EGO-Swarm: A Fully Autonomous and Decentralized Quadrotor Swarm System in Cluttered Environments

Xin Zhou, Jiangchao Zhu, Hongyu Zhou, Chao Xu, and Fei Gao

Abstract— This paper presents a decentralized and asynchronous systematic solution for multi-robot autonomous navigation in unknown obstacle-rich scenes using merely onboard resources. The planning system is formulated under gradient-based local planning framework, where collision avoidance is achieved by formulating the collision risk as a penalty of a nonlinear optimization problem. In order to improve robustness and escape local minima, we incorporate a lightweight topological trajectory generation method. Then agents generate safe, smooth, and dynamically feasible trajectories in only several milliseconds using an unreliable trajectory sharing network. Relative localization drift among agents is corrected by using agent detection in depth images. Our method is demonstrated in both simulation and real-world experiments. The source code is released for the reference of the community.

I. INTRODUCTION

The agility of quadrotors enables this machine to perform single-agent autonomous navigation in unknown environments and multi-agent precise formation control in open or known fields. However, few works combine both of them to present any real-world system which is capable of navigating quadrotor swarms sharing the same unknown space, especially with only onboard processing. The difficulties in deploying multiple quadrotors in unknown environments include but not limit to nontriviality of obstacle parameterization, limited sensing range, unreliable and bandwidth limited communication, and positioning drift caused by inconsistent localization. Several related works, e.g. [1, 2], push the real-world quadrotor swarms capable of obstacle avoidance much further. However, above-mentioned difficulties are always neglected under motion capture systems or in pure simulation for most of the previous works, limiting the application of their algorithms in practice.

This paper proposes a systematic solution that enables high-performance traveling in cluttered environments for quadrotor swarms in the presence of these above-mentioned difficulties. In addition, it requires no external localization and computation or a pre-built map. The proposed system named **EGO-Swarm** is an extension of our previous work **E**SDF-free **G**radient-based l**O**cal planner (EGO-Planner) [3], which lays a solid baseline for onboard local planning of a single quadrotor in unknown environments. The expansion consists of two parts, topological planning, and reciprocal collision avoidance. As depicted in Fig.3, non-convex configuration space may lead to undesired behaviors, such as

All authors are with the State Key Laboratory of Industrial Control Technology, and the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, 310027, China. (email: {iszhouxin, fgaoaa}@zju.edu.cn).

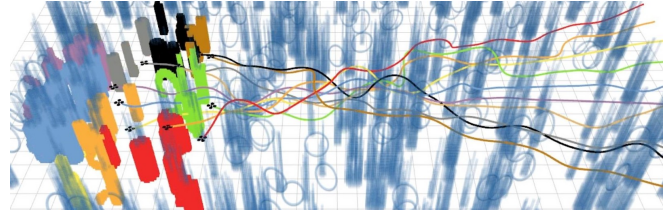Fig. 1: Three snapshots of a flight in forest of three quadrotors.

Fig. 2: Ten quadrotors fly in simulation. Colored blocks represent the local perception of quadrotors. Each color is associated with a quadrotor.

dynamical in-feasibility or crowded navigation. Therefore, a strategy such as topological planning to escape local minima is beneficial. Based on the collision cost formulation in EGO-Planner, the front-end topological path search is done implicitly, therefore requiring nearly no computation. Decentralized reciprocal collision avoidance is achieved by adding a weighted penalty of swarm collision to the objective function. This penalty is evaluated by comparing the agent distribution at a pried of future time with the being optimized trajectory. In order to minimize data transmission and allow unreliable communication, a broadcast network is used to share trajectories. To correct relative localization drift, which can increase to half a meter, observations of witnessed agents and predictions from trajectory evaluations are compared.

Real-world experiments are presented to validate our proposed swarm system. To the best of our knowledge, this is the first systematic solution for fully autonomous decentralized quadrotor swarms in unknown cluttered environments, which means perception, planning, and control are integrated into the onboard system. Comparisons with several SOTA methods show the computation efficiency and robustness. The contributions of this paper are summarized as follows:

1) We extend our previous work EGO-Planner to propose a novel and robust topological planning method with nearly no extra computation.
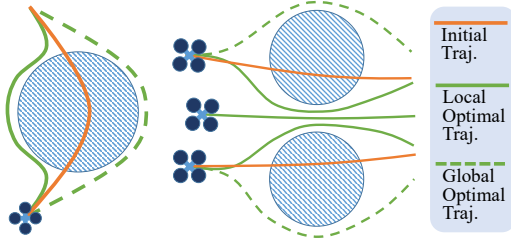2) We propose a decentralized and asynchronous quadro-

Fig. 3: **Left:** Dynamical infeasibility of the local optimal trajectory. **Right:** Three agents crowded through the flight corridor.

tor swarm framework which is insensitive to unreliable communication and localization drift.

3) The proposed method is integrated into a fully autonomous quadrotor system with hardware and software released for the reference of the community[1].

II. RELATED WORKS

A. Single Quadrotor Local Planning

Gradient-based motion planning is the mainstream for quadrotor local planning. Based on the pioneering works [4, 5] that formulate the local planning problem as unconstrained nonlinear optimizations, a series of works [6]–[10] are proposed. They consider smoothness, feasibility, and safety of the trajectory using various parameterization methods, including polynomial and B-spline. Recently, we propose a single-quadrotor navigation system named EGO-Planner [3], which further reduces computation time using a more compact environment representation. This is the work on which this paper is based.

B. Topological Planning

Topological planning is used to escape local minima. Based on the homology equivalence relation in 2-D surfaces originated from complex analysis [11], Rösmann et al. [12] present a trajectory planning method in distinctive topologies using Voronoi and sampling-based front-ends and TEB local planner [13] as back-ends. However, homology equivalence relation in 3-D is far more trivial. To capture distinctive useful paths, Jaillet et al. [14] construct visibility deformation roadmaps that encode richer and more relevant information than representative paths of the homotopy classes. Based on [14], Zhou et al. [15] enable real-time topological planning by proposing an efficient topology equivalence checking. We extend EGO-Planner to accelerate the front-end for topological planning further.

C. Decentralized Drone Swarm

Decentralized approaches have been proposed in, e.g., [16]–[20]. Velocity obstacles is leveraged to guarantee collision-free trajectories for point robots [16], holonomic agents [17], and non-holonomic agents [18]. Liu et al. [21] propose a decentralized and asynchronous planning strategy for drones to avoid static/dynamic obstacles and inter-vehicle collisions. While those algorithms are validated through simulations without integrating sensing, mapping,

[1]https://github.com/ZJU-FAST-Lab/ego-planner-swarm

and planning capabilities. Experimental results have been shown in [19, 20]. [19] enables multi-vehicle point-to-point transitions without external obstacles, and [20] relies on ad-hoc planning priority. However, none of them achieve full autonomy in field environments.

III. IMPLICIT TOPOLOGICAL TRAJECTORY GENERATION OF GRADIENT-BASED LOCAL PLANNING

In this section, we introduce our previous work on EGO-Planner [3] at first, on which the proposed swarm system is based. Then the proposed topological planning strategy is explained.

A. An ESDF-Free Gradient-based local planner

As a gradient-based local planner, EGO-Planner formulates trajectory generation as a non-linear optimization problem that trades off smoothness J_s, collision J_c, dynamically feasibility J_d, and terminal progress J_t. The optimization problem, with decision variables from control points $\mathbf{Q}$ of a uniform B-spline $\mathbf{\Phi}$ used to parameterize the trajectory, is given by

$$\min_{\mathbf{Q}} J_{\text{EGO}} = \sum \lambda_r J_r, \tag{1}$$

where $r = \{s, c, d, t\}$ and the subscripted λ indicates the corresponding weights. Terms of J can be divided into two categories: *minimum error* and *soft unilateral constraint*. Minimum error terms J_s and J_t, which minimize the total error between a linear transformation of decision variables $L(\mathbf{Q})$ and a desired value $\mathcal{D}$, are formulated as

$$J_r = \sum_{\mathbf{Q} \in \mathbf{\Phi}} \| L(\mathbf{Q}) - \mathcal{D} \|_n^n. \tag{2}$$

Soft unilateral constraint terms J_c and J_d, which penalize decision variables exceeding a specific threshold $\mathcal{T}$, are generally expressed as

$$J_r = \sum_{\mathbf{Q} \in \mathbf{\Phi}} \begin{cases} \left\| \frac{L(\mathbf{Q}) - (\mathcal{T} - \epsilon)}{S} \right\|_n^n & L(\mathbf{Q}) > (\mathcal{T} - \epsilon) \\ 0 & L(\mathbf{Q}) \leq (\mathcal{T} - \epsilon) \end{cases}, \tag{3}$$

where parameters S, n, and ϵ affect unilateral constraint approximation accuracy as described in [13]. Transformation $L(\cdot)$ and parameters are chosen according to penalty types.

In EGO-Planner, we propose a novel obstacle distance estimation method from environment information possessed by each $\mathbf{Q}$ independently. The information parameterized by several $\{\mathbf{p}, \mathbf{v}\}$ pairs is highly abstracted from surrounding obstacles, where $\mathbf{p}$ denotes an anchor point at the obstacle surface and $\mathbf{v}$ represents a safe direction pointing from inside to outside of that obstacle, as illustrated in Fig.4a. Then the obstacle distance d_{ij} of ith control point $\mathbf{Q}_i$ to j^{th} obstacle is defined as

$$d_{ij} = (\mathbf{Q}_i - \mathbf{p}_{ij}) \cdot \mathbf{v}_{ij}. \tag{4}$$

$\{\mathbf{p}, \mathbf{v}\}$ pair generation and trajectory optimization procedure are shown in Fig.4a and 4b. First, a naive initial trajectory $\mathbf{\Phi}$ is given, regardless of collision. Then a safe path $\mathbf{\Gamma}$ connecting two ends of the colliding segment of $\mathbf{\Phi}$ is searched. After that, vector $\mathbf{v}$ is generated from $\mathbf{\Phi}$ to $\mathbf{\Gamma}$ and

(a) $\{\mathbf{p}, \mathbf{v}\}$ Pair Generation   (b) A Safe Trajectory

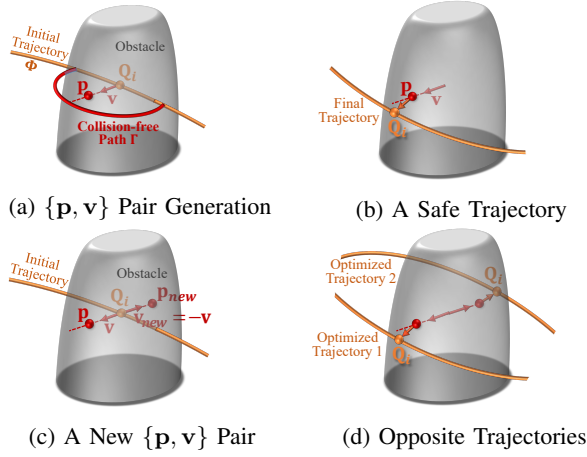(c) A New $\{\mathbf{p}, \mathbf{v}\}$ Pair   (d) Opposite Trajectories

Fig. 4: Illustration of EGO-Planner and topological trajectory generation. (a) A safe path bypassing a colliding trajectory is searched, then a $\{\mathbf{p}, \mathbf{v}\}$ pair is generated. (b) The optimizer finds a safe trajectory that satisfies $d = (\mathbf{Q}_i - \mathbf{p}) \cdot \mathbf{v}$ grater than a constant value. (c) A new $\{\mathbf{p}, \mathbf{v}\}$ pair with $\mathbf{v_{new}} := -\mathbf{v}$ and $\mathbf{p_{new}}$ at the opposite side of the obstacle is generated. (d) A different trajectory satisfying the constraint of $\{\mathbf{p}_{new}, \mathbf{v}_{new}\}$ pair is generated.
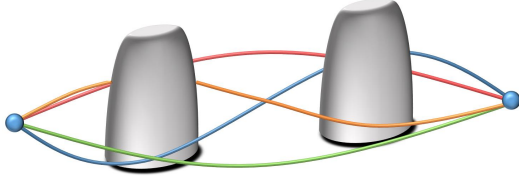

Fig. 5: Four trajectories belonging to the same 3-D homotopy stay in different local minima, which we aim to seek.

$\mathbf{p}$ is defined at the obstacle surface. With generated $\{\mathbf{p}, \mathbf{v}\}$ pairs, the planner maximizes d_{ij} and returns an optimized trajectory. Here we only give a simplified description of the basic idea of EGO-Planner due to the limited article length. Detailed explanations can be found in [3].

B. Implicit Topological Trajectory Generation

Analysis in [14, 15] reveals that the widely used homotopy concept is insufficient to capture candidate trajectories in 3-D cases, as shown in Fig.5. Therefore, Jaillet et al. [14] propose a more useful relation in 3-D space named visibility deformation (VD), and Zhou et al. [15] further extract a subset of VD called uniform visibility deformation (UVD), which enables real-time operation. However, we still use the term *topological planning* in this paper as previous works do without ambiguity. Trajectories satisfying UVD are considered homeomorphic. UVD defined in [15] is

Definition 1: Two trajectories $\tau_1(s), \tau_2(s)$, parameterized by $s \in [0, 1]$ and satisfying $\tau_1(0) = \tau_2(0)$, $\tau_1(1) = \tau_2(1)$, belong to the same UVD class, if for all s, line $\tau_1(s)\tau_2(s)$ is collision-free.

Traditional topological planning methods [12]–[15], which consist of topologically distinct path search and back-end optimization, focus mainly on finding multiple initial paths in distinct homotopy. Unlike those methods, the proposed method constructs distance fields in different directions by inverting $\mathbf{v}$ to $\mathbf{v}_{new} := -\mathbf{v}$. Then a search process de-
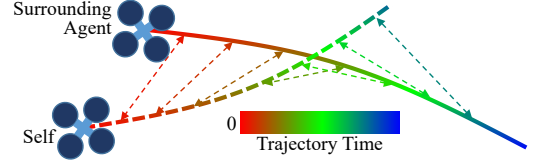

Fig. 6: Drone *Self* generates a trajectory by comparing the distance with a received trajectory from a surrounding drone of the same trajectory time.

termines a new anchor point $\mathbf{p}_{new}$ at the obstacle surface alone $\mathbf{v}_{new}$, as depicted in Fig.4c. They constitute a new pair $\{\mathbf{p}_{new}, \mathbf{v}_{new}\}$, which leads to a different local minima. Note that no explicit path search is adopted, but any pair of paths through $\mathbf{p}$ and $\mathbf{p}_{new}$ respectively violates Def.1 at these two points naturally. Subsequently, distinctive trajectories are optimized in parallel in different threads, as illustrated Fig.4d. The trajectory with lowest cost is executed.

IV. Drone Swarm Navigation

A. Reciprocal Collision Avoidance

Let $x_k(t) \in \mathcal{X} \subset \mathbb{R}^3$ be the position state of the agent k among K agents at time t. $\mathcal{X}_k^{free}(t) \subset \mathcal{X}$ is the free region in the state space of agent k considering the existence of other agents. Thus, $\mathcal{X}_k^{free}(t) := \mathcal{X} \backslash \{i \in \mathbb{Z} \backslash k, i \leq K | x_i(t)\}$ and the valid trajectory $\mathbf{\Phi}_k$ satisfies for any t within the domain of $\mathbf{\Phi}_k$, $\mathbf{\Phi}_k(t) \in \mathcal{X}_k^{free}(t)$, as depicted in Fig.6. Unlike [22], here we ignore obstacles and dynamic constraints which have already get tackled in Sec.III-A.

Similar to the penalty on obstacle collision and dynamical infeasibility, we formulate the penalty function $J_{w,k}$ of swarm collision avoidance for agent k as *soft unilateral constraint*

$$J_{w,k} = \sum_i \int_{t=t_s}^{t_e} \begin{cases} d_{k,i}(t)^2 & dt, \quad d_{k,i}(t) < 0 \\ 0 & d_{k,i}(t) \geq 0 \end{cases}, \quad (5)$$
$$d_{k,i}(t) = \left\| \mathbf{E}^{1/2} \left[\mathbf{\Phi}_k(t) - \mathbf{\Phi}_i(t) \right] \right\| - (\mathcal{C} + \epsilon)$$

where $i \in \mathbb{Z} \backslash k, i \leq K$, t_s and t_e are global start and end time within the time span of trajectory $\mathbf{\Phi}_k(t)$. $\mathbf{E} := \mathrm{diag}(1, 1, 1/\mathrm{c}), \mathrm{c} > 1$ transforms Euclidean distance into *ellipsoidal distance* with shorter principal axes at z-axis to relief downwash risk. Adding the weighted $J_{w,k}$ to Equ.1 yields the total optimization problem for each agent

$$\min_{\mathbf{Q}} J = J_{\mathrm{EGO}} + \lambda_w J_w. \quad (6)$$

Any trajectory parameterization method containing a mapping from decision variables to points on the trajectory is applicable to Equ.5. This paper parameterizes trajectory using a p_b-degree uniform B-spline, which gives the position evaluation a matrix representation [9]

$$\begin{aligned} \mathbf{\Phi}(t) &= \mathbf{s}(t)^\top \mathbf{M}_{p_b+1} \mathbf{q}_m \\ \mathbf{s}(t) &= \begin{bmatrix} 1 & s(t) & s^2(t) & \cdots & s^{p_b}(t) \end{bmatrix}^\top \\ \mathbf{q}_m &= \begin{bmatrix} \mathbf{Q}_{m-p_b} & \mathbf{Q}_{m-p_b+1} & \mathbf{Q}_{m-p_b+2} & \cdots & \mathbf{Q}_m \end{bmatrix}^\top \end{aligned}, \quad (7)$$

where $\mathbf{M}_{p_b+1}$ is a constant matrix determined by p_b, $s(t) = (t - t_m)/\triangle t$ when t belongs to the knot span $(t_m, t_{m+1}]$.

B. Localization Drift Compensation

As a result of individual localization in unknown environments (no reliable and high-frequency loop-closure), drift accumulates during the flight. Xu et al. [23] propose a state estimation method for aerial swarms with extra UWB distance measurements and achieve accurate collaborative localization. However, we focus more on traversing obstacle-rich environments and have to reserve computation and communication resources for other applications. Therefore, inspired by [23], a simplified and lightweight relative drift estimation method is proposed by comparing the predicted position evaluated from received agents' trajectories and the measured positions from depth images of witnessed agents. This strategy works when the trajectory tracking error is negligible and at least one of any two agents that might collide see the other. Therefore, we use the controller from [24] for precise tracking and use wide-angle cameras to reduce the possibility of losing agents.

The drift elimination procedure is as follows. After evaluating the current position $\mathbf{\Phi}_i(t_{now})$ of agent i, a spherical trust region $\mathcal{S} \subset \mathbb{R}^3$ centered at $\mathbf{\Phi}_i(t_{now})$ with radius $\mathcal{R}$ is determined, where $\mathcal{R}$ is an empirical parameter indicating the upper bound of typical drift estimated from experiments. $\mathcal{S}$ is then mapped to the currently captured depth image, which is the region $\mathcal{S}' \subset \mathbb{R}^2$ satisfying

$$z \begin{bmatrix} \mathbf{s}'^\top & 1 \end{bmatrix}^\top = \mathbf{K}\mathbf{T}_\mathrm{w}^\mathrm{c} \begin{bmatrix} \mathbf{s}^\top & 1 \end{bmatrix}^\top, \tag{8}$$

where $\mathbf{s}' \in \mathcal{S}'$, $\mathbf{s} \in \mathcal{S}$, $\mathbf{K}$ and $\mathbf{T}_\mathrm{w}^\mathrm{c}$ are camera intrinsic and extrinsic matrices, z is the deviation of s along the main optical axis from the optical center. $\mathcal{S}'$ is an elliptical conic section which requires complex calculation to acquire. Therefore, we adopt an approximate axis-aligned ellipse $\bar{\mathcal{S}}'$ instead of the exact $\mathcal{S}'$. There is no necessity to precisely define the trust region since it is just an empirical region.

Then we project each point within $\bar{\mathcal{S}}'$ into the world frame and collect points belonging to $\mathcal{S}$, which results in a point cluster $\mathcal{P} \subset \mathcal{S}$. Then the position of agent observation $\mathbf{P}$ is regarded as the center (first raw moment) of $\mathcal{P}$, which is

$$\mathbf{P} = \mu'_1(\mathcal{P}). \tag{9}$$

Equ.9 holds if $\mathcal{P}$ contains only the corresponding agent's observation without any unrelated object, which is not guaranteed. However, since each agent plans trajectories with a clearance to nearby objects, Equ.9 holds for most of the time. Additional criteria are added to improve the robustness for agent detection, such as the number of pixels, the second central moment of $\mathcal{P}$, the deviation of the current measurement to previous ones, etc. Stricter criterion increases false-negative rate, but it is harmless since localization drift alters slowly. Finally, the error between $\mathbf{\Phi}_i(t_{now})$ and $\mathbf{P}$ is fed to a filter, from where the estimated drift is then acquired.

C. Agent Removal from Depth Images

We use occupancy grid map to store static obstacles, and depth images for map fusion. The moving agents are taken care of in Sec.IV-A. Therefore, recording moving agents
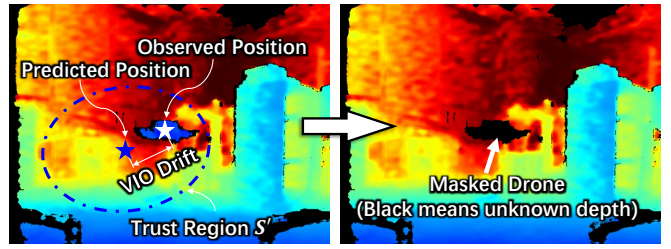

Fig. 7: **Left:** VIO Drift is estimated between a predicted position evaluated from the received trajectory and an observed position in the depth image within the trust region. **Right:** The observed agent is masked out to avoid affecting mapping.
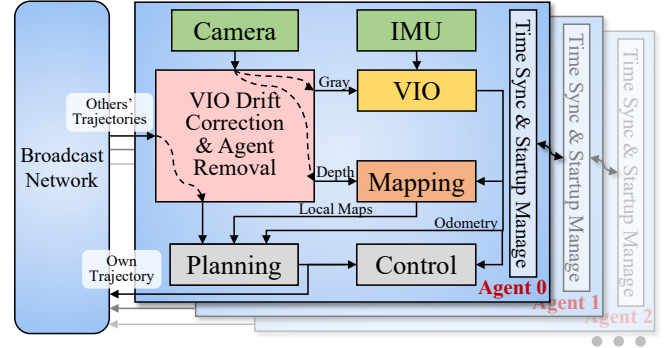

Fig. 8: System Architecture

and treating them as obstacles in the map building are not necessary, even harmful. To eliminate the influence of moving objects, we mask and remove the pixels of agents detected in Sec.IV-B from the depth images, as illustrated in Fig.7. Except that, moving objects covering the majority of the view are interference to VIO. Therefore, agents on gray-scale images are removed as well using the same mask for the corresponding depth images. The criterion for agent detection used here is less strict since false positives are more harmful than false negatives.

V. SYSTEM ARCHITECTURE

The system architecture is depicted in Fig.8, which contains the detailed architecture for a single agent and the multi-agent communication system.

A. Navigation System of An Single Agent

The single agent system, including hardware and software setups is based on our previous work EGO-Planner [3], with an extra module that compensates VIO drift and removes witnessed agents on images. All the components are integrated into a self-assembled 250mm wheelbase quadrotor. For trajectory generation in unknown environments, the local planner is used. Planning is activated when the current trajectory collides with newly discovered obstacles, or the agent is getting close to the end of the current trajectory.

B. Communication Framework

Two networks connect the system, i.e., a broadcast network sharing trajectories, and a chain network to synchronize timestamps and manage the sequential startup.
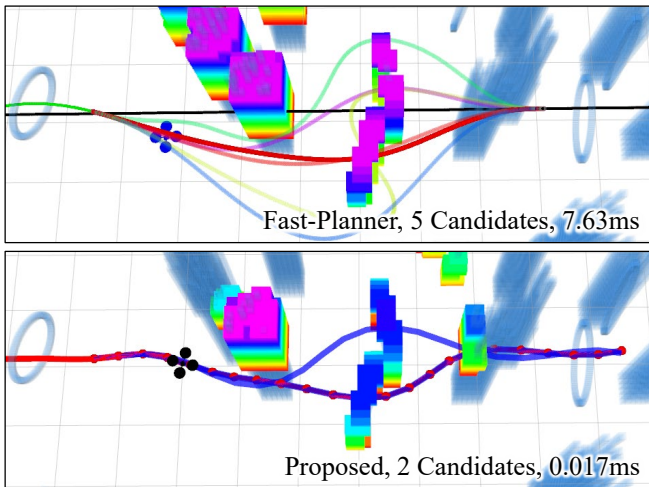
Fig. 9: Comparison of topological path search. The proposed planner finds fewer candidate topologically distinct trajectories but consumes less computation compared to Fast-Planner. Note that the displayed time is only for front-end local minima discovery, while displayed trajectories are optimized for better visualization.

1) Broadcast Network: Once an agent generates a new collision-free trajectory, it is broadcast immediately to all agents. Other agents then receive and store this trajectory used to generate safe trajectories for themselves when necessary. This closed-loop strategy works properly in ideal situations where the connection is stable and the latency is negligible. However, this is not guaranteed in practice. Therefore, we propose two methods to reduce the possibility of collision.

First, one trajectory is broadcast at a given frequency under network capacity. This does not cause computation burden since a typical trajectory containing 3-D waypoints and other parameters are less than 0.5KB in size. By contrast, modern wireless networks such as Bluetooth[2] can reach a speed above 1Mbps. Second, each agent checks collision as soon as a trajectory is received from broadcast network, and if potential collisions are detected, a new collision-free trajectory is generated. This strategy can tackle when multiple agents generate trajectories at a close time without receiving others' trajectories due to latency or packet losses.

In addition, the growth of computational complexity with the increase of agent number is considered. Before planning, each agent compares its current position with received surrounding agents' trajectories, where any trajectory outside the planning range will be ignored.

2) Chain Network: A connection-based stable chain network is used for timestamp synchronization and system startup management. At system startup, agents generate trajectories in a predefined order. Each agent generates its initial trajectory after receiving trajectories from agents with higher priority through the chain network. This strategy avoids chaos caused by simultaneous trajectory generation during system startup since agents have no information of other trajectories at that time.

TABLE I: Comparisons in Obstacle-free Space

Planner	d_{fly}(m)	t_{fly}(s)	Collision	t_{cal}*
DMPC	20.6	**10.8**	**0**	0.668 (s)
RBP	22.9	23.6	**0**	1.18 (s)
ORCA	21.3	28.5	0.375	**0.0225 (ms)**
Proposed	**20.3**	14.3	**0**	0.514 (ms)

VI. BENCHMARK

We present extensive benchmark comparisons about topological planning, swarm planning in empty and cluttered space in simulation on an Intel Core i7-9700KF CPU.

A. Topological Planning

We compare the topological planning performance of the proposed EGO-Swarm with Fast-Planner [15] in terms of candidate trajectory number and computation time for front-end topological path search. As is shown in Fig.9, EGO-Swarm finds fewer candidate trajectories, which means a lower probability of finding the global optimal, but is faster than [15] by two orders of magnitude. Since Fast-Planner finds topologically distinct paths by PRM [25] graph search, path shortening, and path pruning, which are time-consuming but with a higher degree of freedom, compared to the proposed implicit topological path search method.

B. Swarm Planning

1) In Empty Space: We compare the proposed method with DMPC [19], ORCA [16], and RBP [20], in terms of flight distance(d_{fly}), flight time(t_{fly}), collision times per agent, and computation time(t_{cal}). The default parameters, except for maximum velocity and acceleration, are used for each compared method. As depicted in Fig.10, eight agents perform swap transitions on a circle. The results in Tab.I are the average of all agents. t_{cal} is marked by '*', since the computation time we record for offline methods, DMPC and RBP, is the total time to plan all agents' whole trajectory, while for ORCA and EGO-Swarm, it is the local replanning time for each agent.

Tab.I and Fig.10 state that RBP tends to generate safe but conservative trajectories, since the construction of convex *relative safe flight corridor* [20] significantly compresses the solution space. DMPC is designed for distributed deployment. Nevertheless, it requires accurate and high-frequency pose communication, which can not be guaranteed in real-word applications. Efficient rules make ORCA update fast. However, using speed as control commands makes it incompatible with third-order systems, such as quadrotors. The risk of collision also limits its application. By contrast, the proposed method generates the shortest collision-free, non-conservative trajectories with fast computation. Therefore, it enables real-time applications for quadrotors.

2) In Obstacle-rich Environments: We simulate ten drones flying from one side of the map to the other with a speed limit of $2m/s$ and the quadrotor radius of 0.2 meters. Fig.2 is a simulation snapshot of 0.42 obstacles/m^2. Each agent senses the environment independently, and the constructed local maps are displayed in different colors. The

Fig. 10: Planned trajectories by various swarm planners in obstacle-free scenarios.

TABLE II: Performance of EGO-Swarm in Simulation of Various Obstacle Densities

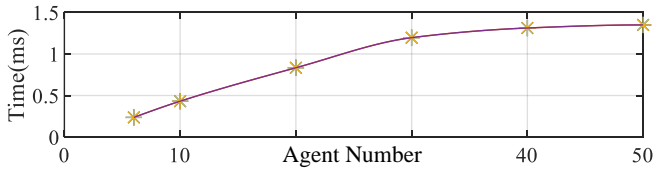| Density(obs/m^2)	d_{fly}(m)	vel.(m/s)	d_{safe}(m)	t_{cal}(ms)
0.42	49.4	1.61	0.208	5.00
0.28	45.4	1.59	0.22	2.87
0.14	43.7	1.57	0.266	1.21
0	42.2	1.55	/	0.49

Fig. 11: The relationship between the computation time and the number of Agents in empty space.

results are summarized in Tab.II, where d_{fly} is the average flight distance, d_{safe} is the closest distance to obstacles during the flight tests. An inverse point-to-point transition is designed to make reciprocal collision avoidance inevitable around the map center. In this scenario, each agent belonging to the swarm manages to plan smooth and safe trajectories.

3) Scalability Analysis: We evaluate the computation performance in a scenario where agents arranged in a straight line fly to random target points 50 meters away. As is depicted in Fig.11, due to the on-demand collision check strategy in Sec.V-B.1, time complexity gradually flattens out with the increase in the number of agents.

VII. REAL-WORLD EXPERIMENTS

A. Indoor

We present several indoor experiments at a speed limit of 1.5m/s, as depicted in Fig.12. The color of the path indicates the approximate flight time of each quadrotor. The top one shows three quadrotors perform a circle swap in empty space, so that reciprocal collision avoidance is necessary. In the middle one, quadrotors manage to pass through a narrow door one after another while avoiding obstacles during the flight. In the bottom figure, we set up a more cluttered environment composed of some vertical and horizontal obstacles. Three quadrotors manage to navigate across this environment.

B. Outdoor

In a forest where trees are spaced about 2 meters apart as shown in Fig.1, three quadrotors start in the forest together and manage to reach the target position outside
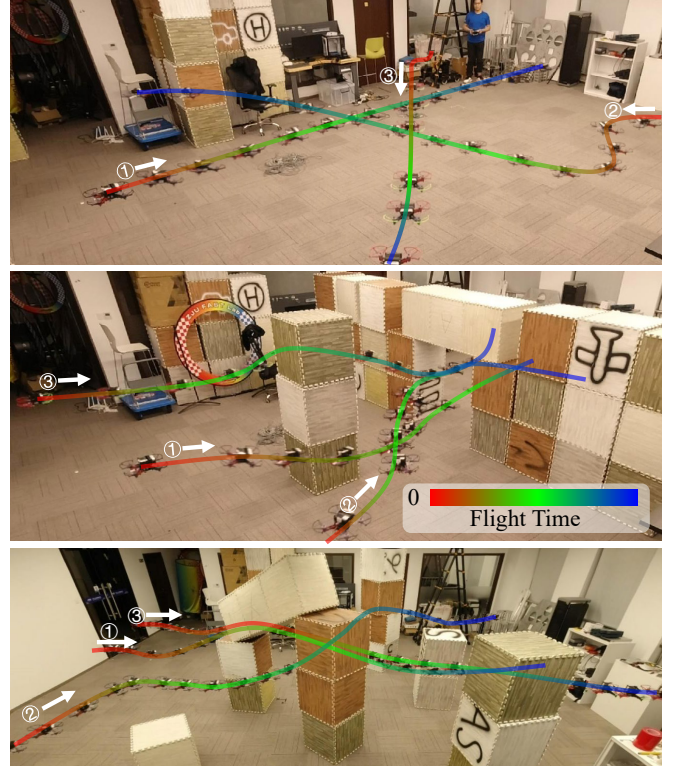

Fig. 12: Indoor Experiments

the forest. The velocity limit is set to 1.5m/s. To further emphasize reciprocal avoidance, we reverse the order of the goal position relative to the start position, making reciprocal avoidance unavoidable, like what we do in Sec.VI-B.2. For more information about the experiments, please refer to the codes and watch our attached video on github[3].

VIII. CONCLUSION AND FUTURE WORK

In this paper, a systematic solution for multi-robot navigation in unknown cluttered environments using only onboard resources is proposed. Benchmark comparisons demonstrate its short computation time and high trajectory quality. Real-world experiments validate its robustness and efficiency.

In the future, we will be committed to achieving precise co-localization and real-world dynamic obstacle avoidance. The main challenge lies in imprecise observations of other drones and unreliable moving object detection and prediction. Finally, we hope to offer the robotics community a complete autonomous navigation solution for quadrotor swarms in complex environments.

[3]https://github.com/ZJU-FAST-Lab/ego-planner-swarm

REFERENCES

[1] K. McGuire, C. De Wagter, K. Tuyls, H. Kappen, and G. C. de Croon, "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment," *Science Robotics*, vol. 4, no. 35, p. eaaw9710, 2019.

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

[3] X. Zhou, Z. Wang, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *arXiv preprint arXiv:2008.08835*, 2020.

[4] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom. (ICRA)*, May 2009, pp. 489–494.

[5] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 4569–4574.

[6] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.(IROS)*, Daejeon, Korea, Oct. 2016, pp. 5332–5339.

[7] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for mavs using uniform b-splines and a 3d circular buffer," in *iros*, Sep. 2017, pp. 215–222.

[8] W. Ding, W. Gao, K. Wang, and S. Shen, "An efficient b-spline-based kinodynamic replanning framework for quadrotors," *IEEE Transactions on Robotics*, vol. 35, no. 6, pp. 1287–1306, 2019.

[9] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[10] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *arXiv preprint arXiv:2007.03465*, 2020.

[11] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based path planning with homotopy class constraints." in *AAAI*, 2010, pp. 1230–1237.

[12] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[13] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*. VDE, 2012, pp. 1–6.

[14] L. Jaillet and T. Siméon, "Path deformation roadmaps: Compact graphs with useful cycles for motion planning," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1175–1188, 2008.

[15] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time uav replanning using guided gradient-based optimization and topological paths," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1208–1214.

[16] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[17] J. Van Den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom. (ICRA)*. IEEE, 2011, pp. 3475–3482.

[18] S. H. Arul and D. Manocha, "Dcad: Decentralized collision avoidance with dynamics constraints for agile quadrotor swarms," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1191–1198, 2020.

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

[20] J. Park, J. Kim, I. Jang, and H. J. Kim, "Efficient multi-agent trajectory planning with feasibility guarantee using relative bernstein polynomial," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom. (ICRA)*. IEEE, 2020, pp. 434–440.

[21] Z. Liu, B. Chen, H. Zhou, G. Koushik, M. Hebert, and D. Zhao, "Mapper: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments," *arXiv preprint arXiv:2007.15724*, 2020.

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

[23] X. Hao, W. Luqi, Z. Yichen, Q. Kejie, and S. Shaojie, "Decentralized visual-inertial-uwb fusion for relative state estimation of aerial swarm," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

[24] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom. (ICRA)*, Shanghai, China, May 2011, pp. 2520–2525.

[25] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.