods. This trajectory planning framework is illustrated in Fig. 2D.

Except for the proposed trajectory planning, we adopt visual-inertial odometry running on each drone independently for aerial swarm localization. However, accumulative odometry drift may result in drone collisions when they continue to report, maintaining a safe distance, so we develop a decentralized drift-correction algorithm by minimizing relative distance error measured from onboard ultra-wideband (UWB) sensors.

As shown in Fig. 2 (A and B), each drone is equipped with full perception, localization, planning, and control functionalities and loosely coupled by a broadcast network sharing trajectories. Coincidentally but reasonably, the proposed system is similar to birds capable of flying freely through the forest while avoiding obstacles and other moving creatures. For example, in short-range navigation, birds mainly rely on eyes and their vestibular system (*41*), and we, accordingly, develop improved visual-inertial odometry. Furthermore, birds adjust path and speed simultaneously to avoid collision while considering flight time and smoothness to save energy (*35*), and we thus propose joint optimization of spatial-temporal trajectories with multiple objectives. Beyond the capability of small birds, we further use the advantage of our electrically powered artificial system characterized by high-fidelity wireless communication for trajectory sharing and high-speed computing for fast planning. Furthermore, decentralized coordination concerning both individual and swarm intelligence is met naturally by our solution, which improves robustness. As Murphy (*42*) pointed out, weakly centralized, distributed organization of the swarm shows higher robustness and resilience and can even retain actions when communication and Global Positioning System (GPS) data are lost.

We propose a versatile multirobot navigation solution, allowing users to incorporate various task-specific requirements and also producing locally spatial-temporal optimal motions in real time. The proposed solution is embodied on drones that are only the size of a

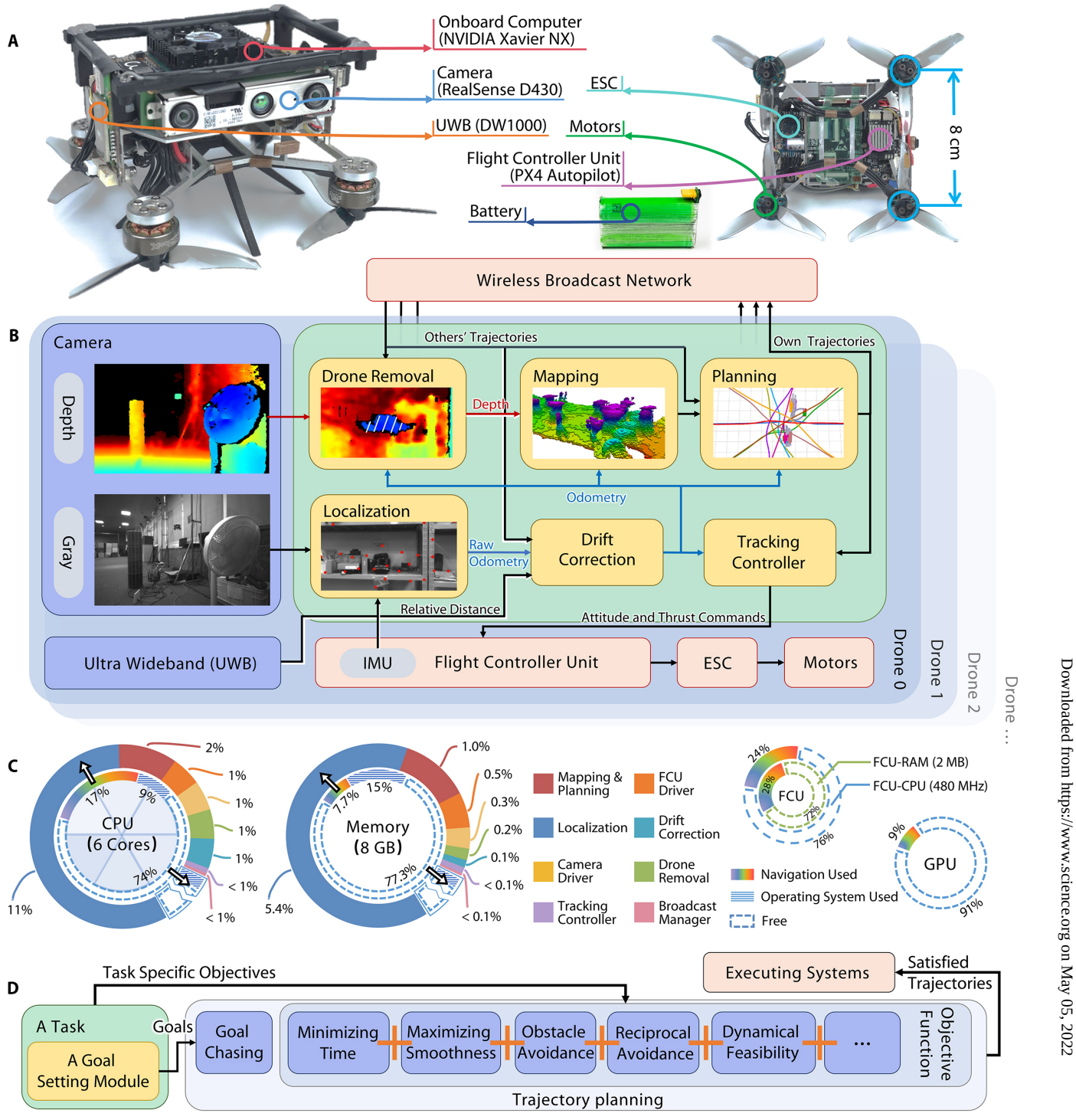


Fig. 2. Hardware and system architecture specifics. (**A**) Hardware components of our flight platform. See the "Palm-sized drone hardware" section for more details. (**B**) The system architecture. Visual-Inertial State Estimator (*5*) and probabilistic occupancy grid (*7*) are adopted for localization and mapping, respectively. (**C**) Computation and memory usage. Planning and mapping run in the same thread to reduce latency. (**D**) The planning framework.

palm and is validated by several demonstrations in unstructured environments as detailed in Results. We have released the software and hardware necessary to accelerate aerial swarm research, which developers can deploy and use to validate their algorithm from simulations and field environments, all based on our platform.

RESULTS

The quality of the proposed planner is assessed in both real flights and simulation. In real-world experiments, we present four proof-of-concept but challenging applications, each of which is modified on the basis of our system solution to validate different aspects of

performance and potential. In simulations, quantitative evaluations of several common metrics are conducted with various state-of-the-art aerial swarm planners.

Fly through dense forest

This experiment is designed to demonstrate swarm navigation with full autonomy in a highly dense wild, i.e., a bamboo forest, without harming themselves or the plants. In this experiment, the penalty function only contains GPPs, and the goal is set 65 m forward, outside the forest. The paths in Fig. 3I reveal a notable advantage of trajectory planning: The planned trajectories always connect one gap to the next one directly and smoothly. Compared with reaction-based methods, the drones always show an explicit and nonsmooth turn-away pattern (*8*) directly in front of an obstacle or before hitting other drones.

From the photos, available space between two bamboos may be less than 30 cm wide, and therefore, only miniature-sized drones are allowed to pass. More severely, these narrow gaps further limit the solution space, especially for drones that have neighbors on both the left- and right-hand sides. The constraints become even tighter when only one available gap exists for multiple drones to pass through together. To achieve safety and efficiency, some naive handcrafted strategies, such as altering altitude to avoid collisions, are undesired because of downwash disturbance and energy wasting. Under such situations, our spatial-temporal trajectory planner

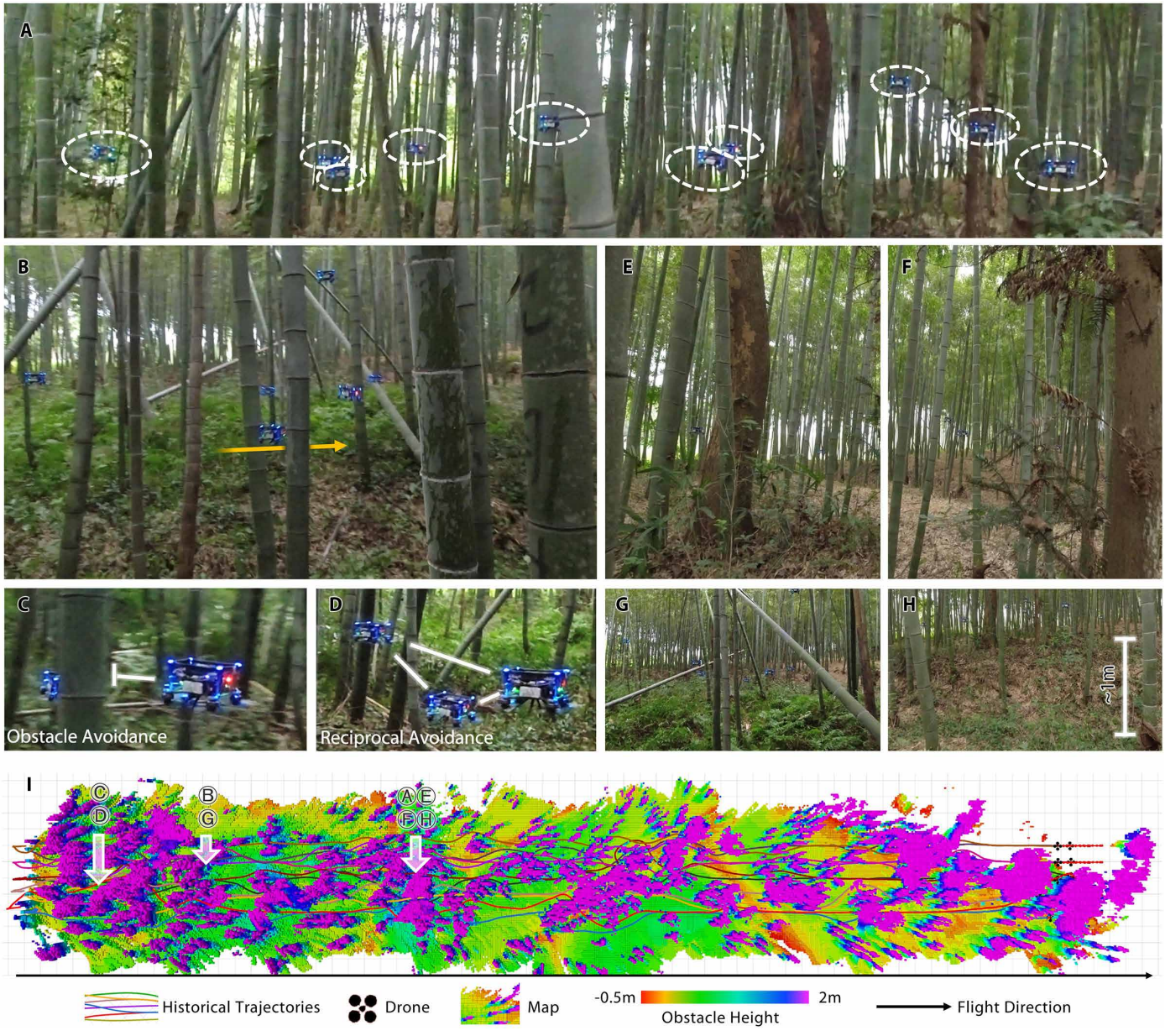


Fig. 3. Challenging wild navigation with bamboos and various other obstacles. (**A**) Ten drones fly through highly dense bamboos. (**B**) A drone flies through a narrow gap. (**C** and **D**) Collision avoidance. (**E**) A trunk. (**F**) Messy branches. (**G**) Tilted bamboos. (**H**) Uneven ground. (**I**) Trajectories and the combined map recorded from each drone. Circled letters with white arrows indicate the places in above panels. Experimental visualization in this paper shares a legend identical to that here.

implicitly finds solutions in a general problem formulation by adjusting time profiles to allow multiple drones to only change necessary velocity smoothly and then pass the gap in a queue, which produces lower cost from the planner's perspective compared handcrafted rules. A quantitative analysis of this procedure can be found in section S4 of the Supplementary Materials.

Except for dense vertically growing bamboos, other kinds of obstacles exist, including tilted bamboos, trunks, low bushes, weedy ditches, uneven ground, and blown leaves blocking cameras (Fig. 3, A to H), which necessitate planning the trajectory in three dimensions. Such an unstructured environment composed of obstacles with irregular shapes and dense distributions validates the capability of navigating in most cluttered places, such as disaster scenarios, let alone in the regular artificial world. The video of this experiment is Movie 2.

Formation navigation in the wild

This experiment demonstrates the extensibility of the proposed unified trajectory planning by adding a formation penalty to GPPs. Formation flight is widely used in drone light shows and has been demonstrated in cooperative transportation (*43*, *32*), but all these demonstrations are presented in empty or manually controlled environments. The proposed system brings the formation into a previously unknown wild environment. Here, the formation is defined as maintaining a preferred moving shape, which means that drones translate with fixed relative positions. Meanwhile, each drone also independently navigates against obstacles. The formation penalty is a regulation term to desired positions that maintain the formation. The obstacle density is reduced in this experiment compared with that in the "Fly through dense forest" section to make the formation distinguishable, but standing bushes, low-to-high trees, and two human-made iron pillars still exist as shown in Movie 3.

Following the planned trajectories, the swarm flies through the woods while staying in formation. From the deformation curve and velocity profile in Fig. 4 (A and D), we conclude that the swarm maintains a formation, although sometimes drones must deviate to avoid prior unknown obstacles and then gather speed to catch up to the formation. Note that at timestamps 12 and 25 s, the average velocity decreases automatically as drones are avoiding trees and increases when they are wholly back to open space. In this case, the velocity altering of some individuals is propagated to the entire formation without explicit preprogramming. This phenomenon shows the implicit balance between safety and flight time where the slowdown near obstacles reserves more reaction time to potential collision and the whenever-possible speedup reduces flight time. The video of this experiment is Movie 3.

Intensive reciprocal avoidance evaluation

Unlike other tests, in which drones move along a similar direction and the actions of avoiding others are not apparent, the experiment shown in Fig. 5 demonstrates random-direction flight in a confined space, therefore maximizing the necessity of inter-robot collision avoidance. This setting mimics the most underlying requirements for dense air traffic among skyscrapers: navigating safely, efficiently, and individually. To validate such capability for 10 drones, goals on a 3-m-radius circle are assigned randomly to drones that have arrived at the previous goals. Only the basic GPPs in trajectory planning are adopted. Except for a thick tree trunk and a camera-mounted tripod in the flight area, to better imitate more real-world situations during the flight, we gradually place cuboid and cylindrical obstacles to mimic newly built buildings, walk through the area as large moving obstacles, and hold and move a drone as natural disturbances. Next, we shut down all the ground localization anchors (only used in this experiment) to imitate the loss of global positioning.

Because safety and efficiency are two main concerns of transportation systems, we evaluate the minimum distance to collision and the total number of completed deliveries (total reached goals) during the 3-min flight. As shown in Fig. 5D, during the entire flight, each drone is modeled as a sphere with 7-cm radius. They manage to keep safe distances from both obstacles and other drones, despite unpredictable events. The number of reached goals increases linearly as time increases. Thus, the near-constant transporting rate under different obstacle densities is achieved owing to the local optimality of the planned trajectories. The video of this experiment is Movie 4.

Multidrone tracking with target occlusion

This experiment demonstrates the potential of adding high data-load hardware and running more computationally expensive software on the proposed miniature platform with extra user-defined objectives. Swarm tracking can be used in multiview aerial photography and videoing, which have been attracting interest recently (*44*, *45*), and they can take comprehensive recordings of the participant and provide more materials for post-editing. To track and record, drones are equipped with extra RGB (red-green-blue) cameras that not only capture vivid videos but also act as representative "high data-load" hardware and simultaneously run video compression, data storing, and object detection to validate the platform's extensibility in computationally expensive tasks. In our experiment, the focus is a human participant moving in the woods. To catch up with the human while avoiding obstacles and other drones, we design a tracking penalty along with GPPs to plan the desired trajectory.

Movie 2. Flying through dense forest.

Movie 3. Formation navigation in the wild.

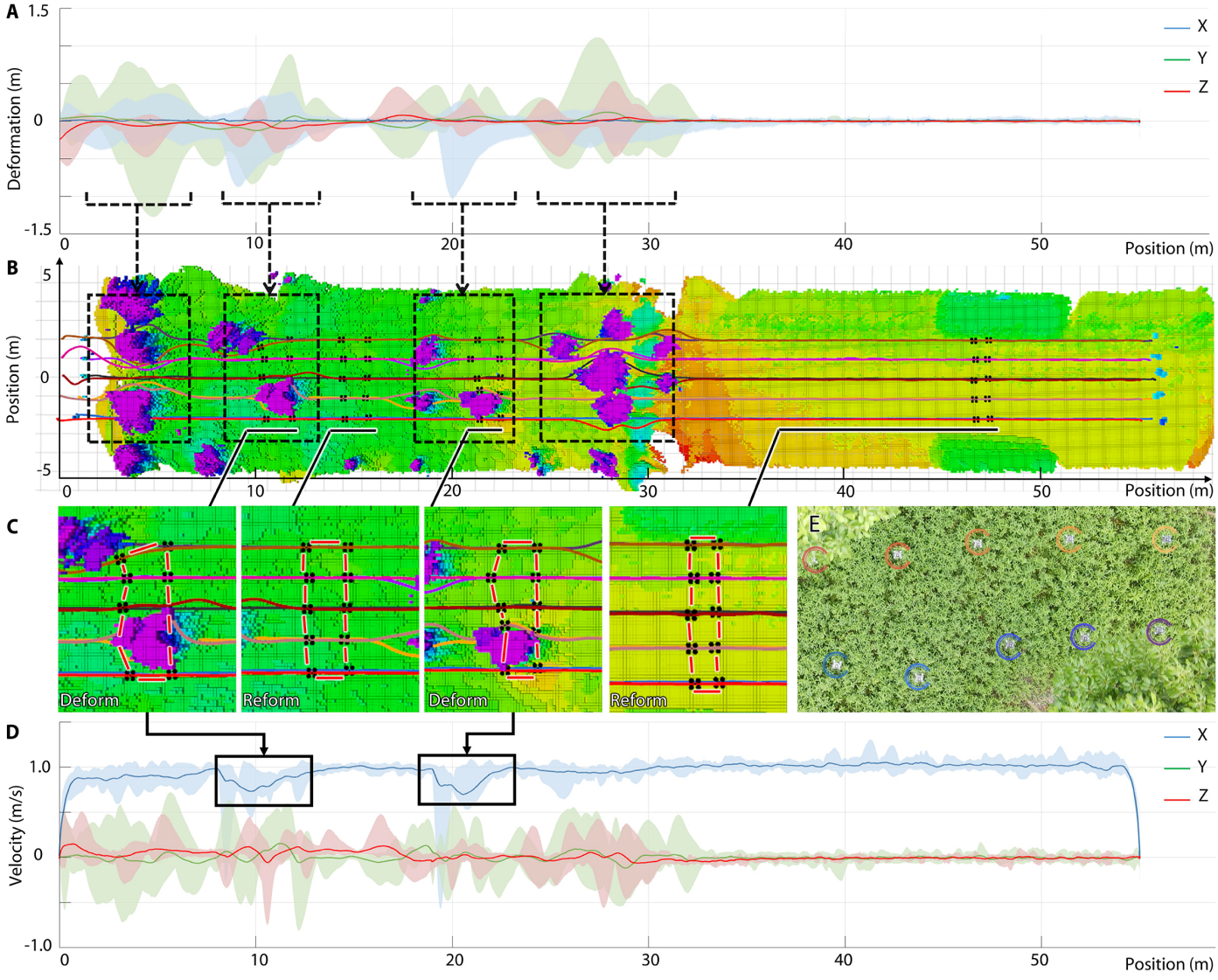


Fig. 4. Swarm navigation in formation with prior-unknown obstacles. (**A**) Deformation is defined as the deviation between current drone position and the assigned position that forms the desired formation. (**B**) Maps, trajectories, and drones recorded during the flight. (**C**) Snapshots of adaptive deformation and reformation to pass through the given area. (**D**) The velocity profile. In (A) and (D), solid curves are average value of all drones, while the top and bottom bounds of the transparent parts indicate its maximum and minimum values. Note that (A) to (D) are aligned in this position to provide clearer evaluation. (**E**) A photograph of the formation of real drones, which are about 1 m apart.

Furthermore, multidrone tracking improves the robustness to occlusion as shown in Fig. 6D because the object position can be acquired by multiple drones through communication. Drones will try to receive and fuse as many observations as possible from itself and others to improve occlusion resistance. From the results in Fig. 6, the human can move forward without worrying about drone collision or losing the target. The video of this experiment is Movie 5.

Benchmark comparisons

We compare the proposed approach against two state-of-the-art planners, i.e., MADER (*37*), and our previous work, EGO-Swarm (*33*). Both planners belong to the category of decentralized, asynchronous trajectory planning methods for drone swarms. Here, MADER shows impressive collision avoidance with both densely placed static and dynamic obstacles. Besides, EGO-Swarm validated in the wild is a lower-complexity systematic solution. The simulation platform is a personal computer with an Intel Core i7 10700K CPU (central processing unit) running at 4.8 GHz and with 24-GB RAM (random access memory) at 3200 MHz, on which drones run in independent threads in parallel to maintain consistency with the real-world, decentralized system architecture.

Figure 7A gives the visualization of planned trajectories in two challenging scenarios, i.e., flying through a narrow gate and an obstacle-rich area at a velocity of 2 m/s. The negative effect of lacking temporal trajectory optimization reveals that both MADER and EGO-Swarm generate detours for later drones to wait for their priors. The proposed planner shows the most smoothest trajectories because drones can adjust time profiles to achieve spatial collision avoidance.

We assess the trajectory-planning performance considering four different metrics under various desired velocities and average

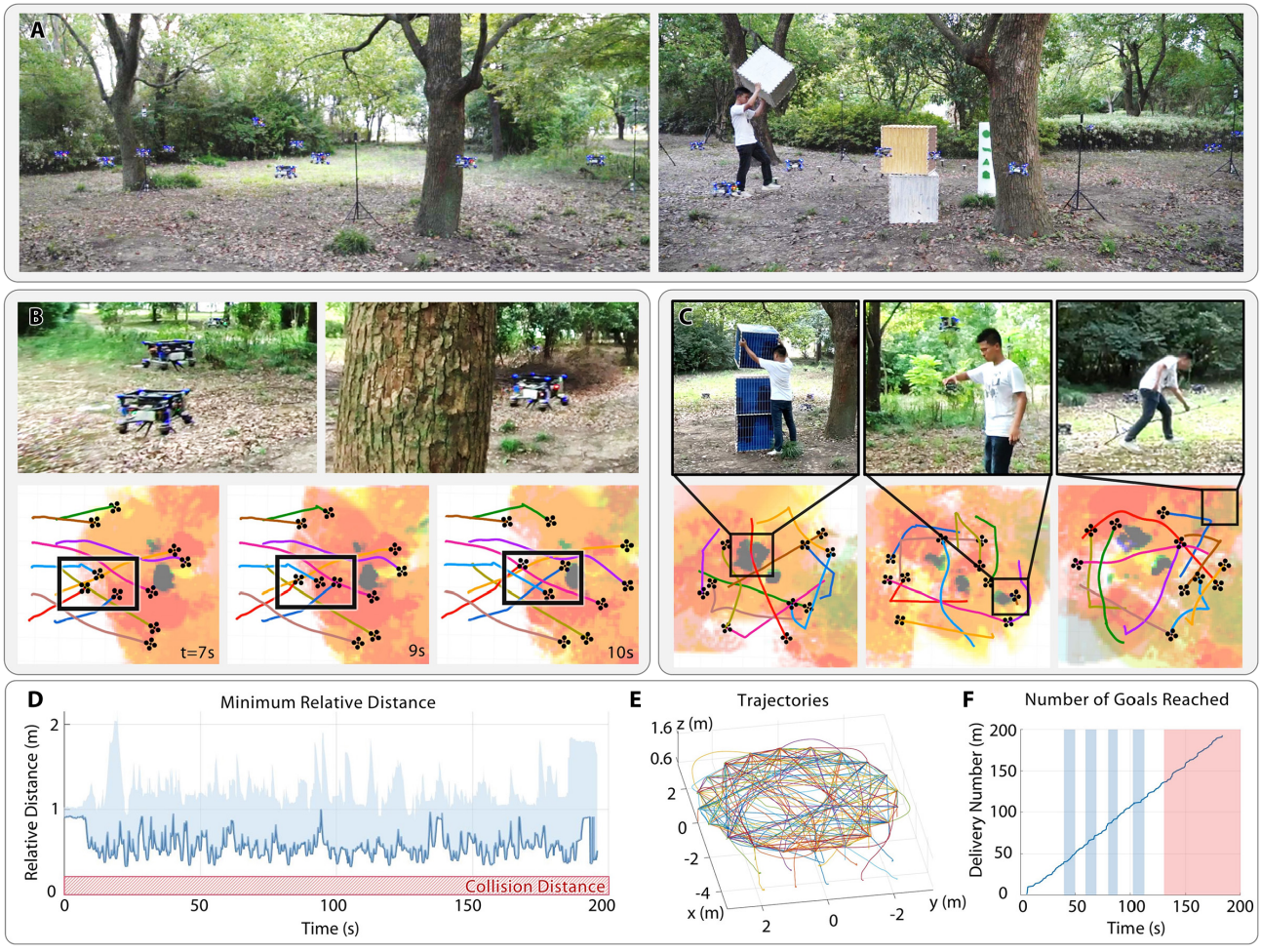
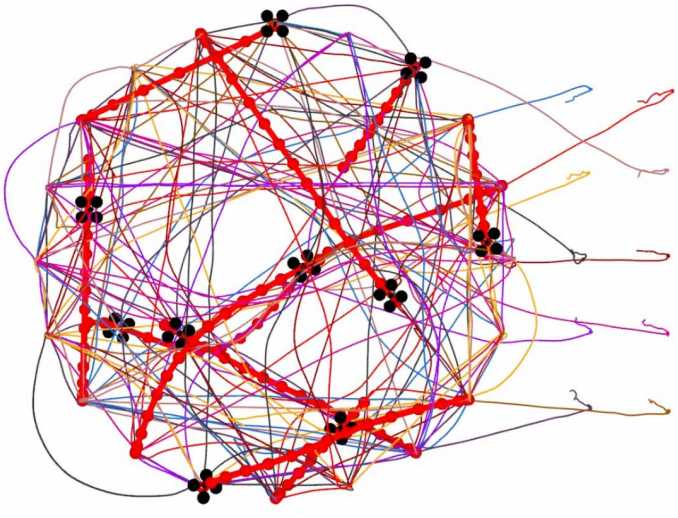


Fig. 5. Evaluation of intensive reciprocal collision avoidance with unexpected events. (**A**) Overview of flying to randomly given goals and adding obstacles during the flight. (**B**) Top pictures show drones avoiding other drones and obstacles. Bottom pictures show a sequence of snapshots recording inter-robot collision avoidance. (**C**) Photos from left to right show a researcher adding obstacles, interfering with a drone, and shutting down global localization anchors during an entire flight. (**D**) Minimum relative distance of each drone to others. Solid line records the lower bound of the minimum that is the closest distance among all agents. Collision distance equals the drone diameters. Safety is guaranteed during the entire flight. (**E**) The trajectories during the 3-min flight. (**F**) Number of goals reached. Blue bars indicate the time slots when new obstacles are added; the red bar means that global localization anchors are shutdown. Note that (A) and (B) to (F) belong to two different flights.



Movie 4. Intensive reciprocal avoidance evaluation.

obstacle spacings in Fig. 7B. Specifically, three approaches are compared with desired velocities varying from 1 to 3 m/s and average obstacle spacings ranging from 2 to 1 m. Among these four metrics, minimum clearance is the minimum value among all drones during their entire flights. The others are the average values for 10 runs under each different parameter setting. The maps remain identical for three planners but are randomly regenerated among 10 runs.

In Fig. 7, our trajectories show fewer jerks at comparable flight times while guaranteeing safety, owing to the spatial-temporal trajectory planning capability. Because MADER uses the MINVO (*46*) basis to parameterize the trajectory and EGO-Swarm uses B-Splines (*47*), the temporal optimizations all suffer from high complexity and are therefore intractable under real-time requirements. Thus, the solution space is not fully exploited, which results in inferior trajectory quality. Furthermore, the proposed method achieves shorter a computing time because our planner has fewer decision variables as described in the "Trajectory representation" section. In particular, MADER optimizes both the trajectories and separating

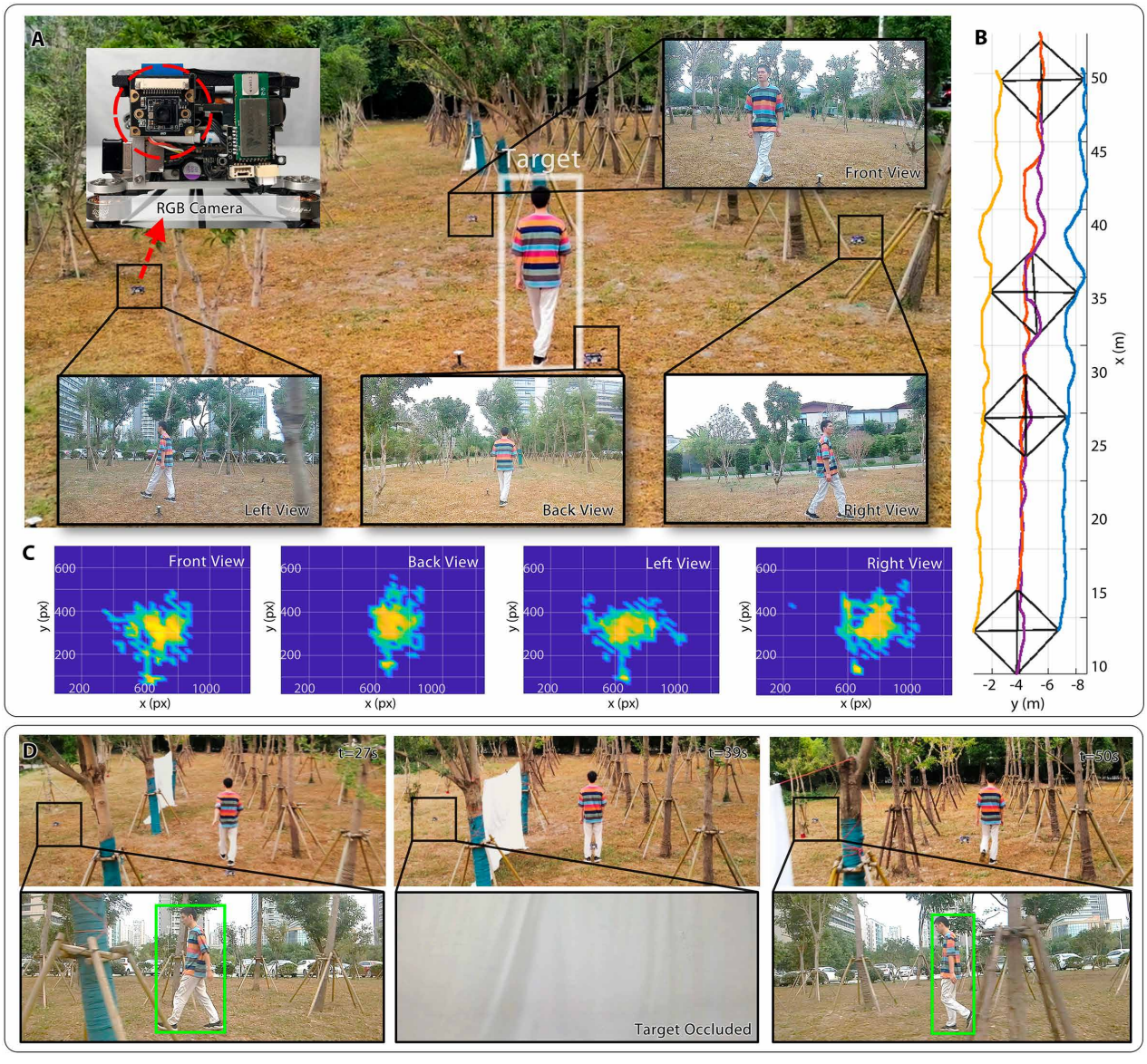
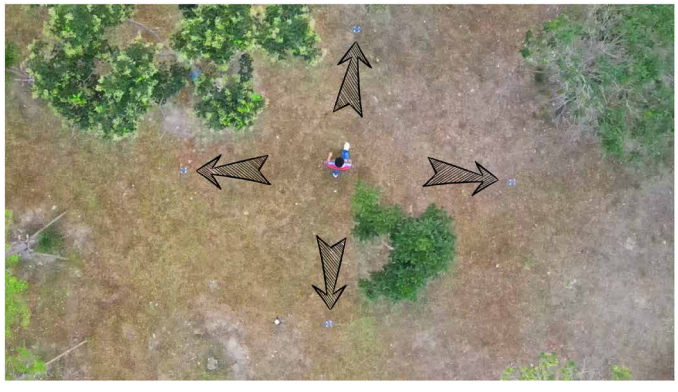


Fig. 6. Multidrone tracking with target occlusion. (**A**) Tracking using four drones equipped with RGB cameras running computationally expensive tasks including video compressing and neural networks. (**B**) Trajectories of four drones. Rhombus with black edges are drone positions at four timestamps. (**C**) Heatmaps of target distributions in camera views. Yellow regions indicate more frequent target appearance. (**D**) A time sequence demonstrates system resistance to target occlusion.



Movie 5. Multi-drone tracking with target occlusion.

planes from others and obstacles, which further introduces extra decision variables. We conducted more tests with an extra pair of centralized swarm trajectory planners, SCP (sequential convex programming) (*48*) and RBP (relative Bernstein polynomial) (*49*), both of which require global maps and sufficient time for offline computation, as given in the Supplementary Materials.

Swarm playground

To encourage more involvement and further development, all the codes are included in the Supplementary Materials to inspire user-friendly running and interaction, which is named the swarm playground. In this playground, users can watch a swarm of 40 drones fly freely to given goals (Fig. 8A), watch seven drones form a centered hexagon (Fig. 8B), or watch drones swap positions under either predefined

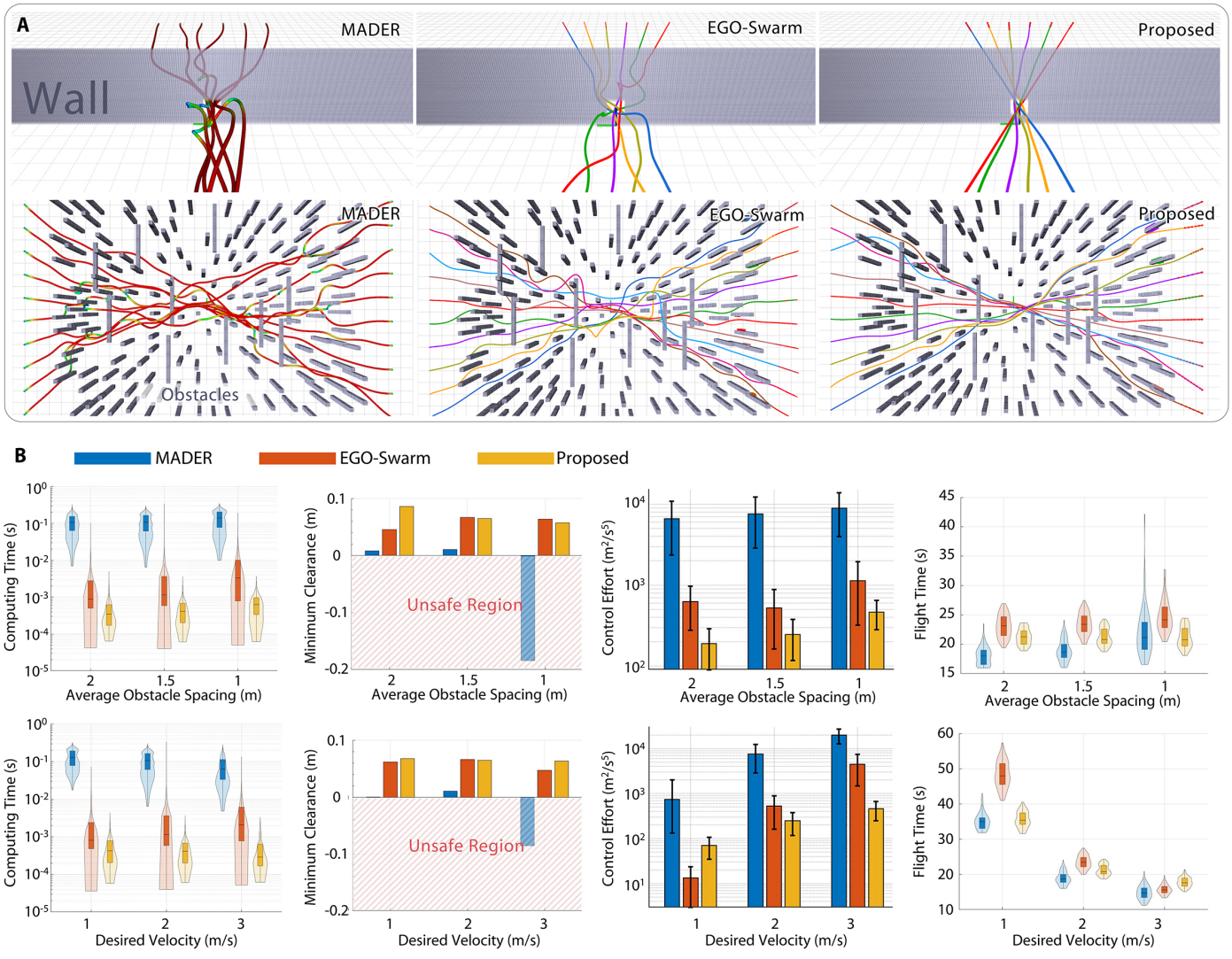


Fig. 7. Benchmark comparisons. (**A**) Trajectory shape visualization. Top: Six drones fly through a narrow gate. Bottom: Ten drones fly across obstacles. All the drones start together at identical desired velocity, and the order of start and goal positions are reversed to enforce reciprocal collision avoidance. The edge length of the grid on the ground is 1 m. (**B**) Metrics comparisons visualized using violin plots and bar graphs. The computing time is the time used for planning. The minimum clearance records the minimum distance to other drones or obstacles subtracted from the 0.2-m drone radius, where a negative value indicates a crash. The control effort (which evaluates smoothness) $S_m := \int j(t)^2 dt$ measures the time integral of squared jerk j during the entire flight, because jerk is directly related to the body turn rate and large jerk makes the drone flight shaky. The flight time measures the time spent to reach the given goals. Bar graphs with or without SD are used. Violin plots record the medium and quartile, along with distribution of the data. The specific values can be found in tables S8 to S13. Plots and graphs can be recreated using the released dataset (*66*).

(Fig. 8C) or endless random goals (Fig. 8D) on a circle. Goals can be given by users as well in a select-and-set way, as in the video game *Command & Conquer: Red Alert 2* (2000). Furthermore, users can more deeply take part in the planning process by acting as a tracked object (Fig. 8E) or dynamic obstacles that drones must avoid (Fig. 8F), using one or multiple Microsoft Xbox Controllers (*50*). All the static obstacles in the playground are randomly generated. The system parameters—including drone numbers, flight velocity, start, and goal positions—can be reconfigured following the tutorials. In addition, if new objectives are added, users are encouraged to evaluate the correctness of problem formulation and parameter settings before real-world deployment. The code can be effortlessly deployed on Ubuntu 16.04, 18.04, and 20.04 with the Robot Operating System being the only dependency installed. The video of the swarm playground is Movie 6.

DISCUSSION

In this work, a modular and hierarchical system achieving swarm intelligence based on high-level individual intelligence is proposed and validated, in which all drones are developed with the capability of sensing the environment and planning a locally optimal trajectory. This framework is adopted because we focus on fully autonomous navigation in real-world unstructured fields without prior knowledge while satisfying TEEM (as discussed in Introduction).

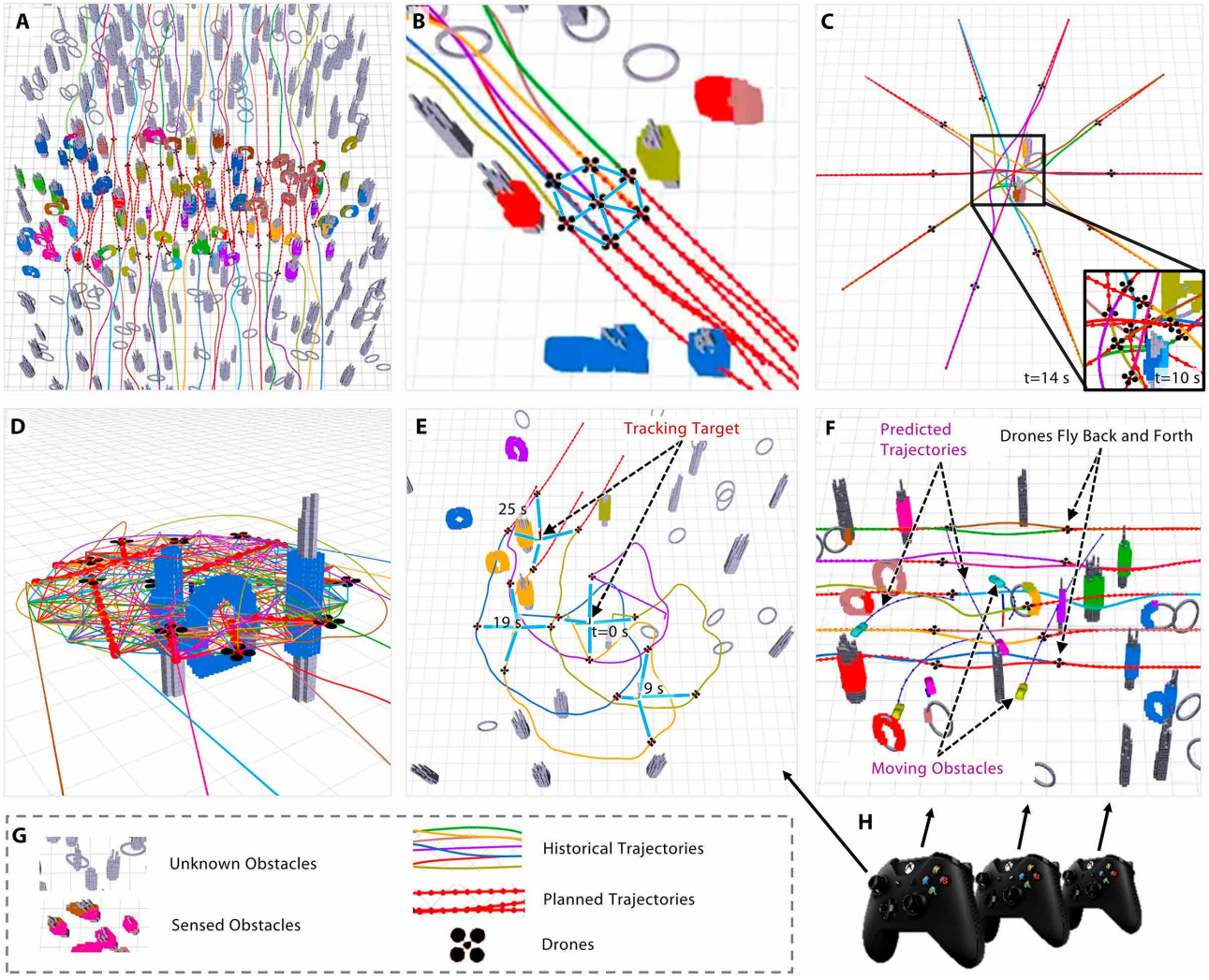
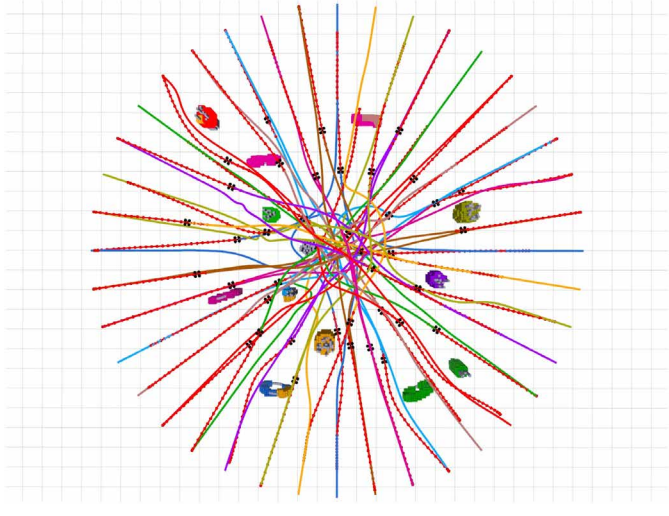


Fig. 8. Pictures generated in the software package of swarm playground. (**A**) A swarm flight of 40 drones. (**B**) Formation flight. (**C**) A circle position swap. (**D**) Endless random goals selected on a circle. (**E**) Swarm tracking with the target manually controlled. (**F**) Avoiding multiple dynamic obstacles that are manually controlled. (**G**) Legend. (**H**) Gamepads used to control the tracking target and dynamic obstacles.



Movie 6. Swarm playground.

From the simulation benchmarks to real-world experiments, we successfully demonstrate TEEM in both software and hardware. The presented underlying capability of navigation is compatible with further incorporating elaborate mission-specific requirements. For example, in formation flight, there can be no prior-defined shape, and thus, drones must only stay close to each other, making them behave like bird flocks. Moreover, the preferred shape can adaptively change according to task requirements. The trajectory planner is capable of avoiding multiple dynamic obstacles, as shown in Fig. 8F, although the obstacles' future paths are not precisely known because they are controlled by users in real time.

Here, we de-emphasize rigorous completeness and optimality proofs to achieve real-world performance because the entire system composed of complex real-world interaction modules has already shown to be difficult to rigorously mathematize. Fortunately, some issues such as the deadlock (inaction due to a stalemate situation owing to resource allocation challenges) in multiagent literature (*51*, *52*), which focus more on completeness and optimality, are naturally mitigated by our method, owing to real-world randomness, asynchronously

triggered planning, and a high degree of freedom in our trajectory representation. Similarly, deadlock is also reported to hardly occur in other trajectory planning methods (*53*) in normal situations, unless under conditions with special environment geometry, such as being inside a narrow and long tube.

In summary, our proposed planner follows a goal-chasing scheme in which users can give goals at any time during the tasks. Such a scheme can seamlessly connect to high-level decision-making and task assignment modules with outputs that are always preferred goals for each robot (*54*, *55*). Furthermore, the proposed multi-objective trajectory planner allows high-level modules to focus on task abstraction without having to worry about common requirements such as safety and dynamical feasibility.

MATERIALS AND METHODS

System architecture

Following the single-to-swarm approach, a decentralized scheme is naturally constructed, in which each drone is equipped with full autonomy for maximal navigation quality. The system architecture is depicted in Fig. 2B. The trajectory-broadcasting network is the only connection between individuals. Therefore, the system is less coupled than in previous work (*33*), which requires a stable chain connection. The mapping module is based on probabilistic mapping (*7*) that shows robustness and efficiency. The drone removal module removes pixels of other witnessed drones in depth from interfering with mapping. A VIO-based localization module along with the proposed drift-correction algorithm computes the six–degree-of-freedom drone states. The controller module commands the drone to precisely track planned trajectories. The planning module that generates high-quality trajectories is the core of achieving TEEM and therefore is further detailed in the "Trajectory representation" to the "Dynamic obstacle avoidance" sections and in the Supplementary Materials. Hardware modules are introduced in the "Palm-sized drone hardware" section. The drone removal module, which takes other drones' trajectories to determine three-dimensional bounding boxes as trust regions, is detailed in section S8 of the Supplementary Materials.

Trajectory representation

Here, spatial-temporal trajectory planning is achieved using a newly developed trajectory representation named MINCO (minimum control) (*56*) that is designed for differentially flat systems like multicopters (*4*). The most advanced aspect of MINCO is that it decouples the space and time parameters of a trajectory for users, on which linear-complexity operations are designed for convenient spatial-temporal deformation. The parameters of a MINCO piece-wise trajectory are the (i) time durations $\mathbf{T} \in \mathbb{R}^M$ of each piece and (ii) the adjacent waypoints $\mathbf{q} \in \mathbb{R}^{3 \times M-1}$ between each pair of connected pieces, where M is the piece number. Then, a three-dimensional point $p(t) \in \mathbb{R}^3$ at time t on the MINCO trajectory is defined by an operation $\mathcal{M}$

$$p(t) = \mathcal{M}_{\mathbf{q},\mathbf{T}}(t) \tag{1}$$

According to the "Optimality conditions" section in (*56*), for an s-integrator chain dynamics ($s = 3$ in this work), MINCO trajectory is by default a $2s - 1$ degree C^{s-1} polynomial spline with constant boundaries and minimum control effort given $\{\mathbf{q}, \mathbf{T}\}$. Its control effort optimization is given by

$$\min_{p(t)} \int_{t_0}^{t_M} \| p^{(s)}(t) \|_2^2 \, \mathrm{d}t \tag{2}$$

where $t \in [t_0, t_M]$ is the domain of the current trajectory. Note that smoothness maximization is done through control effort minimization because we use a jerk-control system model.

Furthermore, MINCO is advanced in converting the given parameters $\{\mathbf{q}, \mathbf{T}\}$ to polynomial coefficients $\mathbf{c}$ and time profile $\mathbf{T}_\mathrm{p}$ with a linear complexity $O(M)$. A more specific correspondence can be expressed as

$$\mathbf{M}(\mathbf{T})\,\mathbf{c} = \mathbf{b}(\mathbf{q}), \mathbf{T}_\mathrm{p} = \mathbf{T} \tag{3}$$

where $\mathbf{b}(\mathbf{q}) \in \mathbb{R}^{2Ms \times 3}$ and $\mathbf{M}(\mathbf{T}) \in \mathbb{R}^{2Ms \times 2Ms}$ is a nonsingular banded matrix for any $\mathbf{T} \succ 0$ as shown in (*56*). Recovering a trajectory enjoys linear complexity via banded PLU factorization. The gradients for the polynomial coefficients is also propagated to MINCO parameters in linear time. This means that once partial gradients of objectives on $\{\mathbf{c}, \mathbf{T}_\mathrm{p}\}$ are acquired, they can be efficiently propagated onto $\{\mathbf{q}, \mathbf{T}\}$, and then optimization can be directly applied to MINCO. Details of $\mathbf{M}$, $\mathbf{b}$ and more characteristics of MINCO are given in the Supplementary Materials. Specifically, computing time from given $\{\mathbf{q}, \mathbf{T}\}$ to $\{\mathbf{c}, \mathbf{T}_\mathrm{p}\}$ by Eq. 3 is approximately 1 μs per polynomial piece on a desktop computer.

Constraints transcription

The differential flatness of multicopters (*4*) implies that their motion planning can be performed on low-dimensional smooth trajectories, such as MINCO. To achieve smooth motions and efficient flights, we define two metrics for smoothness and time, respectively, and then minimize their weighted sum. Decision variables are the MINCO parameters $\mathbf{q}$ and $\mathbf{T}$. Start and terminal states are fixed to ensure continuity. The feasibility requires trajectories to fulfill vehicle's dynamical constraints and to avoid obstacles. The flatness makes it possible to enforce dynamical constraints by restricting magnitudes of trajectory velocity, acceleration, and jerk. Obstacle avoidance is achieved by deforming the trajectory shape.

Continuous-time constraints along the trajectory consist of infinitely many inequalities. To handle the difficulty, we propose a two-step procedure for constraint transcription. First, inspired by (*40*), constraints are enforced via integrals of penalty functions with large enough penalty weights. Second, every integral is evaluated by a finite sum of equally spaced samples along the timeline. The problem finally becomes an unconstrained one that can be solved more efficiently (*57*). For an optimization of time-dependent objectives J under equality constraints $\mathcal{H}$ and inequality constraints $\mathcal{G}$ within time $t \in [t_0, t_M]$, $t_M - t_0 = \mathrm{sum}(\mathbf{T})$, this two-step conversion can be written as

$$\min_{\mathbf{q},\mathbf{T}} \int_{t_0}^{t_M} J(\mathbf{q}, \mathbf{T}, t) \, \mathrm{d}t \tag{4}$$

$$\mathrm{s.t.}\, \mathcal{H}(\mathbf{q}, \mathbf{T}, t) = 0, \mathcal{G}(\mathbf{q}, \mathbf{T}, t) \leq 0 \tag{5}$$

⇓

$$\min_{\mathbf{q},\mathbf{T}}\left(\int_{t_0}^{t_M} J\mathrm{d}t + \int_{t_0}^{t_M} \|\chi_H \cdot \mathcal{H}\|_2^2 + \max\left(\chi_G \cdot \mathcal{G}, 0\right)^3 \mathrm{d}t\right) \tag{6}$$

$$\Downarrow$$

$$\min_{\mathbf{q},\mathbf{T}} \sum_{i=0}^{\kappa} \omega_i \cdot \left(J(t_i) + \|\chi_H \cdot \mathcal{H}(t_i)\|_2^2 + \max\left(\chi_G \cdot \mathcal{G}(t_i), 0\right)^3\right) \tag{7}$$

We omit arguments $\mathbf{q}$, $\mathbf{T}$, and t in Eqs. 6 and 7 for simplicity. In these equations, χ_H and χ_G are user-defined weights with appropriately large entries. $t_i = t_0 + (t_M - t_0)i/\kappa$ indicates a finite number of sampled timestamps, where $\kappa + 1$ equals the sample number, and ω_i is the interval value for integral evaluation. If any integral in J has a closed-form expression, like smoothness and total time, analytical results should be used. The Eq. 7 can be written in a user-friendly form

$$\min_{\mathbf{q},\mathbf{T}} \sum_{x} \lambda_x J_x \tag{8}$$

where J_x are various terms of penalties, i.e., task specifications, and λ_x are relative weights. Subscripts $x = \{s, t, d, o, w\}$ denote smoothness (s), total time (t), dynamical feasibility (d), obstacle avoidance (o), swarm collision avoidance (w), etc.

Solving Eq. 7 suffers from high complexity if a raw piece-wise polynomial trajectory is used because dense matrix inversion is inevitable. In contrast, linear-complexity operations of MINCO in the "Trajectory representation" section greatly reduce the computation overhead within each iteration. Along with the compact parameter representation, the total convergence speed is accelerated by orders of magnitude. In another aspect, because the constraints are converted into objectives, the feasibility is guaranteed by postchecking as described in the "Hierarchical safety guarantee" section.

Trajectory planning procedure

The proposed trajectory planner runs as follows. Step 1. A user or software gives a global goal position. Step 2. The planner selects a local target within the predefined local planning distance along the direction to the goal and then starts the iteration with an initial guess. Step 3. Within each iteration, the solver returns a solution trajectory of the warm-start optimization. Step 4. The total penalty J and the gradients are calculated and then sent back to the solver before returning to step 3. Step 5. A local trajectory from the current position to the local target satisfying task requirements is returned and executed. Step 6. After a given period (always either one or several seconds) or whenever the trajectory has collided with newly sensed obstacles, the planning is reactivated by returning to step 2. This procedure is repeated until the drone reaches the goal. In this work, we use an open-source L-BFGS solver (*58*), which belongs to the category of quasi-Newton methods of optimization for the next candidate trajectory at the next iteration.

General purpose penalties

Maximizing smoothness

According to Eq. 2, the smoothness penalty J_s is defined as the integral of the squared s-order derivatives, which gives

$$J_s = \int_{t_0}^{t_M} \|p^{(s)}(t)\|_2^2 \mathrm{d}t \tag{9}$$

This integral can be analytically calculated because the MINCO trajectory can be represented as piece-wise polynomials according to Eq. 3.

Minimizing total time

A shorter flight time is desirable (*1*) in most cases, so we also minimize the weighted total flight time, which gives the total time penalty J_t as

$$J_t = \mathrm{sum}(\mathbf{T}) \tag{10}$$

Dynamical feasibility

For differentially flat multicopters, dynamical feasibility is guaranteed by restricting magnitudes of the trajectory derivatives. In our work, we limit the amplitude of velocity, acceleration, and jerk by adding a penalty if these derivatives exceed the physical thresholds, which are

$$J_{d,v} = \sum_{i=0}^{\kappa} \max\left\{\left(\dot{p}(t_i^2) - v_{\mathrm{m}}^2\right), 0\right\}^3 \tag{11}$$

$$J_{d,a} = \sum_{i=0}^{\kappa} \max\left\{\left(\ddot{p}(t_i^2) - a_{\mathrm{m}}^2\right), 0\right\}^3 \tag{12}$$

$$J_{d,j} = \sum_{i=0}^{\kappa} \max\left\{\left(\dddot{p}(t_i)^2 - j_{\mathrm{m}}^2\right), 0\right\}^3 \tag{13}$$

$$J_d = J_{d,v} + J_{d,a} + J_{d,j} \tag{14}$$

where v_m, a_m, and j_m are the maximum allowed magnitudes of velocity, acceleration, and jerk, respectively; t_i here and in later sections follows the same definition in Eq. 7. We directly sum $J_{d,v}$, $J_{d,a}$, and $J_{d,j}$ because they share similar magnitudes although with different units.

Obstacle avoidance

Obstacle avoidance is performed on the unordered obstacle map built from the cluttered real world. In our work, we model obstacles as planes $(\mathbf{x} - \mathbf{s})^{\mathrm{T}}\mathbf{v} = 0$, $\mathbf{x} \in \mathbb{R}^3$, for which we treat the obstacle side of the plane as occupied and the other side as free (*14*). Here, $\mathbf{s} \in \mathbb{R}^3$ is a point on the plane, and $\mathbf{v} \in \mathbb{R}^3$ is a normal vector pointing to the free side. Then, for any point $\mathbf{p} \in \mathbb{R}^3$, its distance d_{o} to obstacles is defined as

$$d_{\mathrm{o}} = (\mathbf{p} - \mathbf{s})^{\mathrm{T}}\mathbf{v} \tag{15}$$

This is a highly simplified obstacle representation, but it shows an appropriate balance between fidelity and computation overhead in our previous work for single drone navigation (*14*). A detailed description of $\{\mathbf{s}, \mathbf{v}\}$ generation can be found in the Supplementary Materials. Specifically, generating a plane typically takes 0.1 ms on a desktop computer and thus is sufficient to be executed onboard. Following the definition of d_0, we penalize if $d_0 < \mathcal{C}_{\mathrm{o}}$ with $\mathcal{C}_{\mathrm{o}} > 0$ an obstacle clearance. Then, the obstacle avoidance penalty J_{o} is formulated as

$$J_{\mathrm{o}} = \sum_{i=0}^{\kappa} \max\left\{\left(\mathcal{C}_{\mathrm{o}} - d_{\mathrm{o}}(p(t_i))\right), 0\right\}^3 \tag{16}$$

where d_{o} is a function of $p(t)$, and therefore J_{o} is a function of the MINCO trajectory.

From single to swarm

Swarm flight further requires that no collision occurs between individuals. In real-world usage, we assume bandwidth-limited wireless communication to be available for robots. Therefore, compact

parameters of MINCO are broadcasted in wireless networks. By knowing neighbors' trajectories, a planner can accurately evaluate the swarm distribution and relative velocities at any timestamp in the short term, typically several seconds. This duration is sufficient for safety because a drone always replans a new trajectory before the end of a current one. Reciprocal avoidance is also formulated as a penalty J_{w} that produces a high cost when two drones are too close. Therefore, for the uth drone in a swarm containing U drones, J_{w} is defined as

$$J_{\mathrm{w}} = \sum_{k=1,k\neq u}^{U} \sum_{i=0}^{\kappa} {}^{k}J_{\mathrm{w}}(t_i) \tag{17}$$

$${}^{k}J_{\mathrm{w}}(t_i) = \max\left\{\mathcal{C}_{\mathrm{w}}^2 - {}^{k}d_{\mathrm{w}}(t_i)^2, 0\right\}^3 \tag{18}$$

$${}^{k}d_{\mathrm{w}}(t_i) = {}^{k}d_{\mathrm{w}}({}^{u}p(t_i), {}^{k}p(\tau)) = \|\mathbf{E}^{1/2}({}^{u}p(t_i) - {}^{k}p(\tau))\| \tag{19}$$

where ${}^{u}p(t_i)$ and ${}^{k}p(\tau)$ are the trajectories of the uth and kth drones, respectively. An offset between t_i and τ aligns them to the same global time. $\mathcal{C}_{\mathrm{w}}$, named swarm clearance, is the minimum safety clearance between two drones. The matrix $\mathbf{E} \coloneqq \mathrm{diag}(1,1,1/c)$ with $c > 1$ transforms a Euclidean distance into an ellipsoidal distance with the minor axes at the z axis to relieve the downwash risk from rotors. Next, the optimization problem remains unconstrained and hence can be solved efficiently.

Formation expectation

The formation is defined as fixed vertexes in a local frame $\mathcal{F}$, which moves and rotates with respect to the world frame. To stay in formation, each drone assigned with a vertex plans its trajectories according to others' movements calculated from others' trajectories. When the flight starts after receiving a long-term goal, the formation is required to move strictly along the straight line l, connecting the current position and the goal. The x axis of $\mathcal{F}$ is parallel to l. Then, the frame origin is determined by fitting the formation shape with the current swarm distribution. This formation inference can retrieve the formation position in the near future, which gives a guiding path $g(t)$ to each drone for the trajectory planning. Then, the formation penalty J_{f} is defined as

$$J_{\mathrm{f}} = \sum_{i=0}^{\kappa} \|p(t_i) - g(t_i)\|_2^2 \tag{20}$$

To enforce continuity, we assume a uniform motion beyond the domain of trajectory definition.

Multiview tracking and videoing

To record a participant while avoiding obstacles, an RGB camera must point at the participant and the depth camera in the direction that the drone flies. To avoid obstacles, we align the depth camera with drone velocity. Therefore, the constraints to adjust the view of depth camera are defined as (i) aligning the trajectory velocity $\dot{p}(t)$ with the predicted participant velocity v_p, which gives

$${}^{v}J_v(t) = \|\dot{p}(t) - v_p\|_2^2 \tag{21}$$

To keep the participant in the picture with appropriate size, we (ii) enforce a preferred drone's position ${}^{\mathcal{S}}P_{\mathrm{prf}}$ in the paricipant frame $\mathcal{S}$, which gives

$${}^{p}J_v(t) = \|p(t) - T_{\mathcal{S}}{}^{\mathcal{S}}P_{\mathrm{prf}}\|_2^2 \tag{22}$$

where $T_{\mathcal{S}}$ is a transformation matrix that transforms ${}^{\mathcal{S}}P_{\mathrm{prf}}$ into the world frame. Combining the above parts directly gives the multiview tracking penalty

$$J_v = \sum_{t_i} ({}^{v}J_v(t_i) + {}^{p}J_v(t_i)) \tag{23}$$

where ${}^{v}J_v$ and ${}^{p}J_v$ are directly added for the same reason as in Eq. 10. In our experiment, we use a constant velocity model to predict the participant's movement.

The video from an RGB camera connected via CSI (camera serial interface) is compressed using the JPEG (joint photographic experts group) standard (*59*) implemented by OpenCV (*60*) and stored in a stream. The object detection relies on YOLOv5 (*61*) neural network with model depth multiple set to 0.33 and layer channel multiple set to 0.50, in addition to other default parameters. It is accelerated using NVIDIA TensorRT (*62*). All of the aforementioned software demands substantial computing resource and high-bandwidth input-output. The z axis on the camera frame of the object is estimated using a prior known object height, and its global position is further filtered through a Kalman filter (*63*).

Dynamic obstacle avoidance

Dynamic obstacles with predicted trajectories are treated in the same way as other moving drones from the perspective of decentralized trajectory planning. Therefore, avoiding dynamic obstacles still follows the formulation presented in the "From single to swarm" section, except for different $\mathbf{E}$ and $\mathcal{C}_{\mathrm{w}}$ values according to obstacle shape and volume. In the "Swarm playground" section, a standard bicycle model is used to simulate and predict the movements of dynamic obstacles.

Localization and drift correction

Regarding real-world implementations in which high accuracy and robustness are always desirable, we use VIO to accurately obtain precise and high-frequency state estimations. However, without external positioning facilities, accumulative drift is unavoidable, which may cause inter-robot collisions during a long-range compact flight. To estimate and correct localization drift separately, we leverage relative distance measurements to each other along with their positions calculated from received trajectories. The relative distances are measured by onboard UWB sensors as also adopted in (*64*). For the uth drone in a swarm containing U drones, its range to the kth drone at position p_k is measured as $r_{u,k}$, and then we minimize the total distance measurement error

$$\min_{p_u}(\|p_u - p_{u0}\|_2^2 + \sum_{k=1,k\neq u}^{U} (\|p_u - p_k\|_2^2 - r_{u,k}^2)^2) \tag{24}$$

to acquire the uth drone's position p_u, where p_{u0} is the latest odometry corrected by the last drift estimation. Note that a regularization term for p_{u0} is added to avoid a nonunique or unstable solution, e.g., a whole spherical surface satisfies the minimization when $U = 2$. The problem is solved using numerical optimization, and p_{u0} is also taken as the initial value. To further smooth the odometry while improving accuracy, we estimate the slow-changing drift and apply a low-pass filter to it, instead of directly using the optimized p_u. The drift is then added to the latest odometry from VIO to produce corrected localization.

Furthermore, stationed facilities with ground-truth positions can also be incorporated into the optimization to ensure global consistency as in the "Intensive reciprocal avoidance evaluation" section. Note that global frames are required to be roughly consistent initially; otherwise, the nonlinear optimization lacks reliable initialization. Beyond improved localization accuracy, this approach brings almost no extra communication burden because other drone positions are calculated from trajectories, and UWB shares different radio frequencies with trajectory broadcasting networks. In our systematic solution, we use VINS (visual-inertial navigation system) (*5*) as the VIO and the Ceres Solver (*65*) for optimization. Evaluations of effectiveness from our real-world experiments and a corresponding block diagram are presented in the Supplementary Materials.

Hierarchical safety guarantee

Because the safety constraints are transferred to penalties, the output trajectory from the solver can still be infeasible, so a postcheck after trajectory planning is required. If a safety constraint is violated, the planner increases its weight and then makes another trial to improve the possibility of finding a satisfactory solution. If it is still infeasible after several trials, the current planning is terminated, and the planner waits for 10 ms before activating the next replanning. However, the postcheck after each planning only guarantees feasibility at that time point because the map is changing, so a safety checking process continuously checks collisions in the background. Once unsafety is detected, this process activates a replanning immediately. If this trial fails, and the predicted time to collide is under a threshold, an emergency stop trajectory is generated. After halting, the planning tries to start up again. Such a fallback guarantees safety in the most severe case and recovers the mission afterward.

Palm-sized drone hardware

All the experiments are performed on a 114-mm wheelbase microplatform that we designed, assembled, and released (*66*) with the hardware list here. The total weight of the platform is less than 300 g, including a 100-g battery providing an 11-min flight time. The drone is made up of the following five subsystems, as shown in Fig. 2A.

1) Power and movement suite. Two LiPo batteries of 3000-mAh capacity and 7.4-V voltage are connected in series, and four 6000-kv brushless motors (model 1404) with 3-inch, three-blade propellers are used, which give a thrust-to-weight ratio of 2.4. A four-in-one electronic speed controller with a 15-A maximum current is used. The propellers are mounted at the bottom of the airframe, and therefore, a strong downwash flow will not blow directly onto the body. Such a design improves flight time according to our experiments, and the average power consumption used in hovering is around 120 W.

2) Low-level control unit. A nanoscale flight control unit (FCU) of size 16 mm by 32 mm by 8 mm running PX4 Autopilot (*67*) is built. The hardware following the PX4 standard is composed of a STM32 H7 MCU (*68*) and a BMI088 IMU (*69*) with an 8-GB memory card for logging. Here, we omit all the sensors except for the IMU because we run localization using VIO rather than relying on barometers, magnetometers, or GPS. This unit is responsible for low-level angle control and sending IMU data to the high-level navigation unit.

3) High-level navigation unit. This unit runs all the localization, planning, high-level control, and other task-specific codes and therefore requires sufficient computing performance. In our platform, we use an NVIDIA Xavier NX (*70*), a powerful computer for embedded and edge systems with a six-core CPU, 384-core GPU, and 8 GB of RAM. In our experiments, except for multiview videoing, CPU and GPU usages are all below 40%, which allows for considerable computing reserves for extra potential usage.

4) Sensors. We address the basic sensor settings to miniaturize the drone size while retaining high accuracy. We use a grayscale and depth camera Intel Realsense D430 (*71*, *72*) and an IMU from the FCU. The D430 camera outputs depth images for mapping and stereo grayscale images for localization. The UWB module is a Nooploop LTPS (*73*) with a DW1000 radio chip (*74*) inside.

5) Wireless communication modules. We test and implement two topological structures: (i) a star shape using a single-access-point TP-LINK TL-XVR6000L router with an EDIMAX EW-7822UCL (*75*) USB WiFi adapter and (ii) a decentralized peer-to-peer ad hoc network using an AzureWave AW-CB375NF (*76*) via PCIe (peripheral component interface express) interface. The first structure shows higher bandwidth, whereas the second is more suitable for a large swarm scale as assessed in the Supplementary Materials.

SUPPLEMENTARY MATERIALS

www.science.org/doi/10.1126/scirobotics.abm5954
Sections S1 to S10
Figs. S1 to S17
Tables S1 to S13

REFERENCES AND NOTES

1. P. Foehn, A. Romero, D. Scaramuzza, Time-optimal planning for quadrotor waypoint flight. *Sci. Robot.* **6**, eabh1221 (2021).
2. Nikkei, Teardown of DJI drone reveals secrets of its competitive pricing (2021); https://asia.nikkei.com/Business/China-tech/Teardown-of-DJI-drone-reveals-secrets-of-its-competitive-pricing.
3. Grand View Research, Commercial drone market size, share & trends analysis report by product (fixed-wing, rotary blade, hybrid), by application, by end-use, by region, and segment forecasts, 2021–2028 (2021); www.grandviewresearch.com/industry-analysis/global-commercial-drones-market.
4. D. Mellinger, V. Kumar, Minimum snap trajectory generation and control for quadrotors, in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation* (IEEE, 2011), pp. 2520–2525.
5. T. Qin, P. Li, S. Shen, VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robot.* **34**, 1004–1020 (2018).
6. K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, V. Kumar, Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robot. Autom. Lett.* **3**, 965–972 (2018).
7. H. Moravec, A. Elfes, High resolution maps from wide angle sonar, *IEEE International Conference on Robotics and Automation* (IEEE, 1985), vol. 2, pp. 116–121.
8. K. McGuire, C. De Wagter, K. Tuyls, H. Kappen, G. C. de Croon, Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robotics* **4**, eaaw9710 (2019).
9. M. Müller, S. Lupashin, R. D'Andrea, Quadrocopter ball juggling, in *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2011), pp. 5113–5120.
10. H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, E. Galceran, Continuous-time trajectory optimization for online uav replanning, *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2016), pp. 5332–5339.
11. Skydio, Skydio autonomy™, a new generation of drone intelligence, www.skydio.com/skydio-autonomy (2021).
12. DJI, Mavic series, powerful and foldable for aerial adventure, www.dji.com/products/mavic (2021).
13. J. Tordesillas, B. T. Lopez, J. P. How, FASTER: Fast and safe trajectory planner for flights in unknown environments, *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2019), pp. 1934–1940.
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. B. Rabta, C. Wankmüller, G. Reiner, A drone fleet model for last-mile distribution in disaster relief operations. *Int. J. Disaster Risk Reduct.* **28**, 107–112 (2018).

16. CAL FIRE, California department of forestry and fire protection, www.fire.ca.gov/incidents/ (2021).
17. D. Mackenzie, It's a bird, it's a plane, it's a ... spy? *Science* **335**, 1433 (2012).
18. A. Kamagaew, J. Stenzel, A. Nettsträter, M. ten Hompel, Concept of cellular transport systems in facility logistics, in *Proceedings of the 2011 International Conference on Automation, Robotics and Applications* (IEEE, 2011), pp. 40–45.
19. J. Alonso-Mora, S. Baker, D. Rus, Multi-robot formation control and object transport in dynamic environments via constrained optimization. *Int. J. Rob. Res.* **36**, 1000–1021 (2017).
20. D. Falanga, S. Kim, D. Scaramuzza, How fast is too fast? The role of perception latency in high-speed sense and avoid. *IEEE Robot. Autom. Lett.* **4**, 1884–1891 (2019).
21. F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, S. Shen, Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments. *IEEE Trans. Robot.* **36**, 1526–1545 (2020).
22. W. Hönig, J. A. Preiss, T. S. Kumar, G. S. Sukhatme, N. Ayanian, Trajectory planning for quadrotor swarms. *IEEE Trans. Robot.* **34**, 856–869 (2018).
23. J. A.-Mora, P. Beardsley, R. Siegwart, Cooperative collision avoidance for nonholonomic robots. *IEEE Trans Robot.* **34**, 404–420 (2018).
24. Intel, Intel Drone Light Shows, https://inteldronelightshows.com/ (2021).
25. High Great, High Great Drone Light Shows, www.hg-fly.com/en/ (2021).
26. CollMot Entertainment, CollMot Entertainment Drone Light Shows, https://collmot.com/ (2021).
27. C. W. Reynolds, Flocks, herds and schools: A distributed behavioral model, *Annual Conference on Computer Graphics and Interactive Techniques* (1987), pp. 25–34.
28. S. Hauert, S. Leven, M. Varga, F. Ruini, A. Cangelosi, J.-C. Zufferey, D. Floreano, Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate, in *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2011), pp. 5015–5020.
29. G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, T. Vicsek, Optimized flocking of autonomous drones in confined environments. *Sci. Robot.* **3**, eaat3536 (2018).
30. E. Soria, F. Schiano, D. Floreano, Predictive control of aerial swarms in cluttered environments. *Nat. Mach. Intell.* **3**, 545–554 (2021).
31. Bitcraze, Crazyflie flying development platform, www.bitcraze.io/ (2021).
32. G. Loianno, C. Brunner, G. McGrath, V. Kumar, Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu. *IEEE Robotics and Automation Letters* **2**, 404–411 (2016).
33. X. Zhou, X. Wen, J. Zhu, H. Zhou, C. Xu, F. Gao, EGO-Swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments, in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation* (2021).
34. E. Gent, Watch a swarm of drones fly through heavy forest--While staying in formation, www.science.org/news/2020/12/watch-swarm-drones-fly-through-heavy-forest-while-staying-formation (2021).
35. D. L. Altshuler, M. V. Srinivasan, Comparison of visually guided flight in insects and birds. *Front. Neurosci.* **12**, 157 (2018).
36. C. Ellington, Insects versus birds: The great divide, *AIAA Aerospace Sciences Meeting and Exhibit* (2006), p. 35.
37. J. Tordesillas, J. P. How, Mader: Trajectory planner in multiagent and dynamic environments, *IEEE Transactions on Robotics* (2022).
38. C. Richter, A. Bry, N. Roy, Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments, *Robotics Research* pp. 649–666 (2016).
39. V. Usenko, L. Von Stumberg, A. Pangercic, D. Cremers, Real-time trajectory replanning for mavs using uniform b-splines and a 3d circular buffer, *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2017), pp. 215–222.
40. L. S. Jennings, K. L. Teo, A computational algorithm for functional inequality constrained optimization problems. *Automatica* **26**, 371–375 (1990).
41. R. B. Benson, E. Starmer-Jones, R. A. Close, S. A. Walsh, Comparative analysis of vestibular ecomorphology in birds. *J. Anat.* **231**, 990–1018 (2017).
42. R. R. Murphy, Swarm robots in science fiction. *Sci. Robot.* **6**, eabk0451 (2021).
43. D. Mellinger, M. Shomin, N. Michael, V. Kumar, Cooperative grasping and transport using multiple quadrotors, *Distributed autonomous robotic systems* pp. 545–558 (2013).
44. I. Mademlis, V. Mygdalis, N. Nikolaidis, M. Montagnuolo, F. Negro, A. Messina, I. Pitas, High-level multiple-UAV cinematography tools for covering outdoor events. *IEEE Trans. Broadcast.* **65**, 627–635 (2019).
45. A. Bucker, R. Bonatti, S. Scherer, Do you see what i see? coordinating multiple aerial cameras for robot cinematography, *2021 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2021), pp. 7972–7979.
46. J. Tordesillas, J. P. How, MINVO basis: Finding simplexes with minimum volume enclosing polynomial curves, *arXiv preprint arXiv:2010.10726* (2020).
47. C. De Boor, C. De Boor, E.-U. Mathématicien, C. De Boor, C. De Boor, *A Practical Guide to Splines*, vol. 27 (springer-verlag New York, 1978).
48. F. Augugliaro, A. P. Schoellig, R. D'Andrea, Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach, *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2012), pp. 1917–1922.
49. J. Park, J. Kim, I. Jang, H. J. Kim, Efficient multi-agent trajectory planning with feasibility guarantee using relative bernstein polynomial, *IEEE International Conference on Robotics and Automation* (IEEE, 2020), pp. 434–440.
50. Microsoft, Xbox Wireless Controller, www.xbox.com/en-US/accessories/controllers/xbox-wireless-controller (2021).
51. J. A. DeCastro, J. Alonso-Mora, V. Raman, D. Rus, H. Kress-Gazit, Collision-free reactive mission and motion planning for multi-robot systems. *Robot. Res.* 459–476 (2018).
52. J. Van Den Berg, S. J. Guy, M. Lin, D. Manocha, Reciprocal n-body collision avoidance. *Robot. Res.* 3–19 (2011).
53. J. Alonso-Mora, T. Naegeli, R. Siegwart, P. Beardsley, Collision avoidance for aerial vehicles in multi-agent scenarios. *Auton. Robot.* **39**, 101–121 (2015).
54. M. Coppola, K. N. McGuire, C. De Wagter, G. C. de Croon, A survey on swarming with micro air vehicles: Fundamental challenges and constraints. *Front Robot. AI* **7**, 18 (2020).
55. J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, D. Rus, On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proc. Natl. Acad. Sci. U.S.A.* **114**, 462–467 (2017).
56. Z. Wang, X. Zhou, C. Xu, F. Gao, Geometrically constrained trajectory optimization for multicopters [v2], *arXiv preprint arXiv:2103.00190* (2021).
57. Y. Nesterov, *Lectures on Convex Optimization*, vol. 137 (Springer, 2018).
58. FAST Lab, An Open-Source L-BFGS Solver, https://github.com/ZJU-FAST-Lab/LBFGS-Lite (2021).
59. G. K. Wallace, The JPEG still picture compression standard. *IEEE Trans Consum. Electron.* **38**, xviii–xxxiv (1992).
60. OpenCV team, OpenCV, https://opencv.org/ (2021).
61. YOLOv5 Contributors, YOLOv5, https://github.com/ultralytics/yolov5 (2021).
62. NVIDIA, NVIDIA TensorRT, https://developer.nvidia.com/tensorrt (2021).
63. G. Bishop, G. Welch, An introduction to the kalman filter, *Proc of SIGGRAPH. Course* **8**, 41 (2001).
64. H. Xu, Y. Zhang, B. Zhou, L. Wang, X. Yao, G. Meng, S. Shen, Omni-swarm: A decentralized omnidirectional visual-inertial-uwb state estimation system for aerial swarm, *arXiv preprint arXiv:2103.04131* (2021).
65. S. Agarwal, K. Mierle, Keir, Ceres Solver, http://ceres-solver.org (2021).
66. X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, F. Gao, Dataset - swarm of micro flying robots in the wild (version 1) [data set], Zenodo. https://doi.org/10.5281/zenodo.5804079 (2021).
67. L. Meier, D. Honegger, M. Pollefeys, Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms, *2015 IEEE international conference on robotics and automation (ICRA)* (IEEE, 2015), pp. 6235–6240.
68. STMicroelectronics, Stm32h743, www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-high-performance-mcus/stm32h7-series.html (2021).
69. BOSCH, Bmi088, www.bosch-sensortec.com/products/motion-sensors/imus/bmi088/ (2021).
70. NVIDIA, Jetson xavier nx, www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-nx/ (2021).
71. Intel, Intel realsense depth modules and processors, www.intelrealsense.com/stereo-depth-modules-and-processors/ (2021).
72. L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, A. Bhowmik, Intel realsense stereoscopic depth cameras, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2017), pp. 1–10.
73. NoopLoop, Linktrack, www.nooploop.com/en/ (2021).
74. Decawave, Dw1000, www.decawave.com/product/dw1000-radio-ic/ (2021).
75. EDIMAX, Ew-7822ulc, www.edimax.com/edimax/merchandise/merchandise_detail/data/edimax/global/wireless_adapters_ac1200_dual-band/ew-7822ulc/ (2021).
76. AzureWave, Aw-cb375nf, www.azurewave.com/wireless-modules-nvidia.html (2021).

Acknowledgments: We thank H. Ye, L. Quan, L.Yin, who offered valuable suggestions to the manuscript, and R. Jin for photography and video recording. We sincerely appreciate the work of H. Yin, Y. Li, Z. Wang, J. Guo, X. Zhu, and J. Wang for help on filed experiments. Furthermore, we are truly grateful for J. Zhu's help with benchmark comparisons. **Funding:** This work was supported by the National Natural Science Foundation of China under grant no. 62003299 and 62088101. **Author contributions:** X.Z. contributed to the hardware and software design, wild experiments, and manuscript writing. X.W. contributed to FMU design, power suit testing, communication evaluation, simulation benchmark, and wild experiments. Z.W. contributed to the idea of spatial-temporal optimization with the software of trajectory parameterization and numerical optimization. He also gave advisory suggestions and edited the manuscript. Y.G. contributed to data analysis and artwork. H. Li wrote reliable swarm communication

software and did some early work on localization correction. Q.W. wrote an early version of swarm tracking. T.Y. designed the power management module of the drone and edited an early version of the videos. H. Lu polished the manuscript. Y.C. gave several suggestions about UWB usage and helped with several tests. C.X. provided people and sites for drone testing. F.G. directed the research, provided the primary idea and funding with some key suggestions about software and hardware debugging, and revised the manuscript.
Competing Interests: The authors declare that they have no competing interests. **Data and materials availability:** All data needed to support the conclusions of this manuscript are included in the main text or Supplementary Materials and in an online dataset (*66*).

Submitted 5 October 2021
Accepted 22 February 2022
Published 4 May 2022
10.1126/scirobotics.abm5954

Science Robotics

Swarm of micro flying robots in the wild

Xin ZhouXiangyong WenZhepei WangYuman GaoHaojia LiQianhao WangTiankai YangHaojian LuYanjun CaoChao XuFei Gao

Sci. Robot., 7 (66), eabm5954. • DOI: 10.1126/scirobotics.abm5954