MUI-TARE: Cooperative Multi-Agent Exploration with Unknown Initial Position

Jingtian Yan[1], Xingqiao Lin[1], Zhongqiang Ren[1], Shiqi Zhao[2], Jieqiong Yu[1], Chao Cao[1], Peng Yin[1,*], Ji Zhang[1] and Sebastian Scherer[1]

***Abstract*—Multi-agent exploration of a bounded 3D environment with the unknown initial poses of agents is a challenging problem. It requires both quickly exploring the environments and robustly merging the sub-maps built by the agents. Most existing exploration strategies directly merge two sub-maps built by different agents when a single frame observation is matched, which can lead to incorrect merging due to the false-positive detection of the overlap and is thus not robust. In the meanwhile, some recent place recognition methods use sequence matching for robust data association. However, naively applying these sequence matching methods to multi-agent exploration may require one agent to repeat a large amount of another agent's history trajectory so that a sequence of matched observation can be established, which reduces the overall exploration time efficiency. To intelligently balance the robustness of sub-map merging and exploration efficiency, we develop a new approach for lidar-based multi-agent exploration, which can direct one agent to repeat another agent's trajectory in an *adaptive* manner based on the quality indicator of the sub-map merging process. Additionally, our approach extends the recent single-agent hierarchical exploration strategy to multiple agents in a *cooperative* manner for agents whose sub-maps are merged, to improve exploration efficiency. Our experiments show that our approach is up to 50% more efficient than the baselines while merging sub-maps robustly.**

***Index Terms*—Multi-agent Exploration, Real-time Map Merging, Unknown Initial Pose**

I. Introduction

MULTI-AGENT exploration of an unknown environment is an important problem in robotics and has been investigated for decades [1]–[3]. This paper considers the exploration of a bounded 3D environment with unknown initial poses of agents, which arises in applications such as planetary exploration [4], [5], underground mining [6] and search and rescue [7], [8]. This problem is challenging as it requires (i) quickly exploring the environment and (ii) robustly detecting the overlapped sub-maps built by different agents and merging the sub-maps simultaneously. A common strategy to solve the problem is to directly merge sub-maps when the scenes explored by different agents are detected to be the same [9]. This sub-map merging procedure requires extracting features from the observation of the scenes and associating the features across the sub-maps before merging them, which is often sensitive to false-positive feature matching [10].

To address this issue, sequence matching techniques have been developed to improve recognition accuracy [11], [12]. While those methods use the features from a sequence of observations and are more robust for large complex envi-

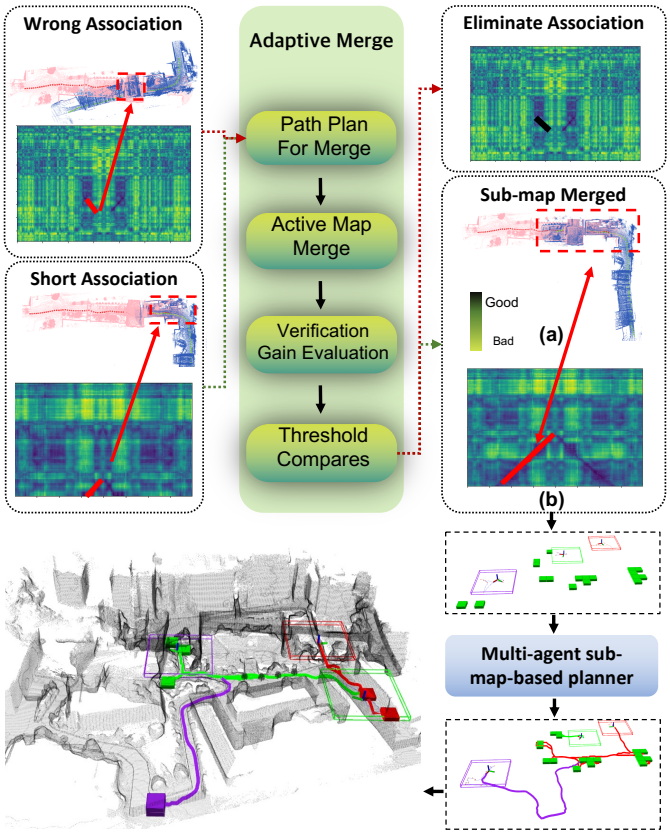

Fig. 1: Multi-agent exploration using Adaptive Merge. The multi-agent system starts exploration independently without knowing their relative poses. After a potential overlap is detected, the adaptive merge module proceeds to verify the overlap and calculate the verification gain. If the data association is incorrect, the module eliminates this association. Otherwise, the overlapped sub-maps are merged into a single sub-map. Using the information from the sub-maps, the multi-agent sub-map exploration module plans paths for exploration. (a) the point clouds merged using the estimated transform, and (b) the data association between the features of two agents (i.e. each axis represents the trajectory of one agent).

Manuscript received: January, 10, 2023; Revised April, 9, 2023; Accepted May, 12, 2023.

This paper was recommended for publication by Editor M. Ani Hsieh upon evaluation of the Associate Editor and Reviewers' comments. This research was supported by grants from NVIDIA and utilized NVIDIA SDKs (CUDA Toolkit, TensorRT, and Omniverse).

[1]J. Yan, X. Lin, Z. Ren, J. Yu, C. Chao, P. Yin, J. Zhang, and S. Scherer are with Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA (jingtiay, xingqial, zhongqir, jieqiony, ccao1, pyin2, zhangji, basti@andrew.cmu.edu).

[2]Shiqi Zhao is with University of California San Diego, La Jolla, CA 92093, USA. (e-mail:s2zhao@eng.ucsd.edu)

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

ronments, they often require two sub-maps to have a long sequence of matched observations. When applying these sequence-based methods to multi-agent exploration, one agent often has to repeat a large amount of another agent's trajectory so that a matched sequence can be detected, which often reduces the overall exploration time efficiency.

In this paper, we propose MUI-TARE (**M**ulti-agent **TARE** with **U**nknown **I**nitial position), a multi-agent lidar-based exploration system that aims to intelligently balance between sub-map merging robustness and exploration efficiency. As shown in Fig. 1, the system has two features: adaptive merging and cooperative multi-agent exploration. First, MUI-TARE uses active verification in an adaptive way, building on AutoMerge [11], a novel framework for merging large-scale maps. With the help of AutoMerge, MUI-TARE can assess the quality score of the current feature association and only increase the overlaps when needed to reach a certain threshold of feature matching quality. This enables agents to repeat another agent's viewpoints in an adaptive manner, based on the quality score, which reduces duplicated exploration and improves the overall exploration efficiency. Second, MUI-TARE employs a hierarchical cooperative multi-agent planning strategy for exploration, which builds upon the single-agent exploration planner TARE [13]. This approach first plans a coarse global path that visits the areas to be explored based on the global information of the agent's map and then plans a detailed local path to explore the current local area around the agent. Instead of naively applying this strategy to multiple agents, our approach plans for agents cooperatively when possible: it begins by planning each agent independently (before any sub-map is merged) and coordinates the agents when their sub-maps are merged.

The **contributions** of this paper are:

- MUI-TARE, a multi-agent sub-map-based planning framework for autonomous exploration of large-scale 3D environments with unknown initial relative poses.
- An adaptive merge method, including active merge distance estimation (as shown in Eq. 2), fallback policy (as shown in Sec. IV-B), and active merge (as shown in Algorithm. 1), which allows MUI-TARE to intelligently interact with AutoMerge to ensures both the exploration efficiency and the sub-map merging robustness.
- Extensive tests of the methods in different environments, where (i) our adaptive merge method leads to up to 29% higher exploration efficiency than the existing active verification strategy while maintaining the robustness in the sub-map merging process [9]; and (ii) our cooperative multi-agent planning reduces the overall exploration time for up to 52% than naively applying TARE to multiple agents without cooperation.

II. Related Work

A. Autonomous Exploration

Single-agent exploration has been extensively studied in both 2D and 3D, and the approaches include frontier-based [14]–[16], receding horizon sampling-based [17], information theoretic approaches [18], [19], traveling salesman-based [13], etc. Multi-agent exploration is more challenging than single-agent and existing work has focused on communication limitation [20], sub-map merging when initial positions of agents are unknown [21], allocating the unexplored areas to agents in a cooperative manner [1], etc.

This paper limits its focus to the problem where agents' initial poses are unknown. Existing multi-agent exploration methods are mostly limited to a 2D environment [1], [9], [21], [22], and extending them to 3D leads to many challenges such as the increased amount of sensory data, a large number of frontiers (i.e., the boundary between the explored and unexplored areas), the increased risk of feature mismatch when merging the sub-maps. We compare our MUI-TARE against two baselines based on [9], [13].

B. Sub-map merging and data association

Sub-map merging problems in 3D require integrating different sub-maps together, and the map is usually represented as 3D point clouds, occupancy grids [23], [24], or 3D meshes [25]. The point clouds based methods mainly rely on geometric-based point cloud registration to convert the point clouds into local 3D maps. Its performance is highly dependent on the robustness of 3D geometric features. This paper considers a lidar-based multi-agent system where maps are represented as point clouds. To merge sub-maps generated by each agent accurately and robustly, the SLAM community has developed many methods by leveraging distinguishable semantic objects [11], [26], bag-of-words vectors [27], learned full-image descriptors [28], and sequence-based place recognition [12], [29], to name a few. Among them, our prior work AutoMerge [11] provides a large-scale map alignment approach for multi-agent systems by leveraging a novel spherical harmonic feature extraction method for viewpoint-invariant place recognition and an adaptive sequence alignment method for accurate loop-closure detection. In most SLAM systems, the agents are passive [11], [25], i.e., the motion of the agents cannot be controlled. In multi-agent exploration, motion planning for agents is needed to achieve both high exploration efficiency and high-quality sub-map merging.

III. Problem Statement

Let $Q \subset \mathbb{R}^3$ denote a bounded space to be explored and let $Q_{trav} \subseteq Q$ denote the traversable space in Q. Let $S \subset Q$ denote all the surfaces in the Q, which represents the generalized boundary between obstacle and obstacle-free space. Let $I = \{1, 2, \dots, N\}$ denote a set of N agents with unknown initial poses in Q_{trav}. Agents are allowed to exchange information with a central station at any time. Let $u_i^t = \{p_{u_i^t}, q_{u_i^t}\}, p_{u_i^t} \in Q_{trav}$ denote a viewpoint of the sensor onboard agent $i \in I$ at time t with $p_{u_i^t}$ and $q_{u_i^t}$ denoting the position and orientation of the sensor respectively. For each viewpoint, the surface perceived is represented as $S^{cov}_{u_i^t}$. Let τ_i denote a path of agent $i \in I$, and let $dist(\tau_i)$ represent the distance traveled by the agent along the path. A sequence of viewpoints $U^i = \{u_i^0, u_i^1, \dots, u_1^{T_0}\} \subset Q_{trav}$ is visited along the path, and the path must satisfy the kinodynamic constraints of the agent. The goal of the problem is to plan a set

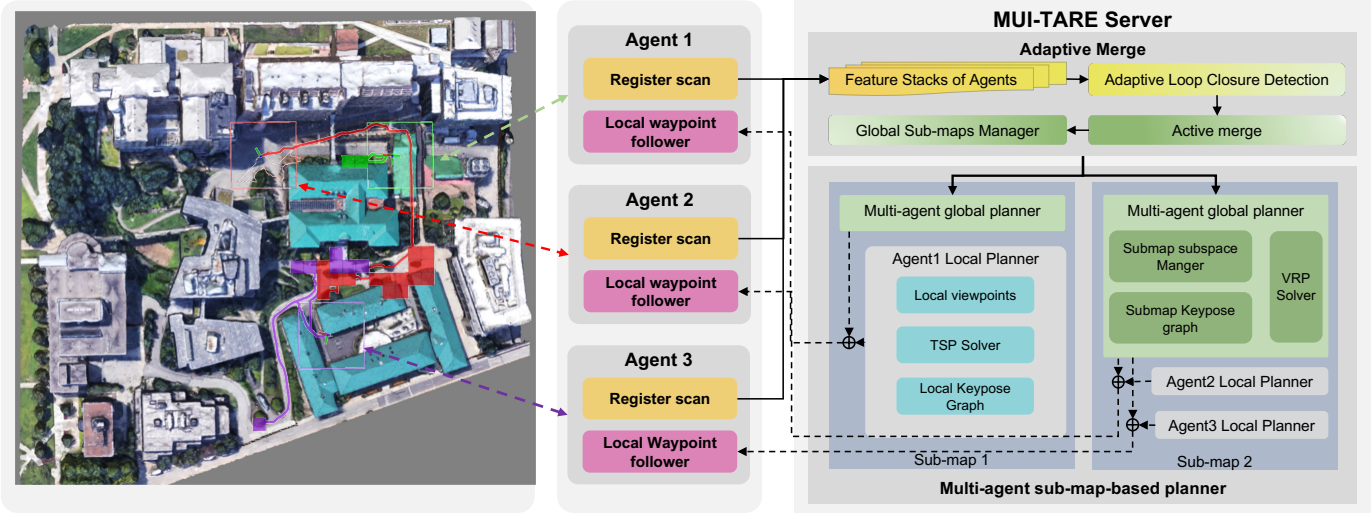

Fig. 2: The overall framework of MUI-TARE. The overall system comprises two components: the adaptive merge and multi-agent sub-map-based exploration planner. In the adaptive merge module, the inner connections of the agents are increased by actively exploring the overlapped region. The merged sub-map is provided along with the relative pose of agents on the sub-map. Given the sub-map, the MUI-TARE uses a multi-agent sub-map-based exploration planner for the exploration path planning of each agent.

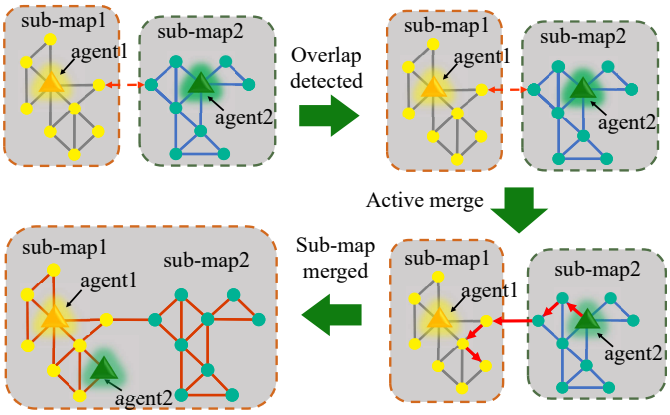

Fig. 3: Adaptive merge. During the exploration process, once a new inner connection is detected, the agent with a shorter distance to the overlap will be used for adaptive merge. This agent will verify and increase the inner connection by actively exploring the potential overlap region. This merge process will finish until the inner connection was proved to be an incorrect data association or the sub-maps got merged.

of kinodynamically feasible paths $\tau = \{\tau_1, \tau_2, \ldots, \tau_N\}$ such that the perceived surface of all agents' viewpoints along the paths covers S (i.e., $\bigcup_1^N \bigcup_0^{T_i} S_{u_i^t}^{cov} = S$) and the longest travel distance of all the paths (i.e., $\max_{i \in I} dist(\tau_i)$) is minimized.

IV. Method

This section begins with a system overview of MUI-TARE in Sec. IV-A. We then discuss the adaptive merge algorithm in Sec. IV-B. Finally, we present the hierarchical cooperative multi-agent exploration method used in MUI-TARE in Sec. IV-C.

A. System Overview

As shown in Fig. 2, the entire system consists of multiple agents and an MUI-TARE server. During the exploration, the point clouds collected by the agents are sent to the server, and the server plans and sends the paths to the agents for execution. MUI-TARE (Fig. 2) consists of two major components: an adaptive and active map merging module, and a multi-agent sub-map exploration module. The map merging module merges the *segments* (i.e., the accumulated point cloud) of agents into sub-maps and estimates the relative pose of agents based on the merged sub-maps. Here, the *sub-map* means the map that is formed by a group of agents whose relative transform is known. Given the point clouds from different agents, our approach detects the potential overlap via AutoMerge server [11]: if two series of *frames* (i.e. the point clouds at a certain pose) are matched, we say a new *inner connection* is established between agents. Then, the server stops the path planning for exploration and switches to an adaptive merge planner to plan a path for the nearby agent to verify if this inner connection is a true overlap. Once the inner connection is confirmed, the nearest agent is further navigated to increase the overlap until this new inner connection satisfies the sub-map merge condition. Finally, the overlapped sub-maps are merged into a single sub-map, and the relative poses of the agents in the merged sub-map are computed.

Based on the sub-maps computed by the map merging module, MUI-TARE runs a hierarchical path planner for each sub-map to plan paths for agents to explore the environment. Specifically, a global coarse path is first planned for agents to visit all the sub-space to be explored. In each sub-space of an agent, a detailed local path is planned to cover all the surfaces in it. MUI-TARE runs the sub-map merging and multi-agent exploration in parallel to explore the entire traversable workspace.

B. Adaptive Map Merging

During the exploration, MUI-TARE maintains a factor graph $G = \{V, E\}$ to represent the inner connections between the

Algorithm 1 Pseudo-code Adaptive overlap estimation

```
while ExplorationNotFinished() do
    MatchedSequence ← AutoMerge()
    for sequence in MatchedSequence do
        if !IsMatchValid(sequence) then
            Remove(sequence)
            continue
        inner_connection ← ConnectionEval(sequence)
        T ← TransformEval(sequence)
        if inner_connection < threshold then
            dist ← EstimateDist(inner_connection)
            ActiveMerge(T, dist)
        else
            SubmapMerge(T)
```

segments of agents. Let $V = \{v_1, ..., v_N\}$ denote the nodes of G which represent the segment obtained by each agent, and the edges $E = \{\omega_{i,j}\}$ denote the connections between segments, where $\omega_{i,j}$ is the inner connection between v_i and v_j. In AutoMerge [11], for place recognition, the inner connection of two segments is determined by the *overlap length* (i.e., the number of consecutive matched frames) and the difference of place recognition descriptors. The inner connection is more stable (higher score) with similar features and longer overlap length and is defined as follows.

$$\omega_{i,j} = \begin{cases} e^{-\frac{\|F_i - F_j\|_2^2 + C_w}{2L_{i,j}^2 + \epsilon}} & i \neq j, \\ 0 & i = j, \end{cases} \tag{1}$$

where F_i, F_j are the concatenation of the feature descriptors extracted from the overlap of v_i, v_j, $L_{i,j}$ represents the overlap length between v_i, v_j. C_w, ϵ are hyper-parameters, where C_w tunes the dependence of $\omega_{i,j}$ on the overlap length, and ϵ is a constant number to avoid zero-division.[1]

To merge sub-maps, only the segments v_i, v_j with $\omega_{i,j}$ larger than a certain threshold are merged. The relative pose between agents, represented by a transform matrix, can be determined after merging the sub-maps. However, a stable inner connection usually requires a long overlap distance which is hard to satisfy when agents are exploring independently without knowing others' poses. Thus, using a shorter sequence of matched frames for overlap detection is necessary to increase the chance that overlap is detected. However, simply using a shorter matched sequence during exploration may cause incorrect data association and large errors in the transform.

To tackle this issue, our adaptive merge module employs a proactive approach to enable the agent to verify the data association and expand the overlap length through a planned path (Fig. 3). As shown in Algorithm 1, during the exploration process, MUI-TARE periodically calls AutoMerge to match the point cloud from agents, which returns a list of matched sequence pairs called Matched Sequence. The proposed method then loops through all the matched sequence pairs in the Matched Sequence. If the sequence pair has been proven to be invalid previously, it is removed from the Matched Sequence. Otherwise, the inner connection between the two sub-maps is evaluated using Eq. 1 along with its transform matrix T. If it satisfies the merge condition, the sub-map merge process is initiated with the graph G being updated. However, if the merge condition is not fulfilled, the planner selects a Merge Agent (MA) whose distance to overlap is shorter and initiates the active merge process to increase the inner connection.

The planner first plans a path for the Merge Agent to move back to the overlapped region by doing A^* search on its past trajectory. Once the Merge Agent reaches the overlap, it will be navigated to visit the frame viewpoints (i.e. the viewpoint where the agent perceived the frame for place recognition [11]) of the overlapped agent to increase the overlap length. Denote the frame viewpoint pose of that agent can be transformed to the coordinate of Merge Agent using T. To assist the path planning of active merge, an estimation of the distance that the agent needs to travel is precomputed. For the correctly matched frames, the feature difference $\|F_i - F_j\|$ is close to zero. Thus, the predicted merge distance can be estimated by:

$$\begin{aligned} A_{dist} &= L_{i,j}^{thresh} - L_{i,j}^{t} \\ &= \sqrt{C_w}\left(-\sqrt{\frac{{L_{i,j}^{t}}^2 + \frac{\epsilon}{2}}{\|F_i - F_j\|_2^2 + C_w}} + \sqrt{\frac{L_{i,j}^{thresh\,2} + \frac{\epsilon}{2}}{\|\widehat{F}_i - \widehat{F}_j\|_2^2 + C_w}}\right) \\ &= \sqrt{C_w}\left(\sqrt{\frac{1}{2\ln \omega_{i,j}^{t}}} - \sqrt{\frac{1}{2\ln \omega_{thresh}}}\right) \end{aligned} \tag{2}$$

where the $\omega_{i,j}^t$ is the current inner connection, ω_{thresh} is the threshold of the inner connection to guide map merging, and $\hat{F}_i, \hat{F}_j$ are the expected feature descriptors when their overlap length reaches the threshold. A_{dist} is the estimated distance the agent needs to travel to establish a stable inner connection.

Based on that, the adaptive merge planner plans a path for the Merge Agent to visit A_{dist} frame viewpoints of the overlapped agent outside of the overlap. We use the greedy strategy to navigate the agent to the closest uncovered frame viewpoint, due to its fast running speed. To address the possibility of encountering incorrect data association during the adaptive merge, a failure detection mechanism is used. In order to evaluate the increase of inner connection versus the duration of the adaptive merge, we have defined the *verification gain* as:

$$G(i, j, t) = \frac{\omega_{i,j}^{t} - \omega_{i,j}^{t_0} + C_t}{\|t - t_0\| + \epsilon} \tag{3}$$

where, $\omega_{i,j}^t$ is the current value of the inner connection, $\omega_{i,j}^{t_0}$ is the value of the inner connection before the agent starts to actively increase the overlap. C_t is the hyper-parameter to control the dependence of $G(i, j, t)$ over time.[2]

During the adaptive merge process, if an incorrect data association is encountered during the active merge, the overlap area is not expected to increase through the active merge. Thus, the verification gain would decrease as the active merge duration increases. When the verification gain falls below a certain threshold, the server notifies the robot to exit the active merge and labels the inner connection as invalid. Otherwise, the agent keeps active merge until it meets the sub-map merge

[1] ϵ is set to $1e^{-4}$ in our implementation.

[2] ϵ is set to $1e^{-4}$, C_t is set to 2 in all the simulation environments.

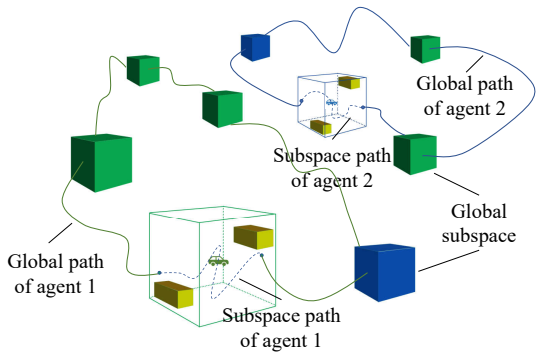

Fig. 4: Multi-agent global planning on sub-map. The global planner tries to get the paths that visit all the exploring subspaces with the minimized longest travel distance. By solving this "min-max multi-depot vehicle routing problem", a set of global coarse paths are generated for each agent to visit all the sub-space that is assigned to it. Here, the green cells represent the sub-space originally uncovered by agent 1, while the blue cells are uncovered by agent 2.

Algorithm 2 Pseudocode for hierarchical planning

for *m := 1 to M* **do**
  D' ← GetDistanceMatrix()
  $\{{}^g\tau_1^m, ...{}^g\tau_{K_m}^m\} \leftarrow GlobalPlanner(D')$
  for *k := 1 to* K_m **do**
    ${}^{local}\tau_k^m \leftarrow$ *LocalPlanner()*
    $\tau_k^m \leftarrow$ *PathConcatenate*$({}^{global}\tau_k^m, {}^{local}\tau_k^m)$
return $\{\tau_1^1, \ldots, \tau_{K_M}^M\}$

requirement. Once the reliable inner connection between sub-maps is established, the sub-maps are merged into a single sub-map as described in [11].

Remark. This module uses the existing AutoMerge paper [11] for overlap detection. However, in order to intelligently balance the robustness and efficiency during exploration we developed several techniques including the active merge distance estimation (Eq. 2), verification and fallback policy (Eq. 3), and adaptive merging algorithm (Algorithm 1).

C. Multi-agent exploration in sub-maps

MUI-TARE extends a state-of-the-art single-agent exploration method TARE to multiple agents. The transform of agents in each sub-map is given by the adaptive merge module, and the paths for agents are planned based on these sub-maps. Let $m = 1, 2, \ldots, M$ be the identifier of the sub-maps, where M is the total number of sub-maps. Initially, each sub-map contains only an agent, and $M = N$. After merging sub-maps, M decreases, and let K_m denotes the number of agents in sub-map m. In each sub-map, a similar subspaces-based representation as mentioned in TARE [13] is used. It divides the workspace Q^m into even subspaces. During exploration, the status of the subspace is set to "exploring", "explored", or "unexplored". If all the surfaces in the subspace are covered, the subspace is "explored". When the subspace contains both uncovered and covered surfaces, the subspace is "exploring". At the same time, for the subspace that is fully uncovered, the subspace is "unexplored". Once new frontiers (dividing points between known and unknown area) are detected in the "unexplored" subspace, this subspace is assigned the "exploring" label. Similarly, if all the surface in the "exploring" subspace is covered, the subspace is converted to "explored" status.

The full exploration path is planned by combining the detailed local path from the local planner and the coarse global path from the global planner. For the local planner, we use the same planner as the one in TARE. It first samples a set of viewpoints that cover the surface in the subspace. Then, a local path is planned by solving the traveling salesman problem (TSP) that visits those viewpoints with the shortest distance.

As shown in Fig. 4, to minimize the overall travel cost among the paths of all agents, we designed a new global planner. Our global planner compute a set of global paths ${}^g\tau^m = \{{}^g\tau_1^m, \ldots, {}^g\tau_{K_m}^m\}$ for K_m agents in the sub-map m to travel through all the "exploring" subspace. Since the agents travel in parallel, in order to minimize the overall travel time, the planner tries to minimize the maximum travel distance among all the agents:

$$\min(d({}^g\tau^m)) = \min_{trajectory} (\max_{agents} (d({}^g\tau_1^m), ..., d({}^g\tau_{K_m}^m))) \quad (4)$$

Where $d(\cdot)$ denotes the total length of the given global path. This problem can be formulated as a so-called min-max multi-depot vehicle routing problem (MDVRP) [30], which is known as an NP-hard problem. To solve it, we choose to use heuristic methods [30] due to their short runtime and high-quality solution in practice.

As shown in the Algorithm 2, for M sub-maps, the distance between all the exploring cells and agents in each sub-map is calculated to form the distance matrix D'. This distance is obtained by running A^* search on the roadmap acquired during the exploration process. Based on the distance matrix D', we use the aforementioned min-max MDVRP method to get K_m global coarse paths $\{{}^g\tau_1^m, ...{}^g\tau_{K_m}^m\}$ for K_m agents in the sub-map m. With the local path planned by the local planner, each agent replaces the part of the global path that falls in the subspace with its own local path to create the full exploration path. Finally, these full exploration paths $\{\tau_1^1, \ldots, \tau_{K_M}^M\}$ are sent to all the agents for execution.

Remark. MUI-TARE uses a similar representation for the workspace and local planning method as in TARE [13]. MUI-TARE uses a new cooperative global planner for multi-agent global planning as shown in the Algorithm 2.

V. Experiments

A. Experiment Setup

We test MUI-TARE in several different simulation environments. Each experiment consists of multiple agents and a central server, each on a different computer. The agents are simulated in Gazebo, and each agent is equipped with a simulated 360-degree Velodyne Lidar. The robots are assumed to be omnidirectional, and the motion planning for the robots is handled by the local planner [13]. In addition, an IMU is fused with the Lidar data for state estimation. During the experiment, the maximum speed of the robot is 3 m/s. In our implementation, the simulated environment is divided evenly into sub-spaces, where each sub-space is a 10m×10m×5m cell.

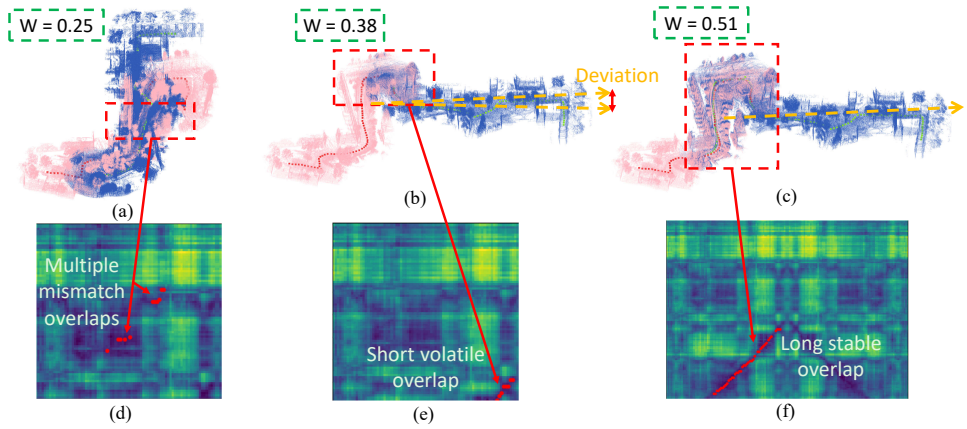

Fig. 5: Merged map of two agents and feature matching matrix. (a)(b)(c) are the result of fusing submap from two agents with varying lengths of overlap. The green and red dots represent the selected positions of the agents. (d)(e)(f) are the feature matching matrices obtained by two robots with different overlaps. In (a)(d), multiple mismatched overlaps exist in the feature matching matrix, which leads to an inability to obtain the correct transformation between two agents. Figure (b)(e) depicts that a short and unstable transformation is obtained, resulting in a deviation in the merged map. However, in (c)(f), a longer overlap is obtained with a higher score, resulting in a more accurate map.

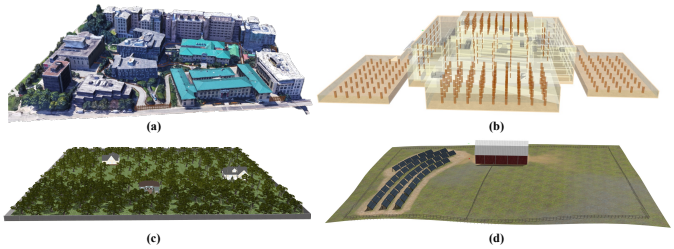

Fig. 6: Simulation environments. Visualization of the environments. (a) Campus. (b) Multi-storage Garage. (c) Forest. (d) Agriculture.

We use the ROS multi-master communication package [31] for data exchange between the agents and the server. We only transfer the point clouds collected by agents and the waypoints computed by the server, and the average bandwidth needed for an agent is 786 KB/s in our experiments, which is similar to the recent literature [9]. The global planner in MUI-TARE re-plans at 1 Hz. As shown in Fig. 6, we run tests in four simulation environments from [32].

- Campus (340m × 340m): A part of the Carnegie Mellon University campus, containing undulating terrains and convoluted environment layout.
- Multi-storage Garage (140m × 130m, 5 floors): A 3D environment with multiple floors and sloped terrains.
- Forest (150m × 150m): A environment consists mostly of trees and a couple of houses.
- Agriculture (100m × 100m): A flat, outdoor environment containing a barn, fences, and a medium-sized solar farm.

All the experiments are conducted on the server with CPU of i5-11400 and GPU of RTX-3060, and the client with CPU of i7-6700 and GPU of GTX-1060.

B. Adaptive Merge Evaluation

1) Baselines: We compare MUI-TARE against the following baseline methods.

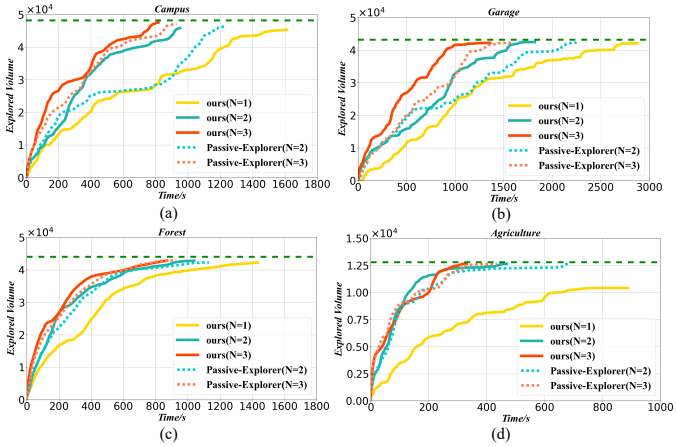

Fig. 7: Result of Test 1. Comparison of our adaptive merge with the Passive-Explorer in terms of exploration volume versus time.

- **Passive-Explorer**: This baseline modified the map merge module compared to our proposed method by directly applying the idea of [11] to our exploration planner. Specifically, the map merge module now passively receives sensor data, and the sub-map merge process occurs until a stable overlap is detected. The same exploration planner for multi-agent exploration is adopted, ensuring consistency and efficiency throughout the entire system.
- **SMMR-Explorer**: This baseline leverages the idea in [9] by using an aggressive merge strategy where the sub-maps are merged once the overlap is detected without any verification. Similarly, the same planner is used for multi-agent exploration.

2) Results and Discussion: In comparison with Passive-Explorer, Fig. 7 shows the explored volume against time while Table I shows the exploration efficiency. In all four simulation environments, our MUI-TARE outperforms the Passive-Explorer in terms of total exploration time. As shown

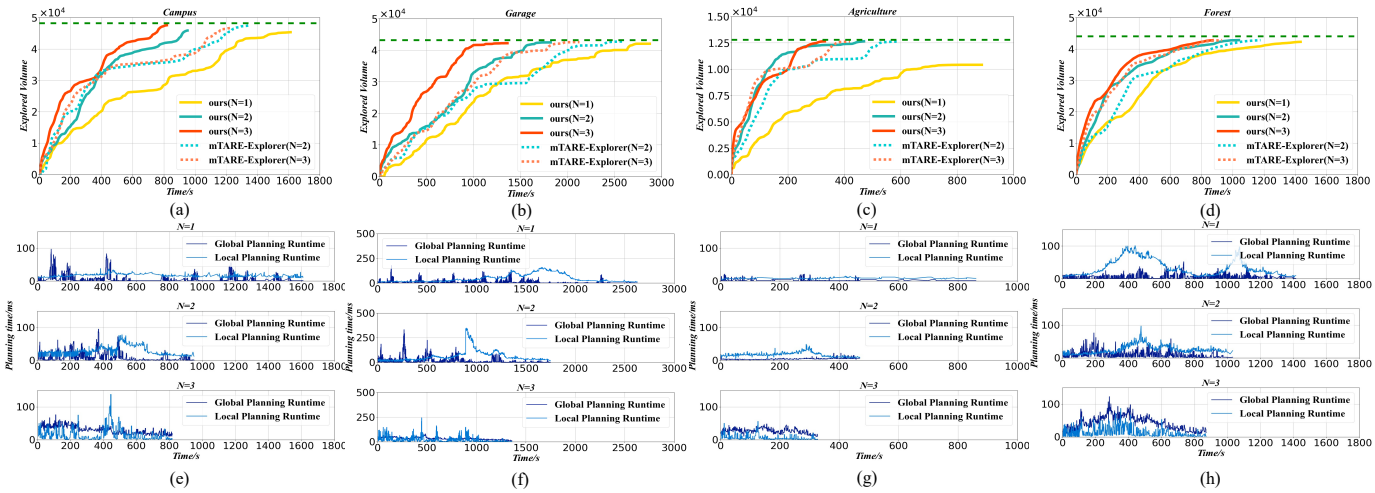

Fig. 8: Exploration efficiency & planning runtime. (**a**), (**e**) represent the exploration efficiency and planning runtime in the Campus environment. (**b**), (**f**) represent the result in the Garage environment. (**c**), (**g**) indicate the result in the Agriculture environment. (**d**), (**h**) show the result in the Forest environment. As shown in the plot, our method shows significant improvement over the mTARE, while keeping the planning runtime in a reasonable range.

TABLE I: Exploration time(s) need for Adaptive Merge compared with Passive-Explorer. We omit the results for SMMR-Explorer because it fails to complete the exploration.

Env	Passive-Explorer		SMMR-Explorer	ours	
	N=2	N=3	N=2, 3	N=2	N=3
Campus	1226.89	930.33	N/A	951.45	**818.36**
Garage	2250.88	1600.67	N/A	1665.11	**1022.92**
Forest	1132.24	928.65	N/A	1039.87	**871.89**
Agriculture	686.02	434.62	N/A	469.64	**330.83**

in Fig. 7(a), we observed that our method is 9% $\sim$ 35% faster for two agents, and 7% $\sim$ 56% faster for three agents.

We then compare our method against SMMR-Explorer which both use AutoMerge for map merging. As shown in Fig. 5(a), the aggressive strategy in SMMR-Explorer can lead to incorrect data association in our tests, which results in an incorrect map after merging. As shown in Fig. 5(b), SMMR-Explorer can also lead to large errors in the computed transform matrix. In comparison, Fig. 5(c) shows the merged map using our approach which results in robust map merging and small errors in the transform matrix. To summarize, due to the adaptive exploration strategy, our MUI-TARE achieves higher exploration efficiency than Passive-Explorer and more robust map merging than SMMR-Explorer.

C. Multi-Agent Exploration Evaluation

This section compares our MUI-TARE with a naive extension of TARE to multiple agents. Specifically, we introduce multi-agent TARE (mTARE) as a baseline, which plans for each agent independently without planning multiple agents together after merging.

1) Results: We compare our method against mTARE in all four environments with 2 and 3 agents. As shown in Fig. 8, the upper plots show the explored volume versus exploration duration while the lower plots show the runtime of the planning during the exploration. We present the exploration time and planning time in Table. II. It shows that the exploration time of our method is 13% $\sim$ 52% faster than mTARE for two agents. For three agents, our MUI-TARE is 7% $\sim$ 51% faster than mTARE. This result verifies the benefit of the multi-agent global planning strategy used in MUI-TARE. With the help of global planning in MUI-TARE, the agents can finish the exploration in a shorter time.

TABLE II: Exploration efficiency & runtime for multi-agent exploration with mTARE. The number in brackets shows the standard deviation of planning runtime.

Env	Metrics	mTARE		Ours	
		N=2	N=3	N=2	N=3
Campus	Exploration Time (s)	1350.83	1234.36	951.45	**818.36**
	Global Plan Runtime (ms)	8.18 (±10.37)	8.34 (±8.95)	11.48 (±14.99)	17.30 (±19.78)
Garage	Exploration Time (s)	2502.13	1989.09	1665.11	**1022.92**
	Global Plan Runtime (ms)	9.99 (±11.52)	9.99 (±12.15)	18.93 (±34.64)	23.19 (±31.16)
Forest	Exploration Time (s)	1178.07	932.70	1039.87	**871.89**
	Global Plan Runtime (ms)	6.39 (±6.64)	7.87 (±10.37)	9.68 (±10.35)	13.77 (±14.64)
Agriculture	Exploration Time (s)	591.02	420.03	469.64	**330.83**
	Global Plan Runtime (ms)	2.75 (±3.69)	7.49 (±7.77)	4.67 (±6.70)	6.72 (±8.27)

Additionally, Table. II shows the average local planning and global planning runtime per call in MUI-TARE and mTARE. We observed that the global planning time grows as the number of agents increases. However, the planning time is still relatively short in the 3-agents case. Finally, as shown in Fig. 8(a), by increasing the number of agents from one to two, the time required to finish the exploration (i.e., the time when the explored volume stop increasing) is reduced by roughly 36%. However, by increasing the number of agents from two to three, this reduction in exploration time is only about 16%. This indicates a diminishing return when increasing the number of agents in an exploration task.

VI. Conclusions and Future Work

This paper presents MUI-TARE, a multi-agent cooperative method for exploration with the unknown initial position. MUI-TARE intelligently balances the robustness of sub-map merging and exploration efficiency, by using an adaptive approach for map merging. Additionally, MUI-TARE extends the recent single-agent hierarchical exploration strategy to multiple agents in a cooperative manner by planning path of multiple agents together after their sub-maps are merged to further improve exploration efficiency. Our numerical results verifies the benefits of our approach.

For future work, one can decentralize MUI-TARE where no global communication is required. In addition, this work mainly focuses on the problem where the initial poses of agents are unknown. One can extend MUI-TARE to address the case where partial prior knowledge about the initial agent pose is available. In MUI-TARE, the adaptive merge is never called after the sub-maps are merged. One can develop a long-term adaptive merge mechanism that seeks to further improve the quality of the merge after the sub-maps are merged. Furthermore, this paper primarily focuses on global-level coordination, it is possible to further consider local-level coordination among agents to improve exploration efficiency. Finally, one can also consider combining MUI-TARE with [33] to ensure agent-agent collision avoidance constraints when exploring cluttered space using multiple robots.

References

[1] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[2] J. K. Verma and V. Ranga, "Multi-robot coordination analysis, taxonomy, challenges and future scope," *Journal of intelligent & robotic systems*, vol. 102, no. 1, pp. 1–36, 2021.

[3] P. Petráček, V. Krátký, M. Petrlík, T. Báča, R. Kratochvíl, and M. Saska, "Large-scale exploration of cave environments by unmanned aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7596–7603, 2021.

[4] M. J. Schuster, S. G. Brunner, K. Bussmann, S. Büttner, A. Dömel, M. Hellerer, H. Lehner, P. Lehner, O. Porges, J. Reill *et al.*, "Towards autonomous planetary exploration," *Journal of Intelligent & Robotic Systems*, vol. 93, no. 3, pp. 461–494, 2019.

[5] M. Hassanalian, D. Rice, and A. Abdelkefi, "Evolution of space drones for planetary exploration: A review," *Progress in Aerospace Sciences*, vol. 97, pp. 61–105, 2018.

[6] C. Papachristos, S. Khattak, F. Mascarich, and K. Alexis, "Autonomous navigation and mapping in underground mines using aerial robots," in *2019 IEEE Aerospace Conference*, 2019, pp. 1–8.

[7] A. Davids, "Urban search and rescue robots: from tragedy to technology," *IEEE Intelligent Systems*, vol. 17, no. 2, pp. 81–83, 2002.

[8] B. Shah and H. Choset, "Survey on urban search and rescue robots," *Journal of the Robotics Society of Japan*, vol. 22, no. 5, pp. 582–586, 2004.

[9] J. Yu, J. Tong, Y. Xu, Z. Xu, H. Dong, T. Yang, and Y. Wang, "SMMR-explore: Submap-based multi-robot exploration system with multi-robot multi-target potential field exploration method," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8779–8785.

[10] S. Yu, C. Fu, A. K. Gostar, and M. Hu, "A review on map-merging methods for typical map types in multiple-ground-robot slam solutions," *Sensors*, vol. 20, no. 23, p. 6988, 2020.

[11] P. Yin, H. Lai, S. Zhao, R. Fu, I. Cisneros, R. Ge, J. Zhang, H. Choset, and S. Scherer, "AutoMerge: A framework for map assembling and smoothing in city-scale environments," 2022. [Online]. Available: https://arxiv.org/abs/2207.06965

[12] P. Yin, F. Wang, A. Egorov, J. Hou, J. Zhang, and H. Choset, "Seq-SphereVLAD: Sequence matching enhanced orientation-invariant place recognition," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5024–5029.

[13] C. Cao, H. Zhu, H. Choset, and J. Zhang, "TARE: A hierarchical framework for efficiently exploring complex 3d environments." in *Robotics: Science and Systems*, 2021.

[14] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, 1997, pp. 146–151.

[15] A. Batinovic, T. Petrovic, A. Ivanovic, F. Petric, and S. Bogdan, "A multi-resolution frontier-based planner for autonomous 3d exploration," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4528–4535, 2021.

[16] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 413–14 423, 2020.

[17] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3d exploration," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1462–1468.

[18] F. Bourgault, A. Makarenko, S. Williams, B. Grocholsky, and H. Durrant-Whyte, "Information based adaptive robotic exploration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2002, pp. 540–545 vol.1.

[19] H. Carrillo, P. Dames, V. Kumar, and J. A. Castellanos, "Autonomous robotic exploration using a utility function based on rényi's general theory of entropy," *Autonomous Robots*, vol. 42, no. 2, pp. 235–256, 2018.

[20] F. Amigoni, J. Banfi, and N. Basilico, "Multirobot exploration of communication-restricted environments: A survey," *IEEE Intelligent Systems*, vol. 32, no. 6, pp. 48–57, 2017.

[21] A. Caccavale and M. Schwager, "Trust But Verify: A distributed algorithm for multi-robot wireframe exploration and mapping," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 3294–3301.

[22] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1339, 2006.

[23] S. Carpin, "Fast and accurate map merging for multi-robot systems," *Autonomous robots*, vol. 25, no. 3, pp. 305–316, 2008.

[24] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[25] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone, "Kimera: From slam to spatial perception with 3d dynamic scene graphs," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1510–1546, 2021.

[26] R. Dube, A. Cramariuc, D. Dugas, H. Sommer, M. Dymczyk, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: Segment-based mapping and localization using data-driven descriptors," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 339–355, 2020.

[27] D. Galvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[28] R. Arandjelovic and A. Zisserman, "All about VLAD," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2013, pp. 1578–1585.

[29] M. J. Milford and G. F. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 1643–1649.

[30] J. Carlsson, D. Ge, A. Subramaniam, A. Wu, and Y. Ye, "Solving min-max multi-depot vehicle routing problem," *Lectures on global optimization*, vol. 55, pp. 31–46, 2009.

[31] A. Tiderko, F. Hoeller, and T. Röhling, "The ros multimaster extension for simplified deployment of multi-robot systems," in *Robot operating system (ROS)*. Springer, 2016, pp. 629–650.

[32] C. Cao, H. Zhu, F. Yang, Y. Xia, H. Choset, J. Oh, and J. Zhang, "Autonomous exploration development environment and the planning algorithms," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8921–8928.

[33] Z. Ren, S. Rathinam, and H. Choset, "Conflict-Based Steiner Search for Multi-Agent Combinatorial Path Finding," in *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.