RACER: Rapid Collaborative Exploration With a Decentralized Multi-UAV System

Boyu Zhou, Hao Xu, and Shaojie Shen, *Member, IEEE*

***Abstract*—Although the use of multiple unmanned aerial vehicles (UAVs) has great potential for fast autonomous exploration, it has received far too little attention. In this article, we present a RApid Collaborative ExploRation (RACER) approach using a fleet of decentralized UAVs. To effectively dispatch the UAVs, a pairwise interaction based on an online hgrid space decomposition is used. It ensures that all UAVs simultaneously explore distinct regions, using only asynchronous and limited communication. Furthermore, we optimize the coverage paths of unknown space and balance the workloads partitioned to each UAV with a capacitated vehicle routing problem formulation. Given the task allocation, each UAV constantly updates the coverage path and incrementally extracts crucial information to support the exploration planning. A hierarchical planner finds exploration paths, refines local viewpoints, and generates minimum-time trajectories in sequence to explore the unknown space agilely and safely. The proposed approach is evaluated extensively, showing high exploration efficiency, scalability, and robustness to limited communication. Furthermore, for the first time, we achieve fully decentralized collaborative exploration with multiple UAVs in the real world. We will release our implementation as an open-source package.**

***Index Terms*—Aerial system, aerial systems, applications, cooperating robots, perception and autonomy.**

I. Introduction

AUTONOMOUS exploration, which utilizes autonomous vehicles to map unknown environments, is a fundamental problem for various robotic applications like inspection, search-and-rescue, etc. Recently, a considerable amount of literature studies autonomous exploration with unmanned aerial vehicles (UAVs), especially quadrotors. It is demonstrated that UAVs are particularly suited to exploring complex environments efficiently, thanks to their agility and flexibility.

Despite the significant progress, most works solely focus on exploration with a single UAV, while little attention has been paid to multi-UAV systems. However, using a fleet of UAVs has incredible potential, since it not only enables faster accomplishment of exploration, but also is more fault tolerant than a single UAV. In this work, we bridge this gap by introducing a RApid Collaborative ExploRation (RACER)[1] approach using a team of decentralized UAVs.

Up to now, developing a robust, flexible, and efficient multi-UAV exploration system has been fraught with difficulties. First of all, approaches that coordinate multiple robots typically rely on a central controller or require reliable communication among the robots. Unfortunately, unreliable and range-limited communication is common in practice, making the coordination vulnerable and less effective. To improve the flexibility and robustness of the system, a decentralized coordination approach with fewer communication requirements is desired. Second, many multirobot exploration approaches solely consider the allocation of frontiers or viewpoints. Because the actual regions explored by each UAV are not accounted for, the strategies often result in interference among robots and inequitable workload allocation. Also, the global coverage routes of the unknown space are not taken into account, so the UAVs may redundantly revisit the same regions, significantly reducing the overall efficiency. To fully realize the system's potential, a more sophisticated collaboration strategy is required. Lastly, each member of the system should be able to fully utilize its capability to fulfill the task. For this purpose, it is crucial that each UAV plans motions quickly in response to environmental changes, allowing itself to navigate and collect information agilely, while avoiding collision with previously unknown obstacles and other agents in the system.

In this article, we present a systematic decentralized coordination and planning approach that enables rapid collaborative exploration, which takes the above-mentioned difficulties into consideration, as shown in Fig. 1. To effectively coordinate the quadrotors, the entire unknown space is constantly subdivided into hgrid online and distributed among the quadrotors by a pairwise interaction. It only requires asynchronous and unreliable communication between pairs of nearby quadrotors, and ensures simultaneous exploration of distinct regions without causing interference among the quadrotors. Moreover, a capacitated vehicle routing problem (CVRP) formulation is proposed for more efficient cooperation. It minimizes the lengths of multiple global coverage paths (CPs) and balances the workload partitioned to each quadrotor, effectively preventing repeated exploration and inequitable workload allocation. Given the allocation of

Manuscript received 3 June 2022; revised 23 October 2022; accepted 27 December 2022. Date of publication 6 February 2023; date of current version 7 June 2023. This work was supported in part by the HKUST Institutional Fund under Grant R9341 and in part by DDF. This paper was recommended for publication by Associate Editor L. Carlone and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. *(Corresponding author: Boyu Zhou.)*

Boyu Zhou is with the School of Artificial Intelligence, Sun Yat-sen University, Zhuhai 510275, China (e-mail: zhouby23@mail.sysu.edu.cn).

Hao Xu and Shaojie Shen are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong SAR, China (e-mail: hxubc@connect.ust.hk; eeshaojie@ust.hk).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2023.3236945.

Digital Object Identifier 10.1109/TRO.2023.3236945

[1]To be released at https://github.com/HKUST-Aerial-Robotics/FUEL

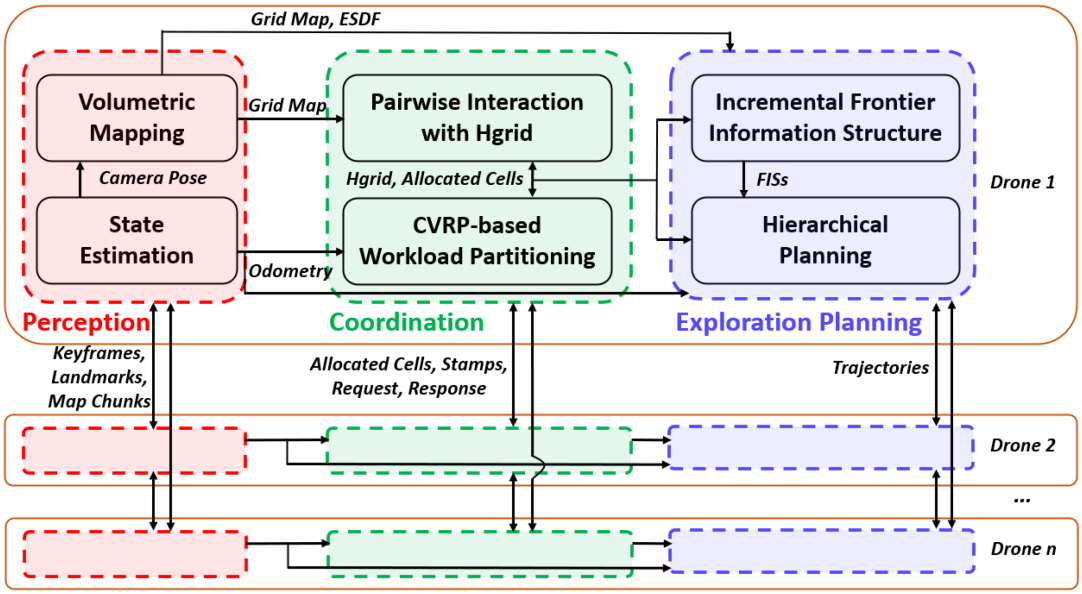


Fig. 1. Overview of our multiquadrotor autonomous exploration system, including the perception, coordination, and exploration planning modules.

unexplored regions, each quadrotor continuously updates its CP, providing an efficient global route to cover the entire unknown space. Under the guidance of the CP, a hierarchical planner finds local paths, refines viewpoints, and generates minimum-time trajectories sequentially to enable agile and safe exploration. The overall approach is computationally cheap and scalable, which enables the team to react quickly to environmental changes, leading to consistently fast exploration.

We evaluate the proposed approach comprehensively by simulations and challenging real-world experiments. Our approach is shown to complete exploration significantly faster than existing centralized and decentralized approaches, has more consistent performance under restricted communication, and is computationally efficient and scalable for a large team. What's more, we integrate our approach with a decentralized state estimation module and conduct fully autonomous exploration in complex environments. To the best of our knowledge, we are the first to achieve such exploration capability in real world, where the quadrotors cooperate in a fully decentralized fashion and build a dense map of the environment completely and quickly. We will release the source code to benefit the community. In summary, the contributions of this article are as follows.

1) A pairwise interaction based on an online hgrid decomposition of the unexplored space, which ensures that the quadrotors explore distinct regions jointly using only asynchronous and restricted communication.
2) A CVRP formulation that minimizes the lengths of global CPs and balances the workload assigned to each quadrotor. It further enhances the cooperation of the quadrotor team.
3) A hierarchical exploration planner extended from [1] that incorporates the guidance of global CPs, which allows higher exploration efficiency. Reciprocal collision avoidance is integrated into the planner.
4) We integrate the proposed approach with multirobot state estimation and conduct fully autonomous exploration experiments in challenging real-world environments. The source code of our system will be released.

In what follows, we review related works in Section II and overview our system in Section III. The hgrid-based pairwise interaction and the CVRP formulation for balanced workload allocation are detailed in Sections IV-B and V-A, respectively. The hierarchical exploration planning is presented in Section VI. Benchmark and experimental results are given in Section VIII. Finally, Section IX concludes this article.

II. Related Work

A. Autonomous Exploration

Robotic exploration, which maps unknown environments with mobile robots, has been investigated over the years. Some works concentrate on quick coverage [2], [3], while others place more emphasis on precise reconstruction [4], [5]. One type of classic approaches is frontier-based approaches, which were presented first in [6] and assessed more comprehensively later in [7]. Shen et al. [8] adapted a stochastic differential equation-based strategy, seeking frontiers in 3-D environments. Different from the original method [6] that chooses the closest frontier as the next target, Cieslewski et al. [2] chose the frontier inside the field of view (FOV) that minimizes the change of velocity. This strategy is beneficial to maintain a high flight speed and demonstrates higher efficiency than that presented in [6]. Deng et al. [9] presented a differentiable formulation, which is useful for gradient-based path optimization. Frontier-based methods have the advantage of simplicity, as many of them simply consider a motion-related objective to generate paths. Some other works also introduce information gain to evaluate the utility of candidate actions [10], [11].

Sampling-based approaches, on the other hand, randomly produce candidate viewpoints and evaluate their information gain to explore the space. As these approaches typically quantify the information gains for numerous candidate viewpoints, the

computational burden is higher. Sampling-based approaches are closely related to the next best view (NBV) [12], where viewpoints are computed repeatedly to model a scene completely. Bircher et al. [13] first introduced the concept of NBV in 3-D exploration. It grows rapidly-exploring random trees (RRTs) within free space and executes the most informative edge in a receding horizon fashion. Later, uncertainty of localization [14], visual importance [15], and inspection [16] were considered under the framework in [13]. To reuse past information, roadmaps capturing navigation history were built in [17] and [18], while a single tree was persistently refined by employing a rewiring scheme motivated by RRT* [4]. To realize faster flight, Dharmadhikari et al. [3] proposed to sample dynamically feasible motion primitives to enable a higher flight speed. In [19], a 2-D manifold was dynamically adapted according to the map built so far, guiding the sampling of viewpoints. It speeds up the computation of sampling but at the cost of suboptimality in the case of a complex structure.

Approaches combining the frontier-based and sampling-based approaches were presented in [1], [5], [20], [21], [22], [23] and [24]. Charrow et al. [20] and Selin et al. [21] proposed to generate global paths based on frontiers and sample paths to explore local regions. Meng et al. [22] discussed a method that samples viewpoints around frontiers and finds the global shortest tour passing through them. Song et al. [5] found inspection paths completely covering frontiers by a sampling-based algorithm. A two-level framework [23] samples viewpoints for incomplete surfaces around the robots and computes the shortest visiting path, which is connected to a coarse global path. Comparing to sampling-based methods, these methods are computationally cheaper as they require much fewer viewpoint sampling, thanks to the targeted sampling around frontiers. Our previous work [1] presented a more computationally efficient method that detects frontier and samples viewpoints around frontiers in an incremental fashion.

Given the candidate targets (frontiers or sampled viewpoints), many existing methods adopt a greedy strategy to make decisions. For example, the closest frontier [6], [8] is selected or the immediate information gain is maximized [3], [4], [13], [17]. The greedy strategy ignores an efficient global route, resulting in unnecessary revisiting and reducing the overall efficiency. This issue was partially addressed in [22] and our previous work [1], which formulate a Traveling Salesman Problem (TSP) to compute the shortest path visiting candidate viewpoints. However, they only considered paths for sampled viewpoints and still lack a global route that cover the entire unexplored region. In this work, we extend [1] to incorporate an efficient CP of the entire unknown space, guiding each quadrotor to explore different unknown regions in a more reasonable order.

B. Coordinating Multiple Robots

A multirobot system has obvious advantages over a single robot, like covering environments faster and providing robustness to single point failures. Central to multirobot exploration is the appropriate allocation of tasks, so that conflicting targets can be avoided and each robot explores different regions simultaneously. One common method is communicating with all robots and allocating tasks with a centralized server [25], [26], [27], [28], [29]. Among the approaches, the most straightforward one is iterative assignment [25], [26], in which the algorithm greedily selects pairs of robot and target based on distance and information gain. However, such a strategy only allocates a single target to each robot, which may not be effective when multiple robots are assigned to nearby targets. To overcome this limitation, approaches based on segmentation [27], [30] and multiple Travelling Salesman Problem (mTSP) [28], [29], [31] are proposed. Wurm et al. [27] divided the already explored area into segments using a Voronoi graph. By assigning robots to different segments instead of single targets, the robots are distributed more evenly over the environments. Similarly, a breath-first-search (BFS)-like clustering algorithm groups the decomposed areas and assign them to robots [30]. Faigl et al. [28] and Dong et al. [29] formulated the coordination as mTSP. Since the problem is NP-hard, divide-and-conquer schemes are proposed to find the approximate solutions. Faigl et al. [28] grouped all targets into clusters by a variant of the K-means algorithm and assigned the clusters to robots. Dong et al. [29] solved a discrete optimal transport problem to find a promising assignment of all targets to robots, while the optimal path of each robot was determined by TSP. Hardouin et al. [31] adopted a TSP-greedy allocation algorithm to greedily assign a cluster of viewpoint to each robot. However, they also considered a centralized architecture where communication is assumed to be perfect.

One major issue of centralized coordination is requiring all robots to maintain communication with the server. However, the requirement is usually impractical, for example, robots may get far from the server while exploring large-scale environments, or signals may be frequently occluded in cluttered environments. Moreover, robots are unable to continue exploration if the server fails. In contrast, decentralized coordination [32], [33], [34], [35], [36], [37], [38] is more robust to communication loss and failure, since each agent in the system is able to operate independently. The first decentralized multirobot exploration was developed in [32], in which all robots share map information and move to the closest frontiers. Although being simple and completely distributed, the coordination is not effective enough since more than one robot may move to the same place. An ant-inspired algorithm was proposed in [39], where robots can mark explored parts of the environment to prevent teammates from visiting the same place. However, an implementation of this method in real world is a challenge. Another different means of decentralized coordination utilizes auction-based mechanisms [33], [34], [35]. Zlot et al. [33] and Smith et al. [34] settled conflicts among robots using a distributed single-bid local auction. To jointly consider the allocation of multiple targets, Berhault et al. [35] formulated a combinatorial auction. Apart from auction-based approaches, Corah et al. [36] extended the greedy assignment method to a distributed version; however, the procedure typically requires several rounds of communication among robots. Corah et al. [40] also discussed different objective functions and their applicability for the distributed assignment algorithm in [36]. Yu et al. [37] utilized a multirobot multitarget potential field to dispatch robots to different frontiers. Best et al. [41] proposed a decentralized Monte Carlo tree search method for

multirobot active perception tasks, in which robots periodically exchange their search tree and update the joint distribution using a distributed optimization. Although these methods can function without global communication, they require simultaneous communication among neighboring robots to achieve a higher level of coordination. Under unstable networks, the coordination deteriorates considerably due to the difficulty in exchanging messages at the same time. To resolve this issue, it is important to simplify the requirement for communication structure, reducing the number of negotiating robots. The most relevant approach to us is the pairwise optimization scheme presented by Klodt et al. [38], where targets are repeatedly reallocated among robots by pairwise interactions. The method evolved from [42], which was originally designed to solve a robot coverage control problem.

For the abovementioned decentralized methods, communication among robots is episodic and opportunistic, since connections are not planned in advance. In some tasks, periodic meetings among robots and the base station are required, which allows enforcing information update at times. To this end, several methods divide robots into different roles, where some robots explore new regions while the others are in charge of delivering information [43], [44], [45]. The two types of robots are coordinated through appropriately selecting the rendezvous points, where they are assumed to connect based on specific communication models [46]. Different from them, Gao et al. [47] presented a method that considers robots equally and allows periodic meetings. Another more restrictive class of works enforce a continuous connection among all robots. In [48], robots explore a scene with hard constraints imposing line-of-sight communication. Rathnam et al. [49] discussed a distributed search-based method where robot configurations disallowing continuous connection are penalized. In [50], the market-based coordination was extended to bias robots to move toward frontiers with a higher probability of communication. Jensen et al. [51], [52] proposed systems allowing robots to reconnect after accidental disconnection during exploration. With a periodic or continuous connection, knowledge can be exchanged at a higher frequency. However, as a side effect, the exploration process may be considerably constrained.

In this work, we consider the problem of a team of robots exploring unknown scenes as quickly as possible, without the need of real-time data streaming with the base station. Our coordination method is inspired by [38]. However, our method differs by adopting hgrid as the elementary task unit to allow more effective task allocation. Meanwhile, instead of approximating each pair's traveling distance and using a local hill descent to improve the allocation locally [38], we find the optimal allocation between two robots by minimizing the exact traveling distance. We also design a request-response scheme to address the issues of inequitable interaction chance and conflicting interactions.

C. Multirobot Exploration Systems

Several work studied multirobot system for autonomous exploration, focusing on different aspects of the system. Motivated by the DARPA Subterranean Challenge, where teams of robots search for objects of interest in underground environments, researchers presented systems toward subterranean exploration using legged, wheeled, and flying robots [53], [54], [55], [56], [57], [58]. Those robots are typically equipped with various sensors and high-gain communications antennas, as well as localization, mapping, and path planning capabilities, allowing them to search for objects in complex, large-scale subterranean environments. These work specialized in the integration of hardware and algorithmic components, as well as subterranean engineering adaptations for system robustness. However, the systems rely more or less on a central computer to accomplish tasks. Meanwhile, coordination among robots is paid with less attention and simple strategies like greedy assignment [53] or auction [55], [57] are adopted. Petráček et al. [58] also discussed a homing strategy such that a group of robots is able to build up a communication tree with the base station. Apart from them, some researchers investigated the mapping [59], [60] and collaborative communication [61] of the system. Corah et al. [59] employed a Gaussian mixture model for global mapping, which maintains a small memory footprint and enables a lower volume of communication. Tian et al. [60] presented a cycle consistency-based method for robust data association, improving the precision of collaborative localization and mapping. Cesare et al. [61] proposed a collaboration scheme that allows robots with low battery to take on the role of a relay to improve communication between team members. In [62], a swarm gradient bug algorithm for autonomous navigation of tiny flying robots was proposed. Each robot uses wall following and received signal strength to avoid collision. It allows lightweight robots to explore unknown environments at the cost of lower efficiency and unavailability of accurate online mapping. Among these works, none of them studied decentralized algorithms that allow consistent effective collaboration even under intermittent communication. Also, none of the works minimizes the overall CP length and balances the workload to exploit the system's full capability.

D. Quadrotor Motion Planning

Relevant to exploration is local trajectory replanning, where gradient-based methods gain wide popularity [63], [64], [65], [66], [67], [68], [69]. The methods were revived by Ratliff et al. [70] and extended to polynomial trajectories [63], [64] and B-spline trajectories [65]. More recently, Zhou et al. [66], [67], [68] further exploited the properties of B-splines, and used topological guiding paths and active perception to achieve aggressive flight in complex scenes. The computation time of trajectory optimization was further reduced by eliminating the need of distance field [69].

To avoid collision between robots, decentralized approaches were studied in [71], [72], [73], [74], [75], and [76]. Velocity obstacles are leveraged to avoid collision of robots with different constraints [71], [72], [73]. Liu et al. [74] proposed an asynchronous strategy to avoid obstacles and other vehicles. Park et al. [75] employed safe flight corridor and Bernstein polynomial to generate feasible trajectories. Zhou et al. [76] extended the gradient-based replanning method [69] to a team of

quadrotors. Our trajectory planning in this work is based on the method in [1] and [66], which generates safe and minimum-time trajectories for fast exploration. We extend it to avoid interdrone collision using asynchronous communication among quadrotors.

III. System Overview

The overall multiquadrotor exploration system is illustrated in Fig. 1. As most recent works [2], [4], [13] do, we aim to build a complete volumetric map of a designated space. Each quadrotor in the system performs decentralized state estimation to localize itself and other quadrotors. In experiments, the state estimation approach in [77] is integrated into our system. Given the estimated relative pose, each quadrotor exchanges map information frequently with nearby ones when communication is available, to allow more informed decision making. The implementation details of state estimation and mapping are presented in Section VII.

The coordination of the quadrotor team relies on the online hgrid decomposition and pairwise interaction. As each quadrotor explores and collects information, the entire unknown space, which is bounded by a predefined boundary as most methods, is constantly decomposed into hgrid containing cells of varying sizes, which serve as elementary task units to be allocated among the quadrotors. Scheduled by a request-response scheme, pairs of quadrotors communicate and update the ownership of the cells by pairwise interaction (see Section IV-B). The cells are partitioned by a CVRP formulation that minimizes the lengths of the CPs and balances the volume of an unknown space to be explored, as is presented in Section V-A.

Given the allocated cells, each quadrotor independently plans paths and trajectories to explore the designated space while avoiding other quadrotors and obstacles. In the assigned regions, it computes the shortest CPs and updates the FISs incrementally as the map changes. The CPs serve as high-level guidance of the exploration, along which optimal local paths passing through a local set of frontier clusters are found. Finally, minimum-time trajectories are generated to cover the viewpoints and avoid collision (see Section VI). The process continues until no frontier is discovered inside the allocated cells and no other cells are assigned to the quadrotor.

IV. Hgrid-Based Pairwise Interaction

We utilize a pairwise interaction to distribute tasks, which only requires intermittent communication with nearby quadrotors. Unlike most approaches that allocate frontiers and viewpoints, our task representation is based on an online hgrid decomposition of the unknown space, which ensures joint exploration of distinct regions without causing interference among quadrotors.

A. Hgrid Decomposition

We constantly decompose the entire unknown space online into a collection of disjoint regions, representing elementary task units to be allocated among quadrotors. The considered decomposition here is hgrid, which is a hierarchical decomposition consisting of cells from coarse to fine resolutions. This structure

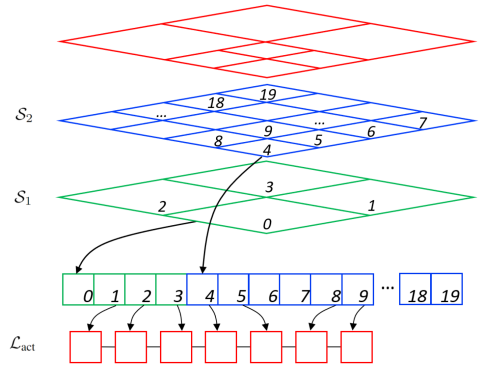


Fig. 2. Example of a 2-D hgrid decomposition with two levels, and its implementation using a single array. For convenience, we illustrate 2-D hgrid here, but in implementation, 3-D hgrid is used. The current decomposition of the environment (red) is recorded by list $\mathcal{L}_{\text{act}}$.

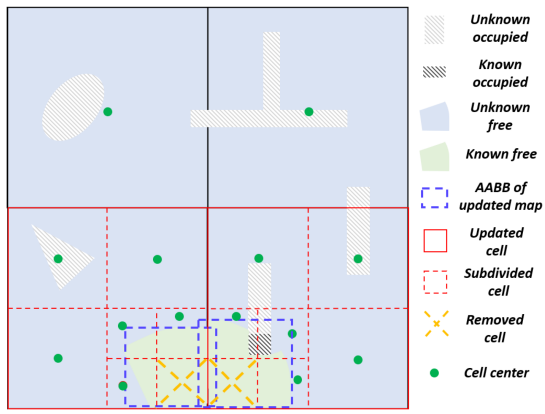


Fig. 3. Illustration of the procedure to update a three-level hgrid. Cells that overlap with any AABBs in $\mathcal{B}_m$ are updated and subdivided into finer cells if necessary. Cells that have been covered completely are removed from $\mathcal{L}_{\text{act}}$.

was originally adopted for broad-phase collision detection in [78], reducing the number of expensive narrow-phase collision detection between pairs of objects. In contrast, we employ hgrid as an efficient representation of the unexplored space, capturing the changing unknown regions in different levels of detail, and dividing them into elementary task units to facilitate multirobot collaboration. For hgrid, any cell in level l is uniformly subdivided into eight subcells (four in 2-D) in level $l+1$. The varying cell sizes of hgrid provide sufficient flexibility to represent the unknown regions. Examples of hgrid are shown in Figs. 2 and 3.

Given the online built map, each quadrotor maintains a hgrid decomposition located in the same world coordinate frame. The establishment of a common coordinate frame is detailed in Section VII. The hgrid contains a total number of L levels $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_L\}$, where $\mathcal{S}_1$ has the coarsest cells and $\mathcal{S}_L$ has the finest ones. For each cell, it records the number of unknown voxels contained by it and the centroid of the unknown voxels. To store the entire hgrid cells with a single array, we use a mapping similar to that of a standard 3-D grid. Let the space be discretized into $h_l = h_{l,x} \times h_{l,y} \times h_{l,z}$ cells in $\mathcal{S}_l$. For a cell whose integer

coordinate is (i_l, j_l, k_l), its index is determined by

$$\gamma(i_l, j_l, k_l) = i_l h_{l,y} h_{l,z} + j_l h_{l,z} + k_l + \sum_{m=1}^{l} h_{m-1}, \quad h_0 = 0. \tag{1}$$

Geometrically, hgrid is similar to an octree-based decomposition [79], [80], but it is implemented using arrays, allowing $O(1)$ retrieval of any cell. Moreover, the linearized data structure adopted in our implementation increases the level of data locality, which improves the CPU data cache hit rate and thus overall performance [78].

B. Online Hgrid Updating

To represent the decomposition of the unknown regions that will be covered by the quadrotors, a list $\mathcal{L}_{\text{act}}$ is utilized, as is illustrated in Fig. 2. During the exploration, it stores a subset of all hgrid cells that have not been completely explored, have not yet been subdivided into finer ones, and are reachable by the sensor. Meanwhile, to facilitate the update of hgrid, we record $\mathcal{B}_m = \{B_1, B_2, \ldots, B_M\}$, which are the axis-aligned bounding boxes (AABBs) of updated map regions, consisting of the region explored by a quadrotor itself and the ones shared by nearby quadrotors.

The online hgrid decomposition of an unknown space is illustrated in Fig. 3 and Algorithm 1. Initially, the environment is completely unknown and is decomposed into the coarsest cells, i.e., $\mathcal{L}_{\text{act}}$ only contains cells in $\mathcal{S}_1$. As the quadrotors build and share map, some regions become partially unknown and the associated cells are further subdivided to represent the yet unknown regions with finer resolutions. The update of hgrid proceeds by iterating each cell s_k in $\mathcal{L}_{\text{act}}$ and screening the ones that have overlap with any $B_m \in \mathcal{B}_m$ (Line 2). The centroid and the unknown voxel number of them are recomputed (Line 4). Then, if a certain ratio α_u of voxels contained by s_k are already known, and s_k is not at the finest level, it is subdivided uniformly (Lines 5–9). Correspondingly, it means removing s_k from $\mathcal{L}_{\text{act}}$ and appending the subdivided cells in the next level to $\mathcal{L}_{\text{act}}$. Meanwhile, if s_k is at the finest level and contains only a tiny number of unknown voxels less than δ_u, it is removed from $\mathcal{L}_{\text{act}}$ (Lines 11–12). Finally, cells whose centroids are unreachable from the current position of the quadrotor are also removed (Lines 13–14), which eliminates regions that cannot be covered by sensors, like hollow spaces in thick walls.

Algorithm 1: Online Hgrid Decomposition.

```
   // L_act = S_1 at the beginning of exploration
1  for s_k ∈ L_act do
2      if ¬ HaveOverlap(s_k, B_m) then
3          continue
4      UpdateCellInfo(s_k)
5      if KnownRatio(s_k) ≥ α_u ∧ Level(s_k) ≠ L then
6          S_div ← FindSubdivided(s_k)
7          L_act.erase(s_k)
8          for s_j ∈ S_div do
9              L_act.append(s_j)
10     if UnknownNum(s_k) < δ_u ∧ Level(s_k) = L then
11         L_act.erase(s_k)
12     if ¬ Reachable(q_i) then
13         L_act.erase(s_k)
```

C. Pairwise Interaction With Hgrid

The pairwise interaction is inspired by [38], in which the problem of task allocation among all quadrotors is decomposed into subproblems that can be solved by pairs of quadrotors. When two robots interact, a local hill descent based on their previous allocation of frontiers is performed by moving one frontier between them, which reduces an approximated path length locally. In this way, tasks are allocated in a distributed fashion, while communication requirements are reduced significantly as only two quadrotors have to communicate at a time.

Our pairwise interaction takes some steps further from [38]. Considering the hgrid cells for any interacting quadrotors q_i and q_j, it finds the optimal allocation between them by minimizing the exact path length, instead of greedily reducing the approximated length. Specifically, let $\mathcal{T}_i$ and $\mathcal{T}_j$ be the two sets of cells before interaction, the target is to find two new sets $\mathcal{T}_i'$ and $\mathcal{T}_j'$, such that $\mathcal{T}_i \cup \mathcal{T}_j = \mathcal{T}_i' \cup \mathcal{T}_i'$ and $\mathcal{T}_i'$, $\mathcal{T}_j'$ minimize the overall length of CPs.

Without the schedule of a central computer, the quadrotors may interact with each other in an uneven manner, which is detrimental to effectively distributing the task. Furthermore, a quadrotor may happen to interact with multiple other quadrotors at the same time, producing several conflicting allocation results. To address the two issues, a request-response scheme is designed, as shown in Algorithm 2. As a preliminary, several types of information are broadcast periodically so that nearby quadrotors maintain a common knowledge about them. Take quadrotor q_i as an example, it broadcasts the hgrid cells that belong to it and the timestamp of the latest attempt to perform pairwise interaction, which are denoted as $\mathcal{T}_i$ and $q_i.T_{\text{att}}$ respectively. Also, q_i records $\{T_{\text{succ},j} | j \in [1, N_q] \setminus i\}$, the timestamps of the latest successful interaction with the other $N_q - 1$ quadrotors.

The interaction starts by finding nearby quadrotors that have communication with q_i (Lines 4–5). Those that have already attempted another interaction within a short period of time ε_{att} will not be considered (Lines 6–7), preventing a quadrotor to interact with several ones simultaneously, which typically leads to conflicting interaction results. Among them, we select q_η, the one that has not experienced successful interaction with q_i for the longest time (Lines 8–9), which ensures that other quadrotors have similar opportunities to interact with q_i. Then, an appropriate partitioning is found and a request message containing the result is sent to q_η (Lines 10–11). The specific partitioning approach is detailed in Section V-A. Finally, it updates T_{att} and waits for a response in a duration of ε_{att} (Lines 12–13). Only if a *succ* response is received, the partitioning result is updated by both q_i and q_η (Lines 14–24); otherwise, the interaction is ignored and new ones will be tried later. Note that a double check of recent interaction attempt is necessary (Line 19). Sometimes q_η may just request an interaction with another quadrotor q_j

Algorithm 2: Pairwise Interaction With Hgrid.

Function RequestInteraction ( ):
- $T_n \leftarrow$ **TimeNow**()
- $T_{\max} \leftarrow 0$
- $\mathcal{Q}_n \leftarrow$ **NearbyQuadrotors**()
- **for** $q_k \in \mathcal{Q}_n$ **do**
  - **if** $T_n - q_k.T_{\text{att}} \leq \varepsilon_{\text{att}}$ **then**
    - **continue**
  - **if** $T_n - T_{\text{suc},k} > T_{\max}$ **then**
    - $T_{\max} \leftarrow T_n - T_{\text{succ},k},\ \eta \leftarrow k$
- $\mathcal{T}_i', \mathcal{T}_\eta' \leftarrow$ **FindPartitioning**$(\mathcal{T}_i, \mathcal{T}_\eta)$
- **Request**$(\mathcal{T}_i', \mathcal{T}_\eta', T_n, i, \eta)$
- $T_{\text{att}} \leftarrow T_n$
- *res* $\leftarrow$ **WaitResponse**$(\varepsilon_{\text{att}})$
- **if** *res* $==$ *succ* **then**
  - $T_{\text{succ},i} \leftarrow T_n$
  - **UpdatePartitioning**$(\mathcal{T}_i', \mathcal{T}_\eta', i, \eta)$

Function RespondInteraction $(\mathcal{T}_i', \mathcal{T}_\eta', T_n, i, \eta)$**:**
- **if** $T_n - T_{\text{att}} \leq \varepsilon_{\text{att}}$ **then**
  - **Respond**(*fail*)
  - **return**
- **Respond**(*succ*)
- $T_{\text{att}} \leftarrow T_n,\ T_{\text{succ},i} \leftarrow T_n$
- **UpdatePartitioning**$(\mathcal{T}_i', \mathcal{T}_\eta', i, \eta)$

and broadcast the updated timestamp. Due to communication latency, this timestamp may be unavailable to q_i when it attempts to interact with q_η. In this case, the double check ensures that q_η does not interact with q_i and q_j at the same time, avoiding conflicting task allocation.

Theoretically, the pairwise interaction may find suboptimal allocation compared with a centralized counterpart. However, with a single round of interaction among the robots, it quickly reaches a promising allocation, as is shown in our quantitative study in Section VIII-D. Also, note that the pairwise interaction does not necessarily reduce the communication events. Instead, it simplifies the communication structure, requiring only two robots to negotiate at the same time, which is more resilient to unstable communication.

V. CVRP-Based Workload Partitioning

To appropriately partition the hgrid cells belonging to a pair of interacting quadrotors (see Algorithm 2), a CVRP formulation is devised. The key idea is minimizing the overall lengths of the quadrotors' CPs and balancing the amount of unknown space allocated to them, making them collaborate more effectively.

A. CVRP Formulation

The optimal CPs of the hgrid cells can be found by vehicle routing problem (VRP) [81], which generalizes the TSP to multiple vehicles. In our setting, there are two vehicles corresponding to the pair of interacting quadrotors q_1, q_2. Different from the standard VRP, where there is a central depot and the vehicles' routes form closed loops, we require open paths starting from the quadrotors' positions and passing the hgrid cells. We reduce this variant into an asymmetric VRP by introducing a virtual depot and properly designing the connection costs.

1) Cost Matrix: Suppose there are N_h hgrid cells, the asymmetric VRP involves $N_h + 3$ nodes, in which there are N_h nodes for the hgrid cells, two nodes for q_1 and q_2, and one for the virtual depot. The associated cost matrix $\mathbf{C}_{\text{avrp}} \in \mathbb{R}^{(N_h+3)\times(N_h+3)}$ has the form:

$$\mathbf{C}_{\text{avrp}} = \begin{bmatrix} 0 & -\mathbf{M}_{\text{inf}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}_{q,h} \\ \mathbf{0} & \mathbf{C}_{q,h}^{\mathrm{T}} & \mathbf{C}_h \end{bmatrix}. \tag{2}$$

$\mathbf{C}_h$ is an $N_h \times N_h$ block that corresponds to the connection costs among all hgrid cells. Since we aim to find the shortest CPs, path lengths between pairs of cells are used as the connection costs:

$$\begin{aligned} \mathbf{C}_h(h_1, h_2) &= \mathbf{C}_h(h_2, h_1) \\ &= \mathbf{Len}\left[P\left(\mathbf{c}_{h_1}, \mathbf{c}_{h_2}\right)\right],\ h_1, h_2 \in [1, N_h]. \end{aligned} \tag{3}$$

Here, $\mathbf{c}_{h_1}$ and $\mathbf{c}_{h_2}$ represents the centroid of unknown voxels in the h_1th and h_2th cells, respectively, while $P(\mathbf{c}_{h_1}, \mathbf{c}_{h_2})$ is the collision-free path, which can be found by standard path searching algorithms.

The costs between quadrotors and hgrid cells are accounted for by $\mathbf{C}_{q,h}$, which is a $2 \times N_h$ block. The connection cost is similar to (3) except for a consistency term:

$$\begin{aligned} \mathbf{C}_{q,h}(i,h) &= \mathbf{len}\left[P(\mathbf{p}_{q_i}, \mathbf{c}_h)\right] + c_{\text{con}}(i,h) \\ & h \in [1, N_h],\ i \in [1, 2] \end{aligned} \tag{4}$$

$$c_{\text{con}}(i,h) = \begin{cases} \beta_{\text{con}}, & q_i \text{ is connected directly to the} \\ & h\text{th cell in the last CP} \\ 0, & \text{Else.} \end{cases} \tag{5}$$

It has been observed that there are sometimes multiple solutions with comparable lengths but distinctive coverage patterns (see Fig. 4). Therefore, $c_{\text{con}}(\cdot)$ is introduced to prevent the paths from changing frequently among different patterns, which could result in inconsistent movements and slow down the exploration.

To reduce our problem to an asymmetric VRP, the connection costs from the virtual depot to the two quadrotors are assigned with the block $-\mathbf{M}_{\text{inf}} = -[M_{\text{inf}}, M_{\text{inf}}]$, where M_{inf} is a huge value. The large negative costs make the virtual depot's node connect the two quadrotors' directly, since it immensely reduces the overall cost of the output routes. In this way, the output of the asymmetric VRP is composed of the desired shortest paths and four extra edges, in which two edges link the depot to quadrotors while the other two ones link two cells to the depot. The reduction of our problem to the asymmetric VRP is better illustrated in Fig. 4.

2) Capacity Constraints: Although the length of the routes is optimized, the actual amount of unknown areas explored by each quadrotor is not considered, which can still lead to unbalanced

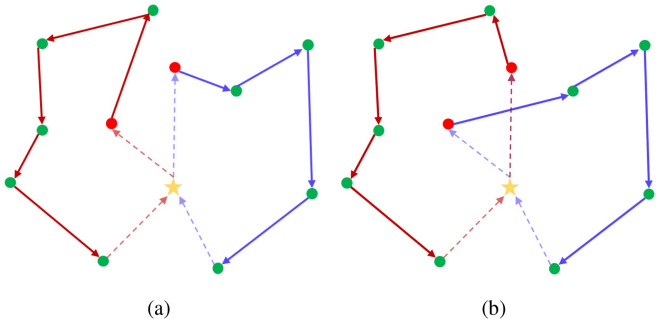


Fig. 4. Illustration of the asymmetric VRP. Nodes associated with the virtual depot, quadrotors, and hgrid cells are shown in yellow star, red circles, and green circles, respectively. The sequentially connected edges shown in red and purple represent the two routes that minimize the overall cost. (a) and (b) Two alternative solutions with identical cost. The desired CPs are obtained by removing the edges connecting with the virtual depot (displayed in dash).

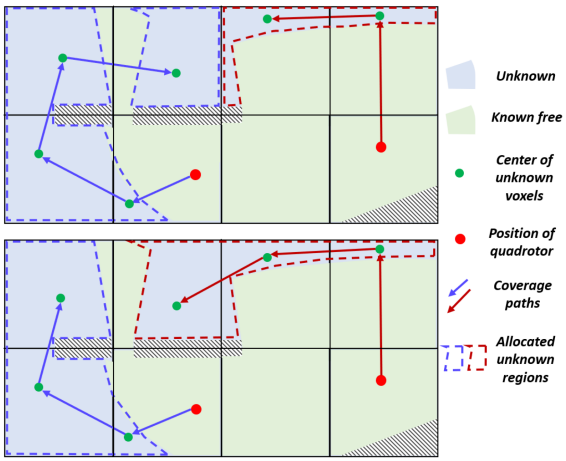


Fig. 5. CPs of two quadrotors and the corresponding allocation of unknown space. Top: only the total path length is considered. The allocation of unknown space is unbalanced, in which the red one could be completely explored much faster. Bottom: the workloads of two quadrotors are more equitable by introducing the capacity constraints.

partitioning of workloads sometimes, as the example in Fig. 5 shows. For this reason, we introduce *capacity* constraints of vehicles to further balance the workload allocated to the quadrotors. For each node ν_k in the VRP, if it is associated with a hgrid cell (suppose the h_kth cell), then a *demand* ρ_k equal to the number of unknown voxels u_{h_k} is assigned. Otherwise, a zero demand is set. We restrict the capacity of each vehicle to a percentage of the total number of unknown voxels. Denoting the set of nodes in the route of quadrotor q_i as $\mathcal{R}_i$, the capacity constraints are

$$\sum_{\nu_k \in \mathcal{R}_i} \rho_k \leq \alpha_\rho \sum_{h=1}^{N_h} u_h, \quad i = 1, 2. \tag{6}$$

With the capacity constraints, the problem becomes a CVRP. To solve it, we utilize an Lin–Kernighan–Helsgaun solver [82] extended by Helsgaun. The algorithm transforms the VRP into an equivalent standard TSP, and uses penalty functions for handling the capacity constraints. Although the problem is known to be NP-hard, our problem scale is small, for which an optimal solution can be obtained in most cases [82].

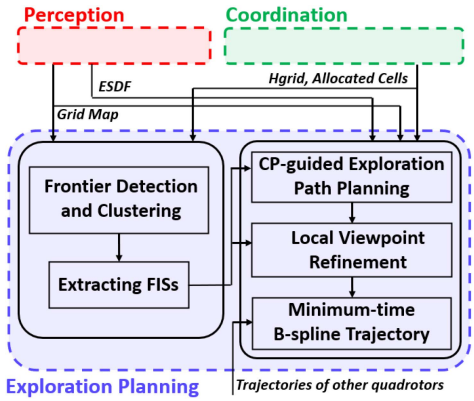


Fig. 6. Detailed components of the exploration planning module.

B. Sparse Graph for Path Searching

The major overhead involved in the CVRP is the computation of $\mathbf{C}_h$ and $\mathbf{C}_{q,h}$, which requires $O(N_h^2)$ path searching. This can be considerably expensive in large-scale environments, when a large number of global paths should be searched directly on the volumetric map. We take inspiration from [24] and [83] to relieve this computational burden. Specifically, a sparse graph embedded in the hgrid is maintained, which is leveraged to do global path searching. For each hgrid cell, a graph node is attached and the node of every adjacent cell is connected to it by a weighted edge, if a collision-free path exists within the ranges of the two cells. Whenever a hgrid cell is updated, the path to every adjacent cell is recomputed on the volumetric map. If a solution is found, the edge weight is updated by the path length. Otherwise, the edge is removed from the graph.

Leveraging this sparse graph, connection costs among hgrid cells and quadrotors can be computed more easily. For every pair of hgrid cells, a path is searched on the sparse graph to approximate the shortest path on the volumetric map. For a hgrid cell distant from the quadrotor, the cost is approximated by the path length from the quadrotor to the closest hgrid cell, plus the path length between the two cells on the sparse graph. In practice, the approximated costs lead to comparable solutions of the CVRP comparing with using the exact costs, but substantially reduce the computation time.

VI. Exploration Planning

The exploration planning approach is based on [1], which extract frontier information structures (FISs) incrementally and plan motions in several hierarchies. In this work, we extend the method to a multi-UAV system, allowing effective and scalable team exploration. We also extend the exploration planning by incorporating the CPs as high-level guidance, significantly improving the exploration rate. Collision avoidance under intermittent communication is also considered. The key components of the exploration planning are shown in Fig. 6.

Algorithm 3: FIS. Frontier clusters are represented by $\mathcal{F}_v$.

1 **for** $F_i \in \mathcal{F}_{\text{v}}$ **do**
2 **if HaveOverlap**$(F_i, \mathcal{B}_m)$ **then**
3 **if IsClusterChanged**(F_i) **then**
4 $\mathcal{F}_{\text{v}}$.**erase**(F_i)
5 $\mathcal{F}_{\text{new}} \leftarrow$ **DetectNewFrontierCluster**$(\mathcal{B}_m)$
6 **for** $F_i \in \mathcal{F}_{\text{new}}$ **do**
7 **if VoxelNum**$(F_i) < \varepsilon_{F,1}$ **then**
8 $\mathcal{F}_{\text{new}}$.**erase**(F_i)
9 **SplitLargeClusters**(F_i)
10 **Separate**$(\mathcal{F}_{\text{new}}, \mathcal{F}_a, \mathcal{F}_s)$
11 **for** $F_i \in \mathcal{F}_a$ **do**
12 $\mathbf{p}_{F_i} \leftarrow$ **Centroid**(F_i)
13 $\mathcal{V}_{F_i} \leftarrow$ **SampleViewpoints**$(\mathbf{p}_{F_i})$
14 **SortAndPrune**$(\mathcal{V}_{F_i}, N_{\mathcal{V}_F})$
15 **UpdateConnectionCosts**$(\mathcal{F}_{\text{v}}, \mathcal{F}_a)$
16 $\mathcal{F}_{\text{v}} \leftarrow \mathcal{F}_{\text{v}} \cup \mathcal{F}_{\text{new}}$

A. Incremental FIS

To facilitate the exploration planning, we adopt the method in [1] to incrementally detect frontiers [6], which are known-free voxels adjacent to unknown ones. Detected frontiers are grouped into clusters and rich information is extracted (see Algorithm 3). The procedure is summarized and more details can be found in [1]. It starts by removing outdated frontier clusters in the updated map, in which a fast prescreening using the clusters' AABBs is performed before a detailed check (Lines 1–4). Next, new clusters are searched. Clusters formed by sensor noises are filtered, and large clusters are split into smaller ones so that each can be covered by a viewpoint (Lines 5–9).

For each frontier cluster F_i, the centroid is computed (Line 12), while viewpoints are uniformly sampled (Line 13), where each viewpoint $\mathbf{q}_{i,j}$ is represented by the position and yaw[2] angle $\{\mathbf{p}_{i,j}, \varphi_{i,j}\}$. Viewpoints with sufficient coverage of the cluster are sorted in $\mathcal{V}_{F_i}$ in the order of descending coverage. Only the first $N_{\mathcal{V}_F}$ viewpoints are preserved (Line 14). Lastly, the connection cost between each pair of clusters is computed to assist the exploration planning (see Section VI-B). For two viewpoints $\mathbf{q}_{k_1,j_1}$ and $\mathbf{q}_{k_2,j_2}$, the time lower bound when moving between them is estimated as

$$t_{\text{lb}}\left(\mathbf{q}_{k_1,j_1}, \mathbf{q}_{k_2,j_1}\right) = \max\left\{\frac{\mathbf{Len}\left[P\left(\mathbf{p}_{k_1,j_1}, \mathbf{p}_{k_2,j_2}\right)\right]}{v_{\max}}, \frac{\min\left(|\varphi_{k_1,j_1} - \varphi_{k_2,j_2}|, 2\pi - |\varphi_{k_1,j_1} - \varphi_{k_2,j_2}|\right)}{\dot{\varphi}_{\max}}\right\} \quad (7)$$

where $\mathbf{Len}[P(\cdot)]$ is the length of a collision-free path and while $v_{\max}$ and $\dot{\varphi}_{\max}$ are the maximal velocity and yaw angle rate, respectively. The connection cost for clusters F_{k_1} and F_{k_2} is then given by $t_{\text{lb}}(\mathbf{q}_{k_1,1}, \mathbf{q}_{k_2,1})$.

[2]The roll–pitch–yaw angles convention is used, which rotates around the world frame's x-axis, then the y-axis, and finally the z-axis.

In contrast to [1], some adaptations are made for a multi-UAV system. Instead of only computing FISs within the map region updated by a quadrotor, regions shared by nearby quadrotors should also be considered to constantly keep a complete list of frontiers. The shared regions are continuously tracked by their AABBs $\mathcal{B}_m$, as mentioned in Section IV-B, to allow incremental updates. Compared to a single quadrotor, a greater number of clusters are detected, imposing a significant computational burden when extracting information for all of them. We circumvent this burden by storing clusters inside and outside the quadrotor's allocated working area in two separate lists $\mathcal{F}_{\text{a}}$ and $\mathcal{F}_{\text{s}}$ (Line 10). Information is only extracted for clusters in $\mathcal{F}_{\text{a}}$, with no effect on exploration planning. If the allocated area changes after interaction (see Section IV-C), associated clusters are reallocated in the two lists accordingly.

B. Hierarchical Planning

Our previous approach [1] employs a hierarchical planning pipeline, achieving significant improvement of exploration rate compared with recent methods [2], [13]. However, it does not consider global coverage routes, so the quadrotor may revisit the same regions repeatedly, leading to decreased performance. To further improve efficiency, we exploit the global CPs to guide the exploration planning, so that the quadrotor visits different regions in a more sensible order.

1) CP-Guided Exploration Path Planning: In Section V-A, the CVRP outputs the CP of the hgrid cells assigned to each quadrotor. When the hgrid is updated as the map changes, we recompute the CP to guide the exploration planning. For the next N_{CP} hgrid cells along the CP, we retrieve the frontier clusters whose centroids lie inside them. Then, we find a path that starts at the quadrotor's current viewpoint, visits each of the clusters, and ends at the $(N_{CP}+1)$th hgrid cell's centroid, as shown in Fig. 7. Inspired by [22], the problem is formulated as a variant of TSP with fixed start and end points. Since TSP is a special case of VRP (the number of vehicle is 1), a procedure similar to that in Section V-A can be adopted to solve the problem, except that an extra end-point constraint introduced by the $(N_{CP}+1)$th hgrid cell should be considered.

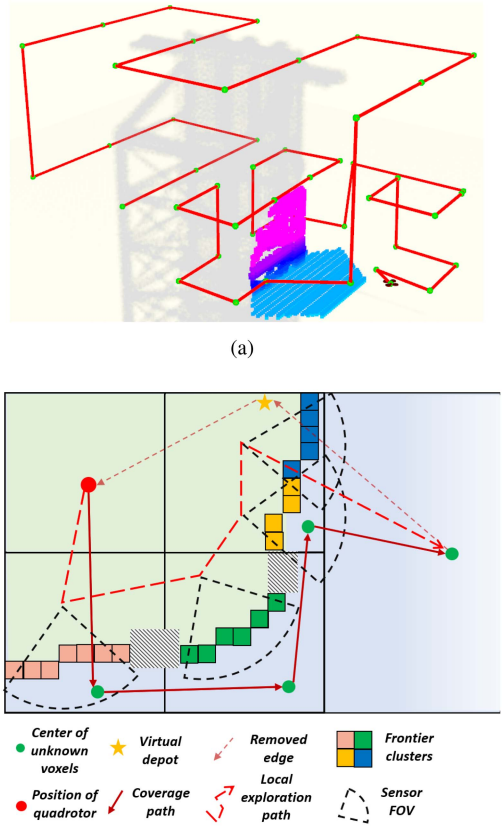
Assume there are N_{ftr} clusters totally, the engaged TSP has $N_{ftr}+3$ nodes, in which N_{ftr} nodes are for the clusters and 3 ones are for the virtual depot, quadrotor, and hgrid cell respectively. The cost matrix $\mathbf{C}_{\text{tsp}} \in \mathbb{R}^{(N_{ftr}+3)\times(N_{ftr}+3)}$ is

$$\mathbf{C}_{\text{tsp}} = \begin{bmatrix} 0 & -M_{\text{inf}} & \mathbf{0} & 0 \\ 0 & 0 & \mathbf{C}_{q,f} & 0 \\ \mathbf{0} & \mathbf{C}_{q,f}^T & \mathbf{C}_f & \mathbf{C}_{h,f}^{\text{T}} \\ -M_{\text{inf}} & 0 & \mathbf{C}_{h,f} & 0 \end{bmatrix}. \quad (8)$$

$\mathbf{C}_f$ is the major symmetric block recording the connection costs between clusters, whose entries are computed by

$$\mathbf{C}_f(k_1, k_2) = t_{\text{lb}}(\mathbf{q}_{k_1,1}, \mathbf{q}_{k_2,1}),\ k_1, k_2 \in [1, N_{\text{ftr}}]. \quad (9)$$

Different from the costs of hgrid cells (3), (9) takes into account not only translational distance, but also the change of yaw angle.



(a) (b)

Fig. 7. Illustration of CP-guided path planning. (a) Sample CP of the subdivided unknown space. (b) Local path is computed to cover the frontier clusters along the CP ($N_{CP} = 3$).

As these costs are already precomputed when new frontier clusters are extracted (see Section VI-A), $\mathbf{C}_f$ can be filled without extra overhead.

$\mathbf{C}_{q,f} \in \mathbb{R}^{1\times N_{ftr}}$ is the cost from the current viewpoint $\mathbf{q}_0 = (\mathbf{p}_0, \varphi_0)$ to the N_{ftr} clusters:

$$\mathbf{C}_{q,f}(k) = t_{\text{lb}}(\mathbf{q}_0, \mathbf{q}_{k,1}) + w_{\text{con}} \cdot t_{\text{con}}(\mathbf{q}_{k,1}),\ k \in [1, N_{\text{cls}}] \tag{10}$$

$$t_{\text{con}}(\mathbf{q}_{k,j}) = \begin{cases} \cos^{-1}\frac{(\mathbf{p}_{k,j}-\mathbf{p}_0)\cdot\mathbf{v}_0}{\|\mathbf{p}_{k,j}-\mathbf{p}_0\|\|\mathbf{v}_0\|}, & \mathbf{v}_0 \neq \mathbf{0} \\ 0, & \text{else} \end{cases} \tag{11}$$

where $\mathbf{v}_0$ is the current velocity, while $t_{\text{con}}(\cdot)$ penalizes large changes in flight direction, which is introduced to enable more consistent movements, similar to (5).

The costs from the clusters to the hgrid cell are accounted for by $\mathbf{C}_{h,f} \in \mathbb{R}^{1\times N_{\text{ftr}}}$, which is evaluated by

$$\mathbf{C}_{h,f}(k) = \mathbf{Len}[P(\mathbf{p}_{k,1}, \mathbf{c}_{N_{CP}+1})]/v_{\max}. \tag{12}$$

We transform our TSP variant to a standard one by introducing a huge negative cost $-M_{\text{inf}}$, which is assigned between the virtual depot and quadrotor, as well as between the hgrid cell and virtual depot. It ensures that in the output route, the nodes of the quadrotor and hgrid cell are adjacent to the depot's. As a result, we can obtain the desired path by removing the depot node and the two edges connected with it (see Fig. 7).

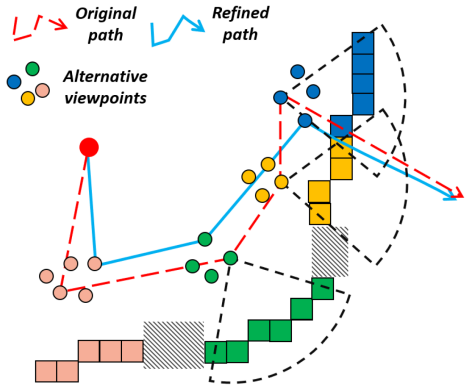


Fig. 8. Local viewpoint refinement based on the graph search approach. Along the local exploration path, alternative viewpoints of each frontier cluster are considered, further improving the path quality.

2) Local Viewpoint Refinement: A promising initial path is found using the CP-guided path planning. However, the path only considers a single viewpoint for each cluster, despite the fact that multiple possible viewpoints exist. To further improve the path's quality, the graph-based viewpoint refinement method in [1] is used. It generates a directed acyclic graph by taking into account the quadrotor's current viewpoint, the viewpoints of each cluster, and the hgrid cell on the initial path. It captures all possible viewpoint combinations along the initial path. Note that to incorporate the guidance of the CP, extra graph nodes and edges are introduced for the next hgrid cell to be explored, which differs from the original method. Given this graph, the Dijkstra algorithm is employed to find the optimal path that minimizes the exploration cost. The process is depicted in Fig. 8.

3) Minimum-Time B-Spline Trajectory Generation: Given the path comprising discrete viewpoints, a continuous-time trajectory executable by the quadrotor is needed. We base our trajectory generation on [1] and [66], which generate smooth, safe, and dynamically feasible B-spline trajectories in real time. We further consider collision avoidance among the quadrotors.

The differential flatness of quadrotor [84] allows us to generate trajectories simply for the flat output $\mathbf{q} \in (x, y, z, \varphi)$. For each quadrotor q_i, we find the uniform B-spline trajectory $\mathbf{\Psi}_{b,i}(t) = (\mathbf{p}_b(t), \varphi_b(t))$ that minimizes smoothness cost and total trajectory time under the constraints of safety, dynamic feasibility, and boundary state. The B-spline has a degree p_b and is defined by a set of $N_b + 1$ control points $\mathcal{Q}_{\text{c,b}} = \{\mathbf{q}_{\text{c},0}, \mathbf{q}_{\text{c},1}, \cdots, \mathbf{q}_{\text{c},N_b}\}$, where $\mathbf{q}_{\text{c},m} = (\mathbf{p}_{\text{c},m}, \varphi_{\text{c},m})$, and a knot span Δt_b. An optimization problem is formulated to find the desired solution:

$$\arg\min_{\mathcal{Q}_{\text{c,b}}, \Delta t_b} J_s + w_t T + \lambda_c (J_{c,o} + J_{c,q}) + \lambda_d (J_v + J_a) + \lambda_{bs} J_{bs}.$$

Here, J_s is the elastic band smoothness cost, T is the total trajectory time, $J_{c,o}$ and $J_{c,q}$ are the penalties to avoid collisions with obstacles and other quadrotors, respectively. J_v and J_a are the constraints of dynamics feasibility, while J_{bs} is the boundary state constraints considering the instantaneous state $(\mathbf{q}_0, \dot{\mathbf{q}}_0, \ddot{\mathbf{q}}_0)$

and the target viewpoint obtained in Section VI-B2. The method has been shown to find high-quality trajectories in real time and can support high-speed navigation in complex environments for a single quadrotor. Detailed formulation can be found in [1] and only the multirobot collision avoidance term $J_{c,q}$ is specified here.

For a quadrotor to avoid collisions with nearby ones $\mathcal{Q}_n$, their trajectories $\mathbf{\Psi}_{b,j}(t), q_j \in \mathcal{Q}_n$ are involved. The distance between q_i and every q_j is enforced, similar to [76]:

$$J_{c,q} = \sum_{q_j \in \mathcal{Q}_n} \sum_{k=1}^{T/\delta t_q} \mathcal{J}(d_{q,j}(T_k), d_{\min,q}) \tag{13}$$

$$d_{q,j}(t) = \left\| \mathbf{E}^{\frac{1}{2}} \left[\mathbf{\Psi}_{b,i}(t) - \mathbf{\Psi}_{b,j}(t) \right] \right\|. \tag{14}$$

Here, $T_k = T_s + k\delta t_q$, where T_s is the start time of $\mathbf{\Psi}_{b,i}(t)$, $\mathbf{E} = diag(1, 1, \beta_q), \beta_q < 1$ transforms the Euclidean distance to account for the downwash effect of quadrotors. The trajectory generation is also performed in a decentralized manner and it occurs at a higher frequency than in [1]. In addition to planning a new trajectory after a new exploration path is computed, each quadrotor shares its latest trajectory periodically via the broadcast network. If a quadrotor receives a trajectory from others and a collision is detected, it immediately generates a new trajectory that avoids the collision while still reaching the same target viewpoint. The constant exchange of trajectories and replanning ensures safety under communication latency or packet losses. A more comprehensive analysis of this reciprocal collision avoidance can be found in [76].

VII. Implementation Details

A. Mapping and Information Sharing

To achieve fast exploration, an efficient mapping framework is essential. In this work, we utilize a volumetric mapping [85], which has shown promising performance in fast autonomous flights [66], [68] and exploration in complex scenes. Similar to [80], which is widely adopted in exploration, it builds an occupancy grid map by fusing sensor measurements like depth images. It allows efficient and probabilistic updates of occupied and free space, and models the unknown space for the exploration planning. Meanwhile, it also maintains an euclidean signed distance field (ESDF) using an incremental algorithm to facilitate the gradient-based trajectory planning (see Section VI-B3). More details about the mapping framework can be found in [85]. The hgrid decomposition is created based on this volumetric map, where it obtains information about known and unknown space.

For a more informed exploration planning, it is important to exchange map data. Unlike some multirobot mapping approaches that share map information as submaps at a low frequency [37], [86], we group the newly observed voxels into *chunk*s and share them immediately with nearby quadrotors when unknown areas are explored. To update the latest map information promptly, the chunks are broadcast by adopting a user datagram protocol (UDP) protocol to alleviate the costly handshakes [87]. Meanwhile, to deal with packet drop and limited communication range, each quadrotor stores a bookkeeping recording the held chunks, which are either produced by itself or received from others. The bookkeeping is broadcast at interval, while every quadrotor receiving the bookkeeping finds the unrecorded chunks and shares the ones it holds. This sharing scheme allows retrieving lost information and relaying messages from quadrotor to quadrotor, which makes the system less sensitive to unreliable and range-limited communication.

B. Decentralized Multirobot State Estimation

In a decentralized exploration system, each quadrotor should be able to localize itself and others independently. In real-world experiments, we employ Omni-Swarm [77], a decentralized state estimator that fuses measurements from camera, inertial measurement unit (IMU), and landmarks and keyframes shared among quadrotors. By fusing the image and IMU data with a tightly coupled optimization-based method, it estimates the ego-motion of a quadrotor at high frequencies. For relative pose estimation, a map-based module is introduced, which relies on the landmarks and keyframes shared among quadrotors and a loop closure detection procedure. Based on it, relative localization and re-localization can be performed by identifying common locations visited by different quadrotors, which also enables fast initialization of the relative state estimation. A graph-based optimization and forward propagation backend fuses the abovementioned measurements, generating accurate and globally consistent estimation in real time. More details about the performance of this estimator, as well as its computational overhead and requirement of bandwidth can be found in [77]

The relative state estimation allows the quadrotors to establish a common coordinate frame. Note that the state estimator should be initialized at the start of the mission. At this point, we position all quadrotors in areas with common visual features so that the map-based module can detect loops and determine relative poses. The estimator then can work in non-line-of-sight conditions, such as when the quadrotors are completely separated by walls, as long as the same site can be visited sometime [77]. Nevertheless, we note that if the quadrotors explore for a long time and constantly do not visit common places, estimation errors can accumulate and cause inconsistent localization. To resolve this issue and simplify the initialization process, the incorporation of bearing measurements might be considered, as discussed in [88]. However, this is an estimation problem that is beyond the scope of this article.

C. System Setup

In real-world experiments, customized quadrotor platforms are used. The sensor used by each quadrotor for dense mapping is an Intel RealSense depth camera D435, which has an FoV of $[87 \times 58]\ deg$. Each quadrotor has a DJI manifold 2G onboard computer that contains an NVIDIA Jetson TX2 module.[3] Each quadrotor weighs 1.32 kg in total. The propulsion system uses

[3][Online]. Available: https://www.dji.com

approximately 246 W of power. The onboard computer consumes up to 15 W, the depth camera consumes 3.5 W, and an UWB module consumes 1.3 W. With all hardware and software modules operational, the quadrotor can hover or fly at a low speed for 15 min.

At the beginning of the exploration, the allocation of unknown space is initialized by running a few complete rounds of pairwise interaction, i.e., every pair of quadrotors interact and reallocate their hgrid cells in each round. This process typically obtains a reasonable initial allocation within a few seconds, thanks to the fast convergence of the algorithm (detailed in Section VIII-D) and the short interaction interval (0.5 s). The CVRP and TSP problems are solved by the LKH3 package,[4] while the trajectory optimization is implemented based on Nlopt.[5] A geometric controller [89] is adopted for trajectory tracking control. All the state estimation, mapping, planning, and control algorithms run on the onboard computer. The quadrotors exchange information through the wireless ad hoc network. The exchange of messages is implemented by the tools in LCM.[6] For simulation, we use a customized simulating package containing the quadrotor dynamics model, map generator, and depth camera model. All simulations run on an Intel Core i7-8700K CPU and GeForce GTX 1080 Ti GPU.

VIII. Results

The proposed approach is evaluated extensively through simulation and real-world experiments. To justify the design of each component in our algorithm, we conduct ablation studies comparing the complete method against baseline variants (see Section VIII-A). We compare different strategies to coordinate the multirobot exploration in Section VIII-B. In Section VIII-C, we study how the number of quadrotors influences the exploration. Finally, the real-world experiments including indoor and outdoor ones are presented in Section VIII-F.

A. Ablation Studies

1) Coordinating Multiple Robots: To validate the effectiveness of the hgrid-based pairwise interaction and the CVRP-based task allocation, we compare our complete coordination approach *Full* to three variants. The first variant *NoHgrid* does pairwise interaction (see Section IV-C) to allocate frontiers among quadrotors. The key difference is that it does not decompose the entire space. Instead, it just regards the frontier clusters as elementary task units, as most approaches do, and partitions the clusters between pairs of quadrotors by an mTSP formulation. The second and third variants *H+BFS* and *H+mTSP* decompose the unknown space into hgrid cells, while different strategies are employed to allocate the cells. *H+BFS* clusters the cells in a BFS manner [30], whereas *H+mTSP* uses an mTSP formulation to reduce the total length of CPs. Except for the difference in coordination, the same path planning approach presented in [1] and trajectory generation (see Section VI-B3) are adopted for a fair comparison.

TABLE I
Ablation Study of Coordination Approach

Scene	Method	Exploration time (s)				Path length (m)			
		Avg	Std	Max	Min	Avg	Std	Max	Min
Pillar	NoHgrid	48.6	**0.80**	49.7	47.6	204.9	8.15	212.1	193.5
	H+BFS	46.9	1.31	48.3	45.1	201.3	8.97	209.7	192.6
	H+mTSP	44.0	0.85	45.2	43.2	194.6	9.05	204.1	182.4
	Full	**38.9**	1.01	**40.6**	**37.6**	**171.8**	**3.21**	**177.6**	**169.3**
Office	NoHgrid	47.5	1.12	49.2	46.2	210.1	**9.01**	219.9	197.5
	H+BFS	45.7	1.28	47.9	44.9	205.4	10.13	212.9	198.5
	H+mTSP	41.7	**1.08**	43.2	40.6	197.5	18.13	221.9	178.5
	Full	**35.4**	2.06	**37.1**	**31.8**	**169.8**	13.22	**179.4**	**146.1**

In the tests, the dynamic limits are set as $v_{\max} = 1.5$ m/s and $\dot{\varphi}_{\max} = 0.9$ rad/s. The FoVs of the sensors are set as $[80 \times 60]$ deg with a maximum range of 4.5 m. The tests are conducted in two scenes, where one contains randomly generated pillars and the other is an office-like environment, as the two scenes shown in Fig. 10. Four quadrotors are initially placed near the boundaries of the explored space. Each approach runs five times in each scene.

As shown in Table I, H+mTSP outperforms NoHgrid in both scenes, demonstrating that using subdivided regions as task units rather than frontier clusters is more effective. Although frontiers provide hints on how to navigate the space, they do not contain information on workloads. Specifically, a large unexplored area may lie behind a small frontier cluster, and vice versa, thus coordinating robots by frontier clusters usually leads to unbalanced partitioning of workloads. In contrast, the areas/volumes of subdivided unexplored areas directly indicate the amount of work, making it a more reasonable choice. Besides, since the hgrid cells represent disjoint regions, it prevents multiple quadrotors from visiting identical places and interfering each other. On the other hand, H+mTSP has a higher exploration efficiency than H+BFS. H+BFS provides a simple heuristic to divide the cells quickly; however, it does not explicitly optimize the length of CP, which limits its performance in complex scenes. Lastly, it is clear from the statistics that the CVRP-based partitioning further improves efficiency by a large margin. Comparing to the BFS and mTSP allocation, the CVRP accounts for not only the lengths of CPs, but also the actual amount of unexplored regions. Hence, workloads are more appropriately distributed.

2) Exploration Path Planning: To examine the CP-guided exploration path planning, we compare our new approach (*Full*) with our previous one [1] (*NoCP*). The work in [1] is shown to substantially outperform recent exploration planning approaches including [2] and [13], completing exploration three to eight times faster. The improvement comes from the consideration of efficient frontier coverage tours, the promising viewpoint, and trajectory optimization and its high computation efficiency. Despite its improvement, it does not consider the coverage route of the entire space, which sometimes leads to unnecessarily long paths. In this work, we show that the efficiency of exploration can be further improved by incorporating the global CPs (see Section VI-B1). To focus on the evaluation of exploration path planning, we compare exploration with a

[4][Online]. Available: http://akira.ruc.dk/~keld/research/LKH-3/
[5][Online]. Available: https://nlopt.readthedocs.io/en/latest/
[6][Online]. Available: https://lcm-proj.github.io/

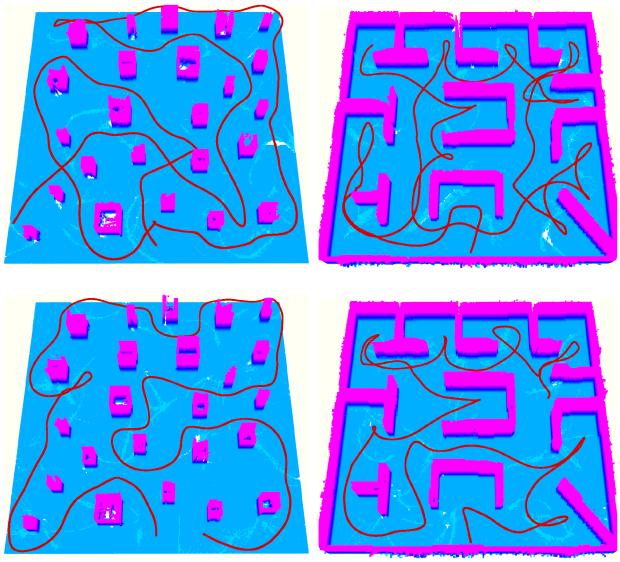


Fig. 9. Ablation study of the CP-guided path planning. Top: without considering the CP, the quadrotor visits the same regions redundantly. Bottom: different regions are visited in a more sensible sequence.

TABLE II
ABLATION STUDY OF EXPLORATION PATH PLANNING

Scene	Method	Exploration time (s)				Path length (m)			
		Avg	Std	Max	Min	Avg	Std	Max	Min
Pillar	NoCP	103.7	8.51	114.6	93.8	146.1	7.89	156.9	138.2
	Full	**86.7**	**4.33**	**92.5**	**82.1**	**117.2**	**3.20**	**121.7**	**114.6**
Office	NoCP	120.7	**1.87**	123.1	118.6	155.9	**1.46**	157.9	154.4
	Full	**96.9**	2.56	**99.5**	**93.4**	**125.1**	4.47	**130.8**	**119.9**

single quadrotor, as is shown in Fig. 9. The same parameters as those in Section VIII-A1 are set.

The advantage of introducing CPs is apparent from Fig. 9 and Table II, which shows a more sensible exploration pattern, a significantly shorter exploration time, and overall path length. Without taking the coverage route into account, it is observed that the quadrotor frequently moves to another region before thoroughly exploring one. As a result, the quadrotor must revisit known regions later in order to explore previously missed areas, resulting in inefficiency. The incorporation of CPs, on the other hand, consistently provides a visitation sequence of the decomposed unexplored regions. As a result, the quadrotor can explore the space in a more rational manner, rather than returning to the same locations repeatedly.

B. Coordination of Multirobot Exploration

To further evaluate our coordination method, we benchmark it against four widely adopted approaches including centralized [25], [28] and decentralized [34], [38] ones. *Iter* [25] uses a central controller that iteratively determines the appropriate target frontier cluster for each robot. In each round, the best pair of robot and frontier cluster is computed, after which the utility of each frontier is updated according to the previous assignments. The process is repeated until each robot is assigned with a frontier cluster. *mTSP* [28] is more sophisticated than *Iter* because it considers the optimal allocation of all frontier clusters for each robot. The allocation problem leads to an mTSP formulation. Different from *Iter* and *mTSP*, the works [34] and [38] do not use a central controller, but have all robots to make decision independently. *Auc* [34] exploits an auction-based architecture to achieve coordination among robots. In the framework, robots continuously negotiate with nearby ones, which allows each robot to explore their optimal target if there is no conflict, and resolves conflicting targets by comparing the expected travel costs and rewards. Like *mTSP*, *Pair* [38] allocates all targets among robots, but it adopts a pairwise optimization in order to achieve decentralized allocation. The idea of pairwise interaction is similar to ours, but it entirely relies on frontier clusters rather than decomposed cells. The same path planning, trajectory generation and parameters as those in Section VIII-A1 are used.

TABLE III
EXPLORATION STATISTIC OF BENCHMARKED COORDINATION APPROACHES

Scene	CR(m)	Method	Exploration time (s)				Total path length (m)			
			Avg	Std	Max	Min	Avg	Std	Max	Min
Pillar	$+\infty$	Iter	51.6	1.90	54.3	50.2	213.1	6.90	221.6	204.7
		mTSP	45.9	2.34	49.2	44.2	187.5	5.14	194.8	183.9
		Pair	48.2	**0.83**	49.3	47.2	204.5	8.18	211.7	193.1
		Auc	51.6	1.72	53.3	49.2	212.9	7.35	222.4	204.4
		Ours	**38.5**	1.24	**40.2**	**37.2**	**171.3**	**1.14**	**172.5**	**169.8**
	10	Pair	51.9	1.70	54.3	50.3	209.2	15.78	229.0	**190.3**
		Auc	53.6	4.11	59.2	49.3	231.8	11.58	248.0	221.4
		Ours	**45.0**	**0.59**	**45.6**	**44.2**	**206.4**	**8.92**	**212.7**	193.8
	5	Pair	60.2	6.70	71.4	53.7	271.0	42.13	343.2	238.1
		Auc	62.3	7.13	72.3	56.3	279.3	45.34	343.4	246.2
		Ours	**46.4**	**1.29**	**47.4**	**44.6**	**208.9**	**14.66**	**228.8**	**194.0**
Office	$+\infty$	Iter	49.4	0.48	50.0	48.8	201.7	13.74	215.9	183.1
		mTSP	45.0	2.30	48.3	43.2	205.3	**8.18**	216.7	197.8
		Pair	47.7	**0.42**	48.2	47.2	209.7	9.31	219.7	197.3
		Auc	50.6	3.40	55.3	47.2	229.3	13.90	247.0	213.0
		Ours	**35.8**	2.14	**37.5**	**32.2**	**169.3**	13.63	**178.6**	**145.7**
	10	Pair	51.6	2.57	54.9	48.6	224.9	**8.55**	235.1	214.2
		Auc	51.5	**1.69**	53.1	49.1	245.1	11.82	260.4	231.6
		Ours	**40.2**	3.06	**43.2**	**36.0**	**176.0**	10.35	**185.2**	**161.6**
	5	Pair	55.9	1.86	58.5	54.1	253.2	**6.37**	259.1	244.4
		Auc	57.7	**1.38**	59.6	56.3	262.0	9.74	275.3	252.2
		Ours	**47.3**	1.53	**48.6**	**45.2**	**202.6**	8.05	**213.3**	**193.8**

First, we compare all five approaches under ideal communication, i.e., connections to the central server or among quadrotors are always available. Results are listed in Table III and Fig. 10. Among the baselines, approaches that consider the allocation of all targets jointly (*mTSP*, *Pair*) performs better than those allocating a single target at a time (*Iter*, *Auc*). Comparing to *Pair*, *mTSP* is marginally more efficient. However, *Pair* has lower communication requirements as only two quadrotors have to communicate at a time, allowing coordination in communication-limited scenarios. In terms of exploration time and total movement distance, the proposed method significantly outperforms all baselines. From Fig. 10(a)–(e), we can see that the paths produced by the baselines are longer and have more intersections than ours, indicating more wastage of energy. In comparison, the proposed coordination approach dispatches the quadrotors more reasonably to explore distinct regions, resulting in minor interference between the quadrotors. It is remarkable that our approach outperforms the centralized coordination, despite being completely decentralized, owing to the more proper

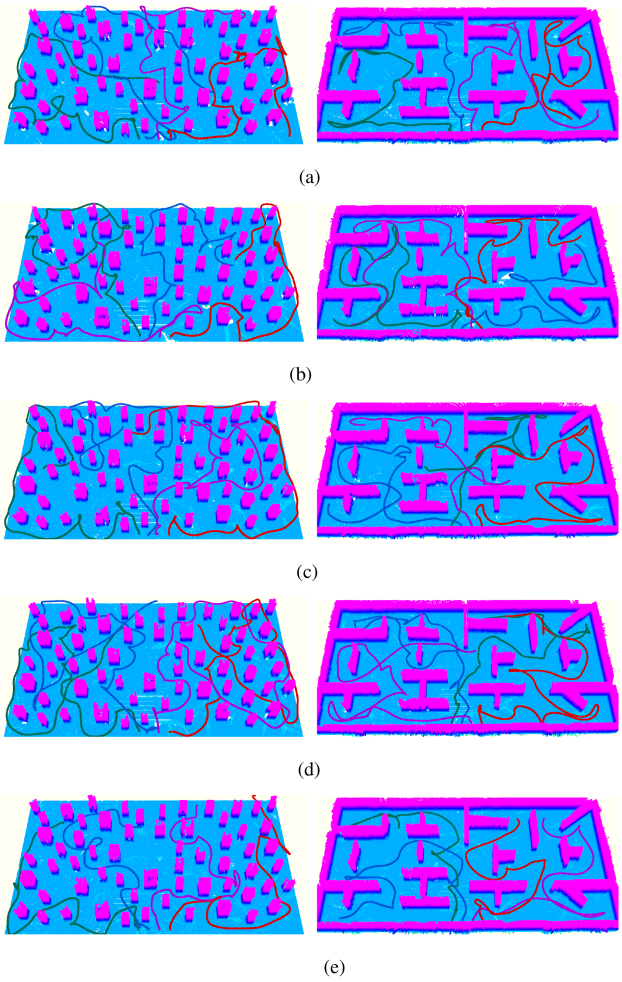


Fig. 10. Comparisons of coordination approaches in unstructured (left) and structured (right) environments. Ours dispatch the quadrotors more effectively to explore distinct regions. (a) Paths produced by approach *Iter* [25]. (b) Paths produced by approach *mTSP* [28]. (c) Paths produced by approach *Pair* [38]. (d) Paths produced by approach *Auc* [34]. (e) Path produced by our approach.

choice of task units and partitioning algorithm, as has been justified in Section VIII-A1.

Coordination under limited communication range: To further assess the robustness of the approaches, different communication ranges are simulated. The quadrotors cannot exchange information for coordination or share map data when they are out of the communication range. Since [25] and [28] require communication at all time, we only compare against [34] and [38]. As the communication range decreases, the exploration time and path lengths of all approaches increase. However, the time and lengths of [34] and [38] increase more particularly in the *Pillar* scene. Because of their limited communication range, quadrotors are less aware of which areas have already been explored by others. As a result, multiple quadrotors may explore the same regions redundantly at different times, wasting a significant amount of time and energy. This phenomenon is less severe in the *Office* scene, as we find that in the structured environment, quadrotors have a better chance of meeting each other and exchanging more information. In comparison to them, our approach has a more consistent performance in both scenes.

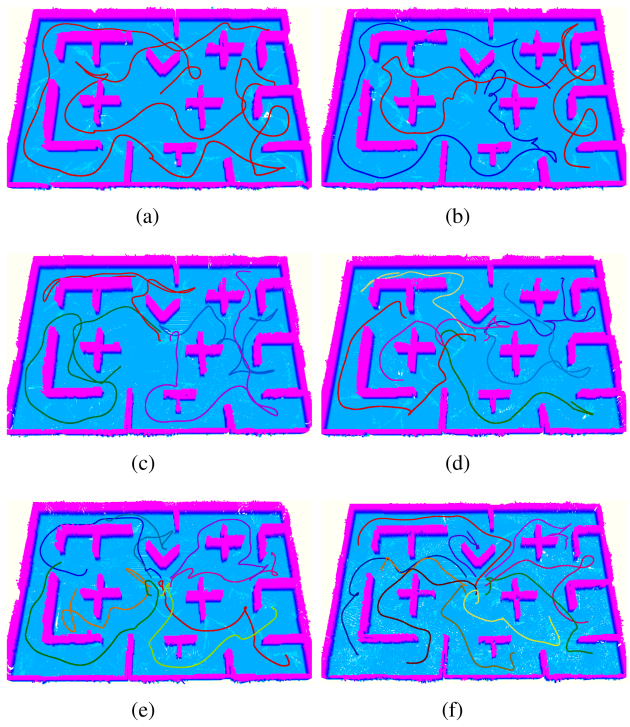


Fig. 11. Exploration paths with different numbers of quadrotors. A large team can be dispatched effectively without significant interference. (a) One drone. (b) Two drones. (c) Four drones. (d) Six drones. (e) Eight drones. (f) Ten drones.

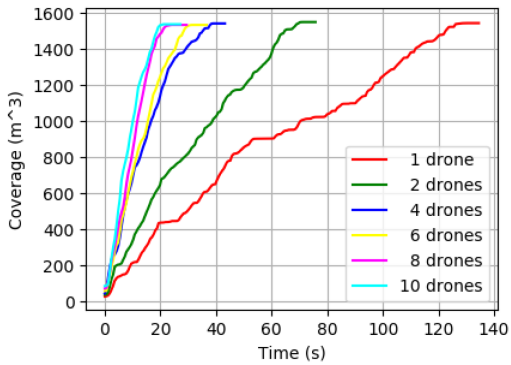
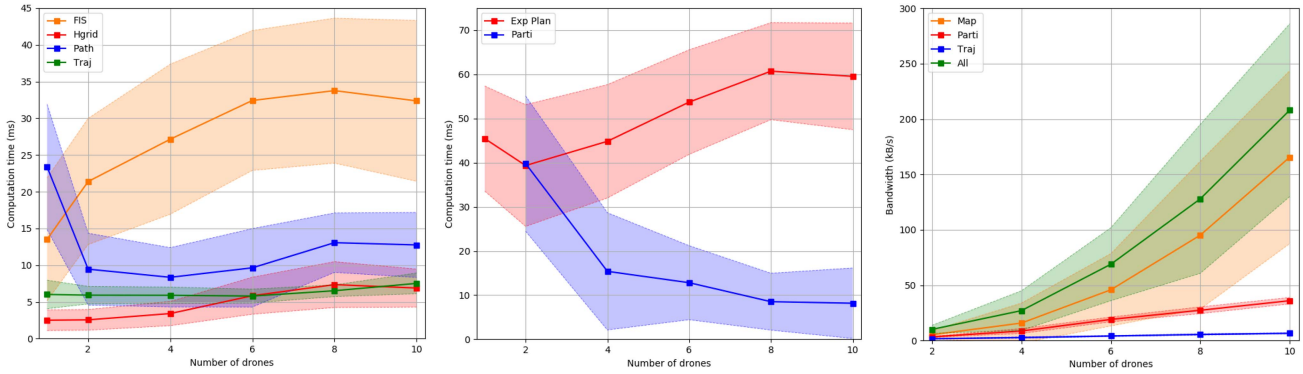


Fig. 12. Progress curves of exploration with different numbers of quadrotors. The exploration rate increases steadily as the team size grows.

The key point is that we subdivide the entire space into disjoint cells, which inherently ensures that a quadrotor does not visit regions assigned to others, even if communication is lost. By contrast, allocating frontier among quadrotors does not have this advantage, so its performance may suffer when communication is limited.

C. Study on Number of Quadrotors

To have a clearer understanding of the proposed approach, we study how the number of quadrotors influences its performance. In each test, the quadrotors start at the center of the scene and explore collaboratively. Samples of exploration paths and progress curves are presented in Figs. 11 and 12, respectively.



(a) (b) (c)

Fig. 13. Computation time and communication bandwidth requirements of key components of the proposed coordination and planning approach. The mean and one standard deviation are displayed.

TABLE IV
STUDY ON SUBOPTIMALITY OF PAIRWISE INTERACTION

#Round	Loss(%)	length1 (m)		length2 (m)		length2/length1	
		Avg	Std	Avg	Std	Avg	Std
\	100	134.74	4.37	485.95	15.76	3.609	0.140
1	80	136.83	4.78	301.79	27.93	2.206	0.199
	60	136.68	4.57	248.61	25.28	1.819	0.177
	40	135.17	4.53	216.49	18.43	1.602	0.143
	20	134.79	4.97	195.97	21.09	1.454	0.149
	0	135.17	5.12	181.23	14.47	1.340	0.096
2	80	135.55	4.59	242.55	24.09	1.790	0.181
	60	135.11	4.45	190.53	18.68	1.411	0.146
	40	134.65	4.53	161.18	13.93	1.197	0.097
	20	135.61	4.28	148.66	6.54	1.096	0.051
	0	135.02	4.70	140.36	6.55	1.039	0.031
3	80	134.55	4.32	217.43	20.35	1.617	0.160
	60	135.71	4.99	169.89	16.15	1.252	0.114
	40	135.79	4.25	148.18	8.12	1.091	0.052
	20	134.73	3.90	138.36	5.79	1.027	0.024
	0	135.28	4.91	136.45	5.44	1.010	0.013

We can see that the exploration rate is consistently improved as the number of quadrotors increases. Even with a large team of quadrotors, our approach is able to dispatch them well, where there are only minor interferences.

Fig. 13(a) and (b) reports the computation time of major modules in our approach. Fig. 13(a) shows the time for the frontier detection and information extraction (*FIS*), update of hgrid (*Hgrid*), the update of CP, path planning and local viewpoint refinement (*Path*), and the trajectory generation (*Traj*). The cumulative computation time of these four modules (*Exp Plan*) is shown in Fig. 13(b). The time of workload partitioning based on CVRP is also shown in Fig. 13(b) as *Parti*. Note that the exploration planning and coordination modules execute at different frequencies, i.e., several exploration paths are replanned between two task allocation, so their computation time is displayed separately. The statistics show that the approach is scalable since the computation time does not increase significantly with more quadrotors. Interestingly, the time of *Parti*, which is the key component to coordinate all quadrotors, decreases as the team size increases. The reason for this is that as more quadrotors work together to explore the same scene, the number of hgrid cells allocated to each pair of quadrotors decreases, making the associated CVRP easier to solve. This is in stark contrast to most coordination approaches, in which computation time increases as the number of robots increases. The time of *Path* decreases noticeably from 1 to 2 quadrotors for the same reason: when there is only one drone, a CP covering the entire space is required, which takes longer to compute. The CPs can be computed faster when more quadrotors are involved. The shorter time of CPs compensates for the longer time of path planning and local viewpoint refinement, resulting in *Path*'s nearly constant computation time.

The communication bandwidth requirements are shown in Fig. 13(c). Three types of messages are exchanged during the exploration: the map information (*Map*), the messages involved in workload partitioning (*Parti*), and quadrotor trajectories (*Traj*). The cumulative bandwidth is shown as *All*. The parameters that have an impact on the bandwidth requirements include the resolution of the volumetric map and the interaction frequency. In our tests, we set them as 0.1 m and 2 Hz, respectively. It can be seen that *Parti* and *Traj* use less than 50 kB/s, while *Map* takes up the most bandwidth. The total bandwidth requirement is much less than that of our wireless ad hoc network (>3 MB/s). It is possible to significantly reduce the size of map data when there are more quadrotors. For example, one can use a communication-efficient map presented in [59].

D. Study on Suboptimality

The pairwise interaction-based coordination is beneficial for communication-limited scenarios, but it may yield suboptimal results. To better understand its performance, we quantitatively compare it to its centralized counterpart. In each test, we randomly generate 100 target points and ten robot positions. For the centralized method, a central server can access the positions of all robots and targets, and a VRP considering all of them is solved to find the optimal paths passing through all targets. In comparison, the pairwise interaction method only has each pair of robot communicate in sequence and reallocate their assigned targets with VRP. At the beginning, the target points are randomly allocated to the robots, and each robot solves a TSP to generate an initial path. To evaluate the effect of unstable

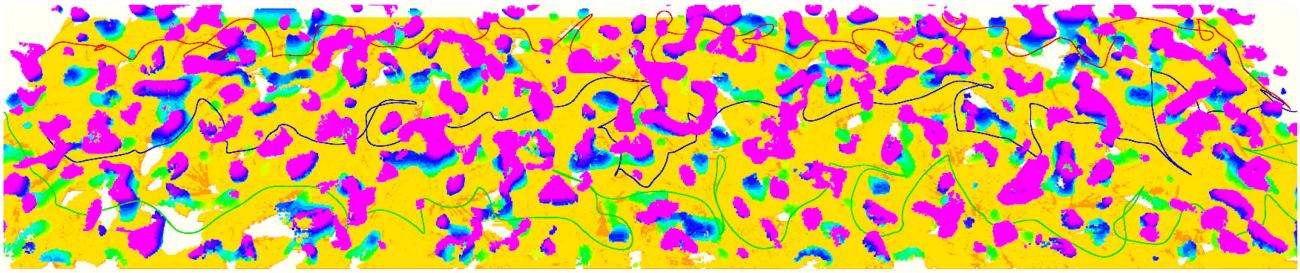


Fig. 14. Exploration test with three quadrotors in a complex large-scale environment. The random map generated by the Perlin noise algorithm is color-coded by height (from yellow to pink). The flight trajectories of the quadrotors are shown in red, blue, and green, respectively. More details can be found in the attached video (available online as supplementary material).

communication on its performance, different packet loss rates (Loss%) are simulated, making a specific portion of interaction fail. Different numbers of interaction rounds (#Round) are also tested in order to see the effect of more interaction. In a complete interaction round, each pair of robots will try to connect and reallocate the target points. We run 50 tests for each specific packet loss rate and interaction round, recording the path lengths and length ratios of the two methods.

The results are listed in Table IV. Apparently, more interaction rounds improve the solution. Under moderate packet losses (Loss $\leq 40\%$), our method achieves competitive performance, i.e., providing paths that are marginally longer than those of the centralized VRP after two interaction rounds. This is remarkable as the robots use only local communication and even suffer from communication losses. With more severe losses (Loss $\geq 60\%$), the generated paths are longer. However, the interaction still significantly improves the initial solutions (Loss $= 100\%$), with a reduction in path length by more than half. It indicates that even with a few interaction events, our method is capable of realizing effective coordination.

E. Exploration in Simulation Environments

To generate complex scenes for exploration, we utilize *mockamap*,[7] which implements the Perlin noise algorithm [90] to generate random 3-D obstacles. We investigate our system in two challenging scenes: a large-scale environment and a complex 3-D environment. The dynamics limits are set as $v_{\max} = 1.5$ m/s, $a_{\max} = 1.0$ m/s, and $\dot{\varphi}_{\max} = 0.9$ rad/s. The attached video (available online as supplementary material) demonstrates these tests more clearly.

1) Large-Scale Environment: One primary motivation for using multiple quadrotors is to explore large-scale environments more quickly. We conduct exploration tests in a simulated complex large-scale scene to observe the behavior of the proposed system. The scene is surrounded by a $20 \times 100 \times 3$ m^3 box, which covers an area of 2000 square meters. As shown in Fig. 14, three quadrotors begin on one side of the scene and cooperatively explore until they reach the other side. The exploration lasts 153 s, and each quadrotor's path length is 176.0, 169.9, and 187.6 m, respectively.

[7][Online]. Available: https://github.com/HKUST-Aerial-Robotics/mockamap

Fig. 15. Exploration in a complex 3-D environment. The visualization is consistent to that in Fig. 14.

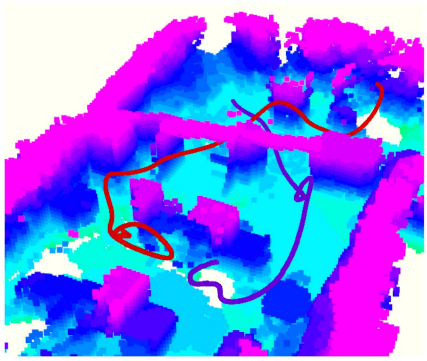
2) Complex 3-D Environment: To see the experiment in a truly 3-D environment, which better exploits the maneuverability of quadrotors and demonstrates our system's capability, we conduct tests in a complex 3-D scene with the size of $10 \times 25 \times 8$ m^3. Fig. 15 shows the result of one trail. The exploration lasts for 30 s, and the path length is 36.7, 42.8, and 42.2 m, respectively.

F. Real-World Exploration Experiments

To validate the performance of the proposed approach in real-world scenarios, we conduct extensive field experiments in both indoor and outdoor environments. In all the indoor experiments, we set the dynamics limits as $v_{\max} = 1.0$ m/s, $a_{\max} = 0.8$ m/s, and $\dot{\varphi}_{\max} = 0.9$ rad/s. The velocity limit is increased to $v_{\max} = 1.5$ m/s outdoor. Note that we do not use any external device for localization or any central server to control the flights. All state estimation, mapping, coordination, motion planning, and control run on the onboard computers. Every quadrotor makes decisions and accomplishes its task in a decentralized fashion. Besides, our algorithm can support higher flight speed than 1.5 m/s. However, faster flight can degrade the quality of collected depth images, which is detrimental to accurately mapping the environment.

First, we conduct fully autonomous exploration experiments in indoor scenes. In the first scene, we test exploration with two quadrotors, as displayed in Fig. 16. Within the experiment area, we randomly deploy obstacles to make up a cluttered environment. We bound the space with a $10 \times 6 \times 2$ m^3 box. Both quadrotors are initialized with their backs toward the space to be explored, in order to reduce the information obtained by

(a)



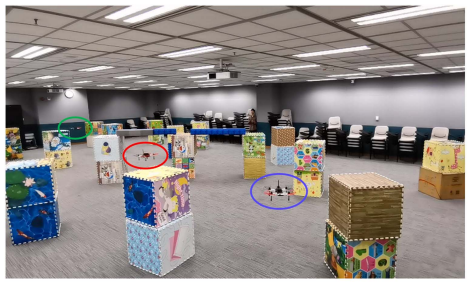
(b)

Fig. 16. Two quadrotors simultaneously explore a complex unknown environment. (a) Composite image of the experiment. (b) Visualization of the online built map and flight trajectories. All quadrotors collaborate in a fully decentralized manner, i.e., no external computer is used to dispatch them, each quadrotor runs the state estimation, mapping, coordination, and planning algorithms independently on its onboard computer.

them before starting exploration. A trigger information is sent to the quadrotors,[8] when they start to explore collaboratively. The space is completely explored in 23 s, when the movement distances of the two quadrotors are 13.4 and 14.4 m, respectively. The experiment validates the capability of our system to dispatch multiple quadrotors. It also examines the ability of each quadrotor to perform 3-D maneuvers, mapping the unknown space quickly and avoiding obstacles agilely. One sample of an online constructed map and the exploration trajectories is presented in Fig. 16.

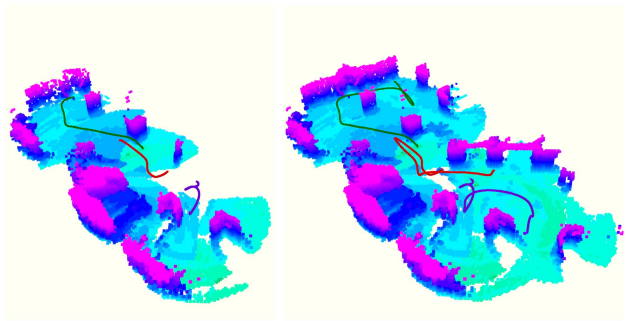
In the second scene, we test our approach in an environment with a larger scale and with three quadrotors. It is more challenging as more quadrotors are involved and communication becomes less stable when distances between quadrotors increases. The environment to conduct the experiments is shown in Fig. 17(a) and (b). The space is bounded by a $15 \times 9 \times 2$ m^3 box. We follow the same settings as those for experiments with two quadrotors. The executed trajectories and online built map after exploring for 8 and 16 s are shown in Fig. 17(b). The exploration is finished in 33 s, when the path lengths for the three quadrotors are 22.3, 23.0, and 25.7 m. The complete map and trajectories are shown in Fig. 17(c).

[8]The information is sent from a ground computer, which serves as an interface to start the experiment. After it, the ground computer no longer controls the quadrotors or runs any algorithm related to the exploration.

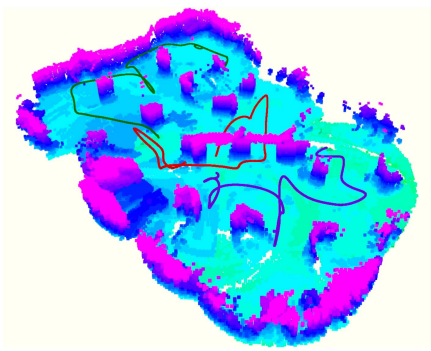


(a)

(b)



(c)

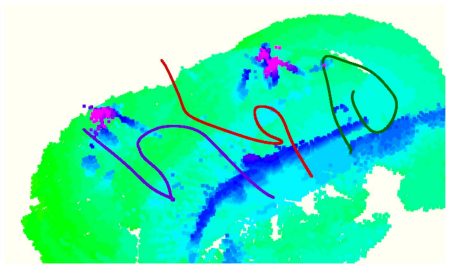


(d)

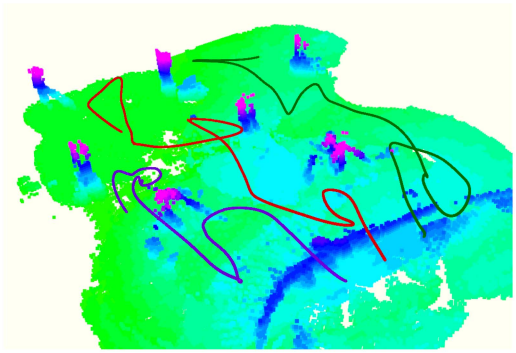
Fig. 17. Fast exploration experiments with three quadrotors in an complex indoor scene. (a) and (b) Snapshots of the flight taken from different sites. (c) Snapshots of online built map and exploration paths at 8 and 16 s. (d) Complete map and paths of all three quadrotors.

Lastly, to validate the robustness of our method in natural environments, we conduct exploration tests in a forest [see Fig. 18(a)]. The size of the area to explore is $15 \times 12 \times 2$ m^3. A snapshot of the map and trajectories after starting exploration for 13 s is shown in Fig. 18(b), and the complete map and trajectories are shown in Fig. 18(c). The exploration lasts for 33 s and the path lengths are 33.3, 25.2, and 34.1 m, respectively.

(a)



(b)



(c)

Fig. 18. Exploration in a forest with three quadrotors. (a) Snapshot of the exploration process. (b) Map and paths at 13 s. (c) Complete map and paths.

Overall, the experiments demonstrate the applicability of the proposed multiquadrotor exploration approach in realistic scenarios, where there is limited communication, restricted computation resource, and considerable noises in perception. We refer the readers to the attached video (available online as supplementary material) for more details about the experiments, such as the online allocation of tasks and the real-time motion planning.

IX. Conclusion

In this article, we present a systematic approach for fast exploration of complex environments using a fleet of decentralized quadrotors. To coordinate the team with only asynchronous and unreliable communication, we decompose the unknown space into hgrid and distribute the task units among quadrotors by a pairwise interaction. It partitions the workloads appropriately among the quadrotors. The overall lengths of coverage routes are minimized and the workloads are balanced via a CVRP formulation, which further enhances the team cooperation. Each quadrotor is capable of exploring the assigned regions safely and efficiently by subsequently building FISs, finding exploration paths, refining viewpoints, and generating minimum-time trajectories. The performance of the approach is evaluated extensively, showing the high exploration rate, the robustness against communication loss, the capability to dispatch a large team of quadrotors, and the high computation efficiency. Moreover, fully autonomous exploration with a fully decentralized multi-UAV system is achieved for the first time. To benefit the community, we will release our implementation. In the future, we plan to improve the reconstruction quality and global consistency of the multirobot mapping module. We will also study more sophisticated strategies to deal with limited communication range, like scheduling meeting events for the robots to share knowledge.

References

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

[2] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 2135–2142.

[3] M. Dharmadhikari et al., "Motion primitives-based path planning for fast and agile exploration using aerial robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 179–185.

[4] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1500–1507, Apr. 2020.

[5] S. Song and S. Jo, "Online inspection path planning for autonomous 3D modeling using a micro-aerial vehicle.," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 6217–6224.

[6] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom. Towards New Comput. Princ. Robot. Autom.*, 1997, pp. 146–151.

[7] M. Juliá, A. Gil, and O. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," *Auton. Robots*, vol. 33, no. 4, pp. 427–444, 2012.

[8] S. Shen, N. Michael, and V. Kumar, "Stochastic differential equation-based exploration algorithm for autonomous indoor 3D exploration with a micro-aerial vehicle," *Int. J. Robot. Res.*, vol. 31, no. 12, pp. 1431–1444, 2012.

[9] D. Deng, R. Duan, J. Liu, K. Sheng, and K. Shimada, "Robotic exploration of unknown 2D environment using a frontier-based automatic-differentiable information gain measure," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics*, 2020, pp. 1497–1503.

[10] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using Rao-Blackwellized particle filters," in *Proc. Robot.: Sci. Syst.*, 2005, vol. 2, pp. 65–72.

[11] S. Ahmad, A. B. Mills, E. R. Rush, E. W. Frew, and J. S. Humbert, "3D reactive control and frontier-based exploration for unstructured environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 2289–2296.

[12] C. Connolly, "The determination of next best views," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1985, vol. 2, pp. 432–435.

[13] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon 'next-best-view' planner for 3D exploration," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1462–1468.

[14] C. Papachristos, S. Khattak, and K. Alexis, "Uncertainty-aware receding horizon exploration and mapping using aerial robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 4568–4575.

[15] T. Dang, C. Papachristos, and K. Alexis, "Visual saliency-aware receding horizon autonomous exploration with application to aerial robotics," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 2526–2533.

[16] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon path planning for 3D exploration and surface inspection," *Auton. Robots*, vol. 42, no. 2, pp. 291–306, 2018.

[17] C. Witting, M. Fehr, R. Bähnemann, H. Oleynikova, and R. Siegwart, "History-aware autonomous exploration in confined environments using MAVs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.

[18] C. Wang et al.., "Efficient autonomous robotic exploration with semantic road map in indoor environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2989–2996, Jul. 2019.

[19] E. Palazzolo and C. Stachniss, "Effective exploration for MAVs based on the expected information gain," *Drones*, vol. 2, no. 1, p. 9, 2018.

[20] B. Charrow et al., "Information-theoretic planning with trajectory optimization for dense 3D mapping," in *Proc. Robot.: Sci. Syst.*, 2015, vol. 11, pp. 3–12.

[21] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, "Efficient autonomous exploration planning of large-scale 3-D environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1699–1706, Apr. 2019.

[22] Z. Meng et al., "A two-stage optimized next-view planning framework for 3-D unknown environment exploration, and structural reconstruction," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1680–1687, Jul. 2017.

[23] C. Cao, H. Zhu, H. Choset, and J. Zhang, "Exploring large and complex environments fast and efficiently," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 7781–7787.

[24] F. Yang, D.-H. Lee, J. Keller, and S. Scherer, "Graph-based topological exploration planning in large-scale 3D environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 12730–12736.

[25] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 376–386, Jun. 2005.

[26] J. Butzke and M. Likhachev, "Planning for multi-robot exploration with multiple objective utility functions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 3254–3259.

[27] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 1160–1165.

[28] J. Faigl, M. Kulich, and L. Přeučil, "Goal assignment using distance cost in multi-robot exploration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 3741–3746.

[29] S. Dong et al., "Multi-robot collaborative dense scene reconstruction," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–16, 2019.

[30] N. Karapetyan, K. Benson, C. McKinney, P. Taslakian, and I. Rekleitis, "Efficient multi-robot coverage of a known environment," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1846–1852.

[31] G. Hardouin, J. Moras, F. Morbidi, J. Marzat, and E. M. Mouaddib, "Next-best-view planning for surface reconstruction of large-scale 3D environments with multiple UAVs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 1567–1574.

[32] B. Yamauchi, "Decentralized coordination for multirobot exploration," *Robot. Auton. Syst.*, vol. 29, no. 2/3, pp. 111–118, 1999.

[33] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, vol. 3, pp. 3016–3023.

[34] A. J. Smith and G. A. Hollinger, "Distributed inference-based multi-robot exploration," *Auton. Robots*, vol. 42, no. 8, pp. 1651–1668, 2018.

[35] M. Berhault et al., "Robot exploration with combinatorial auctions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2003, vol. 2, pp. 1957–1962.

[36] M. Corah and N. Michael, "Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice," *Auton. Robots*, vol. 43, no. 2, pp. 485–501, 2019.

[37] J. Yu et al., "SMMR-explore: Submap-based multi-robot exploration system with multi-robot multi-target potential field exploration method," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 8779–8785.

[38] L. Klodt and V. Willert, "Equitable workload partitioning for multi-robot exploration through pairwise optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 2809–2816.

[39] M. Andries and F. Charpillet, "Multi-robot taboo-list exploration of unknown structured environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 5195–5201.

[40] M. Corah and N. Michael, "Volumetric objectives for multi-robot exploration of three-dimensional environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 9043–9050.

[41] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *Intl. J. Robot. Res.*, vol. 38, no. 2/3, pp. 316–337, 2019.

[42] J. W. Durham, R. Carli, P. Frasca, and F. Bullo, "Discrete partitioning and coverage control for gossiping robots," *IEEE Trans. Robot.*, vol. 28, no. 2, pp. 364–378, Apr. 2012.

[43] J. De Hoog, S. Cameron, and A. Visser, "Role-based autonomous multi-robot exploration," in *Proc. Comput. World: Future Comput., Serv. Comput., Cogn., Adaptive, Content, Patterns*, 2009, pp. 482–487.

[44] V. Spirin and S. Cameron, "Rendezvous through obstacles in multi-agent exploration," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot.*, 2014, pp. 1–6.

[45] M. Guo and M. M. Zavlanos, "Multirobot data gathering under buffer constraints and intermittent communication," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1082–1097, Aug. 2018.

[46] F. Amigoni, J. Banfi, and N. Basilico, "Multirobot exploration of communication-restricted environments: A survey," *IEEE Intell. Syst.*, vol. 32, no. 6, pp. 48–57, Nov./Dec. 2017.

[47] Y. Gao et al., "Meeting-merging-mission: A multi-robot coordinate framework for large-scale communication-limited exploration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2022, pp. 13700–13707.

[48] R. C. Arkin and J. Diaz, "Line-of-sight constrained exploration for reactive multiagent robotic teams," in *Proc. 7th Int. Workshop Adv. Motion Control. Proc.*, 2002, pp. 455–461.

[49] A. Birk et al., "Distributed communicative exploration under underwater communication constraints," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot.*, 2011, pp. 339–344.

[50] W. Sheng, Q. Yang, J. Tan, and N. Xi, "Distributed multi-robot coordination in area exploration," *Robot. Auton. Syst.*, vol. 54, no. 12, pp. 945–955, 2006.

[51] E. A. Jensen, E. Nunes, and M. Gini, "Communication-restricted exploration for robot teams," in *Proc. Workshops 28th AAAI Conf. Artif. Intell.*, 2014.

[52] E. A. Jensen, L. Lowmanstone, and M. Gini, "Communication-restricted exploration for search teams," in *Proc. Distrib. Auton. Robot. Syst.*, 2018, pp. 17–30.

[53] M. Kulkarni et al., "Autonomous teamed exploration of subterranean environments using legged and aerial robots," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2022, pp. 3306–3313.

[54] A. Agha et al., "Nebula: Quest for robotic autonomy in challenging environments; team CoSTAR at the DARPA Subterranean Challenge," 2021, *arXiv:2103.11470*.

[55] N. Hudson et al., "Heterogeneous ground and air platforms, homogeneous sensing: Team CSIRO data61's approach to the DARPA Subterranean Challenge," 2021, *arXiv:2104.09053*.

[56] T. Rouček et al., "System for multi-robotic exploration of underground environments CTU-CRAS-NORLAB in the DARPA Subterranean Challenge," 2021, *arXiv:2110.05911*.

[57] M. T. Ohradzansky et al., "Multi-agent autonomy: Advancements and challenges in subterranean exploration," 2021, *arXiv:2110.04390*.

[58] P. Petráček, V. Krátkỳ, M. Petrlík, T. Báča, R. Kratochvíl, and M. Saska, "Large-scale exploration of cave environments by unmanned aerial vehicles," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7596–7603, Oct. 2021.

[59] M. Corah, C. O'Meadhra, K. Goel, and N. Michael, "Communication-efficient planning and mapping for multi-robot exploration in large environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1715–1721, Apr. 2019.

[60] Y. Tian et al., "Search and rescue under the forest canopy using multiple UAVs," *Int. J. Robot. Res.*, vol. 39, no. 10-11, pp. 1201–1221, 2020.

[61] K. Cesare, R. Skeele, S.-H. Yoo, Y. Zhang, and G. Hollinger, "Multi-UAV exploration with limited communication and battery," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 2230–2235.

[62] 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," *Sci. Robot.*, vol. 4, no. 35, 2019, Art. no. eaaw9710.

[63] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online UAV replanning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 5332–5339.

[64] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3681–3688.

[65] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 215–222.

[66] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3529–3536, Oct. 2019.
[67] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time UAV replanning using guided gradient-based optimization and topological paths," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 1208–1214.
[68] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1992–2009, 2021.
[69] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An ESDF-free gradient-based local planner for quadrotors," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 478–485, Apr. 2021.
[70] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 489–494.
[71] J. V. D. Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Proc. Robot. Res.*, 2011, pp. 3–19.
[72] J. V. D. Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3475–3482.
[73] S. H. Arul and D. Manocha, "DCAD: Decentralized collision avoidance with dynamics constraints for agile quadrotor swarms," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1191–1198, Apr. 2020.
[74] Z. Liu et al., "MAPPER: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 11748–11754.
[75] J. Park, J. Kim, I. Jang, and H. J. Kim, "Efficient multi-agent trajectory planning with feasibility guarantee using relative Bernstein polynomial," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 434–440.
[76] X. Zhou, X. Wen, J. Zhu, H. Zhou, C. Xu, and F. Gao, "EGO-Swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments," *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 4101–4107.
[77] H. Xu et al., "Omni-Swarm: A decentralized omnidirectional visual-inertial-UWB state estimation system for aerial swarm," *IEEE Trans. Robot.*, vol. 38, no. 6, pp. 3374–3394, 2022.
[78] C. Ericson, *Real-Time Collision Detection*. Boca Raton, FL, USA: CRC Press, 2004.
[79] D. Meagher, "Geometric modeling using octree encoding," *Comput. Graph. Image Process.*, vol. 19, no. 2, pp. 129–147, 1982.
[80] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, 2010.
[81] T. Kusnur et al., "A planning framework for persistent, multi-UAV coverage with global deconfliction," in *Field and Service Robotics*. Singapore: Springer, 2021, pp. 459–474.
[82] K. Helsgaun, "An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems," *Roskilde: Roskilde Univ.*, 2017.
[83] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, "Sparse 3D topological graphs for micro-aerial vehicle planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.
[84] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 2520–2525.
[85] L. Han, F. Gao, B. Zhou, and S. Shen, "FIESTA: Fast incremental Euclidean distance fields for online motion planning of aerial robots," *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, pp. 4423–4430, 2019.
[86] R. Dubois, A. Eudes, J. Moras, and V. Frémont, "Dense decentralized multi-robot slam based on locally consistent TSDF submaps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 4862–4869.
[87] M. Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis*. Reading, MA, USA: Addison-Wesley, 1986.
[88] T. Nguyen, K. Mohta, C. J. Taylor, and V. Kumar, "Vision-based multi-MAV localization with anonymous relative measurements using coupled probabilistic data association filter," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 3349–3355.
[89] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE (3)," in *Proc. IEEE Control Decis. Conf.*, 2010, pp. 5420–5425.
[90] K. Perlin, "An image synthesizer," *ACM Siggraph Comput. Graph.*, vol. 19, no. 3, pp. 287–296, 1985.

Boyu Zhou received the B.Eng. degree in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2018, and the Ph.D. degree in electronic and computer engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2022.

He is currently an Assistant Professor with the School of Artificial Intelligence, Sun Yat-sen University, where he co-directs the Robotics, Automation, Perception and Decision (RAPID) Laboratory. His research interests include aerial robots, autonomous navigation, motion planning, and 3-D reconstruction, exploration, and swarm.

Hao Xu received the B.Sc. degree in physics from the University of Science and Technology of China, Hefei, China, in 2016, and the Ph.D. degree in electronic and computer engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2023.

His research interests include unmanned aerial vehicles, aerial swarms, state estimation, sensor fusion, localization, and mapping.

Shaojie Shen (Member, IEEE) received the B.Eng. degree in electronic engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2009, and the M.S. degree in robotics and the Ph.D. degree in electrical and systems engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2011 and 2014, respectively.

In September 2014, he joined the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, as an Assistant Professor, and was promoted to Associate Professor in July 2020. His research interests include robotics and unmanned aerial vehicles, with a focus on state estimation, sensor fusion, computer vision, localization and mapping, and autonomous navigation in complex environments.